COMPUTER SCIENCE & TECHNOLOGY:

COMPUTER PERFORMANCE EVALUATION USERS GROUP

CPEUG

16th Meeting

NBS Special Publication 500-65

U.S. DEPARTMENT OF COMMERCE
National Bureau of Standards

NATIONAL BUREAU OF STANDARDS

The National Bureau of Standards[1] was established by an act of Congress on March 3, 1901. The Bureau's overall goal is to strengthen and advance the Nation's science and technology and facilitate their effective application for public benefit. To this end, the Bureau conducts research and provides: (1) a basis for the Nation's physical measurement system, (2) scientific and technological services for industry and government, (3) a technical basis for equity in trade, and (4) technical services to promote public safety. The Bureau's technical work is performed by the National Measurement Laboratory, the National Engineering Laboratory, and the Institute for Computer Sciences and Technology.

THE NATIONAL MEASUREMENT LABORATORY provides the national system of physical and chemical and materials measurement; coordinates the system with measurement systems of other nations and furnishes essential services leading to accurate and uniform physical and chemical measurement throughout the Nation's scientific community, industry, and commerce; conducts materials research leading to improved methods of measurement, standards, and data on the properties of materials needed by industry, commerce, educational institutions, and Government; provides advisory and research services to other Government agencies; develops, produces, and distributes Standard Reference Materials; and provides calibration services. The Laboratory consists of the following centers:

Absolute Physical Quantities[2] — Radiation Research — Thermodynamics and Molecular Science — Analytical Chemistry — Materials Science.

THE NATIONAL ENGINEERING LABORATORY provides technology and technical services to the public and private sectors to address national needs and to solve national problems; conducts research in engineering and applied science in support of these efforts; builds and maintains competence in the necessary disciplines required to carry out this research and technical service; develops engineering data and measurement capabilities; provides engineering measurement traceability services; develops test methods and proposes engineering standards and code changes; develops and proposes new engineering practices; and develops and improves mechanisms to transfer results of its research to the ultimate user. The Laboratory consists of the following centers:

Applied Mathematics — Electronics and Electrical Engineering[2] — Mechanical Engineering and Process Technology[2] — Building Technology — Fire Research — Consumer Product Technology — Field Methods.

THE INSTITUTE FOR COMPUTER SCIENCES AND TECHNOLOGY conducts research and provides scientific and technical services to aid Federal agencies in the selection, acquisition, application, and use of computer technology to improve effectiveness and economy in Government operations in accordance with Public Law 89-306 (40 U.S.C. 759), relevant Executive Orders, and other directives; carries out this mission by managing the Federal Information Processing Standards Program, developing Federal ADP standards guidelines, and managing Federal participation in ADP voluntary standardization activities; provides scientific and technological advisory services and assistance to Federal agencies; and provides the technical foundation for computer-related policies of the Federal Government. The Institute consists of the following centers:

Programming Science and Technology — Computer Systems Engineering.

[1]Headquarters and Laboratories at Gaithersburg, MD, unless otherwise noted; mailing address Washington, DC 20234.
[2]Some divisions within the center are located at Boulder, CO 80303.

COMPUTER SCIENCE & TECHNOLOGY:

Computer Performance Evaluation Users Group (CPEUG)

Proceedings of the Sixteenth Meeting
Held at Orlando, Florida
October 20-23, 1980

Editor:
Dr. Harold Joseph Highland

Conference Host:
Navy Data Automation Facility
Naval Training Center
Orlando, FL 32813

Sponsored by
Institute for Computer Sciences and Technology
National Bureau of Standards
Washington, DC 20234

U.S. DEPARTMENT OF COMMERCE, Philip M. Klutznick, Secretary
Luther H. Hodges, Jr., Deputy Secretary
Jordan J. Baruch, Assistant Secretary for Productivity, Technology and Innovation
NATIONAL BUREAU OF STANDARDS, Ernest Ambler, Director

Issued October 1980

Reports on Computer Science and Technology

The National Bureau of Standards has a special responsibility within the Federal Government for computer science and technology activities. The programs of the NBS Institute for Computer Sciences and Technology are designed to provide ADP standards, guidelines, and technical advisory services to improve the effectiveness of computer utilization in the Federal sector, and to perform appropriate research and development efforts as foundation for such activities and programs. This publication series will report these NBS efforts to the Federal computer community as well as to interested specialists in the academic and private sectors. Those wishing to receive notices of publications in the series should complete and return the form at the end of this publication.

National Bureau of Standards Special Publication 500-65
Nat. Bur. Stand. (U.S.), Spec. Publ. 500-65, 316 pages (Oct. 1980)
CODEN: XNBSAV

Library of Congress Catalog Card Number: 80-600155

U.S. GOVERNMENT PRINTING OFFICE
WASHINGTON: 1980

For sale by the Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402

FOREWORD

In view of the theme of CPEUG 80 -- CPE Trends in the 80's -- it seems appropriate to reflect upon changes observed and progress made during the 70's as a guide to what we may expect in the next decade.

As the 70's began, the trend of spectacular growth in the number of large-scale computer systems was ending. The computer community began looking toward more efficient use of existing systems in lieu of the seemingly endless proliferation of large-scale systems that had characterized the late 60's. Economic recession in the private sector and the impact of the Brooks Act in the Federal community were prime factors contributing to this change. As a result, increased emphasis was placed on (then) new techniques such as hardware and software monitoring that have since become standard computer performance evaluation tools. Benchmarking became a viable method to test different vendors' systems during procurement of new systems. System simulation and analytical modeling found new applications in computer performance prediction. Throughout the 70's, these techniques were refined and enhanced, new concepts were introduced, and new terminology adopted -- system tuning, installation management, remote terminal emulation, system sizing, the ADP life cycle, capacity planning all became parts of the computer performance jargon of the 70's.

The need for a forum to promote the exchange of ideas and information in this newly developing discipline was fulfilled with the foundation of CPEUG by the United States Air Force in 1971. Later, sponsorship of CPEUG shifted to the National Bureau of Standards because of its responsibilities to explore the applicability of standards to computer performance. CPEUG quickly became the major annual event for those interested in the efficient use of computers within the Federal establishment. As CPEUG grew, its audience diversified so that in recent years roughly half of CPEUG meeting attendees were employees of the Federal Government. The remainder represented companies, universities, and consulting firms who either had an interest in the Federal ADP usage or who merely wanted to keep abreast of the state-of-the-art of CPE.

Throughout the 70's, CPEUG programs addressed techniques and issues that were in the forefront of CPE activity. Thus, in the early 70's, CPEUG concentrated on topics related primarily to hardware performance. However, as the uses of computers became more sophisticated and CPE matured as a technical discipline, CPEUG programs became more diversified -- computer networks, distributed processing, data base management, computer security, and user satisfaction are topics that have been introduced to the CPEUG program within the past two years and are prominent topics for CPEUG 80. We hope that by emphasizing new CPE topics in the CPEUG 80 program, we may stimulate discussion that will help us all try to visualize the state of CPE (and CPEUG) ten years hence. We welcome all of you to CPEUG 80 and hope your experience here will be one that is remembered throughout the 80's.

My sincere thanks go to the many people whose efforts have brought you this fine conference, especially those who have dedicated their time, talent, and effort to serve on the CPEUG 80 Committee. Their names appear elsewhere in these proceedings. In addition, I would like to thank Carole Zerr for her continued excellent assistance.

Dr. James E. Weatherbee
CPEUG Conference Chairman
October 1980

Preface

Last year's Conference theme, "The Expanding Scope of CPE," focused attention upon the roles of CPE in a rapidly expanding and changing technology. This year's theme, "CPE Trends in the 80's," builds upon last year's Conference by emphasizing new roles for the CPE practitioner. Several new and interesting techniques in the traditional CPE areas are presented. In addition, some relatively new areas and their relationship to CPE are addressed. Debate on the relevancy of CPE in these new areas will likely mold the future of this Conference in the 80's. The central issue to be debated is best stated through a question posed by last year's Program Chairman: "How will CPE, traditionally associated with large central computers, change in an era of smaller and cheaper hardware and improved digital communications?"

The technical program for this year's Conference emphasizes new technology areas, procurement issues in the 80's, and new approaches in traditional CPE areas. Six relatively new areas are presented in this year's program: Data Base Management System Performance, Data Communications, Computer Network Performance, Software Improvements, Human Interfaces, and Security, Fraud and Audit. Some papers have been presented in these areas at past Conferences, but this year's program significantly expands the scope of these sessions. The interest shown in these areas will, to a large extent, mold the structure of the future Conferences.

CPEUG is continuing to provide a forum for debating the ADP procurement process within the Federal Government. The keynote address, "Toward a More Efficient ADP Procurement Process in the 80's," highlights the objectives of the program's procurement sessions, panels, and tutorials. Past interest in this phase of the program suggests that CPEUG will continue to explore improved ADP acquisition strategies.

The traditional sessions of CPEUG are again technically stimulating and stress better approaches in CPE. New measurement and prediction techniques are presented to expand the CPE practitioner's set of available tools. Actual experiences are also presented to show each of us what approaches should -- and should not -- be used.

The CPEUG 80 program has been prepared by many people. The Conference Committee, session chairpersons, authors, tutors, and referees all deserve recognition for their time and patience. The excellent administrative support of Cathy Casey and Terri Schroeder merit special recognition. All of these individuals have contributed to the success of this year's program.

James O. Mulford
CPEUG Program Chairman
October 1980

CPEUG 80

Abstract

The Proceedings record the papers that were presented at the Sixteenth Meeting of the Computer Performance Evaluation Users Group (CPEUG 80) held October 20-23, 1980, in Orlando, Florida. With the theme "CPE Trends in the 80's," CPEUG 80 focused on new applications that are expected to grow in the 80's and changes that may occur in traditional areas during the 80's. The program was divided into two parallel sessions and included technical papers on previously unpublished work, case studies, tutorials, and panels. Technical papers are presented in the Proceedings in their entirety.

Key words: Benchmarking; capacity planning; computer performance evaluation; computer performance measurement; computer performance prediction; computer system acquisition; CPE in auditing; installation management; on-line system evaluation; queuing models; simulation; workload definition.

CPEUG Advisory Board

Richard F. Dunlavey
National Bureau of Standards
Washington, DC

Dennis M. Conti
National Bureau of Standards
Washington, DC

Dennis M. Gilbert
Federal Computer Performance Evaluation
and Simulation Center
Washington, DC

Harry J. Mason, Jr.
U.S. General Accounting Office
Washington, DC

Conference Committee

CONFERENCE CHAIRMAN	Dr. James E. Weatherbee FEDSIM/MV Washington, DC
PROGRAM CHAIRMAN	James O. Mulford International Computing Company Dayton, OH
PROGRAM VICE-CHAIRMAN	Judith G. Abilock Price Waterhouse & Company Washington, DC
PROCEEDINGS EDITOR	Dr Harold J. Highland State University of New York Farmingdale, NY
PUBLICATION CHAIRMAN	Peter J. Calomeris National Bureau of Standards Washington DC
ARRANGEMENTS CHAIRMAN	Arthur F. Chantker Federal Aviation Adminstration Washington, DC
REGISTRATION CHAIRMAN	James G. Sprung The MITRE Corporation McLean, VA
FINANCE CHAIRMAN	Carol B. Wilson Fiscal Associates, Inc. Alexandria VA
AWARDS CHAIRMAN	Jeffrey M. Mohr Arthur Young & Company Washington, DC
PUBLICITY CHAIRMAN	Theodore F. Gonter U.S. General Accounting Office Washington, DC
LOCAL ARRANGEMENTS CHAIRMAN	Hadley G. Nelson Navy Data Automation Facility Orlando, FL
VENDOR COORDINATOR	Wayne D. Bennett National Bureau of Standards Washington, DC

CPEUG80

CPEUG80

Referees

Major Robert Feingold
USAF Phase IV PMO/PGY
Gunter AFS AL

Larry Ritchard
NCR Corp.
Dayton, OH

Duane Ball
FEDSIM/MV
Washington, DC

Captain Brett Berlin
FEDSIM/NA
Washington, DC

Bruce Herbert
U.S. General Accounting Office
Washington, DC

Joel Molyneaux
Engineered Systems
Omaha, NE

Arnold Johnson
GSA/ADTS/CFT
Falls Church, VA

Charles Davidson
U.S. General Accounting Office
Washington, DC

Wayne D. Bennett
National Bureau of Standards
Washington, DC

Richard F. Dunlavey
National Bureau of Standards
Washington, DC

Michael Morris
Independent Consultant
Washington, DC

Byron Griffith
AFDSDC/DM
Gunter AFS, AL

Lt. Ken Herbert
USAF Phase IV PMO/PGY
Gunter AFS, AL

Howard S. White
Lawrence Berkeley Laboratory
Berkeley, CA

Dennis M. Conti
National Bureau of Standards
Washington, DC

Mickey Sutton
AFDSDC/DM
Gunter AFS, AL

Larry Davis
McDonnell Douglas Automation Co.
St. Louis, MO

Captain Steve Cristiani
USAF/ASD/ENA
WPAFB, OH

Todd Ramsey
IBM Corp
Bethesda, MD

Peter Petrusch
USAF HQ AFLC/ACT
WPAFB, OH

William Buckles
General Research Corp.
Huntsville, AL

James O. Mulford
International Computing Co.
Dayton, OH

Major David Schafer
USAF HQ SAC/AD
Offutt AFB, NE

Geoffrey Goodman
International Computing Co.
Dayton, OH

Lt. Linda S. Mehalko
USMC SERV. CO. H&S DN ASC
Camp Pendleton, CA

Karen Gordon
University of Maryland
College Park, MD

CPEUG 80

Table of Contents

PROTOTYPING/BENCHMARKING

OPERATING SYSTEM PERFORMANCE METERS

DBMS Performance

On the Simulation Modeling of Network DBMS

Jan A. Aitken
Harry T. Hsu

Directorate of Design Assessment
Federal Computer Performance Evaluation and Simulation Center
Washington, DC 20330

This paper describes an approach to the simulation modeling of Data Base Management Systems (DBMS) - specifically, those DBMS which are based on the CODASYL DBMS concepts and specifications [1-4][1].

The modeling approach provides the system designer (or data base administrator) with a powerful tool for the prediction and evaluation of DBMS performance. Modeling language statements corresponding directly to actual data description language (DDL) and data manipulation language (DML) statements allow the straightforward definition of data base structures (schemas) and representation of application program behavior. A pre-defined model structure which includes representations of DBMS control functions (e.g., data base access control, space management, buffer management) and DML operations, facilitates DBMS modeling. The use of a simulation-time "replica" of a modeled data base enables detailed modeling of data base navigation and accessing.

The DBMS model supports the investigation of many factors affecting DBMS performance, including: workload composition, logical data base structure, physical data base organization, data base size, data base page size, number of buffers and buffer management strategies, data base loading factors, and secondary storage device characteristics.

DBMS performance prediction and evaluation is supported by the automatic collection and reporting of a variety of performance statistics pertaining to (1) data base area queuing, utilization, and accessing, (2) run-unit execution time and data base I/O, (3) buffer utilization and queuing, and (4) run-unit DML request queuing.

Key words: Data base management systems; simulation modeling; performance evaluation; data base; schema; data description language; data manipulation language; CODASYL.

1. Introduction

The use of Data Base Management Systems (DBMS) is becoming increasingly widespread in contemporary computer systems. A DBMS is typically a complex software system with substantial computer system resource requirements. DBMS performance can be a significant factor in overall computer system performance. Consequently, the prediction and analysis of DBMS performance merits a prominent place

[1] Figures in brackets indicate the literature references at the end of this paper.

within the field of Computer Performance Evaluation (CPE).

The prediction and analysis of DBMS performance is a complex issue since DBMS performance can be influenced by many diverse factors related to logical data base design, physical data base design, secondary storage device characteristics, workload composition and characteristics, and DBMS operation/tuning parameter settings. There exists a definite need for tools and techniques for DBMS performance evaluation.

The CODASYL data model [1-4] represents a popular approach to data base management. Many commercially available data base management systems are based on the CODASYL specifications.

In this paper, we present an approach to DBMS simulation modeling for the performance prediction and evaluation of CODASYL-type DBMS. The DBMS model provides the system designer (or data base administrator) with a powerful tool for DBMS performance evaluation and performance prediction. It can be used to:

(1) Predict the performance of a DBMS for a given data base design and DBMS workload

(2) Investigate DBMS performance relative to changing workload, and perform DBMS stress-test studies

(3) Evaluate alternative logical and physical data base designs to achieve a reasonable performance-efficient data base design

(4) Support DBMS performance tuning and investigate DBMS performance problems.

The modeling approach results in a DBMS model which closely simulates the operation of a real DBMS. The fundamental concepts of the model design are essentially the same as those of the CODASYL approach to data base management. Throughout this paper, it is assumed that the reader is already familiar with those concepts. For information on the CODASYL specifications and DBMS based on those specifications, the reader is referred to references 1-11.

The DBMS model has been implemented using the Extendable Computer System Simulator II (ECSS II) [17]. ECSS II, a special-purpose language for constructing discrete-event simulation models of computer systems, is a super-set of SIM^CRIPT II.5 [15, 16].

Section 2 describes the salient features of the DBMS modeling approach. Section 3 describes the DBMS and data base performance-related outputs produced by the model.

2. Description of the Modeling Approach

This section presents the salient features of the modeling approach. A detailed description of the DBMS model is provided in a separate document [18].

2.1 Overview

The structure of the DBMS model closely resembles that of an actual DBMS. A simulated Data Base Control System (DBCS) handles the user interface and manages access to the data base. A simulated data base replicates the content and structural aspects of a real data base. A Data Description Language (DDL), similar to the CODASYL DDL, is used to describe a data base. A set of Data Manipulation Language (DML) commands, encompassing all CODASYL DML operations, is used to characterize application program data base activity. By using the DML functions provided, simulated application programs can store and retrieve data base records, as well as navigate through the simulated data base.

2.2 Data Base

The simulated data base is a replica of the data base being modeled. It is virtually identical to the real data base in both content and structure, except for the fact that it contains no real data.

An area (or realm) is a subdivision of a logical data base. For each area in the real data base, there is a corresponding area in the simulated data base. An area in the simulated data base has the same number of pages as the real area it represents. Thus, for each page in the real data base being modeled, there is a corresponding page in the simulated data base. A page in the simulated data base is used to store simulated record occurrences. The structure of a simulated data base page resembles that of a real data base page.

Record types are mapped to areas. All occurrences of a record type can be stored in only one area, however more than one record type can be mapped to the same area. Record occurrence placement within an area is determined by record type location mode. The location mode specifies how record type occurrences are to be stored within an area, as well as how they are to be retrieved.

A stored record occurrence in an actual data base consists of two parts: PREFIX and DATA. The PREFIX contains all the pointers in the record, representing the relationships between the record and other stored records. The DATA is the useful data which is supplied by the user. For records in the simulated data base, only the PREFIX is stored. Each stored record contains the same number of pointers as the real data base record it represents. The data portion is not stored, and is represented only by an attribute called "data length".

All elements of a set occurrence (including owner and members) are linked together using pointers. The same number of pointers for each set occurrence are stored in the simulated data base as in the actual. For example, if a set member record has a "prior" pointer, then the corresponding simulated member record will also have a "prior" pointer.

The size of a simulated page should in general be much smaller than that of a real data base page because the simulated page contains no real data, only pointers. If the ratio of data length to prefix length is very large, then the ratio of real page size to simulated page size will also be very large. The simulated page size is chosen so that a simulated page can accomodate the same number of records as a real data base page.

The physical data base consists of one or more files. A file is an addressable segment of secondary storage known to the computer operating system. Each data base area is mapped to a file. A simulated data base can contain any number of areas and files. A file in the simulated data base can contain a single area, a portion of an area, or several areas. For each page in the simulated data base, there is a corresponding block in a file.

Each simulated file is mapped to a simulated direct-access secondary storage device. A secondary storage device is characterized in terms of its capacity, transfer rate, and access time.

The simulation of a data base page access involves the following steps:

(1) Determine the data base area which contains the desired page

(2) Determine the file and the relative block within the file containing the desired page

(3) Determine the physical storage device address (device, cylinder) of the block

(4) Simulate a seek operation to position the device read/write mechanism at the required cylinder

(5) Simulate a data transfer operation to read/write the block.

A feature of the data base modeling capabilities is the automatic generation in zero simulated time of a simulated data base. Using this feature, a simulated data base is populated with records according to the information supplied in the model's data base schema (i.e., the identification of record types and, for each record type, a count of the number of record occurrences to be initially stored). The resulting data base will have the specified number of occurrences of each record type. Furthermore, set relationships will be established for the stored records. Since a simulated data base is generated in zero simulated time, this is the most efficient way to produce a simulated data base. This can be used to advantage since, generally, the data base being studied will not be initially empty. In addition, once a data base is generated in this fashion, it can be saved, and copies of it used in future simulation runs, thus avoiding the cost of re-creating the same simulated data base.

2.3 Data Description Language

The model provides a DDL to describe a complete data base schema. No sub-schema description facilities are currently provided. A schema definition in the model is almost identical to a real schema definition. It provides for the description of files, areas, record types, and set types. The principal differences are:

a. In a record type description, individual data items are not described. Data items are treated as a unit and represented for simulation modeling purposes only by a total data length.

b. In the location mode specification for a record type, the DUPLICATES NOT ALLOWED option, specifying whether duplicates are permitted for a defined key, is not supported. It is assumed that duplicates are always allowed.

c. In a set description, the ORDER IS SORTED option is not supported.

Figure 1 presents an example of the model's DDL.

```
DATABASE  MUSIC.DB
   PAGESIZE = 1024 BYTES
   LOADING FACTOR = 0.5
      .
      .
      .
DBFILE  MUSIC.FILE
      .
      .
      .
AREA  MUSIC.AREA
   ID = 10
   PAGES = 100
   DBFILE = MUSIC.FILE
   DATABASE.ADDRESS = 1
      .
      .
      .
RECORDTYPE  PERIOD
   ID = 801
   DATASIZE = 38
   AREA = MUSIC.AREA
   LOCATIONMODE = CALC
      .
      .
      .
RECORDTYPE  COMPOSER
   ID = 803
   DATASIZE = 40
   AREA = MUSIC.AREA
   LOCATIONMODE = CALC
      .
      .
      .
SET  STYLE
   ID = 701
   OWNER.RECORDTYPE = PERIOD
   ORDER = NEXT
   NEXT.DBKEY.POSITION = 1
   PRIOR.DBKEY.POSITION = 2

SET.MEMBER
   RECORDTYPE = COMPOSER
   SET = STYLE
   NEXT.DBKEY.POSITION = 1
   PRIOR.DBKEY.POSITION = 2
   OWNER.DBKEY.POSITION = 3
   MEMBERSHIP = MANDATORY.AUTOMATIC
      .
      .
      .
```

Figure 1. Example of Model's DDL

2.4 Data Manipulation Language

The modeling approach provides a set of Data Manipulation Language (DML) operations for use in characterizing run-unit data base accessing. The DML operations provided encompass most of the CODASYL-specified DML functions, and include the following:

(1) OPEN.AREA
(2) CLOSE.AREAS
(3) STORE.RECORD for location modes DIRECT, CALC, and via a set (VIASET)
(4) FIND.RECORD
    (a) direct
    (b) CALC
    (c) owner, first, last, next, prior of set
    (d) first, last, next, prior of area
(5) MODIFY.RECORD
(6) DELETE.RECORD
(7) INSERT.RECORD
(8) REMOVE.RECORD

The approach to simulating each of the above DML operations is described below.

The CODASYL specifications use the concept of "currency" to define implied operands for most DML commands. Thus, each run-unit has a current record of the run-unit and a current record of each set type, area, and record type accessed by the run-unit. A real CODASYL DBMS automatically updates appropriate currency indicators whenever a record is accessed. The DBMS model also automatically maintains currency indicators in a similar manner.

2.4.1 OPEN.AREA

The OPEN.AREA DML initializes an area for access by a run-unit in one of six usage modes: retrieval, update, protected retrieval, protected update, exclusive retrieval, and exclusive update. The simulated DBCS' Area Manager (reference Section 2.5.1) is invoked to process the OPEN.AREA request. Depending on the current mode of usage of the area and the requested usage mode, the OPEN.AREA request will either be granted or queued. If the request is queued, further execution of the run-unit is not permitted until such time as the request is granted.

2.4.2 CLOSE.AREAS

The CLOSE.AREAS DML terminates a run-unit's access to all areas it has opened. The Area Manager is invoked to process any queued area usage requests which may be granted due to the release of the run-unit's areas.

2.4.3 STORE.RECORD

Record type occurrences are stored in the simulated data base according to the location mode (DIRECT, CALC, or VIASET) specified in the schema definition for the record type.

If the location mode is DIRECT, then the run-unit must supply a data base key when issuing the STORE.RECORD command. The data base key specifies the data base page on which the record is to be stored. The simulated DBCS will try to store the record in the specified page. If insufficient space

is available in the specified page, the simulated DBCS will store the record in a nearby page determined by the DBCS's Space Manager. After a record is stored in this mode, its correct data base key is returned to the run-unit.

For a location mode of CALC, the real DBCS uses the hashing value of a record's data item or data items to determine the data base page on which to store the record. In the DBMS model, since no actual data is stored, a random number is generated for the address of the page on which to store the record. If the distribution of the hashed data item values is close to random, the two approaches should have roughly identical distributions of record occurrences over a specified page range. If there is insufficient space in the target page for the record, then the record is stored in an overflow page. The simulated DBCS' Space Manager (reference Section 2.5.3) is invoked to determine the overflow page. Record storage statistics are automatically collected for CALC records. If a CALC record is stored in its target page, it is considered as a "CALC-fit"; otherwise, it is considered to be a "CALC-overflow". After a CALC record is stored in a page, it is inserted into the CALC set for the page. In the real data base, the position of the new record within the CALC set is determined by the value of the CALC key data item, as well as by the specification in the schema as to whether duplicates are first or last. In the simulation model, a random number is generated to determine where within the CALC set the new record should be inserted.

If the location mode is VIASET, a record is stored by the simulated DBCS in the same way as by the real DBCS. The simulated DBCS uses the current of the specified set to determine where to store the new record. If the current of the set is a member of the set, then the new record is stored into the page containing the current of set (or as close as possible). If the current of the set is the owner of the set, the new record is stored into the same page as the owner (or as close as possible) if the new record is to be stored in the same area as the owner. If the new record is to be stored in a different area than that of the owner, the new record is stored at a distance into its area proportional to the distance of the owner record into its area. Record storage statistics are automatically collected for VIASET records. If a VIASET record is stored in its target page, it is considered as a "VIASET-fit"; otherwise it is considered to be a "VIASET-overflow".

2.4.4 FIND.RECORD

The location mode of a record type, in addition to determining how a record occurrence is to be stored, also determines how a record may be found. Most forms of the FIND DML described in the CODASYL specifications are supported in the model.

If a record is stored using location mode DIRECT, then it can be found (directly) only by using the direct access form of the FIND.RECORD DML command, in which the record's data base key, specifying on which page the record is stored, is supplied.

If a record is stored using a location mode of CALC, then it can be found using the CALC key access form of the FIND.RECORD DML command. A random number is generated for the record's page number (analogous to hashing a record's CALC key to yield a page number). The set of random numbers so generated is constrained to correspond to page numbers on which occurrences of the record type have already been stored. If there is more than one occurrence of the specified record type in a CALC set, a random number is used to decide which occurrence should be selected and thus how much of the CALC set need be searched.

If the location mode of a record type is VIASET, then occurrences of the record type can only be found through set navigation. Since set relationships are maintained among records in the simulated data base just as they are among records in a real data base, set navigation can be simulated just as it would be done in the real data base. The set access forms of the FIND.RECORD DML command (find owner, find first, find last, find next, find prior) are used to perform the set navigation.

The simulated data base structure supports the simulation of area navigation. The simulated data base and the real CODASYL data base are essentially identical in structure. They have the same number of areas, and each area in the simulated data base has the same number of pages as the corresponding area in the real data base. The area access forms of the FIND.RECORD DML commands (find first, find last, find next, find prior) are used to perform area navigation.

2.4.5 MODIFY.RECORD

A record in the simulated data base is modified using the MODIFY.RECORD DML command. Two types of record modification are supported; one involves modifying the CALC key data

item, the other involves modifying other data items. If the CALC key data item is not modified, then the buffer page containing the record is marked as "modified". The simulated DBCS' Buffer Manager (reference Section 2.5.2) will, just like the real DBCS, cause the modified page to be re-written to secondary storage when the buffer is to be reused. If the CALC key data item is modified, then, in addition to marking the buffer page containing the record as "modified", the simulated DBCS also simulates the repositioning of the record within the CALC set.

2.4.6 DELETE.RECORD

A record occurrence is logically deleted from the simulated data base using the DELETE.RECORD DML command. The deletion of a record might result in the logical deletion of many record occurrences in the data base. Since a simulated data base replicates the structural aspects of a real data base, the simulated DBCS performs the DELETE.RECORD DML operation in the same way as the real DBCS.

2.4.7 INSERT.RECORD

A record occurrence is inserted as a member of a set using the INSERT.RECORD DML command.

2.4.8 REMOVE.RECORD

A record occurrence is removed from a set using the REMOVE.RECORD DML command.

2.5 DBCS Operations

The Data Base Control System (DBCS) is the set of DBMS functions for the management of data base access. The DBCS functions having a significant impact on DBMS performance are assumed to be: area management, buffer management, space management, data base access, journaling, and data dictionary access. The modeling approach provides for the representation of each identified DBCS function.

2.5.1 Area Management

The DBCS Area Manager is responsible for coordinating the use of the areas of a data base. An area may be used concurrently by multiple run-units, but certain usage modes restrict or even preclude concurrent usage. The DBCS Area Manager processes a run-unit's area usage request, and, depending on the current usage of the requested area, either grants the request or causes the delay of further execution of the run-unit until the requested usage can be permitted. Area queuing and utilization statistics are automatically collected and reported.

2.5.2 Buffer Management

The modeled DBCS includes a simulated buffer pool and an associated buffer management strategy. The number of buffers in the buffer pool is a DBMS model parameter. Each buffer can contain one data base page. The DBCS Buffer Manager manages the buffers in the buffer pool using a Least Recently Used (LRU) technique. If all buffers are full and the DBCS needs to read a new page, the buffer with the page least recently accessed is reused. If the buffer to be reused currently contains a modified page, that modified page is re-written to the simulated data base and a page I/O is simulated prior to reusing the buffer. An alternative buffer management strategy may be used simply by replacing the model's Buffer Manager module. Buffer queuing and page I/O statistics are automatically collected and reported.

2.5.3 Space Management

Data base space management, in particular overflow handling, is simulated by the DBCS Space Manager. When the simulated DBCS attempts to store a record occurrence in a page and finds that there is insufficient space available in that page, the Space Manager is invoked to identify an overflow page to be used to store the record. The default space management technique provided in the model uses special space inventory pages distributed at intervals throughout the data base to keep track of data base free space. An alternative space management technique can be modeled by simply replacing the model's Space Manager module.

2.5.4 Journaling

Journaling involves creating a journal file, i.e., a file containing before-images and after-images of all data base pages which have been modified. The simulation of journaling activity is supported by the modeled DBCS. Journaling is enabled via the specification of a DBMS model parameter. If journaling is to be simulated, a simulated file is designated as the journal file. Each time a data base page is modified, two journal file I/O operations are simulated, one to represent writing a before-modification image of the page and one to represent writing an after-modification image of the page.

2.5.5 Data Dictionary Access

DBMS operation involves a certain amount of overhead due to Data Dictionary accessing

(e.g., to access schema, sub-schema, and Device Media Control Language (DMCL) definitions). A future enhancement of the DBMS model will be to provide for the simulation of DBMS data dictionary accessing.

2.6 Run-Unit Application Programs

A run-unit application program is represented procedurally. The procedure consists primarily of a sequence of the DML statements describe above. Since the DML operations available in the model correspond in a straight-forward, almost one-to-one, fashion with those of the CODASYL specifications, the modeling of an application program's data base activity can be relatively easily accomplished.

The use of the model's DML statements is demonstrated in the example application program segment shown in Figure 2.

```
OPEN.AREA  MUSIC.AREA  USAGE = EXCLUSIVE.UPDATE

FOR I = 1 TO N.
DO
   STORE.RECORD  RECORDTYPE = PERIOD
LOOP

FOR I = 1 TO M.
DO
   FIND.RECORD  RECORDTYPE = PERIOD
   STORE.RECORD  RECORDTYPE = COMPOSER
LOOP

FIND.RECORD  RECORDTYPE = PERIOD
FIND.RECORD  SET = STYLE  RECORD = FIRST
FIND.RECORD  SET = STYLE  RECORD = NEXT
FIND.RECORD  SET = STYLE  RECORD = OWNER

CLOSE.AREAS
```

Figure 2. Example of Model's DML

3. DBMS Performance-Related Model Outputs

The DBMS model automatically collects and reports a variety of DBMS and data base related performance statistics, which are summarized below.

For each area of a modeled data base:

(a) queuing and utilization
(b) # pages read and written
(c) page access time

For each simulated run-unit:

(a) total execution time
(b) # pages requested
(c) # pages read and written
(d) CALC records stored (CALC-fit, CALC-overflow)
(e) VIASET records stored (VIASET-fit, VIASET-overflow)

For each physical file:

(a) # reads and writes
(b) # seeks and zero-time seeks
(c) seek time
(d) data transfer time
(e) total access time

For the DBMS Buffers:

(a) queuing and utilization
(b) # pages requested
(c) # pages read
(d) # modified pages written

For the DBMS:

(a) run-unit DML request queuing
(b) run-unit concurrency level

4. Concluding Remarks

The DBMS model has been found to be easy to use. Since the DDL and DML functions available in the model correspond in a straight-forward, almost one-to-one, fashion with those of the CODASYL specifications, the modeling of a particular data base and associated application programs can be easily accomplished. Furthermore, since the model includes pre-defined representations of the basic DBMS operations (e.g., data base space management, area allocation management, buffer management, etc.), no modeler effort is required in these areas as long as the default algorithms are accurate for the particular CODASYL-based DBMS being modeled.

Validation of the DBMS model is in progress. The results of the validation will be reported in a future document.

References

[1] CODASYL Data Base Task Group, April 1971 Report, ACM, New York, NY.

[2] CODASYL Data Description Language Committee, Data Description Language, Journal of Development, Document C 13.6/2:13, U.S. Government Printing Office, Washington, DC.

[3] CODASYL Programming Language Committee, CODASYL COBOL Journal of Development, Department of Supply and Services, Government of Canada, Technical Service Branch, Ottawa, Canada.

[4] CODASYL Data Base Language Task Group, CODASYL COBOL Data Base Facility Proposal, 1973, Department of Supply and

Technical Services, Government of Canada, Ottawa, Canada.

[5] Tsichritzie, D. C. and Lochovsky, F. H., Data Base Management Systems, Academic Press, New York, NY, 1977.

[6] Ullman, J. D., Principles of Data Base Systems, Computer Science Press, Potomac, MD, 1980.

[7] Date, C. J., An Introduction to Data Base Systems, Addison-Wesley, Reading, MA, 1975.

[8] Taylor, R. W. and Frank, R. L., CODASYL Data Base Systems, ACM Computing Surveys, Vol. 8, No. 1, March 1976, pp. 67-103.

[9] Cullinane Corp., IDMS Concepts and Facilities, IDMS Data Base Design and Definition Guide, IDMS Programmer's Reference Guide, IDMS Utilities, Wellesley, MA, 1979.

[10] Digital Equipment Corp., DBMS-11 Data Base Administrator's Guide, DBMS-11 COBOL Data Manipulation Language Reference Manual, Maynard, MA, 1977.

[11] Digital Equipment Corp., DBMS-10 Data Base Administrator's Guide, DBMS-10 Programmer's Manual - COBOL, Maynard, MA.

[12] Bachman, C. W., "Data Structure Diagrams", Data Base 1, 2, Summer 1969.

[13] Bachman, C. W., "The Evolution of Storage Structures", CACM 15, No. 7, July 1972, pp. 628-634.

[14] Bachman, C. W., "The Programmer as Navigator", CACM 16, No. 11, November 1973, pp. 663-6581.

[15] Kiviat, P. J., Villaneuva, R., and Markowitz, H. M., SIMSCRIPT II.5 Programming Language, CACI, Inc., Los Angeles, CA, 1976.

[16] CACI, SIMSCRIPT II.5 Reference Handbook, CACI, Inc., Los Angeles, CA, 1976.

[17] Kosy, D. W., The ECSS II Language for Simulating Computer Systems, Rand Report R-1895-GSA, December 1975.

[18] Aitken, J. A. and Hsu, H. T., "An ECSS-Based CODASYL DBMS Simulation Model", (In Preparation).

An Approach to Benchmarking DBMS

BARBARA N. ANDERSON*

Satellite Business Systems
8003 Westpark Drive
McLean, Virginia 22102

This paper presents a data base management system (DBMS) benchmarking methodology developed by FEDSIM to support an acquisition of both a computer mainframe and a DBMS. The DBMS benchmarking methodology provides procedures for:

1. defining a logical data base structure,
2. constructing programs to generate files incorporating the logical relationships, and
3. constructing batch and on-line benchmark programs which incorporate the logical DBMS relationships through the use of vendor-independent interface functions.

In order to execute the benchmark programs, the vendor must implement the logical data base structure in the proposed physical data base structure, by replacing the interface functions with appropriate DBMS commands. The vendor may implement any physical data base structure, as long as the functional requirements specified by the benchmark programs are met.

The benchmark data base and programs were implemented and tested by an agency team of programmer/analysts, with the agency team performing vendor functions. This test of the DBMS benchmarking methodology is described in this paper. Potential problems and benefits of this approach to benchmarking data base management systems are also discussed. Finally, a comparison of the similarities and differences between the U.S. Department of Agriculture's DBMS benchmark methodology and the benchmark methodology presented in this paper is presented.

* The author developed this benchmark methodology with Mr. Kenneth C. Rieck and Mr. Charles A. Self while employed at the Federal Computer Performance Evaluation and Simulation Center (FEDSIM). Mr. Rieck currently works for FEDSIM. Mr. Self works for the Office of Data Management and Telecommunications, Veterans Administration.

1. Introduction

The selection of a Data Base Management System (DBMS) to support corporate data base functions is commonly a "hit or miss" decision based on a review of literature from the vendors and discussions with salesmen. This method of selection may or may not provide the organization with the DBMS that has the capability to provide the required support.

This paper discusses an approach to benchmarking Data Base Management Systems (DMBS) which was developed by the Federal Computer Performance Evaluation and Simulation Center (FEDSIM). This DBMS benchmarking methodology was developed to support an acquisition of both a computer mainframe and a DBMS by a Federal agency. The agency currently relies on a DBMS for a large portion of its information processing. Each computer vendor must not only bid the required hardware, but must also propose a DBMS to meet the agency's data management requirements. The acquisition follows federal procurement regulations for competitive procurements, which specify that no vendor is to be given undue advantage over another.

The data base portion of the benchmark includes both batch and on-line processing. The proposed DBMS must be able to support concurrent access and update by multiple users.

2. Benchmark Data Base

The benchmark documentation provided to the vendor includes a definition of the logical structure of the data base. The logical structure is the basis for the functional description of the data base portion of the benchmark. The vendor may implement any physical data base structure as long as the functional requirements specified by the benchmark programs are met.

2.1 Logical Structure of the Data Base

The logical structure of the data base was determined through a workload study of the procuring agency's current data base processing. This workload study consisted of processing accounting and terminal log tapes to determine the frequency of execution of application modules which process against the data base. The most frequently executed on-line and batch data base application modules were examined as to their access paths through the data base and their mode of access, i.e., read, update, or add. The resulting logical structure is a model of a current agency data base. The model contains all variations of the file relationships found in the application data base.

The logical structure consists of eight files, named A through H, with the relationships shown in Figure 1. Each file consists of a group of related logical records.

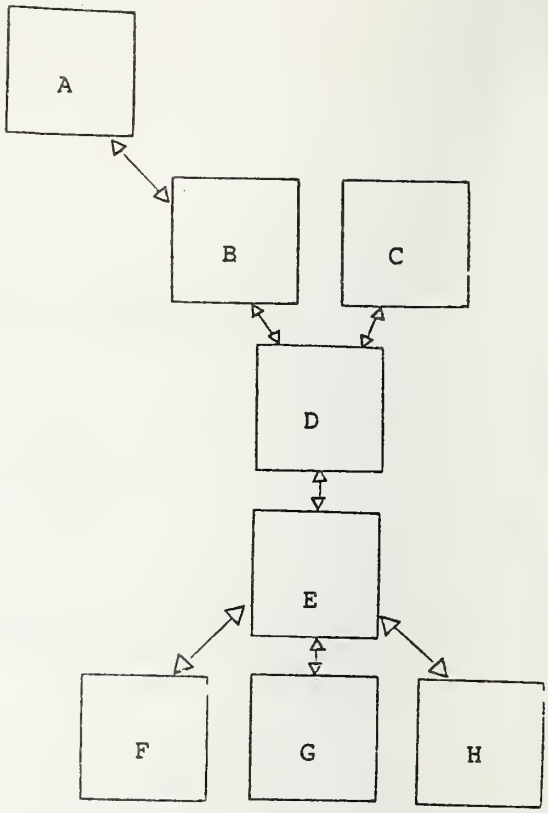


Figure 1. Data Base Logical Structure

(Throughout this paper, the terms set, owner, member, and occurrence are defined as follows. A set is defined as a two-level tree of files in which there is one owner file and one or more member files that belong to the owner file. Any file in a particular set may be either an owner or a member file in another set. A set occurrence is a particular related group of records in a set based on the owner record.)

In the agency data base, files A, B, C, and E are key-retrieval type files. A particular record from one of these files can be retrieved directly, based on the provided key value. The other files must be accessed through their relationships with the key-retrieval type files. Files B and E can be accessed either directly, based on a key value, or through their relationships with their owner files. Figure 2 contains the data base characteristics.

FILE-NAME	NUMBER OF DATA FIELDS	NUMBER OF CHARACTERS PER RECORD	NUMBER OF RECORDS PER FILE	RETRIEVAL TYPE
A	6	100	6,100	KEY
B	8	100	7,500	KEY & RELATIONSHIP
C	18	379	32,000	KEY
D	9	37	32,000	RELATIONSHIP
E	18	268	95,000	KEY & RELATIONSHIP
F	6	85	620,000	RELATIONSHIP
G	12	120	400,000	RELATIONSHIP
H	12	66	1,200,000	RELATIONSHIP

Figure 2. Data Base File Characteristics

All files are generated by programs which automatically incorporate the appropriate logical relationships. The first 24 characters of each record contain the information necessary to determine file and record relationships. The first field contains the file name to which the record belongs. The second field is the unique key number designating the position of the record in the file. All records contain this field regardless of the type of file access. The "top node files," A and C, contain valid information in the first and second fields only. Records for the rest of the files contain information about their specific owner record within a set occurrence. The third field contains the file name of the owner record, while the fourth field is the record key of the particular owner of the set occurrence to which this particular record belongs. File D is the only file that has two owners: the information for the second owner is contained in the fifth and sixth fields. Figure 3 shows the schematic of the 24-character information data fields at the beginning of each record.

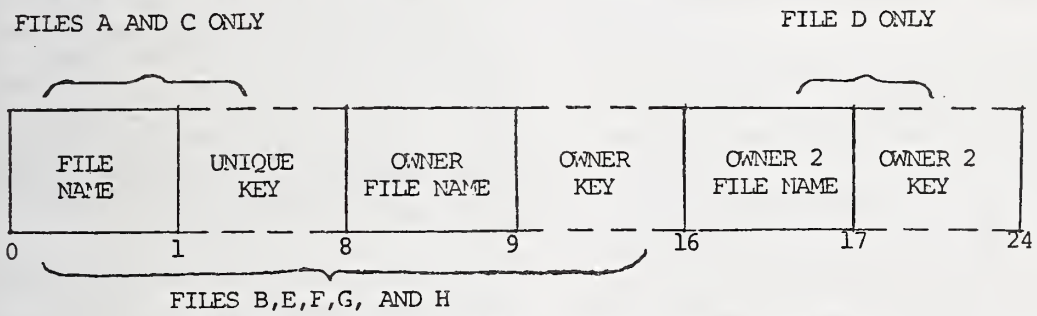


Figure 3. Logical Relationship Information Fields

Example

As an example of the use of the logical relationship information fields, suppose that a record contains the following values in the 24 leading characters:

D0014495B0006995C0014495 ... ...

This would indicate that the record is record number 14,495 of file D. The record belongs to a set in which file B is the owner file. The record belongs to the set occurrence of that set in which record 6,995 of file B is the owner record. This record also belongs to a set with owner file C. Record 14,495 of file C is the owner record of the second set occurrence.

2.2 Data Base Generation

The size of the DBMS benchmark data base is large enough to fill several disk packs, in order to represent more nearly accurately the actual workload. The data base contains approximately 234 million characters. Instead of providing several reels of magnetic tape to the vendors

(along with the attendant problems of read checks, etc.), the vendors are provided with eight file-generation programs; one for each of the eight files. These programs, which are executed at the vendor's site, generate the contents of the data base, including all necessary information concerning logical relationships among records and files. The data fields of each record contain randomly-generated alphabetic and numeric data.

2.3 Physical Structure of the Data Base

The vendor subsequently loads the generated files into the physical data base structure using the proposed DBMS. The logical relationship information fields are data fields, and must be loaded along with the other data fields. The DBMS benchmark programs must be able to manipulate the logical relationship information fields in the same manner as the other data fields. All accesses to the data base are performed on a full record basis.

The logical relationship information provided in the generated records of each file is not intended to specify a physical file structure. The vendor may load the data base in any manner, provided the functional requirements specified by the benchmark programs are met and the benchmark can be run within the allowed time frame.

3. DBMS Interface Functions

A description of the required data base functions within the benchmark programs, such as "adds," "reads," and "replaces," are provided through a vendor-independent DBMS language.

3.1 Purpose and General Description

The DBMS benchmark programs were written in ANSI COBOL and do not contain any architecture-dependent code, except for statements such as SELECT statements, etc. Vendor-independent DBMS functions were developed and placed in the programs to represent DBMS access. The interface functions were inserted into the code in the form of COBOL comment statements. The vendor must replace the interface functions with DBMS commands that perform the required functions.

3.2 Implementation

Access to key-retrieval type files is requested on the basis of a key value, which is provided as a parameter in the interface function. When access to a record based on its relationships with other records and files is required, then the desired relationship path is indicated (owner or member record) and the specific set occurrence is also indicated by using the concept of "the current record of a file." The current record is defined as the last record accessed for that file. A current record must be separately maintained for all files accessed by each particular program. Figure 4 is an example of the provided DBMS interface function for a read which depends on a relationship path. Figure 5 shows the translation of the DBMS interface function into IDS-II commands to perform the specified function. The definition of the interface function and "end-flag" processing requirements is described in the next section.

Communication of the data content of records is provided through an area in the benchmark programs called the "user-working-area." The vendor's code must deliver the record content to this area when a record is being retrieved from the data base, and accept data from this area when a record is being sent to the data base.

3.3 Interface Function Definitions

Variations of three basic DBMS interface functions, DB-READ, DB-REPLACE, and

```
081200  5800-FILE-F-DB-READ-FORMAT-II.

081300*************************************************************
081400*
081500*        DB-READ RECORD OF FILE (F) WHOSE OWNER FILE IS (E),
081600*            RETURN END-FLAG (END-FLAG-F)
081700*
081800*************************************************************
081900  5890-EXIT-FILE-F-DB-READ-II.
082000      EXIT
```

Figure 4. DBMS Interface Function

```
083600  5800-FILE-F-DB-READ-FORMAT-II.
084100      FIND NEXT F WITHIN E-F.
084200      IF DB-STATUS EQUALS STAT-OK
084300          MOVE ZERO TO END-FLAG-F.
084400      IF DB-STATUS EQUALS STAT-FIND-END
084500          MOVE 1 TO END-FLAG-F.
084600      GET F.
084700      IF DB-STATUS NOT EQUAL TO STAT-OK
084800          PERFORM 9000-ERROR.
084900      MOVE F TO FILE-F.
085200  5890-EXIT-FILE-F-DB-READ-II.
085300      EXIT.
```

Figure 5. Example Translation

DB-ADD, are used. No delete function has been specified. There are three DB-READ formats, one DB-REPLACE format, and two DB-ADD formats specified in the programs, each of which provides a certain data base function. The DBMS interface functions are described in the rest of this section.

DB-READ

DB-READ RECORD OF FILE (Key-File-Name)
  USING KEY (Key-Name)

This function retrieves a record from a key-retrieval type file by using its key value and places it in the user-working-area. The Key-File-Name parameter is the name of the file which contains the specified record. The Key-Name parameter is the name of the variable which contains the numeric value of the key of the specified record.

DB-READ RECORD OF FILE (Member-File-Name)
  WHOSE OWNER FILE IS (Owner-File-Name),
  RETURN END-FLAG (End-Flag-Name)

This function retrieves a record from a specific member file based on its relationship with the specified owner file and places it in the user-working-area. The current record of the specified owner file determines the set occurrence. The current record of the owner file has been established previously by another function. The specified record of the member file is either (1) the next member record of the specified set occurrence following the current record of the named file, or (2) if there is no current record of the named member file, then the specified record is the first record of the member file of the specified set occurrence, or if there are no existing member records of the specified set occurrence or the last record in that file that is in the set occurrence has been previously read, then the value of End-Flag being passed back to the program will be set to 1.

DB-READ RECORD FROM OWNER FILE (Owner-File-Name) OF MEMBER FILE (Member-File-Name)

This function retrieves a record from the specified owner file based on its relationship with the specified member file and places it in the user-working-area. The current record of the specified member file determines the set occurrence. The current record of the specified member file has been previously established by another function.

DP-REPLACE

DB-REPLACE RECORD OF FILE (File-Name)

This function performs an update function by replacing the current record of the specified file with the contents of the user-working-area. The current record of the specified file has been previously established by another function.

DB-ADD

DB-ADD RECORD TO FILE (Key-File-Name)
  USING KEY (Key-Name)

This function adds a new record to a top-node file by using its key value. The contents of the user-working-area are added to the data base as a new record in the named file with the key that is specified by the Key-Name variable.

DB-ADD RECORD TO FILE (Member-File-Name)
  WHOSE OWNER FILE IS (Owner-File-Name)

This function adds a new record to a non-key-retrieval type file by using its

relationship with the specified owner file. The contents of the user-working-area are added to the data base as a new record in the specified member file, in particular as the last member of the set occurrence specified by the current owner file record. The current record of the owner file has been previously established by another function.

4. DBMS Benchmark Programs

Both benchmark programs and test programs are provided to allow the vendor to test the proposed implementation of the physical structure and DBMS interface function translation.

4.1 Test Programs

The vendor is provided with two test programs. The first program generates eight files, with a total of 607 logical records. This data may be loaded into a test data base accessible through the proposed DBMS. The vendor can test the validity of the proposed physical structure with this test data base. The procedure used to load the test data base can be used to load the benchmark data base with necessary minor changes for physical size.

The second test program provided to the vendor tests the translation of the DBMS interface functions into the vendor DBMS commands. The program exercises the DBMS commands, using the test data base. A report is generated which shows the actions completed and indicates any incorrect results encountered during the execution.

Possible error messages are:

XXX BAD ADD - CAN NOT READ

XXX BAD REPLACE - CAN NOT READ

XXX BAD READ - CAN NOT READ

XXX BAD REPLACE - WRONG RECORD

XXX BAD ADD - WRONG RECORD

4.2 Benchmark Programs

The benchmark data base is accessed by both batch and on-line DBMS benchmark programs. These programs must have all DBMS interface functions translated into vendor DBMS commands before execution of the programs is possible. The vendor must add any code necessary to handle errors that may occur due to system failure or implementation errors, by causing the executing benchmark program to abort with an appropriate error message. Any Open or Close functions that may be required by the DBMS must be added by the vendor to the benchmark programs. The vendor's source listings will be reviewed by the Government benchmark team to ensure that no changes to program logic have been made.

The specific set occurrences that are accessed by the benchmark programs are determined by an input seed number. The input seed number determines the first record accessed by the program and, through the use of an algorithm, all subsequent record accesses. The input seed numbers used for test purposes are supplied so that the vendor can verify the output produced by the test runs against the sample output listings provided. New input seed numbers are provided by the Government benchmark team for the timed portion of the benchmark to ensure that the vendor does not optimize certain paths within the data base.

There are three DBMS benchmark programs: the batch data base retrieval program, the batch data base adds program, and the on-line data base access program. The access paths of these programs are such that the areas of the data base that are to have records added or replaced by any one program are never read by other programs. This is to ensure that, regardless of the order of program executions, the results of a series of reads through a set occurrence will not change from run to run either in content (changed due to a replace function) or number (increased due to an add function). The effect of simultaneous users on benchmark results is handled through the use of a separate input seed number for each execution of a program that adds or updates a record. This ensures that each execution is adding or updating a different set occurrence.

The number of times each of these benchmark programs is to be executed during the one hour benchmark depends on the year of system life of the proposed system configuration that the vendor is benchmarking. This particular computer acquisition involves an eight year system life span. The number of executions of the benchmark programs increases each year of the system life span by the percentage of expected increase in the batch and on-line workloads to reflect the predicted increase in data base workload over time.

4.2.1 Batch Data Base Retrieval Program

The batch data base retrieval program retrieves a selected record from file E of

the data base based on the value of the input seed number. By following relationship paths from the original E record, data elements are extracted from seven of the eight files of the data base and used to produce an output report. Although the exact record retrieved during any execution of the program will be the same for any given value of the input seed number, a different record will be retrieved if the seed number is changed.

Figure 6 shows how this program accesses the data base. The following conventions hold for this and all subsequent figures:

(1) The arrow indicates the access path direction.

(2) The note beside the arrow line indicates the accessing function.

(3) A shaded box indicates a file accessed by the program.

(4) The unshaded boxes are files not accessed by the program.

(5) The files that have the file name underlined indicate key-retrieval type files.

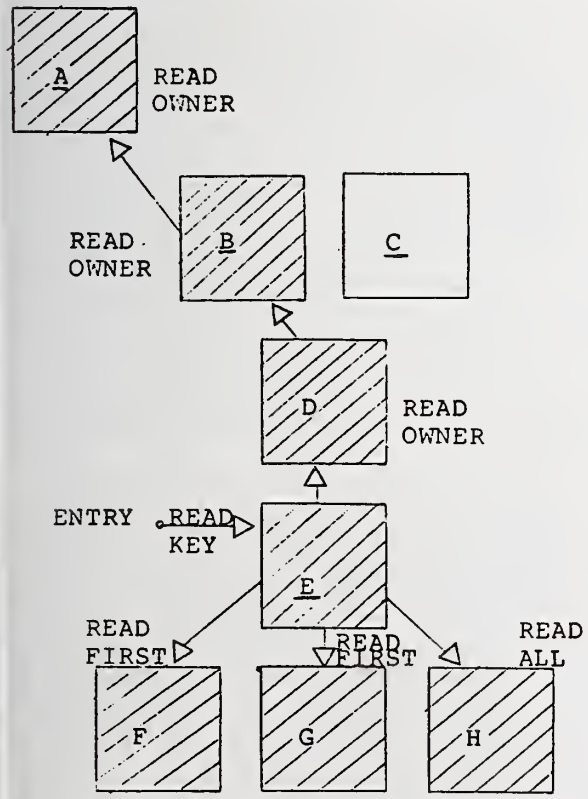


Figure 6. DB Batch Retrieves Program

4.2.2 Batch Data Base Adds Program

The batch data base adds program required a file-generation program that is executed prior to the timed portion of the benchmark. This program generates a tape file of 5,000 transactions required as input to the batch adds program. An input seed number is used to vary the data that is produced. Immediately preceding the timed portion of the benchmark, the Government benchmark team gives a new seed number to the vendor, so that a new set of transactions is generated.

The batch adds program adds the transactions read from the tape to file G, based on a program-calculated key value. This key value is used to determine a set occurrence in the set made up of files E, F, G, and H. A printed report of all the input records and their related file E key value is generated by the program. The number of records to be added is determined by the value of an input card read in at execution time. This effectively imposes the required workload that has been determined prior to the benchmark run. An input seed number determines the data base paths accessed during that particular execution of the program. Figure 7 shows the data base access of this benchmark program.

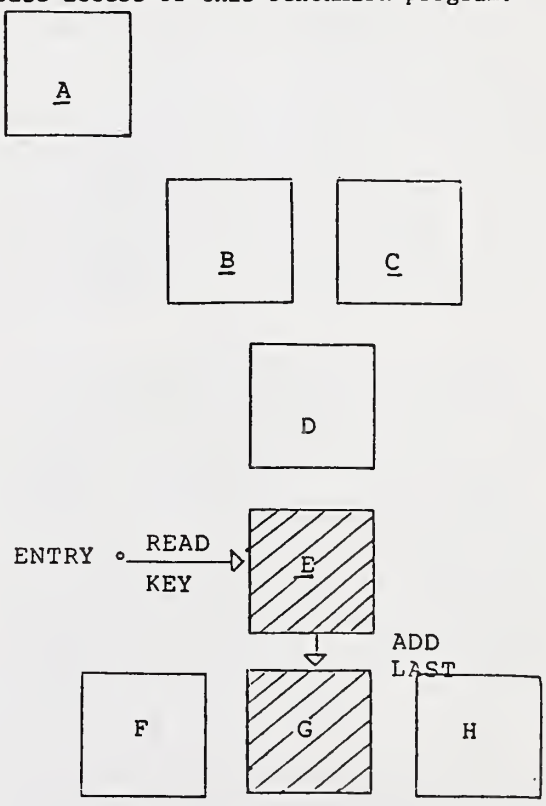


Figure 7. DB Batch Adds Program

4.2.3 On-line Data Base Access Program

The on-line data base access program is executed in an interactive mode through a Remote Terminal Emulator (RTE), i.e., the program is activated by an emulated terminal user. The program accepts a record ID from the terminal and uses the ID to control accesses to the data base key-retrieval type files. Several data values are accepted from the terminal and records are subsequently updated, added, and retrieved through the DBMS. The on-line data access program interfaces with the vendor's DBMS by the same DBMS interface functions used by the batch data base programs.

The interface for the transfer of information between the terminal and the executing program is through COBOL ACCEPT and DISPLAY commands. These commands may be replaced with the vendor's code if necessary. The vendor's code must transfer information between the terminal that activated the program, and the program's data buffers in the user-working-area of the program. This terminal interface may be either direct or through a teleprocessing monitor or transaction processor.

There are four portions of the on-line data base access program: (1) key record update, (2) logical relationship update, (3) add, and (4) retrieval. Figures 8, 9, 10 and 11 show the data base access for each portion of the program.

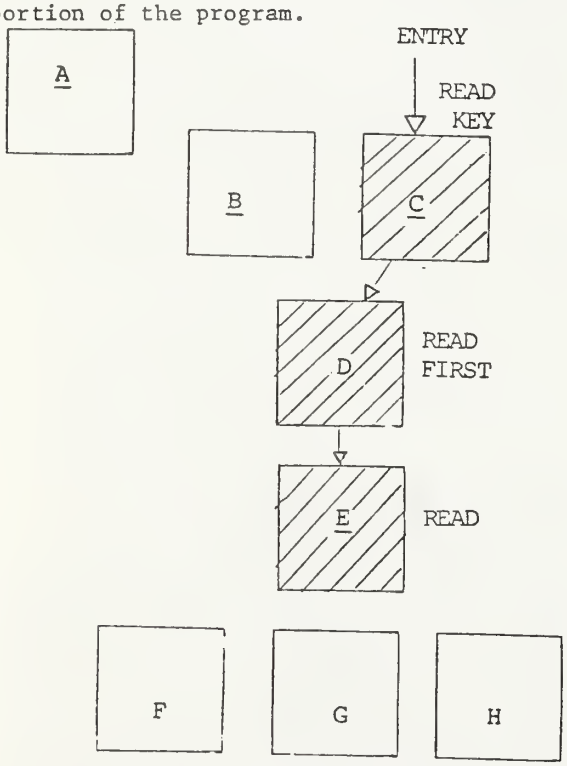


Figure 8. DB On-line Key Record Update Transaction

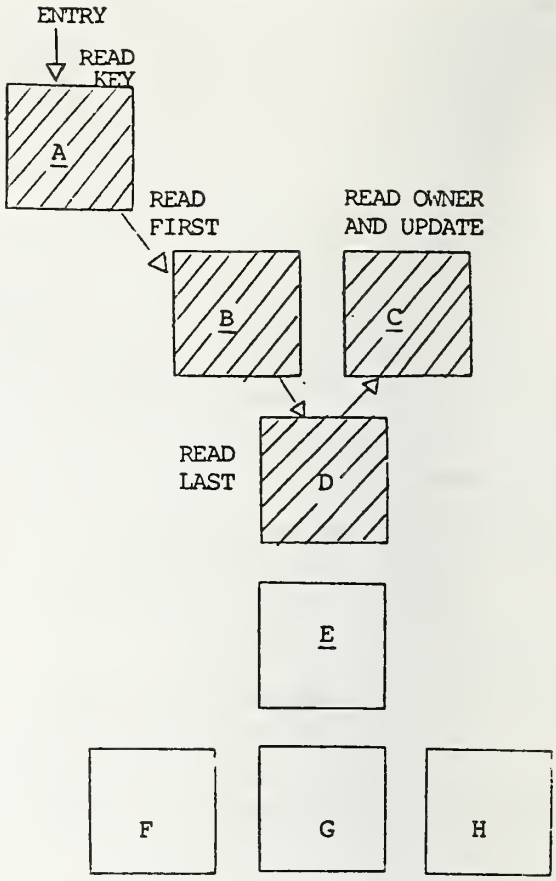


Figure 9. DB On-line Logical Relationship Update Transaction

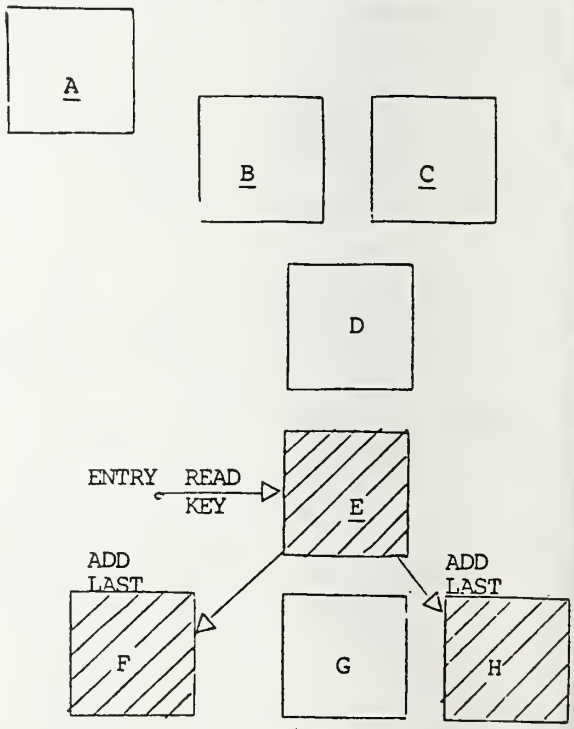


Figure 10. DB On-line Add Transaction

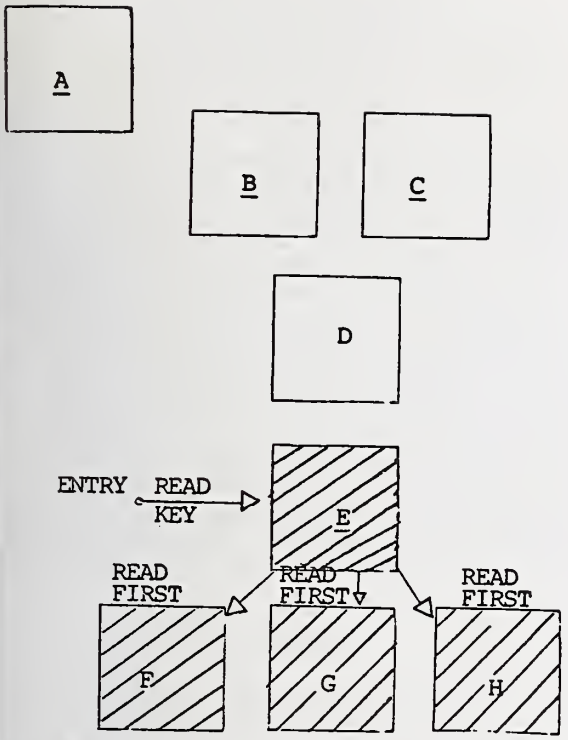


Figure 11. DB On-line Retrieve Transaction

4.3 Testing

This approach to DBMS benchmarking was tested by a team of one programmer and one analyst, from the Government agency acquiring the DBMS. The team was provided with the data base generation programs, the data base benchmark programs, and the benchmark instructions. The team successfully generated the data base and loaded it into a structure which is supported by their current DBMS package. The team successfully translated the data base interface functions in the benchmark programs into the required DBMS commands and executed each program to produce the desired results.

This test cast the agency team in the role of the offering vendor. The analyst of this team had a good background in IDS-I, but no experience with IDS-II, which was the test DBMS package. The programmer had no prior data base experience. The test was accomplished with a minimum of communication between the agency team and the benchmark developers. The team spent approximately three man-months on this project. Taking into account the experience level of the agency team, it is probable that a vendor would be able to accomplish this in a shorter time frame.

4.4 Benchmark Test Evaluation Criteria

The vendor's test results will be evaluated based on the results obtained by the agency's test. The input seed numbers to be used at the Live Test Demonstrations (LTDs) were input to the benchmark programs during the agency's test. The reports generated during the test and samples of the updated files were saved for benchmark test result comparison purposes. The reports generated during the benchmark tests will be compared for accuracy against those generated during the agency's test. At the conclusion of each benchmark test, the vendor will be required to retrieve records from the data base based on a list of record keys provided by the Government benchmark team. This list identifies the file name and unique key of each requested record and includes records that should have been added, updated, read, or had no access made to it during the benchmark test. The contents of these requested records will be compared against the agency's test results. The vendor must successfully match the results of the agency's test, as well as not exceed the one hour benchmark test time constraint, to meet the evaluation criteria.

5. Potential Problems/Benefits

At this time, the agency trying to procure the computer hardware and the DBMS has not released the Request For Proposal (RFP); therefore, there has not yet been any vendor reaction to this DBMS benchmarking approach. The vendor reaction will be crucial in determining whether or not this approach becomes an accepted way to benchmark Data Base Management Systems.

This approach is very expensive in terms of the in-house costs to develop the benchmark and the costs to the vendor to prepare for and run the benchmark. This approach is not appropriate for organizations acquiring only a DBMS, because the relative cost of a DBMS is low compared to the cost to the vendor to prepare for and run a benchmark that uses this approach. The vendor should be willing to spend more in this area if it is a requirement within a large-scale computer system acquisition, where the cost-benefit ratio becomes more attractive to the vendor.

The organization should do a risk analysis study to determine the tradeoffs between the cost to develop and test the benchmark and the cost associated with

acquiring a DBMS which does not meet the organization's needs. The organization should currently have a DBMS, so that there is an available DBMS to test the benchmark programs or have access to another machine for test purposes. The risks associated with sending out untested benchmark programs are too high.

This approach does not take into account either the cost of conversion to the new DBMS or the ease of use to the programmer. The capabilities of a DBMS query language or an on-line update language are also not considered.

An additional problem associated with benchmarking DBMS's is that DBMS performance is affected by the rest of the workload mix that it must compete with for computer resources. Also, the DBMS workload may severely impact the performance of unrelated on-line activity. This problem points out the importance of accurately representing all portions of the workload so that the benchmark test will test the ability of the proposed hardware and software to meet the organization's expected workload requirements.

This approach to DBMS benchmarking appears to have some promise for organizations that currently have a DBMS and need to acquire computer system hardware along with a new DBMS. The appropriateness of this approach must be determined by the individual organization through risk analysis.

6. A Comparison of the FEDSIM and USDA DBMS Benchmark Approaches

The FEDSIM benchmark approach is an expansion of the Department of Agriculture (USDA) approach which has been used successfully in several procurements. The DBMS interface functions of the FEDSIM and the USDA approaches are quite similar in concept. The USDA logical data base structure is a hierarchical structure of three files, each at a different level within the hierarchical structure. Thus the USDA logical data base structure has a depth of three files and a width of one file. All file relationships are one to one. In contrast, the FEDSIM logical data base structure represents a CODASYL network structure, which is necessary to accurately represent the procuring agency's data base processing requirements. The FEDSIM logical data base structure has a depth of five files and a width of three files. There are one to one file relationships as well as one to many and many to one file relationships represented within the logical structure.

Another major difference between the FEDSIM and the USDA approaches is the difference in the manner in which the vendors can access the data base file structures. The USDA approach allows for three separate sets of files: one set for query activities, a second set for update activities, and a third set for addition activities. The FEDSIM approach stipulates that all data base processing must occur against one set of files. This requirement ensures that there will be concurrent on-line and batch queries, updates, and additions against the data base. The concurrent processing requirement is essential to effectively model the procuring agency's actual processing environment.

Reference

(1) United States Department of Agriculture, Automated Data Systems, Kansas City Computer Center, "Live Test Demonstration Manual", 1977.

CPEUG80

Security, Fraud, and Audit

EDP Auditing in the 1980's

or The Vanishing Paper Trail

Richard E. Andersen

Performance Systems Incorporated
30 Courthouse Square
Rockville, Maryland 20850

This paper is designed to provide an overview of (1) traditional auditing methodology and (2) the reasons why these methods must be modified and adapted in order to cope with the "vanishing paper trail" inherent in today's (and tomorrow's) increasingly on-line, data base-oriented, distributed processing, EDP environment.

The three basic types of audits (general, administrative, and applications) will be discussed. Examples of flagrant, and not-so-flagrant, computer crimes will be presented. Practical and proposed solutions for the reduction of these types of DP crimes in the 1980's will be examined.

The major thrust of the paper, however, will address the importance of the applications-oriented audit. The paper will discuss the concept of Computer Management Technology (CMT), and show that this is in reality a type of on-going auditing task. Alternatives will be examined for measuring, managing and controlling key functional areas that have a definite interactive effect on the availability of resources and the performance of system components (or both) in a data processing installation.

Key words: Auditing; audit-in-depth; internal controls; security; auditing aids.

This paper discusses some of the aspects of data processing that, while not in the realm of the data processing exotic, will, I believe, attain ever increasing importance for us in the years of the coming decade.

No level of technology has found itself above the ingenuity of a clever, albeit dubious mind.... not even the computer. One positive, although dubious, distinction in the evolution of man has been his ability to commit criminal acts, no matter how difficult the circumstances. He escapes from escape-proof prisons, he tampers with tamper-proof devices, he burglarizes burgular-proof establishments. These examples merely serve to prove the validity of "Danziger's Law", which holds that as soon as something is invented, someone, somewhere, immediately begins trying to figure out a way to beat the invention.

I'm sure that all of us have heard of myriad examples of computer abuse and/or crime in our own experience. But I think that very few, if any, have valid statistics as to just how much of this subversion goes on. From all indications, a lot more goes on than is ever detected. A lot of computer crime that

does go on, moreover, is never publicly announced. Most security experts have collections of incidents that they have investigated, but which have never been reported to the authorities. Furthermore, some banks and other companies, candidly admit that when an incident is discovered, the corporate victims usually try to avoid the embarrassment and loss of confidence that the publicity might bring.

DP security specialists generally tend to agree that about 85% of detected frauds are never brought to the attention of law enforcement officials. What often happens is that the offender, once detected, is required to make restitution and then is sped on his way...often times with severance pay, and sometimes with letters of reference. One consequence of this is a circulating population of unpunished, unrepentant, and unrecognized perpetrators going from company to company. Probably a more serious consequence, however, is the suppressing of general recognition of the extent of computer crime; and that in turn can lull both makers and users of computers into minimizing it as a threat.

Since the basic topic of this paper is EDP auditing in the 1980's, it might be well to get back to basics and examine some basic terminology:

AUDITING

Source data, methodology, and report conclusions and sums are checked for accuracy and validity through the use of 'studied techniques' and information sources.

AUDIT IN DEPTH

A procedure wherein a detailed examination of all manipulations on a transaction or a piece of information is carried out.

AUDIT TRAIL

The 'trail' left by a transaction when it is processed...the trail begins with the original documents (the transaction entries, the posting of records) and is complete with the report. Validity tests of records are achieved by this method. An audit trail must be incorporated into every procedure. Provision for it must be made early so that it becomes an integral part. In creating an audit trail, it is necessary to provide (a) transaction documentation that is detailed enough to permit the association of any one record with its original source document, (b) a system of accounting controls which provides that all transactions have been processed and that accounting records are in balance, and (c) documentation from which any transaction can be re-created and its processing continued should that transaction be lost or destroyed at some point in the processing procedure.

The sub-title for this presentation is "The Vanishing Paper Trail". Specifically, the inference to be drawn here is that we will arrive at a consideration and discussion of on-line accounting systems. To cite an example of computer crime to make a point: Outside of the world of EDP, most of the present concern among laymen about the latent problems of computer security seems to have emerged since the widely publicized Equity Funding Insurance swindle. While this was really more a case of old fashioned fraud than an instance of computer manipulation, the Equity Funding ripoff could hardly have reached the magnitude it did without the computer's adroitness in fooling auditors from four major accounting firms. The case pretty well demonstrated that conventional auditing procedures were all but helpless when confronting intent and knowhow involving deceit of computers. Auditors are losing their paper trail... that plethora of documents, indeliby inscribed orders, invoices, bills and receipts that the men in green eyeshades used to plow through on the trail of auditing irregularities.

There appears to be at least one group that has perhaps derived benefit from revelations such as those of the Equity Funding swindle. This is the relatively small group of specialists that appear to be able to write programs to make the computer do the auditing. Auditing through the computer rather than around it; that is, to perform various types of cross checks within the computer, and to throw up warnings when certain suspicious transactions occur. This sort of auditing, however, is only as dependable as (like everything else that goes on inside the computer) the computer itself. Unfortunately, computers can be programmed to lie or conceal as easily as they can be programmed for truth.

Rather than throw up our hands in despair and exclaim, "A lock only keeps as honest man honest!", we should continue to search for viable ways to audit through the computer...to continue to modify traditionally ingrained techniques to cope with this vanishing paper trail.

A nominal amount of research in the preparation of this paper revealed, among other things, a promotional brochure distributed by a company which is quite famous in our industry for the fine reference material it publishes. Herewith, a few pertinent quotes:

"Auditing through the computer-rather than around it-will only become a reality if you fully understand the confusing maze called 'data processing'. EDP AUDITING, the only bimonthly updated reference service, provides you with timely, comprehensive information on the DP function, plus the latest methods and techniques enabling you to work with the DP department to improve your performance.

Designed for internal and external auditors, and DP managers, this in-depth 'how to' reference contains the most current information and approaches to the EDP auditing function on the market. And because it is regularly updated to provide an ongoing collection of reference material, its benefits far outweigh those of a bound book. Initially with EDP AUDITING, you'll receive a rugged loose leaf binder containing more than 40 information-filled portfolios. Then, every other month your service will be supplemented with 4 or more portfolios, guaranteeing that you have the most timely facts, methods, and advice at your command.

Let EDP AUDITING help you close the gap between the EDP function and your auditing tasks. Let it enable you to audit through the computer - plus much, much more."

It is in no way my intent to demean this promotional offering regarding EDP Auditing. Quite the contrary...because in the great deal of research and reading undertaken in the preparation of this paper, it is one of the very few practical approaches and suggestions I encountered to help the EDP Auditor with the tasks that are going to be facing him or her...and the more involved and difficult auditing tasks facing him or her in the coming years of the 1980's.

This approach suggests, perhaps, the linking of two separate specialities as a result of our 'technological progress'. This technological progress is usually guaranteed, for a time at least, to create a demand for specialized skills that will be in very short supply. I believe that such a field will be computer auditing, where for all the growth that has occurred over the last few years, there is still a huge gap between what is, and what is needed. I believe the reasons for this are relatively simple. Computers happened too quickly for the accountancy profession, which is accustomed to a more leisurely rate of change. New "marriages", however, tend not to be traumatic for the computer profession. Data processing people are used to seeing their speciality branch out into new fields, but for the accountancy profession, the arrival of the computer has, I believe, produced a bit of a trauma, and in some cases, a nasty shock.

We have all witnessed the increasing use of on-line systems and real time Data Base Management Systems in accounting applications, both in the larger and more sophisticated companies, as well as many applications in the smaller companies. While these systems are well established, they still present many new and different challenges to the computer auditor.

Again, in the old batch systems it was still possible for the auditor to pretend that the computer didn't exist, and audit "around" the computer. He or she simply checked or matched what came out with what went in. Some of these "batch" tools will continue to be used, and with efficiency... the old "hash total" springs to mind. However, we see a move toward concentrating efforts on reviewing data flows within computer systems, in order to establish the degree of reliability that auditors can place on their company's internal controls. These controls are merely a set of procedures and checks designed to ensure that valid transactions, and only valid transactions, are processed, accepted, and recorded completely.

It is often more economical, and better auditing practice, to use the computer to extract exceptions, reports, and records of all abnormal entries (and samples of transactions and balances).

All incorporated companies must have their accounts audited. In order to express their opinion on the accounts (in order to report that they are "true and fair"), the auditors must achieve some reasonable assurance. If the internal controls are sloppy, the auditor has to spend more time checking what lies behind the figures. And the more time he or she spends, the more it costs the clients.

Extraction of samples by computer program, even very large samples, needn't be expensive. It is the verification of the data in the sample that costs money, since someone has to go and see that the source document exists, or that the stock did in fact arrive, etc. In reviewing real time systems, we see that they have revealed a wide range of additional problems relating to weaknesses of internal controls, and weaknesses in the development of interrogation software. An obvious example is inadequately supervised terminals. If anyone, not just authorized people, can get to a terminal, risks of that abuse are increased. Although many installations use passwords, it often happens that the password stays in use for so long that it loses its confidentiality. Even the most elaborate password systems will come crashing to the ground if the operator wanders away from an unattended terminal without signoff. This is often compounded by the fact that terminals tend to be located in large open areas and offices where there is little or no physical security. Again, in reference to on-line systems, accounting data tends to be keyed in without control totals being first generated. This increases the danger of mis-keys and mis-reads going into main files without detection, at least until they turn up as meaningless stock orders, or astonishing paychecks. Guarding against these eventualities means stringent program validation checks on data...more so than in the batch environment. Usually, only those systems that require keyed data to be input by two separate individuals will provide a uniformly high level of input accuracy. Controlling the input and resulting accuracy of master data files is much more difficult than in the old batch systems because the main files are seldom read through in their sequential entirety. This control can often only be achieved through the use of control totals of movements in all key fields. Fairly frequent sequential passes of the main files will have to be built into the systems, and checks made that the sum of the brought forward control totals plus the movement control totals equals the sum of the individual values of the files. Additionally, sufficient transaction history obviously must be kept, so that if the two sets of totals fail to agree, rapid recovery is possible from a sequential pass. In practice, there is a tendency to reduce the amount of transaction history held on on-line files, especially compared to the amount held on most batch systems. The most important difference introduced by on-line, and Data Base Management systems, from an audit point of view, is that the functions of main data file security are being absorbed by the Data Base software itself. This concentration of information means that the Data Base Administrator is becoming a very important person. I don't mean to imply that DBA's are natural fraudsmen, but they are in a unique position of having an excellent knowledge of a broad range of information in the company, and of the relationship of the pieces of information...one with another. Since a great deal of audit thinking on control centers on a division of duties, classically, traditionally, historically...so that no one person has too large an area of influence, the auditors' nervousness about Data Base Administrators is understandable. An old audit maxum is to get as many involved in each process as you can so as to make collusion more difficult. While it is rare that a Data Base Administrator has actual custody of a company's assets, if he is allowed to run software, he has the ideal opportunity of making un-authorized amendments of one sort or another.

Therefore, specifically and particularly in view of the considerations that have to be taken into account as a result of on-line systems, the auditor has a key role to play in the design of intended new systems at the earliest possible stage of the planning. This will enable him or her to assure that the proposed system will incorporate sufficient audit (and good) basic controls. Early involvement of the audit function in the design of new systems cannot be stressed too strongly, because the internal and external auditor are concerned that these control standards and procedures in effect in a DP installation are functional. In addition, the auditor of the 1980's is going to need to test the reliability of the system in processing various types of transactions.

The auditor's evaluation of the sufficiency of internal control procedures will also affect his or her determination of the extent necessary in order to satisfy him or herself regarding the reliability of those systems.

DP has had a major effect upon auditing, due primarily to the loss of this visible paper trail (the vanishing paper trail), and this increases the importance of internal controls...which in turn underscores the need for early involvement of the audit function.

While good system design controls do not in themselves eliminate the need for auditing, they do significantly reduce the need for testing necessary to the audit function. Again, to re-iterate the concept of auditing around the computer, the first reaction to DP by early auditors was to attempt to perform their audits with the printed records and output provided by the system, the theory being that if a sample of the system output was correctly obtained from the system using the system's input, then the processing itself must be reliable. This was a reasonable approach perhaps ten or fifteen years ago when the knowledge of DP auditors was extremely limited. However, the increasing difficulty of applying this approach to the vanishing paper trail, and the development (and continuing need for development) of better auditing methods, have combined to discredit this old approach of auditing around the computer. The only alternative to this is auditing through the computer.

Some examples of this might be (1) the use of test decks using sample transactions, either correct or intentionally incorrect, or (2) the use of specially written computer programs for interrogation and control. What about the aspects of data security? The growing reliance of many companies on DP to process their transactions and store their business data has created, and continues to create, new vulnerabilities and concerns. This is especially true in the on-line environment. A tremendous amount of computerized data, including business forecasts, employment data, payroll records, etc., has to be kept confidential; but maintaining that confidentiality has become increasingly more difficult with today's DP methods.

Advances in direct access and mass storage systems and technology, coupled again with improved and simplified terminal usage, have given numerous people throughout the company access to sensitive business data. Information that was once limited to viewing by top management might now be subject to the prying eyes of employees, or even competitors. Even more of a threat than intentional security violations, however, is the damage to sensitive data through well intentioned error. The great majority of data losses is due to the fact that, simply, people make mistakes. The same technology that makes DP such a boon to business can also greatly amplify the effects of human error. We all have probably experienced examples of that. Irreparable damage to critical company records can be caused by honest employees who inadvertantly misuse data. Additionally, because of the dramatic rise of, and projections for increase in, computer crime, as we will touch upon later, companies and Federal Agencies are demanding increased security, privacy, and control in DP operations.

To meet these demands, EDP auditors of the 1980's must develop a working understanding of DP. As our brochure indicated, they must be able to work with DP personnel, to secure and control this valuable information stored in DP files. The DP auditor of the 1980's will have to know how to audit through the computer rather than around it. And this implies that he or she will have to be able to do numerous things. A partial list of those things follows:

- He or she must be able to deal with the important issues such as the reconstruction of data files, physical security, and the theft of services or data in the system.
- Also, he or she must be able to take an active role in developing policy on auditability, testing, standards, and general controls.
- He or she must be able to provide management with independent assessments of DP decisions, and their impact on the company in general.

He or she must be able to make certain that the alternatives or risks for a project have been carefully considered, and that the technical hardware and software solutions are correct and reasonable, and that the costs are reasonable.

Sheer volume alone makes controlling sensitive data more difficult and all the more critical, because as the amount of sensitive data increases, so does the need to safeguard it. Hundreds and even thousands of jobs are run on computer systems each day. The task of monitoring which jobs are using what data and which resources is beyond the capability of any human operator.

The only feasible solution to the problem of protecting data is to let the computer itself monitor and control data access. This in turn will require data software security systems that will combine data access control with computer access control. We might well ask at this point what should be the goals for the 1980's for a data security system that takes into account auditing considerations. As I see it, there are three fundamental goals:

(1) To determine and detect improper access attempts

(2) To prevent such attempts from succeeding

(3) To record successful accesses and unsuccessful accesses to data, on an audit file

To these ends, a security and auditing system has to be able to (1) prevent all unauthorized access to all forms of business data, including such things as system libraries, application libraries, and business forecasts, (2) provide authorized personnel easy access to their data... ideally without them even being aware of or concerned about the security and auditing mechanism, (3) immediately notify an operator that a security violation attempt is in progress, (4) identify and verify every system job and user, (5) provide an audit trail of every successful and unsuccessful access attempt, being all the while transparent to the user, and finally of course (6) add minimum overhead to the system, be easy to install and use.

Such security systems and auditing aids are on the market now, in varying degrees of completion and efficiency. But in this area, much more needs to be done...and much more education needs to be brought forth to the DP community in general. This is not a one way street... a street where only the auditor needs to learn new techniques.

I believe that there is good news and there is bad news when one considers computer security and auditing for the 1980's. I think that the good news is that computer security systems will become better than ever. There will be help for the EDP Auditor. However, I fear that the bad news revolves around the fact that probably organized crime will make its true data processing debut in the 1980's. It is conceivable that the mob will make use of career criminals trained in DP by prison rehabilitation facilities (perhaps along with top DP technicians) recruited, sometimes unwillingly, from the ranks of otherwise honest DP personnel. I believe that the incidence of large computer crimes will go down, but that the size of the individual crimes will rise. Recently published figures indicate that the national average of $450,000 per computer crime will skyrocket. Professional criminals will be much better at their craft than the DP hobbyist, who is now still the major perpetrator of computer crime.

Financial institutions, which have been working for years toward safety of their data, will probably be better off than most DP users. Many so-called "friendly" information systems and interactive word processing systems provide little if any security provision. Additionally, they actually help the computer crime perpetrator find what he is looking for by putting the information in an easy-to-read format. In the case of information management systems on mini-computers, these offer little at the present time in the way of protection. I believe that security precautions for smaller systems will be one of the greatest needs, and should be one of the highest priorities, of the 1980's. Companies will also be faced with the dilemma of deciding when their computer systems are safe enough. If a company goes too far in the area of security control, no one will be able to use the computer, and production will fall off.

Mr. Donn Parker, an internationally recognized leading world expert on computer crime, abuse and fraud, suggests that the only completely safe computer is one that is not used. He maintains, and I believe correctly, that any security system can be breached with the proper combination of know-how and intent. He suggests that by coupling more frequent auditing with more sophisticated security systems, the incidence of the small DP crimes could be held to a minimum. However, because laws vary from state to state, it is conceivable that a computer thief can even avoid prosecution by committing his crime in a state without computer crime laws. This is an aspect of computer crime that will have to be addressed in the 1980's also.

In our business at Performance Systems Incorporated, we repeatedly come into contact with top management personnel who express their frustration in not being able to establish or bring about specific management controls on their DP departments. We find usually that one of the main reasons that DP is undermanaged, undercontrolled, underdocumented,...and misunderstood... is that historically the DP manager has usually reported to a comptroller or financial officer of some type. Additionally, he has been only loosely responsible to some type of user steering committee for DP. But now we are moving into an era where "corporate directors of DP and Telecommunications activities" (for example) are being appointed who have a DP background as well as formal indoctrination and training in management principles and philosophy. This is an encouraging trend, hopefully to be maintained and even accelerated in the 1980's.

I also believe that these individual companies that have been founded and revolve around DP security concerns are in the vanguard of a major contribution to our DP way of life in the 1980's.

Finally there is an element of semantics involved in the term EDP Auditing. Sometimes there is a conflict between the purely financial aspects of auditing and the computer aspects of the work of (for lack of better term) "the computer systems audit staff". Among the services that we provide at PSI is something that we call the Data Center Review. It certainly is not a new or revolutionary concept, but we think that our approach is a bit innovative. Initially it was suggested that this activity be referred to as an EDP Audit. I believe that wiser heads prevailed, and in discussing the conceptualization and image provided by the term EDP Audit, it was generally agreed that while this was meant to be a systems audit as opposed to a financial audit, we should provide a descriptive phrase that was more meaningful. Therefore, the term EDP Audit was replaced with the term Data Center Review. The mechanics of the Data Center Review merely embrace another type of audit; and this is a facet (and only one facet) of Computer Management Technology (CMT).

This Data Center Review takes into account that in a sophisticated DP environment there exists a necessity to develop highly specialized skills. Departments or branches are formed having well defined missions that are essential but narrow in scope and responsibility. Perhaps this is due in part to the traditional and historical splitting of functional involvement...again, in line with our old audit maxim. Even the elements of vendor support (e.g., systems engineering and customer engineering) tend to align themselves with these various groups according to their individual expertise and experience. In many cases, management decisions are made on the basis of the recommendations of specialized technicians whose performance is measured in narrowly defined responsibilities. Such decisions could have an adverse effect on other equally important functions in the organization.

In the process of conducting Data Center Reviews in the past, our experience has shown that several key areas within an installation have a definite interactive effect on overall system performance. The methods and procedures used to manage and control these areas are vital in meeting and maintaining performance objectives. We have developed and documented techniques for reviewing these areas within an installation. I believe you need an organized and documented method of determining how efficiently and effectively your data processing systems are functioning and operating. The Data Center Review provides you with this capability by identifying the data available from each of the Data Center's functional areas using simply a set of questions, or checklist, to analyze procedures in each of these areas. All data sources are presented as a "shopping list", and the checklists are modular. With this format, an experienced analyst can easily

tailor a data source and checklist package to an installation under examination.

However, too often we see many instances of similar "guidelines and checklists" available, used, filled out, ratings applied,...and yet no analysis...no follow-up...no qualitative reporting is done. Many times, we feel, that is where this approach falls down and is lacking, because all the collected data in the world is of no use unless it is subjected to rigorous analysis directed toward determining the implications of all this collected data. Perhaps the most effective means of emphasizing this point is to quote Johann Wolfgang von Goethe who wrote in 1810: "The modern age has a false sense of superiority, because of the great mass of data at its disposal; but the valid criterion of distinction is rather the extent to which man knows how to form and master the material at his command". So we see, after all, that there is an element of sameness in the world of rapid change in which we live.

We have covered a wide range of topics under the general heading of EDP Auditing, Security, and Computer Fraud. However, no discussion of EDP Auditing, Data and Systems Security, Controls, Checks, and Balances, would be complete, I believe without pondering some questions, even if only briefly: What of OVERcontrol, OVER-auditing, OVERsecurity paranoia, and other potential abuses?

I would therefore like to leave you with a little food for thought, in the form of my only two overhead projector transparencies. I am indebted to Professor Dennie L. Van Tassel of the University of California for his inspiration and (in part) format:

CASHLESS SOCIETY

NATIONAL DATA BANK

DAILY SURVEILANCE SHEET

CONFIDENTIAL

JULY 9, 1997

SUBJECT: NAME
EMPLOYER
ADDRESS
SEX
AGE
MARTIAL STATUS
OCCUPATION/TITLE

PURCHASES:	WALL STREET JOURNAL	.25
	BREAKFAST	2.50
	GASOLINE	22.00
	PHONE (328-1826)	.15
	PHONE (308-7928)	.15
	PHONE (421-1942)	.15
	BANK (CASH WITHDRAWAL)	120.00
	LUNCH	3.50
	COCKTAIL	1.50
	LINGERIE	26.95
	PHONE (369-2436)	.35
	SCOTCH (1/2 GAL)	12.85
	NEWSPAPER	.25

**** COMPUTER ANALYSIS FOLLOWS ****

*** COMPUTER ANALYSIS ***

OWNS STOCK (99% PROBABILITY).

HEAVY STARCH BREAKFAST - PROBABLY OVERWEIGHT.

BOUGHT $22.00 GAS. SO FAR THIS WEEK HAS BOUGHT $38.00 GAS. OBVIOUSLY DOING SOMETHING BESIDES JUST DRIVING TO WORK.

BOUGHT GAS AT 0857. SAFE TO ASSUME HE WAS LATE TO WORK.

PHONE NUMBER 308-1826 BELONGS TO 'FRANKIE THE LAYOFF'. FRANKIE ARRESTED 1978 FOR BOOKMAKING.

PHONE NUMBER 308-7928 - EXPENSIVE MENS' BARBER. SPECIALIZES IN BALD MEN OR HAIR STYLING.

PHONE NUMBER 421-1931 RESERVATIONS FOR LAS VEGAS (WITHOUT WIFE.) THIRD TRIP THIS YEAR TO LAS VEGAS (WITHOUT WIFE). WILL SCAN FILE TO SEE IF ANYONE ELSE HAS GONE TO LAS VEGAS AT SAME TIME, AND WILL COMPARE TO HIS PHONE NUMBERS CALLED.

WITHDREW $120.00 CASH. VERY UNUSUAL SINCE ALL LEGAL PURCHASES CAN BE MADE USING THE NATIONAL SOCIAL SECURITY CREDIT CARD. CASH USUALLY USED ONLY FOR ILLEGAL PURPOSES. IT WAS PREVIOUSLY RECOMMENDED THAT ALL CASH BE OUTLAWED AS SOON AS IT BECOMES POLITICALLY POSSIBLE.

DRINKS DURING LUNCH.

BOUGHT VERY EXPENSIVE LINGERIE NOT HIS WIFE'S SIZE.

PHONE NUMBER 369-2346 - MISS SWEET LOCKS.

PURCHASED EXPENSIVE 1/2 GALLON SCOTCH. HAS PURCHASED 3-1/2 GALLONS SCOTCH IN LAST 30 DAYS. EITHER HEAVY DRINKER OR MUCH ENTERTAINING.

*** OVERALL ANALYSIS ***

LEFT WORK EARLY AT 1700, SINCE HE PURCHASED SCOTCH 1 MILE FROM HIS OFFICE AT 1710 (OPPOSITE DIRECTION FROM HIS HOUSE).

BOUGHT NEWSPAPER AT 1930 NEAR HIS HOME. UNACCOUNTABLE 2-1/2 HOURS. MADE THREE PURCHASES TODAY FROM YOUNG BLONDES. (STATISTICAL 1 CHANCE IN 78). THEREFORE PROBABLY HAS WEAKNESS FOR YOUNG BLONDES.

Tracking Potential Security Violations

R. L. Lehmann

Union Carbide Corporation
South Charleston, WV 25303

Abstract: Security concerns not only involve providing restricted access to computer resources, but frequently require investigative studies to track suspicious utilization for potential company violations. For such studies it is important to have pertinent information for the entire job in order to trace what is being done. Sometimes very little information is available to use as a key for isolating the jobs in question. This paper describes a tool developed at Union Carbide Corp. to enhance such investigations on IBM MVS systems.

Keywords: Audit trails; computer fraud; computer security; job tracking; security violations, SMF retrieval.

1. Introduction

Frequently security violations involve those who are authorized or have access to the sensitive data of concern. Current security measures are of little help in preventing such unprincipled people from obtaining their objectives. Indeed RACF, SECURE and ACF2 minimize access to such areas but to the authorized or knowledgable even they must yield to the master's will. Acknowledging that a completely fool proof system is neither practical nor possible this paper seeks to address the problem once violation is detected or suspected.

Present products may aid in warning of possible violations but what does one do then? Accusing one falsely could have serious consequences. Ability to track jobs based on varying bits of information would be of great help in gathering needed data to determine the nature of the suspected infringement. Since thieves and saboteurs tend to cover up their tracks, sufficient flexibility must be provided to track their use of computer resources. Of course the supersleuth may eradicate all traces of his presence and leave you helpless, but even that condition restricts your concern to only those few 'sharp' individuals capable of such system internal manipulation.

Various questions immediately surface when considering one suspected of violation. What information was accessed by the individual? What computer resources did he or she use? What did the individual do with the information accessed? When was the infringement made and how could it have been avoided?

In order to aid in answering these questions UCC developed a means of accessing computer generated data to piece together information needed in resolving these concerns. Five basic design steps are of importance in describing the resultant facility;

[1] What form of requests want to be made?
[2] What information is desired?
[3] What data is available to supply this information?
[4] How can it be retrieved economically?
[5] How is it to be reported?

Detective work traditionally obtains bits and pieces of information, with which it must collect all relevant data, such that all conditions of fact are met. Tracking security

violations has many similarities with respect to the meager amount of information one may have at the start.

In order to accommodate such tracking, basic clues have been classified into nine major areas. Any combination of these nine variables may be requested to define retrieval conditions. Multiple requests are also permitted in any given run. This is accomplished by ANDing the various conditions defining each request and ORing the number of requests.

2. Form of Requests

Nine basic parameters were isolated to define the primative set of request identifiers. These may be grouped into four basic categories:

1 Job Identification
   a. PGMR - Programmer Name
   b. PGM - Program Name
   c. JOB - Job
2 Data Set Identification
   a. DSN - Data Set Name
   b. VOLSER - Volume Serial
3 Time Period
   a. DATE - Date
   b. TIME - Time
4 Accounting Concerns
   a. CC - Charge Number
   b. RIN - Remote Number

Any combination of these request identifiers may be made, depending upon the amount of information known and restriction of output desired. The syntax related to each is shown below.

2.1 SYNTAX Description

Programmer Name: Perhaps all that is known is that programmer X appears to be violating company standards. It would be desirable to obtain all the information available regarding that programmer. But programmers invariably spell their names differently from one job to another. At times they may use their initials whereas other times simply the last name. Some may separate their initials with blanks, others with periods and a few even use the underscore character. At times abbreviations are even witnessed, especially when the name is long. Consequently it is desirable to be able to simply request a character string that if found anywhere within the name field a retrieval will be accomplished. Such a request is made by simply stating:

```
PGMR='character string'
```

Program Name: In addition to the programmer name, frequently one may need to determine the users of a particular program. When checks run against output from highly confidential programs reveal problems, a report of all individuals using that program may be obtained by specifying:

```
PGM='name'
```

Job Name: Jobs may be traced by simply specifying the job name:

```
JOB='name'
```

Data Set Name: Suppose suddenly one day a critical and sensitive data set has been destroyed. Immediately you wish to know who had been using that data set. A request of the form below will produce a report showing every user for the period of interest accessing the data set having the exact name as specified:

```
DSN='name'
```

Sometimes, however, the exact name with all its qualifiers may not be readily available. The unique identifiers on the other hand are known and it is desired to just specify these to track the users. Similarly, it may be desired to identify those who used any of a family of datasets all having a common string of characters somewhere within its name. This may be accomplished by:

```
DSNX='character string'
```

Volume: Disk or tape volumes containing sensitive data may be accessed in ways other than through standard data set name requests. Circumstances may arise where it is necessary to know who used such a volume and whether it was used for input or output. This request is specified by the statement:

```
VOLSER='name'
```

Date: Frequently one's concern is restricted to a specific period of time. To eliminate data outside the period of interest the date option provides the ability of setting day boundaries. This also illustrates the facility of specifying multiple conditions, which will be described in more detail later. The date is specified in Julian form as the following examples show:

```
DATE=80123
DATE>80119 & DATE<80120
DATE>80112 & DATE<80116 &
             Date¬=80114
```

Time: But the period of interest may be concerned with only a portion of the day. Thus a time option is also available to set the boundaries of time within each day of interest. This condition is in the form of hours and portion of hours as shown below:

```
TIME>20.0
TIME>8.0    & TIME<17.5
```

Charge Number: At times infringement problems may involve users who manage to direct personal computer use to charge numbers other than their own. Ability to track specified charge numbers is also available by requesting:

```
CC='charge no'
```

Input Remote Number: One last request type was designed to enable one of monitor users at a given remote station. Suppose violation is suspected at a given site during the weekend. Activity at that remote station could be retrieved by:

```
RIN='remote number'
```

2.2 Summary

Nine different condition types are provided such that any combination can be used to uniquely define the conditions representing the information known. All except DATE and TIME have character values. In addition, DATE, CC and TIME may utilize the operators =,> or <. The remaining types may only use =. Exception reporting may be accomplished by using the ¬ operator in conjunction with the >, < or = operators. For example the following request:

```
DATE=80246&TIME<6.&RIN='R99.RD1'&CC='9500U'
   &PGMR ='JONES'
```

will retrieve all jobs submitted from Remote 99 from midnight to 6:00 a.m. on day 246 using charge number 9500U except those of Jones.

3. Implementation

To provide flexibility and ease of maintenance it was decided to use a PL/I program to process the input conditions and SAS to do the data retrieval. PL/I was chosen because at the time SAS did not have the string mainpulation capabilities that are now available. The PL/I program processes the retrieval parameters, checks for syntax and other error conditions producing either diagnostic messages, a condition code which terminates the run, or SAS macros generating the SAS code necessary to retrieve the conditions required. These macros are written to a data set which is then concatenated with the SAS routine. Each error message describes the problem encountered as well as the actual request that was made. An example of the diagnostic messages are illustrated below. The selection types MAJ and MIN will be discussed later.

Input Requests:

```
MNR DSN='K.CACI.SIMLIBB' & VOLSER='KDD071' & PGMR='CANDY'
MAJ
MIN DATE='80091' & VOLSER='KRR002' & RIN='R7.RD1'
MIN PGMR=LEHMANN & TIME>'8.' & TIME<12.
MIN PGM='A9500TFST' & CC='95000U'
MIN JOB='ARLLPMITT' & PGMR='R.L.LEHMANN' & VOLSER='KDD01#'
MIN RIN='RD10_RD12' & CC='9000$'
```

Diagnostics:

```
ERR01: INVALID SELECTION TYPE. MUST BE MAJ OR MIN  CONDITION:
ERR03: MISSING OPERATOR IN SELECTION CONDITION.    CONDITION:
ERR08: INVALID SYNTAX- DATE NON-NUMERIC            CONDITION: DATE='80091'
ERR07: INVALID SYNTAX- MISSING QUOTES              CONDITION: PGMR=LEHMANN
ERR09: INVALID SYNTAX- TIME NON-DECIMAL            CONDITION: TIME>'8.'
ERR10: INVALID SYNTAX- PROGRAM > 8 CHARACTERS      CONDITION: PGM='A9500TEST'
ERR16: INVALID SYNTAX- CHARGE > 5 CHARACTERS       CONDITION: CC='95000U'
ERR06: INVALID SYNTAX- JOB NAME > 8 CHARACTERS.    CONDITION: JOB='ARLLPMITT'
ERR13: INVALID SYNTAX- PGMR NAME NON-ALPHAMERIC    CONDITION: PGMR='R.L.LEHMANN'
ERR15: INVALID SYNTAX- VOLSER NON-ALPHAMERIC       CONDITION: VOLSER='KDD01#'
ERR22: INVALID SYNTAX- REMOTE # > 8 CHARACTERS     CONDITION: RIN='RD10_RD12'
ERR24: INVALID SYNTAX- REMOTE # HAS NON-REM CHAR.SCONDITION: RIN='RD10_RD12'
ERR17: INVALID SYNTAX- CHARGE NON-ALPHAMERIC       CONDITION: CC='9000$'
```

4. Information Desired

Having defined the kinds of requests a security officer may want to make, it is necessary to determine what information is required for retrieval. After some iteration of ideas, it was decided that the following set of parameters would be sought on a job level basis. Specifying a job level basis means that all steps associated with the job involving suspected violation are to be included in the report even though some may not contain the conditions requested. For example, if a data set name was requested all the steps of every job utilizing that data set would be retrieved, even though some of them may not specify the data set of interest. This is to facilitate analysis of what was done with the data set. Perhaps one step copied the sensitive information to a new data set after which processing or additional copying to tape was performed. By removing the suspicious activity to the new data set, detection may have been avoided or complicated if only the step containing the

requested data set had been retrieved.

Therefore for each job, the following information will be incorporated in the printout:

REPORT HEADING	DESCRIPTION
JOB	JOBNAME
DATE	DATE
TIME	TIME
PGMR	PROGRAMMER
PGM	PROGRAM
STEPNO	STEP NUMBER
STEP	STEP NAME
CODE	COMPLETION CODE
CC	CHARGE CODE
RIN	INPUT REMOTE NUMBER
ROUT	OUTPUT REMOTE NUMBER
DSN	DATA SET NAMES
VOLSER	VOLUME SERIAL IDENTIFICATION
IO	I/O TYPE CODE

With the above information, one immediately knows who submitted the job, at what time of what day and what programs were executed. In addition, preceding and/or succeeding steps surrounding the step of concern are available and all data sets used by the job are listed giving both their data set name and their volume serial number. Further description is provided to identify whether the data set was used for input or output. In order to locate the source of the violation both the input and output remote locations are provided, as well as the charge code under which it was submitted. Lastly the completion code has been included to validate that the job ran to completion.

5. Data Available to Supply This Information

Now that both the kinds of request and the information desired is determined, we turn our attention to investigate what data is available to supply this information. Being an IBM installation we have available to us SMF (System Management Facility) and the program package RMF (Resource Management Facility) as well as Boole & Babbage's package, CMF (Comprehensive Management Facility). All of these products can provide a data base containing information on all the work performed upon the computing system. After evaluation it was decided to utilize the SMF files for our basic data base, since neither RMF nor CMF provide data set name information. The records of interest that are used are the type 4,5,6,14,15,26 and 35. From these all of the information desired may be retrieved. At the time of the beginning of this project the SE-2 (System Extension) capabilities were not available. Future revision could take advantage of the type 30 records in the SE-2 product and thereby eliminate the need to piece together various bits of job information from different records. Due to SMF generation being an event-driven system, records pertinent to the specific request are generated at various times within the processing period. Since job name, date and time occur in all records, these parameters will define a job stamp for correlating all records of a given request. This job stamp is also used to create a job table associated with the conditions specified by the input requests.

6. Retrieval

Economy Factors: A major factor in dealing with SMF data retrieval is that of economy. Undoubtedly various requests may require looking through a week's or month's worth of data which means dealing with millions of records. Consequently it is desired that only one pass of the SMF data be necessary to retrieve the information to satisfy the request. But since the generation of SMF data is event driven, and all of the information concerning a job that meets the options that have been requested is desired, one cannot determine all of the records needed on the first pass through.

For example, suppose a request was made in which only the data set name was given. That data set may not have been used until let us say, the third step within the job. While processing the SMF data other data sets used in the same job produce SMF records which precede those regarding the desired data set. However, it's not until we reach the record containing the given data set desired that we know with what job it was associated. Consequently, we could not retrieve the earlier records beforehand. Even sorting all of the SMF data, which would be prohibitive, so that all of the records pertaining to a given job are together, would not easily resolve this problem. Thus, it is obvious that multiple passes of some nature must be done. Furthermore, retrieving data from unnecessary records is a waste of computer time. Therefore, some facility should be provided to eliminate as much data as possible beforehand.

A means of subsetting the data file may be accomplished by specifying major, MAJ, conditions. Since all SMF records contain the date, time and job name, if the user knows any of these parameters he may speci-

fy major options and thereby only records containing these specifications will be retained. If multiple requests are made in the same run, all of the requests will be conditioned by the major options. In other words the major conditions are applied first, then the minor options are tested against the resulting records. Multiple MAJ and MIN requests may be made within one run. When more than one MAJ request is specified, the resultant subsetted data file is the union of all major conditions by ORing together all of the major requests to form the retrieval file.

Since the major options simply subset the data file, one must then specify a minor option defining the various parameters of interest. Though this may greatly reduce the amount of SMF data to be processed, a further reduction is accomplished by retrieving only that information from the remaining records that is desired whether or not they ultimately are used.

Subsequent passes will then process only the retrieved data greatly reducing the processing required. As a result, the actual SMF file indeed has only been passed once. During this pass a job table has also been created identifying all of the jobs necessary to meet the required request. In addition, separate data sets have been created containing the information retrieved from each of the SMF records required. At this stage each of these data sets is sorted and merged against the sorted job table. This reduces the data to only that which is necessary to fulfill the required conditions.

7. Implementation

As previously mentioned, to customize the SAS routine to enable flexibility for requesting any combination of the nine conditional parameters, the input requests are processed to generate SAS macros. In this way, only that code necessary to fulfill the given requests is generated. This further reduces processing costs from that of a generalized routine that would have to be concerned with all possible combinations. If any errors were detected in the input requests, these macros are not created and the run is aborted.

A few samples of the SAS macros generated from the user requests are shown below:

User Requests:

```
MIN DSN='SYS1.LIB01'
MIN DSNX='TOPS' & VOLSER='K19762'
MAJ DATE=80091
```

Macros:

```
MACRO  MAJSEL
*      MAJOR CONDITIONS;
IF(DATE=80091)
  ;%
MACRO  GP1CONO
*      MINOR CONDITIONS;
 %
MACRO  GP2CONO
*      MINOR CONDITIONS;
IF(DSN='SYS1.LIB01') & RQST=         1
  THEN OUTPUT GP2TABLE;
IF( POS ¬=   0&VOLSER='K19762') & RQST=         2
  THEN OUTPUT GP2TABLE;
 %
MACRO  MINALL
*      MINOR  DATE TIME AND JOB CONDITIONS;
%
MACRO  MINTYP4
*      MINOR  PROGRAM CONDITION;
%
MACRO  MINTYP5
*      MINOR  PROGRAMMER AND CHARGE CONDITIONS;
%
MACRO  MINTYP14
*      MINOR  INPUT VOLSER AND DSN CONDITIONS;
IF DSN='SYS1.LIB01' THEN DO;RQST=         1
;OUTPUT OUT14; OUTPUT JOBTABLE;END;
    PTR=INDEX(DSN,'TOPS');
  IF PTR¬=0 THEN DO; RQST=         2; OUTPUT JOBTABLE;
     POS=PTR;OUTPUT OUT14;END;
     IF VOLSER='K19762' THEN OUTPUT JOBTABLE;
    RQST=         2;
%
MACRO  MINTYP15
*      MINOR  OUTPUT VOLSER AND DSN CONDITIONS;
IF DSN='SYS1.LIB01' THEN DO;RQST=         1
;OUTPUT OUT15; OUTPUT JOBTABLE;END;
    PTR=INDEX(DSN,'TOPS');
  IF PTR¬=0 THEN DO; RQST=         2; OUTPUT JOBTABLE;
     POS=PTR;OUTPUT OUT15;END;
     IF VOLSER='K19762' THEN OUTPUT JOBTABLE;
    RQST=         2;
%
MACRO  MINTYP26
*      MINOR  INPUT REMOTE NO. CONDITION;
%
MACRO  MINTYP35
*      MINOR  TSO CHARGE NO. CONDITION;
%
MACRO  SORTYP
*      SORT PRINTOUT CONDITION;
      BY JOB DAY RECTIM;
 PROC PRINT; BY JOB DATE NOTSORTED; ID PGMR;
 TITLE SECURITY RETRIEVAL REPORT;
%
```

This request will retrieve all jobs run on day 91 which used a data set SYS1.LIB01 or had the character string TOPS somewhere in the data set name for data sets residing on volume K19762.

User Requests:

```
MIN PGM='UHPROGM'
MIN JOB='ACAPEM' & TIME>10.& TIME<12.
SRT 'JOB' ID 'DATE'
```

Macros:

```
MACRO  MAJSEL
*      MAJOR CONDITIONS;
 %
MACRO  GP1CONO
*      MINOR CONDITIONS;
IF(PGM='UHPROGM')
 THEN DO; RQST=          1; OUTPUT GP1TABLE; END;
IF(JOB='ACBPEM'&TIME>10.&TIME<12.0)
 THEN DO; RQST=          2; OUTPUT GP1TABLE; END;
 %
MACRO  GP2CONO
*      MINOR CONDITIONS;
 %
MACRO  MINALL
*      MINOR  DATE TIME AND JOB CONDITIONS;
     IF JOB='ACBPEM' THEN OUTPUT JOBTABLE;
     IF TIME>10. THEN OUTPUT JOBTABLE;
     IF TIME<12.0 THEN OUTPUT JOBTABLE;
%
MACRO  MINTYP4
*      MINOR  PROGRAM CONDITION;
     IF PGM='UHPROGM' THEN OUTPUT JOBTABLE;
%
MACRO  MINTYP5
*      MINOR  PROGRAMMER AND CHARGE CONDITIONS;
%
MACRO  MINTYP14
*      MINOR  INPUT VOLSER AND DSN CONDITIONS;
%
MACRO  MINTYP15
*      MINOR  OUTPUT VOLSER AND OSN CONDITIONS;
%
MACRO  MINTYP26
*      MINOR  INPUT REMOTE NO. CONDITION;
%
MACRO  MINTYP35
*      MINOR  TSO CHARGE NO. CONDITION;
%
MACRO  SORTYP
*      SORT PRINTOUT CONDITION;
 BY JOB  TIME RECTIM;
PROC PRINT; BY JOB  NOTSORTED; ID DATE;
 TITLE SECURITY RETRIEVAL REPORT;
%
```

Includes the SRT option which provides SORTing and report options, as discussed in the next section.

User Requests:

```
MIN RIN='R108.RD1' & PGMR ¬='SYSTEMS'
MIN CC='1233P' & DATE=80092
SRT 'JOB' ID 'RIN'
```

Macros:

```
MACRO  MAJSEL
*      MAJOR CONDITIONS;
 %
MACRO  GP1COND
*      MINOR CONDITIONS;
IF(RIN='R108.RD1'&POS= 0 & POS¬=.)
 THEN DO; RQST=          1; OUTPUT GP1TABLE; END;
IF(CC='1233P'&DATE=80092)
 THEN DO; RQST=          2; OUTPUT GP1TABLE; END;
 %
MACRO  GP2CONO
*      MINOR CONDITIONS;
 %
MACRO  MINALL
*      MINOR  DATE TIME AND JOB CONDITIONS;
     IF DATE=80092 THEN OUTPUT JOBTABLE;
%
MACRO  MINTYP4
*      MINOR  PROGRAM CONDITION;
%
MACRO  MINTYP5
*      MINOR  PROGRAMMER AND CHARGE CONDITIONS;
     POS=INDEX(PGMR,'SYSTEMS');
   IF POS= 0 THEN OUTPUT JOBTABLE;
   OUTPUT OUT5;
    IF CC='1233P' THEN OUTPUT JOBTABLE;
%
MACRO  MINTYP14
*      MINOR  INPUT VOLSER AND OSN CONDITIONS;
%
MACRO  MINTYP15
*      MINOR  OUTPUT VOLSER AND DSN CONDITIONS;
%
MACRO  MINTYP26
*      MINOR  INPUT REMOTE NO. CONDITION;
    IF RIN='R108.RD1' THEN OUTPUT JOBTABLE;
%
MACRO  MINTYP35
*      MINOR  TSO CHARGE NO. CONDITION;
    IF CC='1233P' THEN OUTPUT JOBTABLE;
%
MACRO  SORTYP
*      SORT PRINTOUT CONDITION;
 BY JOB  TIME RECTIM;
PROC PRINT; BY JOB  NOTSORTED; ID RIN;
 TITLE SECURITY RETRIEVAL REPORT;
%
```

8. Reporting Facility

Of major concern to the user, of course, is the end report. Here again, significant flexibility is desired to provide ease of analysis and ready observation. Though generally, job integrity is desired for reporting, sometimes it is desired to sort the resultant information on one or more of the conditional parameters. Furthermore, it would be helpful to readily separate major sorting breaks and be able to specify what variable to position in the left most column of the report. All of these functions are provided by the input selection type SRT. Along with the MAJ and MIN types, the user may specify a sort request indicating what conditions he desires as sorting parameters and the output variable desired in the left most column. For example:

```
SRT  'JOB,PGMR' ID 'DATE'
```

This request will cause the report to be sorted by job and programmer and provide major reporting breaks by these parameters. In addition the date will occur in the left most column. If no sort condition is specified the default is set to:

```
SRT  'JOB,DATE' ID 'PGMR'
```

No JCL or control cards are necessary to accomplish this as it is accomplished in the SAS routine by generating the proper macro resulting from the input specified. Samples of the output produced from the previous input requests follows:

Example 1.

User Requests:

```
     MIN DSN='SYS1.LIB01'
     MIN DSNX='TOPS'&VOLSER='K19762'
     MAJ DATE=80091
```

```
                    SECURITY RETRIEVAL REPORT                    8119 WEDNESDAY, JULY 30, 1980

---------- JOB=A4402PAD  DATE=80091 ----------

   PGMR        TIME     STEPND  CODE   PGM      STEP   CC     RIN      RDUT      DSN                          VOLSER   ID

INV TRIG EXTRACT  23.6033   1      0    DFSRRC00  GD   4422P  INTRDR                                                
                  23.6033   .      .                                            TOPS.A4403AAD.DG03AA.G0664V00  KT9476   I
                  23.6033   .      .                                            TOPS.A4402P.INVTRIG.G1501V00   K19762   D
                  23.6033   .      .                                            TOPS.PRDD.AC1DDRD              KDDI10   D
                  23.6033   .      .                                            TOPS.PRDD.AB1DCID              KDDI09   D
                  23.6033   .      .                                            TOPS.PRDD.AB2DCID              KDDI09   D
                  23.6033   .      .                                            TOPS.PRDD.AB3DCID              KDDI09   D
                  23.6033   .      .                          RI08.PRI

---------- JOB=A4402PBD  DATE=80091 ----------

   PGMR        TIME     STEPND  CODE   PGM      STEP   CC     RIN      RDUT      DSN                          VOLSER   IO

                23.6033   1      0    IERRCD00  STEP1                                                                  
                23.6033   .      .                                            TOPS.A4402P.INVTRIG.G1501V00   K19762   I
                23.6033   .      .                                            TOPS.A4402P.INVTRIG.G1501V00   K19762   I
                23.6033   2      0    DFSRRC00  STEP2                                                                  
                23.6033   .      .                                            TOPS.PRDD.AC1DDRD              KDDI10   D
                23.6033   .      .                                            TOPS.PRDD.AK1DTAB              KDDI13   D
                23.6033   .      .                                            TOPS.PRDD.AK3DTAB              KDDI13   D
```

Example 2.a

User Requests:

```
MIN PGM='UHPROGM'
MIN JOB='ACBPEM'&TIME>10.& TIME<12.
SRT 'JOB' ID 'DATE'
```

SECURITY RETRIEVAL REPORT 8:27 WEDNESDAY, JULY 30, 1980 2

JOB=ACBPEM

DATE	TIME	STEPND	CODE	PGM	STEP	PGMR	CC	RIN	ROUT	DSN	VOLSER	IO
80092	11.0294	1	0	IFEAAB	FOR	K CHOW	1S40U					
80092	11.0294	.	.							K.FORTMDD.LINKLIB	SYSPGP	I
80092	11.0294	2	0	IEWLF880	LKED	K CHDW	1S40U					
80092	11.0294	.	.							N.YZ.P1008S01.PSLIB1	KDD076	I
80092	11.0294	3	0	PGM=*.DD	GD	K CHOW	1S40U					

JOB=APLUSTOP

DATE	TIME	STEPNO	CODE	PGM	STEP	PGMR	CC	RIN	ROUT	DSN	VOLSER	IO
80092	0.311203	1	4	PLUSDA	PLUS	TOPS LIBRARIAN	4400M					
80092	0.311203	.	.							K.PROD.INFO	KRR017	I
80092	0.311203	.	.							KD89S.TOPS.PLUS.SOURCE	KRL001	O
80092	0.311203	.	.							KG89S.TOPSPLUS.DUMP.G0741V00	K124S6	O
80092	0.311203	2	4	UHPROGM	DELETE	TOPS LIBRARIAN	4400M			RM1.CLOAD	KRR017	I
80092	0.311203	.	.									
80092	0.311203	3	0	IEFBR14	ALLOCATE	TOPS LIBRARIAN	4400M			K.PROD.INFO	KRR017	I
80092	0.311203	4	0	A4400PM	STEP4	TOPS LIBRARIAN	4400M					
80092	0.311203	.	.							KG89S.TOPSPLUS.DUMP.G0741V00	K124S6	I
80092	0.311203	5	0	IEBUPDTE	IEBDTE	TOPS LIBRARIAN	4400M			TOPS.TEST.LOAD	KRS004	I
80092	0.311203	.	.									
80092	0.311203	.	.							TOPS.PLUSDA.XINC.SYSLIB	KRS002	I
80092	0.311203	.	.							TOPS.PLUSDA.XINC.SYSLIB	KRS002	O
									R108.PR1			

JOB=A1006DLY

DATE	TIME	STEPNO	CODE	PGM	STEP	PGMR	CC	RIN	ROUT	DSN	VOLSER	IO
80092	0.159S08	1	4	UHPROGM	STEP							
80092	0.159S08	2	0	IEBGENER	TPMK							
80092	0.159S08	.	.							K.TPMRK	KRS006	O
80092	0.159S08	3	0	IERRCOOO	STEP10							
80092	0.159S08	.	.							C.PROD.INFO	KRR017	I
80092	0.159S08	.	.							BB.A2013010.TAPE3.G1440V00	K15815	I
80092	0.159S08	.	.							BB.A1006020.TAPE3.G0054V00	K19118	I
80092	0.159S08	.	.							BB.A1006010.SORTOUT.G00S6V00	K11694	O
80092	0.159S08	4	0	IEBGENER	STEP1S							
80092	0.159S08	.	.							BB.A5012030.A1006PT1.TRIGGERS.G0001V00	KT8840	I
80092	0.159S08	.	.							BB.POPSCARD.TRIGS.G0191V00	KRS006	O
80092	0.159S08	.	.							SYS1.MODLIB	KDS132	I
80092	0.159S08	5	21760	SIMCHAN	STEP20							
80092	0.159S08	.	.							KL89S.A1400TST.LOADMODS	KRR003	I
80092	0.159S08	.	.							BB.POPSCARD.TRIGS.G0190V00	KRS010	I
80092	0.159S08	.	.							BB.A1006010.SORTOUT.G0056V00	K11694	I
80092	0.159S08	.	.							K.TPMRK	KRS006	I
80092	0.159S08	.	.							BB.A1006020.TAPE3.G0055V00	K11921	O
80092	0.159S08	.	.							BB.A1006020.TAPE4.G1199V00	K12001	O
80092	0.159S08	.	.							C.PROD.INFO	KRR017	I
80092	0.159S08	6	0	P1006CDN	STEP22							
80092	0.159S08	.	.							BB.A1006020.TAPE3.G0055V00	K11921	I

JOB=A1006DLY

DATE	TIME	STEPNO	CODE	PGM	STEP	PGMR	CC	RIN	ROUT	DSN	VOLSER	IO
80092	0.159508	.	.							K.TRIGGER	KRS001	O
80092	0.159508	.	.							SYS1.MOOLIB	KOS132	I
80092	0.159508	7	4	P1006025	STEP25							
80092	0.159508	.	.							BB.A1006020.RECYCLE.G0714V00	K18240	I
80092	0.159508	.	.							BB.A1006020.RECYCLE.G0715V00	K12839	O
80092	0.159508	.	.							K.TRIGGER	KRS001	I
80092	0.159508	.	.							SYS1.MDOLIB	KDS132	I
80092	0.159508	8	21920	SIM1460	STEP30							
80092	0.159508	.	.							SYS1.LIBI401	KRS007	I
80092	0.159508	.	.							BB.A6001201.DAILY.TRANS.CONSOL.G1455V00	K15549	I
80092	0.159508	.	.							BB.A1006030.TAPE3.G1203V00	K11675	O
80092	0.159508	.	.							BB.A1006030.PTCOUT.G0001V00	K11915	D
80092	0.159508	.	.							K.PTC5	KRS010	I
80092	0.159508	.	.							K.TAPE5	K11678	D
80092	0.159508	.	.							C.PROD.INFO	KRR017	I
80092	0.159508	9	0	IERRCO00	STEP31							
80092	0.159508	.	.							C.PROD.INFO	KRR017	I
80092	0.159508	.	.							K.TAPE5	K11678	I
80092	0.159508	.	.							K.TAPE6	K12140	D
80092	0.159508	10	0	89858829	STEP32							
80092	0.159508	.	.							K.TAPE6	K12140	I
80092	0.159508	.	.							SYS1.MDOLIB	KDS132	I
80092	0.159508	11	0	IERRCO00	STEP40							
80092	0.159508	.	.							C.PRDO.INFO	KRR017	I
80092	0.159508	.	.							BB.A1006030.PTCOUT.G0001V00	K11915	I
80092	0.159508	.	.							BB.A1006040.SRTPTC.G0001V00	K12691	D
80092	0.159508	12	21760	SIMCHAN	STEP60							
80092	0.159508	.	.							BB.A1006040.SRTPTC.G0001V00	K12691	I
80092	0.159508	.	.							BB.A1006060.PRCRE5.G1437V00	K21595	I
80092	0.159508	.	.							BB.A1006020.TAPE4.G1199V00	K12001	I
80092	0.159508	.	.							BB.A1006060.TAPE5.G1250V00	K19099	D
80092	0.159508	.	.							BB.A1006060.PRCRE5.G1438V00	K11555	D
80092	0.159508	.	.							TAPE6	WKTAPE	O
80092	0.159508	.	.							K.PU8PUNCH	KRS006	D
80092	0.159508	.	.							SYS1.LIB010	KRS007	I
80092	0.159508	.	.							C.PROO.INFO	KRR017	I
80092	0.159508	13	0	IE8GENER	COPY							
80092	0.159508	.	.							K.PU8PUNCH	KRS006	I
80092	0.159508	.	.							BB.A1006060.PU8PUNCH.G0039V00	K24522	D
80092	0.159508	.	.							BB.A1006030.TAPE3.G1203V00	K11675	I

JOB=A806R007

DATE	TIME	STEPNO	CODE	PGM	STEP	PGMR	CC	RIN	ROUT	DSN	VOLSER	IO
80092	12.0822	1	12	UHPROGM	UHPA95	FLEURANT	8068P					
80092	12.0822	2	0	AE8GENER	CDPYA95	FLEURANT	8068P					
80092	12.0822	.	.							NJ.A8068D.SA95LOC	KRS003	O
80092	12.0822	3	0	AE8GENER	53	FLEURANT	8068P					
80092	12.0822	.	.							NJ.A8068O.SA95LOC	KRS002	I

SECURITY RETRIEVAL REPORT 8:27 WEDNESDAY, JULY 30, 1980 4

JOB=A8567CDA

DATE	TIME	STEPNO	CODE	PGM	STEP	PGMR	CC	RIN	ROUT	DSN	VOLSER	IO
80092	0.339003	1	0	UHPROGM	UNCAT							
80092	0.339003	2	0	P8567041	P8567041							
80092	0.339003	.	.							KD893.A8567CDA.K8567040	KRR001	I
80092	0.339003	.	.							KD893.KCA7365.SUS74938.S7365D80	KDD061	O
80092	0.339003	.	.							K.SCDC.PRODLOAD	KPL002	I
80092	0.339003	3	0	IERRCO00	S8567008							
80092	0.339003	.	.							K.PROD.INFO	KRR017	I
80092	0.339003	.	.							KD893.A8567CDA.A8567040	KRS009	I
80092	0.339003	.	.							K.SCDC.PRODLOAD	KPL002	I
80092	0.339003	4	0	P8567P10	P8567P1							
80092	0.339003	.	.							KD893.P8567P10.REPORT	KDD061	O
80092	0.339003	.	.							KD893.P8567P10.ERRPRT	KDD057	O
80092	0.339003	.	.							K.SCDC.PRODLOAD	KPL002	I
80092	0.339003	5	0	P8567010	P856701							
80092	0.339003	.	.							KD893.P8567010.REPORT	KRR002	O
80092	0.339003	.	.							KG893.A8567CDA.B8567010.G0001V00	K19363	O
80092	0.339003	.	.							K.SCDC.PRODLOAD	KPL002	I
80092	0.339003	6	0	IERRCO00	S8567004							
80092	0.339003	.	.							K.PROD.INFO	KRR017	I
80092	0.339003	.	.							KG893.A8567CDA.T8567015.G3151V00	K13204	I
80092	0.339003	.	.							K.SCDC.PRODLOAD	KPL002	I
80092	0.339003	7	0	P8567005	P8567005							
80092	0.339003	.	.							KG893.A8567CDA.T8567005.G1235V00	K12465	O
80092	0.339003	.	.							KD893.P8567005.VALRPT	KRS004	O
80092	0.339003	.	.							KD893.P8567005.REJRPT	KRR001	O
80092	0.339003	.	.							KD893.P8567005.D8RPT	KRR001	O
80092	0.339003	.	.							K.SCDC.PRODLOAD	KPL002	I
80092	0.339003	8	0	P8567012	A8567012							
80092	0.339003	.	.							KG893.A8567CDA.T8567005.G1235V00	K12465	I
80092	0.339003	.	.							K.SCDC.PRODLOAD	KPL002	I
80092	0.339003	9	0	IERRCO00	S8567004							
80092	0.339003	.	.							K.PROD.INFO	KRR017	I
80092	0.339003	.	.							KG893.A8567CDA.T8567005.G1235V00	K12465	I
80092	0.339003	.	.							K.SCDC.PRODLOAD	KPL002	I

JOB=A8567TCD

DATE	TIME	STEPNO	CODE	PGM	STEP	PGMR	CC	RIN	ROUT	DSN	VOLSER	IO
80092	0.1064	1	4	UHPROGM	SCRATCH	DOUG BAXTER	8567P	R108.RD1				
80092	0.1064	2	0	P8567040	P8567040	DOUG BAXTER	8567P	R108.RD1				
80092	0.1064	.	.							KD893.A8567CDA.K8567040	KRR001	O
80092	0.1064	.	.							KD893.A8567CDA.L8567040	KDD057	O
80092	0.1064	.	.							TOPS.A4401AFD.A4401AF.G1995V00	K12847	I
80092	0.1064	.	.							TOPS.A4401AFD.A4401AF.G1994V00	KT2719	I
80092	0.1064	.	.							KG893.AA567CDA.A8567040.G1388V00	K12275	O
80092	0.1064	.	.							KD893.A8567CDA.A8567040	KRS009	O
80092	0.1064	.	.							TOPS.A4401AJD.A4401AJ.G1988V00	K11756	I
80092	0.1064	.	.							TOPS.A4401AJD.A4401AJ.G1987V00	KT2412	I
80092	0.1064	.	.							KG893.A8567CDA.B8567040.G1387V00	K12248	O
80092	0.1064	.	.							KD893.A8567CDA.B8567040	KRS001	O
80092	0.1064	.	.							TOPS.A4401AFD.A4401AF.G1996V00	K12888	O
80092	0.1064	.	.							TOPS.A4401AJD.A4401AJ.G1989V00	K12183	O
80092	0.1064	.	.							KD893.A8567CDA.K8567040	KRR001	O
80092	0.1064	.	.							KO893.KCA7365.SUS74938.S7365D80	KDD061	O
80092	0.1064	.	.							K.SCDC.PRODLOAD	KPL002	I
80092	0.1064	.	.					R108.PR1				
80092	0.1064	.	.					R108.PR1				

Example 3.a

User Requests:

```
MIN RIN='R108.RD1'&PGMR='SYSTEMS'
MIN CC='1233P'&DATE=80092
SRT 'JOB' ID 'RIN'
```

SECURITY RETRIEVAL REPORT — B135 WEDNESDAY, JULY 30, 1980 — 2

JOB=AAMJEN1A

RIN	TIME	DATE	STEPNO	CODE	PGM	STEP	PGMR	CC	ROUT	DSN	VOLSER	IO
	12.06S3	80092	6	0	POSTPRO	OUTPUT	A M JANESHEK	1233P		CP46.CC1980.GROUP.IPES03.MISCOATA	K00061	I
	12.06S3	80092	6	0	POSTPRO	OUTPUT	A M JANESHEK	1233P		CP46.CC1980.GROUP.IPES03.MISCOATA	K00061	I
	12.06S3	80092	.	.						CP46.CC1980.GROUP.IPES03.MISCOATA	K00061	I
	12.06S3	80092	.	.						CP46.CC1980.GROUP.IPES03.MISCOATA	K00061	I
	12.06S3	80092	.	.						CP46.CC1980.GROUP.IPES03.MISCOATA	K00061	I
	12.06S3	80092	.	.						CP46.CC1980.GROUP.IPES03.MISCOATA	K00061	I
	12.06S3	80092	.	.						CP46.CC1980.GROUP.IPES03.MISCOATA	K00061	I
	12.0653	80092	.	.						AAMJEN1.IPES.V4SPMB7.POF2	KRS010	I
	12.0653	80092	.	.						CP46.CC1980.GROUP.IPES03.PREPOST	KRR017	I

JOB=APLUSZZK

RIN	TIME	DATE	STEPNO	CODE	PGM	STEP	PGMR	CC	ROUT	DSN	VOLSER	IO
R108.RD1	0.047877B	80092	1	0	AEBGENER	OTHER	OFFSITE TAPES	0560M		KG89S.A0560LIB.PLUSOTH.G0836V00	K2n121	I
	0.047877B	80092	.	.						ZZK.KG89S.A0560LIB.PLUSOTH.G0166V00	K12020	O
R108.RD1	0.047877B	80092	2	0	AEBGENER	PL1	OFFSITE TAPES	0560M		KG89S.A0560LIB.PLUSPL1.G2487V00	K20089	I
	0.047877B	80092	.	.						ZZK.KG89S.A0560LIB.PLUSPL1.G0166V00	K12352	O
	0.047877B	80092	.	.						KG89S.A0560LIB.PLUSPL1.G2487V00	K20089	I
	0.047877B	80092	.	.						ZZK.KG89S.A0560LIB.PLUSPL1.G0166V00	K12352	O
R108.RD1	0.047877B	80092	3	0	AEBGENER	COBOL	OFFSITE TAPES	0560M		KG89S.A0560LIB.PLUSCOB.G1157V00	K19998	I
	0.047877B	80092	.	.						ZZK.KG89S.A0560LIB.PLUSCOB.G0166V00	K19244	O
R108.RD1	0.047877B	80092	4	0	AEBGENER	ARCHIVE	OFFSITE TAPES	0560M		KG89S.A0560LIB.PLUSARC.G0131V00	K13057	I
	0.047877B	80092	.	.						ZZK.KG89S.A0560LIB.PLUSARC.G0166V00	K11696	O
R108.RD1	0.047877B	80092	5	0	AEBGENER	AIOS	OFFSITE TAPES	0560M		KG89S.A0560LIB.PLUSAIO.G1375V00	K17927	I
	0.047877B	80092	.	.						ZZK.KG89S.A0560LIB.PLUSAIO.G0166V00	K12886	O
R108.RD1	0.047877B	80092	6	0	AEBGENER	PCSG	OFFSITE TAPES	0560M				
R108.RD1	0.047877B	80092	7	0	AEBGENER	TOPS	OFFSITE TAPES	0560M		KG89S.A0560LIB.PLUSTOPS.G0634V00	K14463	I
	0.047877B	80092	.	.						ZZK.KG89S.A0560LIB.PLUSTOPS.G0166V00	K12776	O
R108.RD1	0.047877B	80092	8	0	AEBGENER	SEQ	OFFSITE TAPES	0560M		KG810.GRNVL.PLUSSEQ.G1815V00	K12957	I
	0.047877B	80092	.	.						ZZK.KG89S.A0560LIB.PLUSSEQ.G0166V00	K12354	O
	0.047877B	80092	.	.								

R108.PR1

JOB=A7826W8K

RIN	TIME	DATE	STEPNO	CODE	PGM	STEP	PGMR	CC	ROUT	DSN	VOLSER	IO
R108.RD1	0.14S961	B0092	1	30	BACKUP	GO	DAN NEAL	4680P				
R108.RO1	0.14S961	B0092	1	30	BACKUP	GO	DAN NEAL	4680P				
	0.14S961	B0092	.	.						KGB9S.KSA2420.WHSE.BACKUP.G301SV00	K12360	O
	0.14S961	B0092	.	.						KINVD.BACKUP	K12360	D
	0.14S961	B0092	.	.						INVM.BACKUP	K12360	D
	0.14S961	B0092	.	.						INVD.BACKUP	K12360	O
	0.14S961	B0092	.	.						KSCHO.BACKUP	K12360	O
	0.14S961	B0092	.	.						JRSTOK.BACKUP	K12360	O
	0.14S961	B0092	.	.						LABTAB.BACKUP	K12360	O
	0.14S961	B0092	.	.						IPKERR.BACKUP	K12360	O
	0.14S961	B0092	.	.						KD89S.KSA2420.KINVOT	KRR004	O
	0.14S961	B0092	.	.						KOB9S.KSA2421.KEYTP	KRR004	O
	0.14S961	B0092	.	.						KO89S.KSA2406.KSCHO	KDD061	O
	0.14S961	B0092	.	.						KO89S.KSA2406.JRSTOK	KDD061	O
	0.14S961	B0092	.	.						KO89S.KSA2407.LABTAB	KDD061	O
	0.14S961	B0092	.	.						KD89S.KSA2420.IEPYRK	KDD061	O
	0.14S961	B0092	.	.						IEPYRK.BACKUP	K12360	O
	0.14S961	B0092	.	.						KOB9S.KSA2406.IPKERR	KDD061	O
	0.14S961	B0092	.	.						KD89S.KSA2421.INVMT	KDD061	O
	0.14S961	B0092	.	.						KDB9S.KSA2421.INVOT	KDD061	O
	0.14S961	B0092	.	.						KLB9S.KSAPOOL.THM.REPLN	KRR001	I
	0.14S961	B0092	.	.					R108.PR1			

JOB=A7B26WPL

RIN	TIME	DATE	STEPNO	CODE	PGM	STEP	PGMR	CC	ROUT	DSN	VOLSER	IO
R108.RD1	0.04S10A3	B0092	1	30	REPLN	STEP1	DAN NEAL	4680P				
	0.04S10A3	B0092	.	.						KDB9S.KSA2421.KEYTP	KRR004	O
	0.04S10A3	B0092	.	.						KDB9S.KSA2421.KEYTP	KRR004	O
	0.04S10A3	B0092	.	.						KDA9S.KSA2420.REMARK.TEMP	SCRT23	O
	0.04S10A3	B0092	.	.						TEMP7	KRS001	O
	0.04S10A3	B0092	.	.						KD89S.KSA2406.JRSTOK	KDD061	O
	0.04S10A3	B0092	.	.						KD89S.KSA2407.LABTAB	KDD061	O
	0.04S10A3	B0092	.	.						KDB9S.KSA2420.REMARK	KDD061	D
	0.04S10A3	B0092	.	.						KD89S.KSA2421.KEYTP	KRR004	O
	0.04S10A3	B0092	.	.						KDB9S.KSA2420.KINVDT	KRR004	D
	0.04S10A3	B0092	.	.						KO89S.KSA2420.KINVDT	KRR004	O
	0.04S10A3	B0092	.	.						KDA9S.KSA2421.KEYTP	KRR004	O
	0.04S10A3	B0092	.	.						KD89S.KSA2406.JRSTOK	KDD061	O
	0.04S10A3	B0092	.	.						KDA9S.KSA2407.LABTAB	KDD061	D
	0.04S10A3	B0092	.	.						KD89S.KSA2420.REMARK	KDD061	O
	0.04S10A3	B0092	.	.						TEMP7	KRS001	O
	0.04S10A3	B0092	.	.						KD89S.KSA2420.SCHD.JRSTOK	KRS002	O
	0.04S10A3	B0092	.	.						KOB9S.KSA2420.REJECT.ANALYS.MILLS	KDD061	O
	0.04S10A3	B0092	.	.						A072SP.KNDOS4	KDD061	O
	0.04S10A3	B0092	.	.						KDA9S.KSA2421.INVMT	KDD061	O
	0.04S10A3	B0092	.	.						KDA9S.KSA2421.INVDT	KDD061	O
	0.04S10A3	B0092	.	.						KDB9S.KSA2420.REMARK.TEMP	SCRT23	O
	0.04S10A3	B0092	.	.						KLB9S.KSAPOOL.THM.REPLN	KRR001	I
R108.RO1	0.04S10A3	B0092	2	0	IERRC000	STEP2	DAN NEAL	4680P				
	0.04S10A3	B0092	.	.						K.PROD.INFO	KRR017	I
	0.04S10A3	B0092	.	.						KDB9S.KSA2420.SCHD.JRSTOK	KRS002	I

SECURITY RETRIEVAL REPORT 8:35 WEDNESDAY, JULY 30, 1980 4

J08=A7826WPL

RIN	TIME	DATE	STEPNO	CODE	PGM	STEP	PGMR	CC	ROUT	DSN	VOLSER	IO
	0.0451083	80092	.	.						K0895.K5A2420.SCH0.OTSRT	KRS006	D
R108.R01	0.0451083	80092	3	30	SQRTOK	STEP3	DAN NEAL	4680P				
	0.0451083	80092	.	.						K0895.K5A2420.SCHD.OTSRT	KRS006	O
	0.0451083	80092	.	.						KL895.KSAPOOL.THM.REPLN	KRR001	I
R108.RD1	0.0451083	80092	4	0	IERRCOOO	STEP4	DAN NEAL	4680P				
	0.0451083	80092	.	.						K.PROD.INFO	KRR017	I
	0.0451083	80092	.	.						K0895.K5A2420.SCHD.JRSTOK	KRS002	I
	0.0451083	80092	.	.						K0895.K5A2420.SCH0.OTSRT	KRS006	O
R108.RD1	0.0451083	80092	5	30	SQRTOK	STEP5	DAN NEAL	4680P				
	0.0451083	80092	.	.						K0895.K5A2420.SCHD.OTSRT	KRS006	O
	0.0451083	80092	.	.						KL895.KSAPOOL.THM.REPLN	KRR001	I
R108.RD1	0.0451083	80092	6	0	IERRCOOO	STEP6	DAN NEAL	4680P				
	0.0451083	80092	.	.						K.PROD.INFO	KRR017	I
	0.0451083	80092	.	.						K0895.K5A2420.SCH0.JRSTOK	KRS002	I
	0.0451083	80092	.	.						K0895.K5A2420.SCHD.OTSRT	KRS006	O
R108.RD1	0.0451083	80092	7	30	SQRTOK	STEP7	DAN NEAL	4680P				
	0.0451083	80092	.	.						K0895.K5A2420.SCHD.OTSRT	KRS006	O
	0.0451083	80092	.	.						KL895.KSAPOOL.THM.REPLN	KRR001	I
R108.RD1	0.0451083	80092	8	30	ANALRQ	STEP8	DAN NEAL	4680P				
	0.0451083	80092	.	.						K0895.K5A2406.JRSTOK	KOD061	O
	0.0451083	80092	.	.						KD895.KSA2421.INVMT	KDD061	O
	0.0451083	80092	.	.						KL895.K5APOOL.THM.REPLN	KRR001	I
	0.0451083	80092	.	.					R108.PR1			
	0.0451083	80092	.	.					R108.PR1			
	0.0451083	80092	.	.					R108.PR1			
	0.0451083	80092	.	.					R108.PR1			

J08=AA567TCD

RIN	TIME	DATE	STEPNO	CODE	PGM	STEP	PGMR	CC	ROUT	OSN	VOLSER	IO
R108.RD1	0.1064	80092	1	4	UMPROGM	SCRATCH	DOUG BAXTER	8S67P				
R108.RD1	0.1064	80092	2	0	P8567040	P8567040	DOUG BAXTER	8567P				
	0.1064	80092	.	.						K0893.AA567CDA.K8567040	KRR001	O
	0.1064	80092	.	.						KD893.A8567CDA.L8S67040	KDD057	O
	0.1064	80092	.	.						TOPS.A4401AFO.A4401AF.G1995V00	K12847	I
	0.1064	80092	.	.						TOPS.A4401AFD.A4401AF.G1994V00	KT2719	I
	0.1064	80092	.	.						KG893.AAS67CDA.A8S67040.G1388V00	K12275	O
	0.1064	80092	.	.						KD893.A8567CDA.A8S67040	KRS009	O
	0.1064	80092	.	.						TOP5.A4401AJD.A4401AJ.G198RV00	K11756	I
	0.1064	80092	.	.						TOPS.A4401AJD.A4401AJ.G1987V00	KT2412	I
	0.1064	80092	.	.						KG893.AA567CDA.B8S67040.G1387V00	K12248	O
	0.1064	80092	.	.						K0893.A8S67CDA.88567040	KRS001	O
	0.1064	80092	.	.						TOPS.A4401AFD.A4401AF.G1996V00	K12888	O
	0.1064	80092	.	.						TOPS.A4401AJD.A4401AJ.G1989V00	K12183	O
	0.1064	80092	.	.						KD893.A8567CDA.K8S67040	KRR001	O
	0.1064	80092	.	.						KD893.KCA7365.SUS74938.S7365D8D	KDD061	O
	0.1064	80092	.	.						K.SCOC.PRODLOAD	KPL002	I
	0.1064	80092	.	.					R108.PR1			
	0.1064	80092	.	.					R108.PR1			

9. Conclusions

Frequently it is desired to analyze computer utilization to determine whether and/or to what extent security violation has occurred. This paper has described one approach to provide a flexible means of retrieving pertinent information and producing relevant reports to aid such evaluation.

Free form inputs of three types were discussed, to provide subsetting the original data base, sorting and formatting the output reports and specifying the retrieval conditions that must be met. These specifications may occur in any order with multiple requests permitted within a given run, except for the sorting type. Only one sort specification is permitted per run.

Any combination of nine retrieval parameters were outlined to use as search conditions to obtain the information desired. Once retrieved, the complete job profile is available for reporting according to the sort conditions specified. In addition to the usefulness relevant to security considerations, this facility has been applied profitably to aid in performance and billing problems.

10. Acknowledgments

I wish to acknowledge the Computer and Data Security Department for their contributions to the design and utilization feedback of this tracking capability. The efforts of R. I. Pettersen have been especially valuable in this regard providing early foresight for critical design specifications, as well as continuous discussion and recommendations during development and implementation stages.

CPEUG80

Software Improvements

Measuring Programming Productivity

Peter F. Zoll

Octopus Enterprises
Post Office Box 126
Geyserville, CA 95441

The paper begins by characterizing the current era as one of decreasing hardware costs and increasing software costs. These trends make the improvement of programmer productivity a critical consideration. The paper procedes with a review of the historical methods of measurement that use statistics such as programs, lines of code, limited lines of code, program correctness, data division lines plus verbs, and project control history. The consequences of imposing these various measures are considered from the aspects of generation of quality programs and impact on programmer morale.

The second section of the paper advances the position that available methodologies depend on the assumption that programmers should be evaluated on their ability to produce a program, as opposed to a product such as a report or a screen. An operationalist methodology suitable for metaprogramming is defined with the key constituents being a data elements dictionary and a set of software routines that generate sections of programs. The paper concludes with some suggested methods for measuring the efficiency of metaprograms' output as well as measuring the programmer's productivity in the new milieu.

Key words: Metaprogramming; operationalism; programmer measurement; programmer productivity.

1. The State of the Art

The introduction of the IBM 4300 ('E') series hardware and its plug-compatible competition has effectively quadrupled the data processing buyer's dollar in terms of equipment. This has been almost entirely offset by the more than doubling of the salaries of programming personnel in the last five years. Currently, in the San Francisco Bay area it is not uncommon to find trainees commanding twenty thousand dollars per year, with contract programmers engaged in nothing more complex than COBOL coding earning, if such is the word, sixty thousand dollars annually.

Monitoring facilities to report the increased productivity of the new hardware have grown in both number and sophistication. Alas, the quantitative and objective measurement of a programmer's productivity is still deficient. Historically, the first (and still most prevalent) programming measurement methodology employed is a carry-over from techniques used to manage some of the more prosaic occupations. Typically, the manager draws up a list of tasks, estimates the effort involved and assigns personnel. The programmers are evaluated on their ability to complete tasks on time and under budget. (1) While economy and timeliness are critical and praiseworthy goals, their attainment depends heavily on the

ability of the manager to estimate accurately while simultaneously assuring a quality product. Failure on the part of the manager often results in programmers working odd and long hours. This was of some short-term benefit to the company, largely because the projects were completed within bounds. The steadyerosion of trained personnel seems to be regarded as harmful in the long run. Worse, information systems implemented under the methodology discussed above can be readily indentified by some or all of the following features:

- o Little optimization of system resource consumption
- o Very conservative technology (to the point of obsolescence)
- o Numerous specification changes due to inadequate allocation of time or resources for design and analysis
- o A significant number of trivial programs
- o A substantial number of highly specialized programs
- o A high level of post-implementation maintenance
- o Coding that emphasizes expediency at the cost of quality

In response to a groundswell of complaint to the effect that a trivial program ought to be weighted less than a complex one, and that more eloquent (or verbose) coders should not be penalized, counting the lines of code was introduced as a measure of potential program goodness. Lines of code has become a fairly popular basis for productivity measurement.(2) Like many other common measures, such as IQ, the exact meaning of a line of code is not always clear. Because persons who administer and evaluate programmers tend to be paid more than programmers, the prospect of having a manager count lines of code does not have economic appeal (not to mention the understandable reluctance of the manager to engage in such a mundane activity). Accordingly, a number of attempts have been made to use the computer to count the lines of code. Presumably the irony of using the programmer's tool to measure the programmers had some appeal.

Those familiar with modern COBOL compilers will be quick to point out that the compiler in many cases provides the number of Data and Procedure Division statements. This means that the use of COPY statements, however commendable, skewed the results, The Remarks section of the Identification Division was also slighted, as were comments. Those installations fortunate enough to have a source library manager such as PANVALET or LIBRARIAN could consider using the library software to count the number of statements. The problem with this approach is that a line containing a trivial amount of code (such as 'ELSE') would be equal to a line containing 50 or 60 characters.

Two divergent schemes were developed to resolve this problem. The limited lines of code approach discards all lines that do not contain a certain minimum number of characters (with fifteen being a common choice). The second approach calculated a ratio between spaces and characters and termed this the program's density. It was asserted that program density correlated well with a programmer's efficiency. A fairly thorough search of the contemporary literature has failed to uncover any publications that vindicate this assertion. At any rate, several ingenious (if not wholly ethical) ideas were employed to counter the measurement programs. Blank cards, formerly used to offset paragraph names or comments, were changed to contain an asterisk in column 7. More extreme examples of this were the use of the literal SPACE or even asterisks in columns 7 through 71. An acquaintance of mine coded all elements of a structure in the general form of e-name, where name was the lowest level qualifier. Using library features he was able to change all es to a twenty character prefix. To add insult to injury, all the names were full qualified. Naturally, he sported an enviable efficiency rating and is now a manager.

The next improvement on the lines of code approach counted both the identification and Environment Division statements in addition to comments, referenced Data Division statements and Procedure Division verbs. The qualification of referenced Data Division statements was deemed necessary to prevent the inclusion of superfluous COPY statements (or their equivalent). This methodology was found to discourage many aspects of structured programming. Database programs were favored, while the results on Report Writer and indexing were mixed. Work on a compensation factor relating bytes of object code produced per verb was abandonned in the face of generally fruitless results.

Scarred veterans of the programming wars will be certain to point out that all the counting of lines is no substitute for having the manager or chief programmer inspect the code. Worse yet, the counter is forced into an ongoing guerilla warfare with the countees (who, in their worst moments, have been known to take delight in

altering the counting program). Finally, the lines of code counting fails to observe the most fundamental law fo programming -- any program that works is better than a program that does not. Lines of code counting does not assure that a program will work, as the (necessary) advent of structured tactics shows; it merely insures that a large number of lines of code will be produced. The most shattering blow of all comes from users. It seems that they are, as a group, more aware of and concerned with reports, screens and so forth, than they are with how many lines of code were produced. In short, lines of code counting does not distinguish between a product and a process.(3)

2. Operationalist Metaprogramming

A recent publication indicates that any hopes for a high improvement in programmer productivity are likely ill-founded.(4) If the traditional measures and concepts are retained, or even adjusted by error refinements (5), this gloomy conclusion is still likely to hold. One may, however, take some solace in recent thoughts by Gerald M. Weinberg (6) to the effect that unusual tasks or work in programming are quite rare. Accordingly, an operationalist approach involving metaprograms may be of some promise. Many metaprograms exist today. The most common example is the commercially available report writers. Here all a user (programmer or not) need do is to define the data elements and relationships, and reports may be generated with relative ease. Operationalist metaprograms exists today to extend this function to cathode ray tube screens and transaction editting. The user need only declare the class of screen element (literal, database element, program variable) and define its position on the screen. The attendant MFS macros (in the case of IMS/DC) are then generated automatically. These are accompanied by appropriate Working Storage and Procedure Division statements. In the case of a transaction, the user need only define the input format and the target database elements. This process of metaprogramming does indeed add yet another language to the Babel (7) existing today. The reduction in monotony and increase in productivity seems to be adequate compensation.

The programmer is then measures on his or her ability to design a database system. The production of file definitions and data elements dictionary entries becomes the critical measure. It can, of course, be abetted by the measurement of additions, deletions and changes. These measures are readily derived from the data elements dictionary. The routine audits now performed by the database administrator to determine aspects of element access and usage can also be used to measure the goodness of the design and the thoroughness of the analysis.

The change is perspective from lines of code and other measures which are used to monitor the clerical functions of computer programming to inspection of design in the milieu of operationalist metaprogramming requires some reorientation. It implies a higher usage of the hardware to reduce the non-creative tasks associated with information processing. This has been a trend throughout the brief history of computing. Programmers will have the option of becoming either metaprogrammers (writing the routines that generate the code) or into designer-analysts. The former will still be aware of and concerned with the reasonable use of hardware, while the latter will gain more ability to aid the organization in its imposition of computer-based information processing and structures on the less desciplined areas of the business environment.

References

(1) Comper, F.A., Project Management for System Quality and Development Productivity Proceedings of the Application Development Symposium.

(2) Evans, B., speech at the 10th Conference on Computer Audit Control and Security

(3) Bradshaw, W.R., 'Application Development Productivity Within IBM Information Systems' Proceedings of the Application Development Symposium

(4) Jones, T.C., 'The Limits of Programming Productivity' Proceedings of the Application Development Symposium

(5) Albrecht, A.J., 'Measuring Application Development Productivity' Proceedings of the Application Development Symposium

(6) Weinberg, G.M., 'The Psychology of Change in Development Methodology' Proceedings of the Applications Development Symposium

(7) Ehrman, J.R., 'The Babel of Application Development Tools' Proceedings of the Application Development Symposium

Also of interest:

Boehm, B.W. et. al, 'Quantitative Evaluation of Software Quality' Proceedings of the Second International Conference on Software Engineering

Jones, T.C., 'Measuring Programming Quality and Productivity' IBM Systems Journel, Volume 17, Number 1, 1978 also in Proceedings of GUIDE 45

Kendall, R.C., 'Management Perspectives on Programs, Programming and Productivity' Proceedings of GUIDE 45

Comparative Performance of COBOL vs. PL/I Programs

Paul J. Jalics

Cleveland State University

Abstract

The comparative performance of COBOL versus PL/1 programs on the IBM/370 is studied on the basis of a substantive benchmark test which has been implemented in both languages. The benchmark program was written so as to use identical data-types and language facilities where they exist, and close approximations elsewhere. Measurement results from each of eleven atomic tests are then used to gain insight into the relative merits of code generated for each language in that particular area. A number of surprising results were found, which in turn necessitated the running of additional experiments to explain the results. These, then, all contribute to give substantial insights into the relative performance of COBOL versus PL/1 Programs.

Introduction

The comparative execution time performance of COBOL versus PL/1 programs is examined on the basis of a substantive benchmark test which has been implemented in both languages. The original benchmark was written in COBOL and is called BNC1. Subsequently, three PL/1 versions were created: BNC6 is the original PL/1 translation with COBOL numeric COMP-3 items being interpreted as PIC '9' items, with PERFORM's being interpreted as PROCEDURE calls; BNC7 which interprets COBOL numeric COMP-3 items as FIXED DECIMAL and enumerates multiple occurrences of sections of code PERFORMed in COBOL rather than use PROCEDUREs; and finally BNC8 which is similar to BNC7 but uses PROCEDURE calls to implement PERFORM. BNC6 can be thought of as a conservative COBOL programmer's translation and BNC7 and BNC8 as more favorable translations for PL/1.

The benchmark breaks down into 11 separate tests each of which is timed separately using an assembler subroutine called XTIME which returns the CPU execution time in units of 10 milliseconds (0.01 sec=1 time unit). There is a repetition factor for each of the 11 tests which is set so as to give the proportion of each category of tests to reflect actual business programming usage (as interpreted by the benchmark's designer). We shall not be concerned with the weights and proportions given each test here, however, since we want to focus our attention on the relative performance of COBOL vs. PL/1 in each of the 11 test categories separately.

Measured Data

The following table summarizes the results for each of the benchmarks run on an IBM 370/158 with maximum Compiler optimization (using VS/COBOL release 2.2 and the PL/1 Optimizing Compiler Version I Release 3.0):

Test Number	Test Description	BNC1(COBOL) CPU time	BNC6(PL/1) CPU time	BNC7(PL/1) CPU time	BNC8(PL/1) CPU time
1	Internal Subroutine Call	12	143	138	146
2	GO TO, GO TO Computed	3	3	2	3
3	COMP-3 or decimal arith	42	160	107	114
4	Binary or COMP arithmetic	9	17	28	29
5	Character MOVE's	120	86	82	80
6	Compare's	36	116	141	77
7	Table Indexing	1	2	1	1
8	Numeric Editing of Decimal	17	46	32	35
9	Numeric editing of binary	22	39	44	45
10	INSPECT,char search & replace	78	14	15	14
11	Table Search	47	32	29	33
12	TOTAL	387	658	619	577

A detailed description of each of the tests and the results obtained for each now follows:

Test 1: Internal Subroutine Call

This test was originally designed to test the speed of execution of the internal subroutine facility in COBOL, namely the PERFORM statement. A sequence of nested PERFORM's is executed; the nesting is three levels deep and the first two levels contain TIMES options for looping. In PL/1 the simplest way to implement this is via PROCEDURE calls without parameters. The looping in the first two levels is done with DO loops.

This test shows the most dramatic differences between COBOL and PL/1 with COBOL being 12 times faster than PL/1. The PERFORM in COBOL is implemented very efficiently whereas a PROCEDURE call is by its nature more complex with dynamically allocated local storage, etc.

One can gain some insight into the factor 12 difference by looking at the table below which outlines typical overhead of the various subroutine mechanisms (as they are used in this benchmark):

COBOL	PL/1
simple PERFORM 6 instructions to get there 2 instructions to return	simple CALL 23 instructions to get there (+possible library call to allocate more memory space) 5 instructions to return
PERFORM UNTIL (with simple condition) 6 instructions to get there +11 instructions on each looping	DO I=1 TO N BY 1; 11 instructions to get there 12 instructions on each looping
PERFORM TIMES 11 instructions to get there 11 instructions on each looping	DO I=1 TO N BY 1; CALL SUB; END; 34 instructions to setup & get there (+possible library call to allocate more memory space) 17 instructions on each looping

Test 2: GO TO, Computed GO TO

This test has a large number of GO TO statements and one computed GO TO per execution of the test (test is repeated 35 times). A glance at the code generated and the numeric result indicates that both COBOL and PL/1 perform quite well in this area and the code generated is very similar.

Test 3: Decimal Arithmetic

This test consists of arithmetic (add, subtract, multiply, divide) to data-fields of various sizes with overflow detection and rounding on 27% of operations, overflow detection on 9% of the operations. This test is implemented via COMP-3 packed decimal data-fields in BNC1. IN BNC6, it was implemented via PICTURE '9' variables and in BNC7 and BNC8 it was implemented with FIXED DECIMAL. Thus the results of BNC1, BNC7, and BNC8 are easily comparable but BNC6 is not since it uses another data-type.

The results of this fairly important test were puzzling and a good number of additional experiments had to be devised to adequately explain the results.

Since BNC1, BNC7 and BNC8 all use the same data-types, it was incredible to find that BNC1 was 2.7 times faster than the PL/1 versions. Looking at the code itself did not give great insight into the performance differences: BNC1 was slightly smaller with 119 instructions total vs. 146 for PL/1. The COBOL code looked like quite a different style from PL/1 for identical operations but the types of instructions were similar. In fact, the lengths of most of the temporary data-fields involved in the computations for PL/1 were often smaller than for COBOL. Rounding seemed to be handled more efficiently in COBOL (4 instructions vs. 7 in PL/1). The use of REMAINDER was more efficient in COBOL whereas it had to be computed with additional arithmetic in PL/1. Statements relating to ON SIZE, ROUNDING, and REMAINDER were temporarily deleted from the test to see if that would explain the huge performance factor of 2.7. Once this was done, the number instructions in COBOL was reduced to 84 and in PL/1 to 85 instructions. But the CPU time did not change appreciably.

Finally, the culprit was found to be the PL/1 error interrupt handler. PL/1 always runs with all hardware overflow interrupts enabled. If a certain abnormal condition occurs, a machine interrupt takes place and the PL/1 error handler is given control. Then if that particular condition is enabled by the PL/1 programmer, the appropriate ON statement(s) are executed. If the condition is not enabled, then the error handler still gets invoked but he quickly decides to ignore the condition. Thus, when the ON SIZE statements were removed as described in the paragraph above, the overflows still occurred and the excessive performance factor of 2.7 did not improve. Finally, the statements causing overflows were removed (10% of arithmetic statements) the CPU times for both COBOL and PL/1 settled down to about 41 time units.

The main conclusion from the above is that PL/1 error interrupts are quite time consuming with a minimum of about 500 microseconds of CPU time per such interrupt. Moreover, it doesn't really matter whether the interrupt is to be ignored by the program or acted upon; the overhead has to be paid regardless.

The big question that remains is: Why wasn't this same overhead observed in COBOL. COBOL, in general, tends to be more lax about overflow and other arithmetic disasters: they are ignored in the usual case. Thus COBOL usually runs with the hardware conditions disabled. In looking at the code generated for the ON SIZE statements, they were detected by doing the arithmetic with a larger number of digits in temporary storage and then additional instructions were generated to detect non-zero values in the upper digits. This contrasts sharply with PL/1 where one does not see any additional code to detect overflow: it is all done by the hardware (COBOL generates an additional 11 instructions to detect the overflow in a typical case).

An additional experiment was carried out to be able to compare the test 3 results for BNC6 (uses PICTURE '9') to BNC1 results. BNC1 was temporarily changed to use USAGE IS DISPLAY which would then be comparable directly to the PICTURE '9' data types of BCN6. Results of this modified BNC1 showed a CPU time of 39 units, which is slightly less than the 42 gotten originally for BNC1 using COMP-3 data-type. This slight decrease was quite surprising when a substantial increase was expected since DISPLAY data types are first PACKED into COMP-3 (packed decimal) then the arithmetic is done, and finally the results are UNPACKed into DISPLAY format. Thus it is surprising that with all the extra work to be done, the CPU time is smaller than for COMP-3 where the arithmetic can be done directly. Previous measurements have indicated that COMP-3 arithmetic is twice as fast as DISPLAY[1].

Looking at the generated code, it became clear that the extra PACK and UNPACK operations were also used to move the data to the temporary workspace and back (this was done with costly ZAP's in the original BNC1). The comparison with BNC6 still didn't work out well because of the error interrupt problem noted earlier.

In summary, the decimal test gave a number of interesting insights into COBOL vs. PL/1 differences. The code generated by PL/1 appears to be just as good as COBOL in most cases (exception: ROUNDING, REMAINDER) but the style of code generation is quite different (for example: a simple add A=B+C generates the sequence of instructions ZAP, AP, MVO, ZAP in COBOL and MVC, AP, MVO, MVN in PL/1). The most dramatic discovery is the very high cost of program interrupts in PL/1 regardless of whether the interrupt is to be ignored or acted upon. This, then, is in sharp contrast to COBOL error handling which relies more on additional instructions generated to detect the error condition. Thus using ON SIZE in COBOL can be done economically whereas in PL/1 it is to be avoided in critical sections of code. A final observation, then, was that COMP-3 and DISPLAY variables performed equally well whereas COMP-3 is normally expected to be twice as fast as DISPLAY.

Test 4: Binary or COMP Arithmetic

This test is identical to test 3 which we have just examined except that COMP data-type is used in COBOL and FIXED DECIMAL for BNC6 and FIXED BINARY for BNC7 and BNC8. The CPU times are also not directly comparable with test 3 since the number of repetitions of the test is smaller than for test 3 (650 repetitions for test 3 and only 100 repetitions for test 4).

The same curious effect showed up here as was observed for the decimal test (test 3). The PL/1 CPU times are 3.2 times as much as the COBOL (ignoring BNC6 which is not a comparable data-type). The number of machin instructions generated is very similar for both COBOL and PL/1. Again the culprit is the error interrupt handler which was invoked a number of times (some are caused by the implementation limit of 32 bit in PL/1 vs. 64 bit limit in COBOL where the benchmark originated). By eliminating the statements causing overflow the CPU times were reduced in COBOL to 6 time units and in PL/1 to 3 time units. So PL/1 is actually significantl faster than COBOL for the binary fields of the mix in this benchmark. The reasons for this include the tendency of COBOL to use more library routines for partial word multiplications and division; also more of the PL/ code is done in binary whereas the COBOL code frequently converts operands to packed decimal, presumably to get proper decimal truncation (since we specify COBOL COMP fields in decimal digit ranges vs. bits in PL/1).

We can compare CPU times with test 3 by multiplying the CPU times for test 4 by 6.5. Thus in COBOL the 9 unit result multiplied by 6.5 becomes 58.5 which is about 50% higher than for COMP-3 fields in test 3. This is somewhat surprising since COMP is commonly twice as fast as COMP-3. The explanation for this is that half of the data-fields are over 10 digits where COMP fields are by far the most inefficient of all.

Test 5: MOVE Test

This test consists of MOVE's of characte fields which vary in size from 1 to 1024

bytes. The sending and receiving fields always have the same length.

This is one of the few tests where PL/1 clearly outshines COBOL, rather surprisingly since COBOL is supposed to be specializing in fixed length character manipulations. One big reason why PL/1 looks better here than it might in real-life situations is that the MOVE's are all of successive fields of one record to successive fields in another record and PL/1 is smart enough to recognize this and instead of generating an MVC for every MOVE, it generates one large MVC for up to 12 individual MOVE's.

This test was repeated with PICTURE 'X' variables in PL/1 as well as CHARACTER variables and the performance results are identical.

In summary, although PL/1 looks better on this test, the code generated and the performance of MOVE operations on character variables is likely to be identical for both COBOL and PL/1.

Test 6: Compare Test

This test does a sequence of compares of various data-fields of numeric as well as character type and of various sizes. The results show COBOL to be at least twice as fast as any of the PL/1's. The great variation among the PL/1 versions deals with the specific data-types of the variables which are different for BNC6, BNC7, and BNC8 in a number of cases. COBOL gets some advantages from its additional language facilities like testing NUMERIC and ALPHABETIC which generate Translate and Test machine instructions rather than the much more overhead of testing it manually in PL/1 (involving multiple tests). The code generated for 88 level condition-names is very convenient for the programmer but the code generated does not seem any more efficient than the corresponding manually constructed sequence in COBOL or PL/1.

Most of the IF statements generate a similar number of machine instructions in both COBOL and PL/1 and the factor of 2 performance appears to be due strictly to the advantages gained in COBOL class tests for NUMERIC and ALPHABETIC.

Test 7: Indexing Test

This test was intended to measure the speed of execution of USAGE is INDEX data-fields in COBOL. INDEX is a rather unique feature of COBOL in which the COBOL programmer SET's the "subscript" which he will later use to access an element of a table. The actual value of the index is the byte offset which points to the chosen element. As a result each manipulation of the INDEX variable requires a multiplication by the element size. Thus the COBOL code is twice as large as the PL/1 which has no equivalent language feature but instead does analogues operations on "subscripts" which have a data-type identical to INDEX variables.

One would expect the CPU time for COBOL to be twice as much as for PL/1. This is not observed, instead the results are about the same and fluctuate quite a bit for separate runs, probably indicating that the total time is so little in relation to our time unit that we are not getting sufficient accuracy.

Test 8: Editing of Decimal Numerica Data

This test involves taking the numeric results calculated in test 3 and moving them to edited fields so that they can be printed in a "pretty" format such as might be expected in a business report. Both COBOL and PL/1 accomplish this editing with editing PICTURE's like PIC Z, $, B etc. Here again, BNC1 can be compared with BNC7 and BNC8 but not with BNC6 which uses PICTURE '9' for the numeric fields rather than COMP-3 (or FIXED DECIMAL for BNC7, BNC8).

The code generated looks about the same for many of the edit moves and in fact COBOL often generates a few more instructions than PL/1 where they are not identical (example: COBOL 9, PL/1 6). In spite of this, the CPU time for COBOL is typically only a half of PL/1 execution time. This can be attributed directly to the fact that PL/1 calls library routines in 3 out of the 17 edit moves whereas COBOL generates in-line code for all of them. When the three statements generating library calls were removed, CPU time went down drastically from 39 units to 9 units which would make the remaining somewhat faster than COBOL.

In summary, the COBOL editing of decimal numeric data is about the same or slightly worse than corresponding PL/1 editing for the cases where PL/1 generates in-line code. For the remainder, PL/1 performs much worse. Thus PL/1 code generation in this area is less polished.

Test 9: Editing of Binary Numeric Data

This test does editing identical to test

8 above except that the source field data-items are binary results calculated from test 4.

Results here are somewhat analogous to test 8 above: COBOL editing is twice as fast as PL/1. The size of the code generated is about the same but both COBOL and PL/1 each call 3 library routines. Two out of the three library calls are for different move edit source statements in COBOL vs. PL/1. It is again clear that the factor of 2 in performance for PL/1 comes from the library calls because when they are removed, CPU time drops from 45 to 17 units.

Test 10: INSPECT verb:

Character Search and Replacement

This test consists of scanning a character field of 40 characters in five different INSPECT statements some of which scan in testing for certain characters, others count the number of occurrences of certain characters, while others do selective replacement of characters. Since PL/1 has no facilities like INSPECT, loops are setup to perform identical manipulations on identical size and type data-fields.

Somewhat surprisingly, COBOL is 4 to 5 times as slow as the equivalent PL/1 code. Looking at the code will very quickly explain why this is so: COBOL calls a library routine to implement each of the INSPECTS. The problem surely comes from the fact that INSPECT is such a general statement with so many possible combinations of functions that the library routine which implements it must be very generalized and parameterized and therefore probably not the most efficient. The parameter list for this library call is so complicated that it takes more instructions in COBOL to prepare the parameter list (110 instructions total) than to perform all of the actions necessary in PL/1 (90 instructions total). This difference can easily account for the factor of 4 to 5 difference in performance.

Test 11: Table Searching

This test searches a table with 30 elements in six different ways, three times using a linear SEARCH, the other 3 being a binary SEARCH ALL. Since neither the SEARCH nor the SEARCH ALL is provided in the PL/1 language, manual code was written to perform identical functions. In the PL/1 code each of the 6 table searches is done via linear searching and since earlier measurements [1] on COBOL indicated that the threshold at which SEARCH ALL becomes more efficient is with tables of at least 50 elements, this should not prejudice the results substantially.

The results indicate that PL/1 code is about 50% faster than the COBOL code. The PL/1 code consists of 102 instructions total versus 142 for COBOL. In addition, COBOL generates a library call for SEARCH ALL whereas all of the PL/1 code is in-line. Certainly the results would not be so favorable to PL/1 with a larger table but then one could without much difficulty write search loops that implement a binary search (SEARCH ALL).

Conclusions

Certainly one cannot expect to get a complete overview of the execution efficiency of two extensive programming languages like COBOL and PL/1 from any one benchmark. The benchmark used is impressively good, however, and serves to give substantial insights into the performance aspects of COBOL and PL/1 programs on the IBM/370 (excluding input/output statement performance).

The following points summarize the observations:

1. Except for the overhead of the PL/1 PROCEDURE call which is substantial, there appears to be no basic grounds why PL/1 programs have to be any less efficient in execution than corresponding COBOL programs.

2. PL/1 performance depends a great deal on the choices made by the programmer in selecting language features, data-types, etc. Thus considerable care needs to be taken if one is to achieve a performance level as good as COBOL. Without such care, it is very easy to get a PL/1 program which runs two to three times as slow as a similar COBOL program.

3. PL/1 code generation is not as good as COBOL in a few areas like editing of numeric data-items and partial-word binary arithmetic among others. The code size is, in general, comparable but more use is made of library calls than was observed for corresponding COBOL programs.

4. PL/1 error handling has a tremendously higher overhead than COBOL in the cases observed, simply because PL/1 makes more use of hardware interrupts to signal such conditions whereas COBOL generates in-line code to identify such situations. Also, it is dangerous thing to specify that error interrupts are to be ignored since the programmer will never know that such errors occur on a regular basis (they should be eliminated instead by additional checking) and serve to degrade performance a great deal.

References

[1] Jalics, P. J., "Benchmarks for Measuring COBOL Implementation Performance", Data Management, Vol. 18, No. 4, April 1980.

[2] Jalics, P. J., "Improving Performance the Easy Way", DATAMATION, April 1977.

[3] Jalics, P. J., "Comparative Performance of COBOL Programs on Mini vs. Large Computer Systems", Proceedings of the First Annual Symposium on Small Systems, New York, N. Y., August 1978 (ACM/SIGSMALL).

CPEUG80

Data Communications

NBS Network Measurement Methodology Applied to Synchronous Communications

Marshall D. Abrams

Institute for Computer Sciences and Technology
National Bureau of Standards
Washington, D. C. 20234

Dorothy C. Neiman

Commtex, Inc.
2411 Crofton Lane
Crofton, Maryland 21114

This paper focuses on the application of the NBS Network Measurement Instrument (NMI) to synchronous data communication. The suitability of the underlying Stimulus - Acknowledgement - Response (SAR) model to support the implementation of this methodology permitting quantitative evaluation of interactive teleprocessing service delivered to the user is described. The logic necessary to interpret SAR components and boundaries depends on character time sequence for asynchronous data communications traffic but entails protocol decomposition and content analysis for character synchronous data traffic. The decomposition and analysis rules necessary to evaluate synchronous communications are discussed and the level of protocol violation detection which results as a byproduct is cited. Extensions to the utility of the Network Measurement Instrument (NMI), deriving from additional workload profiling measures desirable for character synchronous communications, are also presented.

Key Words: Data communications; protocol validation; synchronous; teleprocessing service evaluation.

1.0 INTRODUCTION

In 1979 the National Bureau of Standards contracted with Commtex for the incorporation of the NBS Network Measurement System [1] methodology into a portable Network Measurement Instrument (NMI) [3]. One part of Commtex's implementation entailed extending the NBS concept of performance measurement and evaluation to deal with character synchronous data communication protocols.

This paper focuses on the comparison of data gathering for synchronous and asynchronous data communication and the suitability of the underlying Stimulus - Acknowledgement - Response model to both methods.

2.0 THE STIMULUS ACKNOWLEDGEMENT RESPONSE MODEL

The NBS approach to the measurement of interactive computing focuses on the qualitative and quantitative aspects of the service provided to the terminal user. The assessment of service delivered by such an interactive teleprocessing system is accomplished through the Stimulus - Acknowledgement - Response (SAR) model [2] which views user-network communications as a sequence of transactions composed of a stimulus, and the consequential acknowledgement and/or response from the network/system. Recognition of SAR components from data traffic transmissions requires specific interpretation logic for each transmission mode and protocol to be analyzed.

In order to produce measurement results consistent with subjective human judgment of system performance, it is necessary to differentiate operational feedback (acknowledgement) from information transfer (response) within network output to the user.

2.1 Asynchronous Communications Characteristics

In asynchronous communications the acknowledgement is defined as a fixed header, a known string of characters which occurs at the beginning of data transmission from the network, and all non-printing characters which immediately precede or follow the header. In the absence of a fixed header the acknowledgement consists only of the non-printing characters at the beginning of network output.

3.0 CHARACTERISTICS UNIQUE TO SYNCHRONOUS COMMUNICATION

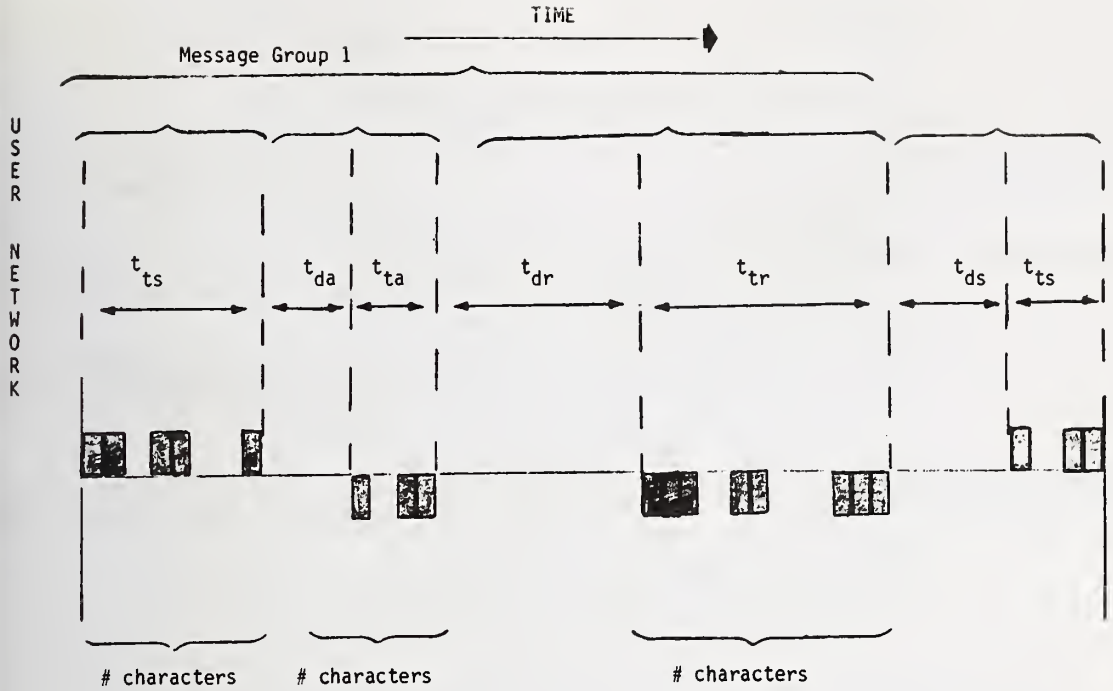
Unlike asynchronous communications where the unit of exchange between user and network is a character, synchronous communications protocols assemble multiple characters into transmission blocks which are transmitted at the maximum network transmission rate. Only when acknowledgement-response boundaries are coincident with transmission unit boundaries is the psychological function of acknowledgement fulfilled. For synchronous communications, acknowledgement is therefore defined as all transmission units containing a fixed header which occur at the beginning of data transmission from the network. A synchronous transmission unit can be identified as acknowledgement not only as a function of content but also as an occurrence in time. In IBM's binary synchronous communications for example, the "limited conversational mode" allows for the substitution of a text message for the link level acknowledgement to user stimulus. This message which often contains format control data is accepted as an acknowledgement.

Although the NMI monitors all communications on the circuit under test, the link control sequences which constitute overhead to the user are not defined as SAR components. An argument can be made for the differentiation of information content and link level control sequences within information transmission blocks; however, accurate information identification would require recognition of the various levels of application dependent protocol. Since the NMI generally strives to be application independent, this type of analysis is not a standard measurement feature and must be accomplished uniquely for given applications.

3.1 Measurement of SAR Model Parameters

Interactive communications are described by the SAR model in terms of the number of characters, transmission time, and delay time associated with each SAR component. Stimulus delay time, for example, equivalent to user think time, is directly measurable by the NMI in asynchronous communications as illustrated in Figure 1. Response time, defined as the elapsed time from the input of the last character of the stimulus until the receipt of the first character of response, is simply derived from the measured variables as follows:

RESPONSE TIME =
ACKNOWLEDGEMENT DELAY TIME +
ACKNOWLEDGEMENT TRANSMISSION TIME + RESPONSE DELAY TIME



t_{ts} = stimulus transmission time

t_{da} = acknowledgement delay time

t_{ta} = acknowledgement transmission time

t_{dr} = response delay time

t_{tr} = response transmission time

t_{ds} = stimulus delay time

Figure 1. Asynchronous SAR Model

Although the SAR model is equally applicable to synchronous communications as to asynchronous communications, measurement of the model parameters is complicated by the local processing and buffering characteristic of synchronous terminals. Because input usually is not buffered at an asynchronous terminal, what is measured as stimulus transmission time corresponds to operator typing time. As illustrated below, several of the important measures must be estimated, rather than measured, when dealing with synchronous communication.

Figure 2 illustrates the SAR mapping to synchronous communications. Input is traditionally buffered in synchronous communications, and its presence is not detectable at the circuit interface prior to transmission. As a result the NMI is unable to measure operator typing time and stimulus waiting time. The stimulus only becomes available for transmission when the operator presses the "enter" key; it is not actually transmitted until the terminal is next polled.

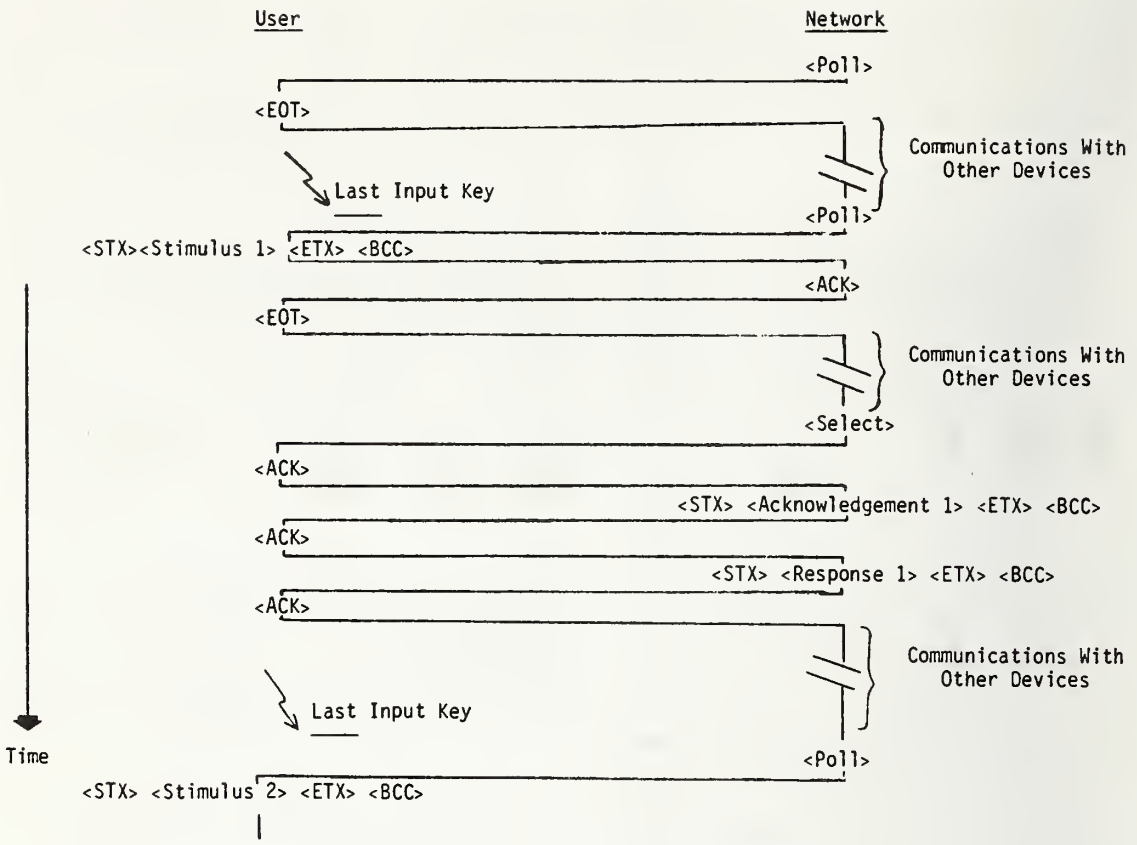


Figure 2. Binary Synchronous SAR Model

As illustrated in Figure 3, although stimulus waiting time is not directly measurable at the circuit interface, it can be bounded. Stimulus initiation occurs within the period delimited by the transmission of the last character of the previous response and the first character of the current stimulus. In the case of polled communications the start of this stimulus waiting time can be more accurately bounded by the time of the last unsuccessful poll to the device under test.

The median of the time interval from the previous (unanswered) poll until the present poll (which elicits transmission) is employed as an estimate of stimulus waiting time. Using this estimate of stimulus waiting time, acknowledgement delay time at the user's terminal can be estimated from the measured parameters by the following equation:

```
ACKNOWLEDGEMENT DELAY
TIME (estimated) =
  STIMULUS WAITING TIME
  (estimated) +
  STIMULUS TRANSMISSION TIME
  (measured) +
  ACKNOWLEDGEMENT DELAY TIME
  (measured)
```

Response time is then easily estimated (Figure 4):

```
RESPONSE TIME
  (estimated) =
  ACKNOWLEDGEMENT DELAY TIME
  (estimated) +
  ACKNOWLEDGEMENT TRANSMISSION TIME
  (measured) +
  RESPONSE DELAY TIME
  (measured)
```

Stimulus delay time at the user's terminal is estimated as a correction to the measured value by the amount of time that the entered stimulus waits for transmission:

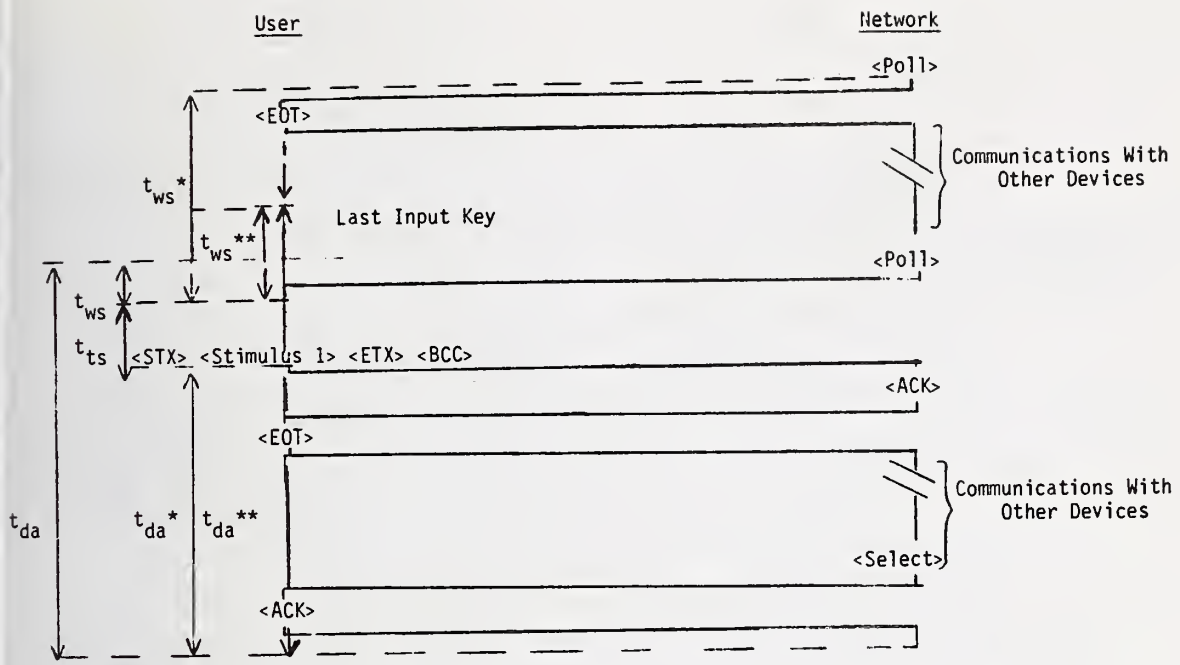


Figure 3. Estimated Acknowledgement Delay Time

STIMULUS DELAY TIME (estimated) = STIMULUS DELAY TIME (measured) - STIMULUS WAITING TIME (estimated)

3.2 ARQ Effects

A major difference between asynchronous and synchronous data transmission is the general presence of an error detection and correction system in the latter class. The most common such system is the ARQ (automatic request for retransmission) mechanism which is invoked subject to the results of a cyclic redundancy check made for each data transmission block. Proper measurement of service delivered to a user terminal requires reconciliation of the resulting retransmission sequences.

Since information transfer is a function of successful user-network communications, rather than unsuccessful transmission attempts, only correctly received transmission units contribute to SAR character count variables. Stimulus delay time ends and stimulus transmission time begins with the first transmission attempt of user input. Stimulus transmission time, as measured by the NMI, is an indication of the time required to successfully transmit the entire stimulus and therefore includes the transmission of unsuccessful transmitted units. The beginning of acknowledgement and response, from the user's viewpoint, is equivalent to the first successfully received transmission unit defined as such from the network. As illustrated in Figure 5, unsuccessful transmissions at the

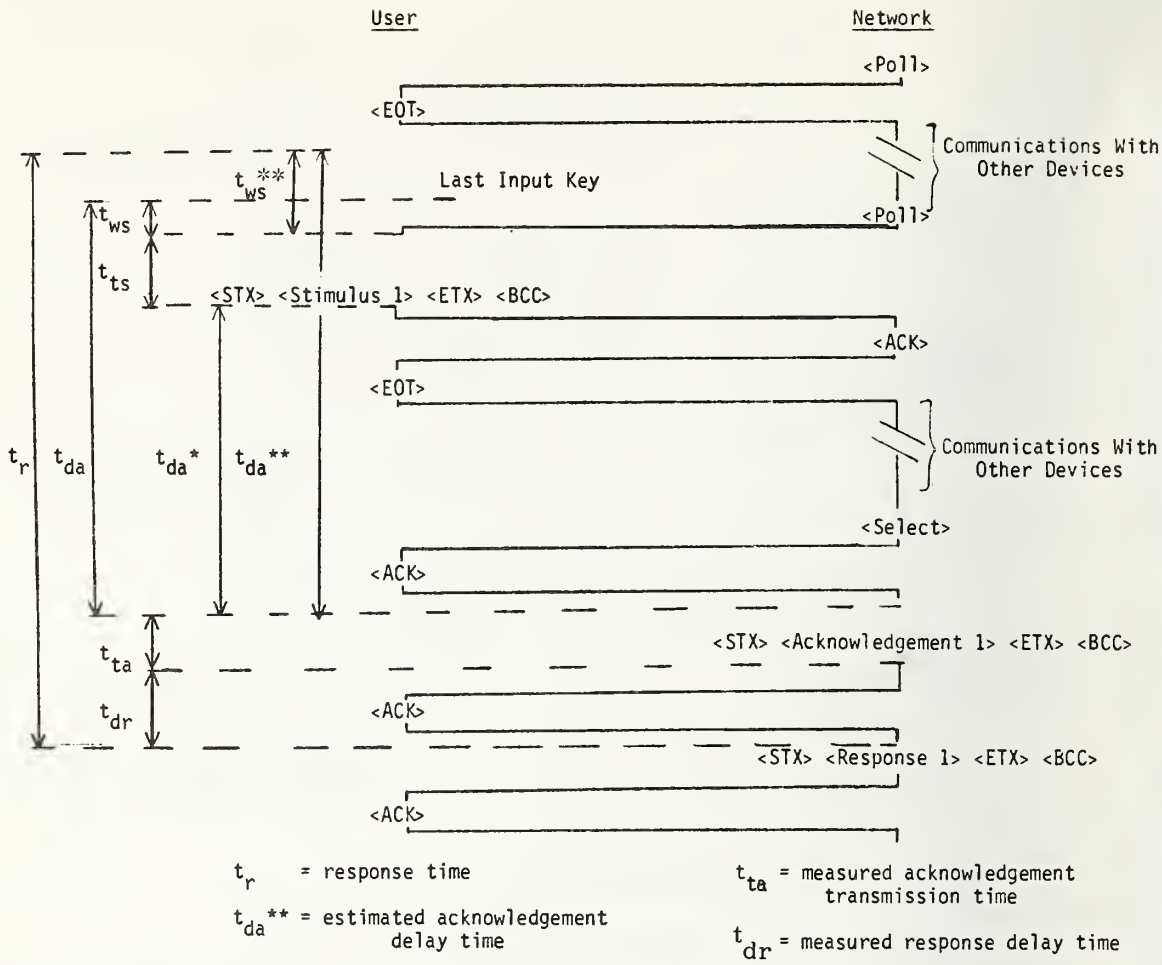


Figure 4. Response Time

beginning of acknowledgement and response contribute to acknowledgement and response delay time; retransmissions of intermediate transmission units contribute to acknowledgement and response transmission times.

3.3 NMI Implementation Characteristics

The NMI SAR interpretation logic is implemented as a finite state machine [3]. The state table which defines this machine is a function of the characteristics of the communications protocol to be measured. For synchronous transmission procedures, the inputs to the SAR analysis are data acquired from the circuit under test which have been separated and grouped by user terminal device address. Each link level transmission must be time stamped and defined by type (e.g., poll, information transmission, etc.). Content analysis is performed and acknowledgement boundaries are designated. Data resulting from this preprocessing is input to the SAR statistical analysis.

3.4 Protocol Violation Detection

Synchronous transmission sequences include a substantial amount of overhead data beyond the information content. The NMI logic which identifies synchronous transmission sequences by type is also implemented as a finite state machine. This protocol decoder dogmatically observes data link transmissions in

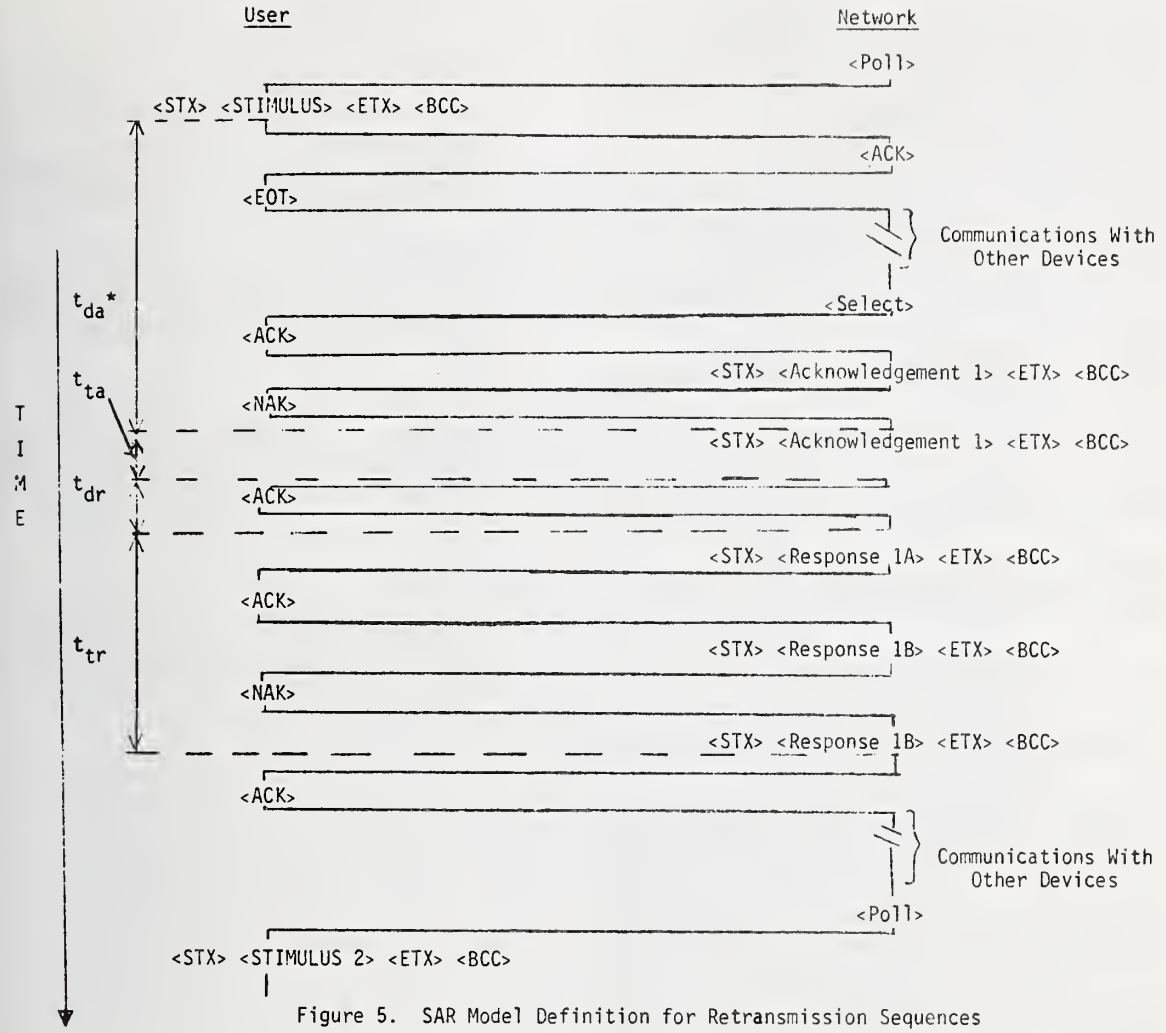


Figure 5. SAR Model Definition for Retransmission Sequences

terms of a given set of rules governing interaction; it will necessarily reject non-compliant transmission sequences. A level of protocol violation detection is therefore a natural byproduct of the design.

4.0 NMI APPLICATION EXTENSIONS

Syntatic protocol checking is insufficient when applications dependent information is needed. Semantic analysis is also required. It is increasingly common, in the data communications environment, for a terminal to access dynamically a diverse mix of applications via the same network connection. This diversity may result from multiple applications supported by a single host processing facility to which the user is connected, or perhaps because the terminal is connected to a resource sharing network. Regardless of the manner by which multiple services are available to the terminal, the user is often concerned with the discrete performance level of individual applications.

Measuring the responsiveness and other characteristics of separate categories of system access involves additional content analysis beyond that precipitated by the need to distinguish SAR components from a specific protocol. To the extent that there is adequate identification data contained within the transmission sequences, this further categorization of measurement results can be achieved.

Identification of categories of system utilization was implemented in the Network Measurement System for asynchronous use of a Univac 1108. Twenty-seven operating system commands were identified by state table syntactic analysis [4].

Examples of such expanded workload profiling capabilities include:

From the service consumer's viewpoint

*Relative responsiveness of one system or application versus another

From the service provider's viewpoint

*Relative frequency of usage for various categories of system services

*Profiles characterizing the typical use patterns of each system service

From either point of view

*Terminal device productivity comparisons

Additional data reduction and reporting software are now being implemented to provide these capabilities within the Network Measurement Instrument. Workload balancing among terminal populations and choosing locations for distributed processing functions deployed within the network are examples of analytical projects which will benefit from additional workload profiling.

5.0 BIBLIOGRAPHY

[1] Abrams, M. D., I. W. Cotton, S. W. Watkins, R. Rosenthal, and D. Rippy, "The NBS Network Measurement System," IEEE Transactions on Communications, October 1977, pp. 1189-1198.

[2] Watkins, S. W. and M. D. Abrams, Interpretation of Data in the Network Measurement System, NBS Technical Note 897, Feb. 1976.

[3] Abrams, M. D. and D. C. Neiman, "The NBS Network Measurement Instrument," CPEUG Proc. of the 15th Meeting, pp. 201-211, October 1979, NBS Special Publication 500-52.

[4] Cotton, I. W., Measurement of Interactive Computing: Methodology and Application, NBS Special Publication 500-48, June 1979.

Introduction to Data Communications

System Performance Parameters

Dana S. Grubb

Institute for Computer Sciences and Technology
National Bureau of Standards
Washington, D. C. 20234

This paper is an introduction to a set of user-oriented data communication system performance parameters that will permit the user to specify, compare, and measure data communication service. The set of parameters is designed to be universal in application for any digital data communication system regardless of the control protocol or network topology used. This set of parameters is also selected to provide a comprehensive specification of data communication requirements. The parameters are the subject of a proposed ANSI standard. The parameters are based on a similar set of parameters contained in Interim Federal Standard 1033. The primary parameters are specific measures of speed (delay and rate), accuracy, and reliability associated with the three primary functions: access, transfer, and disengagement.

Key words: Computer communications; computer networking; data communications; networks; performance requirements; telecommunications.

1. Introduction

Data communication users need to be able to specify their data communications requirements in terms that are useful to them. Their requirements are for the transfer of information between users at a specified delay, rate, accuracy, and reliability.

The user needs to be able to specify the requirements, compare available data communication services, and measure the service delivered. With appropriate user-oriented parameters, the user can avoid costly over-design by properly specifying exactly what is needed; reduce costs by comparing various data communication services; and determine how well the selected data communication service actually satisfies the requirements.

This paper is an introduction to a set of user-oriented data communication system performance parameters that will permit the user to specify, compare, and measure data communication service.

The set of parameters is intended to be applied universally to any digital data communication system regardless of the control protocol or network topology used. This set of parameters is also selected to provide a comprehensive specification of data communication requirements. Individual users may choose to omit parameters that they believe are not applicable to their needs or they may specify nominal values for those parameters.

The parameters are the subject of a proposed ANSI (American National Standard Institute) standard. The parameters are related to a similar set of parameters contained in Interim Federal Standard 1033 adopted by the General Services Administration (GSA) in May 1979.

2. User/System Interface

User requirements apply to data communication service as seen by the end user. There are normally two end users, each with its own interface to the data communication system. (An example of having more than two end users is the case of a message broadcast to all users on the system.)

The end user may be a human operator at a terminal, a device medium (for example, punched cards in a remote job entry terminal), or an application program in a computer. From an end user viewpoint, the data communication service requirement is the transfer of information between end users with specified delay, flow rate, accuracy, and reliability. This interface between end users and the system is termed the user/system interface.

Typical interactions at the user/system interface are an operator keystroke on a terminal keyboard; the punching of a paper tape; or calling an operating system by an application program.

The user/system interface contrasts with traditional DTE (Data Terminal Equipment)/DCE (Data Circuit-terminating Equipment) interface, which is between the terminal or computer port and a data modem or other circuit-terminating device. The DTE/DCE interface has been the point of interconnection between computer equipment and equipment provided by data communication service organizations.

3. User Information

User information consists of all digital information that is intended to cross both of the user/system interfaces. The user information bits are those bits used for the binary representation of the user information transferred from the source user to the system for ultimate delivery to a destination user. When user information is input as non-binary symbols (for example, operator keyboard entries) the user information bits are the bits used to initially encode these symbols.

User information does not include overhead information. For example, parity bits and start-stop bits are overhead information used by the system due to its internal needs. The distinction is that only information crossing both user/system interfaces is user information.

4. Functions

The three primary functions common to any data communication process between two end users are: access, user information transfer, and disengagement.

The access function includes those activities that the originating user and the system must accomplish for the system to accept information for transfer to a destination user. Access starts with an "access request" and ends when the first bit or block of source user information has been transferred to the system. The term block denotes contiguous information bits grouped at the source user/system interface for transfer to a destination user as a unit. A block may be a single ASCII character, a card image, a computer word, or the information field of a frame, depending on the equipment and protocol characteristics associated with the user/system interface.

The user information transfer function includes those activities that the system and the user must accomplish to transfer user information from the source user to the destination user. The function begins when the first bit or block of user information is input to the system and ends when the last "disengagement request" is issued.

There are normally two disengagement functions, one for each of the two end users. Each function begins with the issuance of a "disengagement request" and ends on the issuance of a subsequent "disengagement confirmation" or when that user is free to begin a new access.

5. Parameters

The parameters are divided into two categories: primary parameters and ancillary parameters. The primary parameters are specific measures of speed (delay or rate), accuracy, and reliability associated with the three primary functions; access, transfer, and disengagement. The primary parameters are shown on Table 1. There are also ancillary parameters that qualify the primary speed parameters by describing the extent to which the primary speed parameter values are influenced by user delays.

The primary parameters are developed in two steps: (1) a set of possible outcomes

are defined for each function; and (2) appropriate probability, delay, and rate parameters are defined relative to each possible outcome.

The outcomes are the possible end results that may occur on any given attempt to perform. Using the analogy of the familiar voice telephone call on the dial telephone network, the access function could end in a correct connection, a wrong number, exchange busy, called party busy, or no connection.

The bit and block oriented measures of speed are developed using the average of many consecutive trials. A quantity of bits called a Sample is defined by the standard to serve as a minimum number of bit transfer outcomes for meaningful measurement of parameter values.

5.1 Delay Parameters

There are four delay parameters. Average Access Time and Average Disengagement Time are the average elapsed time for the successful outcome of the access and disengagement functions. Average Bit Transfer Time and Average Block Transfer Time are the average value of elapsed time between start of a bit or block transfer attempt and successful transfer of that bit or block to the destination user.

5.2 Rate Parameter

User Information Bit Transfer Rate is the total number of user information bits successfully transferred to and accepted by the destination user divided by the time that is required for their transfer and acceptance. In the case of message oriented systems (e.g., packet networks), this time is observed at both the source user/system and system/destination user interfaces, and whichever time is longer is used in determining the transfer rate.

5.3 Accuracy Parameters

Incorrect Access Probability is the ratio of total access attempts that result in incorrect access to total access attempts, exclusive of failures attributable to the user.

Bit or Block Error Probability is the ratio of total incorrect bits or blocks to the total successfully delivered bits or blocks. A block is in error if one or more bits in the block are in error; or when some, but not all, of the bits in a block are lost or extra bits.

Bit or Block Misdelivery Probability is the ratio of total misdelivered bits or blocks to the total transferred. A misdelivered block is identified by the occurrence of the misdelivery outcome for any bit transfer outcome in a block. (Due to measurement difficulties, many users will choose to omit this parameter.)

Extra Bit or Block Probability is the ratio of total extra bits or blocks to total received bits or blocks. An extra block is identified by the occurrence of the extra bit outcome for all bit transfer outcomes in the block.

5.4 Reliability Parameters

Bit and Block Loss are considered to be reliability parameters. Bit or Block Loss Probability is the ratio of total lost bits or blocks to the total transmitted bits or blocks. A lost block is identified by the occurrence of lost bit outcomes for all bit transfer outcomes in a block.

Another important aspect of reliability is system availability. The user may be denied service due to a system outage (e.g., a computer failure or communication line failure) or due to system blockage from an excess of communications traffic. It is quite helpful to the user to know whether a service denial is due to system outage or blockage. In the case of an outage, the user calls the service personnel. In the case of a blockage due to overload, the user waits and then tries again for service. Access Outage Probability is distinguished from Access Denial Probability by the failure of the system to issue any active interface signals.

6. Proposed Standard

The ANSI work is being done by Task Group 5 of the ANSI X3S3 Committee and is contained in the working document X3S35/125. The X3S35/125 document differs significantly from 1033. It is expected that the proposed standard (if adopted by ANSI) will also be adopted as a joint Federal Information Processing Standard and Federal Telecommunications Standard, replacing the current interim (and voluntary) standard.

The current schedule calls for the proposed standard to be completed and sent to ANSI X3S3 in December 1980. Assuming this date is met, the time required for review, resolution of comments, and approval make early 1982 a reasonable estimate for approval by ANSI X3S3. Some additional

time will be required for approval as a joint federal standard.

The standard that is currently under development identifies parameters that describe system performance, but it does not define how values will be determined for these parameters. A subsequent ANSI standard is planned for specifying the measurement methodology.

Current federal work includes support for the ANSI committee and a joint NBS (National Bureau of Standards)/NTIA (National Telecommunications and Information Administration) project on measurement methodology for the parameters.

7. Recent History

There are currently two ANSI standards on data communication performance: X3.44 - 1974 (soon to be updated) and X3.79 - 1979. These standards provide parameters for specifying the flow rate, delay, and accuracy of transferred user information and availability of the service. These measures are user oriented; they specify measures of performance with regard to user information. However, they are not independent of system control protocol and are not always measured at the user/system interface.

8. Conclusions

The parameters discussed offer the data communication user an opportunity to specify requirements in user oriented, system independent terms. The proposed parameters may be used to specify and compare data communication services, and to measure the service received.

References

(1) Interim Federal Standard 1033, General Services Administration, May 1, 1979.

(2) Digital Communication Performance Parameters for Proposed Federal Standard 1033, Volumes 1 and 2, NTIA-Report 78-4, U. S. Department of Commerce, May 1978.

(3) Determination of Performance of Data Communication Systems, American National Standard X3.44 - 1974.

(4) Determination of Performance of Data Communication Systems that Use Bit-Oriented Communication Control Procedures, American National Standard X3.79 - 1980.

(5) Criteria for the Performance Evaluation of Data Communication Services for Computer Networks, NBS Technical Note 882, September 1975.

	PRIMARY PARAMETERS			ANCILLARY PARAMETERS
FUNCTION	PERFORMANCE CRITERION			PERFORMANCE TIME ALLOCATIONS
	SPEED	ACCURACY	RELIABILITY	
ACCESS	AVERAGE ACCESS TIME	INCORRECT ACCESS PROBABILITY	ACCESS DENIAL PROBABILITY ACCESS OUTAGE PROBABILITY	USER FRACTION OF ACCESS TIME
USER INFORMATION TRANSFER	AVERAGE BIT TRANSFER TIME	BIT MISDELIVERY PROB. BIT ERROR PROBABILITY EXTRA BIT PROBABILITY	BIT LOSS PROBABILITY	USER FRACTION OF BLOCK TRANSFER TIME
	AVERAGE BLOCK TRANSFER TIME	BLOCK MISDELIVERY PROB. BLOCK ERROR PROBABILITY EXTRA BLOCK PROBABILITY	BLOCK LOSS PROBABILITY	
	USER INFORMATION BIT TRANSFER RATE	TRANSFER DENIAL PROBABILITY		USER FRACTION OF TRANSFER TIME
DISENGAGEMENT	AVERAGE DISENGAGEMENT TIME	DISENGAGEMENT DENIAL PROBABILITY		USER FRACTION OF DISENGAGEMENT TIME

Table 1. Parameters

Computer Network Performance

User-Oriented Carrier Sense Multiple Access Bus Simulator

Marjan Krajewski

The MITRE Corporation
Bedford, MA 01730

This paper describes a general purpose, user-oriented Carrier Sense Multiple Access (CSMA) bus simulation program written in GPSS-V. Designed to fill a need to quickly and easily predict bus performance, it provides the capability to simulate local area computer communication networks governed by one of the CSMA protocols. Simplicity, flexibility, and run-time economy were the key criteria employed in the formulation.

CSMA protocols are a class of stochastic packet switching techniques which allow multiple independent network subscribers to share a single communications channel with reasonably high efficiency. All are based upon the original ALOHA protocol. They possess their greatest potential in meeting the communication needs of bursty, terminal-dominated networks operating over a limited geographical area. In general, these networks consist of low rate devices requiring extremely short response times.

The simulator was designed to be quickly adaptable to different network characteristics. It is user-oriented in that all simulation parameters are documented and initialized in a single section of the program. Subscriber populations can range in size from few to many thousands of terminals. Outputs from a simulation run consist of the system characteristics input by the user and statistical channel performance information gathered automatically by GPSS. This includes the average throughput, subscriber traffic, and deferred and retransmitted packet statistics. Statistics concerning individual packet delays and overall end-to-end response times are formatted and output in the form of delay distribution tables.

The program's primary application domain is the prediction and evaluation of those bus performance characteristics relevant to initial local area network design decisions. These include packet size, bus capacity, and the general suitability of CSMA protocols.

Key words: Bus Networks; Bus Performance; Computer Programs; Contention; CSMA; GPSS; Local Area Networks; Packet Switching; Simulation.

1. Introduction

The tremendous proliferation of digital devices has spurred a great deal of diversity in the technology of their interconnection. Depending upon the device data generation characteristics, response time requirements, and physical separation, numerous techniques exist which can provide the required connectivity. Most of these techniques utilize some form of packet switching such that message packets,

consisting of data appended to a header, form the basic units of information transfer. For those networks limited in geographical extent and consisting chiefly of numerous, relatively inexpensive devices, the most cost-effective packet switching approaches are those involving a common shared channel.

Networks characterized as groups of interconnected devices jointly sharing a common communications channel, or bus, are termed broadcast networks. Devices belonging to such networks interact first by gaining unique access to the bus through some multiple access scheme and then by broadcasting packets over it. Individual devices receive all packets but only accept those containing the proper addresses in the header. Due to the broadcast nature of the bus technique, time is the resource which must be shared.

The potential for efficient time sharing of a communications bus is a major purported advantage of numerous multiple access techniques which have been developed in recent years. Due to mismatches between subscriber traffic characteristics and idiosyncrasies of the individual schemes, however, the overall system performance can be substantially less than efficient. It is imperative, therefore, that the prediction and evaluation of bus performance be a necessary step in the design of all such systems.

Analytical methods are extremely useful in this regard where such models already exist and where a general estimate of bus performance is sufficient. For detailed estimates of specific network implementations, simulation techniques provide significantly greater flexibility and thus, more accuracy.

The simulation program described in this paper was designed to fill a need to quickly and easily predict characteristics of bus performance relevant to initial broadcast network design decisions. Simplicity, flexibility, and run-time economy were the key criteria employed in its formulation. Its range encompasses a class of stochastic channel access protocols known as Carrier Sense Multiple Access (CSMA).

2. Bus Access Protocols

A basic broadcast network architecture is shown in Figure 1. Individual subscribers are connected to a Bus Interface Unit (BIU) through some device-dependent parallel or serial interface. The BIU buffers the data, packetizes it, and then transmits it onto an inbound data stream via a multiple access protocol. Every BIU listens to the outbound data stream continuously in order to detect packets intended for them.

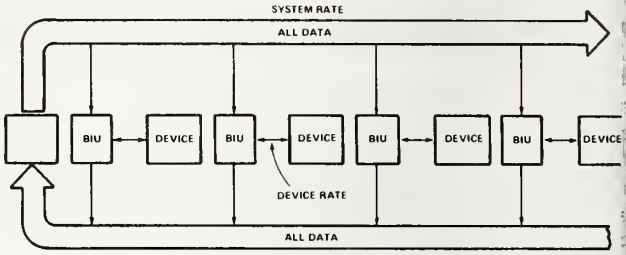


Figure 1. Broadcast Networks

Of the many multiple access schemes which now exist, [1,2][1] each has advantages and disadvantages with different subscriber populations. Some techniques excel with steady, high speed users while others perform best when servicing bursty, low speed users. In general, broadcast networks supporting steady users perform best when channel time is assigned to individual users. Bursty users, on the other hand, receive better service when channel time is unassigned and individual users "contend" for it. The basic advantage of contention-based access protocols is the ability to provide simple, reliable and flexible service to a predominately bursty subscriber population. Such networks are often dominated by low data rate terminal devices requiring extremely short response times at irregular intervals. The primary disadvantage of these protocols is their potential for unstable operation at certain throughput levels.

The first application of contention protocols to computer communication networks was the ALOHA system at the University of Hawaii [3]. With ALOHA (Figure 2), subscribers simply transmit whenever a message packet is ready. Overlapping transmissions from two or more subscribers always result in a collision and mutual destruction of the colliding packets. These are then retransmitted after random delays to prevent the same collision from occurring forever. Requiring all transmissions to conform to imaginary time boundaries (slots) improves upon this by forcing all collisions to overlap completely rather than partially. CSMA protocols [4] improve

[1] Figures in brackets indicate the literature references at the end of this paper.

performance still further by requiring all subscribers to sense the channel prior to transmitting and thereby avoid obvious collisions. Detection of a busy channel results in the immediate deferral of the transmission attempt. Due to the physical separation of individual subscribers, however, a packet may be enroute without other subscribers realizing it. Another subscriber may sense the channel and, finding it empty, transmit its packet into a collision with the enroute packet. The "vulnerable period" when two subscribers may collide is equal to the propagation delay between their respective transmit and receive ports. As the delay increases, the performance of CSMA worsens and approaches that obtainable with the ALOHA protocol. As with ALOHA, subscribers involved in a collision detect that fact and then reschedule their packets with a random delay. The random retransmission delay is the fundamental controllable parameter in the design and operation of all ALOHA-based access protocols. Increasing the delay results in higher obtainable throughput at the penalty of increased system response time.

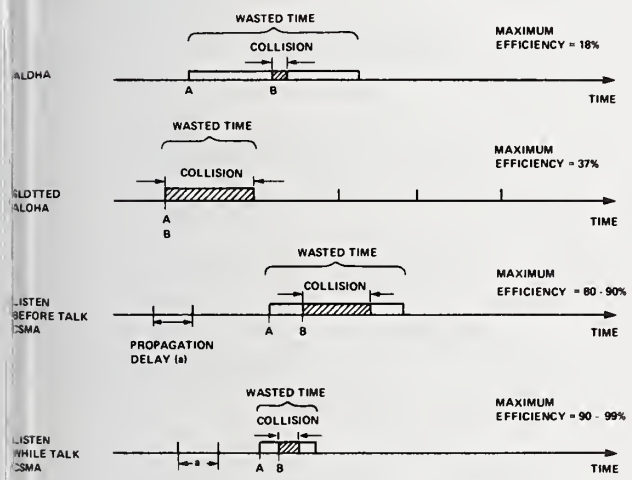


Figure 2. Contention Protocols

CSMA protocols are divided into two general categories: Listen-Before-Talk (LBT) and Listen-While-Talk (LWT). Subscribers operating in an LBT mode do not listen to their own transmissions. Collisions are detected only after the total packet has been transmitted and a positive acknowledgement has not been received within a predetermined interval of time. This acknowledgement may be transmitted on either the common data channel or on a separate one. Collisions tie up the channel for the duration of the colliding packets plus propagation delays. Subscribers operating in an LWT mode do listen to their own transmissions and cease transmitting when a collision is detected. Collisions in this case tie up the channel only for the duration of the overlapping propagation delays. LWT operation, therefore, is inherently more efficient than LBT.

The CSMA protocols can be further differentiated into three sub-categories depending upon the specific deferral algorithm used. Subscribers sensing a busy channel may reschedule their transmission attempt a random time in the future regardless of when the channel becomes idle. Otherwise, they may persist in sensing the channel and transmit with probability 1.0 when the channel becomes idle. Yet another option is to persist in sensing the channel until it is idle and then transmit with probability p by introducing a random delay. These algorithms are referred to as non-persistent CSMA, 1-persistent CSMA, and p-persistent CSMA, respectively.

All six combinations of operating mode and deferral algorithm can be modeled with the simulation program. Acknowledgement transmissions over the common channel can also be simulated.

3. Subscriber Populations

The types of subscriber populations encountered in practice often can be broken down into two generic classes. One class consists solely of subscribers with similar data transmission characteristics. Examples of this might be a collection of distributed processors communicating among themselves or a large collection of terminals transmitting data to receive only devices for storage or display.

A second class consists of numerous interactive subscriber pairs in which each complete message is followed by a reply. This situation can occur in a distributed time-sharing environment where users construct messages at terminals, send them to individual host computers, and await a response.

It is also possible for groups of subscribers sharing the same channel to have dissimilar transmission characteristics. Prime examples are independent, interactive or non-interactive populations or an interactive population where one or a very few host computers service a large number of terminals. In these cases, the populations must be modeled separately.

In all of the above cases, the traffic generated by individual subscribers is usually bursty and therefore possesses non-deterministic properties. Similarly, the random aspects of the CSMA protocols also result in the need for a stochastic system model. Combining dissimilar populations in an analytical model is difficult, at best, whereby in a simulation model it is relatively straightforward. Such situations demonstrate the intrinsic value of a simulation capability.

Two dissimilar populations may be modeled with this simulator. Their subscribers may range into the thousands. Complete messages may consist of several packets with the last packet being shorter than the others if necessary. For convenience, the fixed time delays not associated with transfers over the bus can also be included in the simulation run. In addition, the capability exists to simulate a massive collision in which every subscriber attempts to transmit at time zero. This feature is extremely useful in measuring a system's response to its maximum traffic load and possible channel instabilities.

4. Simulation Model

GPSS was selected as the programming language for the simulator primarily because of its power in defining systems governed by queues [5]. Developed at IBM in the 1960's, its underlying rationale was to provide a general structure for easily and rapidly converting discrete system models into computer programs. Since CSMA bus systems are, by their very nature, discrete queued systems, GPSS offered many advantages. In essence, it allowed more productive time to be spent on model formulation and less on bookkeeping than was possible with non-simulation specific programming languages.

The simulation program is composed of four main sections. An input section allows the user to initialize the various system parameters needed to perform the simulation. The system model section provides the GPSS entity description of the general model implementation. A control section controls the length of the simulation run and initiates the collection of those statistics which require a transient period prior to steady state operation. Finally, a report section formats the simulation output.

The system model section is the most complex of the four. Referring to Figure 3, messages are created as transactions for each population of subscribers according to a joint exponential interarrival time distribution. The mean of this distribution is based upon the aggregate average interarrival time of messages to the system, considering all subscribers in that population. Once created, transactions are assigned appropriate parameter values and sent on to a block which gives them a probability of entrance into the system. This probability is proportional to the fraction of subscribers not in the process of sending a message or waiting for a reply. Those transactions which do not enter the system are destroyed

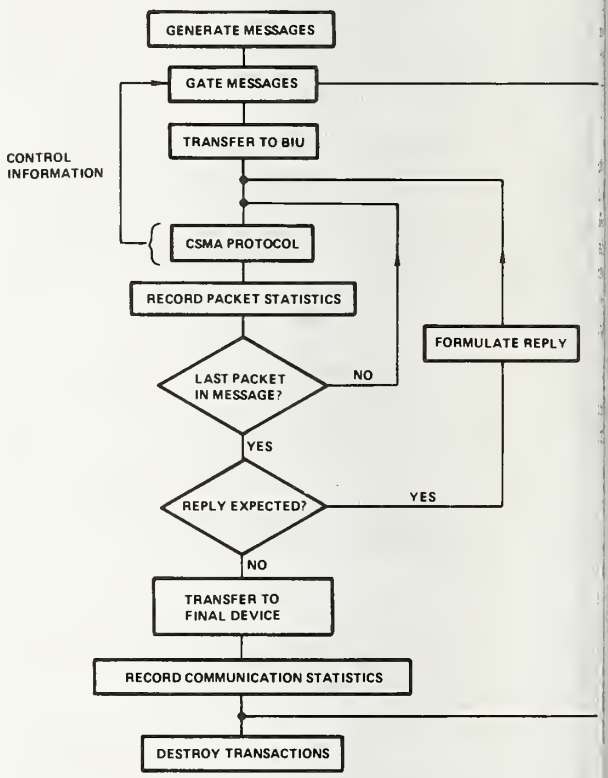


Figure 3. Simulation Model

Once in the system, transactions, representing messages, are first transferred into the BIU in the specified time and then attempt the selected CSMA protocol. A successful transmission is recorded by computing the appropriate delay and throughput statistics. If the transmission represents the last packet in a message, a reply is generated if so desired. If it is not the last packet, the transaction again attempts the CSMA protocol. Replies are delayed for an appropriate formulation time before the CSMA protocol is attempted. Following reception of a complete message and its reply (if any), the appropriate BIU-device

message transfer time is added and the total communication statistics are recorded. The transaction representing that particular communication event is then destroyed and its memory space made available for new transactions.

A complete communication event is termed a call and represents a basic unit of information interchange between network subscribers. An example of this is an interactive terminal to computer communication sequence. In this case, a call is defined as those events occurring between the initiation of the transmission of the first character of the query and the reception of the final character of the response, by the terminal device. If a reply was not generated, the one-way message defines the call.

If an acknowledgement is specified, the successful reception of the acknowledgement packet is required before the next data packet in a message or reply can be sent. Under heavy bus traffic loads, the loss of acknowledgement packets through collisions can, therefore, adversely affect the overall delays.

Certain assumptions were made about the characteristics of the systems to be simulated. These are as follows:

A1. There are no noise or propagation errors occurring on the channel. Retransmissions only result from collisions.

A2. All packets involved in collisions must be retransmitted.

A3. All transmissions are heard by every subscriber on the channel.

A4. The time to sense the presence of other transmissions on the channel and act on this information is negligible.

A5. The new channel traffic generated by each population of subscribers is Poisson distributed.

A6. New messages are not generated until previous ones are completed.

A7. Acknowledgement packets are not themselves acknowledged upon reception.

All are reasonable assumptions for modeling coaxial cable and fiber optic propagation media and terminal-dominated subscriber populations.

Validation of the simulation model involved verifying its accuracy in reproducing predicted system behavior. In Figure 4, theoretical derivations of system performance [6] are compared with simulation runs of identical systems. For this particular comparison there are 50 subscribers, the packet length is equal to 100 times the propagation delay, and the average retransmission delay (R) is equal to 20 times the propagation delay in one case and 100 times in the other. As is evident, both are in excellent agreement.

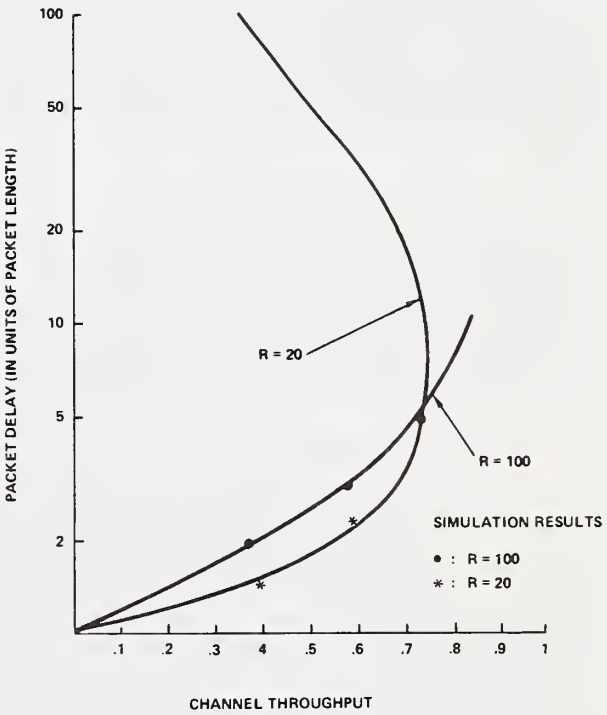


Figure 4. Predicted Delay Comparisons

5. Simulation Initialization

The simulation program was designed to be readily adaptable to different bus systems. It is user-oriented in that all simulation parameters are initialized in a single section of the program. The parameters are divided into five functional groups.

Common system parameters define the general characteristics of the simulation operating environment and BIU operation. They include the CSMA operating mode and deferral algorithm desired, whether an initial massive collision is desired, the

channel data rate, the number of bits per packet, the mean propagation delay, the mean retransmission delay following a collision, the transient time, and the nominal steady-state running time of the simulation.

Acknowledgement parameters define the characteristics of packets transmitted by a receiver back toward the sender to acknowledge a successful data packet reception. They include the number of bits per packet, the packet formulation time, and the waiting time which must pass before a transmitter assumes it must retransmit its original data packet.

Reply parameters define the characteristics of reply messages generated by interactive populations. They consist of the reply formulation time, the number of individual packets making up the reply, and the number of bits in the last packet of the reply.

Terminal parameters define the characteristics of the two individual subscriber populations. For each group, the parameters include the number of subscribers, the mean interarrival time between messages, the number of packets making up each message, and the number of bits in the last packet of the message.

6. Simulation Outputs

Outputs from the simulation program utilize information automatically collected for the particular GPSS entities used in the model. Statistics for delays and throughput are recorded only for the steady-state time period indicated during initialization.

The average system throughput is a running average of the overall channel throughput, including all successfully received data and acknowledgement packets. It is computed as the ratio of time spent in successful transmission to total simulated time, expressed as a percentage.

Individual terminal population call traffic statistics are also collected. The statistics printed out include the total number of calls generated during the run, the maximum number of calls in progress at any one time, the final number of calls in progress at the end of the run, and the average number of calls in progress at any one time.

Deferred or retransmitted packet statistics for the simulation run include the maximum number of transactions in the appropriate queue, the number of transactic in the queue at the end of the run, and the average number of transactions in the queue at any one time.

Delay information is output in the for of frequency tables. Packet delay and individual terminal population call respons times are handled separately. Call respons time is defined as the end-to-end response time encountered during a call. The program outputs the total number of events recorded, mean delays, standard deviation, upper limit of frequency classes, observed frequency, percentage of total entries in a particular frequency class, and cumulative percentage of total entries in a particular frequency class.

7. Applications

The simulation program's primary appli cation domain is the initial determination of packet size, bus capacity and suitabilit of CSMA protocols. Both analysis and simulation results indicate that CSMA technique can become unstable if the bus traffic approaches some maximum value. When this point is reached, the channel becomes saturated with collisions, throughput rapidly decreases to near zero, and delays grow intolerably. It is imperative, therefore, that such a situation be avoided in all actual implementations.

The user-oriented nature of the simulation program has been found to be extreme useful in determining if a proposed system design is operating near the saturation point. If such is the case, design changes could be made and evaluated quickly and inexpensively. Optimization of system response time is also facilitated by this approach.

8. Example

An example demonstrating the simulator utility is its application to the performan analysis of a local area network soon to be installed within the MITRE/Bedford complex. Termed MITRENET, the system is envisioned a a non-persistent LBT bus operating at a 1.152 Mbps rate over a coaxial cable plant. Multifunction terminals will use the network to access host computers providing word processing, scientific computation, an other services. Local printers will also b interfaced to these hosts via the network. The primary contribution to bus traffic loading is expected to result from the word processing function. Two potential termina host protocols were under consideration and needed to be evaluated.

The echo-plex protocol requires every terminal-initiated character to be transmitted as a small, separately addressed packet which is echoed back by the host as another separate packet. The block-transfer protocol requires characters in the terminal-initiated sequence to be echoed and buffered locally, and then transmitted together as fewer, larger packets. The former carried the promise of straightforward implementation while the latter promised significantly greater efficiency. An important question concerned the maximum population the bus could support under each.

The projected network was modeled as in Figure 5. All word processing terminal-host communications were simulated by one interactive (with replies) terminal population, and all host-printer communications by a second, non-interactive (without replies) terminal population. There was one printer for every four word processing terminals. Simulation results indicated that approximately 280 word processing terminals and 70 printers was the maximum supportable population under the echo-plex protocol. Under the block-transfer protocol, over 400 word processing terminals and 100 printers could be supported before bus instability became a problem.

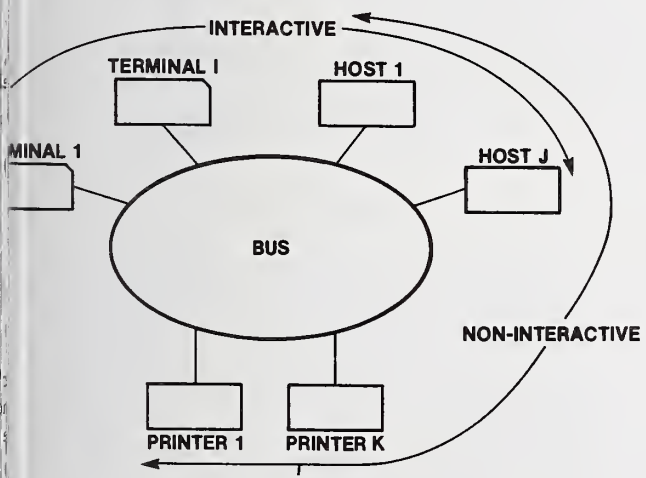


Figure 5. MITRENET Example

The design choices presented by this analysis were twofold. Selection of the echo-plex protocol was recommended if the projected number of simultaneously active word processing terminals would not exceed 280. Selection of the block-transfer protocol, an increase in bus capacity, or a switch to the LWT mode of operation was recommended if the projected number of simultaneously active word processing terminals exceeded 280. As stated earlier, the evaluation of any design changes would be a straightforward and relatively inexpensive task, requiring only that the appropriate input parameters be altered and the simulation rerun.

9. Conclusions

The capability to evaluate bus performance via a flexible, user-oriented simulator is valuable insurance against potentially costly mistakes. By orienting program usage toward users not already familiar with the programming language, minimizing detail in the model's input requirements, and concentrating on reducing run costs, simulators such as the one described here form powerful tools for broadcast network design.

10. Acknowledgements

The author wishes to gratefully acknowledge the efforts of Mary A. Flood for her improvements to the simulator's output format, Norman B. Meisner for his considerably helpful reviews and comments, and Karen M. Fiorello for her patience and expertise in preparing this paper.

11. References

1. R. M. Metcalfe and D. R. Boggs, "Ethernet: Distributed Packet Switching for Local Computer Networks," CACM, Vol. 19, No. 7, July 1976.

2. Meisner, Norman B., et. al, "Time Division Digital Bus Techniques Implemented on Coaxial Cable," Computer Networking Symposium, December 1977.

3. N. Abramson, "The Theory of Packet Broadcasting," TRB76-1, The ALOHA System, University of Hawaii, January 1976.

4. L. Kleinrock and F. Tobagi, "Packet Switching in Radio Channels: Part I - Carrier Sense Multiple Access Modes and Their Throughput Delay Characteristics," IEEE Trans. Commun., Vol. COM-23, No. 12, December 1975.

5. IBM, "General Purpose Simulation System V Users Manual," SH20-0851-1, Second Edition, August 1971, Rev. 1977.

6. F. Tobagi and L. Kleinrock, "Packet Switching in Radio Channels: Part IV - Stability Considerations and Dynamic Control in Carrier Sense Multiple Access," IEEE Trans. Commun., Vol. COM-25, No. 10, October 1977.

A Comparative Evaluation of Local Area Communication Technology

R.L. Larsen, J.R. Agre, A.K. Agrawala

Department of Computer Science
University of Maryland
College Park, MD 20742

The builder of a local area network is immediately confronted with the selection of a communications architecture to interconnect the elements (hosts and terminals) of the network. This choice must often be made in the presence of great uncertainty regarding the available alternatives and their capabilities, and a dearth of comparative information. This was the situation confronting NASA upon seriously considering local area networks as an architecture for mission support operations. As a result, a comparative study was performed in which alternative communication architectures were evaluated under similar operating conditions and system configurations. Considered were: (1) the ring, (2) the cable-bus, (3) a circuit-switching system, and (4) a shared memory system. The principle performance criterion used was the mean time required to move a message from one host processor to another host processor. Local operations within each host, such as interrupt service time, were considered to be part of this overall time. The performance of each alternative was evaluated through simulation models and is summarized in this paper.

1. Introduction

As local computer networks become a viable alternative in operational computing environments, the selection of an appropriate intercomputer communications system design becomes one of the most basic design decisions made. The performance of the overall system can depend on making a proper choice, yet there exists relatively little information of a comparative nature by which to make such a design choice. This situation confronted a NASA study group at Goddard Space Flight Center, and led to a study performed by the University of Maryland Computer Science Department and summarized in this paper.

The presentation is basically structured in four parts. Section 2 provides background information establishing the problem context. In section 3 the approach to the analysis is described, followed in section 4 by a brief description of each of the four alternative systems evaluated. The major performance results are summarized in section 5, and section 6 closes with a summary and some general conclusions.

2. The Local Network Environment

The communications system of a computer network can be viewed as a subnetwork of the computing network. Its major responsibility is to enable the transfer of data messages (files, programs, status and control information) between computers on the network. To the communications subnet, the nodes of the network are viewed as functional "black boxes", which remove messages from the subnetwork, and which provide messages to the subnetwork to be transmitted to other nodes. In a local network, the distance between the components of the system is assumed to be less than a few kilometers. Because of the small distance involved, high transmission rates with low bit error rates are achievable.

The subnet must supply an entry point interface to each of the network host nodes (computers). In reality this interface must handle address translation, transmission protocols, line speed conversion, error control, routing decisions, flow control, and the buffering of messages. Address translation is required to determine where a message received from the host node is to be sent or to determine if a message received from the communications subnet is destined for its attached host node. Transmission protocols determine how a message can be sent over a particular line, which line the message is to be sent on, and the message format. Line speed conversions are necessary when the incoming and outgoing data rates are different as in the case of peripheral devices or host nodes. Error control includes methods that insure the message will be received correctly. Routing decisions determine which path a message should follow. Since buffering techniques are used in all of the technologies studied, buffer management is an important function of the interface. In the evaluation reported here, address translation, transmission protocols, and the buffering of messages were considered. The interactions between the interface (communications processor) and the host processor were also modeled. It is felt that these interactions are likely to have a major impact on network performance. The models attempt to mimic the host to interface interactions as accurately as possible.

Communication in the models is envisioned as being packet based. Data messages are created at the host computer. When a data message has been created at a source host, the source host divides the message into smaller messages of a pre-determined maximum size, called a packet. Using this method, each packet is independently routed through the communications network. Although packet switching involves increased cost (additional control information is required in each packet) and complexity (packets must be reformed into messages at the destination node) this method does provide rapid response time for short messages in the network as well as high throughput for longer messages[5].[1] Packet switching also provides enhanced error control since each packet broadcast is usually shorter than an entire message would be. Another benefit of packet switching is that no host can monopolize the communications network for long periods of time. In this study, the communications medium was allocated on a packet basis.

[1]Figures in brackets indicate the literature references at the end of this paper.

3. Approach

The major objective was to derive comparative performance measures by which a system designer could intelligently select an appropriate communications technology for a local computer network. To achieve this objective, the overall modeling framework had to be fixed, and a common set of performance measures determined. The performance measures which will be reported in the paper for each of the technologies include: mean message time in the system, communications system throughput, and bandwidth utilization; as a function of message arriva rate and system configuration.

An overall system modeling framework was defined which included configurations of 8, 16, and 32 host processors. A communications traffic model was then developed to simulate the generation of messages by host processors for communication to other host processors. Figure 1 illustrates the modeling framework devised for eight hosts.

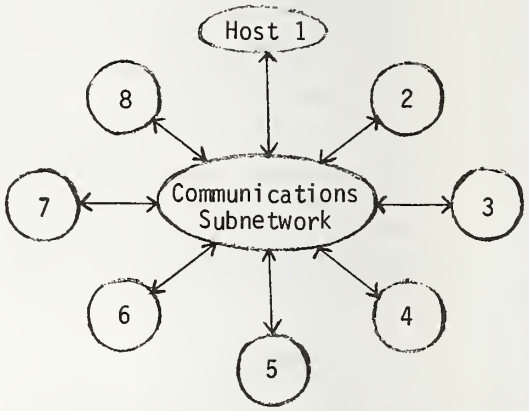


Figure 1. Modeling Framework

From the perspective of each host processor, performance statistics were kept measuring the perceived performance of the communications subnetwork as a function of the arrival rate of messages to be transmitted.

Given the structure of Figure 1, alternative models of communications subnetworks could then be developed and evaluated within a uniform framework and comparative performance measures derived.

The communications traffic model developed includes a Poisson message source within each host processor which generates messages with length uniformly distributed

between 300 and 5000 16-bit words. These messages are then broken up into fixed-length packets of 256 16-bit words (4096 bits). Message destinations are selected randomly and uniformly among the other host processors. The rate of the Poisson sources is increased from low loading levels to communications subnetwork saturation.

Within this modeling framework, candidate communication subnetwork designs were selected and modeled. Due to the complexity of the systems studied, discrete event simulation was determined to be the most appropriate modeling technique, and SIMSCRIPT II.5 [2] was selected as the simulation language.

4. Candidate Communication Systems

Four communication alternatives were selected for detailed modeling: (1) the ring, (2) the cable-bus, (3) circuit switching, and (4) shared memory. For each alternative, if an operational version was known to exist, that design was typically chosen and, in some cases, modified slightly to take advantage of known design improvements. In this section a brief description of each alternative is presented, including an overall high level description, a description of the hardware organization, and a discussion of the message transmission protocol employed.

4.1 Token-Controlled Ring

4.1.1 High Level Description

A ring network is a communication subnet in which host computers are interconnected into a single closed loop (ring) topology by means of a unidirectional transmission medium such as a twisted wire pair and appropriate interfaces to it [3]. A message, in our case a packet, circulates around the ring from source host to destination host, with each ring interface forwarding the packet to the next ring interface and then back to the source host. The scheme is displayed in Figure 2.

The token-controlled ring has a unique 10-bit pattern, called a token, that continuously circulates the ring. A Ring Interface desiring to transmit a packet must wait for the token to reach it. It then changes the token to what is called a connector (essentially a different 10-bit pattern), transmits its packet, and on receipt of the ACK (when the packet is returned to the transmitting host) restores the token pattern. A Ring Interface scanning the information flowing through it, will see a series of connector-packet pairs followed by a token that signals available space for transmission.

4.1.2 Ring Interface Description

All computers are connected to the ring by devices called a Ring Interface. This interface is responsible for control and actual communication between host computers over the ring. This interface is composed of a transmitter, a receiver, a buffer for incoming packets, a buffer for outgoing packets, switches to control the routing of information through the interface and control logic. The control logic is responsible for such functions as address recognition, error detection, and setting ACK flags. It also handles the DMA transfer of packets between the ring interface and the host computer.

4.1.3 Message Transmission Protocol

Event sequence for the transmission of a message from CPU A to CPU B:

(1) CPU A divides the message into fixed size packets and transmits them one at a time via DMA to the outgoing buffer of Ring Interface A.

(2) Ring Interface A waits to receive the token and then changes the token to a connector, transmits the packet and follows the packet by a new token.

(3) The packet moves around the ring to Ring Interface B.

(4) If Ring Interface B's incoming buffer is empty, Ring Interface B copies the packet into its buffer and computes a checksum. If no errors are detected, the ACK flag is set as the packet is forwarded to the next Ring Interface. CPU B is then interrupted to inform it of a received packet, and a DMA transfer is initiated.

(5) The packet continues around the ring until it reaches Ring Interface A, which removes it (and the leading connector) from the ring. If the ACK flag is set, Ring Interface A assumes successful transmission and informs CPU A. Otherwise go to step 2 to perform a retransmission of the packet.

(6) If there are no more packets in the

---

[2] SIMSCRIPT II.5 is a product of CACI.

message, CPU A copies the next packet into the outgoing buffer of its ring interface and goes to step 2.

4.2 Cable-Bus

4.2.1 High Level Description

The cable-bus system modeled is a variant of the Ethernet random access scheme which was developed by Metcalfe and Boggs[7]. This technology makes use of a coaxial cable (Ether) as the communications medium along with a communication protocol that was specially developed to work for cable based schemes. Unlike the Aloha random access scheme, Ethernet is designed to carry two-way traffic between several host computers. This modification enables the transmission of messages for a computer-to-computer, as well as a terminal-to-computer, computer-to-terminal organization.

4.2.2 Hardware Organization

The cable-bus is designed to accomodate up to 256 hosts. Each host is interfaced to the cable by means of a communications processor (CP), Figure 3.

The communications processor consists of a line tap, a transceiver, two buffers: one for incoming data, one for outgoing data, and control hardware capable of interrupting its host. The tap enables physical connection to the Ether. The transceiver is responsible for receiving and transmitting bits over the cable. The control hardware supervises the transfer of packets between the host and the buffers via DMA, carrier detection, truncated packet filtering, and collision consensus enforcement.

Carrier detection refers to detecting if the cable is in use. Communications processors will never transmit when the cable is sensed busy. During transmission of a packet, the communications processor monitors the cable, this is termed interference detection. Interference (a collision) occurs when the signal the communications processor is transmitting and the signal it receives are not the same. In this case the communications processor stops transmitting. Collision consensus enforcement insures that proper action is taken once a collision has occured. If a collision occurs, the communications processor detecting this condition places a noise pulse on the line to make sure all other communications processors are aware of the collision. Next, each communications processor which was involved in the collision begins a collision avoidance procedure. This procedure is also known as "blocked mode". First, a pseudo random number, uniformly distributed, is calculated in hardware. This number is then multiplied by the number of collisions which have occured in trying to send the current packet. The communications processor then waits the period of time indicated by the resulting number before attempting to transmit again.

The primary difference between the modeled cable-bus and Ethernet is that the modeled design incorporates what is termed an "in-buffer collision" scheme [2]. This refers to the case when a communications processor begins to receive a packet while its incoming buffer is full. The communications processor causes an artificial collision which terminates the transmission.

4.2.3 Message Transmission Protocol

Event sequence for transmission of a message from CPU A to CPU B:

(1) CPU A divides the message into a sequence of transactions called packets.

(2) CPU A transmits the next packet of the sequence via DMA to the outgoing buffer of its communications processor (CP A).

(3) CP A waits for the cable to be inactive and then transmits its packet, which results in one of the following:

   a. No collisions detected, packet successfully transmitted.

   b. A collision is detected (go to step 4).

(4) Each CP involved in the collision goes into "blocked mode", calculates the wait period, waits that period, and then go to step 3.

(5) All CPs begin wait period for the cable to once again be inactive.

(6) CP B copies the packet for the cable into its incoming buffer, and performs a checksum on the packet. If the checksum is valid, go to step 7. Otherwise, do not send an ACK and wait for the packet to be retransmitted.

(7) CP B transmits an ACK to CP A. CP B interrupts CPU B and a DMA transfer of the packet to the host is initiated.

(8) CP A receives the ACK. CP A interrupts

CPU A to inform it that the packet was successfully transmitted (go to step 2).

4.3 Circuit Switching

4.3.1 High Level Description

The Data Distributed Network (DDN) as developed by Data General Corporation (DGC) was designed to take advantage of circuit switching techniques to provide communications links that operate at a rate of a few megabits per second [4].

4.3.2 Hardware Organization

As developed by DGC, DDN is designed to accomodate up to 255 host computers. Each host is interfaced to the central switching facility of the DDN by means of an Adapter. DDN thus consists of a central switching facility called the Switch Matrix and as many Adaptors as there are hosts. The components of a DDN are shown in Figure 4.

Each adaptor is controlled by a microprocessor. Under host computer direction, the Adaptor sends data to and receives data from the Adaptors connected to other computers in the network. Data transfer between an Adaptor and its host computer takes place via DMA. Each Adaptor has two 256-word buffers, one for outgoing data and one for incoming data.

The Switch Matrix consists of three units:

(1) Diagnostic Unit - this unit is used to determine faults on the network and is capable of taking corrective actions to keep the network in operation. It is capable of communicating with any of the Adaptors in the network.

(2) Data Routing Matrix - this is the actual circuitry which establishes the physical connection between any two Adaptors, or between the Diagnostic Unit and any Adaptor. This unit is designed to contain up to 36 full duplex links to permit up to 72 computers to communicate simultaneously.

(3) Link Controller - monitors and controls the Data Routing Matrix. It cyclically polls all Adaptors, searching for one which desires to establish communication. When the Link Controller finds such an Adaptor, it cyclically scans the communication links until it finds a free link. Next, the Link Controller determines whether or not the specified destination is currently "busy". If the destination is busy, the Link Controller can not process the request, the current Adaptor is skipped, and the Link Controller continues to search for another Adaptor that desires to establish a communication. Otherwise, the communications link is established and the source begins transmitting its message on a packet basis. When transmission of the entire message is completed the link is dropped.

4.3.3 Message Transmission Protocol

Event sequence for transmission of a message from CPU A to CPU B:

(1) CPU A divides the message into packets and places the packets at the end of its packet queue. When the first packet of the message reaches the head of the queue, it is transmitted via DMA to the outgoing buffer of Adaptor A. Adaptor A now begins to request communications capabilities.

(2) The Link Controller reaches Adaptor A. If a link is available, the Link Controller then determines if Adaptor B is free. If Adaptor B is free the connection is made, otherwise the Link Controller resumes polling (Adaptor A is in a "suspended" state).

(3) Adaptor A prepares for transmission.

(4) Adaptor A transmits its packet.

(5) If the incoming buffer of Adaptor B is empty, the packet is copied into the buffer, else it is lost. The microprocessor of Adaptor B is interrupted upon completion of the transmission. If the packet was successfully received, the interrupt-handling routine of the microprocessor begins transmission of a 3-word packet ACK to Adaptor A, CPU B is also interrupted to inform it of a received packet. If the packet was lost, the microprocessor sends a NACK to Adaptor A.

(6) On receipt of the packet ACK:

   a. The microprocessor of Adaptor A is interrupted. The microprocessor then interrupts CPU A to inform it of an ACK. If this was the last packet of the message, Adaptor A informs the Switch Matrix to drop the link.

   b. When CPU A is interrupted and packets remain to transmit, a DMA

transfer of the next packet of the message to Adaptor A's outgoing buffer is initiated.

c. When the entire packet is received by Adaptor A, this packet can now be transmitted to CPU B (go to step 4).

(7) When CPU B is interrupted:

a. CPU B initiates a DMA transfer of the packet from Adaptor B's incoming buffer into its own memory.

b. Once the entire packet is copied by CPU B, Adaptor B's incoming buffer is reset so that it can now receive the next packet.

(8) If a NACK is received by Adaptor A, Adaptor A's microprocessor is interrupted upon receipt of the NACK which causes a retransmission of the lost packet (go to step 4).

4.4 Shared Memory

4.4.1 High Level Description

It has been found [1] that the use of buffer memories to set up communication links between two computers is appropriate when the distances to communicate are not very large. This method is used by the Launch Processing System at the Kennedy Space Center of NASA.

The Common Data Buffer (CDBFR) is a specially designed piece of hardware designed for intercomputer communication. This hardware permits a shared memory medium for transfer of messages between several CPUs.

4.4.2 Hardware Orgainzation

CDBFR consists of a memory module, memory controller, CPU interfaces called Buffer Access Cards (BAC), and interrupt control logic. Orginally the design permitted up to 32 CPUs to communicate via a single CDBFR. Subsequently the design changed to permit up to 64 CPUs to communicate. Figure 5 illustrates the CDBFR configuration.

CDBFR's memory module consists of from 32K to 64K 16-bit words of storage with a cycle time of about 250ns. The first 1K of memory (page 0) is divided into 32 stacks each of 32 words, with one stack being associated with each CPU attached to the CDBFR. The next 1K of memory (page 1) is reserved for "block data transfer". When a CPU obtains access of Page 1, in block data mode, no other CPU can access Page 1. The remainder of the CDBFR memory is allocated into private write areas for each CPU. While a CPU can read any word of memory, it can only write into its onw private write area, and Pages 0, 1. Each CPU has the responsibility for allocating and managing its own private write area.

Each CPU communicates to the CDBFR through a dedicated interface called a Buffer Access Card (BAC). The CPU sends requests for service, address, data and mode information to the BAC, which then provides this information to the CDBFR at the appropriate time. From the CDBFR, the BAC receives data and error condition information resulting from read and write operations into the CDBFR. The BAC then provides this information to the CPU. The BAC is also capable of transmitting interrupts to and from the CPU.

The Memory Controller is responsible for granting access to the memory. Only one CPU (through its BAC), is permitted access to the memory module at a time. The controller must scan each BAC, starting with the lowest addressed BAC to find the next BAC desiring service. Once a complete scan of the BACs is made the controller starts again with the BAC with the lowest address.

The CDBFR is capable of interrupting the CPUs for message synchronization. This interrupt stack control logic is activated when a CPU writes a word into the stacks of Page 0. When a word is written onto a stack the CPU which owns the stack is interrupted. Interrupts are postponed whenever the host computer's BAC is active with a data transmission. These interrupts are stacked in the CDBFR until the message transmission is complete.

4.4.3 Message Transmission Protocol

Each CPU is permitted only one outstanding message to any other destination at a time. Messages are divided into packets of a fixed size. The private write area is divided into a fixed number of blocks, each the maximum packet size.

Event sequence for transmission of a message from CPU A to CPU B:

(1) CPU A divides the message into a sequence of transactions called packets.

(2) CPU A allocates a block of its private write area to the next packet of the sequence. CPU A then transfers the packet into this block.

(3) CPU A then places a word containing the address of the packet (in the CDBFR memory) on the interrupt stack of CPU B.

(4) Next CPU B is interrupted by the CDBFR. CPU B then reads the packet address word off its stack.

(5) CPU B copies this packet out of the CDBFR into its own storage area.

(6) CPU B then places a word containing the original address of the packet onto the interrupt stack of CPU A. This serves as the ACK of the packet by CPU B.

(7) The CDBFR interrupts CPU A and CPU A reads the word off its stack. CPU A than verifies that the address is the original address of the packet and so assumes that the packet was successfully transmitted.

(8) CPU A then releases the block containing the packet and the packet transfer is complete.

(9) CPU A then returns to step 2 to process the next packet of the message sequence.

5. Performance Comparison

The major performance measures to be compared among the alternative systems are:

a. mean message time in the system

b. throughput capability

c. bandwidth utilization efficiency

To fully understand such a comparison however, one must consider the major differences in design variables among the systems and, while being unable to factor their effects out, at least appreciate their influence. Table 1 summarizes the two most critical design variables for each technology: the transmission rate and the number of physical paths. Two key observations can be made. The transmission rate for the Common Data Buffer is an order of magnitude greater than the others, and DDN utilized multiple physical paths. Since DDN is designed to provide communication among many (up to 255) computers using up to 24 distinct data paths, the number of physical paths available were scaled down for modeling purposes to be consistent with the fewer (8, 16, and 32) number of host processors, with the expectation that this provides one with more realistic insight into the operational performance of DDN. Furthermore, the original design for DDN utilized single-buffering in the adaptors. This design was subsequently modified to include double-buffering. Results for each of these cases are reported subsequently and demonstrate striking improvements for the double-buffered case.

Figures 6, 7, and 8, and Tables 2, 3, and 4 illustrate the primary performance results. Table 2 is a display of message time in a system measured in seconds, versus the intensity of arriving messages, measured in megabits per second. To be noted first is the observation that the ring and the cable-bus have virtually indistinguishable performance characteristics, which are also rather insensitive to the number of host processors attached. What could have been expected to be 6 separate plots on Figure 6, then, become essentially only one. A slight difference is discernible in Figure 8, but its influence is barely observable in most performance measures.

Six curves are shown for DDN. The first set (solid lines) display message time in a system for the single-buffered case, and the second set (dashed lines) display results for the double-buffered case. A significant performance improvement is apparent in the double-buffered case. It is apparent from these figures that double-buffering more than doubles the capacity of the DDN system. For each of the DDN configurations modeled, a processor to link ratio of 4:1 was maintained. The overall bandwidth available, therefore, increased by a factor of two going from eight to sixteen processors, and by another factor of two going to thirty-two processors. This study indicated that this bandwidth is allocated equally efficiently regardless of the number of links. Some saturation effects might be expected as the number of links grows beyond eight, but this was not explored.

Unlike DDN, which can be configured with a variable number of relatively low speed links, CDBFR provides one very fast shared path. When configured with only a few host processors, then, the hosts are unable to provide enough traffic to keep the CDBFR busy. This is shown in Figure 4, where the efficiency (the percent of bandwidth provided which can actually be used) is shown to increase with the number of host processors.

The data on which Figures 6, 7, and 8 are based are provided in Tables 2, 3, and 4 respectively. Data was taken for light, medium, and heavy loading. The notation

"∞" in the tables denotes a saturation condition where an infinite queue of messages to be transmitted was assumed to exist at each host processor.

6. Summary and Conclusions

Four intercomputer communications systems have been described and their performance compared. While these systems utilize different architectures and operate at different rates and capacities, comparable performance measures were stated and utilized to compare the systems. Existing technology provides bandwidth capabilities in the range of one to fifty megabits per second. Proven architectures allocate this bandwidth in different ways to host computers, with varying efficiency. The principal lesson to be learned from this study is to know the communication requirements. Relatively modest ring or cable-bus systems can very efficiently provide relatively modest capabilities. If growth flexibility is a strong requirement, then, perhaps, a DDN-style system is the preferred approach, although this flexibility comes with a substantial price tag. For a relatively large number of hosts with a heavy communications load, short response time requirements, and a potential need for immediate access by each host to a set of dynamic system-wide parameters, a CDBFR-type system may be an attractive alternative.

At the time this study was performed, a brief cost survey of the builders of these systems failed to reveal significant economies of scale in going to larger, faster communications systems for local networks. It seemed to be the case that one could expect to pay for every bit of the capability required.

It is our hope that this performance analysis can be used as a basis for comparison of most current and future communication technologies appropriate for local computer networks. The communications traffic considered in the analysis of these systems was of the most general form. The designer of a local computer network must understand the specific communications traffic expected in a system being designed, however, and understand the peculiarities of it. It was our experience in doing this for a NASA case study that some traffic-dependent performance deterioration can be encountered. This case study and its implications are the subject of a companion paper to this one [6].

Many people made significant contributions to this study. In particular we would like to gratefully acknowledge the contributions of Karen Gordon, Rose McGinnis, Ray Bryant, and Diane Acaron. The computing support for most of this work was provided by the Computer Science Center of the University of Maryland. The research was supported in part by NASA Goddard Space Flight Center under grant No. NAS5-22878 and by the Air Force Office of Scientific Research under grant No. AFOSR78-3654A.

References

[1] Agrawala, A.K., Bryant, R.M., and Agre, J.R., "A Study of Shared Memory as a Communication Medium", Proc. Sixteenth Annual Technical Symposium: Systems and Software, National Bureau of Standards, Gaithersburg, Maryland, June 2, 1977, pp. 51-60.

[2] Agrawala, A.K., Bryant, R.M., and Agre, J.R., "Analysis of an Ethernet-Like Protocol", Proc. of Computer Networking Symposium, National Bureau of Standards, Gaithersburg, Maryland, December 15, 1977, pp. 104-111.

[3] Agrawala, A.K., Agre, J.R., and Gordon, K.D., "The Slotted Ring vs. the Token-Controlled Ring: A Comparative Evaluation", IEEE COMPSAC 78, Chicago, Illinois, November, 1978, pp. 674-679.

[4] Agrawala, A.K., Gordon, K.D., and Agre, J.R., "Performance Analysis of a Circuit-Switched Sommunication Scheme for Local Computer Networks", Technical Report TR-645, Department of Computer Science, University of Maryland, April, 1978.

[5] Agrawala, A.K., and Larsen, R.L., "Distributed Computing - A Review", Technical Report TR-484, Department of Computer Science, University of Maryland, September, 1976.

[6] Agre, J.R., and McGinnis, R., "Comparative Performance Evaluation of Several Local Distributed Computer Networks: A Case Study", Technical Report, Department of Computer Science, University of Maryland, November, 1979.

[7] Metcalf, R.M., and Boggs, D.R., "Ethernet: Distributed Packet Switching for Local Computer Networks", Communications of the ACM, Vol 19, No. 7, July, 1976.

Table 1. Transmission Rate and No. of Paths

Technology	Transmission Rate on Each Path (Megabits/sec)	Number of Physical Paths
Ring	4	1
Cable-Bus	4	1
Common Data Buffer (CDBFR)	32	1
Circuit Switching (DDN)	3	2,4,8

Table 2. Mean Message Time in System (in ms)

	System Arrival Rate (in Megabit/sec)	Number of Nodes 8	16	32
Cable Bus	.9	24.	-	-
	1.2	-	26.	26.
	1.8	35.	-	-
	2.4	59.	57.	57.
	2.7	104.	-	-
	3.0	-	58.	54.
Ring	.6	25.	24.	23.
	1.2	30.	27.	26.
	2.4	87.	59.	49.
Shared Memory	4.5	37.	-	-
	6.0	-	35.	36.
	9.0	75.	-	-
	12.0	-	49.	41.
	12.6	362.	-	-
	16.8	-	83.	48.
Circuit Switched (Double Buffered)	.6	17.	-	-
	1.2	-	17.	-
	1.8	20.	-	-
	2.4	-	-	17.
	3.6	37.	20.	-
	7.2	-	31.	20.
	14.4	-	-	29.
Circuit Switched (Single Buffered)	.6	44.	-	-
	1.2	61.	43.	-
	1.8	140.	-	-
	2.4	-	57.	46.
	3.6	-	107.	-
	4.8	-	-	62.
	7.2	-	-	108.

Table 3. Throughput (in 10^4 16-bit words)

	System Arrival Rate (in Megabit/sec)	Number of Nodes 8	16	32
Cable Bus	.9	6.3	-	-
	1.2	-	8.2	8.2
	1.8	12.3	-	-
	2.4	16.4	16.4	16.4
	2.7	18.5	18.5	18.5
	3.0	-	20.3	20.3
	∞	20.3	20.3	20.0
Ring	.6	4.1	4.1	4.1
	1.2	8.2	8.2	8.2
	2.4	16.4	16.4	16.4
	∞	18.6	20.4	21.3
Shared Memory	4.5	36.	-	-
	6.0	-	42.	42.
	9.0	62.	-	-
	12.0	-	84.	84.
	12.6	82.	-	-
	16.8	-	116.	116.
	∞	94.	160.	192.
Circuit Switched (Double Buffered)	.6	4.1	-	-
	1.2	-	8.2	-
	1.8	12.3	-	-
	2.4	-	-	16.4
	3.6	24.6	24.6	-
	7.2	-	47.3	49.2
	14.4	-	-	98.4
	∞	32.4	64.8	129.6
Circuit Switched (Single Buffered)	.6	4.1	-	-
	1.2	8.2	8.2	-
	1.8	12.3	-	-
	2.4	-	16.4	16.4
	3.6	-	24.6	-
	4.8	-	-	33.8
	7.2	-	-	49.1
	∞	13.9	28.1	54.6

Table 4. % Throughput

	System Arrival Rate (in Megabit/sec)	Number of Nodes 8	16	32
Cable Bus	.9	25.	-	-
	1.2	-	33.	33.
	1.8	49.	-	-
	2.4	66.	66.	66.
	2.7	74.	74.	74.
	3.0	-	81.	81.
	∞	81.	81.	80.
Ring	.6	16.	16.	16.
	1.2	33.	33.	33.
	2.4	66.	66.	66.
	∞	74.	82.	85.
Shared Memory	4.5	16.	-	-
	6.0	-	21.	21.
	9.0	31.	-	-
	12.0	-	42.	42.
	12.6	41.	-	-
	16.8	-	58.	58.
	∞	47.	80.	97.
Circuit Switched (Double Buffered)	.6	11.	-	-
	1.2	-	11.	-
	1.8	33.	-	-
	2.4	-	-	11.
	3.6	65.	33.	-
	7.2	-	66.	33.
	14.4	-	-	66.
	∞	86.	87.	86.
Circuit Switched (Single Buffered)	.6	11.	-	-
	1.2	22.	11.	-
	1.8	33.	-	-
	2.4	-	22.	11.
	3.6	-	33.	-
	4.8	-	-	22.
	7.2	-	-	33.
	∞	37.	38.	36.

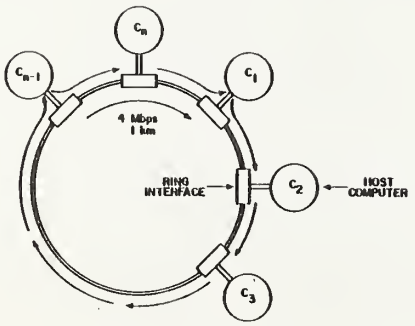


Figure 2. Ring

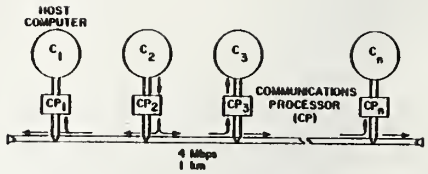


Figure 3. Cable-Bus

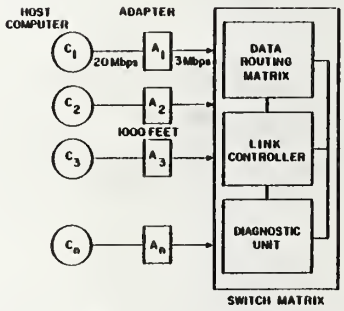


Figure 4. Circuit Switching

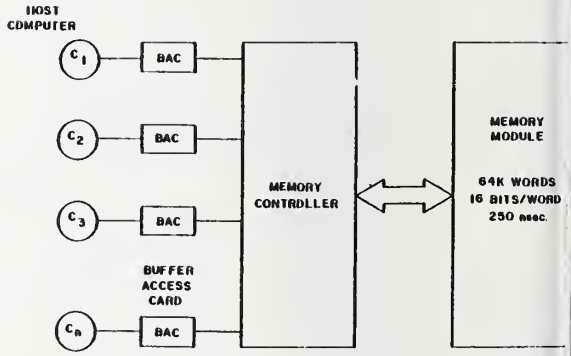


Figure 5. Shared Memory

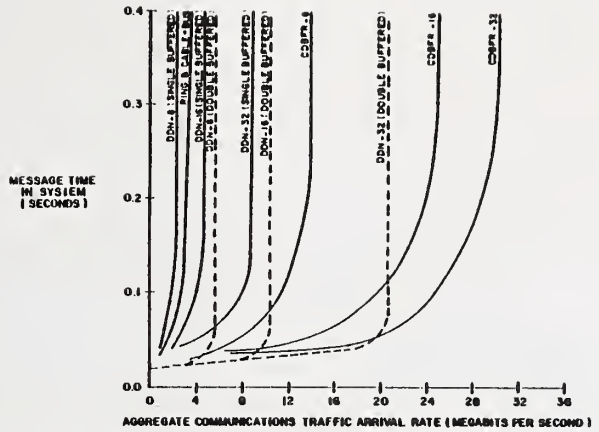


Figure 6. Comparative Performance

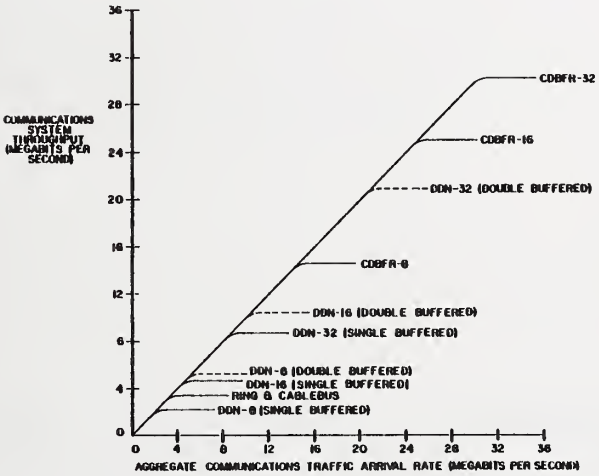


Figure 7. Comparative Thoroughput

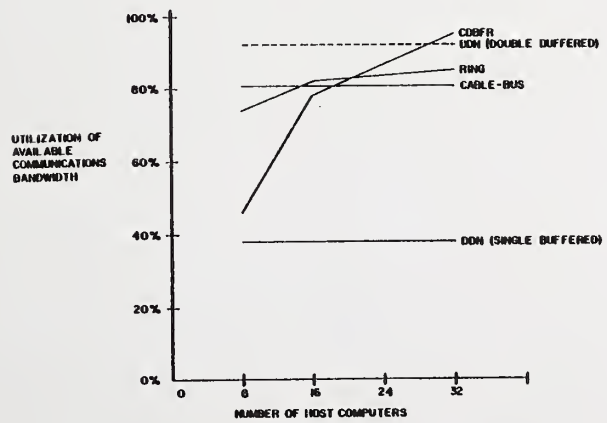


Figure 8. Comparative Efficiency

CPEUG80

Performance Prediction Techniques - 1

Some Properties of a Simple Deterministic Queuing Model

Rollins Turner

Digital Equipment Corporation
Systems Performance Analysis Group
Maynard, Massachusetts

Abstract

A simple queueing network with fixed service times is defined. This network might be an appropriate model for two asynchronous devices sharing the use of a single resource, where all service times are fixed. An algorithm permitting efficient computation of system behavior is described. Certain general properties of the system are determined.

Key words: Fixed service time; queueing network.

1. Introduction

There are many components in computer systems that perform some action in a fixed amount of time. Examples include memory subsystems, synchronous busses, and communications links. When such components are included in a queueing network model of a higher level system they are most accurately represented as servers with fixed service times. Unfortunately, most of the results of queueing theory do not apply to fixed service times. Useful results can often be obtained by substituting a probability distribution for the fixed time but important characteristics will sometimes be lost. Operational analysis can be used in some cases, but again some important questions cannot be answered [1,2]. In this paper we examine one particularly simple queueing network with fixed time servers, and develop from first principles some of its properties. We mention in passing that work reported here was originally motivated by the need to explain some counterintuitive results from measurements on actual computer system components. The model developed in this paper does explain those results, which will be discussed later in the paper.

The model is of some interest as a mathematical object in its own right. It has a number of rather surprising properties, including discontinuities in its response time functions. This model can provide some interesting tests for conjectures about queueing networks with no restrictions on service times, both in stochastic analysis and operational analysis. It should be considered as a potential source of counterexamples when a general theorem is under study.

The remainder of this paper is divided into three sections. Section 2 defines the model and lists some of the questions to which it can be applied. Section 3 discusses the behavior of the system and how it depends on the service times. An algorithm is given by which one can efficiently compute the time from the completion of a wait until the next wait. Using this algorithm, one can compute essentially all quantities of interest about any specific system. In Section 4 we look at certain general properties of this model, treating the service times of the servers as variables. A graphical representation of systems in a two dimensional parameter

space provides some useful insights into the systems' behavior.

2. Definition of the Model

The model that we shall examine is a simple closed queueing network with fixed service times. The network consists of a single central server, designated Server C, and two peripheral servers, called Server A and Server B. There are two customers in the system. Customer A visits Server A, then proceeds to Server C, where he may have to wait for service. After receiving service from Server C, Customer A returns to Server A. Customer B follows a similar pattern between Server B and Server C. Queueing can occur only at Server C, and the wait time can never exceed one service time for Server C. A diagram of the system is shown below in Figure 1.

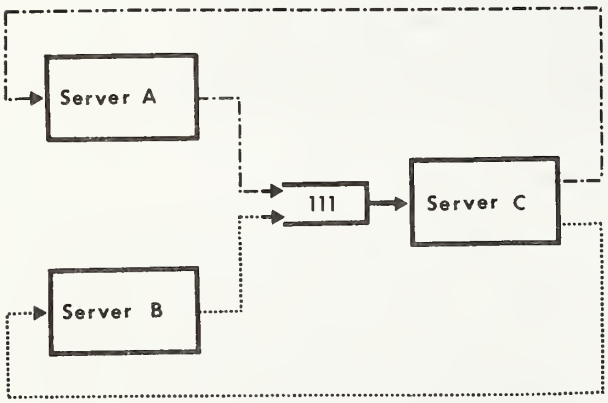


Figure 1. Simple Deterministic Queueing Network

This system might serve as a model for two IO devices sharing a bus, or perhaps for a processor and a single IO device sharing access to a memory. Because the service times are fixed, most of the results of queueing theory do not apply. We also note that the "routing homogeneity" requirement of operational analysis is not satisfied.

The system is characterized by three parameters, the service times of the three servers. These times are called a, b, and c. The state of the system is specified by where each customer is and how long he has been there. Given the state of a system, its future behavior is completely determined. The time of all future events and the state of the system at any time in the future can be computed by means of a step by step calculation of system events.

Given that the system is deterministic, one might ask whether there is anything interesting to be learned from a mathematical analysis. The answer is yes. This simple system turns out to have some surprisingly complicated properties, and there are a number of questions, possibly of practical importance, whose answers are not immediately obvious. Some of the questions discussed in this paper are the following:

1. Given a starting state, which customer will first have to wait and when?
2. What will be the service rate for each customer and utilization of each server?
3. How does a change in the service rate of one of the peripheral servers affect the overall system behavior?

3. Behavior of a Specified System

We know that the future behavior of any given system can be determined by direct computation. Furthermore, after an initial transient, the system state is always periodic. To show this, we need to define two particularly useful system states. Let State A be the state that occurs when Customer A has just begun service at Server A and Customer B has just begun service at Server C. Similarly let State B be the state that occurs when Customer B has just begun service at Server B and Customer A has just begun service at Server C. One of these two states will inevitably follow whenever any queueing occurs.

Suppose then that some queueing occurs. Since one of the two possible states follows each wait, a state is sure to be repeated after no more than two waits. Since the future behavior is determined given the state at any instant, the system behavior is periodic. If there is no queueing, the two peripheral servers must be synchronized, so that Customer A completes m cycles in exactly the same amount of time in which Customer B completes n. Hence after m

cycles by Customer A, and n by Customer B, the original system state will be repeated.

Since the system behavior is periodic, we only need to compute the times of successive system events until we see an earlier state repeated. We know then that the sequence of events between the two occurrences of that state will be repeated indefinitely. However, it is possible to have an arbitrarily long sequence of system events between repetitions of a system state. It is desirable to have a computationally efficient algorithm for determining when a state will be repeated. Algorithm CYCLES, shown below, fills this requirement. The following paragraphs provide some background and motivation for the development of this algorithm. They do not, however, give a proof of its correctness.

In the following discussion "a" and "b" are assumed to be greater than "c". (If either "a" or "b" is smaller than "c", the system is relatively straightforward.) Suppose we begin observing a system in State A. Let us define an "A-Cycle" as the sequence of events:

1. Customer A begins service at Server A.
2. Customer A completes service at Server A and goes to C.
3. Customer A completes service at Server C and returns to A.

Similarly, let us define a "B-Cycle" as the sequence:

1. Customer B begins service at Server C.
2. Customer B completes service at Server C and goes to B.
3. Customer B completes service at B and returns to C.

(Note that each type of cycle is defined as if we begin observing the system in State A.)

If a B-Cycle ends within c time units prior to the end of an A-Cycle, Customer B will find Server C busy and will have to wait. Upon completion of service by Server C, the system enters State A. If a B-Cycle ends at a time greater than c and less than 2c prior to the end of an A-Cycle, Customer B will find Server C idle and begin service immediately. Upon completion of service by Server A, Customer A will find Server C still busy with Customer B and will have to wait. Upon completion of service for Customer B the system enters State B. Thus after beginning in State A, the queueing will first occur when a B-Cycle ends prior to the end of an A-Cycle by an amount less than 2c.

To determine the first occurrence of either State A or State B following State A, we must find the smallest values of positive integers m and n that satisfy the equation:

$$0 = m(a+c) - n(b+c) \leq 2c$$

If m' and n' are the solutions we define the variable d as:

$$d = m'(a+c) - n'(b+c)$$

We see that d can be computed as REM(m'(a+c), (b+c)) where REM is the remainder of the first argument divided by the second.

If d is exactly 0 the system is conflict free, with State A returning to State A after m' A-Cycles. If d is in the range between 0 and c, then State A returns to State A after m' A-Cycles, with Customer B waiting for d time units. If d is in the range from c to 2c, then State A goes to State B after m' services by Server A, with Customer A waiting for (2c-d) time units. Customer A makes m'-1 visits to Server C during this interval. By reversing the roles of a and b we can determine which state will follow State B and when, in the same way we have just done for State A.

Algorithm CYCLES can be used to compute the value m' for any given values of a, b, and c. The inputs to CYCLES should be (a+c)/c and (b+c)/c. CYCLES returns the value of m'. From m' we can compute n' and d, from the above equations.

Algorithm CYCLES returns a value that is the number of times Customer A will visit Server A prior to one or the other customer's having to wait, given that the system started in State A. Hence it gives the time from State A until the next occurrence of either State

A or State B, whichever comes first. To determine what happens following State B we can use the same algorithm reversing the two arguments. (The entire system is symmetrical in terms of A and B.)

It can be seen that Algorithm CYCLES will always terminate with no more than LOG2((b+c)/c) recursions. On each successive call, the second argument will be no more than half the value of the preceeding call. When the second argument is less than two, the algorithm terminates.

Algorithm CYCLES (A,B)

Let A1 = REM(A,B)

If A1 $\leq$ 2 then Return 1

Else if A1 $\leq$ B/2 then

$$\text{Return} \left\lceil \frac{\text{CYCLES (A1 - REM(B,A1),A1) * B}}{\text{A1}} \right\rceil$$

Else

$$\text{Return} \left\lceil \frac{\text{CYCLES (REM(B,B-A1),B-A1) * B - 2}}{\text{B-A1}} \right\rceil$$

4.0 General Properties

Having seen how to determine the future behavior of any sepcific system, we turn our attention to general properties of all such systems. One question of interest is "For what values of a, b, and c is the system conflict free?" The answer is provided by the following theorems.

Theorem 1. For any real c if

a = (2n-1)c
b = (2m-1)c

where m and n are positive integers then the system is conflict free.

Proof: Suppose the system starts in State A. Then until the first conflict, Customer A's arrivals for service by Server C come at multiples of (a+c) minus c. Customer B's arrivals come at multiples of (b+c). Since both a and b are multiples of c, requests for Server C come only at multiples of c. Thus it is impossible for the first wait to occur due to a customer's arriving at Server C during service for the other customer. The only possibility is that both customers arrive at Server C simultaneously. This will occur if and only if there are positive integers i and j such that

i(2nc) - c = j(2mc)

or 2ni - 1 = 2mj

But this has no solution, because the left side is odd and the right side is even for any choice of i and j. Hence no conflict can occur following State A. A similar argument shows that no conflict can occur following State B.

For initial conditions other than State A or State B, a wait may occur. However, following a wait the system will have to enter either State A or State B. Hence after the initial transient no conflict can occur.

[Q.E.D.]

Theorem 2. For any real c if

a0 = (2n'-1)c
b0 = (2m'-1)c

and if m' and n' are relatively prime, then they are the smallest positive integers, m and n, for which

m(a0+c) = n(b0+c)

Proof: Suppose there are smaller values m" and n" for which

m"(a0+c) = n"(b0+c)

then

m"(2n') = n"(2m')

$$\frac{m''}{n''} = \frac{m'}{n'}$$

But this is impossible since m" and n" were assumed to be smaller than m' and n', and m' and n' were assume to be relatively prime.

[Q.E.D.]

For a system described by the parameters in Theorem 2, the overall cycle consists of m' A-Cycles in parallel with n' B-Cycles. During this overall system cycle there will be no conflicts, and State A will lead to another State A after m'(a0+c) time units. However there will be one "near miss", as shown by the next theorem.

Theorem 3. Let

$$a0 = (2n'-1)c$$
$$b0 = (2m'-1)c$$

with m' and n' relatively prime. Then at some point during the overall system cycle starting with State A, Customer A will arrive for service at Server C just as Customer B is completing service. Thus State A leads to State B at this point, and State B will lead to State A.

Proof: Customer A's i'th request for service at Server C will come at time i(2nc)-c. Customer B's j'th request will come at time j(2mc). Customer A will arrive at Server C just as Customer B is leaving if and only if there are positive integers i and j such that the following equation is satisfied:

$$i(2nc) - c = j(2mc) + c$$

$$i(2nc) - j(2mc) = 2c$$

$$in - jm = 1$$

Since n and m are relatively prime, 1 is their greatest common divisor. We know from Euclid's algorithm [3] that there are positive integers i and j that satisfy the equation. Thus for the system described, State A leads to State B after i services by Server A. Reversing the roles of A and B we can compute the number of services by Server B after which State B leads to State A. During the entire system cycle there are no conflicts.

[Q.E.D.]

4.1 Graphical Representation

In characterizing systems in terms of the parameters a, b, and c, it is useful to have a graphical representation of the systems. In the following discussions we let our time unit be c. With this convention, we can represent any system by two parameters, a and b. Thus we can represent any system as a point on an a-b plane.

What we have seen so far is that there is an array of conflict free points in the a-b plane. These are the points whose coordinates are positive odd integers. If the odd integers are relatively prime, the point has the additional property of directly specifying the number of A-Cycles and B-Cycles in the overall system cycle. Next we shall see that each of these special points is the beginning of a line, all points of which represent conflict free systems. Furthermore all systems represented by points on this line have the same pattern in their overall system cycles. In the following theorem we see how to determine the line segment of conflict free points given a minimal conflict free point (a0,b0).

Theorem 4. Let a0 and b0 be given by

$$a0 = (2n-1)c$$
$$b0 = (2m-1)c$$

where n and m are positive integers, and are relatively prime.

Then the system represented by

$$a1 = a0 + \Delta a$$
$$b1 = b0 + \Delta b$$

with $\frac{\Delta a}{\Delta b} = \frac{n}{m}$

is conflict free if and only if Δa and Δb are greater than zero.

Proof: Starting from State A, until a conflict occurs, Customer A's i'th request for service at Server C comes at $i(a0+\Delta a+c)-c$. Customer B's j'th request comes at $j(b0+\Delta b+c)$. A conflict will occur if and only if there are positive integers i and j such that

$$-c < [i(a0+\Delta a+c)-c] - [j(b0+\Delta b+c)] < c$$

$$-c < [i(2nc+n\Delta b/m)-c] - [j(2mc+\Delta b)] < c$$

$$0 < (in-jm)(2c+\Delta b/m) < 2c$$

$$0 < in - jm < \frac{1}{1 + \Delta b/(2mc)}$$

If $\Delta b > 0$, this is impossible because the term on the right is a fraction between 0 and 1 while the term in the middle is an integer. Thus the system represented by (a1,b1) is conflict free.

If, on the other hand, $-2mc < \Delta b < 0$, the term on the right is greater than 1. Since m and n are relatively prime Euclid's algorithm guarantees the existance of integers i and j such that

$$in - jm = 1$$

Hence there is a solution for i and j, and the system is not conflict free. [Q.E.D.]

We see from Theorem 4 that corresponding to each minimal conflict free point (a0,b0) there is an infinite set of additional conflict free points. These are the points along the infinite line segment beginning at (a0,b0) and continuing through the a-b plane with a slope of n/m, or (a0+1)/(b0+1). Figure 2 shows these line segments for a part of the plane. A little more reflection shows that if we restrict a and b to rational values these are the only conflict free points. We note that the lines whose points we can classify as conflict free or not conflict free radiate from the point (-1,-1) in the a-b plane. Each of these "radials" passes through a minimal conflict free point, (a0,b0), and has slope (a0+1)/(b0+1). Points to the right of (a0,b0) are known to be conflict free, while those to the left are known not to be. Now, any point on the a-b plane with rational coordinates is on one such radial. Thus it is conflict free if and only if it is to the right of the corresponding minimal conflict free point. And, from this we see that these radials represent all the conflict free points in the a-b rational plane.

In summary, we have two ways of determining whether a given system is conflict free. We can use Algorithm CYCLES to compute the number of cycles between repeated states. From this information we can compute the variable d, and if it is zero the system is conflict free. The value of d also tells us the pattern of states that will be repeated in the system's long term behavior.

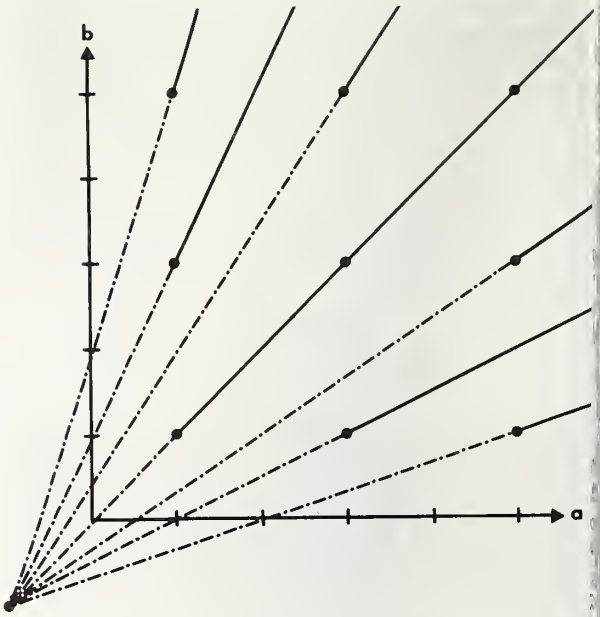


Figure 2. Conflict Free Points

Using the results of Theorem 4 we can determine directly whether or not the system is conflict free. Given a and b for the system (as rational numbers), we express the ratio, (a+1)/(b+1) as a ratio of relatively prime integers, n/m. If $a \geq (2n-1)$, the system is conflict free. Otherwise it is not. So far, however, this approach does not tell us anything about the behavior of the system if it is not conflict free. In the following section, we shall see that the graphical approach is also useful in determining properties of systems that are not conflict free.

4.2 Geometry of Regions Having a Common Pattern

In the following section we discuss regions of the a-b plane in which all points represent systems having a common pattern in their overall system cycles. Complete proofs for the properties claimed for these regions have not yet

been developed. Proofs are given that certain properties are necessary conditions for points to be in the same region, and it is conjectured that these same conditions are also sufficient.

Theorem 5. If (a1,b1) is any conflict free point, with m(a1+c) = n(b1+c), then a point (a1,b2) will have the same pattern from an initial state of State A only if

$$\Delta b = (b1-b2) < \frac{c}{n}$$

Proof: Suppose the system begins in State A and consider the end of the overall system cycle. At this point the system returns to State A with Customer B arriving for service at Server C just as Customer A is leaving. If Customer B had arrived earlier by any amount less than c, he would have found Server C busy and been required to wait for that additional amount of time. The overall pattern would have been unchanged with State A leading to State A after n B-Cycles (and m A-Cycles.) In order for this to occur when b is decremented it is necessary that

$$n\Delta b < c$$

or

$$\Delta b < \frac{c}{n}$$

[Q.E.D.]

Theorem 6. If (a1,b1) is any conflict free point, with m(a1+c) = n(b1+c), then a point (a1,b2) will have State A lead to State B after m services by Server A, only if

$$c/n < (b1-b2) < 2c/n$$

Proof: Again consider the end of the overall cycle. If Customer B arrives earlier by an amount greater than c but less than 2c, then he will find Server C idle. When Customer A arrives for what would have been his m'th service by Server C, he will find Server C busy. Upon completion of service for Customer B, the system will reach State B.

The overall pattern will be the same in terms of number of services by Server A and Server B. The necessary condition for this scenario is

$$c < n\Delta b < 2c$$

or

$$c/n < \Delta b < 2c/n$$

[Q.E.D.]

From Theorems 5 and 6 (and the conjecture that the stated conditions are sufficient as well as necessary), we see that below each radial of conflict free points there are two strips, each of vertical height c/n, in which the overall system cycle is essentially the same as along the radial. An example is shown in Figure 3.

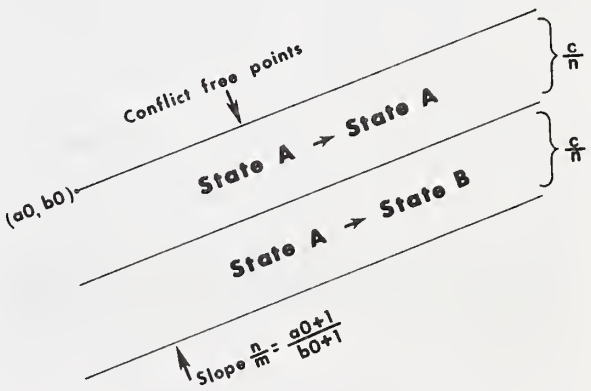


Figure 3. Region With a Single Pattern

These strips extend indefinitely to the right, since the conflict free radial does. However we do not yet know anything about how they are bounded on the left. That is the subject of the next theorem.

Theorem 7. Given a minimal conflict free point (a0,b0) with

$$a0 = (2n-1)c$$
$$b0 = (2m-1)c$$

a point (a2,b2) with a2 < a0 and b2 < b0 (and less than 2c/n below the radial) will have the same pattern as (a0,b0) if and only if

$$\frac{\triangle b}{\triangle a} = \frac{b0 - b2}{a0 - a2} > \frac{i}{j}$$

where in - jm = 1

Proof: It was shown in Theorem 3 that for (a0,b0) Customer A will arrive for service at Server C just as Customer B is leaving following Customer A's i'th service by Server A. At this point Customer B will have completed j visits to Server B. If we decrement both a and b there will be a conflict at this point if Customer A's cumulative savings exceed those of Customer B. Hence in order to avoid a conflict at this point, and a different pattern, it is necessary to have

$$i\triangle a < j\triangle b$$

or

$$\frac{\triangle a}{\triangle b} < \frac{i}{j}$$

[Q.E.D.]

Hence the end boundary for the regions with the same pattern is a line through (a0,b0) with slope i/j, as shown in Figure 4.

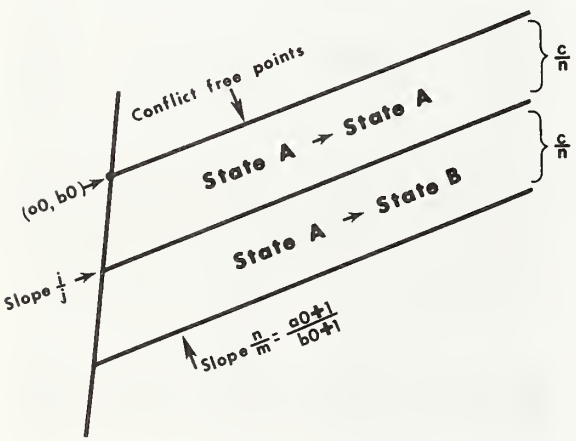


Figure 4. Region With a Single Pattern

Figure 5 shows a number of these regions for a segment of the a-b plane. We note that there are blank areas within the figure. We can never draw a complete map of these regions for any part of the plane that includes any conflict free points. This is because there are an infinite number of regions within any finite distance above a conflict free point. Note that all points in a given region in Figure 5 will have a common pattern, starting with State A and continuing until either State A or State B is reached. To see the regions having a common pattern starting with State B, we would reverse the axes. (This map is not symmetric.) The regions defined by the intersections of regions in these two maps specify sets of points that will have the same pattern starting with either initial state and continuing indefinitely.

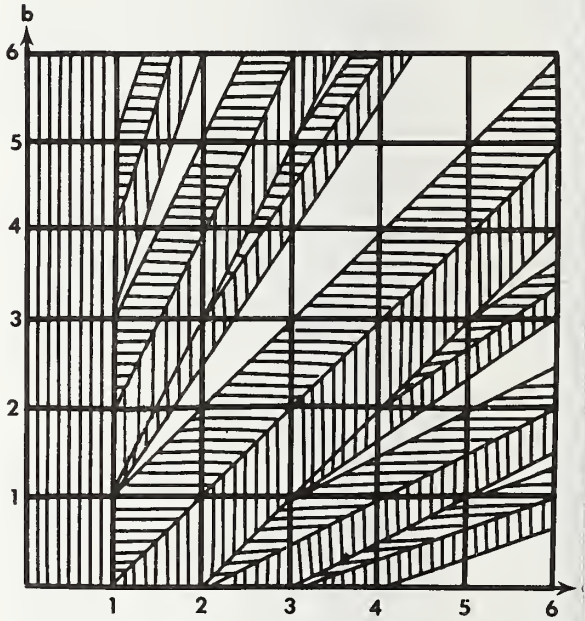


Figure 5. Regions With a Common Pattern From State A to Either State A or State B

It was claimed that there are an infinite number of distinct regions within any finite distance above a conflict free radial. This can be seen by considering what happens if we decrement the value of "a" very slightly from any value for which (a,b) is conflict free. As an example let (a,b) = (3,3), and consider the following timing diagram:

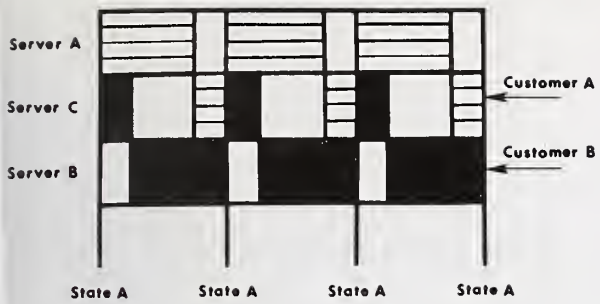


Figure 6. Timing Relationships for a Conflict Free System

We See that Customer A and Customer B are synchronized, with one A-Cycle per B-Cycle. Now if the value of "a" were slightly decremented, Customer A would arrive at Server C slightly earlier on successive cycles. Eventually Customer A would gain enough on Customer B that he would arrive while Server C was still serving Customer B. Thus we would have State A leading to State B after some large number of cycles. It turns out that this is the transient, and that State B will always lead to State B after one cycle. However the transient can be arbitrarily long. The smaller the difference between a and b, the more cycles it takes for Customer A to catch up with Customer B. This is why there are an infinite number of regions in any finite area above a conflict free point in the a-b plane.

4.3 Effects of Changes in Service Time

The final question that we shall address is that of the effect of changes in service time for one of the peripheral servers on the throughput of the other. One might guess intuitively that speeding up one peripheral server would always give its customer an advantage in competition for the central server, and thus could only lower the throughput of the competing customer. Looking at Figure 5, we see that this is not the case. There are some situations in which speeding up one customer will help the competing customer.

For any given value of "b" there are many values of "a" for which the system is conflict free. At any of these points each server operates at the same rate that he would if only his customer were present. Between these conflict free points, there are always points at which Customer B must sometimes wait for Server C. At such points Server B operates with reduced throughput. There are both larger and smaller values of "a" for which Server B's throughput will be higher. Hence to know the effect of a change in either service time it is necessary to look at the specific patterns before and after the change. The effect of a change could be either an increase or a decrease in the throughput of the unchanged server. Measurements showing this effect with actual computer system components were the original motivation for the work leading to this report.

Conclusion

A queueing network model with fixed service times has been defined and studied. We have seen that this deterministic system, which is quite simple to describe, can have rather complex behavior. Unlike most of the commonly used stochastic models, this model exhibits discontinuities in the functions describing performance variables in terms of model parameters. A geometrical description of the parameter space provides useful insight into the general properties of such systems. A computational algorithm permits efficient calculation of the performance of a system with any specific set of parameter values.

Thanks to Jim Bouhana for several suggestions for improvements in clarity, and to an anonymous referee for pointing out several errors in the original draft. Discussions with my colleague Joel Emer led to the original formulation of the model as presented here.

References

[1] P.J. Denning and J.P. Buzen, "The Operational Analysis of Queueing Network Models," Computing Surveys, Vol. 10, No. 3, Sept. 1978, pp. 225-261.

[2] K.C. Sevcik and M.M. Klawe, "Operational Analysis Versus Stochastic Modelling of Computer Systems," Proceedings of the Computer Science and Statistics: 12th Annual Symposium on the Interface, J.F. Gentleman (ed.), University of Waterloo, Waterloo, Ontario, Canada, pp. 177-184.

[3] See any introductory text on number theory.

A Highly Parameterized Tool for Studying Performance of Computer Systems

Herman D. Hughes

Department of Computer Science
Michigan State University
East Lansing, Michigan 48824

ABSTRACT--A highly parameterized simulation model is described which allows experiments to be performed for computer performance evaluations studies. The results of these experiments can be used to evaluate the effect of changing the hardware configuration, the workload, the scheduling policy, the multiprogramming level, etc. The model is constructed to function either as a batch or time-sharing system, or as a combination of both. This simulation model also has the potential of providing dynamic feedback for the scheduler. A discussion of the design, implementation, and use of the model is presented. Examples are provided to illustrate some possible uses of the model and verifications of the results obtained from the model.

Key words: Simulation model, queue, scheduling policies, workloads, hardware configuration, model validation, system performance, events, cumulative distribution function.

*Research supported jointly by AFOSR Grant 78-3547 and Division of Engineering Research

1. Introduction

In order to provide sufficient information for evaluating changes to computer systems, both the hardware and software must be evaluated with respect to the efficiency in performing their required tasks. There are certain realistic constraints which make it virtually impossible to effect changes to existing systems for the purpose of studying computer system performance. Many of these constraints, however, may be overcome by the use of a flexible computer simulation model [2].

An emphasis of this investigation is to focus on providing a tool for assisting analysts in making decisions on various performance strategies. In order to develop such a tool, it is obvious to this investigator that a fairly complex model of a computer system is required.

The number of existing simulation models cited in the literature for computer systems and computer subsystems is too massive for an adequate review. However, it should be mentioned that a large percentage of the previous work related to computer system simulations has focused on answering questions relative to specific computer systems/computer subsystems.

This paper describes a highly parameterized simulation model which allows experiments to be performed for which the hardware configuration, the workload, and the scheduling policy can vary. The model is event-driven and is designed to accommodate systems as simple as batch with uniprogramming, to more complex systems which make use of time-sharing, multiprogramming, and virtual memory principles. Major components of the model are described in the next section of this paper.

Several experiments are presented to illustrate the potential use of the simulation model. Typical output from the model includes: performance indicies (i.e., response time, throughput rate, execution-time dilation, paging rate, swapping rate, etc.), queue statistics, utilization measures, and a profile of the system. This output is available at the job-step level or at the overall system level, and is broken down by system overhead and users' statistics.

The model may serve as a tool for providing guidance to system analysts, capacity planners, and individuals involved in courses such as system programming, operating systems, simulation, and performance measurements/evaluations.

II. DESCRIPTION OF MODEL

The model is written in a high-level language--ANSI standard FORTRAN--and is implemented on a CDC Cyber 750 computer. There are several components to the model, and each component corresponds to a FORTRAN subroutine. These components and their functions will be examined after a discussion of the flow of transactions through the system.

The high level flow of jobs (job-steps) through the system is depicted in Figure 1. Each job-processing step listed below corresponds to an event within the model [7].

Step 1:
A job (batch or interactive) arrives randomly or according to a specified distribution. Upon arrival, the following job characteristics are determined either randomly or according to a pre-defined distribution: (1) the total CPU time, (2) the average amount of central memory (CM) requested, and (3) the number of I/O requests.

Step 2:
The job makes a request for CM allocation. If the CM space requested is not available, the job enters the CM queue.

Step 3:
After the job enters the CM, it immediately requests the CPU. If the CPU is free, it is assigned to the job and executes until some blocking condition occurs (i.e., a system interrupt, the time-slice used up, the job is completed, or an I/O request is encountered). In the former two cases, the job releases the CPU, but is placed back into the CPU queue.

Step 4:
When a job issues an I/O request, the CPU is released, and a specific disk is requested. Since the total CPU time and the number of disk requests for a job are predetermined, it is assumed that the instances of I/O are uniformly distributed over the elapsed time of a job, when it is run in a uniprogrammed mode.

Step 5:
In order for a job to access a designated disk, both the disk and the associated channel must be free. Otherwise, the job enters a disk queue. If the disk and the channel are both free, a "disk-seek" time is generated. During the "disk-seek" time, the disk is busy, whereas the channel is not.

Step 6:
After completing the disk-seek, a "rotational delay" time is generated. When this time expires, the channel is requested again, and if available, the data is transferred over the channel. The disk and the channel are both busy during the "transfer" time.

Step 7:
When the data transfer is completed, the disk and the channel are both freed, and the job proceeds to request the CPU again.

Step 8:
Upon completing all the CPU and I/O tasks for a given job, the CM allocated for that job is released. If the job is a batch job, it leaves the system; otherwise, the job is an interactive job, and has just completed a "system response cycle", so a "user think-time" is generated.

III. JOB EVENTS

The job-processing steps listed by Steps 1-8 represent only a subset of the events within the model. Other events included in the model are highlighted by Figure 2. Those events which appear in the flowchart boxes have event-times which are predetermined and, therefore, can be placed on the future-event list. The set of events whose event-times cannot be determined in advance are just listed below the flowchart(refer to Figure 2). For example, if a batch job is in the CM-queue, the next event is requesting CM; but since it depends on when other jobs will leave the system and make space available, the event time cannot be determined. On the other hand, if a job obtains the CPU at time t_o, and the inter-

I/O request time T is known, then the next event for this job (release the CPU) can be scheduled at time t_o+T, and hence placed on the future-event list.

IV. QUEUE STRUCTURE FOR MODEL

The model consists of several queues

(i.e., future-event queue, CM queue, CPU queue, channel queue, free-record queue, disk queue, etc.). Each of these queues forms a ring with a coincident head and tail. Records in the queues are constructed as doubly-linked lists with pointers to the immediate predecessors and successors.

Job-records in the future-event queue are always in ascending order of the next event-time, whereas jobs in each waiting queue are always in decending order of job priority. The queue discipline, FCFS, is applied to jobs of the same priority.

Figure 3 illustrates a typical queue structure for job-records within the model. Notice that a job (job-record) can only appear in one of the queues at a time, and that records which are not currently active are attached to the free-record queue.

The average queue length ($\bar{\ell}n$) can be derived as follows:

$$\bar{\ell}n = [\ell_o(t_1-t_o)+\ell_1(t_2-t_1)+\ldots +\ell_{n-1}(t_n-t_{n-1})]/(t_n-t_o)$$

$$=[-\ell_o t_o+(\ell_o-\ell_1)t_1+(\ell_1-\ell_2)t_2+\ldots +(\ell_{n-1}-\ell_n)t_n+\ell_n t_n]/(t_n-t_o)$$

$$= [\sum_{i=1}^{n} (\ell_{i-1}-\ell_1)t_i+\ell_n t_n]/t_n$$

$$\bar{\ell}_n = [\sum_{i=1}^{n} (\ell_{i-1}-\ell_i)t_i]/t_n+\ell_n$$

where $\{t_i\}$ $(o \le i \le n)$ denotes the time when a job enters or leaves the queue, and ℓ_i is the queue length at time t_i, with $t_o=o$. This formula was used throughout the model for calculating average queue lengths.

V. THE GENERATION OF DISTRIBUTIONS

Distributions used by the model are generated via a set of Cumulative Distribution Functions (C.D.Fs). The C.D.Fs are defined by the user supplying a set of discrete functions (e.g., 11 points), thus permitting the generation of a desired distribution. As an example, assume that the job arrives according to Poisson distribution with mean $\lambda = 3$ job/sec. Then, the inter-arrival time (random variable X) is known to have an exponential distribution of mean $\theta=1/3$, hence the C.D.F. function can be approximated as follows:

$F[X \le t] = 1-e^{t/\theta} = 1-e^{3t}$, where $F[X \le t]$ takes on the values 0.0, 0.1, 0.2, ..., 0.9, 1.0:

$F[X \le t]$	0.0	0.1	0.2	0.3
t(approx.)	0	0.0351	0.0744	0.1189

0.4	0.5	0.6	0.7	0.8
0.1703	0.2310	0.3054	0.4013	0.5365

0.9	0.9995*
0.7675	2.534

*Here, since $F[X \le t]$ equals 1.0 only when $t \to \infty$, we use the value 0.9995 in order to avoid $t=\infty$. The approximated C.D.F. is shown in Figure 4(a).

After the C.D.F. has been approximated, and a sample is desired from this distribution, we only need to generate uniformly distrbuted random numbers over the unit interval (0,1), and perform an inverse transformation on the C.D.F.. This transformation involves a table look-up and a linear interpolation procedure to obtain sample values. A possible pitfall of this approach is that when a discrete function is used to approximate a continuous function, some error must be tolerated. Figure 4(b) illustrates the case where a value x' may be generated which is slightly larger than the actual value x. Clearly, as the number of points used to approximate a function increases, the accuracy of the sample values also increases.

Another problem related to the generation of various distributions in modelling is the independence of the set of random numbers generated for different distributions. For this model, the independence of the generation of jobs within job-classes was achieved by using a different random-number seed for generating each job-class.

VI. MODELLING VARIOUS COMPONENTS OF SYSTEM

HARDWARE (see Figure 1) - The following hardware is modelled, and can be configured in various ways.

CPUs
DISKS
CHANNELS
TERMINALS
MEMORY

The timings of the hardware components are relative, and may be redifined by the user.

SOFTWARE - In order to simplify the model, the details of the operating system are not modelled explicitly; instead, tim-

ings for system overhead are included in system tasks (i.e., paging, swapping, scheduling, etc.) [1].

PAGING - Paging is modelled as a high-priority system job which is activated at a certain rate (specified as a parameter by the user). This high-priority job will use the CPU, channel, and the disk to read/write one page of data. By using this approach for modelling paging, the contention for devices can be simulated very easily.

The paging rate is defined for some fixed multi-programming level (i.e., MPL=7), and will be varied by the model as a function of: (1) the number of interactive users, (2) the memory size, and (3) the MPL level.

SWAPPING - Swapping is not modelled explictly; instead it is modelled as a high-priority system job which is activated when (1) TTY jobs get into or out of think-time (CM is actually freed), or (2) jobs request disk I/O, but are blocked. An input parameter controls the swapping rate. If the CM-queue length is greater than this input parameter, swapping occurs.

The system resources used by swapping are: the CPUs, disks, and the channels.

STORAGE ALLOCATION - the acquisition and release of main storage for the application programs are modelled. The user specifies the memory size via input parameter.

SCHEDULING - Originally, jobs coming from the same class (batch, system, or interactive) are assigned the same priority (specified as a parameter). This convention may be altered if it is desired to assigned different priorities to jobs within the same class. Each time a job changes queues, its priority is recalculated. The calculation proceeds as follows:

Internal priority = original priority + (CPU time used) x $weight_1$ +(system residence time) x $weight_2$ + (CM size) x $weight_3$, where $weight_i$ are input parameters.

By altering input parameters such as initial job priority, internal priority weight, quantum size, MPL, etc., different scheduling algorithms can be investigated. Since the model collects statistics such as queue lengths, and utilization information, the results of these statistics can be used to provide dynamic feedback to the scheduler 3 .

WORKLOAD - Each batch job-class is characterized by its CM request, CPU time, and number of disk I/O requests. An interactive job is characterized by its CM request, CPU time, number of disk I/O requests, and the length of its think-time. These job characteristics are defined by distribution functions. For example, a user's think time may be simulated by sampling from an exponential distribution with mean = 16 [5]. An approach for generating representative work load data to drive a simulation model will be discussed in a forthcoming paper.

VIII. MODEL VALIDATION AND EXPERIMENTAL RESULT

It is an established fact that the validation of a simulation model is a complex process [4]. This model was validated by (1) verifying the logic of the FORTRAN program, (2) using a constant model to verify the accuracy of the statistics produced by the simulation model, and (3) using stochastic processes to check the correctness of the simulation. Since the first two steps used for validation are straightforward, the results which make use of stochastic processes (step 3) are presented in experiments which follow.

Experiment 1.

no. of CPUs = 1
no. of disks = 1
no. of channels = 1
size of CM = 300K words
multiprogramming level = 1 (uniprogramming)

The job-parameters were:

mean arrival rate λ= 3 jobs/second (Poisson distributed) from a single batch class.

mean CPU service time μ_1 =0.1 sec/job
mean disk service time μ_2 = 0.05 sec/job
avg. CM request per job = 50K words

The above three job-parameters are distibuted exponentially.

The system is depicted in Figure 5. A job enters the system only when there is no other job running. It uses one half of the CPU time, then leaves the system. It should be noted that the channel is not modelled here, since no contention exists for it in a uniprogramming mode.

The simulation results are tabulated in Table 1 along with the analytic results. An explanation for the calculations made in rows 1-6 of Table 1 is given below.

1. Row 1-- λ is given as an input parameter. (λ=3)
2. Row 2--the CPU utilization ($\cup$CPU) is calculated as;
   $\cup CPU = \lambda \mu_1 = 3.0 \times 0.1 = 0.3$

3. Row 3--the disk utilization ($\cup$ disk) is calculated as:
$\cup$ disk = $\lambda \mu_2$ = 3.0 x 0.05 =0.15

4. Row 4--the calculation for the turnaround R, is based on a M/G/1 system.

$$R = E(\text{service time}) + \frac{\lambda E(\text{service time}^2)/2}{1-\lambda E(\text{service time})}$$

$$= (\mu_1+\mu_2)+\frac{\lambda}{1-\lambda(\mu_1+\mu_2)}[(\mu_1+\mu_2)^2-\mu_1\mu_2]$$

$$= (0.15)+\frac{3}{1-0.45}[(0.15)^2-(0.1)(0.05)]$$

$$= 0.15 + 0.095 = 0.245 \text{ seconds}$$

5. Row 5--avg. CM queue length = total wait time/total time

=(turnaround time - service time) x (no. of job completion)/total time

=(0.245-0.15) x throughput
≃ 0.095 x 3.0=0.285

It should be noted that the avg. CM queue length may be thought of as an "occupancy factor" for the CM queue.

6. Row 6--the average number of jobs in the system (excluding the CM queue) is, by Little's Law), [E(service time) x throughput]. So, the average

CM utilization =[E(service time) x throughput]x E(CM request)/CM size
=(0.15) x (50) x 3.0/300
= 7.5 x 3.0/300
= 0.075

Also, the Operational Approach proposed by Buzen [8] can be used to check the consistency of the model as follows:

Little's law states that the average number ($\bar{n}$) of jobs in the system, including those waiting in the CM-queue, is given by:

$\bar{n}=xR$ where x = throughput
R = turnaround time.

Hence, we have

$R=\bar{n}/x$

Now, $\bar{n}$= avg. CM queue length + $\cup$ CPU + $\cup$ disk

= 0.184 + 0.296 + 0.149
= 0.629 (in the 400-job case) and
x≃ 2.48, hence

$R=\bar{n}/x \simeq 0.629 \div 2.48 = 0.255$ seconds, which is close to the result for the 400-job case.

Experiment 2.

To further validate the model, let's suppose that during a job's access to the disk, a disk-seek time and a latency (rotational delay) time were generated. Consider the following configuration:

no. of CPUs = 1
no. of disks = 1
no. of channels = 1
size of CM = 300K words
multiprogramming level = 1 (uniprogramming)
The job parameters were:

mean arrival rate $\lambda = \frac{1}{3}$ jobs/second (Poisson distributed) from a single batch class.

avg. disk-seek time = 0.04 seconds
avg. latency = 0.01 seconds

The above two job-parameters are distributed exponentially.
Figure 6 depicts the system of concern, and the results are shown in Table 2.

Considerably more validation was done on the simulation model, but will not be presented in this paper [4]. The remaining experiments are presented to illustrate some of the more interesting outputs from the model. Appendix I contains an example of a system profile produced by the model. A system profile enables one to observe the degree of overlap of resource utilization during a selected time interval.

Experiment 3.

This experiment is to investigate the effects of varying the quantum (time-slice) size and observe the performance of a multiprogramming computer system. The system under study has the following configuration:

no. of CPUs = 1
no. of disks = 8
no. of channels = 2; 4 disks/channel
size of CM = 128K
multiprogramming level = 10
system overhead due to job-swapping = 2 to 3 msec.

The impact of various quantum sizes on the system's behavior is plotted in Figure 7. Basically, for quantum sizes of 1.0 to 0.3 seconds, the system performs much the same, because the average inter-I/O time is relatively small compared to the quantum sizes. As the quantum size decreases to 0.08 seconds, the turnaround time and the CPU queue length are considerably reduced, hence we get better performance from the system.

However, if the size of the time-slice gets too small, the system overhead in creases significantly, and therefore degrades the system performance. So this simulation can provide some guidelines for determining the size of the time-slice.

Experiment 4.

In order to analyze the effects of different multiprogramming level (MPL) on the system performance, a set of experiments were performed with various MPLs. The general configuration is depicted in Figure 1, with the following specificiations:

no. of CPUs = 1
no. of disks = 8
no. of channels = 2; 4 disk/channels
size of CM = 128K
quantum size = 0.1 second

disk-seek time = 0.04 seconds
rotational delay = 0.01
disk service time = 0.2 seconds

The above three job-parameters are distributed exponentially.

Batch jobs (jobsteps):

mean arrival rate λ = 1/2.8 jobs/second (Poisson distributed)

mean CPU service time = 2.0 seconds
avg. no. of disk I/O = 5 times

The above two job-parameters are distributed exponentially.

Interactive jobsteps:

no. of terminals = 10

user think-time, Z = 18 seconds
mean CPU service time = 0.2 seconds
avg. no. of disk I/O = 2 times

The above three are distributed exponentially.

The system also has paging and swapping overhead as explained in previous sections.

Figure 8 shows the plot of the MPL vs system performance in terms of batch turnaround-time, TTY response time, system overall throughput, and system overhead, etc.

Experiment 5.

Suppose that the system is now dedicated to interactive users, and we wish to study the behavior of the response-time as the number of terminals increases. The system has the same configuration as described in Experiment 4, except that the MPL is set at 7. Workloa characteristics for this experiment are described by the following parameters:

mean CPS service time per interaction = 0.
seconds
avg. no. of disk I/O requests = 2 times
user think-time = 18 seconds

The above three job-parameters are distributed exponentially

Figure 9 shows the various performance indice obtained as a result of varying the number o terminals.

VIII. SUMMARY AND CONCLUSION

We have presented a general-purpose simulation model which is capable of simulating a wide variety of computer systems. The majo advantages of this model can be characterize as the following:

i. the structure of the model is genera enough to be tailored for many compute systems, and yet,
ii. the model is highly parameterized so tha it can closely approximate a rea system by specifying the hardware an software configurations;
iii. the (batch and interactive) workload that drive the simulation model ca also be defined handily by a set o job-parameters;
iv. the model can be easily modified to accomodate different scheduling algorithms.

Several uses of the model may be cited. It can serve as a tool for the analysis o system performance due to upgrading o chang-ing scheduling policies. It may als be used to predict the system's future performance with different workloads. Sectio VII illustrated some of these applications by a set of experiments. While the numerical results of the simulation model may no be completely accurate; it nevertheless indicates the trend of improvements or degradations, thereby providing guidelines to the analysis of complex computer systems.

I would like to express my appreciation fo two graduate students (Liang Li and Jef Perdue) who provided a significant contribution to the development of this paper.

References

(1) Hughes, H.D., "GPSS Simulation Model of MVS", Interim Technical Report, Dow Chemical, September 1979.

(2) Schwetman, H.D. Jr., A Study of Resource Utilization and Performance Evaluation of Large-Scale Computer Systems, Ph.D. thesis, the University of Texas at Austin, Computation Center, August 1970.

(3) Bunt, R.B. and Hume, J.N.P., "A Simulation Study of a Demand-Driven Scheduling Algorithm", Symposium on Simulation of Computer Systems III, 1975.

(4) Theorey, Toby J., "Validation Criteria for Computer System Simulations", Symposium on Simulation of Computer Systems III, 1975.

(5) Hughes, H.D., "Some Predicted Results for the APL System", Abstracted in the Proceedings of the ACM Computer Science Conference, 1978.

(6) Coffman, E.G. Jr., and Wood, R.C., "Interarrival Statistics for TimeSharing Systems", Communications of the ACM, Vol. 9, No. 3, July 1966.

(7) McDougal, M.H., "Computer System Simulation", Computing Surveys, Vol. 2, No. 3, 1970.

(8) Denning, P.J. and Buzen, J.P., "The Operational Analysis of Queuing Network Models", Computer Surveys, Vol. 10, No. 3, September 1978.

(9) CHO, A.C., Strauss, J.C., "A Simulation Study of Dynamic Dispatching", Symposium on the Simulation of Computer System III, 1975.

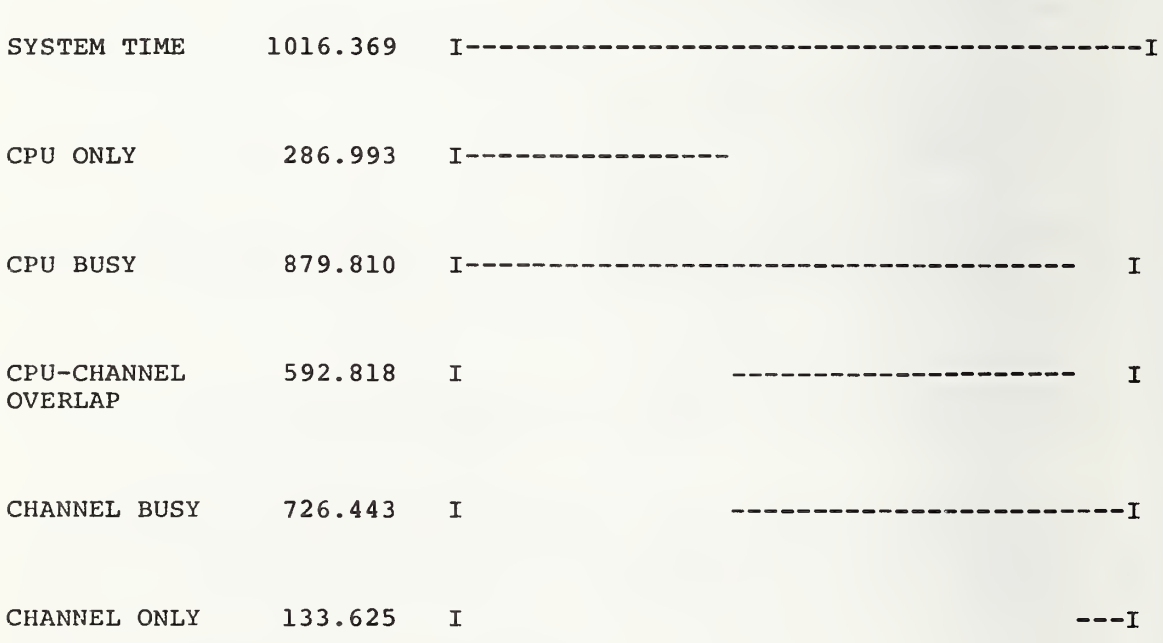
APPENDIX I

SYSTEM PROFILE FOR THE SIMULATION RUN:

CPU AND CHANNEL STATUS: 0=IDLE; 1=BUSY.

CPU	TIME FOR EACH COMBINATION
0	136.579
1	879.810

CHANNEL 1	2	3	TIME FOR EACH COMBINATION
0	0	0	289.946
0	0	1	70.455
0	1	0	206.797
0	1	1	40.659
1	0	0	154.167
1	0	1	31.632
1	1	0	193.887
1	1	1	28.627



ITEM	Simulation results			Theoretical
	200 jobs	300 jobs	400 jobs	results
Mean arrival rate λ (≃ throughput)	2.52	2.68	2.48	3.00
Avg. CPU utilization	0.293	0.295	0.296	0.300
Avg. disk utilization	0.147	0.148	0.149	0.150
Turnaround time	0.242	0.232	0.255	0.245
Avg. CM queue length	0.166	0.172	0.184	0.285
Avg. CM utilization	0.090	0.083	0.085	0.075

TABLE 1. Simulation and Analytic Results for Experiment #1

ITEM	Simulation results for 300 jobs		Theoretical results
	Seek & latency constant	seek & latency exponentially distributed	
Mean arrival rate λ (≃ throughput)	0.296	0.296	0.333
Avg. CPU utilization	0.656	0.656	0.667
Avg. disk utilization	0.146	0.156	0.150
Avg. channel utilization	0.135	0.143	0.133
Turnaround time	4.684	4.732	4.689
Avg. CM queue length	0.574	0.578	0.658

TABLE 2. Simulation and Analytic Results for Experiment #2

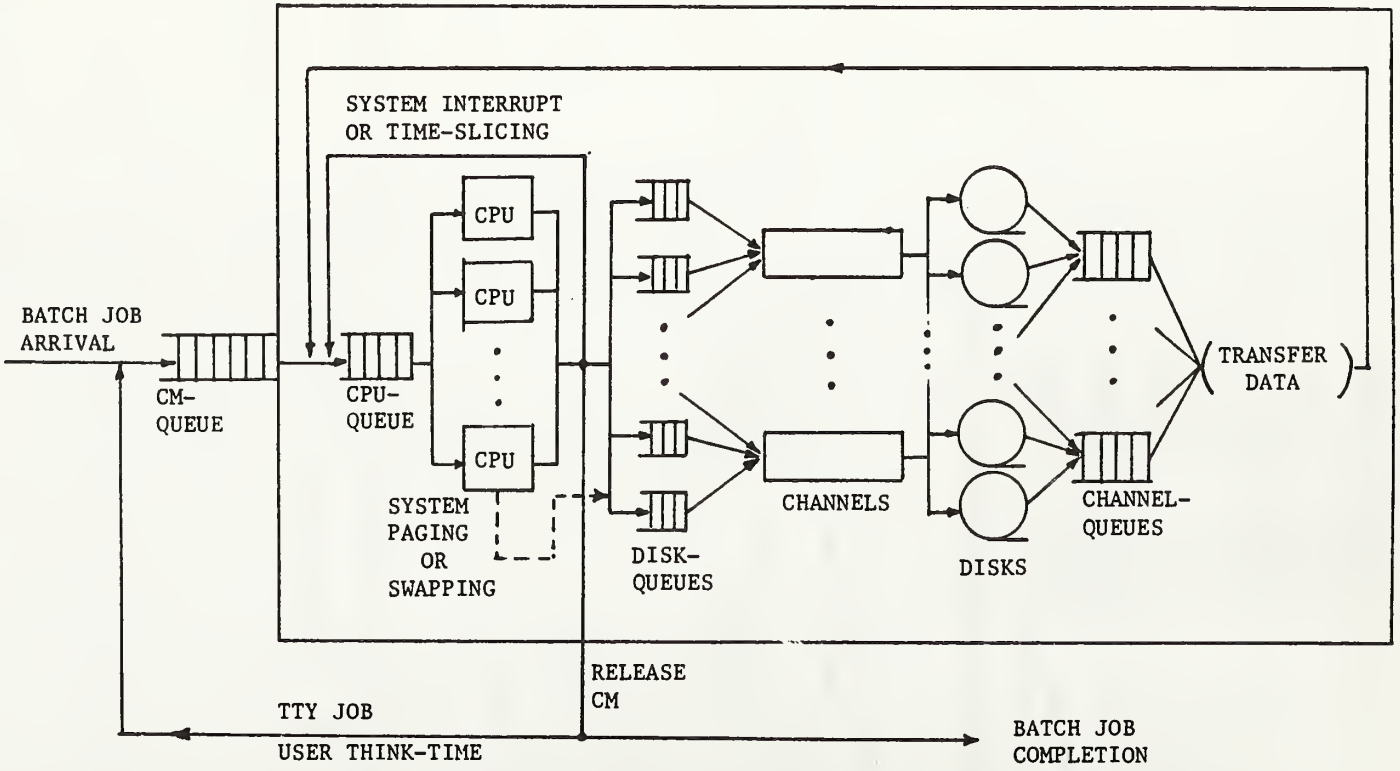
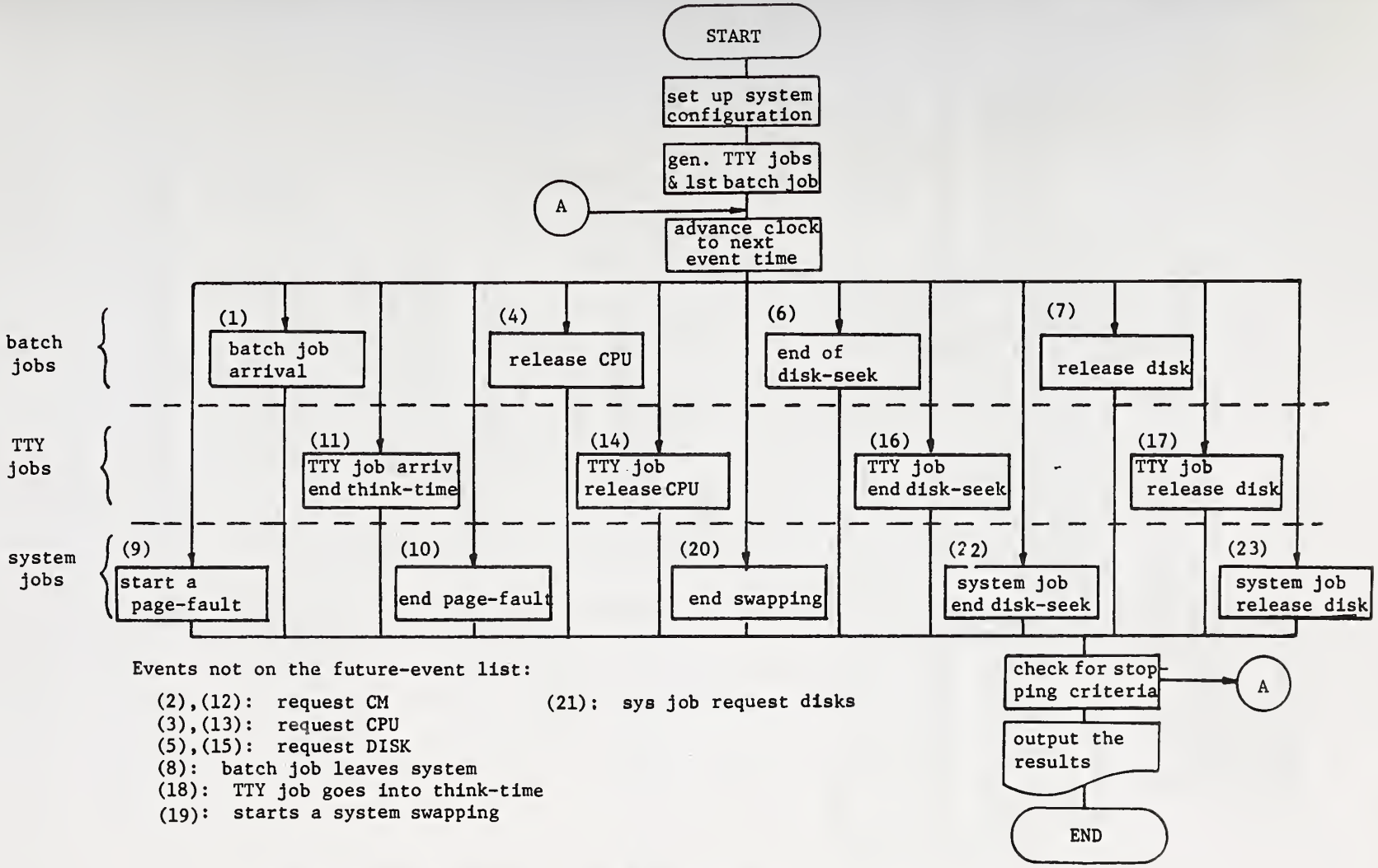


FIGURE 1. The Job-Flow of the Model



Events not on the future-event list:

(2),(12): request CM
(3),(13): request CPU
(5),(15): request DISK
(8): batch job leaves system
(18): TTY job goes into think-time
(19): starts a system swapping
(21): sys job request disks

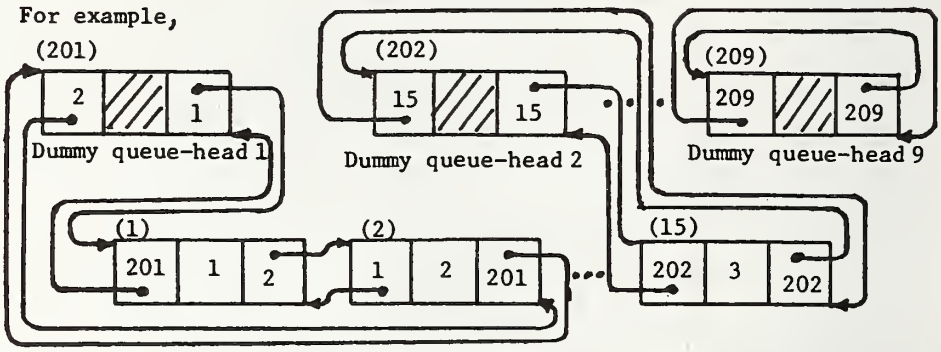
FIGURE 2. Main Flowchart of the Model by Events

A record node has the following format:

(pointer)

predecessor link	user-job number	successor link

where (pointer) is the record number



To put the above queues in the form of an array, we have:

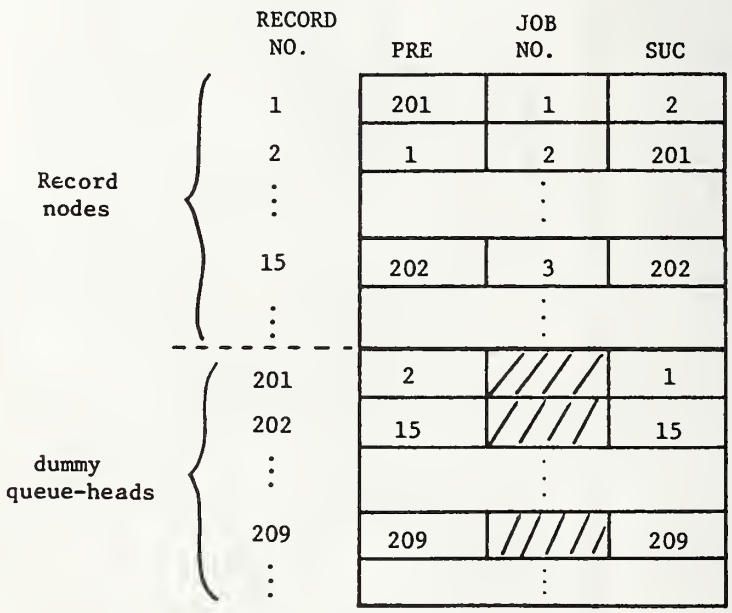


FIGURE 3. Doubly-linked queue structure in the model.

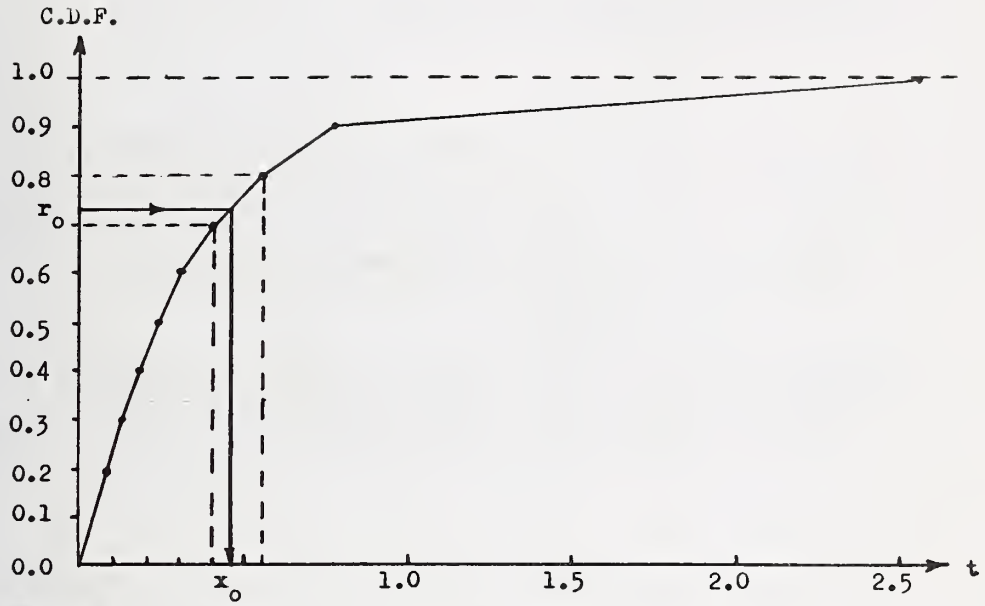


FIGURE 4(a) Generation of a random variate x_o from a discrete C.D.F.. (r_o is a random number uniformly generated between 0 and 1.)

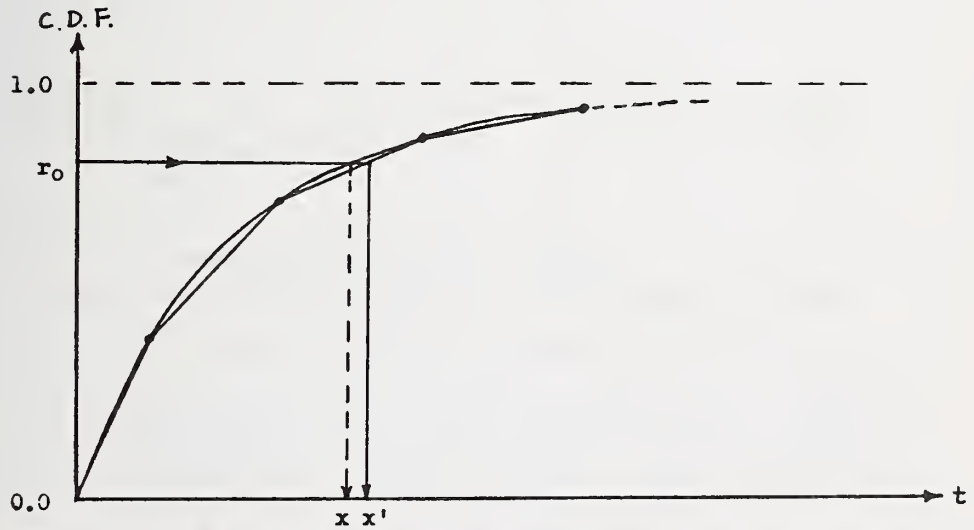


FIGURE 4(b) Example of the error of approximation by the discrete C.D.F.. (x' is the estimation of x.)

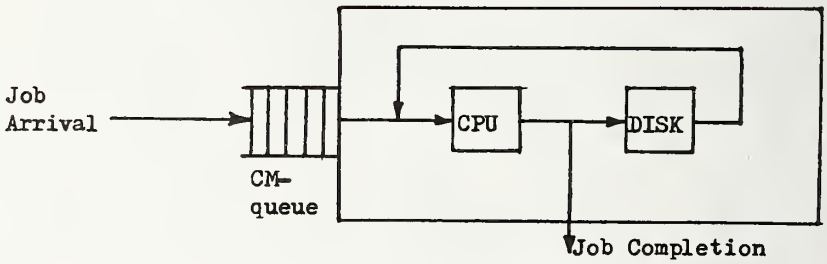


FIGURE 5 A simple 1-CPU, 1-disk uniprogramming system.

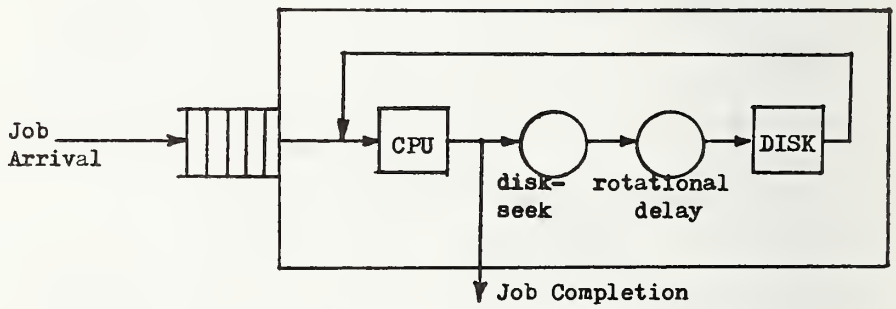


FIGURE 6 A simple system with disk-seek and latency time.

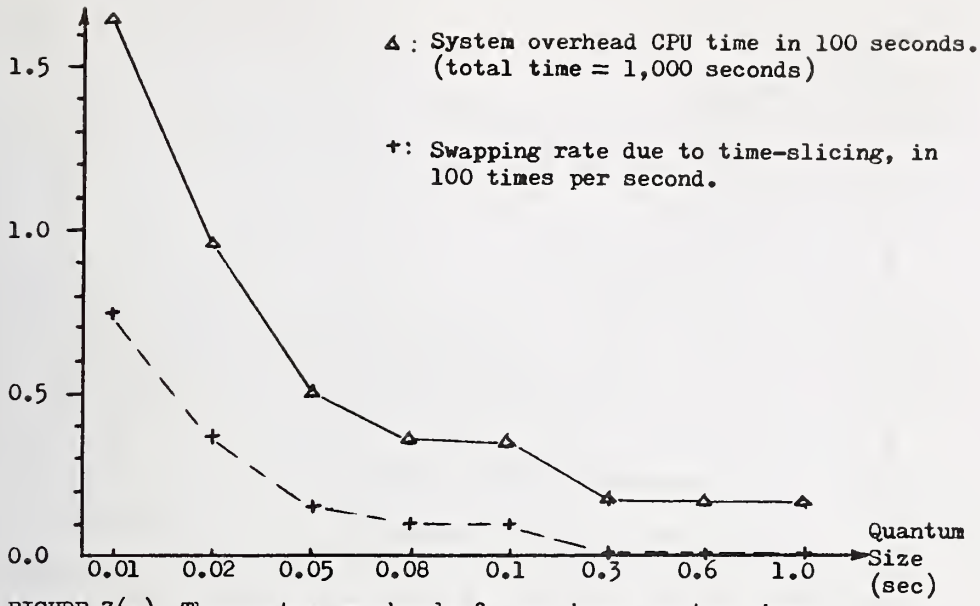


FIGURE 7(a) The system overheads for various quantum sizes.

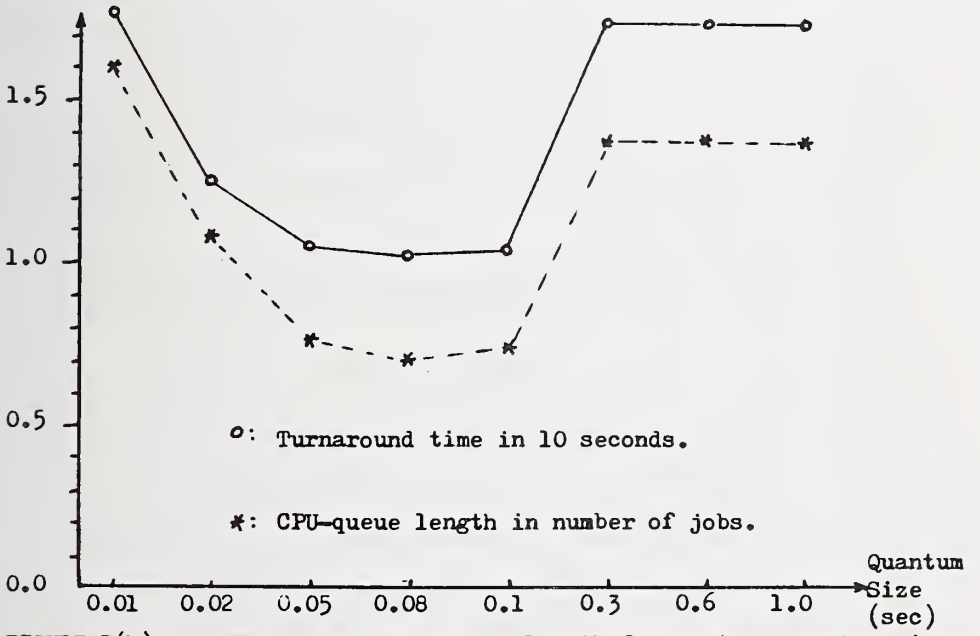


FIGURE 7(b) Turnaround and CPU-queue length for various quantum sizes.

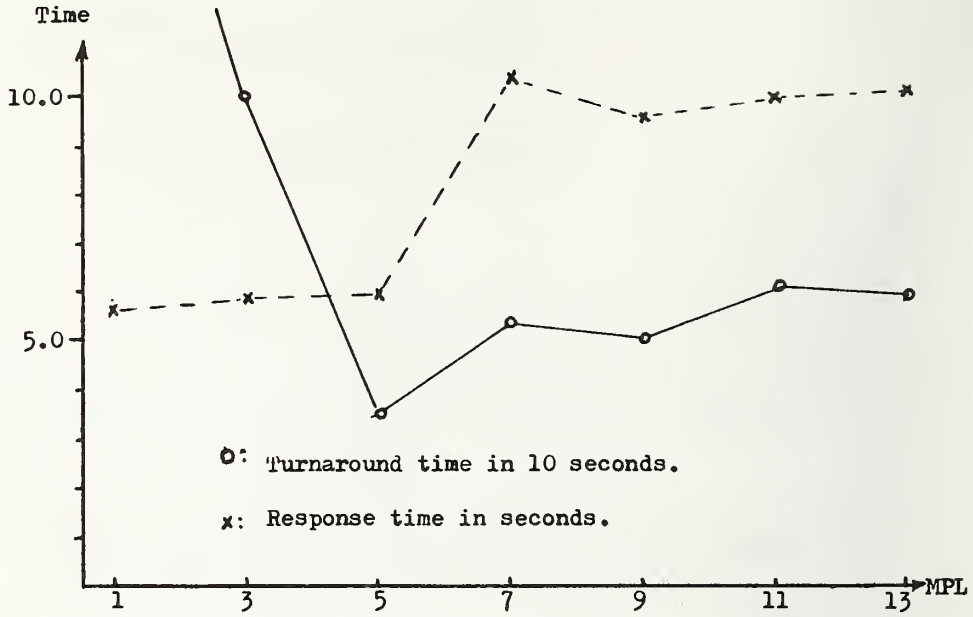


FIGURE 8(a) Turnaround and response time under different Multi-Programming Levels.

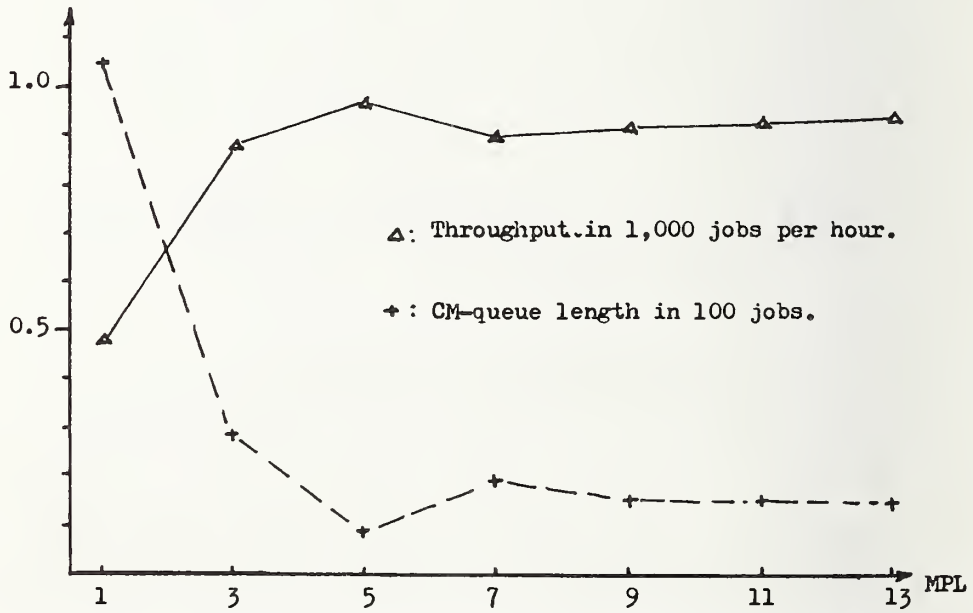


FIGURE 8(b) Throughput and CM-queue length under different Multi-Programming Levels.

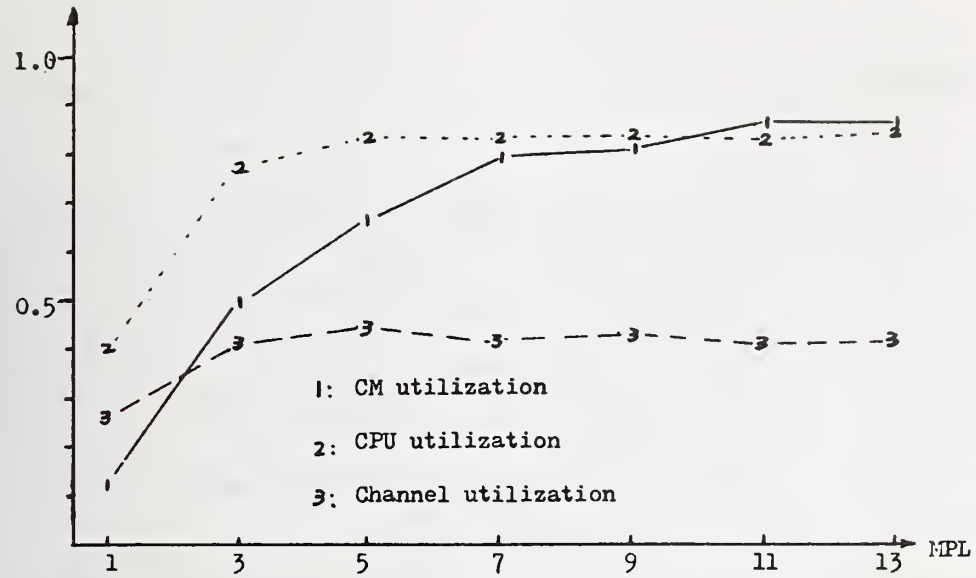


FIGURE 8(c) System utilizations under different Multi-Programming Levels.

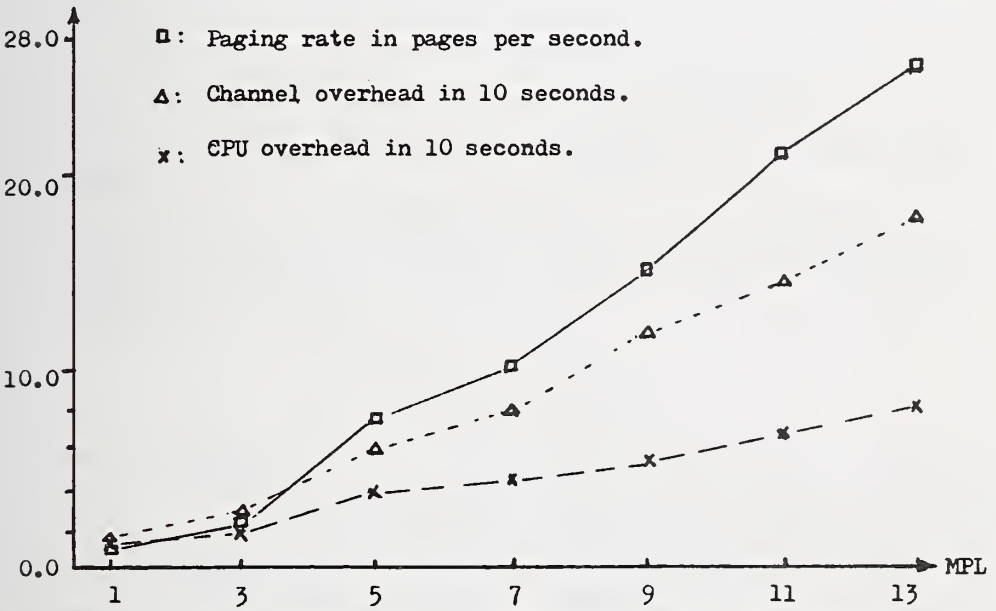


FIGURE 8(d) System overheads under different Multi-Programming Levels.

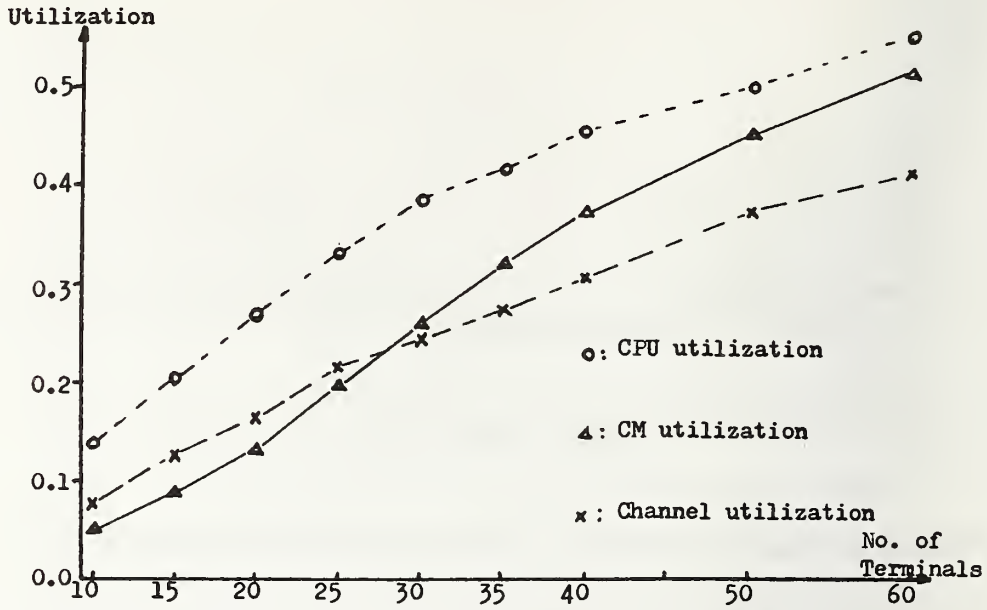


FIGURE 9(a) System utilizations vs. number of terminals.

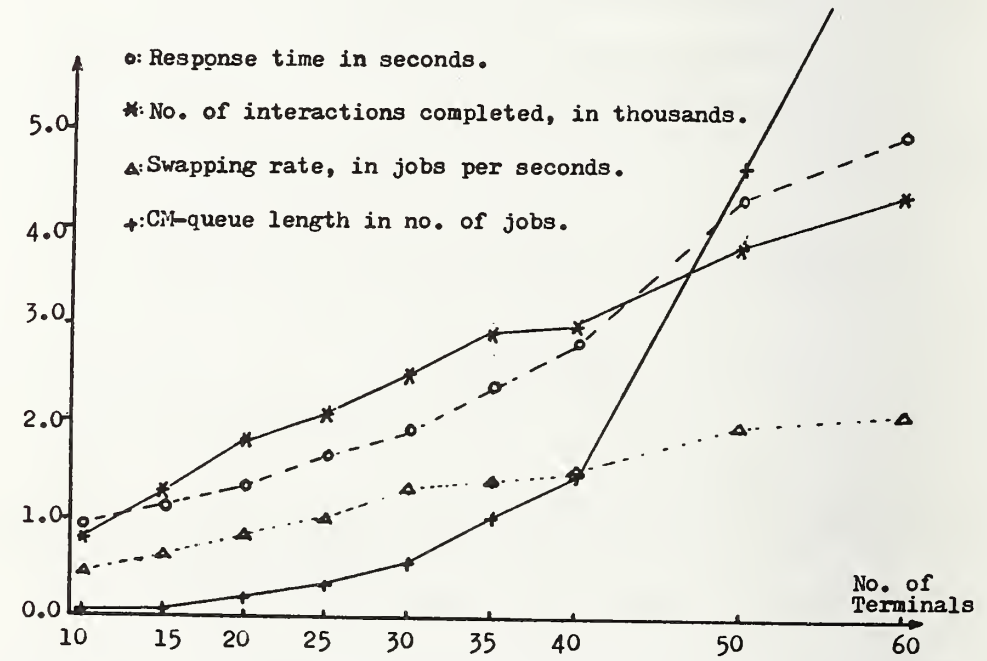


FIGURE 9(b) System overheads and response time vs. number of terminals.

Optimal Selection of CPU Speed, Device Capacities, and Allocation of Files with Variable Record Size

Kishor S. Trivedi and Robert A. Wagner

Department of Computer Science
Duke University
Durham, N.C. 27706

This paper extends a previous model for computer system configuration planning developed by the authors. The problem is to optimally select CPU speed, device capacities, and file assignments so as to maximize system throughput subject to a fixed cost constraint. In our earlier paper we assumed that the record sizes for all files are equal, the block sizes of all devices are equal and these two in turn are equal. In this paper we extend our earlier results to allow each file a distinct record size and each device a distinct block size.

Key words: Capacity planning; configuration planning; file assignment problem; optimization; performance evaluation; performance- oriented design; queuing networks.

1. Introduction

This paper is an extension of a model for computer system configuration planning developed in [1]. We are interested in optimally selecting the CPU speed, the capacities of the secondary storage devices, and the allocation of a given set of files across the secondary storage devices. The objective of optimization is to maximize system throughput subject to budgetary limitations. In [1], it was shown that any relative maximum of this complex nonlinear programming problem is also its global maximum. A technique for significantly reducing the computational complexity of the optimization problem was developed. Finally, an interesting subproblem, known as the file assignment problem, was also discussed.

* This work was supported in part under the National Science Foundation grant number MCS 78-22327 and the National Library of Medicine Program Project grant number LM-03373.

discussed.

Two simplifying assumptions were made in our earlier model. The first assumption was that all secondary storage devices use the same fixed block size. In practice, block size is likely to vary with the type of device. The second assumption was that a logical file I/O request always generates a physical I/O request. This assumption may nothold in practice. Suppose that the logical record size of a file is smaller than the physical block size of a device so that several records are packed into a single physical block. Then a logical I/O request will give rise to a physical I/O request only if the block in which this record resides is not already resident in main memory. Section 2 develops an extension of our earlier model assuming that files are accessed in a sequential fashion. Each device may have a different block size and each file may have a different record size. Any relative maximum of the modified optimization problem is also its global maximum. This optimization model has one variable per file-device pair (giving the amount of file i to be loaded onto device j). The resulting problem has too many decision variables to be solved quickly. However, we have developed a simple file loading rule which reduces the number of decision variables from one per file-device pair to one per device (specifying the size of that device). The file loading rule is then used to determine exactly what parts of each file are to be used to fill each device. This loading rule is deterministic - no elaborate computation is needed to use it. We have shown in [1] that this file loading rule is always optimal. This rule is presented in more detail in Section 2. Section 3 presents an example of use of our model.

We are interested in determining several hardware configuration parameters (CPU speed and device capacities) as well as software configuration parameters (file assignment). The throughput of the system is modeled using a closed queuing network of the central server type [2]. This is in contrast to several efforts ignoring queuing delays [3,4]. Other authors have used open queuing network models which are more suitable for computer-communication network design [2,5,6,7]. For hardware configuration planning, several authors have used decision models of closed queuing networks [8,9,10,11,12,13]. Similarly, several models of file assignment are available [14,15,16]. Apart from our earlier paper [1], the work closest to the present paper is that by Arora and Gallo [17]. Our model may be thought of as a refinement of their model that does not require the approximate decomposition into two submodels that they proposed. In fact, we derive an exact decomposition of the design problem.

2. The Basic Model

Inputs to our model consist of workload parameters and device cost/performance characteristics. The workload is specified by the resource requirements of a typical program. Let W_0 be the total number of instructions required, on the average, to complete the execution of a program. There are f files numbered 1, 2, ..., f. Associated with file i is its logical record size r_i (in bytes), its total size S_i (in bytes), and an activity n_i where n_i denotes the total number of logical I/O requests directed to file i, on the average, during the execution of a typical job.

There are m secondary storage devices and the cost of device j ($1 \leq j \leq m$) is assumed to be proportional to its capacity K_j (in bytes), that is, $COSTDEVICE_j = C_j K_j$. The total cost of the I/O devices is given by $F(\bar{K}) = \sum_{j=1}^{m} C_j K_j$. The block size of device j is d_j bytes. The CPU cost F_0 is modeled by a power function of its speed b_0 (in MIPS), so that

$$F_0(b_0) = C_0 \, b_0^{\alpha_0}.$$

We assume that for each j ($1 \leq j \leq m$) and each i ($1 \leq i \leq f$), d_j is exactly divisible by r_i and that files are accessed sequentially. Since d_j / r_i logical records of file i are stored (blocked) on a single physical block of device j, the probability that a request for a logical record of file i on device j generates an I/O request to device j is r_i / d_j (≤ 1). Let X_{ij} denote the number of bytes of file i assigned to device j. Thus a fraction X_{ij} / S_i of all logical requests to file i will be directed to the portion of file i residing on device j. Only r_i / d_j fraction of these requests generate a physical I/O request to device j. Therefore, the total number of physical I/O requests, W_j, directed to device j is given by

$$W_j = \sum_{i=1}^{f} n_i \frac{X_{ij}}{S_i} \frac{r_i}{d_j}$$

Let t_j denote the average service time (to fetch one block of size d_j) from device j. Then $y_j = W_j \, t_j$ is the total time demanded by a typical program on device j. Let t_0 be the average time to execute an instruction so that $t_0 = 1 / b_0$. Then

$$y_0 = W_0 \, t_0$$

is the total CPU time required by a typical program. The average service times for the I/O devices are assumed to be fixed parameters in this model while the CPU instruction delay t_0 is a control variable.

In order to make the queuing model computationally tractable, we assume that if the scheduling discipline of a device is FCFS (First Come First Served), then the service time distribution is exponential. Any differentiable service time distribution is permitted provided the scheduling discipline is either PS (Processor Sharing) or LCFS-PR (Last Come First Served Preemptive Resume) [18].

The number of jobs circulating in the central server model will be denoted by n and is also referred to as the degree of multiprogramming. We consider n to be a fixed parameter but a discrete search for the optimal value of n can be performed if desired.

The throughput of the system is given by U_0 / y_0 [19; pp. 252-253], [20] where

$$U_0 = y_0 \, G(\bar{y};n-1) / G(\bar{y};n)$$

and

$$G(\bar{y};n) = \Sigma\{ \prod_{j=0}^{m} y_j^{k_j} \mid \sum_{j=0}^{m} k_j = n \}$$

Then the throughput $T(\bar{y};n)$ is given by

$$T(\bar{y};n) = G(\bar{y};n-1) / G(\bar{y};n).$$

Instead of maximizing throughput, we will set up our problem to minimize the reciprocal of the throughput, denoted by $z(\bar{y};n)$. Now the problem of the optimal selection of the CPU speed, device capacities, and file assignments (CSDCFA) can be stated as:

CSDCFA Problem:

$$\min \quad z(\bar{y};n) \tag{1a}$$

Subject to (i=1,2,...,f; j=1,2,...,m)

$$\sum_{i=1}^{f} X_{ij} \leq K_j \tag{1b}$$

$$\sum_{j=1}^{m} X_{ij} = S_i \tag{1c}$$

$$F_0(W_0 t_0) + F(\bar{K}) \leq \text{COST} \tag{1d}$$

$$y_j = \frac{t_j}{d_j} \sum_{i=1}^{f} n_i X_{ij} r_i / S_i \tag{1e}$$

$$t_0 \geq 0 \tag{1f}$$

$$y_j \geq 0, \; K_j \geq 0 \tag{1g}$$

$$X_{ij} \geq 0 \tag{1h}$$

The constraint (1b) assures us that the capacity of each device is not exceeded by the file assignment, and the constraint (1c) guarantees that every byte of every file is placed on some device.

The CPU delay t_0, device relative utilizations $y_j (1 \leq j \leq m)$, the device capacities $K_j (1 \leq j \leq m)$, and the file assignment variables $X_{ij} (1 \leq i \leq f, 1 \leq j \leq m)$ are the decision variables for a total of m*(f+2) + 1 variables. The values of t_j, $d_j (1 \leq j \leq m)$, S_i, n_i, r_i $(1 \leq i \leq f)$, W_0, n, m, f, and COST are assumed to be fixed parameters for this problem.

With an appropriate change of variables, the optimization problem (1) above reduces to problem (9) in [1]. Therefore, the following results are easily derived:

THEOREM 1

The CSDCFA problem (1) is a convex programming problem (assuming that $\alpha_0 \geq 0$), hence any relative minimum is also its global minimum.

It should be noted that X_{ij}'s, K_j's, and S_i's are now measured in bytes rather than blocks as in [1]. Similarly, C_j is the cost per byte of device j.

A special case of problem (1) is the file assignment (FA) problem where the device capacities $K_1, K_2, \ldots, K_m$ and the CPU delay t_0 are fixed parameters. As a corollary to theorem 1, we obtain the convexity of the FA problem.

COROLLARY 1: The file assignment problem is a convex programming problem.

Any solution to the CSDCFA problem can be shown to have an ordered characteristic as in [1]. Specifically, files may be ordered by non-increasing n_i / S_i value (access probability). These ordered files can then be assigned to devices, filling each device in non-increasing order of per-byte cost (C_j). We have shown that every optimum solution necessarily has this form. Since this form involves as decision variables only the device capacities, we are able to reduce the number of decision variables from m*(f + 2) + 1 to 2m + 1 .

3. An Example

We now consider a numerical example illustrating the use of our model. Assume that there are 10 files with the parameters as specified in Table 1. The total number of instructions to be executed per program is one million, the budget is 3 million dollars, and the main memory cost per degree of multiprogramming is assumed to be $50,000. The input data for the I/O devices and the CPU is given in Table 2. The block sizes of the two devices are d_1 = 3,840 bytes and d_2 = 1,920 bytes.

The design model was exercised for each degree of multiprogramming starting from 1 up to 8. The resulting optimal values of throughput, device capacities, and CPU speed in MIPS are given in Table 3. Scanning the table we determine the optimal degree of multiprogramming to be 7 . The optimal file assignment, though an output of our model, is not reproduced here.

The total CPU time needed by our 370/165 to compute the solution values above was 1.68 seconds. Thus, many design options, such as variations in the number of, and speeds of, I/O device can be attempted; the detailed choice of device capacities, CPU speeds, degree of multiprogramming and file placement can be done automatically, at low cost, using this optimization technique.

Note that device capacities and file allocation variables are assumed to be continuous variables in this paper while in practice they are discrete. The effect of this assumption has been analyzed in [21] and will be further explored in a future paper.

Briefly, these papers show that discretizing the continuous solution according to our methods may cause a throughput degradation of a factor $\leq 1 + (m-1)\max_i \{n_i/S_i\}$, and a cost overrun of at most $d_{max}*(f*C_{max} + (f+m)*(C_{max}-C_{min}))$. This latter extra cost is the cost of storing one additional maximum-size block per file, plus the cost of moving one block per file and per device from the least expensive to the most expensive device. We believe that, for practical problems, the cost overrun is trivial, and the throughput degradation is less than 1%.

The model reported in this paper was implemented as a computer program by Dr. A.K. von Mayrhauser.

References

[1] K. S. Trivedi, R. A. Wagner, and T. M. Sigmon, "Optimal Selection of CPU Speed, Device Capacities, and File Assignments," JACM , July 1980.

[2] L. Kleinrock, Queuing Systems Vol. II: Computer Applications, Wiley-Interscience, New York, 1976.

[3] C. V. Ramamoorthy and K. M. Chandy, "Optimization of Memory Hierarchies in Multiprogrammed Systems," JACM, Vol. 17, No. 3 (July 1970), pp. 426-445.

[4] C. K. Chow, "On Optimization of Staging Hierarchies," IBM Journal of Research and Development, Vol. 18, No. 3 (May 1974), pp. 194-203.

[5] K. M. Chandy, J. Hogarth, and C. H. Sauer, "Selecting Capacities in Computer Communication Systems," IEEE Trans. on Soft. Eng., Vol. SE-3, No. 4 (July 1977), pp. 290-295.

[6] S. T. Chanson and P. S. Sinha, "Optimization of Memory Hierarchies in Multiprogrammed Computer Systems with Fixed Cost Constraint," Technical Report, Department of Computer Science, University of British Columbia, 1979.

[7] S. Mahmoud and J. S. Riordan,

"Optimal Allocation of Resources in Distributed Information Networks," ACM Transactions on Database Systems, Vol. 1, No. 1 (March 1976), pp. 66-78.

[8] W-W. Y. Chiu, "Analysis and Applications of Probabilistic Models of Multiprogrammed Computer Systems," Ph.D. Dissertation, Department of Electrical Engineering, University of California, Santa Barbara, California, 1973.

[9] D. Ferrari, Computer Systems Performance Evaluation, Prentice-Hall, Englewood Cliffs, New Jersey, 1978.

[10] S. K. Kachhal and S. R. Arora, "Seeking Configurational Optimization for Computer System Configuration Planning," Proceedings ACM Annual Conference, 1975, pp. 96-101.

[11] K. S. Trivedi and R. E. Kinicki, "A Mathematical Model for Computer System Configuration Planning," in Performance of Computer Installations, D. Ferrari (ed.), North-Holland, Amsterdam, 1978.

[12] K. S. Trivedi and T. M. Sigmon, "A Performance Comparison of Optimally Designed Computer Systems with and without Virtual Memory," Proceedings, 6th Annual International Conference on Computer Architecture, 1979.

[13] K. S. Trivedi and R. A. Wagner, "A Decision Model for Closed Queuing Networks," IEEE Trans. on Soft. Eng., Vol. SE-5, No. 4 (July 1979), pp. 328-332.

[14] D. V. Foster and J. C. Browne, "File Assignment in Memory Hierarchies," in Modeling and Performance Evaluation of Computer Systems, Beilner and Gelenbe (eds.), North-Holland, Amsterdam, 1976.

[15] D. V. Foster, L. W. Dowdy, and J. E. Ames, "File Assignment in a Star Network," Technical Report 77-3, Systems and Information Science Department, Vanderbilt University, Nashville, Tennessee, 1977.

[16] T. G. Price, "Probability Models of Multiprogrammed Computer Systems," Ph.D. dissertation, Department of Electrical Engineering, Stanford University, Palo Alto, California, 1974.

[17] S. R. Arora and A. Gallo, "Optimization of Static Loading of Multilevel Memory Systems," JACM, Vol. 20, No. 2 (April 1973), pp. 307-319.

[18] K. M. Chandy, J. H. Howard, and D. F. Towsley, "Product Form and Local Balance in Queuing Networks," JACM, Vol. 24, No. 2 (April 1977), pp. 250-263.

[19] P. J. Denning and J. P. Buzen, "The Operational Analysis of Queueing Network Models", ACM Computing Surveys, Vol.10, September 1978, pp. 225-261.

[20] T. Giammo, "Extensions to Exponential Queueing Network Theory for use in a Planning Environment," Proceedings of the IEEE COMPCON, Fall 1976.

[21] R. A Wagner and K. S. Trivedi, "Hardware Configuration Selection Through Discretizing a Continuos Variable Solution," Proceedings of 1980 Int. Symp. on Comp. Per. Mod. Meas. and Evaluation, Toronto, Canada.

Table 1: File Activity Profile

file	record size r_i (bytes)	No. of records S_i/r_i	Av. Activity Count n_i
1	480	84,000	13.86
2	320	125,000	14.85
3	480	84,000	9.9
4	960	84,000	7.92
5	960	521,000	28.71
6	320	188,000	4.95
7	320	500,000	3.96
8	320	625,000	3.96
9	320	625,000	3.96
10	960	1,125,000	6.93

Table 2: Device Input Data

Device #	Label	Cost Curve C_j coeff.	Cost Curve α_j exponent	Speed t_j	Controller Cost
0	CPU	$1,147,834	0.55309	Variable	
1	I/O	0.0030575	1.0000	0.00378	$24,496
2	I/O	0.000155	1.0000	0.02938	$19,910

Table 3: Output of the Design Model

degmul n	TPUT jobs/sec	Capacity K_1 (million bytes)	Capacity K_2 (million bytes)	CPU speed b_0
1	0.986	201.28	2203.2	2.599
2	1.293	201.28	2203.2	2.479
3	1.423	212.12	2189.48	2.29
4	1.487	214.68	2186.92	2.161
5	1.518	209.88	2191.72	2.07
6	1.532	205.8	2195.8	1.999
7	1.535	201.28	2203.2	1.92
8	1.528	201.28	2203.2	1.816

CPEUG80

Performance Prediction Techniques - 2

MVS Performance Prediction Using Mechanically - Generated Queuing Models

R. J. Feil, New York Telephone, New York, NY
B. A. Ketchledge, AT&T, Piscataway, NJ

A central issue in capacity planning for IBM MVS environments is the prediction of system performance under varying workload and configuration assumptions. This paper addresses the issue of performance prediction in the context of a general approach to capacity planning for MVS environments. In particular the paper covers:

a. Definition of an RMF-based measurement strategy for MVS environments.

b. Software for mechanical generation of BEST/1 queueing models of MVS systems. (BEST/1 is a proprietory product of BGS Systems, Inc.)

c. Calibration and use of the resulting BEST/1 models for capacity planning studies. An actual case study will be presented.

Key Words: Capacity, Planning; queueing models.

1. Introduction

This paper discusses an approach to capacity planning for MVS environments which has grown out of a Bell System Capacity Planning project. Before exploring this approach, it is worthwhile to overview the project itself.

The Capacity Planning project has as its goal the definition and implementation of a Capacity Planning Methodology. The current phase of the project involves the development of a Capacity Planning and Management System (CPMS) for use in IBM - compatible mainframe environments employing MVS as the System Control Program. This effort focuses on developing techniques to:

1. Establish Current Status: Characterize current status of the mainframe environment in terms of workload, hardware utilization, and service.

2. Verify previous plan: Isolate and resolve significant deviations from a previously developed capacity plan.

3. Generate Future Workloads: Forecast and characterize new workloads, as well as growth trends in existing workloads.

4. Forecast Processing Performance: Predict future processing capabilities taking into account workload and hardware/software changes.

5. Generate Recommendations: Generate and report capacity planning recommendations to management.

During 1979, a project team developed and implemented major portions of an IBM - compatible methodology. Prior to the formal start of the trial, two commercial tools were selected as potential aids for capacity planning. SAS, a proprietary product of the SAS Institute, Incorporated, is a data manipulation, reporting and statistical analysis tool. BEST/1, a proprietary package of BGS, Incorporated, is a performance prediction tool. Evaluation of these tools in the context of the CPMS was a major goal of the effort.

The specific objectives of the project were as follows:

1. Develop a workable approach to capacity planning (CP) in an IBM MVS environment using available tools and packages.

2. Evaluate the usability, flexibility, and validity of the data manipulation tool, SAS, and the performance prediction tool, BEST/1.

3. Test the methodology in a series of actual CP studies.

The main effort to date has concentrated on systems with these characteristics:

1. MVS or MVS/System Extensions (MVS/SE) as the System Control Program.

2. IBM or an IBM - compatible mainframe as the CPU.

3. Workloads including batch, TSO, IMS/DL-I, and IMS/DC.

The approach used in the development and trial of the methodology using the CPMS software has been threefold:

1. Definition of a workable approach to each of the five functional CP areas listed above.

2. Integration of the commerical tools (SAS and BEST/1) into these approaches.

3. Execution of a series of CP studies. These studies allowed the CPMS software to be further defined and the operational characteristics of the tools to be evaluated.

In the case of BEST/1, a series of preliminary validation studies were carried out. These studies had the objective of verifying the analytic algorithms used in BEST/1 and its ability to model various data processing situations.

The intent of this paper is to discuss two major aspects of the MVS Capacity Planning Methodology developed thus far:

1. Software and techniques to characterize the current status of the computer system.

2. A methodology and supporting software for prediction of processing performance.

In particular, Section 2 of this paper discusses a SAS - based, 'Performance Management Subsystem' (PMS) which employs data from the IBM Resource Management Facility (RMF) to report on status of the computer system. Section 3 of this paper extends this discussion into the question of forecasting processing performance. Software to reduce raw measurement data (SMF and RMF) and mechanically generate BEST/1 queueing models is described. Section 4 discusses a case study in which the model generation software was employed.

2. Establishing Current Status

Techniques to characterize the current status of the computer system stand at the foundation of the CP methodology. Beyond the general goal of providing an answer to the question 'Where are we today?', this area of the methodology addresses several specific issues:

1. What data sources and metrics should be employed to measure workload, utilization, and service?

2. How are functional workloads to be defined and measured?

3. How should the raw measurement data be reduced, stored, and displayed so as to allow its use in bottleneck identification and resolution?

4. How can the reduced measurement data be massaged to provide input to the portion of the methodology dealing with performance prediction?

The approach to characterizing current status is based on use of a widely-available measurement tool (SMF/RMF) and an available data manipulation tool, (SAS). (Characterization of IMS - related workloads involves additional tools). Highlights of the methodology are:

1. Definition of a standard MVS Installation Performance Specification (IPS) across all systems in the data center. MVS Performance Groups (PG's) are assigned functionally.

2. Workloads corresponding to various functionally - related PG's are grouped together into 'Logical Workloads.' This classification is maintained consistently across all systems to be studied.

3. The concept of a PL/I-generated 'Workload Resource Table' which ties together workload demand (by 'Logical Workload') with service and hardware utilization measures provides a vehicle for interpretation of SMF/RMF data into modeling terms ('Predict Processing Performance'). This concept is discussed in Section 3.

4. Existing workloads are characterized (on an ongoing basis) using RMF data only. RMF Service Units play a principal role in this characterization. These units are also used to characterize consumption of hardware resources. SMF workload data is used only in modeling ('Predict Processing Performance').

The methodology is implemented through use of SAS software written to process raw SMF-RMF data. The PMS software consists of two functional modules:

1. An SMF-RMF data extraction program which allows the user to process SMF-RMF Record Types 0,4,8,34,40,70-75 and generate one or more SAS datasets for each record type.

2. A set of SAS - based report generation programs which selectively process the extracted SMF-RMF data and generate both technical and management-oriented reports.

The example reports on the following pages illustrate some of the salient aspects of this approach to characterizing current status. The reports were generated using SMF-RMF data collected from an IBM 370/158-AP processor supporting both TSO and batch users.

The first report displays the weekly average CPU utilization of the AP processors by two hour intervals (referred to as 'Durations'). Based on data extracted from the RMF Type 70 record, this report allows the user to easily identify intervals of peak CPU usage.

The next report is a SAS 'star chart' (generated using the SAS procedure PROC CHART) which shows the distribution of batch job terminations ('TRNSTERM') by two-hour intervals. The source for this data is the RMF Type 72 record. Only those Type 72 record fields corresponding to batch Performance Groups were used in constructing this chart. This selection is accomplished via an internally coded table of RMF Performance Group definitions.

The following report shows the distribution of TCB CPU time (RMF Type 72 record CPU Service Units) across the various functional workloads. It is done in SAS 'pie chart' format.

The next report is a SAS 'block chart' of CPU-I/O overlap across two-hour intervals. On the left the various system states are displayed. 'WAIT' refers to CPU Idle; 'OVERLAP' refers CPU and Any Physical Channel Busy; 'CPU BOUND' implies CPU Busy and All Channels Idle; 'I/O BOUND' implies CPU Idle and Any Channel Busy .

The final report displays physical channel usage by two-hour interval. It is used to identify potential I/O load imbalances.

These and other reports from the PMS are reviewed regularly by both tuning and capacity planning personnel. As such they provide support for day-to-day tuning efforts as well as characterization of current system status.

3. Predicting Processing Performance

As discussed earlier, the effort to develop software and a methodology for determining current status of the computer system has been matched with a complimentary effort to define methods to predict processing performance. In this section an approach to predicting performance given a specific workload and hardware configuration is presented. Techniques to develop workload forcasts for input to the performance prediction process are not discussed.

The approach developed for predicting performance is based on the use of the BEST/1 performance prediction tool supplemented with software to mechanically generate the models. Major aspects of the methodology are as follows:

1. Performance prediction is performed using the BEST/1 package.

2. Use of BEST/1 is supported by an extensive set of modeling strategies. MVS batch, TSO, and IMS environments are supported. (IMS/DC models require additional measurement and data reduction tools).

3. These modeling strategies are implemented in PL/I-based software which further manipulates the SAS-reduced SMF/RMF data and mechanically generates BEST/1 models. These models employ the 'Logical Workload' concept discussed earlier.

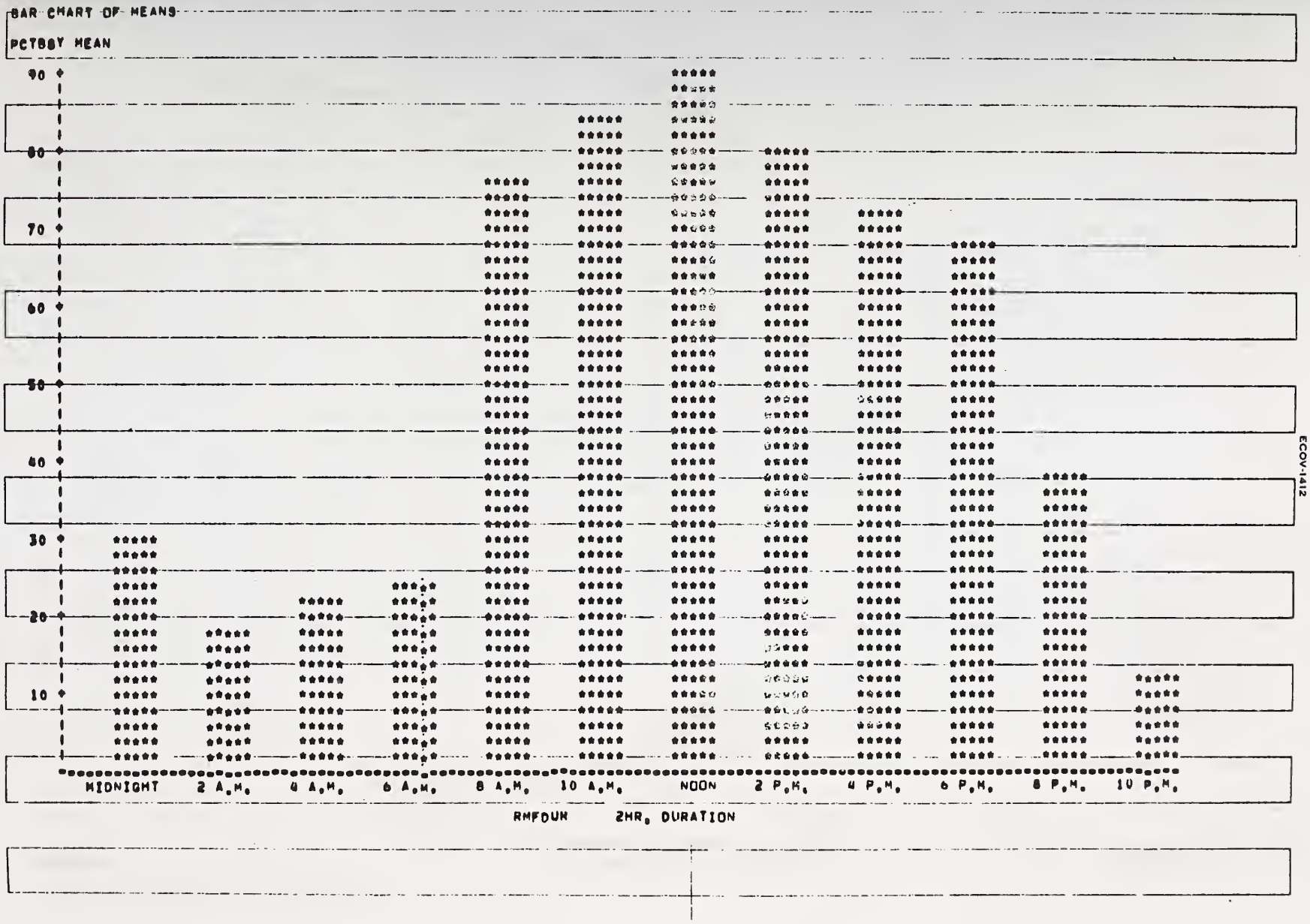
4. Calibration and validation procedures for refinement of the mechanically-generated models are defined.

Selection of BEST/1 for use as the performance prediction tool was motivated by several factors. Among these were the following:

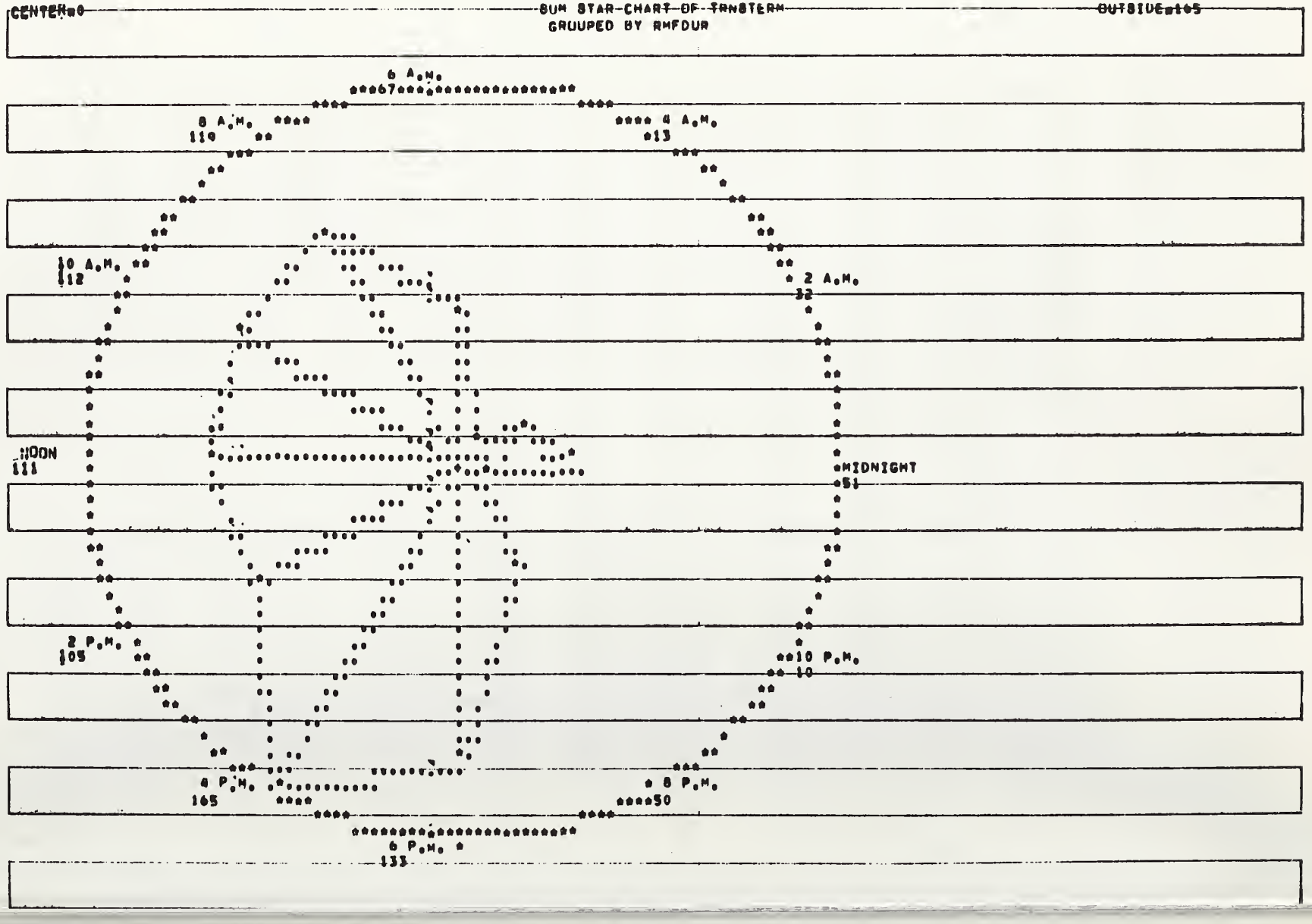
1. The demonstrated capability of queueing-based analytic models to give reasonably accurate predictions of system performance as evidenced by the performance literature over the past 5-10 years.

1. The demonstrated capability of queueing-based analytic models to give reasonably accurate predictions of system performance as evidenced by the performance literature over the past 5-10 years.

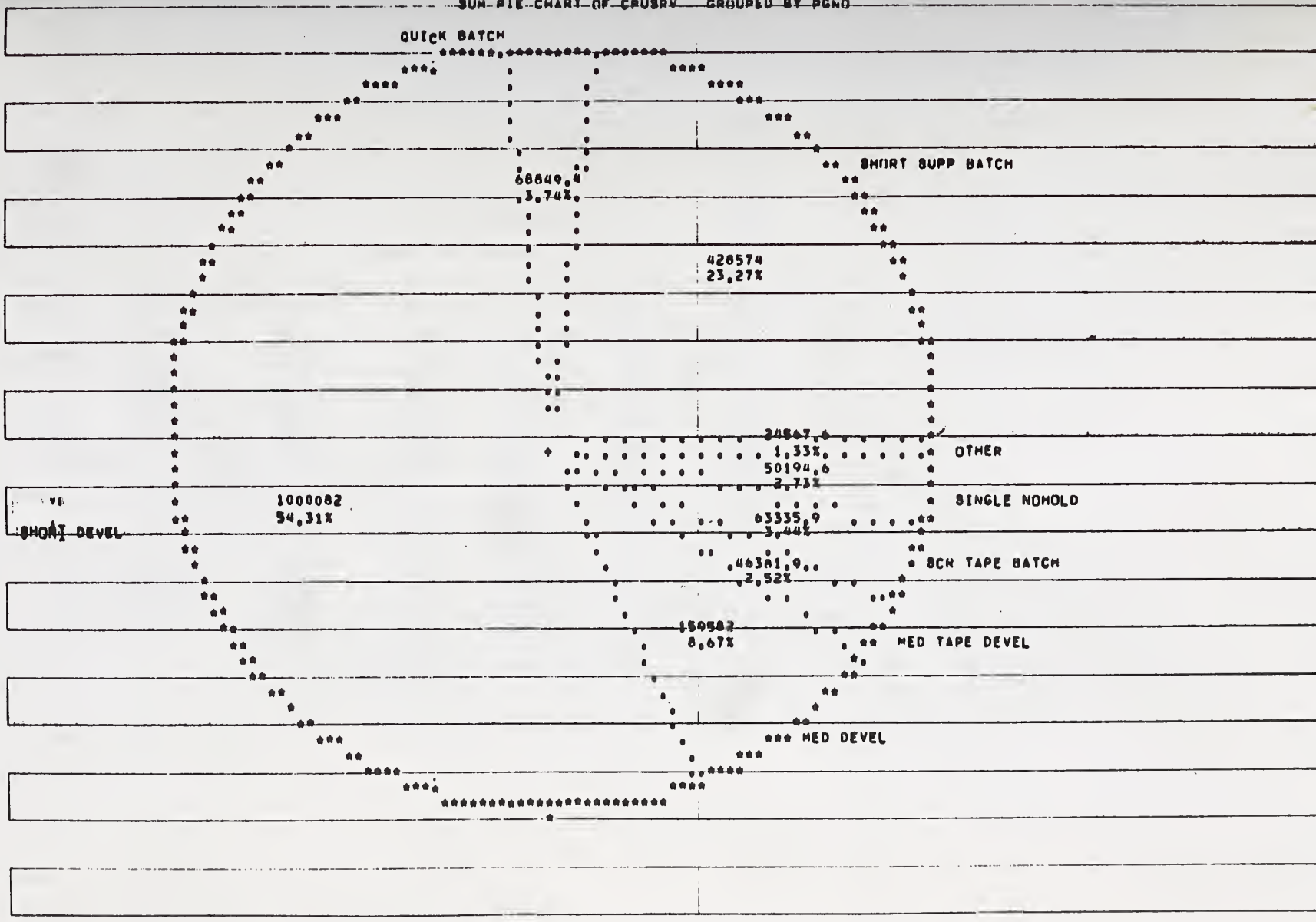
AP/CPU MEAN PERCENT BUSY ACROSS DURATIONS



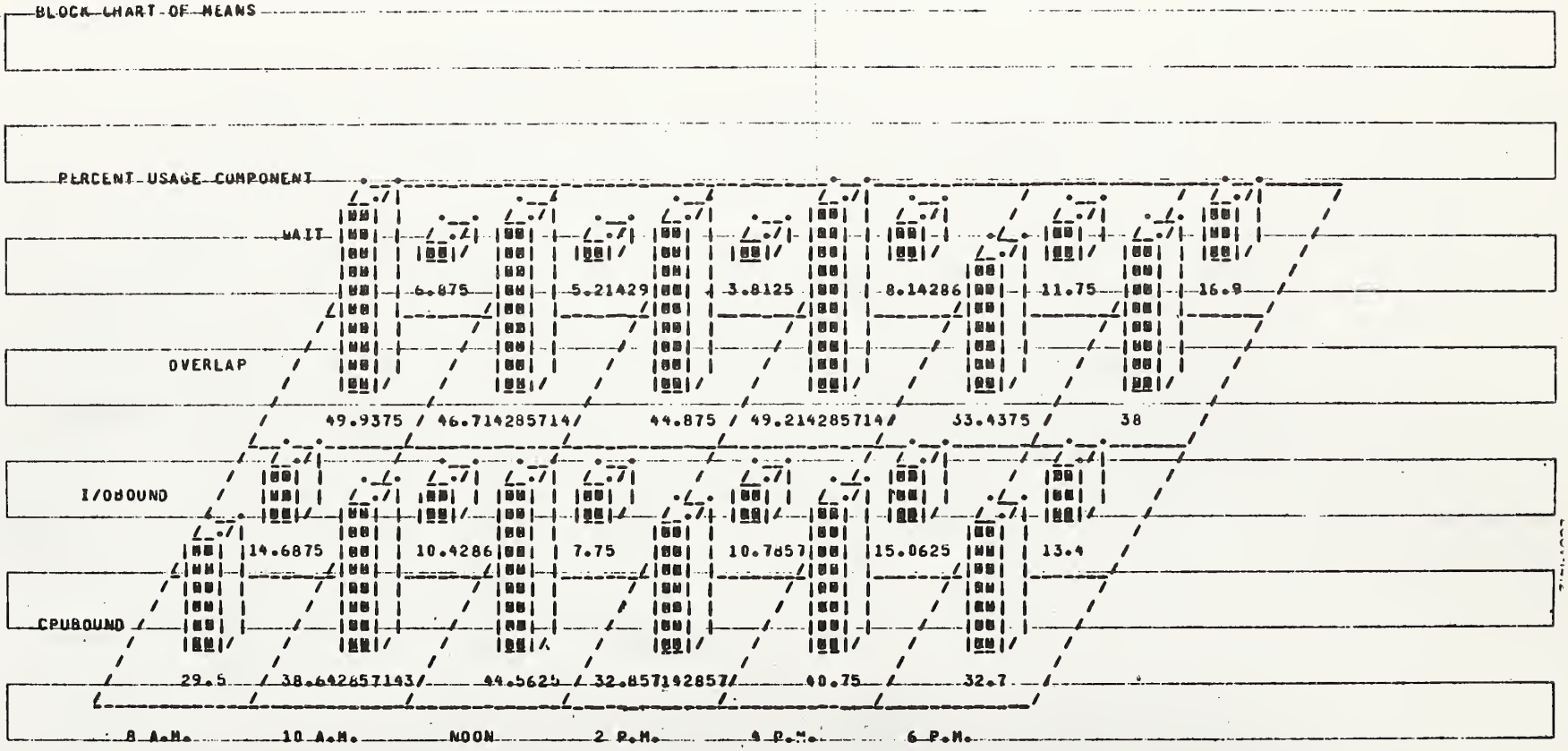
BATCH USE OF SERVICES ACROSS FULL DAY



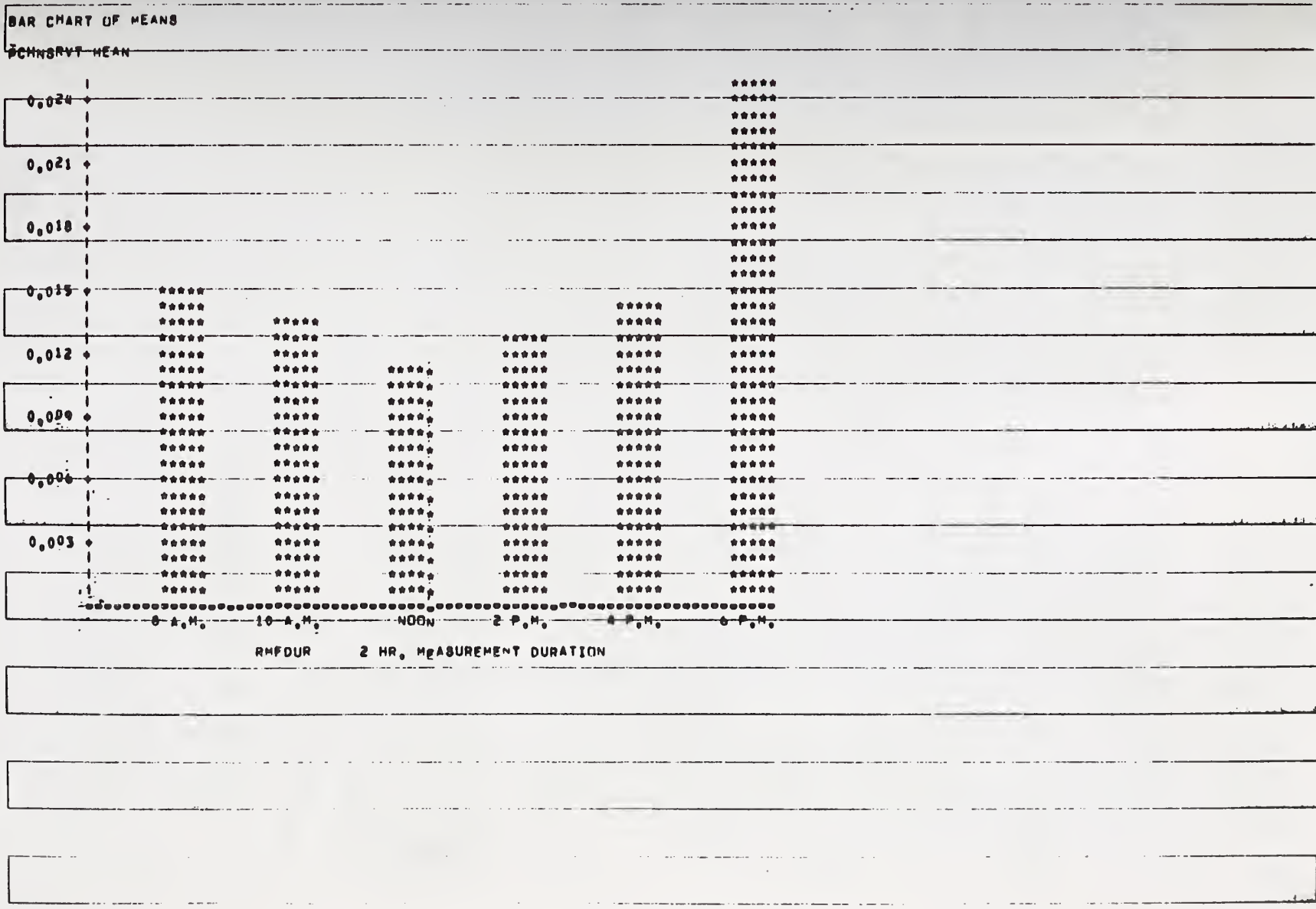
BATCH USERS OF SERVICES ACROSS FULL DAY



AVERAGE HARDWARE COMPONENT USAGE DISPLAYS ACROSS DURATIONS



AVERAGE PHYSICAL CHANNEL SERVICE TIME ACROSS DURATIONS
CHANNEL ID=2 CPU ID OF CHANNEL=1



2. The desire to provide a performance prediction tool with interactive capabilities and a good user interface.

3. The vendor-independent nature of the BEST/1 package, allowing for potential extension of its use into non-IBM DP environments.

Development of software to mechanically generate BEST/1 models was likewise motivated by several factors:

1. The need to make implementation of the CP methodology as cost - effective as possible. This implied that mechanization of appropriate portions of the methodology, such as model generation, should be carried out.

2. Successful use of BEST/1 requires development of suitable modeling strategies in which algorithms to manipulate raw data for creation of the BEST/1 model are defined. The algorithms tend to be specific to both the software environment modeled (e.g., TSO, IMS, CICS) and the type of measurement data used. Mechanization of these modeling strategies and their associated algorithms enables a broader range of CP analysts to employ BEST/1 in a consistent manner.

The remainder of this section discusses first, the Model Generation Component (MGC) of CPMS. After discussing this software the process by which it is used to generate calibrated BEST/1 models is overviewed. This will provide background for the Case Study addressed in Section 4.

The MGC of CPMS presently consists of three functional modules:

1. An SMF-RMF data extraction program written in SAS. This an enhanced version of the analogous PMS module.

2. A SAS-based interface program ('SASBEST') which massages the extracted SMF-RMF data and generates conventional OS files.

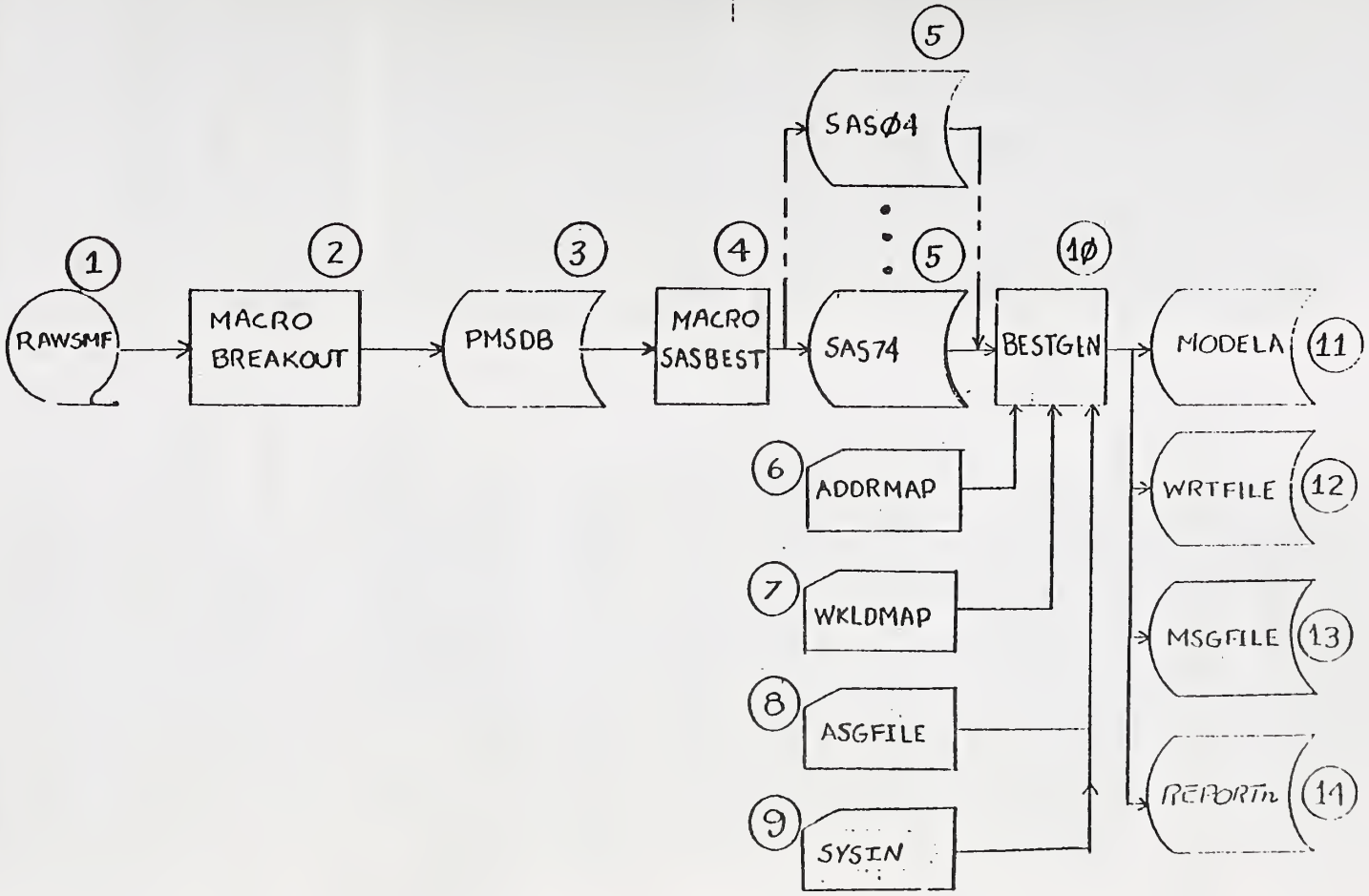
3. A PL/I module ('BESTGEN') which accepts the OS files written by SASBEST and generates a BEST/1 model, among other outputs.

The flow chart on the following page illustrates the structure of the model generation system. Raw SMF-RMF Data (1) from a specific time interval provides the input to the overall system. (The measurement interval is selected based on representativeness criteria outside of the scope of this paper.) The SMF-RMF data is processed by the SAS data extraction program (2) and results in creation of a 'database' (3) in SAS format. This database consists of a number of SAS datasets, each corresponding to a specific SMF or RMF record type and each with a specific name and set of variables. For example, one such dataset is named PMSDB.T70-CPU and contains variables such as CPUBSY0 and CPUBSY1. These variables indicate the CPU utilization on processor 0 and 1 of the mainframe, respectively, at a given point in time. This data is extracted from the RMF Type 70 record.

The SAS database of extracted SMF-RMF data provides input to another SAS routine, SASBEST (4), which creates OS files accessible to PL/I. In additon, these routines perform sorts on the various SAS datasets so as to accomadate the PL/I model generation program. The SASBEST output files (5) are input to the model generation program, BESTGEN (10).

CPMS OVERVIEW

Model Generation Component



ISSUE Release 1.1

DRAWN 2/11/80 ENGR

TITLE CPMS OVERVIEW MODEL GENERATION COMPONENT

NO. OF SHEETS THIS SET

SHEET

149

2. The desire to provide a performance prediction tool with interactive capabilities and a good user interface.

3. The vendor-independent nature of the BEST/1 package, allowing for potential extension of its use into non-IBM DP environments.

Development of software to mechanically generate BEST/1 models was likewise motivated by several factors:

1. The need to make implementation of the CP methodology as cost - effective as possible. This implied that mechanization of appropriate portions of the methodology, such as model generation, should be carried out.

2. Successful use of BEST/1 requires development of suitable modeling strategies in which algorithms to manipulate raw data for creation of the BEST/1 model are defined. The algorithms tend to be specific to both the software environment modeled (e.g., TSO, IMS, CICS) and the type of measurement data used. Mechanization of these modeling strategies and their associated algorithms enables a broader range of CP analysts to employ BEST/1 in a consistent manner.

The remainder of this section discusses first, the Model Generation Component (MGC) of CPMS. After discussing this software the process by which it is used to generate calibrated BEST/1 models is overviewed. This will provide background for the Case Study addressed in Section 4.

The MGC of CPMS presently consists of three functional modules:

1. An SMF-RMF data extraction program written in SAS. This an enhanced version of the analogous PMS module.

2. A SAS-based interface program ('SASBEST') which massages the extracted SMF-RMF data and generates conventional OS files.

3. A PL/I module ('BESTGEN') which accepts the OS files written by SASBEST and generates a BEST/1 model, among other outputs.

The flow chart on the following page illustrates the structure of the model generation system. Raw SMF-RMF Data (1) from a specific time interval provides the input to the overall system. (The measurement interval is selected based on representativeness criteria outside of the scope of this paper.) The SMF-RMF data is processed by the SAS data extraction program (2) and results in creation of a 'database' (3) in SAS format. This database consists of a number of SAS datasets, each corresponding to a specific SMF or RMF record type and each with a specific name and set of variables. For example, one such dataset is named PMSDB.T70-CPU and contains variables such as CPUBSYØ and CPUBSY1. These variables indicate the CPU utilization on processor Ø and 1 of the mainframe, respectively, at a given point in time. This data is extracted from the RMF Type 7Ø record.

The SAS database of extracted SMF-RMF data provides input to another SAS routine, SASBEST (4), which creates OS files accessible to PL/I. In additon, these routines perform sorts on the various SAS datasets so as to accomadate the PL/I model generation program. The SASBEST output files (5) are input to the model generation program, BESTGEN (10).This contains a subset of the data items originally contained in SMF-RMF Record Types 4,34,40,70,72,73 and 74.

Hence data containing resource demand by the various workloads (Record Types 4,34,40 and 72) as well as utilization and multiprogramming levels on the system (Record Types 70,73 and 74) is available to the BESTGEN program.

BESTGEN also accepts a number of manually created files. These files supply control information to the program:

1. The ADDRMAP ('Address Map') (6) file contains a mapping from physical device addresses into 'Logical Addresses'. (These Logical Addresses correspond to BEST/1 'Servers' in the mechanically generated model).

2. The WKLDMAP, ('Workload Map') (7) maps actual workloads, as identified by RMF Performance Group Number/Period on the SMF-RMF Type 4 and 72 records, into 'Logical Workloads'. Each Logical Workload corresponds to a BEST/1 Workload in the mechanically-generated model.

3. The SYSIN (8) file contains user - assigned values to various control parameters. These parameters control which phases of the BESTGEN program are executed, and let the user overide or modify various internal BESTGEN algorithms.

4. The ASGFILE ('Assignment File') (9) provides a vehicle for the user to give BESTGEN special environmental information. Among other things, the user can assign Capture Ratios to various workloads via this file. Likewise, the user can force BESTGEN to allocate the activity on various devices to one or more workloads via control cards.

(It should be noted that the BESTGEN program includes routines to mechanically generate WKLDMAPs and ADDRMAPs suitable for most modeling applications.)

Execution of the BESTGEN program results in the creation of a number of output files and reports:

1. The BEST/1 model created by BESTGEN is written to the MODELA (11) file. It is a complete ready-to-run BEST/1 model.

2. The WRTFILE (1) file receives one 'Workload Resource Table' for each time period analyzed. For example, if the user specified in the SYSIN file that raw measurement data for the interval 5 A.M. to 11 A.M. was to be processed on a one-hour basis, one Workload Resource Table would be generated for each of these intervals: 5-6 A.M., 6-7 A.M., ... 10-11 A.M.

   The Workload Resource Table contains data derived from all of the various SMF-RMF Record Types processed. It contains all the information necessary to build a BEST/1 model for the interval to which it pertains.

3. The MSGFILE (13) is used to record informatory and warning messages generated by BESTGEN during its execution.

4. The various reports created by the BESTGEN program are written to the REPORT files (14).

In its present form the BESTGEN programs generates the following reports:

1. Workload Resource Table. This report summarizes the essential workload and hardware parameters of the system being modeled. Referring to the example on the next page, note that the columns of the report refer to the various Logical Workloads. The upper part of the report gives various information regarding each workload, such as:

   a. WKLD TYPE - The 'type' of workload, using BEST/1 terminology. 'BP' workloads are batch; 'TS' are time-sharing; 'TP' workloads are transaction processing.

   b. WKLDNAME - A descriptive name given the workload by the user in the WKLDMAP input file.

   c. CPU UTILIZATION - The estimated utilization of the CPU by this workload, as computed by BESTGEN using the SMF-RMF data.

WORKLOAD RESOURCE TABLE

DATE	TIME INTERVAL	CPU UTILIZATION
80086	14.50 - 15.50	93.04

WORKLOAD/SERVICE MATRIX
UNITS= SEC

WKLD NUMBER		1	2	3	4	5	6	7
WKLD TYPE		JS	TS	TS	TS	TS	BP	BP
WKLD NAME		TSOUS	TSOUS	TSOUS	TSOUS	TSOEL	QUICK	BATS
CPU UTILIZATION		0.00	0.83	38.33	12.09	0.00	10.30	8.6
ELAP/RESP TIME		0.00	3.66	4.91	5.95	0.00	67.39	642.1
AVE TERMS/MPL		0.00	0.41	20.48	6.13	0.00	0.62	0.8
THRUPUT		0.00	71.56	2278.52	561.68	0.00	33.13	4.8
CPU SERVICE		0.00	30.03	1379.74	435.30	0.00	370.73	312.6
TAPE MOUNTS		0	1	9	1	0	3	3
I/O SERVICE								
POOL	TYPE	UTIL						
140	DASD	2.7	0.00	0.00	0.00	0.00	0.00	0.00	0.00
141	DASD	9.4	0.00	0.00	15.74	277.44	0.00	0.00	46.66
142	DASD	8.2	0.00	0.00	0.00	0.00	0.00	0.00	0.00
143	DASD	0.1	0.00	0.00	0.00	0.00	0.00	0.00	0.00
145	DASD	2.6	0.00	0.00	0.00	0.00	0.00	0.00	0.00
146	DASD	3.9	0.00	0.00	102.11	0.08	0.00	0.62	0.00
250	DASD	43.5	0.00	0.00	0.00	0.00	0.00	0.00	0.00
251	DASD	24.7	0.00	13.85	628.21	242.18	0.00	3.27	0.00
252	DASD	22.3	0.00	26.49	598.85	178.54	0.00	0.00	0.00
253	DASD	1.5	0.00	0.00	19.09	0.00	0.00	32.09	0.00
254	DASD	19.3	0.00	0.00	10.25	25.58	0.00	1.23	65.03
255	DASD	13.8	0.00	0.00	0.00	0.00	0.00	0.00	0.00
256	DASD	1.6	0.00	0.00	2.98	0.00	0.00	22.88	0.00
380	DASD	0.9	0.00	0.00	28.98	2.15	0.00	0.00	0.00
382	DASD	4.9	0.00	0.00	0.00	0.00	0.00	0.00	0.00
384	DASD	7.7	0.00	0.00	0.00	0.00	0.00	0.00	0.00
397	DASD	0.9	0.00	0.00	0.00	0.00	0.00	0.00	0.00
39F	DASD	2.9	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3B0	DASD	38.3	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3B1	DASD	18.0	0.00	0.00	133.46	0.00	0.00	8.77	0.00
3B2	DASD	1.1	0.00	0.00	0.67	0.00	0.00	1.47	0.00
3B3	DASD	0.1	0.00	0.00	4.68	0.00	0.00	0.00	0.00
3B4	DASD	10.0	0.00	0.00	6.45	0.93	0.00	0.93	195.20
3B6	DASD	9.6	0.00	0.00	0.00	0.00	0.00	344.88	0.00
1$$	DASD	15.5	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2$$	DASD	36.9	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3$$	TAPE	37.2	0.00	8.66	923.40	34.28	0.00	2.00	151.84
4$$	TAPE	11.7	0.00	0.00	0.00	0.00	0.00	0.00	123.32
5$$	TAPE	4.3	0.00	0.00	0.00	0.00	0.00	31.72	0.00

d. I/O SERVICE - The total time (in seconds) spent by this workload at the Logical Address ('POOL') listed at the left. Note that the actual utilization of the physical devices corresponding to this Logical Address is also shown ('UTIL').

The major use of the Workload Resource Table is in model calibration, where results from the execution of the generated BEST/1 model are compared to the Workload Resource Table.

2. Multiprogramming Report - This report gives estimates of multiprogramming depth, throughput, and service for each Logical Workload.

3. I/O Service by Pool - This report summarized total EXCPS (I/Os) by Logical Address as well as milliseconds of device busy. A 'milliseconds per EXCP' ratio is also given, which is useful in identifying cases in which EXCPs are a poor measure of relative usage of a device by the various workloads.

4. CPU Time Report - This report displays TCB and SRB CPU time totals by workload type (e.g., 'BP', 'TS', or 'TP') as well as total CPU utilization. It provides input to SAS-based Capture Ratio analysis routines.

A typical modeling study using this software might proceed as follows:

1. Determine the scope and objectives of the study. Most importantly, decide what question(s) will the proposed models answer?

2. Identify and obtain relevant data. SMF-RMF is the minimum data required. Operating system parameters (IPS and VATLST especially) are also used.

3. Select the time interval of interest. This choice is based on the questions the model will attempt to address as well as availibility of valid measurement data. CPMS includes a number of SAS routines which are useful for this analysis.

4. Extract and reduce the SMF-RMF data using the CPMS extraction program and SASBEST.

5. Prepare BESTGEN user inputs. These include the ADDRMAP, WKLDMAP, SYSIN and ASGFILE files. A number of issues must be addressed in this phase, such as:

   a. Strategy for modeling Operating System functions.

   b. Workload aggregation (WKLDMAP).

   c. Peripheral modeling approach (ADDRMAP).

6. Execute BESTGEN and create the initial BEST/1 model.

7. Calibrate the BEST/1 model by executing it and matching the results of the BESTGEN output reports (especially the Workload Resource Table). This is in general an iterative process.

8. Address the 'what-if' questions laid down during the scope/objectives phase.

To date the CPMS Model Generation Component has been used in several studies. The next section discusses one such application of the system.

4. Case Study

In this section calibration and use of typical BEST/1 model is described. The system modeled was an IBM 370/158-AP configuration supporting approximately 20 to 25 TSO users and 4 to 6 active batch initiators during prime-shift hours. Typical CPU utilizations varied between 60-75% during these periods.

Construction of the model was facilitated by use of the mechanical model generation software incorporated into the CPMS. One important goal of the study was to test that software in the context of modeling a live operational environment. Another goal was the development and validation of modeling strategies for MVS/TSO.

The modeling and validation approach employed was divided into these phases:

a. Mechanical generation of a base BEST/1 model of a prime-shift hour with heavy load.

b. Calibration of the model to reflect operational consideration, e.g., tape mounts, system pack usage, and varying TSO MPL levels.

c. Modification of the calibrated model of heavy load to reflect lower multiprogramming (MPL) levels occurring at periods of reduced loads.

For the purpose of creating a base BEST/1 model of the period of heaviest load, a particular hour was selected. Four workloads identified as active during this interval were modeled. They are listed in Table 1 at the end of the paper.

The approach used in modeling the hardware configuration was highlighted by:

a. Representation of each active DASD device and tape channel by an individual BEST/1 Server.

b. Approximation of the 370/158-AP CPU by a 370/158 uniprocessor with 1.8 times the internal speed as a normal UP.

c. Use of SAS-based linear regressions to apportion device busy times across workloads where necessary. This technique was applied to system-oriented packs such as SWAP, CVOL, SPOOL, and SYSRES volumes.

Calibration of the base model involved providing estimates of tape mount time (for batch workloads) and assuring a correct distribution of CPU time across workloads. The distribution of CPU time was accomplished via SAS analysis of SMF capture ratios (ratio of SMF TCB CPU time to actual CPU time). Additionally, estimates of user think time were calculated for each of the two TSO workloads.

Results

Calibration of the base model resulted in good overall agreement between predicted and actual values; as shown in Table 2 at the end of the paper.

Validation of the model was accomplished by varying the model parameters to reflect a different interval on the same system. Table 3 compares this interval to the original interval represented in the base model.

Adjustments made to the base model of heavy load to reflect the interval of lighter load were as follows:

a. The number of TSO terminals in the two TSO workloads was reduced from 13 and 6 to 8 and 4 respectively.

b. The batch MPLs were all scaled by the ration of 1.80/3.50, i.e., the ration of Batch MPLs for the two intervals. Note that the mix of batch workloads (i.e., the relative MPLs of the various batch workloads) was not adjusted. Since the primary validation criteria was TSO response time and total batch throughput, exact representation of the MPLs of the various batch workloads occurring in the interval of lighter load was not considered necessary.

Results of the model are shown in Table 4.

Based on the relatively good results obtained, it seems that the base model is valid and quite robust.

Several conclusions were based on this study:

a. BEST/1 is applicable to modeling of live operational environments provided that data concerning operational factors such as tape mount time is available.

b. SAS can be used as an aid in statistical analysis of measurement data during model construction and calibration. SAS-based linear regressions were used in both determining appropriate capture ratios and partitioning system pack usage across workloads.

c. Use of the BEST/1 priority feature (PRIORITY =) is essential for valid modeling of mixed batch-time sharing MVS environments.

d. The mechanical model generation software incorporated into the CPMS is adequate for generation of BEST/1 models.

TABLE 1

Workload Name	Functional Description
TSOUSER-App.A	TSO users involved in application 'A' development
TSOUSER-App.B	TSO users involved in application 'B' development
BATTEST	Batch users doing application testing.
BATTEST-DEV	Batch users involved in application development.

TABLE 2

Quantity	Actual	Predicted	Relative Error
CPU Utilization (%)	69.5	68.5	-1.4%
TSO Response Time - App. A. (sec.)	2.40	2.65	+10.4%
TSO Response Time - App. B. (sec.)	2.93	2.83	-3.5%
Batch Throughput *(Jobsteps/hr.)	72.2	63.0	-12.7%
TSO Command Throughput **(Cmds/hr.)	2308.0	2302.0	-0.3%

TABLE 3

Interval Comparison
(Actual Values)

	Interval	
	Light Load	Heavy Load
CPU Utilization (%)	46.80	69.50
TSO Response Time - App. A. (sec.)	2.08	2.40
TSO Response Time - App. B. (sec.)	2.80	2.93
Batch Throughput (Jobsteps/hr.)	82.30	72.20
Logged on TSO Users	12.00	19.00
Batch MPL	1.80	3.50

TABLE 4

Quantity	Actual	Predicted	Relative Error
CPU Utilization (%)	69.50	68.50	-1.4%
TSO Response Time - App. A. (sec.)	2.40	2.65	+10.4%
TSO Response Time - App. B. (sec.)	2.93	2.83	-3.5%
Batch Throughput *(Jobsteps/hr.)	72.20	63.00	-12.7%
TSO Command Throughput **(Cmds/hr.)	2308.00	2302.00	-0.3%

An I/O System Model for 303X Processors

Sushil Bhatia
Phillip Carroll

International Business Machines Corporation
P.O.Box 950
Poughkeepsie, New York 12602

In the 303x input/output system certain jobs are critical in the sense that the system pays a penalty in reduced performance if they are not finished in some maximum allowable time. The purpose of the model is to determine the extent of the penalty a system will pay in a heavily loaded environment.

The model of the input/output system of the 303x processors decomposes the system into a three level hierarchy of server/requestors. The levels are: 1. (highest) the teleprocessing controllers, 2. the byte channel interface, and 3. (lowest) the director of the 303x channel group. Each server services requests from the next higher level of the hierarchy (in a strict priority order defined by the system design) and makes requests to the next lower level of the hierarchy.

In a typically heavily loaded situation, the rare situations when the queues for the shared resources become excessive, some jobs may pay a performance penalty. The analytic techniques used to determine the probability of these rare situations of excessive queue length and the model validation results will be described.

Introduction

Evaluation of channel performance for channels designed with shared resources (e.g., 303X directors) is more complex than those with nonshared resources (for example stand-alone channels of the 370 Model 168). Queueing theory is applicable to the evaluation of systems with shared resources and has been used quite successfully in the evaluation of computer system performance. However, it has not been used to evaluate the detailed hardware designs of the systems with shared resources, or to predict the occurrence of rare events. The paper deals with the first successful application of queueing theory to the evaluation of performance of channel hardware and in predicting the occurrence of rare events, which in some cases have channel performance implications.

Model Need

Greater care needs to be exercised in configuring, and in

designing, channels using shared resources than is required in configuring and designing stand-alone channels. The increased sensitivity of the I/O system performance to details of configuration, combined with the greater complexity of the I/O system, generated the need for a model to assist the systems engineer and the channel designers in the analysis of the various interacting resources. Such a model should be easy to use, have a reasonably fast execution time, and should be able to handle the wide variety of I/O devices attachable to the system. It should also have an acceptable level of accuracy. The model was created to meet these objectives.

System Overview

An I/O device experiences performance degradation if its time-dependent function requests are not completed by the I/O system within the allowable time. The phenomenon is labelled an overrun. On some devices, the degradation due to an overrun is negligible and transparent to the user; on some it is marginal and barely noticeable whereas on other devices it results in human intervention and hence is visible. As examples, IBM 3705 Communications Controller on WRITE operation, sends idle characters on the transmission line if the data has not been received from the channel. This results in lower teleprocessing (TP) line utilization but the user view of system performance is not affectd. On some mechanical devices, overruns create a fixed time delay until mechanical synchronization can be reestablished, for example, DASDs and line printers. While on unbuffered devices like IBM 2501, card reader and IBM 2701 data adapter unit, human intervention is required on the occurrence of an overrun. The model estimates the overrun rates for each device in the specified configuration. It does not deal with or analyze the performance effects just described.

The contention for shared resources is the primary cause of overruns on I/O devices. These critical resources utilized by different devices, the contention for which may cause a device overrun, are shown in Table 1.

Table 1 Critical Resources of Devices

Device	Critical Resources
Devices on Block Channel	1. Channel Group Processor 2. Main Memory
Devices on Byte Channel	1. Byte Channel I/O Interface 2. Channel Group Processor 3. Main Memory
Teleprocessing	1. TP Controller's Interface Adaptor 2. Byte Channel I/O Interface 3. Channel Group Processor 4. Main Memory

For all the devices listed in Table I, when a higher-numbered critical resource is in use by a device, all the lower-numbered critical resources are also being used by that device, e.g., when a TP line is using the channel group processor then it is also using the byte channel I/O interface and the TP controller's interface adapter (TPCI adapter). Thus, multiple critical resources may be used simultaneously by a device.

The channel group processor and byte channel interface service the requests by a predetermined nonpreemptive priority scheme. The service mechanism used by main memory and the TPCI adapter is very complex and its description is beyond the scope of this paper.

Model Overview

As mentioned earlier, the current version of the model is limited to the prediction of the overrun rates on each device in the system. An overrun occurs if a certain sequence of work (labelled critical sequence) is not completed by the channel in the time allowed by the device. This allowable time

is labelled critical time. If the work involved in the critical sequence is not completed within the critical time, an overrun occurs. Hence, to determine the occurrence of an overrun, the critical sequence response time (CSRT) is calculated. It is then compared with the critical time. If CSRT exceeds the critical time, an overrun occurs. This is illustrated in Figure 1.

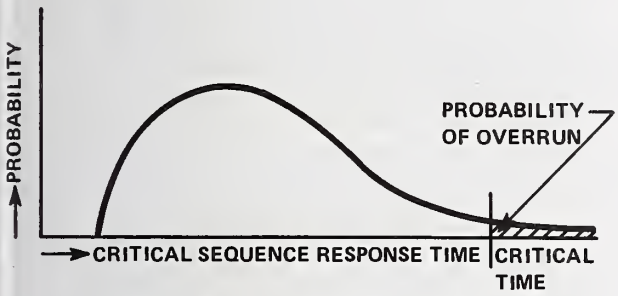


FIGURE 1. CRITICAL SEQUENCE RESPONSE TIME DISTRIBUTION

An analysis of the critical sequence work indicates that for some portions of the work one shared resource is required. For some other portions, more than one shared resource is required, and for some portions of work dedicated resources are used. As an example, consider the diagram shown in Figure 2. During intervals t_2, t_3 and t_5, only shared resource #1 is being used. During the intervals W_3 and t_4, when the request is waiting for and receiving service from the shared resource #2, the #1 shared resource is not released. Hence the effective service time of the shared resource #1 is t_6, which is the total time the next request for the shared resource #1 would see it busy. During t_1, only dedicated resources are being used and since these are not shared with other requests, no time is spent waiting for these resources to become available.

The service times of the requests, when a single resource is being utilized (that is, times t_1, t_2, t_3, t_4, and t_5), can be determined from the analysis of the system design. But the waiting times, (W_1, W_2 and W_3) depend on the load on the shared resources and the service discipline of the resource (e.g., FCFS or priority). Analytic techniques requiring use of queueing theory are used in the model to determine these waiting times. When the queue forms for the use of a single shared resource, as in W_1 and W_3 in Figure 2, the load on that resource only is required for the analysis. But when multiple shared resources are used simultaneously, hierarchical techniques are used for analysis, for example, W_3 is first calculated from the analysis of load on the shared resource #2. This enables one to determine the effective service time t_6 of the shared resource #1. W_2, the waiting

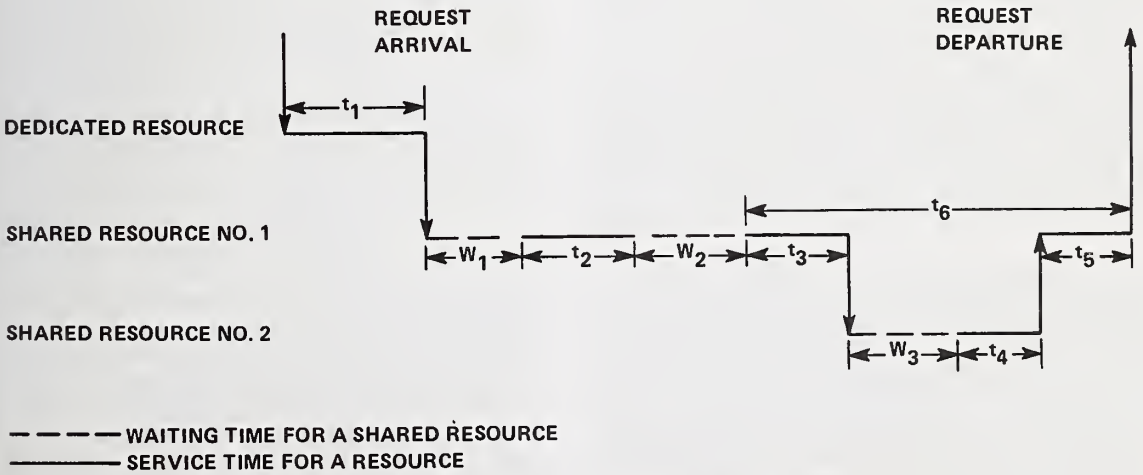


FIGURE 2. CRITICAL SEQUENCE TIMING DIAGRAM

time for shared resource #1 can then be calculated by the analysis of the load on this resource. This hierarchical technique is repeated as many times as necessary to evaluate the situation when two or more shared resources are held simultaneously.

An analysis of the main memory response time behavior, which was later supported by the measurements on the system, indicated that the variation in response time was very small. Hence, for the sake of simplicity, it is represented in the model by a constant value. The value chosen is on the conservative side based on the laboratory measurements of the system.

The byte channel I/O interface and the channel group processor service the requests by a strictly predetermined nonpreemptive priority discipline. For the channel group processor, the priority is based on the channel number of the channel making the request and the type of service required. For the byte channel I/O interface, the priority is based on the position of the device on the SELECT-OUT line and whether the device is wired for selection at SELECT-OUT or SELECT-IN time. Both the channel group processor and the byte channel I/O interface service one request at a time. The service time distribution of the requests at each priority level is determined from the analysis of system design and the use of hierarchical modeling where necessary. Successive arrivals are assumed to be independent of each other and a Markovian arrival pattern is assumed. Under these conditions, the M/G/1 nonpreemptive priority queueing theory developed by Takacs [1] is used to calculate the moments of waiting time distribution at each priority level.

The service discipline in the TP controller for the use of an interface adapter by the TP lines is very complex. It uses a dynamic priority algorithm and services one TP line at a time. For the sake of simplicity the dynamics of the priority algorithm are not modeled but are represented by a fixed nonpreemptive priority scheme. The priority of the TP lines are specified by the model user. The service time of the interface adapter for the different TP lines were obtained from the TP controller's designers. As with other shared resources, the successive arrivals to TP line are assumed to be Markovian in nature although it is known that they are not. With these simplifications and assumptions, the M/G/1 nonpreemptive priority queueing theory mentioned earlier is also applicable for the analysis of waiting times for the interface adapter. Again, hierarchical modeling technique, described earlier, is used as many times as required.

The heart of the model is the determination of the critical sequence response time distribution. The analytic queueing technique mentioned earlier enables us to determine n moments of waiting time distribution when the arrival rate and n+1 moments of the service time distribution of the priority server are known. The implications of this loss of one moment from queueing model input to output are demonstrated with the aid of example shown in Figure 2.

Let us say that we wish to generate n moments of waiting time, W , of the shared resource #1. Thus we require n+1 moments of the service time of this resource. But this service time consists of the waiting time of resource #2 and thus we require n+1 moments of the waiting time of resource #2. This will require n+2 moments of the service time of resource #2.

Thus we see that each application of the hierarchical technique results in a reduction of one moment. From the simplified critical sequence work diagram for TP line shown in Figure 3, and the previous discussion on hierarchical modeling, it may be seen that three levels of hierarchical representation are required to evaluate the waiting time for TP controller's interface adapter. It will be shown later that three moments of response time are used to obtain the approximated probability density function of the response

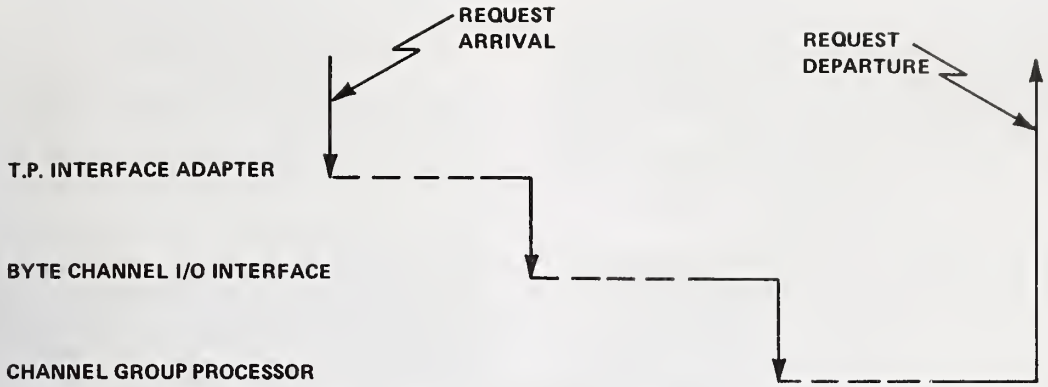


FIGURE 3. SIMPLIFIED CRITICAL SEQUENCE WORK A T.P. LINE SERVICE

time distribution. Hence, to determine the approximate response time distribution of TP interface adapter, we have to start with six moments of service time distribution of the channel group processor. This process leads us to have five moments of waiting time distribution for the channel group processor, and four moments of waiting time distribution for the byte channel interface. Thus more moments are available than are actually used in the model for the calculation of overruns on the block and byte channel devices directly. But these six moments are required to calculate the overruns on TP lines.

The principal output of the model is an estimate of the expected overrun rate for the specified input configuration and load. We have described how moments of response time are calculated for various overrunable sequences of work which we have called critical sequences. Having obtained these moments, and having determined the critical response time values for each case which, if exceeded, will result in overrun; the next step is to determine from this information the probability that the critical response will be exceeded.

This process has two conceptual stages. First, a set of moments are used to derive an equation which describes a probability density function consistent with the moments. Second, some approximation method is used to evaluate the integral of this curve beyond the critical value.

A typical approach used is to assume a normally distributed variable and to use the calculated mean and variance as parameters for transforming the value of interest into standard deviations of an equivalent unit normal distribution. Standard formulas are available for computing highly accurate approximations of this cumulative function [2].

This approach may be reasonable for roughly estimating areas that are typically greater than, say, 20% of the total cumulative distribution; particularly when the approximated distribution is roughly symmetrical. However, in the cases considered by the model, an area to be calculated is typically less than 1/10000 of the total area. Furthermore, typical response times are highly skewed toward the smaller values rather than being symmetrical like the normal. Therefore, this technique would generally result in estimates too grossly inflated to be of practical value. At another extreme, one might consider assuming exponential service times. This could result in gross underestimates where the actual distribution is more balanced.

The method used in the model is to assume that the response time is distributed according to one of the family of lognormal distributions. Lognormal curves begin at a finite lower bound, terminate at infinity, and possess a continuous range of skewness.

With knowledge of at least

three moments, the coefficient of skewness can be calculated and a lognormal distribution of the skewness can be fit. Any lognormal distribution can be transformed into an equivalent unit normal distribution, and the critical time can then be stated as standard deviations of the unit normal. Having calculated this transformation, the model uses a standard approximation formula for the normal cumulative which is accurate to five decimal digits. Finally, this probability estimate is converted to overruns per minute at the I/O load described to the model.

Model Validation

The model has had extensive use in the past year. In addition, its output was compared with measurements made using an extensive set of controlled test workloads. In one group of tests, I/O loading details were varied while holding total I/O loads at a relatively constant high level. In another group, a fixed set of I/O programs were operated at several load levels. This collection of measurements permitted extensive study of variations in workload and the corresponding model estimates. Included were tests involving solely the byte interface, solely block channels, and combinations of both.

Model Accuracy

On the byte interface, data overruns are typically of the type which cause some user inconvenience such as an aborted job. It is desirable that the total absolute number of such overruns be held below some specified rate in terms of time rather than throughput. Therefore, an important performance question with respect to a given configuration is what level of throughput can be achieved at a given overrun rate. In this context, a model should ideally give conservative overrun estimates, but less conservative than any simpler alternative estimating technique. We have found that the model consistently overestimates byte channel overruns but has been able to predict substantially higher safely achievable throughput rates than those predicted by the available alternatives.

On the other hand, DASD overruns merely impose a minor response time penalty on the work involved. Therefore, a more meaningful performance criterion is the percentage of successful operations, that is, those without overrun. We have found that the model consistently yields estimates with +0.1% of the percentage of successful operations.

A primary intended use of the model is the evaluation of alternative device arrangements on channel, or the amount of overrun increase to be expected due to increases in load. The model has been consistently accurate in predicting the magnitudes of such changes whenever the changes are substantial.

In addition to the above mentioned uses, the model has had the following uses:

1. A special-case version was developed to predict channel response times for a unique proposed customer application involving customer-written real-time channel programs.

2. Other special versions have been used to assist in evaluation of engineering design alternatives.

Summary and Conclusions

The available literature on system models, based on analytic queueing techniques, suggests that the current state-of-the-art is the derivation of the average values of response times for the shared resources. In our situation, the average response time is of no particular interest, and only those rare situations when the response times exceed a prespecified value are of concern. Hence, in our situation, a model which accurately calculates the tail of the response time distribution is required. Comparison of the model to lab measurements at known points on the

tail of the response time distribution, have shown that the model has the required accuracy. Hence, the model has been successful in predicting the occurrence of rare events arising in queueing situations.

The model was initially intended to be used by the IBM Systems Engineers for performing configuration analysis of IBM 303X I/O systems. Its use has extended beyond this requirement to include engineering design tradeoffs and to perform special systems analysis, (for example, I/O throughput analysis). We feel that the techniques developed within the model can be further applied to other queueing systems in predicting the occurrence of rare situations of large queue lengths.

References

[1] Lajos Takacs, "PRIORITY QUEUES," Operations Research, Vol. 12, pp. 63 (1964).

[2] Handbook of Mathematical Functions, National Bureau of Standards, 1964.

Configuration and Capacity Planning In a Distributed Processing System

K.C. Sevcik
G.S. Graham
J. Zahorjan

Quantitative System Performance
Toronto, Ontario and Seattle, Washington
and
Computer Systems Research Group
University of Toronto

The distributed health claims processing system of a major insurance company was based on hardware that soon proved inadequate for the processing load. A decision was made to replace the entire system with new hardware and software through an acquisition and development process scheduled to take three years. We were asked to undertake a study with the goals of (1) comparing alternative proposed configurations for the replacement system, and (2) assessing the adequacy of alternative transitional systems based on the existing software intended for use during the development of the replacement system. In this paper, we describe the latter aspects of the study. Using analytic modelling and simple asymptotic bound analysis, we were able to show that the performance of a proposed transitional system was likely to deviate significantly from the expectations based on intuition and information from the hardware vendor. Subsequent benchmark tests, motivated by the results of our study, conformed closely to our predictions. Without the information that resulted from our brief study, the company might have purchased hardware inadequate for its intended purpose.

Key words: Asymptotic bound analysis, benchmark tests, capacity planning, configuration analysis, queueing network modeling.

1. Problem Context

A major insurance company has decentralized its health claim processing, which is dominated by claims under group health insurance policies, by establishing mini-computer systems at approximately twenty geographically distributed sites. At each site, five to forty clerks with terminals enter and process claims. Each clerk can process about 80 claims per day. Telecommunications is used to provide data interchange between the head office and the remote sites.

A few months ago, it was determined that the systems in place at the sites were not providing adequate response, and that it was necessary to acquire more powerful hardware. The aquisition cycle plus conversion of applications systems was planned to take place in the next three years. Seven different mini-computer systems were proposed to serve as the basis of computing at each site. The company wished to reduce the field of candidates down to two or three before proceeding with benchmarking studies. In addition they wished to investigate several alternatives for transitional systems. Using the existing software applications systems, hardware upgrades were necessary to handle the workload between 1980 and 1983.

Two analytic modelling studies were carried out to support the company's decisions. In one, the simple technique of asymptotic bound analysis of a single class queueing network model provided insight into the capabilities of alternative proposed transitional systems.

In the second study, a single class queueing network model was used to predict the performance of each of the candidate configurations for the long-term system. We produced predictions of throughput and mean response time as functions of the number of active terminals, and of the number of terminals leading to the highest system throughput.

The remainder of this paper describes the modelling techniques and assumptions and the results of the first phase of this study, the evaluation of the alternative transitional systems.

2. Method

At the time our study started, the existing system was overloaded at most of the sites. Even with only a moderate number of active terminals mean response time was in the 30 to 60 second range. The productivity of the claims processing clerks was impaired by the slow response from the system. The vendor of the existing hardware had two newer systems capable of executing the same software as the existing system. The insurance company realized that a transitional upgrade to the existing system was needed because the workload was expected to increase steadily over the three year period during which the equipment selection and application system conversion for the long-term system was to take place.

Discussions with the vendor gave the insurance company the impression that replacing the existing system (EX) by either of the two newer systems (TR1 and TR2) would improve performance by estimated factors of

1.5 to 2 for TR1, and
2 to 3.5 for TR2

Of the twenty sites, the company wished to know which sites could be upgraded to the less expensive TR1 system and which would require the TR2 system to provide acceptable performance during the transitional period.

The information we were given to support our analytic modelling study included:

(1) system measurements of the existing system taken at several sites under "live" workloads,

(2) system measurements of the existing system taken during "benchmark" trials in which varying numbers of clerks entered transactions from fixed scripts,

(3) information from the vendor on some aspects of the relative performance of systems EX, TR1, and TR2.

From the live system measurements over 15 minute intervals at several sites, we were able to determine that each health claim processing transaction involved four to eight terminal interactions, with an average of five. Although there was some variation from site to site, the overall average service requirement per interaction was 4.6 seconds at the cpu and 4.0 seconds at the disk on the existing hardware. These numbers were obtained by dividing respectively the cpu busy time and the disk controller busy time over fifteen minute intervals by the total number of terminal interactions during the interval. The distribution of think time was highly skewed and could not be measured directly under live conditions. We used a rough estimate of mean think time of 60 seconds.

The "benchmark" tests were done on both the existing hardware and the more expensive of the two proposed transitional systems (TR2). A script designed to closely resemble health claims transaction processing was followed repeatedly by

clerks at terminals. On each of the two types of hardware, three 15 minute trials were done, in which one, four, and eleven clerks participated respectively. From these trials, we were able to (1) verify the speed ratios of both the cpus and disks between the EX and TR2 systems, and (2) quantify the growth of memory contention overhead with the number of terminals. No benchmark tests were done on the TR1 system because none was available in Toronto.

The vendor-provided information on the relative speeds of the three systems included the following. The cpu instruction rate of the TR2 system is 1.5 times that of the EX system, while the TR1 cpu is only .9 as fast as that of EX. Both the TR1 and TR2 systems use the same disk, which has an average service time approximately half that of the EX system disk. The rotational speeds of the two disks are the same, but the seek times are only half as long with the newer disk, and some io is avoided entirely in the newer system relative to the Ex system. The benchmark tests substantiated the 1.5 cpu speed ratio between the TR2 and EX systems, and indicated that the average disk service time was reduced by slightly more than a factor of two.

With the above information, we were able to estimate the service time requirements (without overhead) that would exist under live loads on the TR1 and TR2 systems:

	EX (observed)	TR1 (estimated)	TR2 (estimated)
CPU	4.60 secs.	5.12 secs.	3.10 secs.
DISK	4.00 secs.	1.88 secs.	1.88 secs.

3. The Model

Each terminal interaction cycle consists of (1) think time, (2) data entry, (3) processing, and (4) data output. The processing involves bursts of cpu activity interleaved between io operations. In representing the system as a queueing network model, we represent the terminals, the cpu, and the disk as the three congestion points. (Although some sites had two physical disk drives, the disk controller did not permit them to be active simultaneously. For this reason, having only a single disk service center in the queueing network model is appropriate.) Each clerk corresponds to a token that moves from service center to service center in the model. While the clerk is thinking and entering data, the token resides at the terminal service center. During processing, the token alternately visits the cpu and disk service centers. At the start of terminal output, the token moves back to the terminals service center. For more detail about the theory and applications of queueing network models, see [Gr].

The parameters required to specify the queueing network model are:

(1) the number of active terminals,
(2) the mean think time (including time to enter input and receive output),
(3) the mean cpu time per interaction, and
(4) the mean disk service time per interaction.

This queueing network model was analyzed in two ways for each of the three sets of parameter values representing the EX, TR1 and TR2 systems respectively. First, we carried out "asymptotic bound analysis" as described by Denning and Buzen [DB]. Next, we did a single-class analysis based on product-form queueing network analysis [DB]. In both cases we used the queueing network solution package, **THE**solver [GZ]. Both analyses are described in the following sections.

4. Asymptotic Bound Analysis

Asymptotic bound analysis is a simple technique for quickly determining upper bounds on throughput and corresponding lower bounds on mean response time as a function of system load. It is based on the service requirements of transactions at each of the system resources. With the definitions

L_i = total service requirement per interaction at device i,
Z = mean think time,
M = number of active terminals,
X = throughput, and
R = mean response time,

asymptotic bound analysis consists of the following relationships [DB]:

$$X \leq 1/L_b \quad [1]$$

$$R \geq \sum_i L_i \quad [2]$$

$$X \leq M/(Z+\sum_i L_i) \quad [3]$$

$$R \geq M \times L_b - Z \quad [4]$$

where b is the bottleneck device, such that $L_b \geq L_i$ for all i.

Equations [1] through [4] each have an intuitive interpretation. Equation [1] indicates that if each interaction requires L_b seconds at device b, then interactions can certainly be completed no faster than one every L_b seconds on average. Equation [2] states that mean response time is at least as great as the sum of service requirements at all devices. (This assumes no internal parallelism within a job, such as io/compute overlap.) Equation [3] is based on the fact that, with a single clerk active, the throughput is $1/(Z + \sum_i L_i)$ since each interaction has average length $(Z + \sum_i L_i)$. Adding more users will lead to contention, so the throughput rate can rise at most linearly with the number of users. Finally, equation [4] shows that at least time $M \times L_b$ is required to serve each user at the bottleneck device, so a user's time for a complete terminal interaction (which is $T + Z$) can be no smaller than this on average.

Inserting the service requirements for each of the three models into the four equations leads to the graphs shown in figures 2 and 3. A brief glance reveals that, at high loads, system TR1 is in fact *inferior* to the system EX. This, of course, is a consequence of the fact that system TR1 has a slower cpu in a cpu bound environment. Thus, rather than a performance gain factor of 1.5 to 2, performance degradation will be seen in moving from system EX to system TR1 whenever the number of active terminals exceeds some threshold (probably in the vicinity of fifteen terminals). Figures 2 and 3 indicate a substantial performance gain in moving from system EX to system TR2, although the gain in throughput cannot quite be a factor of two or more as originally expected.

5. Single-Class Analysis

In the previous section, we developed upper bounds on throughput and lower bounds on mean response times. By making a few additional modelling assumptions that have proven viable in many computing environments, we can obtain point estimates of throughput and mean response time. These point estimates coincide with the bounds at $M=1$ and asymptotically as M becomes very large, and form a smooth curve in the middle.

The additional assumptions made to obtain the point estimates were:

(1) Homogeneous Routing - The probability that a job completes after a particular cpu visit is independent of other jobs and of the number of visits that the job has made to the cpu, and

(2) Homogeneous Service Times - The time before a job's service completes at a device depends only on the job and the device.

With these assumptions, an efficient computational algorithm yields the throughputs and mean response times with various numbers of users [DB]. One queueing network solution package that incorporates this algorithm is **THE**solver [GZ]. The appendix shows the **THE**solver input to do both the asymptotic bound and single class analysis. Figures 4 and 5 show both the asymptotic bounds and the point estimates obtained from single-class analysis for throughput and mean response time respectively.

6. Conclusions

On the basis of our study additional benchmark tests were done in order to re-assess the advisability of involving system TR1 in the transitional plan. The benchmark studies of the TR1 system confirmed our prediction that the performance of TR1 became worse than EX once the number of terminals reached 10 or so, and the performance gain of TR1 over EX at lower loads was negligible. Consequently, there was no performance reason to invest in TR1 systems for any sites. The results of the benchmark tests are included in figure 5.

In this study, very simple queueing network modelling techniques proved valuable in evaluating the proposed hardware. Without the simple modelling study that we carried out, the company might have ordered TR1 systems without doing benchmark tests on them. In that case, they would have been destined for a great disappointment with the arrival of the new equipment. Instead, they are now in the process of installing TR2 systems at all processing sites.

7. References

[DB] Denning, P.J. and J.P. Buzen, The Operational Analysis of Queueing Network Models, *Computing Surveys* 10 (3), September (1978) 225-261.

[Gr] Graham, G.S., editor, Queueing Networks Models of Computer System Performance special issue of *Computing Surveys*, September (1978).

[GZ] Graham, G.S., and J. Zahorjan, **THE**solver User Guide, Computer Systems Research Group, University of Toronto (1980).

8. Appendix

Below are input statements to THEsolver that produce the information about the TR2 system in figures 2 to 5.

```
CO    'SYSTEM TR2 ANALYSIS'
CL    CLAIMS END_CL
DV    CPU DISK END_DV
LD    CLAIMS CPU 3.10
                 DISK 1.88
                 ***
      END_LD
AB
SN    CLAIMS 1 TO 32 END_SN
TH    60
LF    1 TO 32 END_LF
QU
```

The purpose of the commands are as follows:

CO -comment
CL -there is one class called "claims"
DV -there are two devices called "cpu" and "disk" respectively
LD -the service time requirements per terminal interaction for health claims are 3.10 seconds at the cpu and 1.88 seconds at the disk
AB -determine upper bounds on throughput and lower bounds on mean response time
SN -solve for throughput at multiprogramming level from 1 to 32
TH -mean think time is 60 seconds
LF -solve for system throughputs and mean response times for numbers of active terminals from 1 to 32.

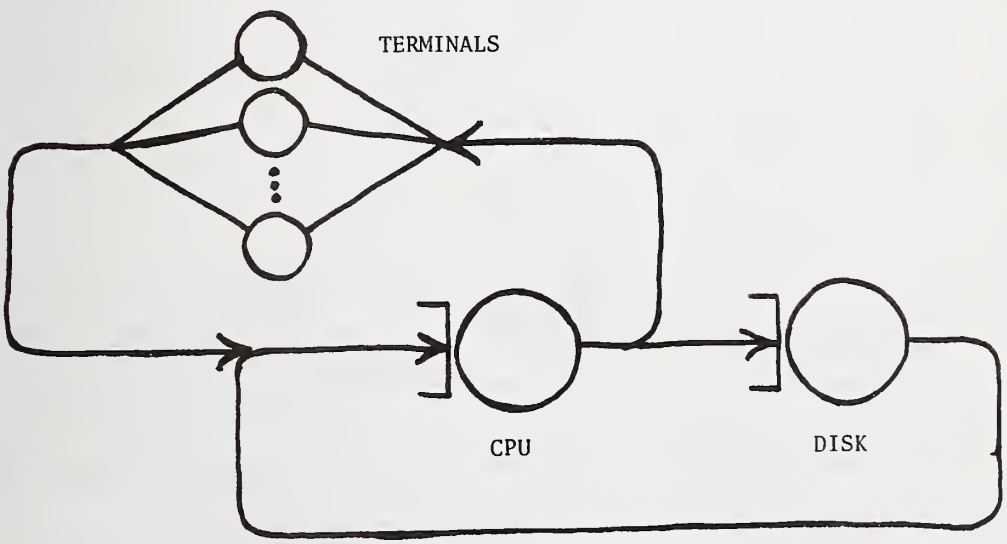


Figure 1. The Queueing network model

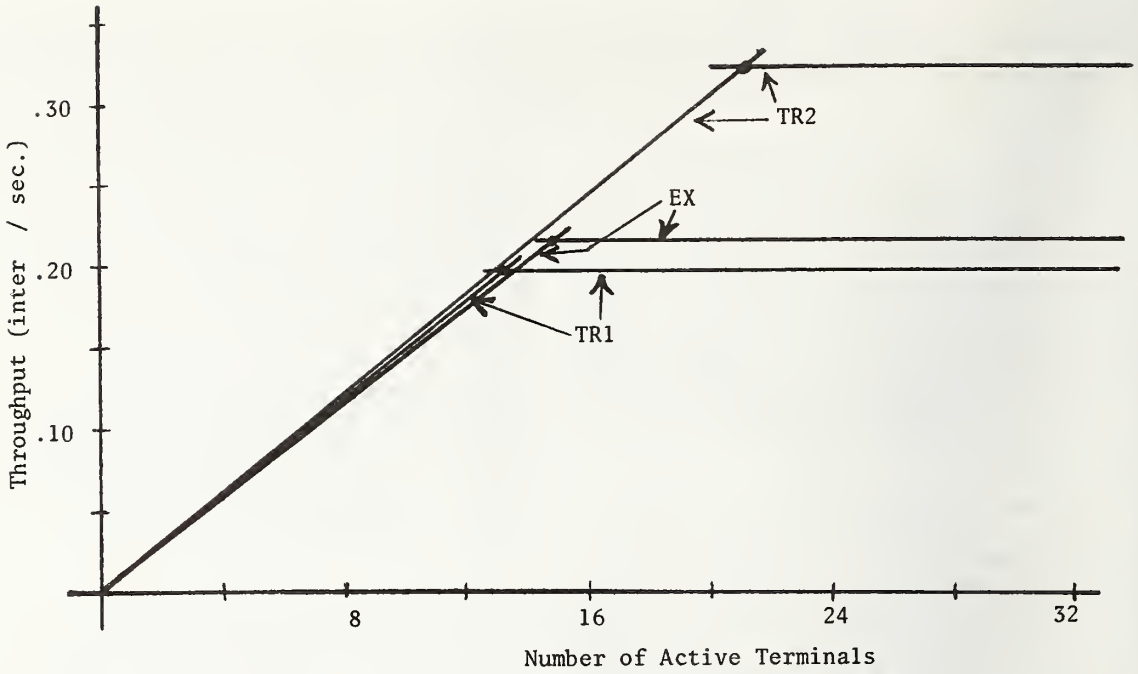


Figure 2. Upper bounds on throughput.

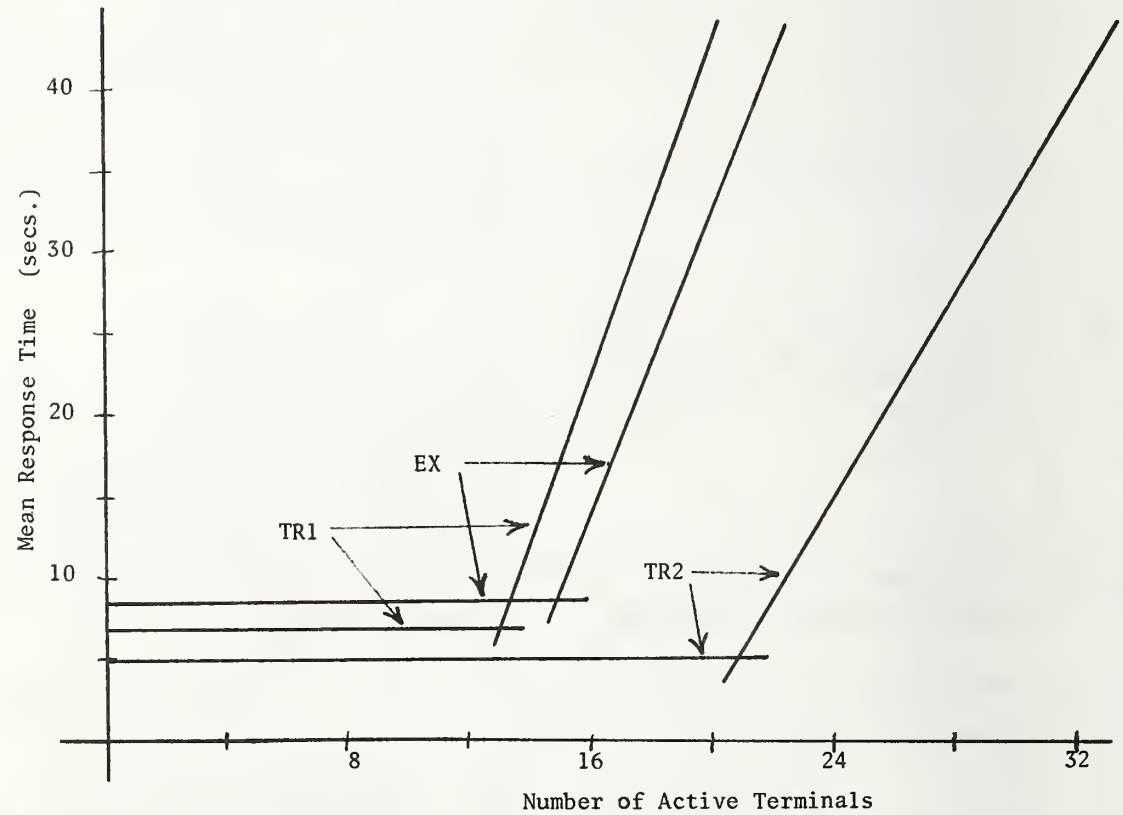


Figure 3. Lower bounds on mean response time.

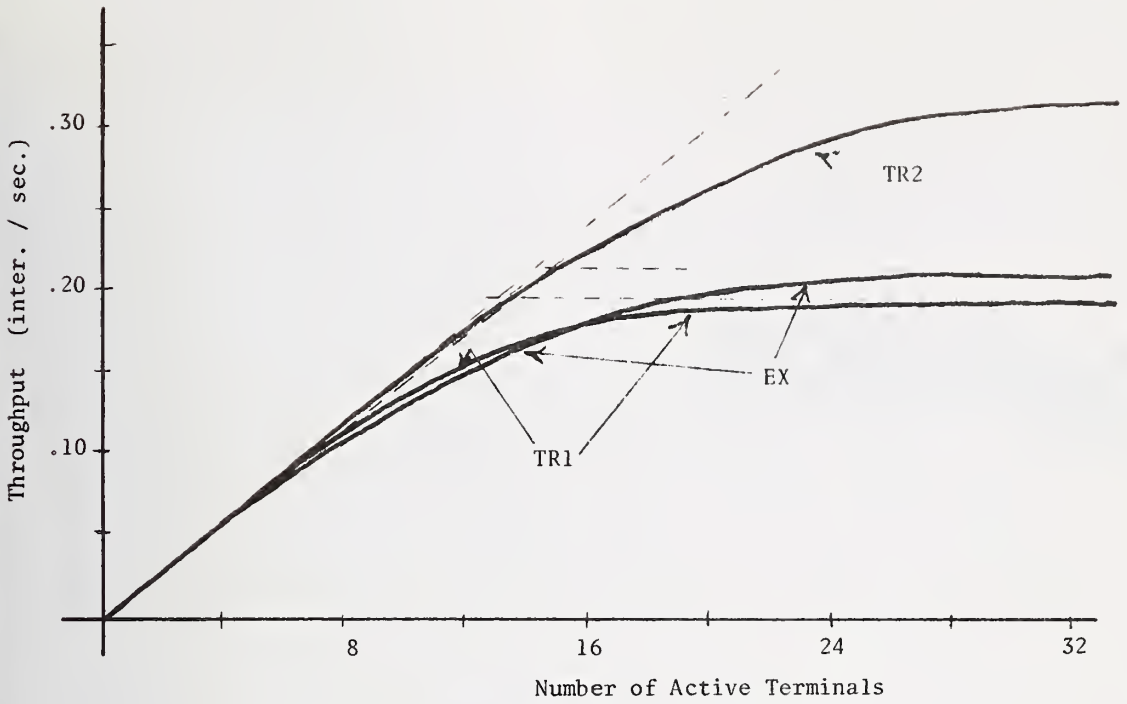


Figure 4. Estimates of throughput.

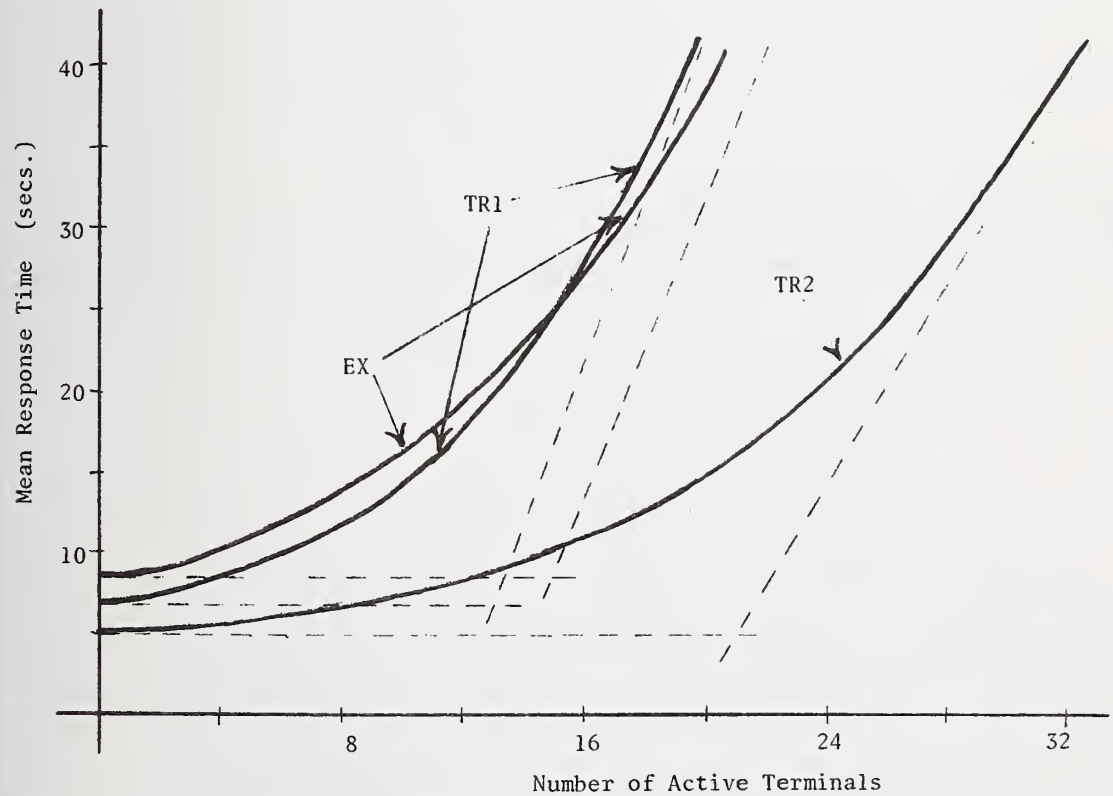


Figure 5. Estimates of mean response time.

CPEUG80

Capacity Planning

File Allocation Methodology for Performance Enhancement

Sujit R. Kumar
Digital Equipment Corporation
Maynard, Massachusetts

Robin B. Lake
Case Western Reserve University
Cleveland, Ohio

C. Tom Nute
General Dynamics Corporation
Fort Worth, Texas

Abstract

Storage device configuration is an important issue in the performance of computer systems. The designer or installation manager has to map user and system demands for storage space across the range of I/O devices available on the system. We present here a methodology for optimizing such mapping. A generalized performance model for interactive systems is developed and validated. The model incorporates the device organization, and given the descriptors for storage devices, predicts performance figures for a broad range of system configurations and workloads. Alternately, given the target performance figures for the installation, it helps work out alternative system configuration for improved performance. We present cost benefit tradeoff studies based on hypothetical scenarios to determine optimal file storage strategies from the point of view of the user as well as the installation.

Key words: I/O Resource Allocation; Systems Performance Modeling; Logical Storage to Physical Device Mapping; O.S. Performance Prediction; File Binding; Systems Storage restructuring; Storage Partitioning; O.S. Tuning.

1. Introduction

A modern computer system is best considered as a collection of resources. The various funtional components of the computer system -- central procesor, memory, peripheral devices, system utilities -- are the resources of the system, and the system's performance depends on the effectiveness with which these resources are brought together by the hardware and deployed by the software to perform specific tasks for the user of the system.

Performance aspects -- namely performance measurements, analysis, modeling, tuning, and prediction -- have been the focus of increasing interest lately (c.f. bibliographies compiled by [Agajanian-76], [Mohan-78]). The roots of this trend can be traced to the early

annals of Computer Engineering. Rudimentary operating systems were first evolved in an effort to improve the utilization of the increasingly costly computer hardware. These operating systems were just job schedulers for the CPU and I/O channels trying to keep these resources working at capacity as much as possible. The benefits of multiprogramming in a computer installation were then practically realizable. Thus, from the very beginnings of the computer, performance issues have motivated, guided, and decided basic trends in its evolution.

2. Storage Allocation

The peripheral storage devices in a computer system cater for permanent system and user files; they also provide space for swapping, spooling, temporary system and user files necessry for compilation, inter-process communication ('pipes' and 'filters' in UNIX[1]), assemblies, and so on. Because the performance of any computer system is dependent on the speed at which information in these areas can be accessed, the distribution of these accessing demands across the various classes of devices is an important issue in the analysis of system performance, and has been the goal of our research [Kumar-80].

3. Analytic Modelling

Analytic modeling seeks a set of equations relating the chosen performance measures to the parameters of the system. A simulation approach is often used when these equations are not soluble, and cannot be expressed in a closed form, or when it is not obvious what these equations are. An analytic model of a computer system usually consists of a graphical, job-flow representation of the system with a set of mathematical relationships of its various parameters. Our goal is a model which describes storage usage and can be used to investigate the performance impacts of alternative peripheral storage configurations in a computer system.

The point we wish to emphasize is that a variety of resource allocation usage policies must be interpreted in tractable terms and factored into the overall model before we can express the inter-relationship between storage characteristics and the performance measures we are interested in. We will term such a model an 'integrated' one. Providing the ability to express different system policies and device configurations makes the model a generalized one. We present next such an integrated, generalized, analytical (IGA) performance model for interactive computer systems.

3.1 The IGA Model

Figure 1 presents the generalized computer system model. The model is a closed network one as we are mainly interested in interactive computer systems where a fixed number of terminals, proceeding in think-wait cycles, submit jobs to the system.

Devices are partitioned into numbered resource classes. The numberical subscript, if any, referes to the particular device within the class. Thus device 1 is the cpu; 0 refers to the outside world (user terminals); 1_3 represents the third cpu in a multiprocessing environment.

Jobs submitted to the system are loaded into main memory when space is available according to the job scheduling policies for the particular system. Jobs resident in memory and waiting to be processed are given control of a cpu according to the system's cpu scheduling policies. In the UNIX Operating System (OS), this cpu switching policy is a preemptive round-robin within priority classes, with a one-second time slice limit.

The mechanism outlined above describes the situation where a job is assigned to s single cpu; note, however, that parallel processing is not precluded by the model; parallel processing may be modelled, without loss of generality, by assigning multiple cpu's to a single job. Alternatively, each job may be assumed to

---

[1] Trademark of Bell Laboratories [Ritchie-74]

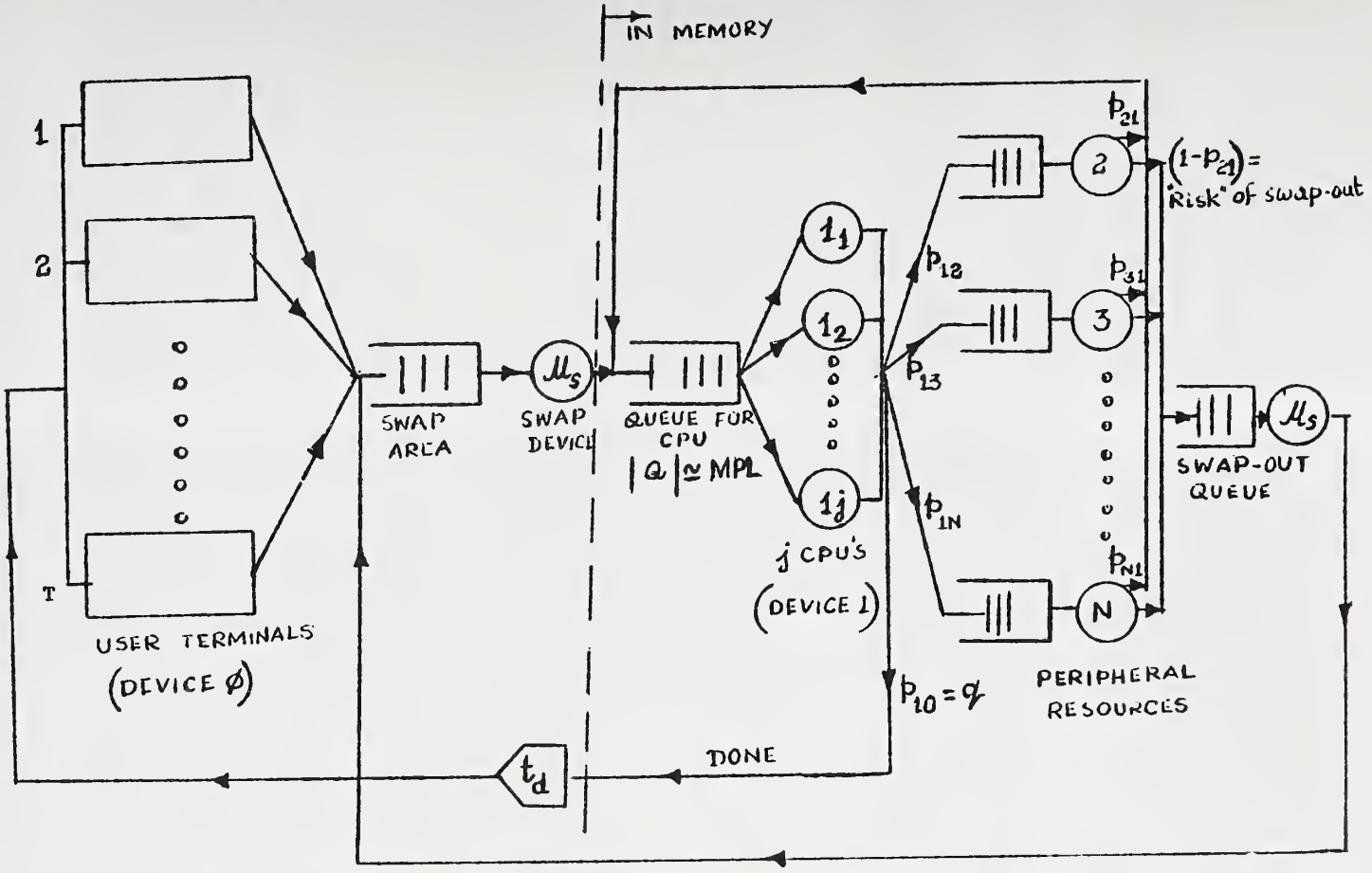


Figure 1. Generalized Computer System Model

'spawn' multiple processes, each assigned a cpu for execution.

In general, the model shown in figure 1 imposes no assumptions regarding the degree of multiprograming. In the UNIX OS the multiprogramming level (MPL) is a variable quantity which is determined by the total amount of available memory and the size of each job. The effective MPL at any instant in a multiprocessing environment is the total number of active jobs normalized by the number of active processors -- the degree of multiprocessing.

A job in the system may require service from resources other than the cpu. The central server processing network imbedded in the model accomodates these resource demands also. From the cpu, the job may proceed to any of the peripheral resources shown. Service at these stations progress according to the requirements set by the job and the operating system policies. Each of these peripheral nodes may, in turn, be decomposed into a subsystem modeled with multiple servers and queues [Courtois-75]. However, in the integrated model each is represented by a single node with its associated queue.

While being serviced or waiting for service by a peripheral device, the job is precluded from consideration by the cpu switcher. These 'blocked' memory-resident jobs may be removed from main memory, if required, to swap-in ready jobs. Such swapping is intitiated by the scheduler in the UNIX system and, in general, is dictated by the operating system policies as well as the mix of jobs present in the system. Associated with each I/O device access is a finite 'risk' probability of being evicted from main memory. These probabilities vary from device to device. If it is not swapped out, a job when ready to run again waits its turn on the round-robin queue. If swapped out, usually to a peripheral storage device, the job encounters an additional delay waiting its turn To be loaded into main memory once more.

It is possible that the cpu service burst required by the job exceeds the time slice allotted. Such quantum run-out is a frequent occurence for cpu-bound jobs; when this occurs, the job will be preempted to allow others at higher or the same priority level to execute. Once preempted, the job queues up for its next turn at the cpu in the round-robin queue to which it is assigned. At this juncture, at the discretion of the memory scheduler, the job may be swapped out of memory altogether if there are ready jobs in swap-area waiting for memory space. This is represented in the model by a dummy resource access on quantum run-out.

Termination of executing jobs is modeled by a quitting probability, q, which routes the job out of the system:

$$q = p_{10} = 2 - \sum_{i=2}^{N} p_{1i} \qquad (1)$$

There is a delay (for system housekeeping) after which the appropriate user is notified of termination. The user, after an appropriate think-time, submits a fresh job to the system. The user terminals may thus be viewed as yet another peripheral node in the system; the service time of this node corresponds to the user's think time. The number of jobs circulating in the system is a function of the number of users. In systems where multi-tasking is not permitted, the number of jobs is exactly the number of users present.

The mechanism outlined above is sufficiently general to be valid for a variety of interactive systems. We have discussed how the UNIX system can be described in terms of the general model and shall apply the model with simplifying assumptions, to the UNIX OS to derive performance measures such as response times from the devices and workload parameters of the installation.

The analytic model was validated for a PDP-11/45 UNIX system. The UNIX system Instrumentation and its control and trace data reduction utilities developed for these studies are being distributed by the authors through the UNIX-user's group; operational analysis techniques [Buzen-78], [Denning-78], were used to simplify the analysis of the IGA model.

The choice of UNIX for validating and applying the model was dictated by its availability and usage convenience, but it has turned out to be a fortunate choice, as UNIX though primarily implemented on small machine, has 'big league' characteristics as far as features are

concerned. So, it is more demanding in terms of modeling versatility and most other operating systems have only a subset of these features.

The workload at any computer installation has a definite pattern of I/O data access requirements. These I/O demands are serviced by the peripheral devices configured in the system. Our performance model predicts the response time encountered by users of the interactive installation by analyzing the characteristics of the workload and the resources present at the installation. The system designer and the installation manager can improve user response times by employing the performance model to determine the optimum mapping strategy between I/O demands and peripheral devices.

4.Some Experimental Results

The starting point for the series of performance evaluation experiments was the normal system configuration (C0) with 3 peripheral clusters present -- the fixed head disk (HS), the multiple platter drives (DV), and the removable disk packs (RK). Usually the removable disk packs are reserved for user-mountable file systems. The logical I/O system mapping for this configuration is shown below:

```
/     ->        HS

swap  ->        DV

/tmp  ->        DV

/usr  ->        DV
```

In the UNIX system "/" refers to the root directory of the tree structured file system. User directions reside in /usr; so /usr/Kumar would be the pathname for a particular user's area. All devices are assigned logical files. Thus writing to /dev/tty3 would accomplish the logically expected result system priveleges permitting. /tmp is used by the OS for scratch files in compiles, edits and other system utilities.

The twin DIVA drives are logically partitioned into 6 different subdevices (dv0 through dv5); our system instrumentation traces activity at the logical device level, enabling us to differentiate between, say, /tmp and /usr activity even though they are mapped to the same physical device. Note, this would be difficult to implement with hardware system instrumentation.

Figure 2 tables parameters for some common storage devices. The normal configuration C0 has 124K words of memory of which the lower 48K is 1 sec core and the rest is 750 sec MOS. The usual complement of card readers, lineprinter, magtape, crts and remote dial-in ports are also present.

The configuration was exercised by running benchmark scripts built through analysis of the actual user load on the system, to simulate typical user demands on the installation for controlled and reproducible experimentation. Runs were made simulating different degrees of user loading up to the maximum possible loading. The limit was set by system process table size which decides the maximum number of current process that may be accomodated. Different think times were set in the user scripts run to evaluate response changes with different user characteristics. The analytic performance model was then solved for the particular system configuration, with parameters determined by the system instrumentation, to get response time and system throughput measures under different user loading conditions. As will be demonstrated, these measures match the actual system performance closely, validating the analysis of the model.

Figure 3 shows the performance model for the normal configuration C0. The synopsis of data from a benchmark run for this parameter configuration is given in Figure 4. The parameters for the model have been obtained from system instrumentation data analysis utilities developed.

Figure 5 plots the predicted and actual measured response for different loading conditions. At lower values of system loading, the model's predictions are consistently found to be conservative for this and other configurations analyzed. That is, the measured response is better (lower) than that predicted. At higher levels of loading, the model predicts slightly better response than that actually measured by benchmark runs. This may be due to any of the following factors:

DEVICE	TIME IN MSEC.		
	SEEK	LATENCY	TRANSFER OF 32K AREA
DIVA (DD25)	10-55	12.5	255
RF11	0	17.0	550
RK05	50	20.0	835*
RP03	29	12.5	325
RS04	0	8.5	70
EMU	0	6 μs set up time	40

*2.84 msec/256 word transfer time

Figure 2. I/O Device Parameters

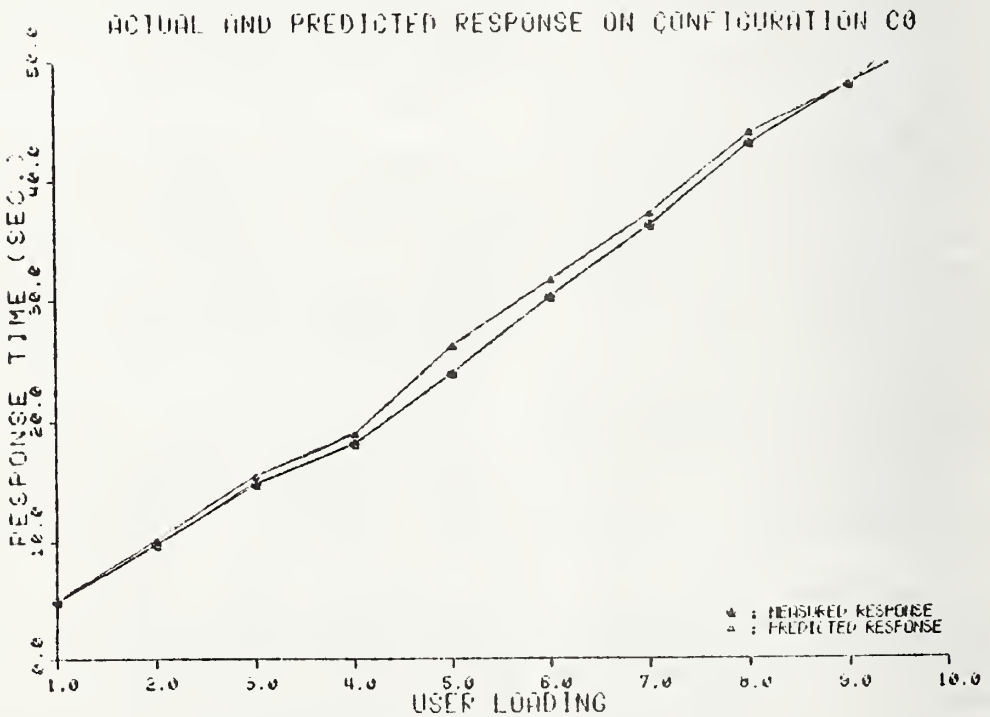
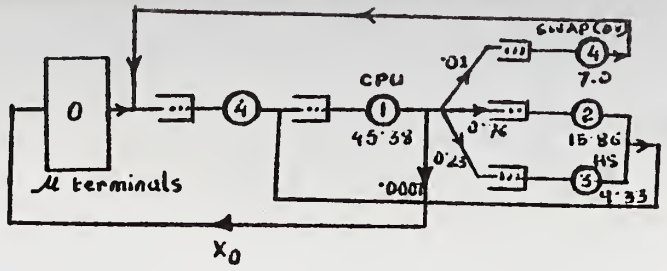


Figure 5



Real: 6:19
Proc: 5:8.7
η_{CPU} = 81·45%
MPL = 1·23

V_1= 7365
V_2= 5564
V_3= 1688
V_4= 112

V_0 = 1=0.0001V_1
V_1 = $V_0+V_2+V_3+V_4$
V_2 = 0.76V_1
V_3 = 0.23V_1
V_4 = 0.01V_1

S_1 (CPU MST) = 45.38 msec
S_2 (DU MIST) = 15.86 msec
S_3 (HS MST) = 9.33 msec
S_4 (Swap MST) = 66.0 msec

Overlaps

-42.45	V_1S_1	= 334.22
	V_2S_2	= 88.25
- 7.76	V_3V_3	= 15.75
- 1.63	V_4V_4	= 7.40
393.87 AdjΣ	Σ	445.62

$V_b S_b$= 334.22 Now $R \geq MV_bS_b - T_t$ i.e. $4.46 \leq R \leq 5.94$ 5.25 secs.

5.05 secs.

Lower bound on Response time (R)$\geq$ 334.22/75 $\geq$ 4.46 secs.

Actual observed Response per command = 5.05 secs.

$\eta_1 = \frac{334.22}{393.87}$ = 84.86% $\leftrightarrow$86.29% (from instrumentation)

Upper bound on response $= \sum_{1=1}^{4} V_1 si/75 = 445.62/75$ = 5.94 sec

Tighter upper bound (knowing overlap behavior) = 393.87/75 = 5.25 secs.

Figure 3a. CO Analysis

5 Script

$V_1 = 53502$
$V_2 = 43691$
$V_3 = 9280$
$V_4 = 530$

$S_1 = 34.22$
$S_2 = 16.20$
$S_3 = 9.60$
$S_4 = 39.42$

$V_1S_1 = 1830.98$
$V_2S_2 = 707.69 - 679.38$
$V_3S_3 = 89.05 - 86.19$
$V_4S_4 = 73.89 - 65.70$

$\Sigma V_iS_i = 2701.61$

Adjusted $\Sigma V_iS_i = 1870.34$

$V_0 = 1 = 0.0001V_1$
$V_1 = V_o+V_2+V_3+V_4$
$V_2 = 0.8166V_1$
$V_3 = 0.1735V_1$
$V_4 = 0.0099V_1$

Real= 30:02.0
Proc= 28:46.50

$\eta_{cpu} = 95.81\%$ MPL = 5.22

$\eta_1 = 97.90\% \longleftrightarrow 98.37\%$ from Instrumentation

Now $V_bS_b = V_1S_1$ ... CPU is bottleneck . ∴ $R \geq 1830.98/750 \geq 24.41$ secs per command

Actual Observed Response True per command = 24.03

From 1 script w/o sha : for M= 5 terminals $R \geq MV_bS_b - Z \geq 5 \times 334.22/75$ secs/command

≥ 22.28 secs.

$22.28 \leq R \leq 26.25$ secs.

↓

24.03

From statistics observed here, a tighter upper bound is realizable adj. $\Sigma V_iS_i = 1870.34/75 = 26.94$ secs.

i.e. $22.28 \leq R \leq 26.25$ secs

↓ 24.94

24·03 secs.

Figure 3b. CO Analysis

	1Script	5Script
Total Duration	387.30 secs	1861.32 secs
CPU Idle	13.71% (User - 24.37% Sysint - 2.21%)	1.63%
Scheduler Idle	99.13	97.03% Blocked = 0.27%
Process Swaps	112, .07 secs/swap Size = 1709.15	530, 0.14 sec/swap Size = 2234.51
Swap	Active = 1.91% Overlap 0.42% No queuing for swap	Active = 3.97% Overlap = 3.53% Queuing 0.27%
Total Disk Interrupts	7364	53501
DV's Busy	88.25 secs	707.69 secs
Total Transfers	5676 (2018016 wds)	44221 (23537600 wds)
Overlap with CPU	10.96%	36.5%
HS Busy	15.75 secs	89.05 secs
Total Transfers	1688 (432128 wds)	9280 (2375680 wds)
Overlap with CPU	1.98%	4.63%
HS-DV overlap	0.56%	2.72%

Figure 4 Configuration - CO: Data Synposis

a) Increased contentions for data base accesses, memory, and cpu at higher degrees of loading and increased process management overheads on the part of the operating system, which were being neglected, become noticeable now.

b) To reduce the interaction of concurrent scripts (prevent them from falling into step) activation are delayed (by 10 or 20 seconds) in the benchmarks when the run is started. Thus the level of interference due to other jobs varies throughout the benchmark run and the actual measured values reflect this.

5. Cost/Performance Benefit Tradeoffs in I/O Mapping Strategies

In this section we shall examine allocation policies for using a fast, solid state device, the Extended Memory Unit (EMU). The EMU[1] is a plug compatible replacement for head-per-track disks and its salient parameters are shown in Table 2. It features a zero seek time and very fast data trasfer rates.

From the System Instrumentation we observed that data transfers between processes and peripheral devices occurred in 256 word blocks. The swap sizes however, were considerably higher. For data accesses, the seek and latency times constitute a large proportion of the total activity, as the transfer time for a single block (512 bytes) is 2 or 3 msecs. only, even for a slow device. In the case of swapping, fast accesses (seek and latency) are not as important because the transfer time accounts for a major portion of the peripheral activity. Devices like the EMU which are non-rotational and have no seek or latency times (requiring a set-up time of a few μsecs. only) are thus more attractive for these short data accesses.

We observed that there is a high degree of contention in data accesses. Besides mutual interference in accesses, potential overlap with the processor is also denied by his physical device contention on independent logical I/O accesses. The best response time improvement would come from mapping the /usr files on to the EMU. However, this is not feasible due to space constraints.

[1] EMU is a trademark of Monolithic Systems Corp., Englewood, Colorado.

An alternative proposal is to have a partion on EMU reserved for use as a user assignable volume. We shall investigate the merits of this proposal next and establish optimal strategies for the user applicable in any interactive system.

Performance improvements are realizable with this proposal, because of 2 factors:

1) Faster data access with the EMU, reducing processor idle times in waiting for data.

2) Higher degree of overlap from splitting the user's logical files over different devices, and reduced intereference between different accesses to the same device.

We have observed that mapping some accesses even to a slower device may be benefical. Factor 2, outlined above, explains why this happens.

Assume that t_n is the access speed (blocks/sec) for the new device and t_o is the access speed for the old device. Let the total user area be b, such that b_n blocks in the user area may be partitioned on to the new device, leaving b_o blocks behind in its original position. Assume also that the capacity of the new device is B blocks and $B \geq b_n$. Let T_o be the run duration on the older configuration.

Assume that P_o was the degree of device-cpu overlap in the original confguration. During this a total of $P_o t_o T_o$ blocks were accessed on the old device. Assume that accesses were uniformly distributed over the user area. Then after reconfiguration

$$PoToto \frac{bn}{bo+bn} \left(\frac{1}{tn} - \frac{1}{to}\right)$$

represents the time savings on device-cpu overlapped accesses to the fraction of user space allocated on the faster device. Then, the performance gain realizable, in terms of savings in access and overlap times is represented by:

$$\frac{bn+bo}{to} - \frac{bn}{tn} - \frac{bo}{to} + PoToto \frac{bn}{b}\left(\frac{1}{tn} - \frac{1}{to}\right) \quad (2)$$

There is also an additional savings realized by overlapped accesses of the two user areas. This is a second order effect and if Pon represents the degree of overlap between the old and new user

areas, the total reconfiguration gain in terms of time, is:

$$ToPon + bn\left[\left(\frac{1}{to} - \frac{1}{tn}\right).\frac{1}{b}.(bo+bn-PoToto)\right] \quad (3)$$

Expression (3) above is a polynomial of the form:

$$A + Bbn + Cb_n^2 \quad (4)$$

The installation's cost accounting policy decides the additional charges levied with EMU usage and thus has to be balanced against the benefits realizable. In general, there are two ways of assigning charges:

1) A flat usage fee when the volume is assigned to the user.

2) Prorated usage cost function.

The proration may be a linear one based on space and time usages. Often, there is an initial assignment cost built-in also (a minimum charge). Alternatively, an exponential size or time proration algorithm may be used at the installation for load levelling and for promoting equitable usage (discourage hogging the resource).

Figure 6 shows the Cost/Benefit tradeoff for a family of normalized benefit polyomials and different usage charge policies enforced at the installation. The curves A, B, C, D are the savings realized by allocating different proportions of the user's logical file b_n to the EMU. We have illustrated 3 different charge policies -- a flat fee EMU assignment cost, a size prorated linear cost function, and a non-linear combination of the two to promote equitable sharing of the EMU amoung the user population. We see that B just breaks even marginally when 100% of b_n is mapped to the EMU. A is always a losing proposition unless there is additional file area space partitioning. In fact, for the system policies shown, any benefit curve lying in the area below B is not cost justifiable.

With the benefit characteristic C, the flat charge policy dictates that EMU allocation greater than 70% of b_n only are justifiable, whereas with the linear usage cost prorated function, only allocations under 70% are justifiable. An interesting threshold effect is observed for D, when there is a exponential usage charge function. Here the strategy should be to allocate between 45% and 65% of normalized b_n on to the EMU or simular device.

The thresholds above determine break-even points for operation, when the response time benefits achieved through reconfiguration evenly balance the costs incurred in such an operation. The user may derive optimal operating strategies from the preceeding analysis. Thus, for the last case an allocation around 60% gives the best performance gain.

6. Conclusions

The use of analytic models for performance studies is well accepted ([McKinney-69], [Browne-77]), we have demonstrated its use as a tool to study and reconfigure I/O accessing demands, for system performance enhancement. The methodology for this is based on the simple premise that during the execution of a job, various system and user areas residing in peripheral storage devices have to be fetched or updated, and the rate at which this information can be accessed is an important factor affecting the performance of the computer system. In addition to the raw speed limitations of the peripheral devices, interference and contention between accesses initiated by the same job, or by other concurrently executing jobs, introduce processing delays. We have, then, certain accessing demands for peripheral storage, and a specified set of devices available, the problem being the determination of an optimal mapping strategy between demands and devices. This issue has not been studied to date, and our generalized performance model served as an excellent tool for such activities.

The performance model developed is simple to use and has proved to be an accurate means of performance prediction and evaluation. It was formulated for flexibility and generality, to serve as an analytic tool in performance studies, and is sufficiently general to be set up to represent any interactive system.

The I/O mapping methodology developed is applicable to any computer system, the only requirement is that some means of reconfiguring the system storage areas be available. On systems with tree-structured files (e.g., UNIX) the mapping between storage area accesses and physical devices is an explicit one and is easier to restructure as the logical partitioning is distinct. In this

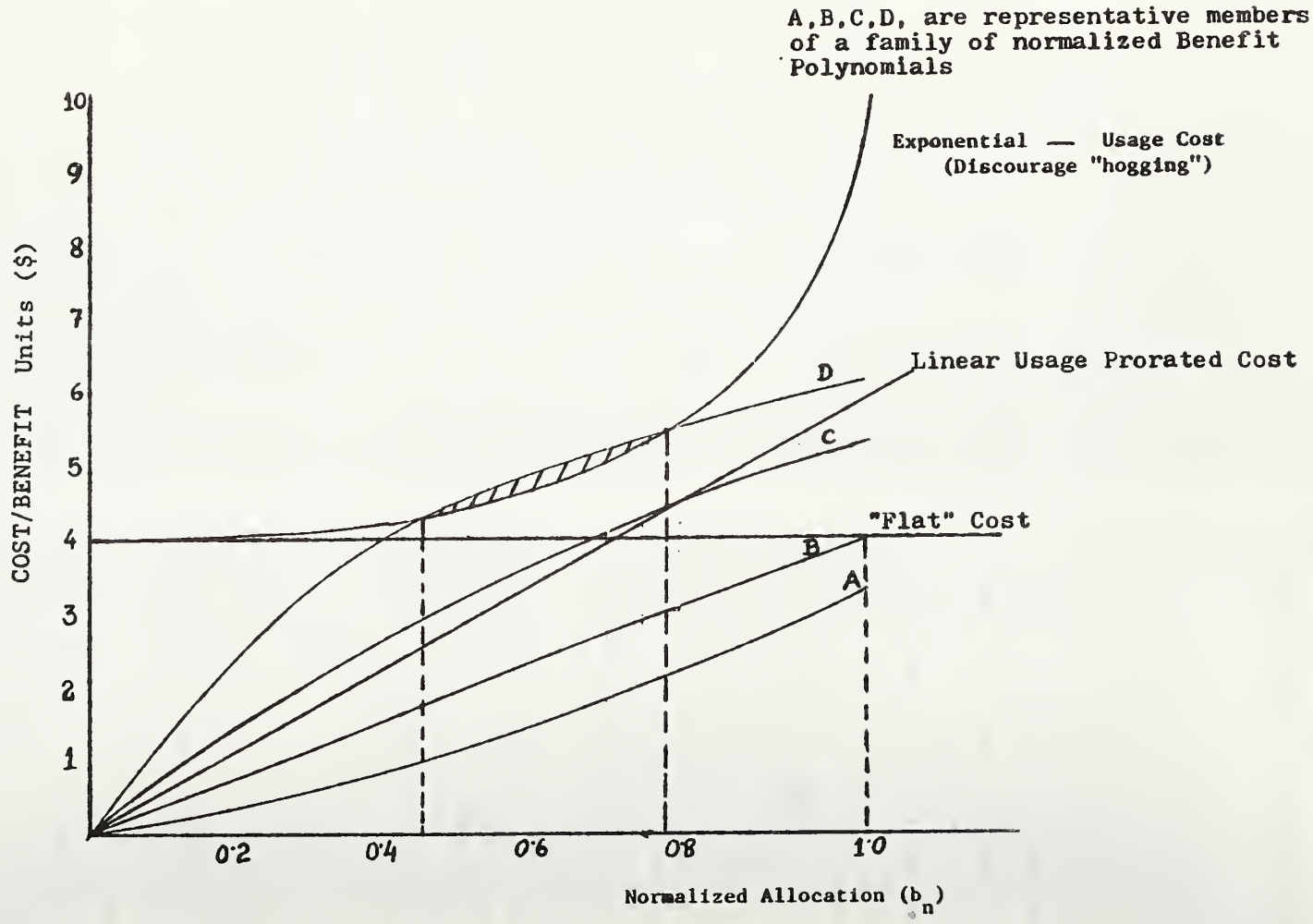


Figure 6 COST/ PERFORMANCE BENEFIT PLOTS

context, we may point out that it is not necessary to map a storage area to a faster device to obtain improved performance. We observed some cases where reassigning a storage area to a slower device results in response time improvements because of reduced contention delays, and improved overlap.

We observed that using the EMU as a swap device gives only marginal response time improvements, and that too at higher degrees of system loading when there is abnormal swap activity. The best strategy for performance enhancement would be to allocate the short 'bursty' transfers to devices like the EMU. Thus, on the UNIX system response time improvement is expected if the user areas are mapped on to the EMU or similar peripheral devices.

As this may not always be feasible, because of size constraints, we suggest a user-mountable volume strategy, outlined and analyzed before, as a feasible technique for response time improvement. Allocating the root directory (/) of the tree-structured file system on the EMU would also produce a significant response time improvement for the entire user community. From the point of view of system throughput, this may be a better strategy, especially when the mount-volume approach does not provide enough space for multi-user assignments.

Accesses to the paging device in a paged memory environment is another example of frequent, short accesses. The EMU (or similar device) should yield significant performance improvements when used in such an environment; however further research and experimentation is required to substantiate and quantify the degree of performance improvement.

In full swapping environments, when space is not any constraint, the configuration strategy suggested is to partition the EMU or similar peripheral, into two individually ported (dual-ports) file subsystems, one for global access areas (the root in UNIX) and the other for user-assigned file allocation. For systems with small memory configurations, where heavy swap-loads occur at moderate user loading, using the second partion as a swap device would improve the response times for all users currently on the system, instead of favouring ony those assigned to the mountable partition. If dynamic reconfiguration were feasible, an optimum strategy would be to assign the second partition for system use -- swapping -- whenever a predetermined load threshold is exceeded.

7. Acknowledgments

The authors thank the Computer Science and Biometry Departments at Case Western Reserve University, Cleveland, Ohio for the cooperation and facilities extended. Funding and equipment for the experimental portions of this research were provided by the Department of Biometry.

8. References

[Agajanian-76] Agajanian, A.H., "A Bibliography on System Performance Evaluation", ACM Sigmetrics, January 1976.

[Brown-77] Brown, R.M., Browne, J.C., Chandy, K.M., "Memory Management and Response Time", Communictions of the ACM, Vol. 20, No. 3, March 1977, pp. 153-165.

[Browne-75] Browne, J.C., Chandy, K.M., et. al., "Hierarchical Techniques for the Development of Realistic Models of Complex Computer System", Proc. IEEE, Vol. 63, No. 6, June 1975, pp. 966-976.

[Buzen-78] Buzen, J.P., "Operational Analysis: An Alternative to Stochastic Modeling", Proc. Int. Conf. on Performance of Computer Installations, North-Holland Publishing Co., 1978, pp. 175-194.

[Courtois-75] Courtois, P.J., "Decomposability, Instabilities, and Saturation in Multiprogramming Systems", Communiations of the ACM Vol. 18, No. 7, July 1975, pp. 371-376.

[Courtois-77] Courtois, P.J., Decomposability: Queueing and Comuter System

Applications, Academic Press, 1977.

[Denning-78] Denning, P.J., Buzen, J.P., "The Operational Analysis of Queueing Network Models", Computing Surveys, Vol. 10, No. 3, Sept. 1978 pp. 225-262.

[Kumar-80] Kumar, S.R., "An I/O Resource Allocation Methodology for Performance Evaluation and Enhancement Studies of Interactive Computer Systems", Ph.D. Dissertation January 1980, Case Western Reservce University, Cleveland, Ohio.

[McKinney-69] McKinney, J.M., "A survey of analytic time sharing-models", Computing Surveys, Vol. 1, No. 2, June 1969, pp. 105-116.

[Mohan-78] Mohan, C., "Survey of Recent Operating System Research, Design and Implementation", Operating Systems Review January 1978.

[Ritchie-74] Ritchie, D.M., Thompson, K., "The UNIX Time-Sharing System", Communications of the ACM, Vol. 17, No. 7, July 1974, pp. 356-375.

Capacity Analysis of Shared DASD Control Units

Floyd L. Pedriana

County of Los Angeles
Data Processing Department
Downey, CA 90242

Abstract

The IBM DASD control unit is an expensive and important component in an IBM 370 processing system. As is the case with most computer equipment, the data handling capacity is not well defined in terms of its impact on system throughput. Defining this is an especially difficult task where the control unit is connected to several CPU's. A queuing model can address this situation, however, the standard modeling equations do not account for a multipath environment. In addition, the collection of model input data is not a simple process. The data collection and model verification procedures are discussed in detail in this paper. In addition, some significant observations are made regarding the operation of the control units. This study resulted in a recommendation of limiting the control unit load to less than 50 SIO's per second.

1. Findings

The control unit performance data alculated using the queuing model described n this paper compared very well with ctual measurement data. Therefore, it ppears that the queuing model used here is valid mechanism to use in the evaluation f the capacity of control units that are hared between central processors.

If a hardware monitor is not available, he percent of time that a shared control nit is busy can be determined using a evice and channel activity distribution able.

Using the method above to determine ontrol unit utilization, the service time er SIO is easily determined by noting the otal number of SIO's in a specific time nterval. This service time calculation can be verified using the average block size and data transfer rates.

The "Control Unit Busy" flag in the Unit Control Block is an indication of control unit queuing (not utilization). The SIO records stored in GTF files are a good source to use in determining the relative number of occurrences of this "Control Unit Busy" flag.

The following two factors may contribute significantly to disproportionate DASD service to the central processors sharing control units.

a. The faster CPU will experience a smaller percentage of "Control Unit Busy" indications.

b. The 3830 control unit has a built-in logic mechanism that, in certain cases, will process SIO's according to a priority determined by the arrangement of channel cable connections.

At the installation involved in this study, it was recommended to limit SIO activity to less than 50 per second for each 3830 control unit. This means that the guideline of one control unit for each string of eight DASD units used at this installation, provides much more control unit capacity than is actually needed.

2. Background

The DASD Control Unit (3830) is a fairly large and expensive piece of computer equipment and since there is considerable flexibility in the way that it may be incorporated into the system configuration, it was decided to perform a detailed analysis of the control unit operation. The objective of this study was to determine techniques and guidelines for use in the capacity analysis of IBM control unit configuration and loading.

The software monitors, GTF, RMF, and CMF were all used for parts of this study. These monitors have certain limitations in a study of this type, however, a hardware monitor was not immediately available and it was decided to proceed as best as possible without one.

The general approach to this problem was as follows. First, some general DASD performance data was collected and reviewed. Then a mathematical model was proposed and some specific data was collected and analyzed. At several points in this process, calculated values were compared to measured values in order to verify the validity of the model, the calculation, and the general understanding of the situation. Finally, all of the data was reorganized and consolidated into meaningful values which have been summarized in this report.

In the entire process, a conservative approach was used in order to provide for a margin of error in data collection and analysis. For example, all of the data used was obtained during brief periods of highest activity on all systems. These periods were mid-morning or mid-afternoon. This, therefore, assures that the analysis will account for the capacity requirements during the peak demand periods rather than for the average requirements.

The objectives of this paper are to define the control unit mechanization and to describe procedures to help evaluate the control unit capacity. This paper is not intended to be a tutorial on queuing model theory. The model used is a very basic one for which all of the equations are readily available in many books.

3. DASD Control Unit Operation and Model

The main function of the 3830 control unit is to select and transmit data and control information between the disk devices and the I/O channel at the central processor. The actual data transmission time is generally very small compared to the total I/O time. This is due to the delays encountered in the channels, the disk device, and also in the I/O Supervisor (IOS) software. The majority of the I/O delay time is usually due to mechanical device time or file contention.

Figure 1 shows the interconnection between the various channels and devices. This figure shows that any CPU can get to any device through either of two channels and through either of two control units. This provides for backup in case of hardware failure and provides for improved performance when a single channel or control unit becomes overloaded and cannot handle all of the traffic by itself.

When a control unit detects a request to transmit data by the channel (indicated when the "Select Out" bit is set by the channel interface), the control unit responds by setting either the "Operational In" tag or the "Status In" tag on the channel interface [1]. In the case of the former, I/O processing continues. In the case of the latter, the control unit sets a combination of bus bits that indicate some difficulty in completing the I/O. The combination of "Status Modifier" (bus bit 1) and "Busy" (bus bit 3) indicate that the I/O must be delayed due to the control unit being preoccupied handling another I/O request. The control unit also sets an internal flag called "Request Pending". This flag is used later by the control unit to initiate reconnection to the central processor. Meanwhile, IOS places this request on a queue and sets the "Control Unit Busy" bit in the Unit Control Block (UCB). This process has led to some misunderstanding regarding the significance of this bit. This bit is incorrectly labeled since it is not always set when the control unit is busy but rather only when an I/O request has been delayed due to a busy control unit. The meaning of this bit would be clarified if its name were

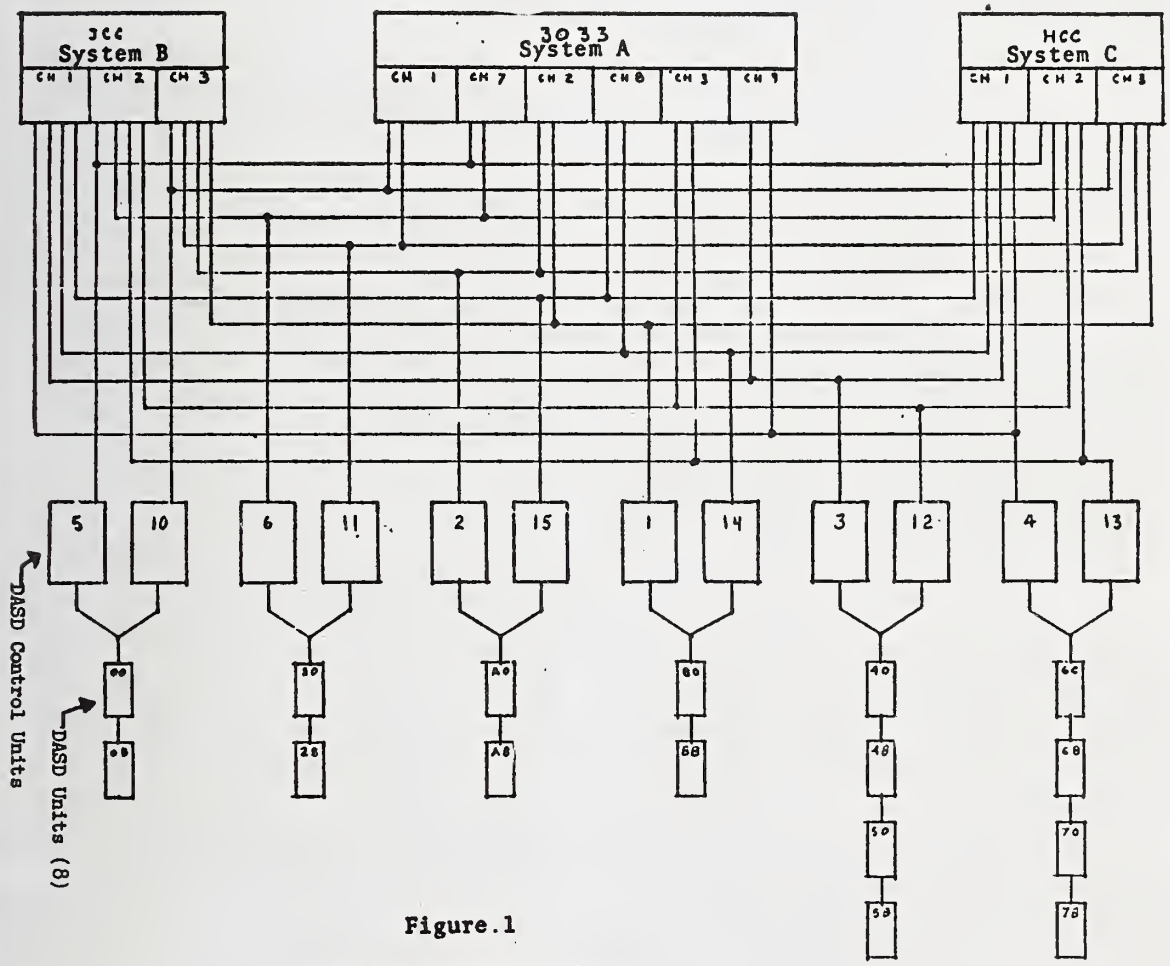


Figure.1

Logical Congiguration

changed to "Control Unit Delay". When the control unit becomes free it sends an I/O interrupt (Device End - status bit 5) to the central processor. IOS then tries to complete the original I/O which has been queued by starting the process all over.

In order to define a mathematical model of the system, it is necessary to understand the interrelations between the various components of the system as discussed above. Although there are many types of models defined in the literature, none of them exactly fit the shared DASD - multipath configuration of this problem. However, after making some simplifying assumptions and examining various models in detail, it was found that the class of models which are described in Kendall notation as M/M/c could be used quite successfully for this application. In order to fit this simple model to the actual configuration, the control units are considered to be servers that are independent of the rest of the system configuration and interaction.

It is beyound the scope of this report to explain this class of models and the interested reader is directed to reference [2]. In selecting this model it is assumed that the interarrival times of I/O requests and control unit service times are exponentially distributed. This is not an unusual assumption and is generally a good approximation even if the data is not exponentially distributed. The key items that must be defined in order to use this model are the expected control unit service time for each I/O request and the expected arrival rates for each I/O request. In this discussion an I/O request is defined as a Start I/O machine instruction (SIO - op code of 9C).

4. Control Unit Service Time

The expected service time for an I/O operation was determined by two different methods in order to have a high degree of confidence in the calculations. The first method was based on actual measurements, while the second is based on calculations using engineering hardware specifications.

Determining service time from performance measurements is a fairly straight forward process: The amount of time spent servicing all SIO's is simply divided by the number of SIO's serviced. The time spent servicing all SIO's is the percent of time that the control unit is busy, multiplied by the total elapsed time of the data collection period. Control unit busy time can be determined from channel busy time and the distribution of device activity. Since each control unit is connected to three systems, the data had to be collected on all three of these systems simultaneously. The control unit time is the same as channel time except for time spent in error recovery and in format write erase. Therefore, the percent of time that the control units are busy is determined by the total channel and device time as measured on all three systems.

The difference in speed between the 3350 and the 3330-11 must also be taken into account. Figure 2 is a matrix showing the distribution of channel and control unit time. Figure 2 also has a comparison of channel time as calculated using the above process, and as measured using the software monitor. Since these values compare very well, it is safe to assume that this method of determining the control unit busy time is valid.

At the same time that this control unit data was collected, the quantity of SIO's being processed was also collected. This data was averaged over several hours and used to find the average control unit service time for each I/O. This value was 8.5 ms. Using the data transfer rate of 1,198,000 bytes per second and an average block size of 7000 bytes, the service time is found to be 5.8 ms. These values are in the 5 to 10 ms range and, therefore, establish a high level of confidence in the approximate value for control unit service time. The more conservative 8.5 ms was used in this analysis.

5. Model Validation

The M/M/c model itself was validated by comparing three separate calculations with measured values. These three values are: (1) the probability of an I/O request encountering a busy control unit, (2) the amount of time that an I/O request can expect to wait due to a busy control unit, and (3) the level of control unit utilization. The symbology used in these calculations is explained in detail in reference 2 and is summarized in Figure 3. The items above are determined by the following relationships.

$$C(c,u) = \left(\frac{u^c}{c!}\right) \times \left[\frac{u^c}{c!} + (1-\rho) \sum_{n=0}^{c-1} \left(\frac{u^n}{n!}\right)\right]^{-1}$$

$$Wq = \frac{C(c,u)\ E(s)}{c(1-\rho)}$$

$$\rho = \lambda E(s)/c$$

$$\text{Where } u = \lambda E(s)$$

	Control Units / Channels	5	10	6	11	2	15	1	14	9	12	4	13	TOTAL (Columns)	TOTAL Measured
158 #1 Jcc SYSTEM B	1						.02		2.38	.08		12.70		15.18	15.4
	2	1.12		.47							.08			1.67	1.4
	3		1.12		7.47	.02								8.61	8.2
SYSTEM A 3033	1		11.74		6.2									17.94	16.7
	7	11.74		6.2										17.94	17.1
	2					14.85		7.21						22.06	21.0
	8						14.85		7.21					22.06	24.6
	3										10.85		12.77	23.62	23.2
	9									10.85		12.77		23.62	21.2
158 #2 Hcc SYSTEM C	1						1.34		8.22	6.13		.01		9.70	9.3
	2	.03		.01							6.13		.01	6.18	6.3
	3		.03		.01			2.22						3.60	3.9
						1.34									
	TOTAL	12.39	12.89	6.68	13.68	16.21	16.21	9.43	11.81	17.06	17.06	25.48	12.78	172.18	172.3

Figure 2

Channel - Control Unit
Activity Matrix

Figure 4 is a tabulation of the values calculated using the preceding equations and the same values as recorded by GTF and RMF. These validation calculations were made using the raw RMF data (see figure 5) accumulated for control units 2 and 15. These control units which control 3350 drives, had the highest traffic rate (51.02 SIO's per second) during the period of measurement. For general modeling purposes the degree of agreement between the calculated and measured values is considered very high, and indicates that the model can be used with a relatively high level of confidence.

N	Number of customers in steady state system.
C(c,u)	P(N≥c) ; probability all c servers are busy; Erlang's C formula
u	Traffic intensity measured in Erlang's; Minimum vlaue for steady state system.
ρ	Probability that any particular server is busy; server utilization.
Wq	Expected (mean) time in the queue, excluding service time for steady state system.
E(s)	Expected (mean) service time for one customer.
λ	Mean arrival rate of customers into queuing systems.
c	Number of Servers.

Figure 3. Symbology Summary

	M/M/c Model Calculations	RMF or GTF Measurements
Wq	.423 ms	.515 ms
C(c,u)	.077	.066
ρ	.217	.2 (appr)

Figure 4. Model Validation Data

Control Unit Number	Central Processor	SIO Count	% of Time "CU busy" bit is best
5, 10	A	197041	.06
	B	20562	.12
	C	3007	0
	Total	220610	.18
6, 11	A	101775	.03
	B	105216	.03
	C	87	0
	Total	207078	.06
2, 15	A	335775	.38
	B	396	.03
	C	31211	2.22
	Total	367362	2.63
1, 14	A	90597	.11
	B	69238	.17
	C	71206	.11
	Total	231041	.39
3, 12	A	103400	1.04
	B	169752	.14
	C	1150	.57
	Total	274302	1.75
4, 13	A	121996	.21
	B	2003	.43
	C	172050	.06
	Total	296049	.70

Note: RMF data was collected between 14:00 and 16:00.

Figure 5. RMF Data Summary

6. Control Unit Capacity

There are two distinct advantages to developing a system model in an analysis of this type. First of all, the process of putting together the pieces of the model requires the analyst to become familiar with the detail operation and loading of the subsystem in question, thereby, making an intuitive appraisal more realistic. Secondly, the model itself can be used to answer questions regarding the subsystem which cannot be answered directly from measured values. The parameters of the model can be modified to reflect changes in the system, the workload, or the desired response time and the model will then yield specific quantitative values for evaluation.

One area of concern regarding the control unit capacity is the significance of the percentage of I/O requests that encounter a control unit busy condition. Figure 6

shows that this percentage is 7.49 for the 3350 control units and 9.46 for the 3330 control units. Although these percentages are not particularly high, the ultimate concern of course is how these values relate to user service levels.

Control Unit Number	SIO Count	Control Unit Busy Count	Control Unit Busy Percent of SIO's	Average Busy Percentage
5	6,212	350	5.63	
10	6,464	417	6.45	
6	10,278	646	6.28	
11	11,188	685	6.12	
2	10,180	681	6.69	
15	9,501	624	6.57	
1	13,549	1,524	11.25	
14	12,256	1,348	11.00	7.49 (3350)
3	28,777	3,486	12.11	
12	29,031	3,166	10.91	
4	16,761	1,223	7.30	
13	15,599	1,172	7.51	9.46 (3330)
Total	169,796	15,322	9.02	

Data collected during peak period duration of 20 minutes.

Figure 6. GTF Control Unit Data Summary

The M/M/c control unit model was used to produce Figure 7 which shows the relationship between control unit loading and the increase in service time to the user. Figure 7 shows that this relationship is non-linear and that it starts to increase rapidly as the number of SIO's per second increase beyound 100 per pair of control units. However, even at a loading level of 100 SIO's per second the delay time is in the millisecond range (1 millisecond = .001 second). If a very conservative estimate of 2 ms is made regarding the amount of time during peak loading periods that each SIO can spend at the control unit without affecting the user, then it follows that the control units have the capacity to safely process about 100 SIO's per second (50 SIO's/sec. for each unit in a pair of 3830's).

7. Observations

In reviewing the GTF data, an interesting phenomenon was observed. Figure 8 shows that there is a definite relationship between the number of SIO's that encounter a busy control unit and the central processor that issued the SIO. The data clearly indicates that SIO's originating on System A (TSO and batch) have a very high probability of being executed immediately. While those originating on System C (Health Care Systems) have a much higher probability

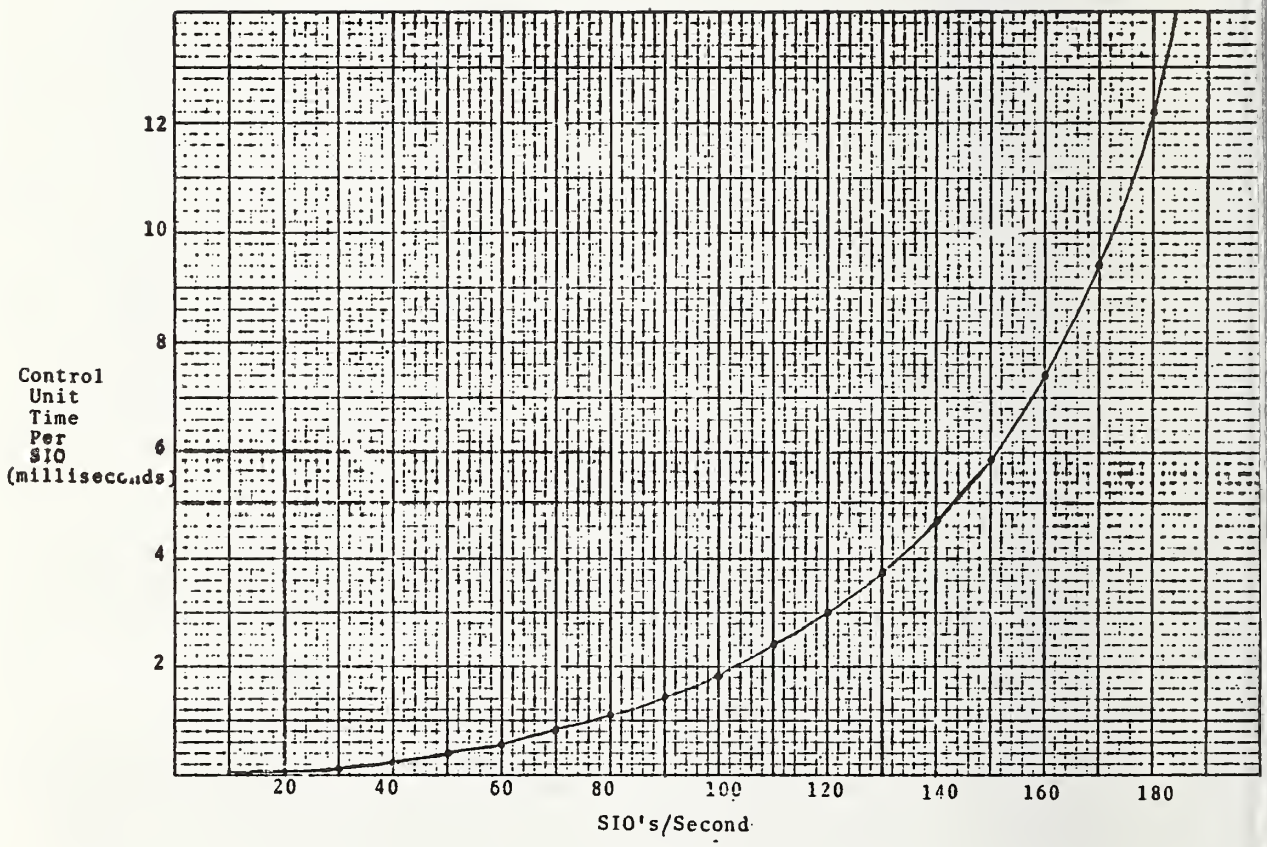


FIGURE 7

Control Unit Load Effects on User Service

of being delayed due to a busy control unit. There is a definite priority mechanism in effect here but it is not clear what the nature of this mechanism is.

DASD Model	Central Processor	SIO Count	Control Unit Busy Count	Control Unit Busy Percent of SIO
3350	A	27583	489	1.77
	B	29080	2737	9.41
	C	22965	3050	13.28
	Total	79628	6276	7.88
3330	A	23378	1501	6.42
	B	30055	2222	7.39
	C	36735	5324	14.49
	Total	90168	9047	10.03

Figure 8. GTF Central Processor Data Summary

Part of the reason for this shift is accounted for by the speed of the 3033 processor. If two systems are notified at about the same time that an I/O path has become available, the faster of the two will be the one that gains control first while the slower one gets rejected when it finally attempts to use that path. This same effect can occur if CPU's of equal speed are notified in sequence rather than simultaneously. The first one notified will probably be the one to gain control. Another explanation for this effect was found in the electronic circuit diagrams for the 3830 control unit (see reference 3). The 3830 circuits are designed to give priority to the input ports at 3830 connectors in a certain fashion. The input ports are designated as "A", "B", "C", and "D" on the hardware itself and in the hardware documentation. A spot check of the system cables indicated that system A was connected to input port "A" on all control units, and system B and C are similarly connected to ports "B" and "C" on all control units. The 3830 circuit diagram (Figure 9) shows that the hardware priority for the input ports is in the order A B C D, with A being the highest priority. Although this arrangement of cables may simplify the documentation of the control unit cabling layout, it appeared to be causing a performance bias in favor of the system which processes the batch workload. Therefore, it was necessary to recable the control units to properly balance the selecting priority defined in the hardware.

References

[1] IBM System/360 and System/370 I/O Interface Channel to Control Unit Original Equipment Manufacturer's Information; GA22-6974-1.

[2] Arnold O. Allen, Probability, Statistice, and Queuing Theory with Computer Science Applications. Academic Press, New York, 1978.

[3] IBM Control Unit Model 3830, Automated Logic Diagrams (ALD) Volume 1, page CS101.

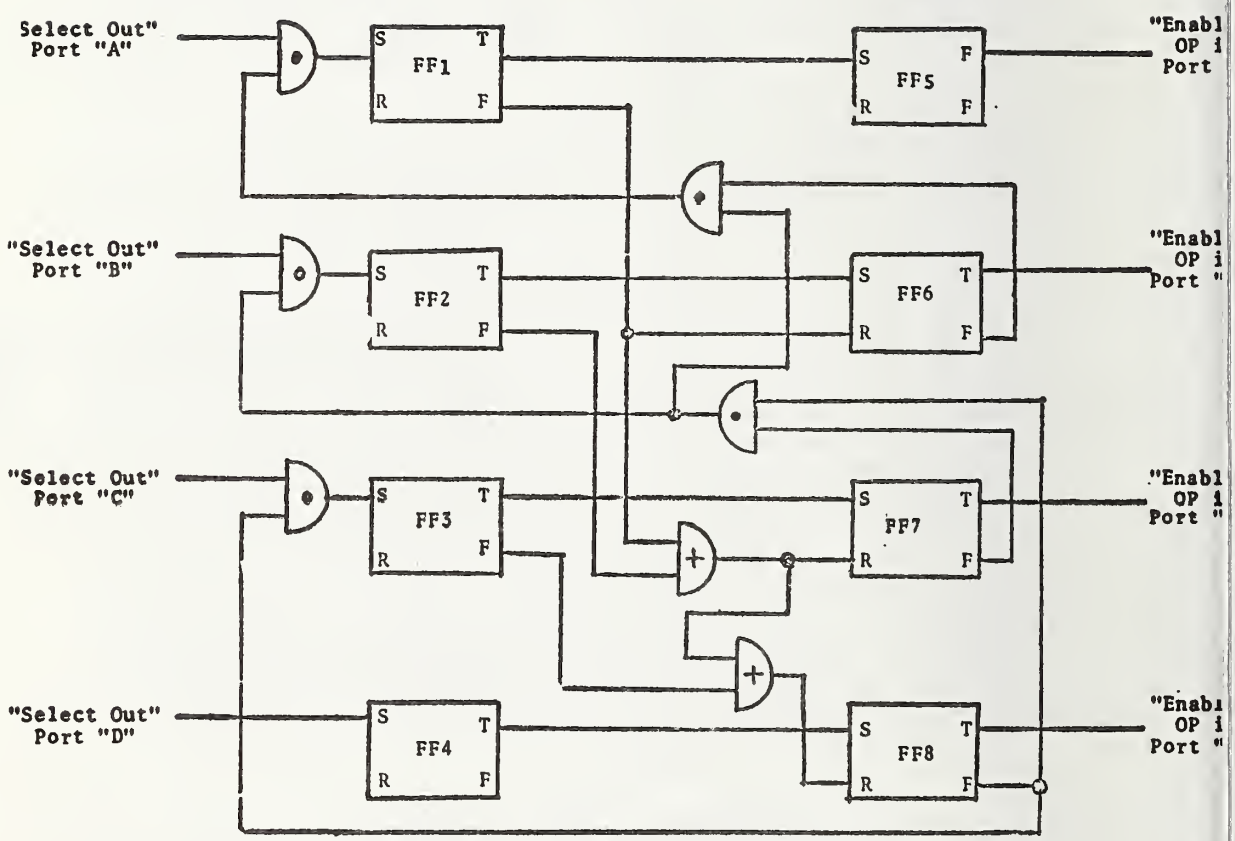


Figure 9

3830 Priority Circuit

A Note on Computer System Capacity Planning Through Material Requirements Planning

K. O. Salawu

Bell Laboratories
Piscataway, New Jersey 08854

A computer installation is likened to an industrial production factory. Demands for goods and services are either not met (shortage, disservice) or oversupplied (inventory or waste). For given shortage and wastage costs, a smoothened, short-term production/service plan can be drawn from predicted demand or workload. This brief note employs simple graphical methods used by production engineers for aggregate production scheduling and the arithmetic of materials requirement planning to impute measures of disservice to users and poor utilization of computer resources. Noting that production/service objectives are hardly unique, the methods of goal programming that incorporate the satisfaction of multiple objectives are judged to be more appropriate for formulating production/service plans.

1. Introduction : Computer Capacity Planning Objectives

In manufacturing systems, sales forecasts guide maximum investment and minimum profit objectives as well as decisions on the total work-force levels, the production rate, net inventory levels and the stability of these rates and levels. Here, our working concept of computer capacity planning is the efficient allocation of data processing resources to satisfactorily execute the offered workload hence we note that the characterization of an installation's workload directly influences the data processing objectives of the installation's management.

As the speed and flexibility per unit cost of computer hardware decreases rapidly, and the unit cost of producing and maintaining software rises, we observe that a unit of the user's time is now more expensive than that of the computer, the tool used. It seems rational-economic then that the provision of high overall levels of service to the user, which conserves his time and effort, should now assume a much higher priority than cost and performance (utilization) considerations. In fact, the mechanistic performance/cost criterion which emphasized the evaluation of the

performance of the computer subsystem, is giving way to a systemic or holistic view of the performance of the computer and its environment [Ferrari, 1978].

Notwithstanding the fact that workload characterization remains one of the most formidable problems in performance evaluation for capacity planning, it is now widely accepted that the most important input for computer capacity planning is a clear definition of user service objectives, and to a smaller extent, estimates of the availability and utilization of the total computer system. Furthermore, these overall objectives vary in their degrees of importance, depending on the utility preferences of the installation's management [Grochow, 1972].

2. Short-term Capacity Planning

Figure 2-1 is a histogram of the average CPU consumption by all applications at a computer installation [from Bronner, 1979]. A similar histogram may, for instance, have been constructed for the average daily printer utilization by the various batch job categories at an installation. Furthermore, the histogram in Figure 2-1 could be differentiated into its component histograms, i.e. one histogram per application. These latter histograms will show the extent and time of service received from the CPU resource by each application.

Suppose our window is between 9 a.m. and 5 p.m. every day, and some representative application usually consumed hourly rates of CPU service as shown in line 2 of Table 2-1. Figure 2-2 shows the histogram of CPU utilization by this application. Employing simple graphical methods used by production engineers for planning for uniform production rates, we first plot the cumulative service requirements as curve OPQR in Figure 2-3. The slope of the straight line OR would give the average requirements as well as the service rate per hour that this application would have.

Figure 2-1 : Histogram of CPU Consumption by Applications at an Installation

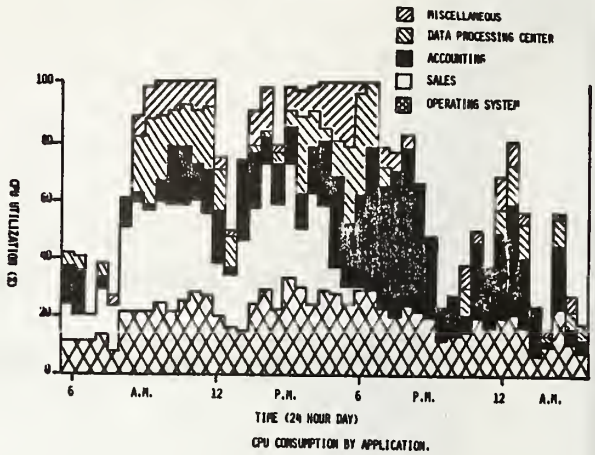


Table 2-1: % CPU Consumption by a Representative Application During Window

Time of Day	9-10	-11	-12	-1	-2	-3	-4	-5 p.m.
% CPU Consumed	19	9	23	57	72	90	38	12
% CPU Dedicated	40	40	40	40	40	40	40	40
Surplus (inventory) or Shortage with no Setup	21	52	69	52	20	-30	-28	0

As a feasible production plan must cover the demand in every period, the broken line ST will be drawn parallel to OR to pass through the point Q on the curve which is perpendicularly farthest from OR [Buffa and Miller, 1979]. However, the intercept OS that this line makes on the utilization axis is equivalent to "inventory" carried into the window at 9 a.m.. In practice, this would mean that CPU consumption was anticipated, provided and consumed before the demand for such service was made. In the case of batch processing, this scenario might be workable through workload shifting operations and for transaction processing, it may be necessary to stagger the work hours of some terminal operators themselves.

In either event, one may assess these rescheduling operations as "setup costs" at the beginning of every window. However, no buffer inventories of % CPU time will be offered hour by hour in addition to the constant average hourly service "dedicated" to each application. While an attempt to smoothen service is good, penalties must be borne for our inability to shift the workload backward (inventory) or forward (backlogging), at will, to fit the constant service rate.

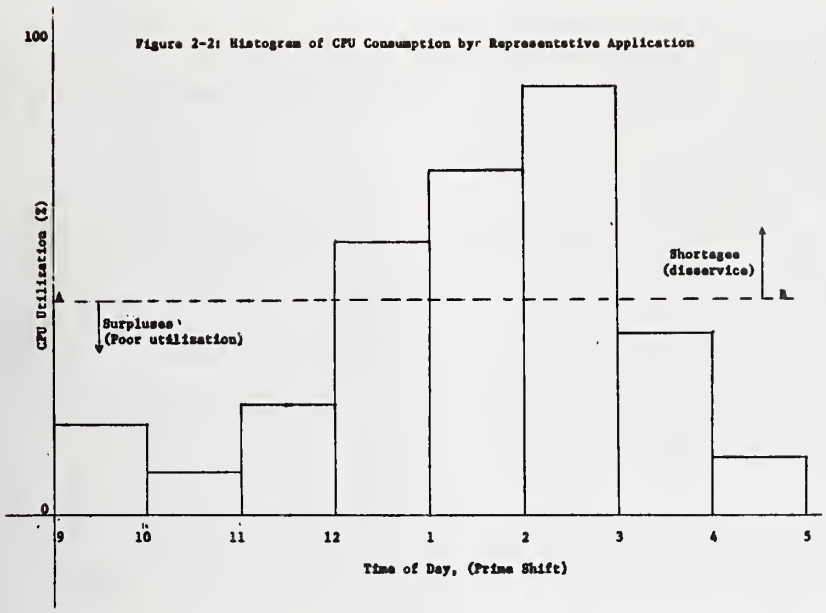


Figure 2-2: Histogram of CPU Consumption by Representative Application

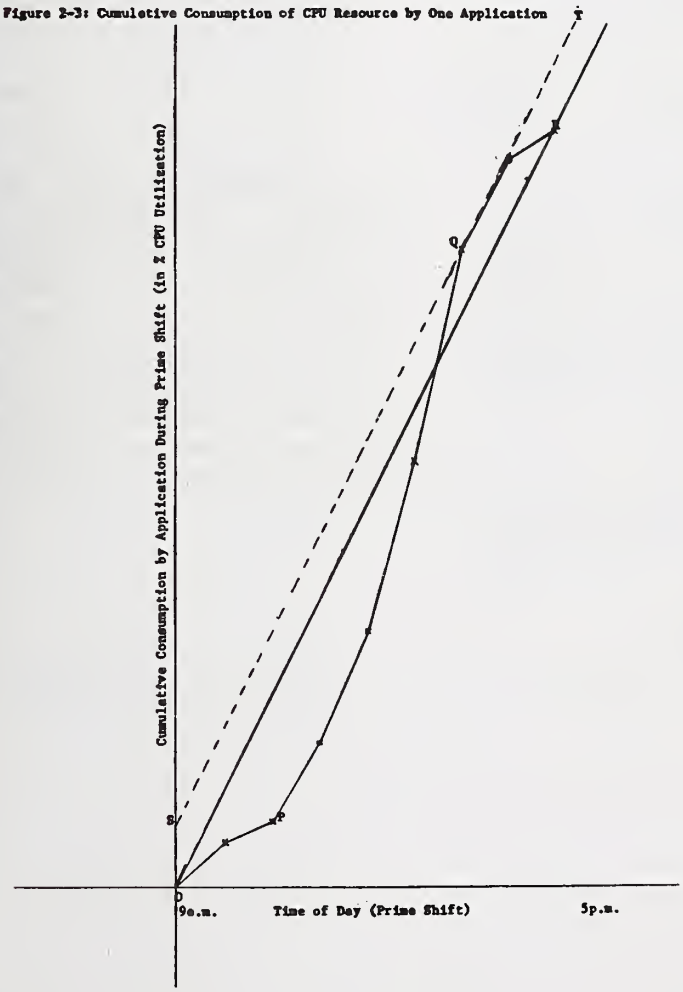


Figure 2-3: Cumulative Consumption of CPU Resource by One Application

The last line of Table 2-1 gives the surplus or shortage of service for each hour in the window without using any beginning inventory at 9 a.m.. The entries on this line are simply the differences between the cumulative % CPU dedicated and the cumulative % CPU consumed by the application by the end of each hour. The provision of any amount of beginning inventory will increase every entry on this line by exactly that amount. In our example, it would be wasteful to start with an inventory of more than 30% CPU, the most negative inventory balance for this application. This value of 30 is the same amount indicated in Figure 2.3 as the intercept OS. We propose that shortages be related to measures of disservice while surpluses should be related to poor utilization of resources.

3. Aggregate Computer Capacity Planning Using Goal Programming

Suppose x_{it} is the amount of resource consumption requested by the application with priority i, (i = 1,2, ...,k) in period (hour) t, during a window of T hours, $t=T$. Let

$$\bar{x}_i = \frac{\sum_{t=1} x_{it}}{T},$$

be the average hourly % CPU dedicated to the application. Using requirements planning formulation, let S_i be the amount of % CPU used as setup or beginning inventory for this application and let net inventory in hour z be represented by

$$I_{iz} = S_i + z\,\bar{x}_i - \sum_{t=1}^{t=z} x_{it} \quad \text{for } z=1, 2, \ldots,T.$$

The average inventory per window will then be given by

$$\bar{I}_i = \frac{\sum_{z=1}^{z=T} I_{iz}}{T}$$

while the shortage in hour z will be $I^-_{iz} = -I_{iz}$ iff $I_{iz} < 0$ and 0 otherwise. The total deficiency in the supply of the resource to the application would be

$$I^-_i = \sum_{z=1}^{z=T} I^-_{iz}$$

Let c^-_i be the unit cost of shortage or disservice for the application with priority i; c^+_i the unit cost of holding unused resource in inventory per unit per window and c_i the cost of setting up beginning inventory of 1% CPU. These unit costs may vary with the time of day but we shall not introduce these complexities here. We note however that $c^-_i \geq c^-_j$ if priority i is higher than j. This means that we assume that the value of service to each class of users, as evaluated by the installation management, is consistent with the priority given to that class of users. The resulting multidimensional objective would then be to

$$\text{Minimize } Z_i = c_i S_i + c^-_i I^-_i + c^+_i I_i$$

for each of the $i=1, 2, \ldots,k$ applications.

subject to

$$\sum_{i=1}^{i=k} x_{it} \leq 100 \quad \text{for } t=1,2,\ldots,T$$

$$\text{and } x_{it} \geq 0$$

Using this approach, the solution procedure successively seeks the achievement of the objectives in order of priority. In effect, higher priority goals may very well be achieved at the expense of the lower priority objectives, [Lee and Moore, 1974]. For instance, when a resource is known to be totally consumed, i.e. 100% utilization, one expects that service requirements of higher priority users are met possibly to the exclusion of those of lower priority users.

4. Suggested Extensions

A different pair of l_i and l_i^- is associated with each S_i. In our illustrative application, for instance, one may examine the following four $\{S_i, l_i, l_i^-\}$ triplets or bundles: $\{0,19.5,58\}$; $\{10,29.5,38\}$; $\{20,39.5,18\}$ and $\{30,49.5,0\}$. An installation management's preference or indifference among the bundles will be used to order the operationally feasible combinations and this ranking may then be used to impute the numerical relationships between c_i, c_i^- and c_i^+.

So far, objectives have been formulated and priorities assigned along job class lines. A more useful framework may be one where management's multiple goals are specified in terms of turnaround/response time, availability/utilization of (sub)systems, workload balancing and queueing problems in the system. For example, a top-priority, one-sided goal that is not open-ended may be that the turnaround time for a given class of batch jobs be no more than a numerically specified interval of time.

Most companies now view distributed systems to be more efficient in providing better service to the users than are centralized systems [Svobovoda, 1978] because of the former's virtually superior availability, reliability, flexibility and security. One of the most troublesome problems now introduced by the addition of telecommunication (switching and transmission) interfaces to the existing hardware/software interface in centralized systems is how to distribute shared resources such as databases both physically (hardware topology) and logically (software protocol) in a way that minimizes congestion and deadlock in the system.

The goal programming approach suggested here, being a multidimensional form of linear programming, will be well suited to the allocation of scarce resources in these computer networks. Furthermore, other forms of goal programming which permit one to make progress toward all objectives simultaneously, [Bradley, Arnoldo and Magnanti, 1977], may also be used.

REFERENCES

[1] Bradley, S.P., Arnoldo, C.H. and Magnanti, T.L., Applied Mathematical Programming Reading, Mass.: 1977.

[2] Bronner, LeeRoy, Capacity Planning Implementation, IBM Technical Bulletin GG22-9015-00, January 1979.

[3] Buffa, E.L. and Miller, J.G., Production-Inventory Systems: Planning and Control, Homewood, Illinois: Richard D. Irwin, 1979.

[4] Ferrari, D. Computer Systems Performance Evaluation, Englewood Cliffs, New Jersey: Prentice-Hall Inc., 1978.

[5] Grochow, J.M., "A Utility Theoretic Approach to Evaluation of a Time-Sharing System," in Freiberger, W. (ed.) Statistical Computer Performance Evaluation, New York: Academic Press, 1972 pp. 25-50.

[6] Lee, S.M. and Moore, L.J., "A Practical Approach to Production Scheduling," Production and Inventory Management, 1st Quarter, 1974, pp. 79-92.

[7] Svobodova, Liba, "Performance Problems in Distributed Systems," Proceedings of CIPS Session '78, Edmonton, Alberta, Canada; May 23-25 1978.

CPEUG80

Statistical Methods

Adaptive Load Control in Batch-Interactive Computer Systems

Samuel T. Chanson and Prem S. Sinha

Department of Computer Science
University of British Columbia
Vancouver, B.C. V6T 1W5
Canada

This paper presents a systematic approach to estimate the saturation point of a large computer installation using operational analysis. An expression for saturation point as defined by Kleinrock is derived in terms of measurable and operational quantities. Using stochastic programming and time series analysis the optimal number of batch jobs that should be activated within next interval is then computed, so that the system is neither underutilized nor over-saturated.

Keywords: Multiprogramming; response time; throughput rate; saturation point; load control; queuing theory; operational analysis; optimziation.

1. Introduction

Most large scale computer systems employ some form of load control to maintain a high throughput rate and/or to provide an acceptable level of service to the users. For paging systems this is often accomplished by manipulating the degree of multiprogramming or equivalently the size of the resident sets of the active processes to keep the system from becoming saturated. Previous work directed towards this end is primarily represented by the development of the working set policy [1,2,3,4] where working set is defined as minimum number of pages needed to run a program efficiently. More recently, efforts to optimize the system work capacity lie mainly in keeping some measures related to program behavior (usually paging behavior) within some predetermined bounds [5,6,7,8]. The 50% criterion [7] for example, aims at maintaining the utilization of the paging device to around 0.5. The L=S criterion [6] proposes to keep the system life time to approximately that of the page swap time. The knee criterion [5,8] suggests that the mean resident set size of each process should be maintained at the value associated with the primary knee of its life time function, where life time is defined to be expected time between two successive page faults [5]. Though the most robust of the three, the knee criterion, is also the most costly to implement and involves the largest amount of overhead.

Though these criteria are not based on mathematical models and cannot be proved to be optimal, they aim at increasing the throughput rate by loading the system up to the point when the measured indicator suggests further increase in system load may cause 'thrashing'. The methods cannot be applied to non-paged systems. Furthermore, for interactive systems and combined batch-interactive systems, one is interested not only to maximize the system throughput rate but also to guarantee good response times to the interactive jobs (possibly at the expense of the batch jobs). Landwehr [9] studied a combined batch-interactive system and

proposed a scheme to activate batch jobs based on the terminal load. The emphasis of the study, however, was on model formulation and validation. There was no attempt to prevent the system from saturation or to optimize performance. As well, there is no easy or systematic way of determining the values of the break points. Hine et. al. [10] studied the problem from a slightly different viewpoint. Their goal was to control the main memory allocation for each class of jobs to provide different response times to each while maximizing the CPU utilization. They employed a mathematical model but optimization was achieved by an exhaustive search technique. A heuristic was also given which provides good but not optimal results.

In this paper we study the performance of a combined batch-interactive computer system using the operational analysis technique proposed by Denning and Buzen [14]. The control algorithm determines from time to time the number of batch jobs (if any) to be activated from the batch queue. The control criterion aims at keeping the system from saturation (to be defined in the next section) while minimizing the mean number of jobs waiting to be activated. The effect is to maximize the throughput of the system while maintaining good response time for interactive jobs.

2. Estimation of System Saturation

Definitions of system saturation have been proposed [12,13,14]. Invariably the system is considered saturated at the point the response time vs system load curve starts to rise rapidly. Kleinrock [12], for example, using the number of active terminals as the load, defined the system's saturation point to correspond to the intersection of the mean normalized response time curve asymptote and the horizontal line corresponding to the minimum response time (i.e., when there is only one active terminal). (See Figure 1). If a system is not allowed to get saturated according to this definition, the mean response time of the active jobs will not exceed an acceptable level. However, the implicit assumption is that the program population considered is both homogeneous and stationary. Our approach is to compute the system saturation load at small intervals (such as a few seconds) during which the stationary assumption is justified. The homogeneous assumption is discussed below.

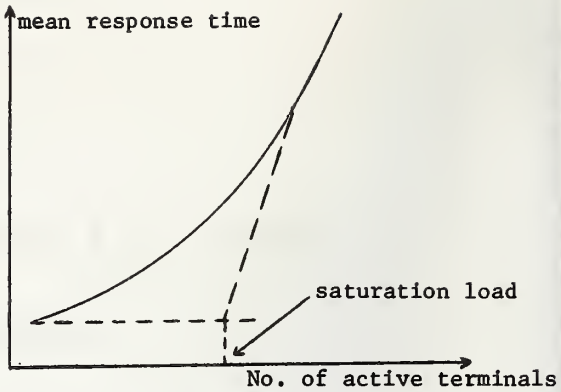


Figure 1. Mean Response Time vs the Number of Active Terminals

The computer system is often modelled as a central-server model [11,13]. Consider such a model with M service centres and a degree of multiprogramming equal to N. Each service centre consists of a device and its associated queues (Figure 2).

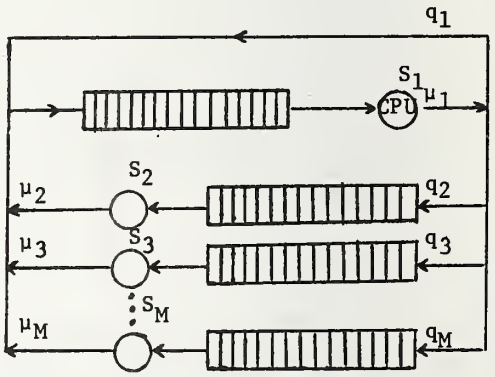


Figure 2. Central-server Model with M Service Centres

The service centre S_1 is the CPU service centre (the central server). On completion of the CPU service a job either leaves the system or joins another service centre. A job leaving service centre S_i, i=2,3,...,M must join the central server.

2.1 Notation

The quantities defined here and computed in section 2.2 are mean values within an observation period and as such are functions of time which is omitted for clarity.

T : observation period

X_i : observed number of completions at centre S_i during T

B_i : the total amount of time during which the service centre S_i is busy during T

C_i : observed number of requests for centre S_i during T

q_i : request frequency, the fraction of jobs proceeding next to service centre S_i on completing a service request at the central server $= C_i/X_1$, $i \neq 1$

2.2 Operational Quantities

We now compute the operational quantities in order to obtain an expression for saturation point.

Mean service rate of server $S_i = \mu_i = X_i/B_i$

Utilization of server $S_i = \rho_i = B_i/T$

System throughput rate $\bar{T} = (X_1 \cdot q_1)/T$

$$= \frac{X_1}{B_1} \cdot \frac{B_1}{T} \cdot q_1$$

$$= \mu_1 \rho_1 q_1 \qquad (1)$$

Utilisation $\rho_i = B_i/T$

$$= \frac{X_1}{X_i} \cdot \frac{B_i}{T} \cdot \frac{X_i}{X_1}$$

$$= \frac{B_1}{T} \cdot \frac{X_1}{B_1} \cdot \frac{B_i}{X_i} \cdot \frac{X_i}{X_1}$$

Under the job flow balance assumption $X_i = C_i \quad i \neq 1$

$$\therefore \rho_i = \rho_1 \cdot \frac{\mu_1}{\mu_i} \cdot q_i, \quad i \neq 1 \qquad (2)$$

$$\therefore \sum_{i=1}^{M} \rho_i = \rho_1 + \sum_{i=2}^{M} \rho_1 \cdot \frac{\mu_1}{\mu_i} \cdot q_i$$

When there is only one job in the system

$$\sum_{i=1}^{M} \rho_i = 1 .$$

$$\Rightarrow \rho_1(1) = \left[\sum_{i=2}^{M} \frac{\mu_1}{\mu_i} \cdot q_i + 1 \right]^{-1} \qquad (3)$$

We use Little's law to compute the mean response time

$$\bar{R}(N) = N/\bar{T} .$$

$$\text{by (1)} \quad = \frac{N}{\mu_1 q_1 \rho_1} = \frac{Nq_i}{\mu_i \rho_i q_1} \quad i \neq 1 \qquad (4)$$

$$\text{by (3)} \quad \bar{R}(1) = \frac{1}{\mu_1 q_1} \left[\sum_{i=2}^{M} \frac{\mu_1}{\mu_i} \cdot q_i + 1 \right] \qquad (5)$$

The equation of the asymptote (as N approaches infinity) is more difficult to derive. Let us first consider the simple case of a non-virtual memory system. The asymptote occurs at the point when the utilization of a service centre (i^* say) reaches unity (i.e., it becomes the system's first bottleneck).

From Buzen's analysis [11], i^* is that service centre which has the highest utilization in an interval (i.e., i^* may vary from interval to interval as the work load characteristics change). If it is the CPU the equation of the asymptote is simply

$$\bar{R}(N) = \frac{N}{\mu_1 q_1} . \qquad (6)$$

Otherwise, using equation (4) and noting that μ_i as well as the ratio (q_i/q_1) remains unchanged as N increases, the equation of the asymptote is

$$\bar{R}(N) = \frac{Nq_{i^*}}{\mu_{i^*} q_1}, \quad i^* \neq 1 . \qquad (7)$$

For a paging system, the eventual bottleneck as N approaches infinity must be the paging device but it need not be the first device to saturate.

Case (i), the paging device is not the first to saturate.

In this case, as the system is saturated before the paging device is fully utilized, the asymptote should be computed based on the first device to reach saturation and equation (7) is still valid (see Figure 3).

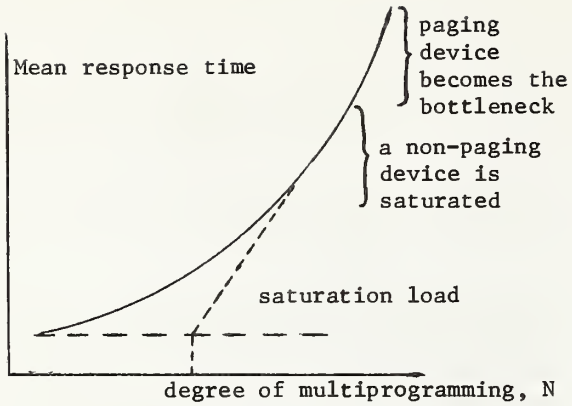


Figure 3. Mean response time vs N, a non-paging device is the first to be saturated

Case (ii), the paging device is the first to saturate.

The ratio q_{i*}/q_1 continues to increase as N increases and approaches infinity as N approaches infinity. A realistic approach consistent with the one used in Case (i) is to use the value of q_{i*}/q_1 corresponding to the point the paging device first becomes fully utilized. However, this ratio is not easy to estimate. The observed value of q_{i*}/q_1 can be used if the system is close to saturation (i.e., $N^* \doteq N$, see below) when the parameters are measured. Otherwise the saturation load will be under-estimated. This is not a problem when the work load is light. As can be seen subsequently, if the system work load then gets heavy, the control policy will adjust to it and the observed ratio will again approach the desired value. The saturation load N* satisfies equations (5) and (6)

$$N^* = 1 + \sum_{i=2}^{M} \frac{\mu_1}{\mu_i} q_i \quad \text{if the CPU is the bottleneck} \qquad (8)$$

or it satisfies equations (5) and (7).

$$\text{i.e., } N^* = \frac{\mu_{i*}}{\mu_1 q_{i*}} \left[\sum_{i=2}^{M} \frac{\mu_1}{\mu_i} \cdot q_i + 1 \right] \qquad (9)$$

All of the above equations can also be derived using queuing theory.

Most proposed schemes assume a fixed saturation load. The Michigan Terminal System [15] for example computes the values of five load factors at fixed intervals and if one or more exceeds the corresponding predetermined static saturation value, the system is assumed to be saturated. For the 50% criterion, the saturation point corresponds to when the utilization of the paging device exceeds 0.5+c, where c is some small positive constant. The L=S criterion to a certain extent assumes the system to be saturated when the system life time is below the page swap time, which is fixed for a given paging device.

In a previous report [15], we have shown that the saturation load is really a function of the characteristics of the current work load and cannot be very well represented by some constant measures. For the present model, these work load characteristics are q_i and μ_i, i=1,2,...,M. Any model which does not take this into consideration will sometimes over-estimate and sometimes under-estimate the system saturation load. The fact that the over-estimation on the average is equal to the under-estimation provides no comfort when the goal is to optimize performance at all times.

3. Load Control

The first criterion for load control is to keep the system from saturation. From Figure 1, it is seen that the mean response time increases rapidly beyond this point. Furthermore Denning [16] has shown that 'thrashing' (and thus reduced throughput rate) occurs when the paging device is saturated. For multiprogrammed paging computer systems, the simplest way to accomplish this is to keep the number of active jobs below N* given in equation (8) or (9). Since the system throughput rate is a non-decreasing function of N before the system saturates [8], activating N* jobs whenever possible will also maximize the throughput. There are three cases to consider:

(i) the system is saturated (i.e. N>N*),

(ii) the system is under-utilized (i.e., N<<N*),

(iii) the system is close to but not yet saturated.

Case (iii) is the interesting case since if the system is under-utilized, it is unnecessary to apply any control measure but to activate each job as it arrives until the condition for case (iii) is reached. If the system load is then properly controlled, the system should attain saturation (case (i)) infrequently and only for brief periods. The control, when the system is saturated, could simply consist of not activating any more batch job until the system comes out of saturation. If the system is in the saturation state frequently and for extended

durations then it is highly probable that the hardware is inadequate to handle the normal work load and should be upgraded. Thus we shall consider only case (iii) in this paper. We note that many systems (e.g. the Michigan Terminal System [15]) do not apply any control until saturation is detected. This, in our opinion, does not constitute proper load control.

4. Optimization

We define the following variables all of which are functions of time T which is omitted for clarity:

- I : number of batch jobs waiting to be activated during T,
- J : number of terminal requests during T,
- E_t : expected number of terminal arrivals in the next interval.
- E_b : expected number of batch arrivals in the next interval,
- D : expected number of job departures (both batch and terminal) in the next interval,
- N_b : optimal number of batch jobs that should be activated,
- N_t : optimal system capacity (measured in terms of the number of jobs) to be reserved for expected incoming terminal jobs,
- S : remaining system capacity (defined as the number of additional jobs that can be accommodated without saturating the system).

S can be approximated by

$$S = N^* - N + D$$

The problem is to determine how many of the S jobs should be filled by waiting batch jobs (if any). Our objective is to maximize the mean system throughput rate without saturating the system. This is equivalent to minimizing the expected number of jobs that have to wait at each interval because admitting them would saturate the system. We shall minimize a weighted sum of the waiting batch and terminal jobs which is a more general problem.

Let the weights be C_1 and C_2 for batch and terminal jobs respectively. The optimization problem is therefore

$$\min\{(E_t - N_t)C_1 + (E_b - N_b)C_2\} \tag{10}$$

$$\left.\begin{aligned} \text{subject to} \quad & N_t + N_b \leq S \\ & N_t \leq E_t, \quad N_b \leq I \end{aligned}\right\} \tag{11}$$

The problem is equivalent to

$$\text{Max } Z = N_t \cdot C_1 + N_b \cdot C_2$$

subject to the constraints given by (11).

If $C_1 > C_2$ (i.e., terminal jobs are favoured), it is easy to see that the solution to the above optimization problem is

$$\begin{aligned} N_t &= E_t \text{ if } E_t < S \qquad N_b = S - E_t \\ N_t &= S \text{ if } E_t \geq S \qquad N_b = 0 \end{aligned} \tag{12}$$

In the above computation, it is assumed that E_t and D are available at the beginning of the interval. If $E_t < D$ then we may have $N+N_b > N^*$ for a short period at the beginning of the interval. This problem can be alleviated by spreading out the activation of the N_b batch jobs throughout the interval instead of all at once at the beginning. It remains to show how E_t and D can be computed using smoothed statistical estimates.

Let P_t be the expected prediction of the parameter for the period $[t,t+1]$. Let x_t be the observed value of the parameter at time t. P_t can be expressed as

$$\begin{aligned} P_t &= (1-\beta)(X_t + \beta X_{t-1} + \beta^2 X_{t-2} + \ldots) \\ &= (1-\beta)\sum_{i=0}^{\infty} \beta^i X_{t-i} \end{aligned}$$

where the exponential weight factor β is a constant between zero and one. Similarly,

$$P_{t-1} = (1-\beta)\sum_{i=0}^{\infty} \beta^i X_{t-i-1}$$

$$P_t = (1-\beta)X_t + \beta P_{t-1}. \tag{13}$$

Now let the error made at time (t-1) in predicting x_t be ϵ_t, then

$$\epsilon_t = X_t - P_{t-1} \tag{14}$$

substituting in equation (13),

$$\begin{aligned} P_t &= X_t - \beta\epsilon_t \\ &= P_{t-1} + (1-\beta)\epsilon_t \end{aligned} \tag{15}$$

Now if ε_t is relatively small, we do not recompute the value of β. If E_t is large, we find a new value of β which will minimize the sum of the squares of errors given by

$$\sum_{i=t}^{-\infty} \{X_i-(1-\beta) \sum_{j=0}^{\infty} \beta^j X_{i-j-1}\}^2 \qquad (16)$$

In practice, the summation in (16) does not have to involve many (k,say) terms before β^k approaches zero. β does not have to be very accurate and standard techniques exist for its efficient computation.

To summarize, the control procedure consists of the following steps:

1. During an interval T, observe D, J, N, q_i, μ_i, i=1,2,...,M.

2. Compute N* using equation (8) or (9).

3. Estimate the expected number of terminal arrivals and total departures in the next interval using equation (15).

4. Compute the number of batch jobs N_b to be activated in the next interval using equation (12).

5. Terminal jobs are immediately activated upon arrival.

5. Conclusion

A model to estimate the saturation of a computer system has been presented which is capable of adjusting to varying work load characteristics. Based on this, the number of batch jobs that should be activated to minimize a weighted sum of the number of jobs that will have to wait upon arrival without saturating the system in a combined batch-interactive environment is computed using optimization theory. The approach thus provides good mean response time to the terminal jobs and maximizes the system throughput rate under that condition.

The second level of load control - that of the selection of the type of jobs to be activated has not been discussed in this paper. Work has started in this direction. However, because of the difficulty of predicting accurately the resource demands of a job before it is executed and because of the adaptiveness of the proposed scheme which is capable of correcting itself it may be that equally good results can be achieved without it.

The use of the number of jobs to characterize the system load is of course not precise as jobs do not necessarily have the same resource demand characteristics. However, it has been shown to produce useful results [17,18]. It is made even more acceptable for this application because of the built-in adaptiveness of the proposed policy. If the average resource demand of the activated jobs in the next time interval is lighter than that of the previous interval (on which the remaining system capacity was estimated), then the updated remaining system capacity will increase and more jobs will be activated in the next interval. On the other hand, if the average resource demand is heavier, then the remaining system capacity will decrease and fewer (or no) jobs will be activated in the next interval.

References

[1] Denning, P.J., "The working set model for program behaviour", Comm. of ACM 15, 5 (May 1968), 323-333.

[2] Denning, P.J., "Virtual memory", Computing Surveys 2,3 (Sept. 1970), 153-189.

[3] Denning, P.J., "Third generation computer systems", Computing Surveys 3, 4 (December 1971), 175-216.

[4] Rodriguez-Rosel, J. and Dupuy, J.P., "The design, implementation, and evaluation of a working set dispatcher", Comm. of ACM 16,4 (April 1973), 247-253

[5] Denning, P.J., Kahn, K.C., Leroudier, J Potier, D. and Suri, R., "Optimal multi programming", Acta Informatica, 7 (1976 197-216.

[6] Denning, P.J. and Kahn, K., "An L=S criterion for optimal multiprogramming" Proc. Int'l. Symp. on Computer Performance Modeling, Measurement and Evaluation, (March 1976), 219-229.

[7] Leroudier, J., Potier, D., "Principles of optimality for multiprogramming", Proc. Int'l Symp. on Computer Performance Modeling, Measurement and Evaluation (March 1976), 211-218.

[8] Graham, G.S. and Denning, P.J., "On the relative controllability of memory policies", Proc. Int'l. Symp. on Computer Performance, Modeling, Measurement and Evaluation, (August 1977), 411-428.

[9] Landwehr, C.E., "An endogenous priority model for load control in combined batch-interactive computer systems", Proc. Int'l. Symp. on Computer Performance, Modeling, Measurement and Evaluation, (March 1976), 282-287.

[10] Hine, J.H., Mitrani, I. and Tsur S., "The control of response times in multi-class systems by memory allocation", Comm. of ACM 22,7 (July 1979), 415-424.

[11] Buzen, J.P., "Analysis of system bottlenecks using a queuing network model", Proc. ACM-SIGOPS Workshop on System Performance Evaluation (April 1971), 82-103.

[12] Kleinrock, L., "Certain analytic results for time-shared processors", Information Processing (Proc. IFIP Congress 68), (1968), 838-845.

[13] Ferrari, D., Computer Systems Performance Evaluation, Prentice Hall, 1978.

[14] Denning, P.J. and Buzen, J.P., "The operational analysis of queuing network models", Computing Surveys 10,3 (September 1978), 225-261.

[15] Chanson, S.T., "Saturation estimation in interactive computer systems", Technical Report 79-7, Dept. of Computer Science, University of British Columbia, (June 1979).

[16] Denning, P.J., "Thrashing: its cause and prevention", Proc. FIPS, 33, (FJCC, 1968), 915-922.

[17] Scherr, A.L., "An analysis of time shared computer systems", Ph.D. Thesis, Dept. of Electrical Engineering, M.I.T., Cambridge, Mass. (June 1965).

[18] Chanson, S.T. and Ferrari, D., "A deterministic analytic model of a multiprogrammed interactive system", NCC, AFIPS Conference Proc., 43(1975), 645-652.

Sensitivity Analysis and Forecasting for Large Scale IBM Computer Systems: A Methodological Approach and Case Study

Carl Steidtmann

Staff Consultant
Mountain Bell
Denver, Colorado

This paper outlines the development and use of a new method of forecasting computer capacity for large scale IBM computer systems at Mountain Bell. The model that was developed to accomplish this task uses three different statistical techniques. A representation of the computer system is developed using two stage least squares multiple regression. An autoregressive integrated moving average process is then used to individually forecast the level of use for each of the components of the system. These values are then placed into a system of simultaneous equations which are then solved to give the desired results. The data that was used in this project was collected over the course of a year by hardware monitors off of an IBM 3033 that was the global processor in a MVS/JES3 triplex in Mountain Bell's Colorado/Wyoming processing center.

Key words: Box-Jenkins; forecasting; multiple regression; sensitivity analysis; simultaneous equations.

1. Introduction

In data processing two of the most difficult forecasting problems that management must face are:

1. When will our present computer system run out of capacity?

2. If we make a change in our present computer system, what will be the impact of the change on the different components of the system?

By developing the proper answers to these questions management in data processing can provide the best possible service to the organization at the least cost. The model presented in this paper seeks to provide data processing management with the answers to the above questions. In answering these questions the model gives its user an understanding of when new hardware needs to be ordered and what the impact changes in the present computer system will have on the other components of the system. By forecasting computer utilization on a component by component basis through the use of simultaneous equations, sensitivity analysis can be conducted to ascertain the impact of a change in one system component on all of the other components of the system.

2. The Theoretical Model

In developing the theoretical dimensions of this model two design criteria were constantly kept in mind. The first criteria was that the output of the model had to provide useable forecasting information. The second criteria was that the model had to have the capability of answering 'what if' questions. While it is a relatively simple process to develop a forecasting model for computer capacity, creating a model that

gives its user the capability of performing sensitivity analysis is a bit more difficult. In trying to meet these two design criteria it was felt that the best results could be obtained through the combination of three separate forecasting techniques. The three techniques that are used in this model are multiple regression, Box-Jenkins time series analysis and simultaneous equations.

Multiple regression alone can be used as an effective forecasting tool if the data conforms to a rather strict set of assumptions. Since the data in question did not conform to several of these assumptions multiple regression was used only to quantify the interrelationships that exist between the different components of the system. A second statistical method, time series analysis, was selected to make point forecasts about the level of system component utilization. While time series analysis would by itself provide an adequate means of forecasting component utilization, it fails to take into account the dynamic nature of the process that is under study. To incorporate this perception of a dynamic process into the model, the equations that were developed in the multiple regression portion of the study were merged with the point forecasts that came from the time series analysis. This merger of the output of these two statistical techniques produced a set of equations that were solved simultaneous to produce the final forecast.

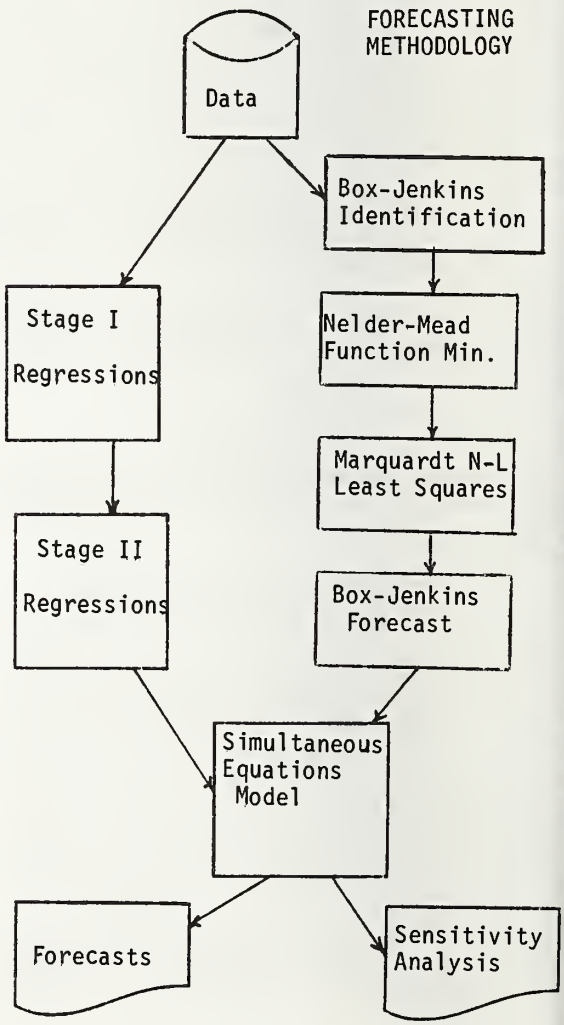
2.1 Model Development

A methodological flow chart that outlines each step involved in the development of the forecasting model is given in Figure 1. The first step that must be taken in the model design process is an evaluation of the data that is available for forecasting purposes. This data can come from System Management Facility (SMF) data, hardware monitors, or Resource Management Facility (RMF) data. If other sources of data are available these can also be used. Once this data evaluation process is completed the selection of the system components that are to be used in the model can proceed. There are no restrictions as to which system components can or should be used in the model. Selection of system components depends entirely on the interests of the model builder and the availability of the data. Some of the components that can be used include:

1) Central Processing Unit Busy
2) Information Management System (IMS) Transactions
   a) Rate
   b) Volume
   c) Response Time
3) Problem Program State Level
4) Supervisor State Level
5) Swapping Rate
6) Paging Rate
7) Channel Busy
8) Control Unit Busy
9) Device Busy
10) TSO Usage
    a) Number of Users
    b) Response Time
    c) Volume of Transactions
11) Resource Usage by Application

Once all of the data has been collected and the appropriate variables have been identified, the next step is to determine the correlations and the interrelationships that exist between the selected system components. This task is accomplished through the use of the statistical tool of two stage least squares multiple regression.

FIGURE I



2.2 Multiple Linear Regression

If there are n different system components that have been included in the analysis then each individual system component will yield an equation of the following form:

$$Y_n = B_0+B_1X_1+B_2X_2\ldots+B_{n-1}X_{n-1}+U_n \quad (1)$$

where:

- Y_n is the dependent system component
- B_0 is the intercept value
- $B_1\ldots B_n$ are the regression coefficients
- $X_1\ldots X_n$ are the observations of the independent system components
- U_n represent the error terms

Using matrix notation for each system components we get:

$$\underline{Y} = \underline{X}\underline{B} + \underline{U} \quad (2)$$

Where:

$$\underline{Y}=\begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \\ . \\ . \\ . \\ Y_n \end{bmatrix} \underline{X}=\begin{bmatrix} 1,X_{21},X_{31}\ldots X_{t1} \\ 1,X_{22},X_{32}\ldots X_{t2} \\ 1,X_{23},X_{33}\ldots X_{t3} \\ . \quad . \quad \ldots \\ . \quad . \quad \ldots \\ . \quad . \quad \ldots \\ 1,X_{2n},X_{3n}\ldots X_{tn} \end{bmatrix} \underline{B}=\begin{bmatrix} B_1 \\ B_2 \\ B_3 \\ . \\ . \\ . \\ B_t \end{bmatrix} \underline{U}=\begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ . \\ . \\ . \\ U_n \end{bmatrix}$$

Where:

- t = The number of system components under consideration
- n = The number of data observations being used in the analysis

Extending the application of multiple regression to all t components of the computer system we get the fundamental equation of the model. This can be expressed as:

$$\begin{aligned} Y_1 &= B_{01}+B_{11}X_1+B_{21}X_2\ldots B_{t-1,1}X_{t-1}+U_1 \\ Y_2 &= B_{02}+B_{12}X_1+B_{22}X_2\ldots B_{t-1,2}X_{t-1}+U_2 \\ Y_3 &= B_{03}+B_{13}X_1+B_{23}X_2\ldots B_{t-1,3}X_{t-1}+U_3 \\ &\vdots \\ Y_t &= B_{0t}+B_{1t}X_1+B_{2t}X_2 \quad B_{t-1,t}X_{t-1}+U_t \end{aligned} \quad (3)$$

Using matrix notation this set of equations can be reduced to:

$$\underset{t\times 1}{\underline{Y}} = \underset{t\times t}{\underline{B}} * \underset{t\times 1}{\underline{X}} + \underset{t\times 1}{\underline{U}} \quad (4)$$

The beta values in this last equation represent the technical coefficients of the system under consideration while the X values represent the different system components that were used.

The beta values must be interpreted within the context of the X and Y components of the equation in which the beta values are found. In their totality the beta values can be viewed as indicators of overall system performance and balance. Generally speaking, positive beta values should be viewed as indicators that the system is in balance. Similarly, negative beta values should be viewed as possible indicators of system bottlenecks. For example, if the Y value represented Central Processing Unit (CPU) busy, a negative beta coefficient on an X value that represents channel busy for some Direct Access Storage Device (DASD) channel tells us that as channel busy increases, CPU busy declines. This situation would seem to indicate that the processor in question was I/O bound on the channels that had negative beta coefficients. Further analysis would be required to confirm this conclusion.

Under different circumstances if the Y value represented channel busy a negative beta coefficient on an X value that represented channel busy for another channel indicates that as channel X busy increases channel Y busy decreases. This situation would tend to indicate that the channels in question were in contention with each other. A similar anlaysis and conclusion could be drawn for individual devices and control units that exhibited the same X,Y and beta coefficient values.

Positive beta coefficients under the right circumstances could also be viewed as an indication of potential system problems. If the Y value represents IMS response time and the X value in question represented channel busy a positive beta coefficient would tell us that as the channel busy increased so would the IMS response time. The channel in question could represent a bottleneck for the IMS application. Again, further study would be necessary before a definitive conclusion could be drawn.

2.3 Assumptions

In using the statistical tool of multiple regression, the analyst must make a number of assumptions about the nature of the data and the relationships between the variables that are involved in the analysis.

The first assumption implicit in one's use of multiple linear regression is the assumption that the relationship that exists between the Y and X values is a linear one. A low R^2 value is a statistical way of saying that the independent variables do not explain the variation in the values of the dependent variable. [1]

The second assumption that is implicit in the use of multiple regression is that the error terms (U_n) exhibit homoscedasticity, that is to say that the errors have a constant variance over the range of empirical observations.

A third assumption that is made in using multiple regression is that the error terms (U_n) are not autocorrelated. The existence of autocorrelation in error terms can be tested via use of the Durbin-Watson statistic. The absence of autocorrelation means that the level of one set of observations of independent values does not influence any other set of observations. In using data that measures device utilization of the different components of a computer system this assumption is often violated since what transpired on a computer system today can influence what will happen tommorrow. [2]

The fourth assumption that is made in using multiple regression is that the error terms are normally distributed. In dealing with a large number of data observations as is the case in doing computer capacity forecasting this assumption does not pose a problem. [3]

The fifth assumption is that collinearity does not exist between the independent variables of the multiple regression equation. It would be virtually impossible to select independent variables that exhibited no collinearity. If several variables are found to be collinear then they should be dropped from the analysis since they offer no additional explanation of the variation of the dependent variable. [4]

The final assumption of a multiple regression model is that the postulated equation or model is in fact the true model. This assumption of model significance can be evaluated through the use of a F-test.

2.4 Box-Jenkins Time Series Analysis

Once the structural characteristics of the computer system have been ascertained, the next step in the model building process is to forecast a trend in the use of all of the components of the system that make up the model. Collectively these trended values wil represent the X vector in the equation:

$$\underline{Y} = \underline{B} * \underline{X} + \underline{U} \qquad (5)$$

The statistical technique that was selected to perform this task was Box-Jenkins time series analysis.

In developing the forecasted values for the $\underline{X}$ vector, using the Box-Jenkins approach involves a three step process. The first step is to identify the functional form of the Box-Jenkins model to be used. Once the form of the model has been identified the next step is to estimate the value of the parameters of the functional form. Once both the functional form of the model and the parameters of the model have been estimated the final step in the process is to make a forecast.

2.5 Model Identification

The purpose of model identification is to determine the appropriate class of model that the data represents within a general class of models. In making this determination the value of three different parameters that represent the nature of the data must be identified.

The first parameter or order that must be identified is the degree of differencing that is required to make a time series stationary. A stationary time series is one that has a fixed mean value while a nonstationary time series such as the utilization of a computer system have an increasing or decreasing mean value. A stationary process can be squeezed from a nonstationary one by differencing the nonstationary process 'd' times. Thus we have:

$$\nabla X_t = X_t - X_{t-1} \qquad (6)$$

The number of times that the time series must be differenced to achieve stationarity becomes the value of 'd'.

Once the time series has been differenced as many times as necessary to produce a

stationary process, the autoregressive parameter, p, and the moving average parameter, q, can be estimated to give us a model that describes the time series model of order (p,d,q).

The process of determining the order of the autoregressive parameter requires a combination of applied statistics and tempered judgment. The process begins by calculating the values of the autocorrelation coefficient. These values represent the ratios of the covariances of the time series at different time lags. Thus the autocorrelation coefficient for lag j is:

$$P_j = \frac{V_j}{V_0} \qquad (7)$$

Where:

$V_j = E[Z_t - r)(Z_{t+j} - r)]$

$r = 1/t * \Sigma Z_t$

Z_t = The data observation of the time series at time t.

The order of the autoregressive process is generally assumed to be 0,1, or 2. If the time series in question has an autoregressive order whose value is zero then only P_1 will be a nonzero value. All of the remaining P_j values will equal zero. For an autoregressive time series whose order is 1, the value of the autocorrelation coefficients will tail off in an exponential manner as j increases.

For an autoregressive process whose order is 2 the values of P_j will exhibit either a mixture of exponentially increasing and decreasing values or they will exhibit a damped sine wave pattern.

Partial autocorrelation coefficients can next be derived using the autocorrelation coefficients that were calculated in the last step. For a time series whose autoregressive order is K we have:

$$P_j = \emptyset_{k1}P_{j-1} + \ldots \emptyset_{k(k-1)}P_{j-k+1} + \emptyset_{kk}P_{j-k} \qquad (8)$$

$j=1,2,\ldots.k$

Where all of the above Ø values represent the partial autocorrelation coefficients. Putting these equations into a matrix structure yields the Yule-Walker equations that can be expressed as:

$$\begin{bmatrix} 1 & \rho_1 & \rho_2 & \cdots & \rho_{K-1} \\ \rho_1 & 1 & \rho_1 & \cdots & \rho_{K-2} \\ \cdot & \cdot & & & \cdot \\ \cdot & & 1 & & \cdot \\ \cdot & & & \cdot & \cdot \\ \cdot & & & 1 & \cdot \\ \cdot & & & & \cdot \\ \cdot & & & & \rho_1 \\ \rho_{K-1} & \rho_{K-2} & \cdots & \rho_1 & 1 \end{bmatrix} \begin{bmatrix} \emptyset_{k1} \\ \emptyset_{k2} \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \emptyset_{kk} \end{bmatrix} = \begin{bmatrix} \rho_1 \\ \rho_2 \\ \rho_3 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \rho_K \end{bmatrix}$$

These equations are then solved for all relevant levels of K, the order of the autoregressive process. [5]

Once all of the values of Ø have been determined the order of the moving average process can be determined. Like the order of the autoregressive process the moving average order is generally assumed to be o,1, or 2. The order of the moving average process is assumed to be zero if all of the Ø values except $\emptyset_{11}$ are zero. The order of the moving average process is assumed to be 1 if the Ø values decline in an exponential manner. The order of the moving average process is assumed to be 2 if the Ø values represent either a mixture of increasing and decreasing exponential movement or if they exhibit a damped sine wave pattern. [6]

It should be stressed in concluding this section that the process of model identification is an iterative process that in many cases must be done repeatedly in order to get the desired results. This is due largely to the fact that time series model identification remains an art, and not science, that is based on rules of thumb rather than unbreakable laws of science.

2.6 Model Identification

Once the values of p,d, and q have been identified for the time series in question, the next step in the process is to identify the values of the parameters of the process that was identified in the previous step. The parameters to be estimated depend on the characteristics of the model that has been identified. A simple moving average model of order q, MA(q), takes the form:

$$Z_t = A_t - \theta A_{t-1} - \ldots \theta_q A_{t-q} \qquad (9)$$

Where:

$A_t = Z_t - Z_{t-1}$

The parameters of the model to estimated are:

$$\theta_1, \theta_2, \ldots \theta_q$$

A simple autoregressive model of order p, AR(p), takes the form:

$$Z_t = \phi_1 Z_{t-1} + \phi_2 Z_{t-2} \ldots \phi_p Z_{t-p} + A_t \quad (10)$$

The parameters to be estimated from this model are:

$$\phi_1, \phi_2 \ldots \phi_p$$

A more complex model that mixes both the autoregressive and the moving average model of order p,q or ARMA(p,q) takes the form:

$$Z_t = \phi_1 Z_{t-1} + \phi_2 Z_{t-2} \ldots + \phi_p Z_{t-p} + A_{t-1} - \ldots \theta_q A_{t-q} \quad (11)$$

The parameters to be estimated from this model are:

$$\phi_1, \phi_2, \ldots \phi_p, \theta_1, \theta_2, \ldots \theta_q$$

Finally, an autoregressive integrated moving average model of order p,d,q or ARIMA(p,d,q) takes the form:

$$Z_t = (1+\phi_1) Z_{t-1} + \ldots \phi_{p+d} Z_{t-p-d} - \theta_1 A_{t-1} \ldots - \theta_q A_{t-q} + A_t \quad (12)$$

The parameters to be estimated from this model are:

$$\phi_1, \phi_2 \ldots \phi_{p+d}, \theta_1, \theta_2 \ldots \theta_q$$

The statistical methodology that was used to estimate these values was a Marquardt Algorithm for nonlinear least squares. The purpose of the algorithm is to fit a function to a given set of data points and in so doing estimate the values of the parameters in the function.

The algorithm begins by asking for preliminary estimates of the parameters. These initial values were estimated using the Nelder-Mead Function Minimization technique which uses a simplex technique to find a minimum value for a known function given the constraints as defined by the user data.[7] In cases where there is only one parameter to estimate the Nelder-Mead technique proves sufficient by itself. Given the initial values to start with, the following derivatives are evaluated for each step in the Marquardt process.

$$X_{it} = - \frac{\partial a_t}{\partial b_t} \quad (13)$$

Where:

a_t represents the residual values of the estimated function.

b_t represents each of the parameters to be estimated.

From the X_{it} values a I x J matrix, referred to hereafter as $\underline{X}$, is formed where:

$$X_{ij} = \Sigma X_{it} X_{ij} \quad (14)$$

The next step is to form a $\underline{Y}$ vector where the elements of $\underline{Y}$ are:

$$Y_i = \Sigma X_{it} A_t \quad (15)$$

The $\underline{Y}$ vector and $\underline{X}$ matrix produce a set of modified linearized equations much in the same fashion as found in multiple regression and in a like manner the values of the $\underline{B}$ vector are solved for where:

$$\underline{Y} = \underline{XB} \quad (16)$$

The values of the $\underline{B}$ vector, the $\underline{Y}$ vector, and the $\underline{X}$ matrix are scaled before the system of equations is solved for $\underline{B}$. The scaling factor that is used in this step is:

$$C_i = \sqrt{X_{ii}} \quad (17)$$

Thus:

$$X^*_{ij} = X_{ij} / C_i C_j \quad (18)$$
$$B^*_i - B_i / C_i$$
$$Y^*_i = Y_i / C_i$$
$$X^*_{ii} = 1 + \Omega$$

The new parameters for the $\underline{B}$ vector are next determined and the sum of the squares is calculated for the new set of $\underline{B}$ values. The new $\underline{B}$ values. are derived as:

$$\underline{B}_1 = \underline{B}_0 + \underline{B}^* \quad (19)$$

Where:

$\underline{B}_0$ = The original estimates of $\underline{B}$

$\underline{B}_1$ = The new estimates of $\underline{B}$

$\underline{B}^*$ = The values derived from the previous step.

If the sum of the squares of the new estimate is less than the old, the parameter correction, $\underline{B}^*$, are tested for convergence. If all are smaller than the user set convergence value then convergence is assumed and a covariance matrix of the B_1 estimates is produced and the process is complete. If convergence has not occurred then the value of Ω is reduced by a user set factor of M and the entire process is repeated using the original $\underline{B}_0$ values. If the sum of the squares of the new estimates is greater than that of the old estimates then Ω is incremented by a factor of M and the entire process is again repeated using the original values of $\underline{B}_0$. [8]

2.7 Forecasting

Once the time series model in question has been identified and the parameters of the model have been estimated the last step in this phase of the model building process is to generate a forecast for what will become the values of the $\underline{X}$ vector variables of our general model.

The methodology that was selected to make the forecasts for each of the values in the $\underline{X}$ vector of the general model was the difference equation approach as suggested by Box and Jenkins.

The difference equation approach to time series forecasting begins by taking a conditional expectation of the difference equation form of the model that was identified for some time origin t. For example, if the model that was identified in the first step of the time series process was an autoregressive integrated moving average model of order (1,1,1), the difference equation for this type of model would be:

$$\hat{Z}_{t+1}=(1+\phi_1)Z_t-\phi_2 Z_{t-1}+A_t-\theta_1 A_{t-1} \qquad (20)$$

Taking the conditional expectation of this difference equation would yield:

$$\hat{Z}_{t+1}=(1+\phi_1)Z_t-\phi_2 Z_{t-1}+A_t-\theta A_{t-1} \qquad (21)$$

$$\hat{Z}_{t+2}=(1+\phi_1)\hat{Z}_{t+1}-\phi_2 Z_t-\theta A_t \qquad (22)$$

$$\hat{Z}_{t+i}=(1+\phi_1)\hat{Z}_{t+i-1}-\phi_2\hat{Z}_{t+i-2} \quad \text{for all } i>2 \qquad (23)$$

For all forecasts beyond the time period of t+2 the A_t and θA_t elements of the difference equation fall out because the future expectation of all of the error components A_t equal zero. Thus:

$$E(A_t) = 0 \qquad (24)$$

The A_t value has a direct influence on the forecast for the first two time periods of the forecast since the actual value of the A_t values can be used. For forecasts beyond time period three in the above equation the actual A_t values cannot be used since they have not taken place and as such are not known. This is not to say that the influence of the A_t values is not embedded in the forecasts at lead times greater than two as all of the future forecasts are indirectly dependent on the forecasts in time periods one and two which in turn are dependent on the value of A_t. [9]

The Z_t values that are used as seed values to start the time series are derived from a linear time series least squares. The mean value of the time series could be used if the time series was stationary. Since this is not the case for any of the components in a computer system, these seed values must be derived before the final forecast can be made.

Once the forecast for each of the independent $\underline{X}$ vector variables has been made, the forecast values are plotted out into the future and the general model itself can now be solved.

2.8 Simultaneous Equations

One of the shortcomings of both multiple regression and time series analysis is the requirement that a distinction be made between explanatory and dependent variables. Once this distinction is made these statistical techniques imply that there is a certain degree of causality whose direction goes from the explanatory to the dependent variables. In a computer system maintaining this distinction of explanatory and dependent variables based on assumptions of causality is a difficult if not spurious task. This is the

primary reason for turning to the technique of simultaneous equations where the values for all of the variables in the model can be determined at the same time. In dealing with the statistical technique of simultaneous equations the steps in the forecasting process that must be made are those of model identification, estimation, and the solution.

2.9 Model Identification

The crux of the identification problem resolves around the statistician's ability to estimate the functional equations of the system under consdieration from the data that is available. In layman's terms the identification problem asks the question of whether a solution exists to the system of equations that is both meaningful and unique.

Whether or not a system of equations can be identified depends on its ability to meet the standard rank and order conditions that have been established for simultaneous equation systems. The order condition, which is a necessary but not sufficient condition for system identification states that in a model of M equations each equation in that model will be identified if it excludes at least M-1 of the variables appearing of the variables appearing in the model. The rank condition which is both a necessary and sufficient condition for model identification states that in a system of M equations with M variables that are to be determined internally to the model an equation within the system of equations is considered to be identified if and only if at least one nonzero determinant of the order (M-1),(M-1) can be constructed from the coefficients of all of the variables in the model that were excluded from that particular equation in question but included in the other equations in the system. [10]

Once it has been determined that the model in question can be identified, the next step in the process is to estimate the equations in the system.

2.10 Model Estimation

The method of model estimation that was used for this step in the model building process was two stage least squares multiple regression. Since multiple regression was discussed in length earlier in this paper, comments about its use will be limited. A two stage least squares approach is taken so as to minimize any correlation that may exist between dependent variables that appear in each regression equation and any error term of the other independent variables.

The elimination of this correlation which would lead to bias in the final system of equations is accomplished through the use of two stage least squares regression. The first stage regression produces estimates of all of the dependent variables based on the original data. The second stage regression produces estimates of all of the dependent variables based on the estimates of the first stage.[11]

2.11 Model Solution

The system of equations is solved by a reduced form algorithm that expresses each of the unknown variables of the system as a function of the predetermined values of the model. This is done through a process of substitution that reduces each equation down to the point where there is but one unknown variable in each equation. Once the process of reduction through substitution has taken place the solution of each of the remaining equations for the value of the unknown variables becomes a trivial task.

3.0 The Data

A listing of the data that was used in the case study is available on request from the author. In total there were 323 observations where each observation represents the average percentage utilization of a particular set of system components over the course of a day for the entire year of 1979. The data were captured by a hardware monitor and stored by ten minute intervals on a data base. For the purposes of this study, each of the 144 daily observations were aggregated and then averaged to produce the actual data observations. The different data elements that were used in this study are:

Julian Date

Central Processing Unit Busy

Problem Program State Busy

Idle-Stop

Channel 1 Busy - DASD Channel

Channel 2 Busy - DASD Channel

Channel 3 Busy - MSS Channel

Channel 4 Busy - DASD Channel

Channel 5 Busy - Tape Channel

Channel 7 Busy - DASD Channel

Channel 8 Busy - DASD Channel

Channel 9 Busy - MSS Channel

Channel A Busy - DASD Channel

Channel B Busy - Tape Channel

The major shortcoming of the data exists in the periodic gaps that can be found. These gaps in the data were due to either a failure in the hardware monitor, a failure in the system that was under study, or an error on the part of the hardware monitor operators. In total these gaps represent less than 10% of the total data that could have been captured and in no way should bias the overall results of the model.

4.0 The Fitted Model

In the multiple regression segment of the model building process a two stage least squares approach was taken to insure that the equations that were produced for the simultaneous equation portion of the model were unbiased. The second set of regressions were run using the estimates generated from the first set of regressions. In the second stage regressions all coefficients that were not significantly different from zero at a confidence interval of 90% using a two-tailed t-test were dropped from consideration. The set of beta coefficients that were produced by this two stage process can be found in Table I.

In addition to the beta coefficients there are a number of other vital statistics that were derived from each regression to ascertain whether or not any of the original regression assumptions were violated. A listing of these statistics for both the first and second stage regressions can be found in Table II along with their corresponding dependent variable.

The statistics of interest in determining the usability of the regression equations are the values of R^2, F, and the Durbin-Watson statistic. An examination and comparison of these statistics between the Stage I and Stage II regressions will reveal that the critical statistics in question are pretty much the same. The F values from the Stage II regressions were across the board higher than the F values from Stage I although this is probably due to the dropping of the insignificant variables in performing the Stage II regressions. The F values in Stage I varied from a low of 97 to a high of 541 and in all cases represent highly significant values at a 95% confidence interval. The F values from the Stage II regressions varied from a low of 130 to a high of 1643 and were also in all cases highly significant at a 95% confidence interval.

The R^2 values for both Stage I and Stage II regressions were similar. In both cases the R^2 values varied from .77 to .95 which would tend to indicate that a high degree of the variation in the dependent variables can be explained by the independent variables.

The statistic that casts a dark pale over the results in general is the Durbin Watson statistic. As discussed earlier, the Durbin-Watson statistic is a measure of autocorrlation that might exist in the data under study. As anticipated the Durbin-Watson statistic in both Stage I and II regressions were in the range of.9 to 1.7 which indicates that autocorrelation exists. The existence of autocorrelation will not bias the estimates of the beta coefficients which is our primary concern. At worst, the existence of autocorrelation will bias the variance of our estimates of the beta's which in turn would mean that the values computed for the F and R^2 statistics are not as good as we had previously believed them to be.

Plots of the residuals from both Stage I and II regressions are availbale from the author. A visual examination of these plots indicated that several of the standard linear regression assumptions about the nature and distribution of the error terms have been at the very least violated in spirit if not in fact. The assumption in question here relates to the lack of autocorrelation among the residuals, the existence of a constant variance and the selection of the X values by a fixed and repeated sample. Again, as in the case of the spurious Durbin-Watson statistics it was felt that these violations of the standard assumptions would only bias the variance of the regression coefficients and not the coefficients themselves.

4.1 Time Series Model Identification

The process of identifying the structure of a time series is an art and not a science. In trying to make a determination of the values of the parameters p, d, and q there must be a trade off made between the different criteria that can be used for the purposes of identification. Given this disclaimer, the following is an explanation of the structure of the time series that were developed as inputs into the general model. The output listings for the autocorrelation coefficients and the partial autocorrelation coefficients are available from the author.

For the simplicity, the identification notation used by Box-Jenkins to characterize the different types of models will also be incorporated here. Thus a simple first order autoregreesive model becomes AR(1) while a comparable moving average model would be MA(1). An integrated autoregressive model would be ARI(1,1) while a comparable moving average model would be IMA(1,1). Combining

Table 1

Beta Coefficients

Dependent Variable	1	2	3	4	5	6	7	8	9	10	11	12	13
1	44.06	.61	-.40	-.36	-.29	.67	.24	0	0	.29	-.39	.30	0
2	-2.92	.20	0	0	0	0	0	.44	0	-.09	.17	-.16	0
3	77.00	0	-.69	-.82	0	.41	0	0	0	.24	0	0	-.46
4	2.33	0	-.02	0	.23	.37	.17	0	.91	-.17	-.54	0	-.07
5	.83	0	0	.36	-.03	.24	.25	0	-.37	.56	0	-.09	.08
6	-3.48	0	.04	.42	.11	.05	.17	0	-.39	-.16	.79	0	0
7	-.53	0	0	.31	.27	.28	.05	0	-.17	0	0	.24	0
8	.89	.21	0	0	0	0	0	0	0	0	-.21	.14	.82
9	.03	0	0	.82	-.21	-.34	-.11	0	0	.17	.49	.14	.04
10	-3.83	-.08	.04	-.27	.62	-.34	0	.09	.38	.06	.18	.40	-.15
11	.07	.05	0	-.39	0	.65	0	-.12	.42	.10	-.02	.25	.07
12	-.45	-.06	0	-.09	-.08	0	.17	.08	.23	.29	.43	.05	0
13	2.8	0	-.04	-.14	.10	0	0	.94	.13	-.14	.09	0	0

Table II

Stage I Statistics

Regression Number	Dependent Variable	F	R^2	Durbin-Watson
1	CPU Busy	109	.79	1.149
2	Problem Program State	111	.81	1.471
3	Idle-Stop	97	.77	.933
4	Channel 1 Busy	333	.92	1.264
5	Channel 2 Busy	137	.84	.949
6	Channel 3 Busy	255	.91	1.008
7	Channel 4 Busy	160	.85	1.019
8	Channel 5 Busy	540	.95	1.408
9	Channel 7 Busy	408	.94	1.345
10	Channel 8 Busy	164	.86	1.758
11	Channel 9 Busy	385	.94	.963
12	Channel A Busy	222	.90	1.090
13	Channel B Busy	541	.95	1.310

Stage II Statistics

1	CPU Busy	130	.79	1.167
2	Problem Program State	269	.81	1.442
3	Idle-Stop	215	.77	.926
4	Channel 1 Busy	499	.93	1.283
5	Channel 2 Busy	207	.84	.920
6	Channel 3 Busy	383	.91	.988
7	Channel 4 Busy	295	.85	1.003
8	Channel 5 Busy	1643	.95	1.391
9	Channel 7 Busy	616	.94	1.341
10	Channel 8 Busy	180	.86	1.764
11	Channel 9 Busy	516	.94	.956
12	Channel A Busy	298	.90	1.084
13	Channel B Busy	857	.95	1.291

both the autoregressive and moving average forms would yield ARMA(1,1) and in its integrated form this would be ARIMA(1,1,1).

An examination of the autocorrelation and partial autocorrelation coefficients from the CPU busy time series revealed this data to take a structure such that p=2, d=1, and q=o. As such we can call this an ARI(2,1) model or an autoregressive integrated model of order 2,1. None of the partial autocorrelation coefficients of this time series at any degree of differencing had either an exponential or a damped sine wave pattern thus eliminating the moving average component of time series from consideration. In examining the autocoreelation coefficients after a first differencing had been taken, a damped sine wave pattern could be discerned. This pattern in the autocorrelation coefficients tells us that the data is autoregressive of order 2. Taken with the need to difference the data once to achieve this pattern we can conclude that we have a time series of the structure ARI(2,1). As could be expected, the other components of the system that were directly related to the central processing unit also took on an ARI(2,1) model structure. These components were the problem program state and idle-stop.

For channel 1, a DASD channel, the autocorrelation coefficients of the original series, before taking a first difference, were not significantly different from zero after the first lag. These coefficients, however, exhibited a damped sine wave pattern after both a first and second difference was taken which suggests a model that could have at least an ARI(2,1) structure.

An examination of the partial autocorrelation coefficients after both a first and a second differencing revealed that they both declined in a smooth exponential manner. As such we have a moving average component in this time series with the total model taking the form of either ARIMA(2,2,1) or ARIMA(2,1,1). For the sake of parsimony the second model structure was used.

For channel 2, also a DASD channel, an examination of the autocorrelation coefficients revealed that regardless of the degree of differencing they did not follow any type of set pattern. As such it was assumed that this time series did not have an autoregressive component to it. An examination of the partial autocorrelation coefficients revealed that after a second differncing the coefficients declined in an exponential manner. As such this time series was assumed to take the structure of a first order moving average model with two degrees of differencing or simply IMA(2,1). All of the other DASD and mass storage channels produced similar autocorrelation and partial autocorrelation coefficients and as such were all classified as IMA(2,1) type models.

For channel 5, a tape channel, the partial autocorrelation coefficients exhibited no discernable pattern thus eliminating the moving average component of the model from consideration. The autocorrelation coefficients of the original time series exhibited a damped sine wave pattern which made this time series a second order autoregressive time series with no degrees of differencing or AR(2). Channel B, the other tape channel, also took on a AR(2) model structure.

A summary of the different model structures can be found in Table III. It is interesting to note the commonality of model structure taken on by the different system components. All of those components that had anything to do with the central processing unit took on a common AR(2,1) structure while all of the DASD and MSS channels took on an IMA(2,1) structure with the exception of channel 1. This could be due to the fact that channel 1 has the system residence packs on it which would conform to very different performance characteristics when compared to the average DASD channel. Both tape channels took an AR(2) model structure.

4.2 Time Series Estimation

Having identified the structured form of the time series to be forecasted, the next step in this process is to develop the difference equations for each of the four different types of time series that were identified in the previous step. Once the appropriate difference equation has been determined for each of the time series in question, the parameters of each difference equation can be estimated.

The four difference equations that were used for forecasting the input values for the $\underline{X}$ vector of our general model are derived from the functional structural forms that have already been identified. These four equations are:

For the AR(2) model:

$$Z = \emptyset_1 Z_{t-1} + \emptyset_2 Z_{2-t} + A_t \qquad (25)$$

For the ARI(2,1) model:

$$Z_t = (1+\emptyset_1)Z_{t-1} - \emptyset_1 Z_{t-2} + (1+\emptyset_2)Z_{t-2} - \emptyset_2 Z_{t-3} + A_t \qquad (26)$$

TABLE III

BOX-JENKINS TIME SERIES SUMMARY

TIME SERIES	NATURE OF THE SERIES	MODEL STRUCTURE
1. CPU Busy 2. Problem Program State 3. Idle-Stop	Second Order Autoregressive Integrated with 1 degree of differencing	ARI(2,1)
4. Channel 1	Second Order Autoregressive Integrated First Order Moving Average with 1 degree of differencing	ARIMA(2,1,1)
5. Channels 2 - 4 Channels 7 - A	First Order Integrated Moving Average with 2 degrees of differencing	IMA(2,1)
6. Channel 5 Channel B	Second Order Autoregressive with 0 degrees of differencing	AR(2)

TABLE IV

Equation	Model Form	Parameter Estimates $\emptyset_1$	$\emptyset_2$	$\emptyset_1$
1 - CPU Busy	ARI(2,1)	.41926	-.39292	NA
2 - Problem Program State	ARI(2,1)	.45068	-.44883	NA
3 - Idle-Stop	ARI(2,1)	.45091	-.45946	NA
4 - Channel 1	ARIMA(2,1,1)	.50073	-.29656	-.29915
5 - Channel 2	IMA(2,1)	NA	NA	1.05
6 - Channel 3	IMA(2,1)	NA	NA	1.05
7 - Channel 4	IMA(2,1)	NA	NA	1.05
8 - Channel 5	AR(2)	.99999	7.45E-9	NA
9 - Channel 7	IMA(2,1)	NA	NA	1.05
10 - Channel 8	IMA(2,1)	NA	NA	1.05
11 - Channel 9	IMA(2,1)	NA	NA	1.05
12 - Channel A	IMA(2,1)	NA	NA	1.05
13 - Channel B	AR(2)	.99999	6.49E-7	NA

NA = Not Applicable

For the IMA(2,1) model:

$$Z_t = 2Z_{t-1} - Z_{t-2} + A_t - \theta_1 A_{t-1} \quad (27)$$

For the ARIMA(2,1,1) model:

$$Z_t = (1+\phi_1)Z_{t-1} - \phi_1 Z_{t-2} + (1+\phi_2)Z_{t-2} - \phi_2 Z_{t-3} + A_t - \theta_1 A_{t-1} \quad (28)$$

Given these difference equation forms the next step in the model building process is to estimate the ϕ and θ parameters of the models in question. This part of the model building process was accomplished through a two step approach that used a Nelder-Mead function minimization process to develop initial estimates and then a Marquardt non-linear least squares algorithm to develop the final estimates. For the IMA(2,1) models that had only one parameter, the Nelder-Mead process produced estimates that were more than adequate for our purposes. A summary of the parameter estimates can be found in Table IV.

4.3 Time Series Forecast

Having both identified the proper model form for each of the thirteen system vectors the last step in the time series analysis portion of the model building process is to develop a point forecast for each system vector. The point in time that was selected as the focus of this forecast was one year from the point in time when the empirical data came to an end or roughly speaking January 1981.

In making this type of time series forecast the most important step is selecting an appropriate starting point or "seed" value from which to begin the analysis. If the time series in question are stationary and have a fixed mean then the mean value of the time series is the appropriate starting value. Unfortunately for our purpose none of the time series that we are manipulating are stationary and as such the appropriate starting values must be derived.

For our purposes these starting values were derived using a simple two variable least squares linear regression. The output from these regressions are available from the author. In each of the regression plots the Y axis represent the dependent value of one of the thirteen system components that are under consideration while in every case the X axis represents time. The dotted line on both sides of the regression line represents a 95% confidence interval. The extremely low value of the R^2 statistics for all but one of the regressions is an indication as to why the method was used only to establish a starting point for our forecast and not to make the full forecast itself. Using this method the starting values that were used for the time series analysis for each of the system components is as follows:

Equation	Description	Starting Value
1	CPU Busy	48.33
2	Problem Program State	23.80
3	Idle-Stop	27.72
4	Channel 1 Busy - DASD	14.33
5	Channel 2 Busy - DASD	15.36
6	Channel 3 Busy - MSS	9.99
7	Channel 4 Busy - DASD	12.99
8	Channel 5 Busy - Tape	30.21
9	Channel 7 Busy - DASD	13.59
10	Channel 8 Busy - DASD	9.11
11	Channel 9 Busy - MSS	10.64
12	Channel A Busy - DASD	12.57
13	Channel B Busy - Tape	36.39

The final step of the forecasting process for the time series analysis portion of the model is to place the starting values for each of the system components into their appropriate equations and then to plot out a forecast for the values in question for an entire year. The forecasted values that were used derived from this process and used in the final simultaneous equation solution process are as follows:

Equation	Description	Starting Value
1	CPU Busy	74.23
2	Problem Program State	36.14
3	Idle-Stop	33.63
4	Channel 1 Busy - DASD	21.73
5	Channel 2 Busy - DASD	26.22
6	Channel 3 Busy - MSS	13.61
7	Channel 4 Busy - DASD	16.61
8	Channel 5 Busy - Tape	30.10
9	Channel 7 Busy - DASD	16.48
10	Channel 8 Busy - DASD	13.82
11	Channel 9 Busy - MSS	17.88
12	Channel A Busy - DASD	16.19
13	Channel B Busy - Tape	36.25

4.4 Simultaneous Equations

The final step of this entire process was to bring together both the Box-Jenkins estimates and the linear regression estimates into a system of simultaneous equations. The equation that was used to accomplish this step in matrix form was:

$$\underline{Y} = \underline{B}\ \underline{X}$$

The level of utilization for the different system components that was derived from the solution of this equation were as follows:

	Component	%Utilization
1)	CPU Busy	76.15
2)	Problem Program State	145.69
3)	Idle-Stop	81.91
4)	Channel 1	21.74
5)	Channel 2	59.29
6)	Channel 3	26.08
7)	Channel 4	33.26
8)	Channel 5	36.63
9)	Channel 7	31.06
10)	Channel 8	50.02
11)	Channel 9	26.64
12)	Channel A	47.53
13)	Channel B	14.41

While the Channel and CPU busy forecasted estimates have the appearance of reasonableness both Problem Program State and Idle-Stop do not. This anomaly could be due to any number of factors and more will be said about it in the concluding remarks.

5.0 Concluding Observations

In a very literal sense the output of this model represents point estimates of component resource utilization for Mountain Bell's Colorado/Wyoming processing center's 3033 JES3 global processor for January 1981 based on the assumption that no major hardware or software changes are made before that data. Should such changes be made, however, the real value of the model will come into play as such changes can be reflected in changes in the value of the $\underline{X}$ vector. These new $\underline{X}$ vector values can then be used to once again solve the system of simultaneous equations to ascertain the impact of the change in one of the system components on the entire system. As such the value of the model is three fold. First, the model does have the capability of forecasting future computer resource utilization. Secondly, the model also has the capability of pointing out current as well as potential system bottlenecks. Finally, the model has the ability to answer 'what if' questions about proposed system changes.

In using this model for capacity planning at Mountain Bell there are a number of directions that will be taken in the future. The first such step must be to refine the current model. The case study that has been presented in this paper illustrates the potential pitfalls that are involved in trying to develop a dynamic system model. The results of the simultaneous equation system points to two areas in particular, problem program state and idle-stop, where additional work needs to be done to refine the technical coefficients that were developed in the multiple regression portion of the model building process. The final results of these two system components should not be viewed as proof of the inappropriateness or weakness of the methodology but rather as opportunities where further work should produce more useable results. The inappropriateness of these values indicates a need to reexamine the variables that were used to forecast these values in the model.

Another task that needs to be addressed with respect to the future use of this methodology at Mountain Bell is the task of model verification. Before this model can be used with confidence for the purposes that it was designed, it must be tested using historical data to ascertain its useability and its accuracy.

Once the model variables have been stabilized, and the model results verified, the last task that must be accomplished before this methodology can be broadly applied is that the software that was used to develop these results must be rationalized, consolidated and simplified so as to make it possible for the average technician, who is unfamiliar with the statistical techniques involved, to produce a forecast.

5.1 Acknowledgements

It would have been impossible for me to have completed this paper without the generous contribution of time and effort by all of the members of the Hardware Forecasting and Performance Evaluation Group at Mountain Bell. I would like to give a very special thanks to Ms. Esther Martin for her secretarial skills and unlimited patience. The errors that remain in this document despite these individual's efforts remain my sole responsibility.

References

[1] Bhattacharya, Gouri and Johnson, Richard, Statistical Concepts and Methods, John Wiley & Sons, Inc. New York, c. 1976, p.396.

[2] Johnston, J., Econometric Methods, McGraw-Hill Book Company, c. 1972, pp. 246-249

[3] Box, George, and Jenkins, Gwelyn, Time Series Analysis, Holden-Day, c. 1970, pp. 88-89.

[4] Box and Jenkins, pp. 173-177

[5] Box and Jenkins, pp. 64-65.

[6] Box and Jenkins, pp. 169-176

[7] Nelder, J.A., and Mead, R.A., "A Simplex Method for Function Minimization", The Compten Journal, 7, 1965, pp. 308-313.

[8] Marquardt, D.W., "An Algorithm for Least Squares Estimation of Non-linear Parameters", Journal of the Society for Industrial Applied Mathematics, June 1963, pp. 431-441.

[9] Box and Jenkins, pp. 129-132.

[10] Gajarati, Damodar, Basic Econometrics, Basic Books, New York, c. 1977, pp. 360-363.

[11] Johnston, pp. 346-350.

CPEUG80

Measuring System Performance

A Performance Evaluation Study of UNIX[1]

Luis Felipe Cabrera[2]

Computer Science Division
Department of Electrical Engineering and Computer Sciences
and the Electronics Research Laboratory
University of California, Berkeley CA 94720.

Different performance aspects of computer systems on which the timesharing operating system UNIX runs are presented.

A comparison of the performance of three installations is made and the method discussed. The effects of distinct upgrading changes made in the systems, namely, the addition of a cache memory, of a disk drive and of main memory are also reported.

Key words: Comparison; performance of computer systems; UNIX; upgrading changes.

1. Introduction

Since its first release by Bell Laboratories in 1974, the timesharing operating system UNIX [1] has run on a wide variety of mainframes. Primary among them is the PDP-11 family manufactured by Digital Equipment Corporation. Moreover, UNIX also runs on Digital's recently released VAX 11/780, even though its hardware presents drastic differences with the PDP-11 family, and has also been ported to the INTERDATA 7/32 and to the UNIVAC 1100 series machines.

It has then become of interest to measure the effect that the underlying hardware has on the performance of the various computer systems on which UNIX runs. In what follows a study of three computer systems is presented. Each was deliberately chosen to be substantially different from the other two.

When we compared systems, we tried to assess the effect on performance of leaps in technology and design. On the other hand, when we analyzed an upgrading change, we observed the effect this new hardware component had on the performance of the given system. Moreover, the upgrading changes made to each of the systems provide good examples of what may be achieved when such a change is implemented in an installation.

All the systems studied operate in the Electrical Engineering and Computer Sciences Department of the University of California at Berkeley. Measurements were taken over a span of five months (April through August 1979) beginning by the third week of the Spring Quarter of 1979.

The rest of the paper is divided as follows. Section 2 presents the basic design decisions about the experiment as well as the measurement techniques and the reduction of the data. In section 3 we present and analyze diagrams which compare the performance of the systems. Section 4 shows the results of the upgrading changes and section 5 contains our conclusions.

---

[1] UNIX is a trademark of Bell Laboratories

[2] This research was supported in part by the National Science Foundation under grants MCS78-24618 and MCS78-07291, by a research grant of the Pontificia Universidad Católica de Chile, and by the Italian National Research Counsil under grant CNR 78.02548.07

2. Measuring UNIX Performance

2.1 Preliminaries

When comparing the performance of different installations, many problems are resolved if the same operating system runs in all of them. This is because the identical user interface makes it posible to define a high-level language benchmark and use it unmodified in each of the systems.

When this is the case, *functional equivalence* of the benchmark is immediately obtained [4]. Whether this equivalence corresponds or not to a *resource consumption* equivalence will depend on the actual implementation of the operating system. Determining whether or not it yields a *responce time* equivalence is the main purpose of our comparison study.

Choosing response time as the main observed performance index of our study is justified by our belief that in any timesharing system, from the user's point of view, what counts most is the responsiveness of the system to user supplied tasks. Nevertheless, using standard UNIX instrumentation, we have also monitored system time (the cpu time spent in the operating system) and user time (the cpu time spent executing the command) for each of our tasks. This has proven very helpful in analyzing possible causes for effects noticed while studying the performance of each system.

Characterizing the work load is a central problem in any benchmarking experiment where a comparison between systems is made. The question about the conditions under which the experiment should be performed has been given a number of different answers. Perhaps the best way to answer it is to design an experiment where a benchmark is executed in each system on a stand alone basis and where complete control of the work load is achieved by loading the system with some kind of internal or external driver. The main advantages of this method are total reproducibility and absolute control of all the activities in the system.

The main difficulties it presents are related with the availability of tools to drive the system and with the design of the work load that those tools will implement. The (artificial) work load under which the system is to be studied must be such that results obtained from its usage should yield information about the system's performance under its natural work load.

Lack of such tools at the time when we had to make this decision led us to the less sophisticated approach of monitoring the systems periodically under their *natural* work load with the aid of a *shell script*[3]. Then, in order to analyze our results we had to find satisfactory characterizations of load with respect to which the performance indices corresponding to a given task would be measured. We decided to use single variable characterizations of load for this purpose.

We have characterized work load by taking two different yet related viewpoints. One is an *outside* viewpoint (how much work is being done on the system) and the other is an *inside* viewpoint (how much work is being demanded from the system).

For comparisons as well as for analyzing upgrading changes, the first viewpoint yields two rather coarse but very natural characterizations of load: the number of users (logged in while our benchmark begins to run) and the number of active users (those which are executing some task when our benchmark begins to run). For comparing installations we found that the most interesting characterization based on the second viewpoint was that based on the number of real processes. This is the number of processes which are likely to run while one of the tasks in our benchmark will be running. Its precise definition is the following: a process in the process table is a real process if it is not a *login shell* or a *sleeping shell*[4]. It is clearly less coarse than the above two characterizations.

The three systems we monitored had the C shell [2] running in addition to the ordinary shell. To make the experiment portable in our environment, we wrote a script for the C shell which was run on the background (with the same priority as any user process) in each of the systems. Its text is reproduced in [8] together with the text of all programs and files used by it. Documentation

[3] In UNIX the *shell* is a command language interpreter through which the user inputs its tasks to the system [1]. Moreover it is also a programming language [2] thus shell programs (*shell scripts*) can be written with it.

[4] A *login shell* is a process waiting for input from a terminal in which no user is logged in. In UNIX, whenever a user inputs a command, the shell forks spawning a child shell which executes the command. While the child shell executes the command, and unless the command has been issued followed by an &, the parent shell sleeps till the completion of the command. We call *sleeping shells* this last type of process.

regarding the usage of the script and about changes needed to convert the script into one for the standard UNIX shell can be found in [8].

2.2 The Script Driver

Our strategy for monitoring each system's responsiveness was to run periodically a set of predefined benchmarcks. This was achieved by using a shell script which contained these tasks together with commands which gathered statistics about the work load and the time it took the tasks to complete. The script was run every other week during the normal operations of the systems. The script, after cycling through its instructions once, would go to sleep for twenty minutes and then wake up to initiate the cycle again.

This data gathering method can be categorized as a time-sampling tool [3] and in fact is cery similar to Karush's [7] *terminal probe* method. By using it we measure the work load of the system as well as the dependency of our performance indices on the underlying equipment. Thus we are actually evaluating the installation.

Although running our script affects the load of the system, and thus its responsiveness, it was felt that this was irrelevant for a comparison study because all systems were going to be presented with the same script. In fact the main purpose of the comparison experiment is precisely to observe how each system reacts to this stimulus.

Our commitment to use standard UNIX features, for portability reasons as well as for assuring the functional equivalence of the benchmark, made us decide upon the usage of the *time* command [5] as the measurement tool for our tasks. The time command returns, upon completion of the command with which it is called, three measurements: response time (the elapsed time during the command), system time and user time. These last two are accurate to one tenth of a second while the first is accurate to one second. The time command truncates, does not round off.

This low resolution of time together with our desire that no individual measurement be off by more than 10% led us to consider tasks which would never take less than five seconds to complete. On the other hand we could not come up with a set of tasks that would overload the system every time they were run, if we were to run them periodically for an extended period of time. Our script was designed as a compromise between these requirements.

Four tasks were timed at each run: a C compilation, the execution of a CPU-bound job, the retrieval of the manual page of the on-line copy of the UNIX Programmer's manual, and a mix, which included the above three plus some I/O bound tasks and two system commands; the mix was called "Script", even though it did not correspond to the actual script we were running.

2.3 The Tasks Chosen

Our desire was to measure a set of tasks that would be representative of the user's tasks and that would stress distinct aspects of a configuration. The lack of a Pascal compiler or interpreter for the VAX at the time made us use the language C [6]. Two of our tasks were then the compilation of a short CPU-bound program and its execution. Our CPU-bound program executes its inner sequence of instructions 100,000 times. In all performs 1,200,000 integer arithmetic operations plus those needed for the for loops. It was decided against using floating point arithmetic because one of the measured systems, the 11/40, does it in software. This precluded any meaningful hardware-oriented comparison.

As for our third task, rather than having a strictly I/O bound task we felt it would be more interesting to choose a task which would involve more features than just the speed of the disks and the efficiency of the I/O subsystem. Thus, the command *man man*, which retrieves the entry for the manual page out of the on-line copy of the UNIX Programer's manual, was chosen, because, as the on-line copy is kept in compact form on disk to save space, this copy is retrieved using the formatting program *nroff*, a utility program widely used in text processing. To avoid problems when running the script, the output of man man was sent to /dev/null instead of sending it to a real terminal. This has the effect of discarding the already formatted text of the retrieved page.

The last task timed is what we called Script. For technical reasons we used the *date* command to determine total elapsed time. The only drawback is that it does not yield system time nor user time for this task. Script, besides including the above three tasks, also includes six short I/O-bound tasks, a couple of date commands and two commands which gather extensive data about the system. Script is thus a fairly balanced task and can be considered as an indicator of throughput.

2.4 The Systems Measured

Our main motivation when choosing which systems to monitor was diversity. Our choices were the following:

(1) A PDP 11/40 with 200K bytes of main memory, one DIVA disk controller with three DIVA disk drives which have 50M byte disk packs. This system has 23 ports and no floating point arithmetic unit.

(2) A PDP 11/70 with 1.3M bytes of main memory, a 2K byte cache memory, one DIVA disk controller with four DIVA disk drives which have 50M byte disk packs and an RS04 fixed head disk used as swapping device. This system has 81 ports.

(3) A VAX 11/780 with 512K bytes of main memory, an 8K byte cache memory, one Digital disk controller with one RP06 disk drive with 177M byte disk packs. This system had 16 ports.

Throughout the rest of this paper we shall refer to these systems as the 11/40, the 11/70 and the VAX respectively.

The configurations described above are those which each system had during most of our data gathering period.

The work loads of the three systems are as diverse from each other as their underlying equipment. The 11/40 is mostly used for administrative matters. The 11/70 is mostly used by undergraduate students in coursework-related activities. The VAX is primarily used by advanced students in research related tasks.

Given this diversity of the natural work loads, the problem of work load characterization for the comparison of the responsiveness of the different systems becomes very important as well as more difficult. Our characterizations described in 2.1 proved to be reasonably adequate. Of course, the analysis of our results was inevitably influenced by considerations about the work load of the individual systems.

2.5 Reducing the Data

The data points provided periodically by our script were gathered in appropiate files and latter processed using time stamps. We were thus able to plot mean user-time, mean system-time and mean response-time versus Users, Active Users, Processes and Real Processes for each task in each system together with the standard error of the corresponding sample. The standard error of a sample is defined as the square root of the ratio between the variance and the cardinality of the sample. For response time we found the standard error to be consistently large and thus ploted 90th-percentile curves. All the diagrams mentioned above can be found in [8].

To prevent outliers from appearing in our curves we chose the minimun size of samples by the iterative method of plotting the curves with samples of increasing size until outliers would not appear. Whenever we did not have enough points in a sample we clustered samples corresponding to consecutive values of the variable. From this larger sample the 90th-percentile was chosen. The x-coordinate used for such a point was the average of the x-coordinates corresponding to the clustered samples. We used samples of size sixty.

We believe this approach to go a long way in solving the conflicting problems of insufficient number of data-points, presence of outliers in the samples, large variance of some samples and finite amount of resources

Table 1 displays the amount of data points gathered. Only for the 11/70 we felt had an insufficient amount and this was due to the large number of ports this installation has. Taking our experience as a model, one should try to obtain fifty measurements per value of the variable characterizing load to avoid clustering samples.

Table 1. Number of Sample Points per System.

SYSTEM	Number of Sample Points
11/40	536
11/70	1007
VAX	1368

3. Comparison of the Installations

Rating different installations with our method presents several difficulties. They all emanate from the diversity of the work loads and the lack of satisfactory characterizations for them.

Those characterizations based on an external view of the system fail to be accurate because they are dependent on too many factors. Clearly, a student doing research submits jobs which are of a different caliber

from those submitted by a user editing a file. Thus we believe that our comparison curves based on this view penalize the VAX because of the quality of its users (see 2.4). Likewise, our characterizations of load based on an internal viewpoint fails to distinguish between types of jobs.

For comparing response time, we found that our characterization of load in terms of real processes provided us the best overall view as to how the systems performed relative to each other. Moreover, it also seems to be the best single indicator of the true level of activity of each system.

Finally, the disparity of configurations is also an important degrading factor for the VAX. This system was supporting all of its I/O activity, including swapping, in one arm. In contrast the 11/70, which swaps on drum, has several other spindles to handle the rest of the I/O.

We have chosen to present here only two of the four tasks monitored because they are representative of the behavior observed. The other results may be found in [8].

3.1 CPU-bound Job

As mentioned in section 2.3 this task was the execution of the code generated by a very simple C program which had two nested for loops. Given its small size (the C code has 27 lines, including comments), the probability of being swapped out from core while on the system is minimal. Thus, assuming same scheduling in all three systems, its processing time only depends on the lengths of the CPU and memory cycles, the effect of the load in terms of I/O interrupts and the size of the ready queue.

In figure 1 we see that the VAX ranks uniformly first while the 11/40 ranks third. This same ranking was observed under our external characterizations of load, but there the slopes of the curves of the 11/70 and the VAX were almost identical. This is the task where difference between the three systems is most visible.

We could observe the amount of user time the CPU-bound job took in each system. At the *five* user level the 11/40 took on the average 21.0 seconds, the 11/70 8.5 seconds and the VAX 5.66 seconds. At the *fourteen* user level the 11/70 took 8.61 seconds while the VAX took 5.81 seconds. We feel that user time for this task is a very good indicator of the speed of computation of the machines. The VAX does show to be a substantially faster

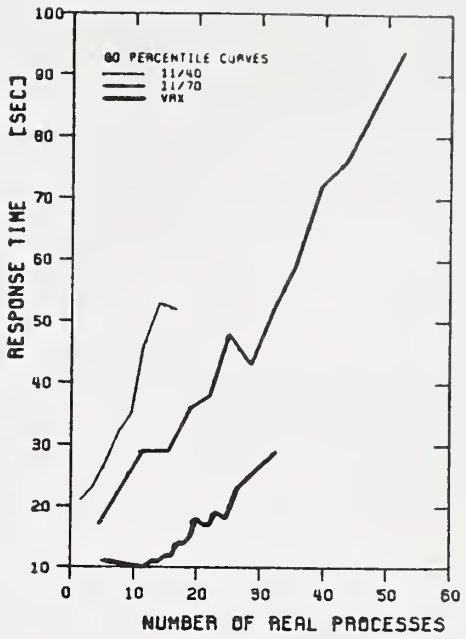


Figure 1. Response Time of CPU-bound job versus Real Processes.

machine than the other two.

Given that no I/O is done, we expected to observe the fairly linear behavior we obtained. This was true for all of our characterizations of load. Response time for the 11/40 consistently showed to have a larger rate of growth than for the other two systems.

3.2 The Script

Figure 4 contrasts with figures 2 and 3 in that it shows the VAX no longer being slower than the 11/70 throughout the whole range of load. Script includes several tasks which make extensive use of the I/O subsystem, and so the inferior VAX I/O subsystem must be a heavy degrading factor. Figure 4 also suggests that with a better configuration the VAX might outperform the 11/70 at all loads. One would clearly need to add more main memory to increase the threshold of load that saturates the system and also disk drives to increase parallelism in I/O service.

The results observed for this task are similar to those obtained for the C compilation and the command man man. The only difference with the pattern observed here is that in the latter task, when ranked using our external characterizations of load, the VAX ranks third, while when using Real Processes it ranks second. A possible reason

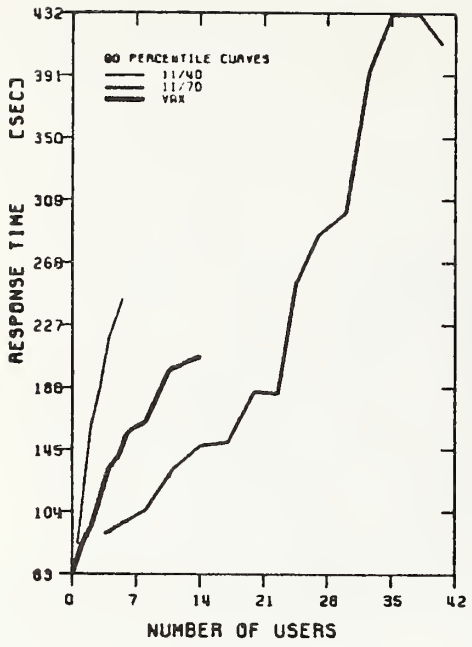


Figure 2. Response time of Script versus Number of Users.

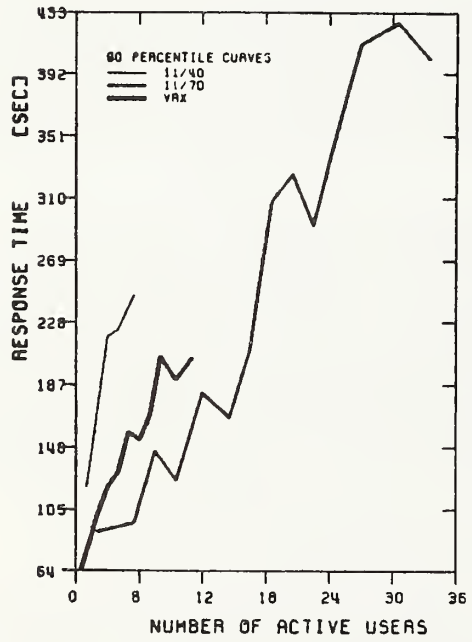


Figure 3. Response time of Script versus Number of Active Users.

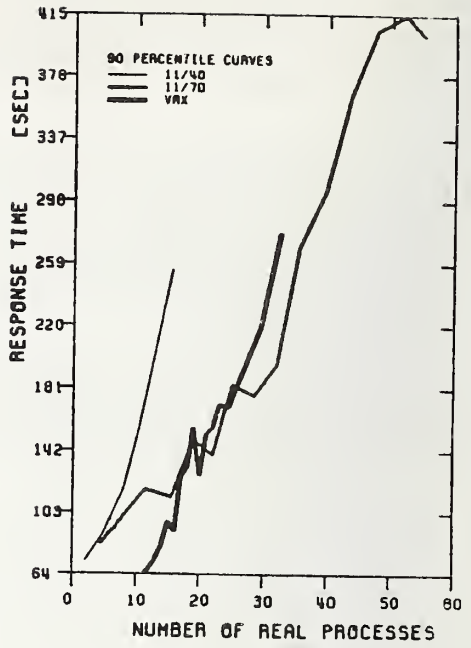


Figure 4. Response time of Script versus Number of Real Processes.

for this was given at the beginning of section 3.

4. Effects of Upgrading Changes

All diagrams presented in this section are 90th-percentile curves, and the size of samples has been chosen, as before, so as to exclude outliers. We have analyzed these changes using the characterization of load which is best understood: the number of (logged in) users.

4.1 Adding a Cache Memory to the 11/40

Perhaps the most spectacular use of a histogram of all measurements taken on a given system for a given task is that of witness to a basic configuration change. The phenomenon of multiple peaks in performance indices for a given task such as user and system time is almost certainly associated with a basic configuration change. In effect, the addition of the cache memory to the 11/40 was responsible for shifting to the left the peaks in each task's histogram.

Most remarkable is the case of the CPU-bound job, where the peak in user time went down from 23.0 seconds to 19.6 seconds, the mean user time decreased from 23.13 seconds to 19.71 seconds, and the standard deviation of the user time sample decreased from 0.25

to 0.20. The system time also improved dramatically for this task: its mean decreased from 0.68 seconds to 0.33 seconds, and its standard deviation decreased from 0.88 to 0.36; figure 5 depicts the shifting of peaks in the case of user time for the CPU-bound job.

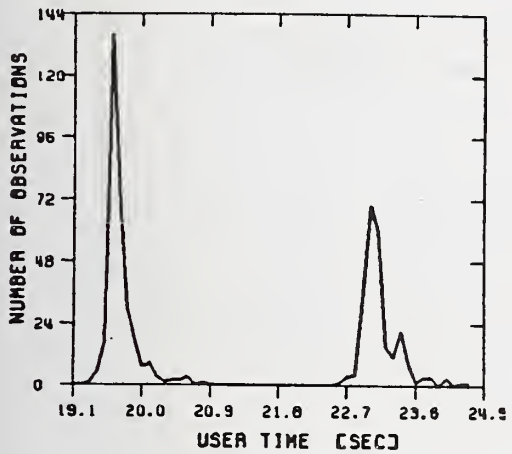


Figure 5. Histogram of User Time for the CPU-bound Task in the 11/40.

Figure 6 shows the remarkable effect of the change on the CPU-bound task, while in figure 7 we may appreciate the very positive effect of the change on the Script task. The variation observed in figure 7 is representative of what happened with each of the other two tasks. The curves were obtained using 239 measurements without cache memory and 303 measurements with the cache memory. A minimun sample size of 60 was used for this plot.

Because of figure 7 it can certainly be stated that both responsiveness and throughput are enhanced when a cache memory is added. It is only a matter of cost-benefit analysis to determine its size and type.

4.2 Adding Main Memory to the 11/70

It is unfortunate that we only had 175 measurements before this change was made and that they were obtained at the beginning of our data gathering period. In our university environment, the systems are not "pushed" by the users as hard as they are latter in the quarter, when the users have become more sophisticated. Thus, for this change, our curves actually show the effect of two sources of variation: addition of 1M byte of main memory and increase of the load. The 830 measurements we had after the new memory was installed would have allowed us to apply statistical techniques, such as the analysis of variance, that would have provided us with the relative weighs of the two sources of variation. However, this kind of analysis was precluded by the minimal amount of samples we could gather before the addition of memory.

Thus figure 8 must be interpreted carefully. In it the 0.3M byte curve may be too

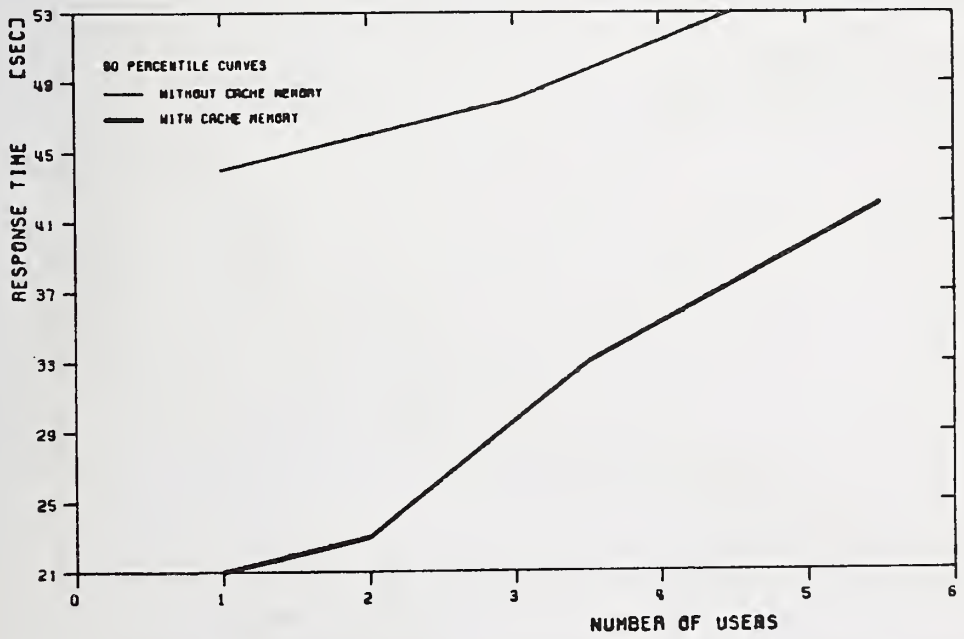


Figure 6. Response Time of CPU-bound job versus Users.

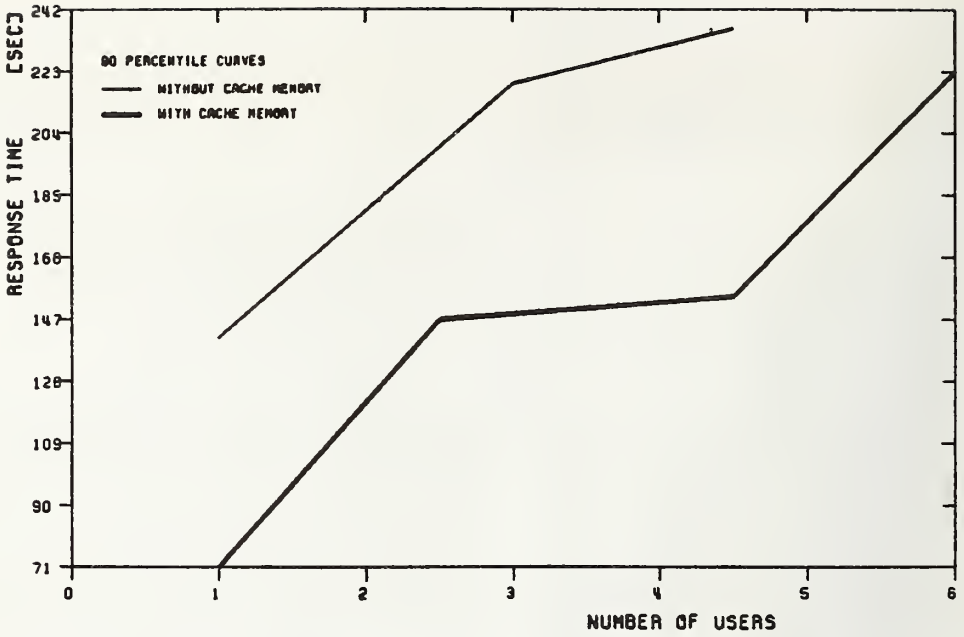


Figure 7. Response Time of Script versus Number of Users in the 11/40

optimistic, given that the users were not performing tasks as resource consumming as those in the 1.3M byte curve. Nevertheless, the response time is much lower in the 1.3M byte configuration. It is also seen that the saturation point of the system occurs at a lower level of load with a smaller amount of main memory.

The addition of main memory certainly increases the amount of multiprogramming the system handles before saturation, and this has a very positive effect on the response time of trivial tasks. Unfortunately, this effect is not measured by our script and so we can not report on it directly, although it influences the response time of the Script task.

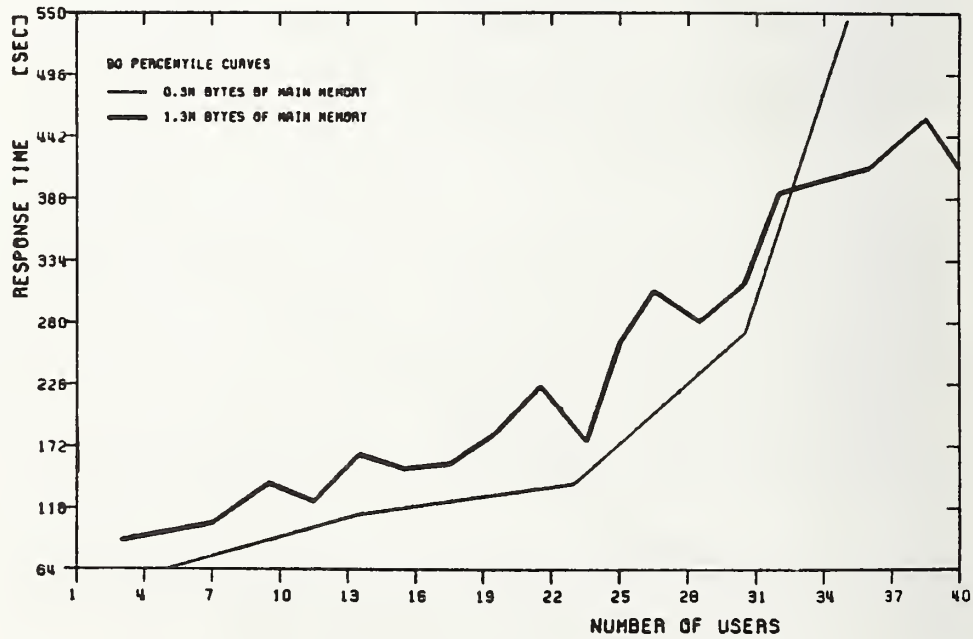


Figure 8. Response Time of Script versus Number of Users in the 11/70

4.3 Adding a Second Disk Drive to the VAX

The addition of a CDC 9762 disk drive with a CDC 9400 controller and 80M byte disk packs to the VAX configuration was a mixed blessing. It proved to yield an overall improvement in the responsiveness on the system, as seen in figure 10, but for one type of task it proved to be disadvantageous.

CPU-bound jobs took longer to finish with the two disk configuration (see figure 9).

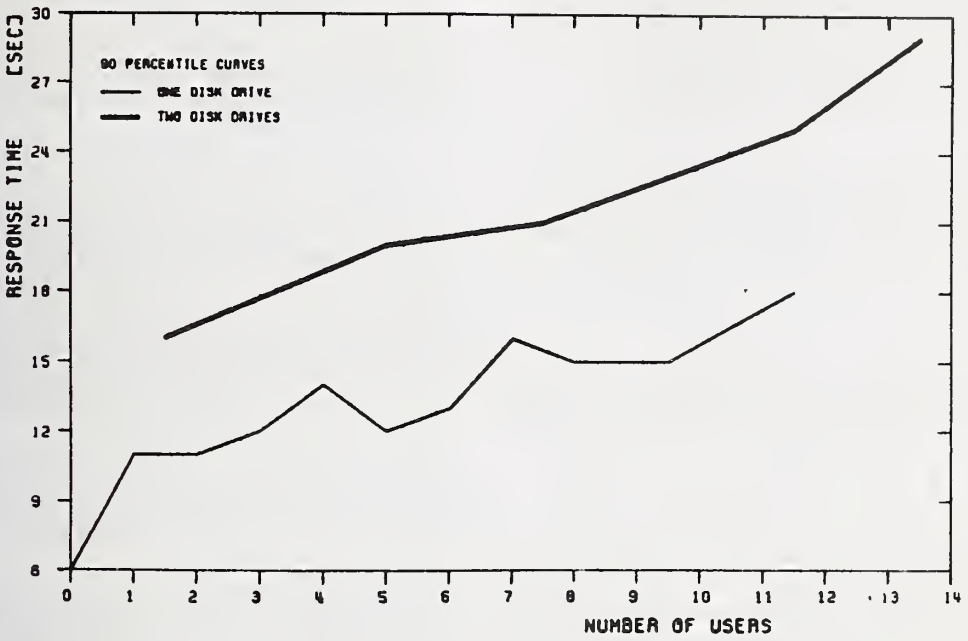


Figure 9. Response Time of CPU-bound Job versus Number of Users in the VAX

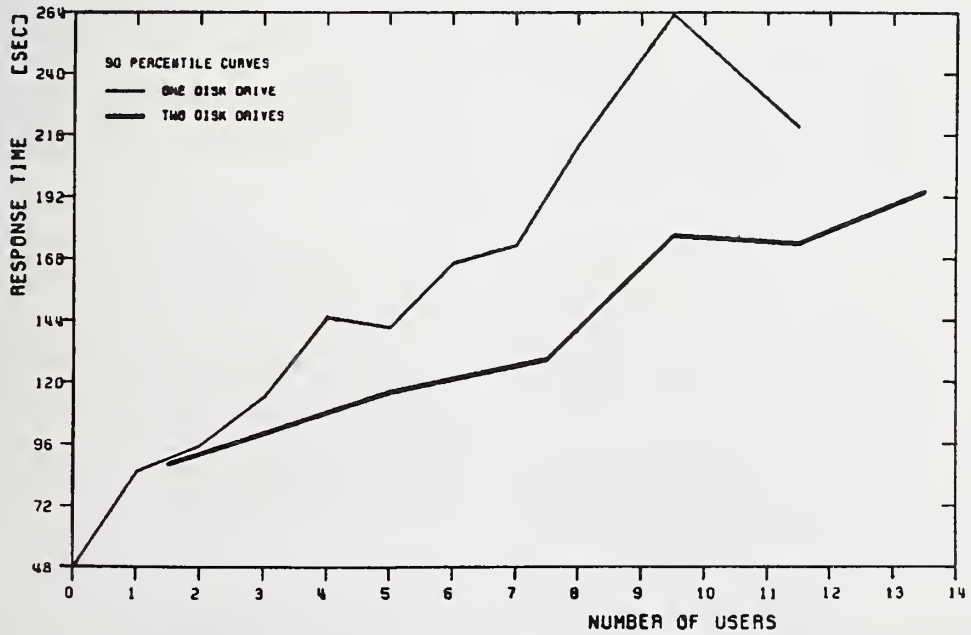


Figure 10. Response Time of Script versus Number of Users in the VAX

When analyzing the sample of user time for this task for the one-disk configuration (934 measurements) and the two-disk configuration (433 measurements), we see that mean user time remained essentially unaltered; 5.66 seconds with one disk and 5.69 seconds with two disks. Moreover the standard deviation (0.21) of the user time sample did not change. However, mean system time increased from 0.29 seconds in the one-disk configuration to 0.41 seconds in the two-disk configuration.

We believe this increase in system time to be directly related to the cause of the longer response time observed: it probably is an indication that now jobs get interrupted more often due to completed I/O requests and thus sent to wait queues more times than before.

As was expected, all tasks with I/O activity which made advantageous use of the new file configuration run quite faster [8]. The file system was reconfigured in such a way that temporary files, like those created by compilers, editors, text formatters, and several other utility programs were placed on the disk not containing user files. It could also be observed that the rate of growth of response time decreased for tasks of this type as well as for the Script task. Thus, the two-disk configuration is best suited for supporting more users with less risk of system saturation.

5. Conclusion

The increasing number of different systems on which the same timesharing operating system, UNIX, runs makes it possible to study the influence of the equipment on a system's global performance. In this paper we have studied three different systems, compared their performance and analyzed the effects of different upgrading changes.

Lack of portable tools that would enable us to load the systems in a controlled way led us to design an experiment which probed each system periodically while the natural work load was being executed. This method has many drawbacks when comparing installations, but the consistency of our curves shows that a study can be based on it. We also believe the method not to be too expensive in terms of resources used: in our case, it requires about 3% of the total CPU cycles. This amount does depend on the system. Moreover a great advantage of this method is its total portability. New systems can easily be added to our study.

Our single-variable characterizations of load forces one to observe several work load variables simultaneously in order to draw solid conclusions. This is specially critical when the natural work load is non-stationary.

No one system presented itself as a clear best although it is obvious that a configuration having a PDP 11/40 is outperformed by the other two. Moreover, certain additions, such as a cache memory, prove to be almost indispensable.

Having observed that the change from the PDP 11/40's to PDP 11/70's was very beneficial (faster CPU, better responsiveness, larger address space), the question was whether the same conclusion could be reached for the change from the PDP 11/70 to the VAX 11/780. The main obstacle against drawing strong conclusions from our study was the marked difference in each system's configuration. Unfortunately, the VAX 11/780 did not have as much main memory as the PDP 11/70 and it only has one disk. Moreover, the size of UNIX had grown in the VAX implementation, thus leaving proportionally less core space for user processes.

The higher speed of the VAX CPU was confirmed by our observations. Better yet, when a second disk was added to the VAX configuration, it performed tasks such as C compilations faster than before, ranking now very close to the 11/70. The performance that such tasks now achieved makes us believe that each system is at least as good as the other. At this point it has to be enphasized that the C compiler for the VAX is a fairly portable one while the one the PDP 11/70 has was essentially custom made. This is of course an advantage for the VAX (but probably a performance disadvantage).

6. Acknowledgements

I am very grateful for the amount of help I received from my professors, colleages and staff of the Department while working on this project. It exceeded all my expectations. In particular I would like to thank Domenico Ferrari for encouraging me to work in this area and for his guidance, Robert Fabry for introducing me to his earlier work at Berkeley, and the PROGRES group for their helpful remarks.

7. References

[1] Ritchie, D.M. and Thompson, K., "The UNIX Timesharing System". CACM 17 (7) (July 1974), pp. 365-375.

[2] Joy, William, "An Introduction to the C Shell" Computer Science Division, Department of EECS, University of California, Berkeley. December 1978.

[3] Ferrari, Domenico, "Computer Systems Performance Evaluation", Prentice-Hall 1978.

[4] Ferrari, Domenico, "Characterizing a Workload for the Comparison of Interactive Systems", Proceedings 1979 NCC, pp. 789-796.

[5] UNIX Programmer's Manual
Bell Telephone Laboratories Incorporated, Holmdel, New Jersey. 7th Edition, December 1978.

[6] Kernigan, Brian and Ritchie, Dennis, "The C programming Language", Prentice-Hall Software Series, 1977.

[7] Karush, A.D., "The Benchmarking Method Applied to Time-Sharing Systems". Rept. SP-3347, System's Development Corporation. Santa Monica, CA. August 1969.

[8] Cabrera, Luis F., "Benchmarking UNIX: A Comparative Study". MS Project Report, University of California, Berkeley, November 1979. The text is available as the ERL Memorandum No UCB/ERL M79/77.

I/O Performance Measurement on CRAY-1 and CDC 7600 Computers

Ingrid Y. Bucher
Ann H. Hayes

Computer Science and Services Division
Los Alamos Scientific Laboratory
Los Alamos, NM 87544

Disk I/O transfer rates and overhead CPU times were measured as functions of buffer size and number of logically independent I/O channels for several operating systems and 16 I/O routines on the Cray-1 and CDC 7600 computers. By parameterizing the codes for a variable number of channels, buffer sizes, and words transmitted, the effect of these variables is observed for buffered, nonbuffered, and random-access I/O transmissions. To measure CPU-overlapped performance, I/O was performed concurrently with a pretimed compute loop. Rates, sector overhead, and CPU transmission speeds were calculated upon completion of I/O. Effects of memory blocking due to vector operations were observed. Methods and results are presented in this paper.

Key words: I/O performance; CPU transfer rates; overhead CPU; compute-and-test loop.

1. Introduction

Due to the high computational speeds of large scientific computers, I/O rates may be the factor limiting execution speeds of certain application programs. It is desirable, therefore, to

- provide users with criteria for the selection of I/O procedures most suitable for their programs;
- determine how well existing operating systems and I/O routines approach the maximum capabilities of the hardware; and
- learn where improvements might be possible.

For these reasons, a study was undertaken at the Los Alamos Scientific Laboratory (LASL) to investigate I/O performance on the Cray-1 and CDC 7600 computers.

Disk I/O rates, as well as the times during which the CPU was unavailable for computing while I/O was being performed, were measured. The measurements were taken as functions of buffer size and the number of logically independent I/O channels used in performing the operations. The tests were executed on Cray-1 and CDC 7600 computers at the following installations: LASL, Lawrence Livermore Laboratory (LLL), and Cray Research Incorporated (CRI). Sections 2 and 3 describe the methods of measurement and analysis of the resulting data. In Sec. 4, results obtained for various operating systems and I/O routines are discussed.

2. Measurements

The following processes were measured: unformatted reading or writing from and to disk; reading or writing with concurrent computing; and concurrent reading, writing, and computing.

The test programs measured two quantities as functions of buffer size B and the number of logically independent I/O channels N used to perform the I/O operations: transfer rates R(B,N) in words per second per channel, and overhead CPU times per sector $T_{OH}(B,N)$. The latter are defined as the times when the CPU is unavailable for computing while a sector (512 words) of data is being transferred to or from disk. Buffer sizes were multiples of 512 computer words, except for BUFFER IN/ BUFFER OUT operations for which program buffers were multiples of 511 words. Four logically independent channels were available at all Cray-1 systems at LASL and LLL. The CDC 7600s were equipped with three logically independent channels at LASL, two at LLL.

Each test program reads and/or writes N one-million word files by repeatedly filling (emptying) a program buffer of preset length B. The process is repeated for each buffer size. For several routines, I/O can be performed either sequentially or by choosing disk addresses at random. Rates were measured for single channels in two ways;

(1) The non-overlapped (or synchronous) part of the test called for reading (writing) a buffer and waiting for I/O completion, then repeating this sequence until the entire file was read (written).

(2) The overlapped (or asynchronous) part executed a pretimed compute-and-test loop while waiting for I/O completion.

The duration of the compute-and test loops had to be short enough not to slow down I/O operations. For several cases, overlapped rates exceeded the non-overlapped rates considerably, indicating that the system's frequency of testing for I/O completion was not high enough or that the time required to return from the interrupt was too long.

The tests were run on dedicated system time, with other users and system diagnostics blocked out. Great care was exercised to select timing routines that measured wall clock time and that had high enough precision. Experience showed that this was a non-trivial problem. Rates R(B,N) were obtained by dividing the total number of words transferred W (an integer multiple of 512 * B approximately equal to 10^6N) by the measured time T_t required for the transfer and the number of channels N:

$$R(B,N) = \frac{W(B,N)}{T_t(B,n) * N} \quad . \tag{1}$$

Overhead CPU times $T_{OH}(B,N)$ were measured in the following way: After initiating the transfer of 512 * B words on each of N channels, a compute-and-test loop was started that performed a series of multiplications and then tested each channel for I/O completion. If I/O was complete on any channel, it was reinitiated immediately before the compute-and-test loop resumed. The process was repeated until all files were transferred. The number of times the compute-and-test loop was executed during the complete file transfer N_{loop} was measured as well as the duration of one compute-and-test loop t_{loop}. The overhead CPU time per transfer of 512 words is given by

$$T_{OH}(B,N) = \frac{T_t(B,N) - N_{loop} * t_{loop}}{W(B,N)/512} \tag{2}$$

In many cases, the numerator in Eq. 2 is a small difference of two large numbers, i.e., the overhead times are small compared to the transfer times of 512 words. For these cases the resulting values of T_{OH} are extremely sensitive to even small errors in any one of the numerator terms, which therefore had to be measured with high precision. The total transfer times $T_t(B,N)$ were of the order of several seconds and could easily be measured with high accuracy (better than 10^{-6}) by calls to the cycle counter or microsecond clock. The integer loop count N_{loop} is free of error. However, the measurement of the duration of the compute-and-test loop t_{loop} (which ranged from 20 μs to 1000 μs) required special attention. Three calls to the timing routine were made to accurately time and deduct the duration of the timing call itself. All I/O status tests were included in the timed loop, and parameters were set in such a way that the branches of the loop transferred to were the same and that I/O status checking was done in the same way as when I/O was busy. The essential section of code that includes the timing of the compute-and-test loop for each number of channels and the timing of the file transfer overlapped by the compute-and-test loop is given in Appendix A. Compute-and-test loops were timed several times, and occasional skewed values caused by system disturbances were discarded. The remaining values agreed to better than 1 μs. To assure that no systematic errors were overlooked, tests were run with compute-and-test loops of several lengths differing by factors of about 2 for each routine. No systematic deviations were found.

To prevent certain hardware problems on the CDC 7600 caused by accessing the same memory location too often, the calculations were performed on subscripted variables. These loops automatically vectorized on the Cray-1 and were subsequently replaced by scalar loops to obtain longer compute loops. Overhead CPU results obtained from tests run with vector compute-and-test loops on the Cray-1 showed considerably higher overhead than those using scalar arithmetic, due to memory lockout during a vector operation. This indicates the sensitivity of the operations involved. Some tests were run with compute-and-test loops that required no memory access at all. The results were identical to those with loops performing only scalar operations.

Measured values of transfer rates and overhead CPU times were subject to some random fluctuations, which were especially pronounced for ALAMOS, a LASL-produced operating system for the Cray-1. Part of these variations is due to fluctuations of the rotation rate of the disks that is nominally $\pm$ 2%.

3. Analysis of Data

The disk units attached to the Cray-1 (DD-19) and the CDC 7600 (819) are very similar. Each unit consists of 40 recording surfaces subdivided into 411 cylinders for recording data. A read-and-write head is associated with each recording surface. The 40 heads are divided into 10 head groups of 4 heads each. The four heads of a group are used in tandem to transfer data to and from disk in such a way that parts of a single computer word will reside on four recording surfaces. During one disk revolution of 1/60 seconds, a head group will pass over and therefore be able to read or write one track of data. For the Cray-1, a track contains 18 sectors; for the CDC 7600, a track contains 20 sectors of 512 computer words each. Switching from head group to head group is

accomplished electronically and rapidly; therefore, a maximum of 10 tracks, constituting one cylinder, can be transferred to or from disk without mechanically repositioning the read-and-write heads or missing a disk revolution. Repositioning of the heads is a mechanical and therefore slow process. For sequential access, one disk revolution is missed at each cylinder boundary. This results in a maximum transfer rate of R_{max} of one cylinder per 11 disk revolutions for sequential access of large files extending over many cylinders. For the Cray-1

$$R_{max,Cray} = \frac{180*512 \text{ words}}{11*1/60 \text{ second}} = 502.7 \text{ kword/s}; \quad (3)$$

for the CDC 7600

$$R_{max,7600} = \frac{200*512 \text{ words}}{11*1/60 \text{ second}} = 558.5 \text{ kword/s}. \quad (4)$$

In practice, the maximum transfer rates will not be reached if additional disk revolutions are missed or if the density of data written on the disk is less than optimal. If B sectors are transferred to or from disk per I/O call, M disk revolutions are missed per call in addition to those at cylinder boundaries, and S sectors are transferred per revolution, then the number of disk revolutions N_B needed to transfer B sectors is given by

$$N_B = B/S + B/(10*S) + M \quad (5)$$

The number of disk revolutions per sector $N_{revs}(B) = N_B/B$ is

$$N_{revs}(B) = 1.1/S + M/B \quad , \quad (6)$$

and the associated transfer rate is

$$R(B) = \frac{512 \text{ words}}{N_{revs}(B)*1/60 \text{ second}} = \frac{512*60 \text{ words}}{(1.1/S + M/B) \text{ seconds}} \quad , \quad (7)$$

If more than one logically independent channel is employed for data transfer, Eqs. 3-7 should be applicable to each channel independently.

Equation 6 indicates the number of disk revolutions $N_{revs}(B)$ per sector should be a linear function of 1/B, the reciprocal of the buffer size. To analyze the experimental data, the measured values of $N_{revs}(B)$ were plotted for each I/O routine as a function of 1/B. Most plots were indeed linear, and the constants S and M could be determined from the zero intercept and the slope of each line. As an example, Fig. 1 represents data measured by the routines BUFFER IN/ BUFFER OUT on the Cray-1. The number of disk revolutions per sector for BUFFER OUT as plotted as a function of the reciprocal of the buffer size is a straight line, the slope of which corresponds to M = 3; that is, three disk revolutions are missed per I/O call. The zero intercept corresponds to a transfer of 18 sectors per revolution. The data for the BUFFER IN operation can be fitted by two straight lines, with the same zero intercept. For $B \geq 18$ sectors, two disk revolutions are missed; for $B \geq 16$ sectors, only one disk revolution is missed per I/O call. These results are interesting but not characteristic of most I/O routines on the Cray-1 and CDC 7600. The most frequently encountered sets of coefficients were M = 0 and M = 1 (one disk revolution missed per I/O call) and S = 18 for the Cray-1 or S = 20 for the CDC 7600 (maximum number of sectors per disk revolution).

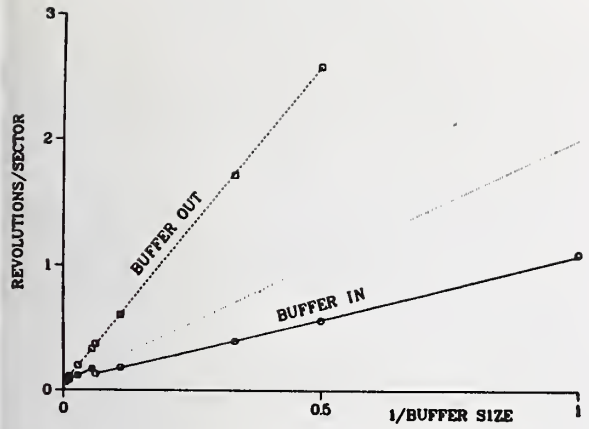


FIGURE 1. Disk revolutions per sector for BUFFER IN/OUT as a function of the reciprocal of the buffer size.

An example of results obtained on the CDC 7600 is shown in Fig. 2. Routines performing random-access I/O transmission were used for this test. The solid line represents data for sequential access and corresponds to one disk revolution missed per I/O call and a transfer of 20 sectors per disk revolution.

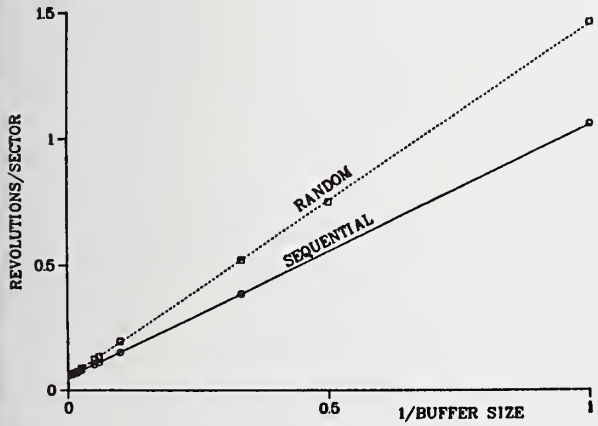


FIGURE 2. Revolutions per sector for transfer by WDISK/RDISK (7600) as a function of the reciprocal of the buffer size.

Using a random number generator to determine the disk addresses at which to start transmissions, the test was performed a second time using the same routines. Data for these results are also fitted by a straight (dotted) line, with the same zero intercept, corresponding to 20 sectors transferred per disk revolution. Due to more frequent seek operations, 1.4 disk revolutions are missed per I/O call.

To interpret results obtained for overhead CPU times, it is reasonable to assume that the overhead CPU time per sector $T_{OH}(B,N)$ consists of two contributions, the CPU time T_{trans} required to actually transfer the data, and 1/B times the CPU time T_{call} needed to initiate and complete the system call for I/O:

$$T_{OH}(B,N) = T_{trans} + 1/B*T_{call} \quad . \qquad (8)$$

Experimental results indicate that the assumptions leading to Eq. 8 are valid in most cases. Plots of $T_{OH}(B,N)$ versus 1/B are straight lines with a zero intercept of T_{trans} and a slope of T_{call}.

Figure 3 shows data obtained for the Cray-1 using random-access I/O routine RDISK. The lower curve represents data obtained using a scalar compute-and-test loop, with $T_{call} = 660 \pm 10$ μs and $T_{trans} = 6 \pm 2$ μs. The tests were also run using a vectorizable compute-and-test loop. Because of memory lockouts during the vector calculations, the CPU overhead times are slightly higher, as observed in the upper curve.

Data obtained from CDC 7600 tests is shown in Fig. 4. Both sequential and random tests were performed as previously described. The zero intercept is the same for both tests; the overhead CPU times per sector are slightly larger for the random test due to more frequent seeking.

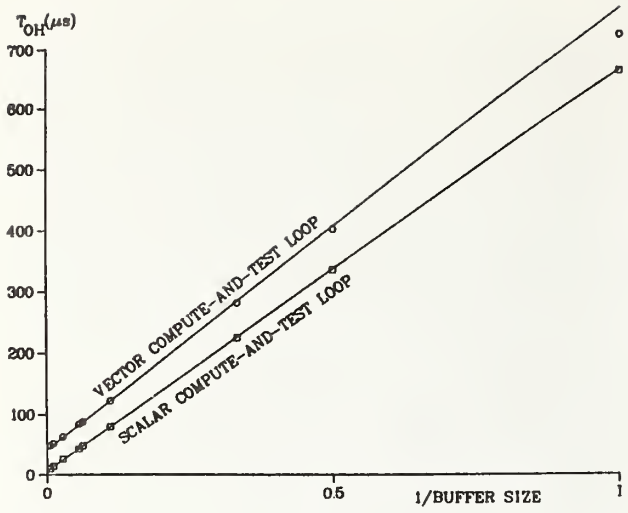


FIGURE 3. Overhead CPU times per sector for RDISK (CFTLIB) as a function of the reciprocal of the buffer size.

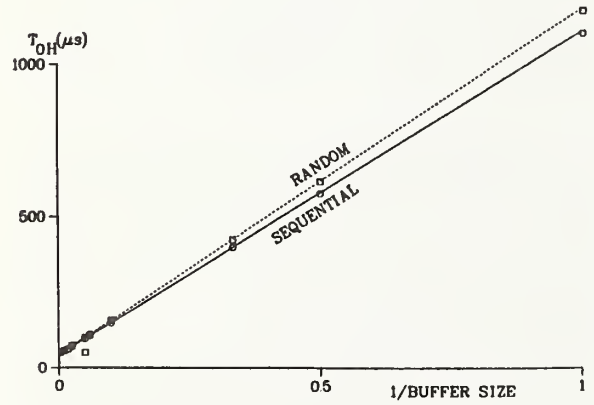


FIGURE 4. Overhead CPU times per sector for WDISK/RDISK (7600) as a function of the reciprocal of the buffer size.

4. Notes on Results

Measurements made on both the Cray-1 and the CDC 7600 employed a wide variety of operating systems, libraries, and I/O routines. Both sequential and random writing/reading, buffered and nonbuffered I/O, Fortran versus assembly language (CAL), and overlapped/non-overlapped tests were run and analyzed.

Tests performed on the Cray-1 used program buffer sizes of 1, 2, 3, 9, 18, 36, 90, and 180 sectors to ensure that any peculiarities occurring at track and cylinder boundaries would be observed.

In almost all cases, overhead CPU times can be represented by Eq. 8. The random access routines IZDKIN/IZDKOUT, RDABS/WRABS, and RDISK/WDISK are all very efficient with very small overhead CPU times. In some cases, the implementation of a very short compute-and-test loop (< 30 μs) ensured that no disk revolutions were missed, even at buffer size B = 1.

Buffered operations were measured using BUFFER IN/BUFFER OUT and BINARY READ/WRITE statements. Some of the tests were run with system buffer sizes of 1/4, 1/2, 1, and 2 times that of the program buffer size to determine the effect on rates. It was apparent that in all cases where this approach was taken, the resultant rates were a direct function of the system buffer size.

Program buffer sizes for tests run on the CDC 7600 were set at 1, 2, 3, 10, 20, 40, 60, 100, and 200 sectors. The results of these tests were generally analogous to those seen on the Cray-1. Overhead CPU times were somewhat larger, with the smallest exhibited by RDISK/WDISK routines with $T_{call} = 1050 \pm 50$ μs and $T_{trans} = 46 \pm 3$ μs. Again, on random access routines, transfer rates are consistent with a transfer of 20 sectors per disk revolution and one disk revolution missed per I/O call. Tests that generated random-disk addresses by a random-number generator showed an increase to 1.4 in the number of disk revolutions missed per I/O system call, due to more frequent and longer seek operations.

For BUFFER IN/BUFFER OUT tests, as with the Cray-1, the rates were solely dependent on the size of the system buffer used in the

transfer. For read requests, two revolutions were missed per processing of a system buffer, except for the minimum system buffer size of B = 2 sectors, for which four disk revolutions were missed.

The situation for write operations was more pronounced: four disk revolutions are missed per system buffer processing for B > 4. For B = 2, it was found that nine disk revolutions are missed per call. This is caused by excessively high overhead times associated with this buffer size.

The raw data from these tests can be obtained from the Computer Science and Services Division's Research and Applications Group at the Los Alamos Scientific Laboratory.

5. Summary

Characteristics of all routines tested are summarized in Table 1. The following general observations can be made.

With the exception of some BUFFER IN/BUFFER OUT routines, the maximum number of sectors is transferred per disk revolution: 18 for the Cray-1, 20 for the CDC 7600.

For the Cray-1, maximum transfer rates of 503 kword/s for both overlapped and non-overlapped reads and writes were achieved on the COS operating system. The overhead CPU time for an I/O system call T_{call} = 480 μs is smallest among the operating systems tested; the CPU time required for the transfer of one sector T_{trans} = 42 μs is slightly higher than that observed on the CTSS operating system.

The CTSS operating system has library routines that achieve maximum transfer rates for both overlapped and non-overlapped writes. The reads are more sensitive to proper timing of I/O requests than the writes. Non-overlapped reads always miss one disk revolution per call. For overlapped I/O, using the most efficient routines available, maximum transfer rates can be achieved only by testing I/O completion at < 100-μs intervals. Observed overhead CPU times are T_{call} = 650 μs for the I/O system call and T_{trans} = 7 μs for a one-sector transfer. On a single channel, these are short enough not to degrade I/O performance; for multichannel tests and small buffer sizes, some rate degradation will occur.

At least one disk revolution is missed per I/O call for all routines on the ALAMOS operating system. This is attributable to the high overhead CPU time of about 4000 μs for the I/O call. This is considerably higher than the 926 μs required for one sector to pass under the read/write head.

CDC 7600/LTSS operating system routines also miss at least one disk revolution per I/O call. The minimum overhead CPU times for initiating an I/O call are T_{call} = 1050 μs for RDISK/WDISK routines, just slightly longer than the 833 μs required for one sector to pass under the read/write head. The minimum measured transfer CPU time per sector is T_{trans} = 46 μs.

It is apparent that maximum I/O rates can be achieved only if one chooses I/O routines carefully and performs frequent enough testing for I/O completion on small buffer sizes.

Table 1

Performance Characteristics of Cray-1 and CDC 7600 I/O Routines

Machine	Operating System	Routine	Library	Sectors per disk revolution	Disk revolutions missed per I/O call	T_{call} (μs)	T_{trans} (μs/sector)
CRAY-1	CTSS	IZDKIN	BASELIB	18	0 or 1[a]	630	6
		IZDKOUT		18	0	630	10
		RDISK	CFTLIB	18	0 or 1[a]	660	6
		WDISK		18	0	660	6
		RDABS	FORTLIB	18	0 or 1[b]	650	7
		WRABS		18	0	650	9
		BUFFER IN	CFTLIB	18	1 or 2	1580	50
		BUFFER OUT		18	3	14000	150
		BUFFER IN	FORTLIB	18	1	---	---
		BUFFER OUT		18	0 or 1	---	---
		READ	CFTLIB		0 or 1	730	10
		WRITE			0	730	10
		READ	FORTLIB	18	$\geq$ 1	---	---
		WRITE		18	2	---	---
	ALAMOS	System calls (read)		18	1[c]	4050[c]	155[c]
		System calls (write)		18	1[c]	4050[c]	155[c]
		BUFFER IN		9	2	---	---
		BUFFER OUT		9	1	---	---
	COS	System calls (read)		18	0	480	42
		System calls (write)		18	0	480	42
CDC 7600	LTSS	IZDKIN	BASELIBF	20	1	1600	50
		IZDKOUT		20	1	1700	50
		RDISK	FTNLIB	20	1	1050	46
		WDISK		20	1	1050	46
		BUFFER IN	FTNLIB	20	2	~1750[d]	~100[d]
		BUFFER OUT		20	4	*[d]	*[d]
		READ	FTNLIB	20	1	*[d]	*[d]
		WRITE		20	1	*[d]	*[d]
		BUFFER IN	ORDERLIB	20	5	---	---
		BUFFER OUT		20	2	---	---

a) Depending on length of compute-and-test loop for overlapped I/O. Always 1 for non-overlapped I/O.
b) 0 for B$\leq$24, 1 for B$\geq$36.
c) for B$\geq$9
d) Equation 3.5 does not apply.

Appendix A

Example of Code for Timing of the Compute-and-Test Loop and File Transfer

```
C     SET NUMBER OF CHANNELS
      DO 1000 NCHAN=1,MAXCH
            ICOMP=0
            DO 110 N=1,NCHAN
                  NEXT(N)=0
                  DONE(N)=.TRUE.
110               CONTINUE
            CALL IDLE
C
C     TIME COMPUTE-AND-TEST LOOP FOR EACH NUMBER OF CHANNELS
           T0=TIMEF(DUM)
           T1=TIMEF(DUM)
C
115        CALL COMPUTE(NCOMP)
           ICOMP=ICOMP+1
C
           DO 160 N=1,NCHAN
                 IOC=N+4
                 IND=DSP(4,N)
                 IF((IFTBL(3,IND).GE.0).OR.DONE(N))GO TO 160
                 IF(NEXT(N).LT.MAX) GO TO 150
                 DONE(N)=.TRUE.
                 GO TO 160
150              CALL RDISK(IOC,BUFF,NWDW,NEXT(N)*NWDW)
                 NEXT(N)=NEXT(N)+1
160              CONTINUE
C
           ALDONE=.TRUE.
           DO 170 N=1,NCHAN
170              ALDONE=ALDONE.AND.DONE(N)
           IF (.NOT.ALDONE) GO TO 115
C
           T2=TIMEF(DUM)
           TLOOP(NCHAN)=1000.*(T2-2*T1+T0)
C
C
C     SET BUFFER SIZE, ARRAY ISECT CONTAINS BUFFERSIZES
            DO 900 NBF=1,9
                  NWDW=ISECT(NBF)*512
                  MAX=NWDS/NWDW
                  ICOMP=0
                  DO 310 N=1,NCHAN
                        NEXT(N)=0
                        DONE(N)=.FALSE.
310                     CONTINUE
                  CALL IDLE
C
C
C
C     GET TIMINGS FOR OVERLAPPED I/O
C
                 T0=TIMEF(DUM)
C
                 GO TO 355
340              CALL COMPUTE(NCOMP)
                 ICOMP=ICOMP+1
C
```

```
C     TEST EACH CHANNEL WHETHER I/O BUSY, IF NOT REINITIALIZE
355               DO 360 N=1,NCHAN
                        IOC=N+4
                        IND=DSP(4,N)
C                       IF CHANNEL N BUSY OR DONE GO TO 360
                        IF((IFTBL(3,IND).LT.0).OR.DONE(N)) GO TO 360
                        IF(NEXT(N).LT.MAX) GO TO 357
                        DONE(N)=.TRUE.
                        GO TO 360
357                     CALL RDISK(IOC,BUFF,NWDW,NEXT(N)*NWDW)
                        NEXT(N)=NEXT(N)+1
360                     CONTINUE
C
C     TEST FOR FINAL I/O COMPLETION
                  ALDONE=.TRUE.
                  DO 370 N=1,NCHAN
370                     ALDONE=ALDONE.AND.DONE(N)
                  IF (.NOT.ALDONE) GO TO 340
C
                  T1=TIMEF(DUM)
                  TIO(NCHAN,NBF)=1000.*(T1-T0)
C
900               CONTINUE
1000        CONTINUE
```

Forecasting Computer Processing Requirements: A Case Study

Ronald D. Tomberlin

Environmental Protection Agency
Research Triangle Park, NC 27711

This case study describes a recent experience the author had in updating a computer workload forecast. It presents an example of a workload forecast developed without regard for future update requirements and the subsequent problems encountered when trying to perform an update on the original study. The update methodology used is described and a series of recommendations is provided to help the reader avoid problems of the type experienced by the author.

Key words: Benchmark; capacity management; capacity planning; workload forecasting; workload update.

1. Introduction

Workload forecasting is inherent in all competitive procurements of computer equipment and services. There must be some measure to indicate the amount of processing required to support the future requirements of an organization. As a result, most organizations have been involved in at least one attempt to forecast their computer workload requirements. But, situations arise when neither the time or funding is available to support a detailed study. When this is the case, organizations tend to enhance, expand, or update their most recent forecast.

Often these updated forecasts do not employ the detailed analytical methods normally used in forecasts of this type. Intuitive reasoning replaces technical excellence. If results appear "reasonable" they are generally accepted. More often than not these results prove to be adequate. But, developing a process to determine these results can sometimes prove to be quite a challenge. Lessons are learned in these situations and, if shared with others, can make this process of updating a workload forecast easier. Such is the case with the presentation of this study.

This case study discusses a situation in which an organization contracted for a workload forecast and later decided that this forecast should be updated with more recent historical data. A gross estimate of processing growth was all that was required. Unfortunately, the original study was not designed to facilitate an update. As good as the original study was, it proved to be extremely difficult to update.

What follows is an outline of some of the major problems that had to be overcome in order to perform an update of this original study. This case study does not demonstrate how an initial workload forecast should be performed. The technical literature [1][4] provides many articles that offer guidance in this regard. This study deals only with one organization's attempt to update its outdated workload forecast.

There are two primary reasons for discussing this case study. First, the study is presented to acquaint the reader with the

types of problems an organization had to consider in updating a workload forecast. In this respect, only enough of the detail work is presented to explain how the problems were solved. Second, and most important, this case study presents a series of recommendations to aid those organizations who might be preparing an initial workload forecast or who might want to update a study at some future time. It is hoped these recommendations prove beneficial in aiding the development of workload forecasts with increased utility for other organizations.

2. Background Information

The organization requiring the update, in 1977 as today, was facing tremendous growth in the need for ADP support to meet the many demands of the user community. In such a dynamic environment it was difficult, at best, to meet the ADP requirements of the user community. At the time of the original study the organization was supporting six major offices at the national level and ten regions, each with varied requirements. Two major computer centers, one with Sperry Univac equipment, the other with IBM equipment, were providing the majority of the support to the user community.

The original workload forecast was an extremely detailed study performed in 1977. Historical data for the period October 1976 - April 1977 were used. In addition to historical data, management surveys were taken at various levels of the organization. These surveys were used to quantify management's position on future organizational growth. Based on this historical data and management survey information, the workload forecast for the organization was generated. Three different scenarios were developed to cover minimum, normal, and maximum periods of growth for the organization. Each scenario included processing projections for each computer center and each regional and national office for the

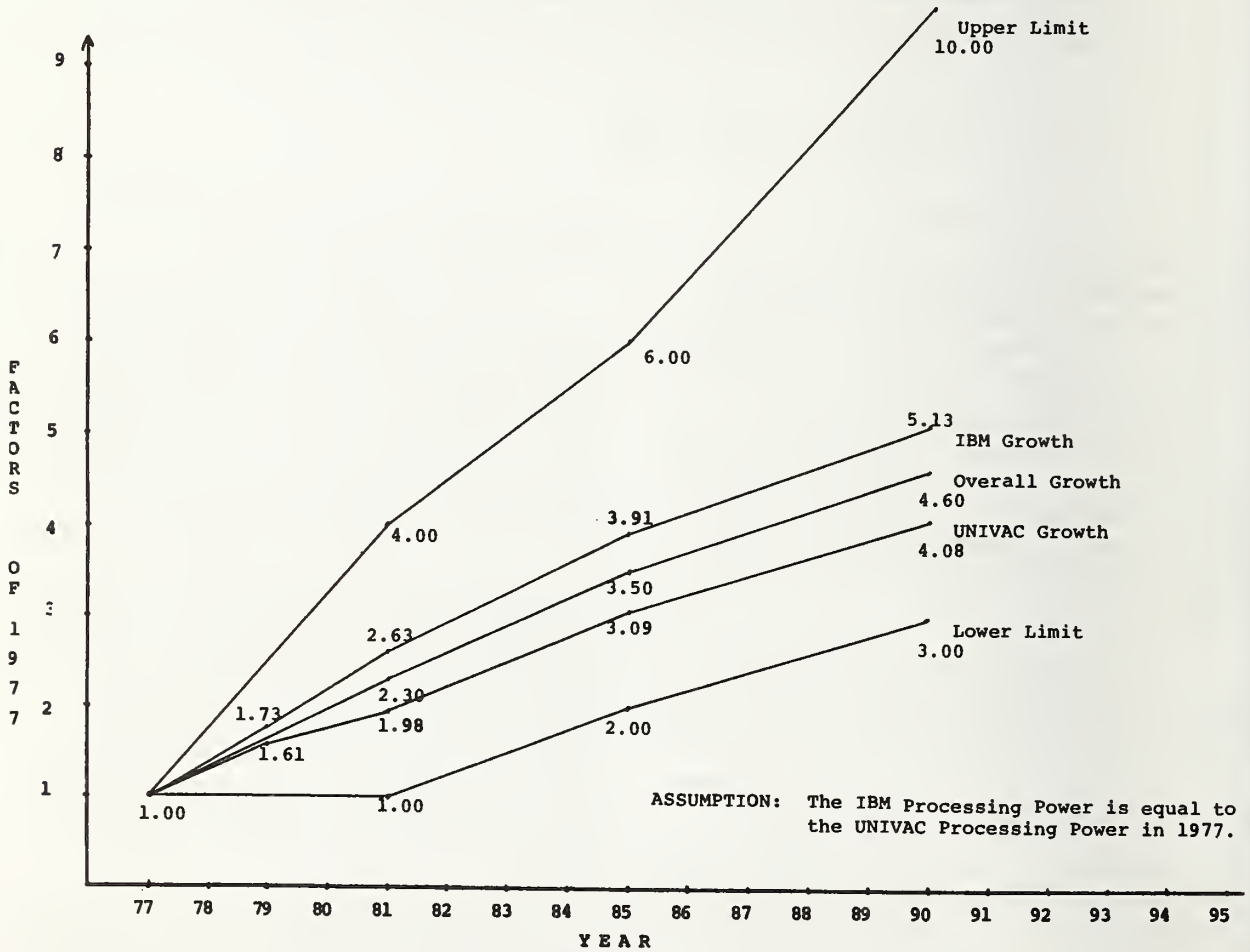


Figure 1. Original Study Processing Requirements

target years of 1979, 1981, 1985 and 1990.

3. The Update

Due to the large volume of data provided in the original study, data were extracted and consolidated in a graphical format to depict the agency's projected growth in processing for both computer centers. For ease of comparison, all data were "normalized" to a base year of 1977 (See figure 1).

Figure 1 depicts the future processing requirements of the organization as derived from the original study. The various lines are not representations of continuous functions. The points at each of the target years are the "normalized" processing requirements of the organization. Figure 1 is presented only to show the original chart to be updated. Other problems associated with figure 1 will be discussed later.

The basic idea of the update was to take 1979 historical processing data, develop a new graph, and compare it to the original information to determine the accuracy of the projection methodology of the original study. At first glance the problem seemed simple enough. Just accumulate the historical data, process it with the software package which generated the original results and prepare a new chart. But as it turned out, the actual update was not quite that simple. There were a few "minor problems" that precluded a simple solution to this problem.

4. Problems Encountered

The first problem associated with the update was to locate anyone who might provide insight as to how the graphs from the original study were prepared. In a period of only two and one half years, the expertise associated with the study had virtually disappeared.

A second discovery was the software package used in developing the original forecast was not available. No provision had been made to retain a copy of the support software for future use.

The original study used management survey information as an integral part of the original forecast. But due to the short period of time allocated for the update, new management surveys were not feasible. As a result, the original management information had to be used in the update.

The majority of workload forecasts, once completed, become the official basis for any future references to workload growth. This study was no exception. It was the basis for depicting processing growth for the organization. Once a study becomes established within an organization, it becomes almost impossible to replace it. It can be updated, enhanced or expanded, but never discredited or discarded.

During the period between the original study and the update there were major system changes at both computer centers. Also, several times during this period the accounting algorithms were changed to reflect the fact that both computer centers were non-profit organizations. Each time a change was made, the previous accounting data, relative to workload forecasting, was invalidated. Benchmarks or test programs could have been utilized to supply the linkage across configurations, but this was not done.

The majority of problems encountered were directly associated with gaining access to data needed to perform the update. Data were either fragmented, inaccessible or non-existent. Also, data were recorded in different units in 1977 and 1979 at one of the centers. There were changes in contractor support which made location of some required data impossible. The final problem was deciding how to combine the work at each of the centers to develop one measure of agency workload. In the past (see figure 1) these processing requirements had simply been added together but this was considered an unrealistic assumption for the update. In the update the Univac and IBM systems were treated separately.

Other problems were encountered in the course of this update. But these were the major problems that had to be resolved before the original processing forecast could be updated.

5. Update Methodology

Given the list of problems that had to be resolved, one might assume an update of the original workload forecasting study was, if not impossible, at least unrealistic. Nevertheless, the original study had to be updated, and the update had to be performed without benefit of the original software package and new management survey information. The gross nature of the update and an insufficient amount of time were the two primary reasons new management surveys were not taken.

Since the Univac system accounting data for 1977 and 1979 were reported in SUP hours, they were used for that portion of the update. SUP hours included central processor usage, input/output usage, and executive services. In 1977 the IBM system accounting data were reported in Resource Hours. Resource Hours included central processor, input/output, and memory usage. However, in 1979, after a change in contractor support, the data were reported in Computer Utilization Units (CUU's). A factor relating CUU's in 1979 to Resource Hours in 1977 had to be developed to provide the transition between the update and the original study. This will be duscussed later.

Because of the differences in accounting data, it was decided that an update would be performed for each of the computer centers. With the many assumptions made, no attempt was made to combine the information from the two centers to develop an overall organizational forecast.

The following seven step procedure was developed for the update. The procedure for each computer center was essentially the same until the final step. For the purpose of explaining the procedure, all references will be to the Univac system.

The first step was to determine the total resources projected for each of the ten regions and the six national offices for each of the three scenarios mentioned earlier. These totals were required for each of the target years of the original projection (1979, 1981, 1985, 1990). Each computer center was treated separately. All of this information was extracted from the original study.

Step two was to generate a "normalized" total processing value for each target year. Once tables for total processing for each of the three scenarios were created, the totals for each office and region for each target year were totaled to get the total for each of the target years. These values were then divided by the 1977 total to get a set of "normalized" values. These values are represented as scenarios 1, 2, and 3 in figures 2 and 3 which will appear later.

The third step required each of the values for each office and region for 1981, 1985, and 1990 in the table just created to be divided by its corresponding value for 1979. This created a table of multipliers representing the original management survey information for each scenario for the target years of 1981, 1985, and 1990. Step five of the procedure used this table of multipliers.

The fourth step was concerned with the actual data for 1979. The SUP data in the accounting report for calendar year 1979 were averaged for the same offices and regions mentioned earlier. Since overhead was just a lump sum figure in the reports and accounted for approximately forty to forty-five percent of the total it had to be included. It was proportionally spread across the figures for each of the offices and regions.

Step five involved taking the monthly average figure from the actual 1979 data for each office and region and multiplying it by the corresponding values from step three to develop the new projections based on the actual 1979 data. This procedure allowed the original management projections to be applied to the 1979 data.

In step six the new projected values for each scenario for each target year were totaled. These totals were also divided by the actual total for the base year of 1977 to "normalize" the data for comparison to the original study.

The final step in the procedure was simply to plot these "normalized" values for comparison purposes.

Figure 2 is a graph of the "normalized" values for the Univac System. All three scenarios from the original forecast are included. Only the scenario 1 values (normal) are plotted for the update.

There is a large difference in the actual and predicted values for 1979. Because of this difference a graph of the new values plus an accumulated error of ten percent per year was developed. Due to the numerous assumptions made in the updating procedure it was felt this second graph could be considered a "reasonable" upper bound for the new workload forecast. These new forecasted values are represented by dashed lines in figure 2.

The same procedure was followed for the IBM system. All references to the 1977 processing data were stated in Resource hours and references to the actual 1979 data were stated in Computer Utilization Units. In attempting to form a bridge between the two units, consideration was given to going back to the original data and developing an artificial algorithm. Time

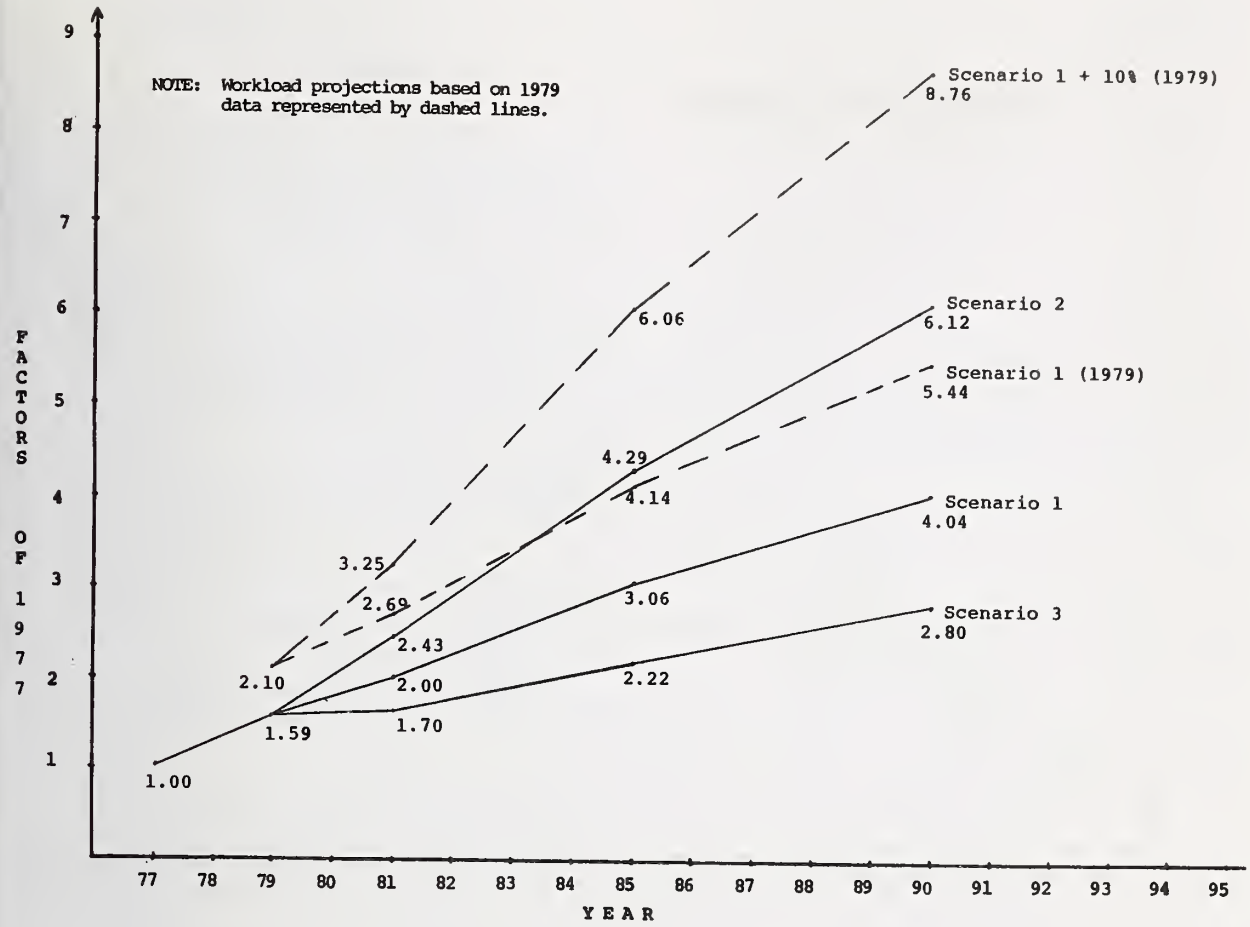


Figure 2. 1977 Univac Workload Projections

constraints made this option unworkable. As a last resort, CPU hours for the years 1977 and 1979 were used to develop a conversion factor that could be used to compare the original and updated data. Figure 3 is a graph of the "normalized" values for the IBM System after the conversion factor had been applied. The labeling on the graph is the same as for figure 2.

The data for figures 2 and 3 were provided in the same fourteen year format as the original study. However, workload forecast information projected beyond three years has historically been shown to provide very little utility.

At this point the obvious question must be asked. Just how good are the new projections of processing workload? This is a very difficult question to answer. If the original management surveys were good then it is probably acceptable. A reexamination of the procedure in 1981 is the only sure way to get a reasonable measure of its accuracy. At best, it can be no better than the original information. Time, as always, will be the ultimate judge.

6. Recommendations

Obviously, if certain things had been done, the update of the computer workload forecast could have been accomplished with less effort. The following list of recommendations has been developed as a result of this update. This list is by no means complete. It is only meant to serve as a starting point for those organizations considering a workload forecast.

Workload characterization is essential. Select those measures relevant to your ADP operation. For those uncertain about what measures to select, a review of the recent technical literature is recommended [2][3][5]. It is difficult to know where you are going or how fast you are getting there without knowing where you have been.

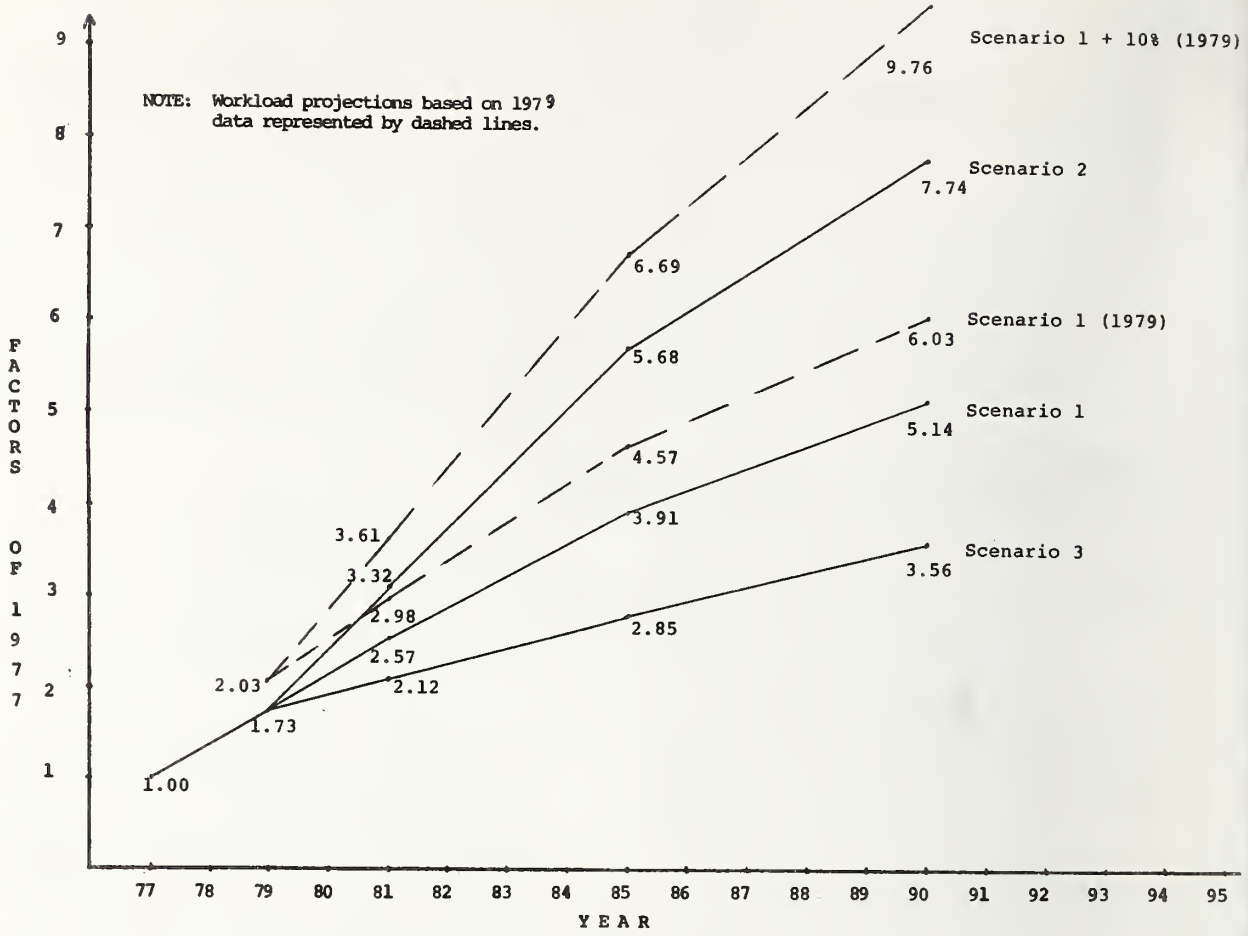


Figure 3. 1977 IBM Workload Projections

Second, do not consider the workload forecast as a single, one-time operation. Make it just one part of an on-going capacity management/capacity planning process for the organization.

Third, if the workload forecast is to be performed by a contractor, insist that all input data files and support software be made a part of contract deliverables. You will need these later.

Fourth, require detailed documentation on the methodology used in the forecast. Eventually someone will wonder how it was done.

Fifth, benchmarks, though expensive to develop and maintain, should be used if hardware and software systems change frequently. These changes invalidate accounting data relative to workload forecasting but a benchmark can provide the bridge between the old and new data.

Sixth, if benchmarks are not available, then a set of sample programs can be used. As with benchmarks, the linkage is essential if the data are to be used.

And finally, for those looking for a starting point for reviewing the technical literature, the IBM System's Journal, volume nineteen, number one, 1980, entitled Installation Management/Capacity Planning is recommended. It is an excellent source of information as well as a good source for additional references.

7. Acknowledgements

I gratefully acknowledge the assistance of Dr. Thomas F. Gatts of the University of Alabama, and Dr. David F. McAllister of North Carolina State University who felt this study warranted documentation and encouraged me to do so. Thanks also to Sharon Walker, Cheryl Mooring, and Leslie Batten without whose help this case study would not have been possible.

REFERENCES

[1] Agrawala, A. K., Mohr, J. M., and Bryand, R. M., "An Approach to the Workload Characterization Problem," Computer, Vol. 9, No. 6, June, 1976, pp. 18-32.

[2] Bronner, L., "Overview of the Capacity Planning Process for Production Data Processing," IBM Systems Journal, Vol. 19, No. 1, 1980, pp. 4-27.

[3] Cooper, J. C., "A Capacity Planning Methodology," IBM Systems Journal, Vol. 19, No. 1, 1980, pp. 28-45.

[4] "Guideline for Writing Computer Procurement Support Documentation," U. S. Department of Agriculture Handbook Supplement: DIPS Manual Chapter 3, March 1977, pp. 7-25.

or

McNeese, James E., "Computer Workload Forecasting," Proceedings of the 15th CPEUG, San Diego, CA, Oct. 1979, pp. 113-120.

[5] Tripathi, S. K., Gordon, K.D., and Agrawala, A. K., "An Optimal Sample Size Allocation Scheme for Benchmark Design," Proceedings of the 15th CPEUG, San Diego, CA, Oct 1979, pp. 105-111.

CPEUG80

Human Interfaces

Data Processing User Service Management

P. S. Eisenhut

IBM Data Systems Divsion, East Fishkill
Route 52, Hopewell Junction, N.Y. 12533

After 30 years since the advent of the electronic computer, we are fast entering the era of the information society. The trend is towards increasing use and dependence of business upon data processing. Data processing professionals must provide applications which perform adequately with respect to the business needs of users. Data processing professionals who concern themselves with the functional and technological aspects of data processing must now concern themselves with the performance or effectiveness of data processing in the business environment.

To do this requires a coordinated management process. This management process starts during the development of new data processing applications with user oriented service objectives. The process includes measurements, problem diagnosis, reporting of service and follow-up action, all on a regular basis. It also includes a formal capacity planning process which relates to committed levels of service as well as computing load.

Making the transition to an information society requires a new attitude among data processing professionals and a commitment from data processing management. Good data processing managers will view User Service Management as being critical to their own success.

Key words: Computer performance evaluation; data processing performance; data processing service; information system design; information system usability; performance management.

1. Introduction

Computing technology has developed at a very rapid rate during the 30 years since its beginnings. During this time, more computing capability has been provided at dramatically decreasing costs. Yet, it seems much less attention has been focused on the Data Processing - User Interface depicted in Fig. 1. For any given enterprise, a Data Processing Organization (D/P) will provide service to various user organizations within that enterprise. Ideally, this service should be in response to user needs. The interface breaks down because users find it difficult to express needs in terms meaningful to D/P; and because D/P finds it difficult to determine those courses of action which best satisfy user needs within limited D/P resources.

The following are examples of this breakdown in the D/P - User Interface.

Example #1

This example is provided by Richard Nolan (Nolan, July-August 1977). The D/P department of a medium size company auto-

mates the marketing organization's order entry system. The vice president of marketing is alarmed when he sees his monthly bill from D/P rise to 25% of his budget and to over 3 times the cost of order entry prior to computerized order entry. He demands an explanation from D/P. The representative of D/P explains that D/P bills so much for CPU, so much for elapsed time, so much for kilobytes per minute, adjusted to reflect virtual storage, and so much for EXCPS. He asks the vice president of marketing which of these he would like to control. This explanation is the last straw for the vice president of marketing. He phones the president of the company with the following complaint:

"I'm strapped. Data Processing is charging me for services that are essential, and I can't do anything about the cost. When I try to get down to how to control the costs, all I get is technical gibberish...."

Example #2

A large manufacturing site of an international corporation decides to improve productivity by installing a computerized dispatching system for equipment maintenance personnel. It is designed to have on-line terminal input and output and operate in real time. Another location already has such a system, and D/P estimates one systems analyst and one programmer can complete the implementation in one year. Three years later, three programmers are still rewriting programs to make the system usable to the maintenance organization. Response times at the terminals are over 15 seconds, and terminal access to the data is only possible 75% of the time. Under pressure from manufacturing management, the maintenance organization gives up on D/P. Within three months the maintenance organization implements their own manual process using work order tickets and telephones.

Other examples of the breakdown in the D/P - User Interface include the following:

° Reluctance by D/P to provide users with higher speed lines for fear that additional load will result on the central computer.

° Removing a transaction from a performance report because its response time was unacceptable.

° Sixteen weeks of unacceptable transaction response times because a programmer changed the procedure for purging the data base just prior to leaving the company, but didn't tell anyone!

° User reporting unacceptable performance to top management based on their own measurements, and D/P attacking the measurement process.

In 1975 the IBM users group, SHARE Inc., sponsored a study to extrapolate the trends in the data processing industry through 1985. The authors of the SHARE Inc. study make the importance of the D/P - User Interface quite clear when they state:

"Our main conclusions are that over the next decade, the major tasks of the data processing industry will be to improve:

° The quality of data processing services - as perceived by the end users of these services;..." (Dolotta, et al, p. 9, 1975)

According to the SHARE study, the trend is for the operation of business to rely more and more upon data processing. It is imperative that data processing organizations take steps to better meet the needs of users. This paper introduces the framework of a management process to accomplish this. It then discusses some of the key problem areas encountered in trying to implement this management process in a typical D/P organization.

2. The Management Process

2.1 New Application Development

The management process starts by managing the design and development of new user applications. A typical project management system to control the progress of a new application calls for project milestones and phases as indicated in Fig. 2. Key phases are problem definition, design specification, development (programming), and trial production followed by a final acceptance. In a D/P environment, this process often falls short of the mark because it manages function at the exclusion of performance.

Users require both performance and function. Functional requirements are those describing what processes the program, equipment, and hopefully, people do. The performance requirements are quantifiable expressions of how well the functions are performed. Figure 2 shows how performance management relates to the project phases.

Management must require that design specifications include objectives for per-

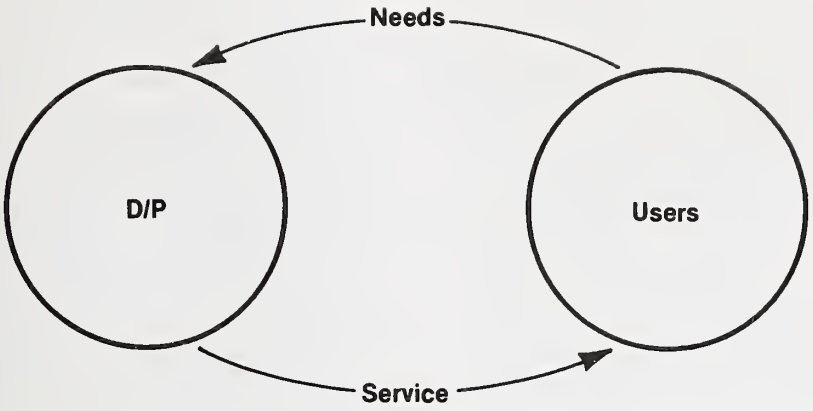


FIGURE 1. Data Processing-User Interface

Milestone	Phase Completed	Performance Management
1	Problem Definition	Understand User Performance Requirements
2	Specification and Design	Agreed Design Objectives for Performance
3	Development and Test	Developed Measurement Process
4	Accepted Trial Implementation	Performance Meets Design Objectives
		Objectives Become Commitments

FIGURE 2. Application Project Management

formance. Figure 3 illustrates a possible categorization of user performance requirements in terms meaningful to users. If these are the categories of requirements, then they also must be the categories of performance objectives to be specified as part of the design. Design specifications should also include how performance will later be measured. The application project should not be allowed to proceed past milestone 2 until this requirement is met. Likewise, the application should not be formally accepted as completing milestone 4 until the design objectives have all been satisfied as measured during the trial production period. At milestone 4, these design objectives may now become "commitments" by D/P.

Objectives must be jointly developed and agreed by all parties affected by the outcome. Various departments or organizations may be involved. Programmers, data base administrators and hardware configurators can control response time at a user's interactive terminal. Data center operations and operating system support personnel can control availability and reliability of the applications to the user. Data center operations and job schedulers can control turnaround time for job completion or the degree to which scheduled reports are on time to the user. Users, as well as programmers, control workloads. All parties control cost. All affect the profits of the enterprise. Therefore, all are involved in the management process.

2.2 Overall Process

Figure 4 illustrates the overall user service management process starting with new application development as just discussed. By the time the application achieves milestone 4, there should be a committed set of performance objectives.

Given this set of committed performance objectives, there must be regular (e.g., weekly) performance measurements of how well D/P and the users are performing relative to these objectives. These performance measurements must then be analyzed and evaluated to determine the reasons for deviations from the agreed objectives. Then, as performance versus objectives is reported to the D/P and user management, reasons for the deviations and recommended actions may also be reported. Management must then make final decisions on the actions or projects to undertake. These action projects result in the attainment of old objectives and in the establishment of new ones, thus closing the cycle.

The D/P Plan, usually done once or twice per year, is shown in the center of Figure 4. Measurements determine existing load versus D/P resource capacity (manpower, CPU, DASD, etc.) and performance level. Given the existing load, users and application programmers participate in the projection of load. The D/P Plan is at least a two year projection of load and necessary D/P resources required to handle the load within the limits of the agreed performance objectives. The performance objectives associated with the Plan are those currently committed as well as design objectives that will become commitments. The result of the Plan is a set of actions, which may be managed as milestone type projects that support these objectives and the projected load.

The management process described in Figure 4 is dependent upon its many subprocesses for success. D/P management must understand the large amount of coordination required, and they must be willing to assign staff responsibilities for performing these subprocesses.

2.3 Benefits

There are several key benefits to the User Service Management Process. Improved performance means that users will be more productive. For example, improved response time at a terminal means less user wait time and potentially fewer terminals and users for the same job or task. Also, for example, a manager that receives a manufacturing line trouble report prior to the start of a shift can assign people to resolve the trouble before those people get involved in other things. Because it benefits them, users are motivated to participate in the process.

Increased user satisfaction results not only from improved performance, but also from improved trust and communications. If a reporting system allows both D/P and users to see performance in like terms, there will be fewer escalations to higher levels of management to resolve disputes.

"Management by Objectives" becomes possible if measurements and reporting are done against performance objectives. The objectives give purpose to actions like data base reorganization, adding higher speed communication lines or a faster printer, rescheduling batch jobs, or program modification.

- Scheduled Availability of Service
- Unscheduled Down Time
  - Mean Time of Service Unavailability
  - % of Schedule Not Available
  - Maximum Time of Outage
  - Number of Failures Or Service Interrupts
- Response Time (Interactive)
  - Average
  - Percentiles
  - Tolerances
- Turnaround Time (Unscheduled Batch)
- Report Timeliness (Scheduled Batch)
  - % of Reports Available to User on Time
- Workload Capability
  - Average
  - Peak
- Cost Charged

FIGURE 3. Performance Requirements

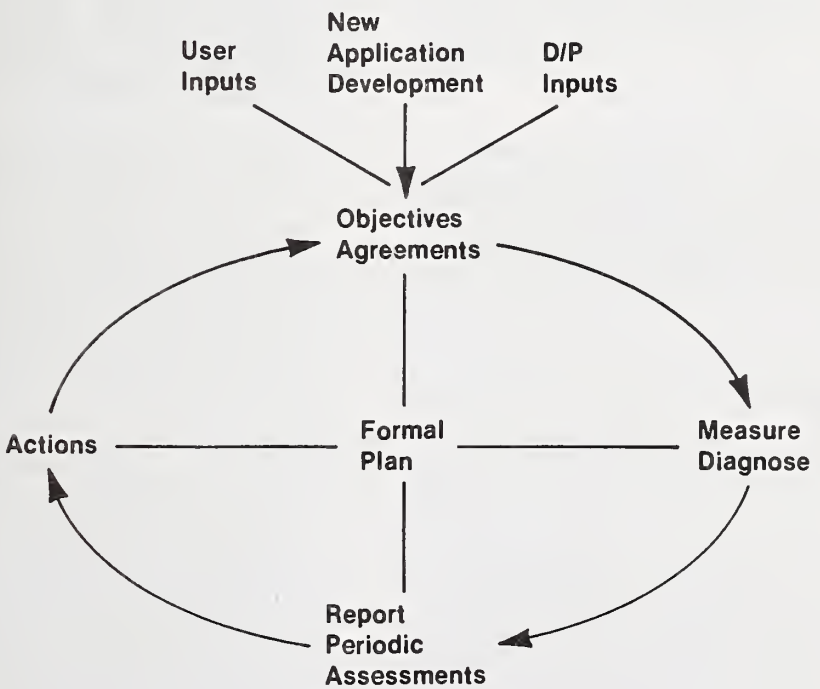


FIGURE 4. User Service Management Process

Rather than fight fires, management can make changes based at least partly on the effects those changes will have in user productivity and satisfaction. It also becomes possible to talk to users about the tradeoffs between more function and more performance.

Justification of plans and hardware proposals becomes easier. It is quite often necessary for D/P management to justify actions to higher levels of management. Such actions might include procuring major pieces of hardware like computers, mass storage devices, or printers. If higher levels of management understand that objectives are important for the business, and if D/P management can show that these actions contribute to attaining the objectives, then D/P management will find these actions easier to sell.

3. Problem Areas

The successful implementation of the management process requires overcoming several problem areas. Perhaps of even greater importance is that before general acceptance is possible, higher levels of management, or at least the manager of D/P, must understand and support the process and the methods to be used to overcome the problem areas. The following discussion concerns these problem areas and suggestions for dealing with them.

3.1 Defining Objectives

Generally the objectives should be grouped according to a categorization similar to Figure 3. Specific definitions, however, will vary from one enterprise to another. The following general suggestions may help to avoid problems with acceptance or workability:

(1) Only define objectives that can and will be measured.

(2) Be sure that the results of measurements against objectives can and will be reported to users.

(3) Attempt to define service objectives which can be directly linked to the business objectives of the user organization. For example, if a D/P service for process control is out for more than X hours, it may impact the user's ability to produce product. Defining the maximum outage allowed would be a critical objective.

(4) Attempt to define objectives as close to the D/P - User Interface as possible. For example, define a response time at the terminal as seen by the user rather than at the host computer.

(5) Allow the key users to choose objectives most meaningful to them.

(6) Pick a time frame long enough to smooth out the effect of short term and random events, and short enough to allow attention and action before too much damage is done.

(7) Remember, "To have an objective and not meet it is not nearly as bad as never to have had one."

3.2 Specifying Values of Objectives

What values should be set for the objectives? For example, should average terminal response be 2 or 15 seconds? Should it be the same for data input as for graphics? The following suggestions for setting values may be helpful:

(1) Values for design objectives should be reasonable as determined by similar applications or technologies. Values for commitments should be achieved over a sustained period as determined by measurements. However, commitments should not be lowered because of degraded performance.

(2) Values should be related to business needs. For example, the timing of a report required to assign priority to work moving through a manufacturing line may be more critical than an employee education record. Therefore, the manufacturing report may be required daily by 7:30AM 95% of the time, whereas the education report may not be required at any particular time of day.

(3) Terminal response times should not exceed 15 seconds if the user is to remain at the terminal. (Martin, p. 326, 1973.) Faster user "think" times demand faster terminal responses. (Doherty, p. 154, 1979.) This is why graphics requirements are in subseconds. Consistency of response is also important. The spread of response times is specified by a percentile value (90 percentile) as well as the average.

(4) Tolerances should be established around the objective values to recognize random events. For example, if the average actual response is 5 seconds, and if results are reported daily, then some days may average 6 seconds and others 4. The tolerance (possibly 1 second) should recognize this fluctuation. Action would be taken only when the objective value plus the tolerance is exceeded.

(5) Objectives should offer some challenge. "You don't make progress looking backwards."

3.3 Form of Agreement

How formal should this agreement of objectives be? Should it be a "contract" with a specific user organization? Initially, users may balk at the term "contract" which may imply a formal and legally binding obligation. On the other hand, the complete absence of a written document would imply a lack of agreement and would also preclude communicating a common understanding of what the objectives really were. The ideal solution seems to be to make available to all parties to the agreement, a common "Document of Understanding" (DOU). This document should contain the performance objectives agreed upon as well as a statement of the philosophy and reasons for the agreement. It may avoid future arguments if the document also contains references to the source of measurements and manner of reporting actual performance, and the conditions for taking corrective actions.

With what level of user organization should D/P have a DOU? Too high a level will lose participation by the actual end users. Too low a level will mean too many contracts, all subject to change with reorganization. Administration could be costly. Too low a level can also mean too much interdependence between user organizations. It may not be possible to manage service to one organization without affecting service to another.

There is much evidence to indicate a trend in the years ahead to more and more interdependence between user organizations. (Nolan, March-April 1979.) This interdependence is characterized by many organizations using common data bases and applications, and the passing of information between organizations. An example of this is an online IMS application for tracking the flow of work-in-process through manufacturing. Jobs are released to manufacturing departments by various production control departments who make entries into the IMS data base. Claims of progress through successive fabrication steps are made by various manufacturing departments. Information is passed to an accounting department for product, costing, and to inventory control departments for inventory adjustments. In this data base application, performance is controlled by changing the application or by changing IMS. For example, a change to the data base structure or a new version of IMS are possible actions. However, these changes affect all users of the application or all users of IMS.

One way of dealing with this interdependence is to have separate DOUs for each D/P service offering, for example, one for IMS, one for TSO, one for unscheduled batch, etc. A DOU could have separate objective values for each application if values were not common across the service offering. The DOU would then be a common agreement for all users of the same service or application. This application orientation is, of course, fully consistent with the idea of setting performance objectives for new applications as discussed earlier.

"All players on the team must play according to the same game plan if the team is to win."

3.4 Measurement Tools

It has been said, "If you can't measure it, you can't manage it." A corollary is that it is meaningless to have an objective, if there is no way of knowing where one is relative to that objective. On the other hand, it is also meaningless to measure where you are relative to an objective that is irrelevant. It is often said that measurement tools in current use provide too much data and not enough information. What is really being said is that the measurements are compared to objectives whose relevancy is non-existent or at lease unclear. For example, to a user who has not been receiving reports on time, the channel utilizations from 2:00 to 3:00AM on the host computer would not be relevant. The user only understands that his report schedule has been met only 50% of the time this past week. What is needed are tools and procedures to bridge the gap between the user's view of performance against an objective and the evaluation of the causes of that performance.

First, there is a need for tools which measure actual performance against user agreed performance objectives. Tools of this type are almost non-existent. Therefore, measuring the percent of time that reports were available to a user on schedule must be done by a manual process. Response time at the terminal is another example. Use of hardware monitors have been unsuccessfully attempted. Alternatives have been to create a typical terminal transaction and invoke it automatically. Another alternative might be to measure response only at the host computer and estimate the teleprocessing time. Yet the most relevant alternative may be to have the terminal user record responses in a log according to a prearranged sampling plan. This certainly involves the user in the process.

Secondly, there is a need for tools to relate higher level performance measures to lower level measures. In a preceding example, the reason that the user reports were available only 50% of the time may have been due to channel utilization, to DASD contention, to high central processor load, or to people in the printout room not tearing reports off and putting them into the user's pickup bin. Reports are needed to summarize this type of information over a period of time and make it available to a problem analyzer on a timely basis. Unfortunately, such tools are not standard, and home-grown programs and manual data synthesis are the rule.

Thirdly, there is a need for an organized approach to analyzing problems relative to the performance objectives agreed to by users. This means D/P management must invest in qualified people who can use the available tools, or can program new ones, for the purpose of problem diagnosis. This diagnosis is necessary whenever actual performance deviates from user-agreed objective. Recommendations for appropriate action should come from this process.

3.5 Reporting

Assume that "What the user doesn't know can hurt you." Unless performance is reported to them, users may blow minor problems out of proportion and escalate them to higher levels of management as a means of guaranteeing acceptable levels of performance. Therefore, performance should be reported to users. D/P services users, not the other way around. Users have the right to know.

Developing DOU's or contracts with users is a formal process specifying user-oriented performance objectives. It makes sense that the process of reporting actual service back to the users should also be a formal process. D/P management must assure that the report includes performance expressed in the same terms as those agreed to in the DOU, and is available regularly according to the agreed frequency. D/P management must also assure that problems and corrective actions are reported relative to the user service.

One of the problems with reporting is that there are no standard program packages to generate a comprehensive report. Each location or enterprise may have different formats and contents depending upon the DOU or contract. Furthermore, the variety of input required and the scarcity of measurement tools means that the collected data input must come from a large number of first line departments. To a large extent, the collection of data and the summarization of data into a meaningful report will be a manual process. The first line departments involved in the reporting process may hesitate to devote resources to work that is not directly in line with their specialized missions. It may be necessary for management to redefine department missions to include responsibilities for data collection and reporting. Generally, D/P management may find that a great deal of coordination is required to assure effective and timely reporting.

Figure 5 shows a logical organization to accomplish coordinated reporting. Someone will be assigned the clerical responsibility for pulling together the data collected by the various first line departments doing measurements. This person will summarize the data into predefined formats for regularly scheduled presentations and formal reports to users and D/P mangement. An analyzer will also be assigned responsibility for problem diagnosis and action followup. This analyzer will make use of the detailed and traditional performance measurement and modeling techniques to explain poor user service shown in the management presentations. He will recommend courses of action to improve user service. The formal reports

to user and D/P management should include the status of actions being taken to correct poor user service. Management coordination is accomplished by establishing a reporting process, by formally assigning responsibilities for the reporting tasks, and by following up on action items.

4. Mentalities

Perhaps the biggest obstacle to successful implementation of the user oriented process of management are the mentalities of the organizations involved.

4.1 The User Mentality

Users are of many different personalities and professions. They may be accountants or clerks, or they may be engineers or management staff. Users are specialists in their own profession. The user works with output units, such as invoices processed, inventory part status, production schedules, etc. Users tend to be alike in their view of D/P. According to the study by SHARE INC., (Dolotta, et al, p. 43, 1976.):

"As a rule they have no interest in how the system works as long as their needs are met; that is, as long as the data processing system provides the services they desire at what they consider to be reasonable cost. A failure to provide these services is a failure of the entire data processing system in these users' eyes."

4.2 D/P Mentality

The D/P mentality is best described by Louis Fried (Fried, p. 30, 1979).

"The programmer's programmer is one who can design a program that will operate faster using the least amount of memory, to produce a desired result. The programmer's interest is directed toward ultimate efficiency in use of the computer. As might be expected from one drawn to a highly individual activity oriented toward abstract and machine-related problem solving, the programmer is generally introverted. He finds his best expression in dealing with situations that do not include human relationships."

The D/P mentality has great difficulty understanding user needs and has great difficulty adopting the user service management concept.

Traditionally, professionals in D/P begin as programmers and so this mentality is extended to programmer-analyst and to systems analysts dealing with hardware architecture and systems software. Lower levels of D/P management who are brought up in this environment may also exhibit the same mentality. Lower levels of D/P management may be afraid to "dictate" objectives to their staffs lest they lose popularity. Staff may fear management evaluation or lack of support in their efforts to achieve objectives. The following typical comments were made by lower levels of D/P management, when asked to agree on user-oriented objectives, and are indicative of the D/P mentality:

- "The users aren't complaining, service levels must be OK. Don't need all this."
- "I won't sign unless they sign."
- "I can't agree on objective unless I am certain I will achieve it."
- "I don't have control over service."
- "I should not be responsible, they should."
- Can't commit to anything, things are always changing."
- "User will use the objective for ammunition and turn me in everytime I miss it."
- "My manager told me not to sign anything unless he sees it first."
- "I can't sign anything unless my staff first commits to it."

4.3 Renaissance Man

The D/P mentality is necessary and vital, but there needs to be more of another type of mentality in D/P. This new mentality is one that can bridge the user mentality to the D/P mentality. It is a total systems view. It is a business view. Enter "Renaissance Man."

This new mentality parallels very closely, what Louis Fried calls the System Analyst's Analyst. (Fried, pp. 31-32, 1979.)

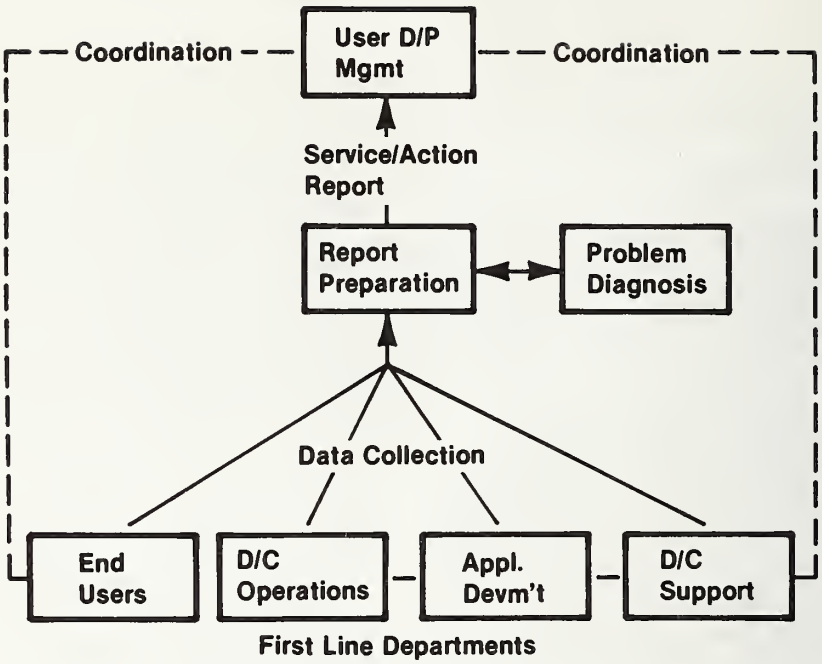


FIGURE 5. Organization for Performance Reporting

Manufacturing	D/P
Production	Data Center Operations
Prod./Inv. Control	Operations Support
Product Engineering	Application Development
Process Engineering	Software Support
Tool Engineering	Hardware Support
Industrial Engineering	Plans, Measurements and Reporting

FIGURE 6. Manufacturing-D/P Analogy

"The 'Systems Analyst's Analyst' is one who can design and implement a system that is appropritate to the problem, make the optimal use of the organization's resources, be integrally related to the other systems in the organization, and do the job intended. Ideally, he'll function as an extension of the organization's management...."

"....He is keenly aware that all systems are run by people. His concern is centered around aligning the goals and activities of the people with the goals and activities of the system. Characteristically, the analyst is an extrovert. He is oriented toward interpersonal relationships."

It is time to view D/P as a business, or at least as a business within a business. Figure 6 shows that there is a direct analogy between organization of a manufacturing enterprise and the organization of D/P. "Customers" become "Users;" "Product fabrication" becomes "Data Center Operations," etc. The same professional skills that make the manufacturing enterprise successful can therefore make D/P successful.

Where does Renaissance Man come from? It may be necessary to change hiring practices. Traditionally, D/P has hired programmers with no prior experience or with computer science degrees right out of school. These neophytes were then required to acquire company-sponsored eduction and work up through the ranks of programming responsibilities to become systems analysts and into lower levels of D/P management. This is an unlikely source for Renaissance Man. Perhaps a better source is D/P professionals from application areas or with business backgrounds. For example, MBA's, production control analysts and industrial engineers are possible candidates.

Renaissance Man's first step must be to reorient the top level of D/P management to the new business view of user service. According to Geoffrey Goodman (Goodman, p. 69, 1979.):

"The first steps in establishing a computer performance system are critical to success. Lack of attention to organizational and management needs are as likely to lead to failure as lack of attention to technical details."

The top manager of D/P is best able to see the business view. He doesn't get to the top by being myopic. In fact, the good D/P manager will view user service as critical to his own success. Indeed the top D/P manager may himself become Renaissance Man! Once the top D/P manager has the business view of user service, the lower level of D/P management and the rest of the organization will follow. The change will occur through a combination of direction from the top, education, reassignment of job responsibilities, and restaffing. Following this, the technical part of managing user service will then be relatively easy. The first step to reorient D/P management may prove to be Renaissance Man's biggest contribution.

5. Conclusion

If Data Processing is to continue to advance, a new process of performance management is required. The systems analyst, the industrial engineer, or even the information administrator, will prove to be the Renaisance Man who brings about the birth of usable data processing. The technology of flying machines came 30 years before commerical passenger travel by air. Likewise, after 30 years the technology of computing machines is about to enter its own era of usability. Some will call this period the Information Revolution. The need for Renaissance Man is here now. Each of you is challenged to assume this role!

6. References

(1) W. J. Doherty and R. P. Kelishy, "Managing VM/CMS Systems for User Effectiveness," IBM Systems Journal, Vol. 18, No. 1, 1979.

(2) A. Gayle, "Perspectives on Business Management - Data Processing a Manageable Resource," IBM Systems Research Institute Publication, TR73.003, June 4, 1979.

(3) G. H. Goodman, "Methodology for Establishing a Computer Performance Management System: A Case Study," CPEUG, 15th Meeting, N.B.S. Special Publication 500-52, pp. 69-76, 1979.

(4) S. Lowry and D. Schafer, "Implementing a Performance Management System," Proceedings Guide 48, Philadelphia, pp. 875-898, May 23, 1979.

(5) R. L. Nolan, "Managing the Crises in Data Processing, " Harvard Business Review, March-April 1979.

(6) R. L. Nolan, "Controlling the Costs of Data Service, " Harvard Business Review, July-August 1977.

(7) R. L. Nolan, "Computer Managers to Data Resource Managers," Proceedings Guide 48, Philadelphia, pp. 1797-1805, May 24, 1979.

(8) R. A. Sills, "Data Center Performance Reporting," Proceedings Guide 48, Philadelphia, pp. 469-474, May 23, 1979.

(9) D. P. Norton and K. G. Rau, "A Guide to EDP Performance Mangement," Q.E.D. Information Sciences Inc,. Wellesley, Massachusetts.

(10) C. B. Wilson, "Technology Assessment: ADP Installation Performance Measurement and Reporting," N.B.S., September, 1979.

(11) L. Young, "IE-The Ideal Interface with MIS," Industrial Engineering, November, 1979.

(12) Data Processing in 1980-1985, A Study of Potential Limitations to Progress, T. A. Dolotta, et al, John Wiley & Sons, by SHARE Inc., 1976.

(13) Practical Data Processing Management L. Fried, Reston Publishing Company, 1976.

(14) Design of Man-Computer Dialogues, J. Martin, Prentice-Hall, Inc., 1973.

CPEUG80

Installation Management

Performance Evaluation of Computer Operations Procedures

Patrick A. Drayton

Southwestern Bell Telephone Company
St. Louis, Missouri 63101

A discussion of the measurements required for an evaluation of the operating procedures of a large computer system data center. Included are the guideline values which should be achievable with good operating techniques and solutions to frequently encountered problem situations.

Key words: Performance, Operations, Measurements.

1. Introduction

1. Today's data processing hardware is experiencing an exponential advance in technology. Coupled with this is an increasing trend towards more sophisticated operating systems and user software (data base systems, intelligent networks). In marked contrast, there has been negligible attention placed on the operating procedures used in data centers. This is unfortunate since DP operations personnel must cope with the advancements in technology. Some examples should help illustrate the point.

As CPU hardware becomes more powerful, more jobs are processed simultaneously which places increasing pressure on the operations personnel in the areas of scheduling and resource management. In addition, all inefficiencies in tape mounting and WTOR replies are magnified by the larger multi-programming level. As the peripherals become faster and more complex, the operator's actions become more important. For example, in many installations the introduction of non-impact printers displaced numerous impact printers because of the five-fold increase in print speed. To operations this means that a five minute job setup time has the impact of 15-25 minutes of total print throughput based on previous hardware.

Operating systems, have progressed from single thread real storage systems to virtual machines. This complex environment has added problems to operations such as swapping, paging more powerful commands, intricate messages, and job entry subsystems. But there is still only one person sitting at the master console who must digest all the messages on the screen and make the appropriate response, or take the correct preventive action.

One additional area of technical advancement is user software systems. The increased use of on-line data base systems and time sharing systems has resulted in increasingly difficult operational problems. Instead of simply two or three batch jobs active at one time, operations is now responsible for

hundreds of terminal users. Each of these could have need of operator interaction to resolve response time delays, data base problems and hardware failures. In summary, there is an increasing potential for user-related performance problems.

It is clear that many areas of the DP environment have greatly advanced in the last ten years. However, the ability of operations to cope with these new technologies has not always kept pace. Recently a number of software products have entered the market whose aim is to automate portions of the operator environment. These include tape management systems, disk space control systems, automatic scheduling systems and operations interface subsystems.

This discussion attempts to explore the areas of performance as they relate to the actions of the operators. This includes the following:

- Sources of measurement data
- Guideline values with which to evaluate operations performance
- Recommended solutions to problems.

It is hoped this will both increase awareness of the importance of operations in the data center and provide a means of evaluating their impact on overall system performance and throughput. The specific tools and methods are aimed at large IBM MVS data centers. However, the basic principles should apply to all operating/hardware environments.

2. Sources of Measurement Information

There are six major sources of information concerning operations performance. A brief description of each follows:

- Hardcopy log - this is a data set that contains a copy of all messages that appeared on any console plus all operator replies to program/system requests. All entries are time-stamped.
- SMF (System Management Facilities) - accounting type data on all jobs including resource consumption and termination status.
- RMF (Resource Measurement Facility) - IBM's system monitor which reports on device usage, paging, CPU usage and SRM (System Resource Manager) activity.
- Timings - stopwatch timings of events.
- Observations - monitor data may indicate a problem, but without observing the situation, the cause may go undetected.
- Interviews - these include discussions with users, operators and data center managers. It is an attempt to get their perspective on what is right/wrong in the data center.

3. Measures, Guidelines and Corrective Action

This section is divided into five areas:

- Tape Operations
- Disk Operations
- Printer Operations
- Scheduling
- Quality Control

In each area, the operation to be measured, the guideline value and potential corrective actions will be discussed.

3.1. Tape Operations

For any data center with an appreciable amount of batch work, the operation of the tape pool is of vital importance. Four measures of performance are discussed: scratch tape mount time, input tape mount time, allocation recoveries and tape load failures.

3.1.1. Scratch Tape Mount Time

This is a measure of the time between a request for a non-specific scratch tape and the satisfaction of that mount request by an operator. Both the request and the volume recognition message can be found on the hardcopy log. If the average time required to mount a scratch tape is greater than 60 seconds, there are problems in this area. Some possible solutions are:

1) Insure that a supply of scratch tapes are within easy reach of the tape pool operator. That is, the tape console and the scratch tapes should be as close to each other as possible.

2) Review labeling procedures. Do not prelabel scratch tapes and require that a particular volume is needed be satisfy the request. If a physical label is attached after each tape file is created, the use of a label printer driven by the mount messages greatly reduces mount time.

3) Insure that scratch tapes are easily distinguished from used tapes. An operator should have no doubt as to the status of a particular volume.

4) Use of a tape management system relieves the need for rings in scratch tapes, since it will protect unexpired data sets and it relieves the need for external labels except for tapes shipped to other locations.

5) Review the tape drive configuration. The ideal is a square or rectangle with all drives facing the operator. For tape pools which are shared between systems it is best to designate strings of drives as dedicated to a particular system. There will also need to be strings which are designated as sharable between systems to facilitate variances in workload. The dedicated drives should be localized in an area near the operator's console to reduce mount times.

3.1.2. Input Tape Mount Time

This is the time between a request for a specific volume and its satisfaction. Unfortunately IBM systems without a local modification do not recognize the satisfaction of input tape mounts. There are two available sources for the data required to make this measurement. They are:

- A manual timing between the issuance of the message and the tape mounting

- Average percent mount pending for all drives from RMF tape device usage report.

If the manual timing results in an average greater than three minutes or RMF reports an average mount pending percentage greater than ten, a problem exists. Some potential solutions are:

1) If input tapes remain in the storage area until a mount request appears on the console, pre-staging tapes based on a job schedule will greatly reduce mount time. One method of accomplishing this is to have someone review all job run requests and through the use of a tape management system, note which input tapes each job requires. This will allow for the pulling of these tapes prior to the job executing. Some automated scheduling packages can generate a list of tape volumes that will be needed up to 24 hours in advance.

2) After an output tape has been created, it should not be refiled immediately unless it is known that no subsequent step/job will be using that volume as input. It is best to place newly created tapes on a holding cart for from one to two hours prior to refiling.

3) All tapes should have external serial numbers on the spine of the cartridge/case to facilitate quick identification.

4) For input tapes prestaged in the operator area, quick retrieval is enhanced if they are grouped in some logical manner. This could be in serial number bands or by jobname/application group.

5) The tape drive configuration should be conducive to quick access by the operator. (See #5 under scratch tape Mounts.)

3.1.3. Allocation Recoveries

An allocation recovery occurs when a job step requires more physical peripherals (in this case tape drives) than are physically on-line to the system. This job step and all subsequent jobs allocating tape drives will wait until the allocation request is satisfied. Therefore, a small number of these can greatly impact throughput. These can be measured by an analysis of the hardcopy log. NOTE: SMF has a record type for allocation recovery but does not time them, nor are all responses counted. If more than 10% of the tape mounts result in an allocation recovery message, a problem exists. Some possible solutions are:

1) Each job which requires tape drives should be so identified to the operator along with the highest number of drives required concurrently. This can be accomplished either by using a position in the jobname (i.e., seventh position indicates the number of drives), by use of the JES /* SETUP card, or by use of a special job class.

2) A master console operator should be constantly reviewing the number of drives available before releasing a tape job for execution. The standard IBM commands are available for this purpose.

3) If the tape pool is shared between two or more CPUs, excellent and constant communication needs to exist between the operators.

4) Review tape requirements of all jobs for possible reduction. Small tape files may be candidates for disk. Use of flip/flop on tape files should only be done when the file is greater than three reels. Allocate files to the same drive whenever they are not required to be open at the same time (i.e., SORTIN and SORTOUT files). Use FREE&CLOSE on any tape files which are closed an appreciable amount of time before EOJ (i.e., SORTIN when SORTOUT is to disk).

3.1.4. Tape Load Failures

A tape load failure occurs whenever a tape fails to become ready the first time it is mounted on a drive. The operator is then required to remount the tape. An abundance of these are aggravating to the operators and degrade job throughput. If more than 5% of the tape mounts are resulting in load failures, a problem exists. Some potential solutions are:

1) Institute a practice of tape leader inspection after each load failure rather than just retrying the load operation. If the leader is damaged, it should be clipped before the tape is reloaded. Failure to do this will only cause the problem to get worse.

2) For tapes in "easy-loader" cartridges, be aware of situations where the tape is removed from the cartridge to accommodate non-cartridge loading drives. If these exist, they are probably the cause of leader damage. Removing the tape from the cartridge also causes the leader to not be positioned near the cartridge opening, which will degrade load performance.

3) For tapes not in cartridge, the leader should be retained with a

sponge rubber grommet to prevent damage.

4) The temperature and humidity in the computer room and in the tape storage area should be within manufacturers specifications. Temperature in the range of 65-75 degrees and relative humidity of 35-45% is usually acceptable. Deviations from these could cause excessive static and inhibit loading.

5) The hardcopy log can be used to identify problem tape drives. The vendor should insure that those drives are at the proper maintenance level and that they are operating within specifications for airflow and suction.

6) All tape drives should be cleaned at least once every eight hours.

3.2. Disk Operation

Control of the disk pool is extremely important. Almost all jobs will either use permanent disk data sets or require temporary work space. Three measures of performance are discussed: allocation recoveries, space allocation and ENQ delays.

3.2.1. Allocation Recovery

In a data center with mountable disks, allocation recoveries can impact throughput. An allocation recovery occurs whenever a mountable disk request cannot be serviced on the available drives. If more than 10% of the disk mount requests (as measured on the hardcopy log) result in an allocation recovery, there is a problem. Exclude from this count allocations for temporary space. Some potential solutions are:

1) Review scheduling procedures. The master console operator should be made aware of all mountable disk requirements either via JES /* SETUP cards or a list of all jobs' disk requirements.

2) Attempt to schedule together, jobs which require the same mountable disk. This reduces both the number of mounts and the potential for allocation problems.

3) If scheduling does not relieve the problem, evaluate the cost effectiveness of adding hardware and making the problem disks permanently resident.

3.2.2. Space Allocation

Insufficient temporary disk work space will significantly reduce throughput as a job waits for another job to release space. This problem is difficult to measure, but the console operators are very aware when it is occurring. For this reason, interview and the operators note the level of complaints about space allocation problems. If these are high, some potential solutions are:

1) Review space requirements of all large users of scratch space. After identifying the jobs, certify that their space requests are justified. For SORT WORK data sets which are created in one step and passed to another, console/sysout information should be available from which actual space required can be calculated. For data sets the use of SMF EXCP counts on file creation plus block size information can be used to calculate space needed. For files entirely used within a step, SMF EXCP counts can give activity but the programmer will need to be questioned on the number of times the data is accessed.

2) Always allocate in cylinders or rounded blocks to increase performance and decrease fragmentation.

3) For files which vary greatly in their volume, use the primary allocation for the expected volume with a secondary allocation to handle unusual volumes. Potential problems exist with this if the secondary is not available in the

middle of a long running job. Use of RLSE in the JCL will allow freeing of unneeded space after the file is closed.

4) Dedicate disk packs for use as scratch packs and insure that no permanent data sets are allocated to them.

5) Regularly execute maintenance which scratches unneeded data sets and data sets remaining after a system crash.

6) Avoid scheduling two known large work space users at the same time.

3.2.3. ENQ Delays

ENQ delays occur when two jobs require the same resource and one of them cannot share the resource. The most common occurrence are two jobs accessing the same data set and one has DISP=OLD. However, ENQ delays can also occur for catalogs and VTOCS. Often an ENQ will result in a swap out of the second job requesting the resource. If RMF indicates you are experiencing more than .01 ENQ swaps per second, a problem exists. Use detail RMF enqueue reports to identify the causes of the ENQ swapping. Based on these findings, the following solutions are available:

1) The ERV (enqueue residence value) in IEAOPTxx member of PARMLIB should be set at 500 to allow time to resolve enqueues before swapping occurs.

2) DSN - if many ENQ are occurring because two jobs are requesting the same data set, attempt to either reschedule the jobs or have their execution tied together via an IEBGENER step whereby a successful execution of one job will cause automatic submission to JES of the subsequent job. Also, review the use of DISP=OLD to insure that updating of the data is being done. If this is a read-only operation DISP=SHR should be used.

3) VTOC - ENQs occurring on the VTOCs of storage packs, indicates a need for more scratch packs. Contention for allocation of work space is lessened when there are more candidates for allocation.

4) Catalog - ENQ problems involving catalogs usually only occur on large TSO systems. They are indicative of the need for a split of the problem catalog into smaller user catalogs. The most logical breakdown is best and easiest to maintain (i.e., by department, application, location).

3.3. Printer Operation

The performance of the printer subsystem is vitally important in that it produces the product the end-user actually receives. Quality, clarity and promptness are important in this area. The main measure is user satisfaction. If during the user interviews, complaints about printed quality are raised, the printer operation needs to be investigated. If delivery of printouts is sited as a problem, RMF can be used to measure printer usage. If the average percent not ready is less than 10% and the busy percentage is not in a range of 30-70%, printer operating procedures should be questioned. Some potential solutions are:

3.3.1. Quality Problems

1) For impact printers, the ribbon should be changed regularly. It is better to have a set interval rather than waiting until it is noticed that the printing is getting light.

2) For nonimpact printers, the toner should be refilled on a regular interval.

3) Cleaning of the printer either by the vendor or by the operator should be done regularly. Once a shift for impact, every hour for nonimpact.

4) Preventive maintenance is extremely important and deviations from the vendor's recommended schedule should not be allowed. It may be

inconvenient, but in the long run it will provide higher quality output and more sustained periods of up time.

3.3.2. Output Delivery Problems

1) Insure that the configuration is conducive to good work flow. Are the paper load/retrieve areas on the printer easy to access? For example, the 3800 paper is both loaded and removed from the front. Having two 3800s face each other will allow one operator to quickly service both.

2) Adequate storage area should be available around the printers to pre-stage both stock and special forms paper so that the operator does not have to retrieve subsequent boxes from the stock room.

3) Separator sheets between jobs should be employed. They allow job differentiation and eliminate the need to remove the paper after each job completes.

4) Some type of destination address should be attached before the printout leaves the printer area. This can either be computer-generated or be added by the operator by using a distribution list.

5) If the printers are averaging greater than 70% busy, look into acquiring additional hardware. Higher usage than this is difficult over extended periods due to maintenance, downtime and job setup.

3.4. Scheduling

As has been previously mentioned, the use of an accurate job schedule is necessary for all data centers. If during the interviews with the user community, service complaints arise, scheduling should be investigated. SMF accounting data can be used to identify the flow of work through the CPU. This data, along with a list of commitments to the user can be used to determine the specific service problem areas. Once identified, the following action should be taken:

1) Perform an analysis of the abends experienced by the problem application(s) using SMF data. Depending on the abend code, different actions should be taken.

   - USER and OCX - usually a programming problem, but could be caused by invalid input data/parameters. Review data verification procedures and redesign programs to delete invalid data and print appropriate message rather than abend.

   - 222 - operator cancels should be eliminated. Investigate the reason for any cancellations, inadequate resources, data sets unavailable or input media not ready and correct the situation.

   - X37 - usually a space problem. Assure that adequate work space is available at all times. This often occurs during runs of unusually heavy volumes. These problems can often be avoided by use of secondary space allocations.

   - X13 - label problem. This usually indicates a problem in matching JCL information with the physical data set attributes. Use of the catalog and a tape control system, and eliminating the coding of volume and DCB parameters on input files should correct this.

   - 001 - usually caused by read/write errors on hardware. Review PM procedures and use LOGREC data to pinpoint problem devices.

- JCL errors - review methods of updating procedures. Use of TSO with initials or comments by each modification is best. Use of reader procs where the jobs are submitted via TSO eliminates card decks and many override problems.

2) Review critical path of application. Insure that operators are aware of which jobs are in a direct line with the required output and which are sideline jobs which can be processed as time becomes available. A special SRM performance group can be used to guarantee an application's service levels.

3) Review start-end time of each job in the application's cycle using SMF data. For each job, the following questions should be asked:

    - Is its current start time the earliest it can begin? Is the data actually ready sooner?

    - Is this dependent job being started as soon as possible? For example, if job B requires a tape out of step 2 of job A, it should not wait for all the steps of job A to complete before beginning execution. Insert an IEBGENER after step 2 in job A, which automatically submits job B.

4) Review relative priority of applications, i.e., who gets delayed when a "hot" job is put on the system. These priorities should be documented and known by everyone.

3.5. Quality Control

Control of a job's input and output is extremely important for a successful operation. The measure of quality control is two-fold; first user complaints will be heard when they are dissatisfied with their reports, second SMF can be used to report on job reruns. If more than 4% of the jobs are rerun, a problem exists. Some potential solutions are:

1) Install a position of quality control which monitors all traffic, both in and out of the computer room. This would include the following:

    - Review all user report requests for accuracy and completeness.

    - Insure all JCL overrides are error free.

    - Notify master console as input becomes available.

    - Review all SYSOUTS for good EOJ.

    - File all SYSOUTS for later reference.

    - Extract run control totals as needed.

    - Forward abends/JCL errors to person responsible.

    - Review all output for compliance with user requests and for quality.

    - Log all activity in and out.

    - Log all problems, including time, person referred to, solution and time lost.

2) Install a change control procedure. This should insure that both system program and procedure updates are reviewed for completeness and are applied per instructions.

4. Conclusion

The procedure outlined above is not a one-time experience which is only

entered into when the wolves are at the door. For a data center to operate effectively, a system of continuous performance monitoring must be followed. This insures that few surprises arise and that adjustments are made in a calm, controlled atmosphere rather than a "red alert" panic situation. It is also helpful if the measures are done on a work shift basis with the results published to establish competition between the shifts.

5. Guidelines

Listed below are the guidelines discussed in each area:

	Measurement	Acceptable Range
1.	Scratch tape mount time	0-60 seconds
2.	Input tape mount time	0-3 minutes 0-10% mount pending (RMF)
3.	Tape allocation recovery	0-10% of tape mounts
4.	Tape load failure	0-5% of tape mounts
5.	Disk allocation recovery	0-10% of disk mounts
6.	Disk space allocation	operator feedback
7.	ENQ delays	0-.01 ENQ swaps/seconds (RMF)
8.	Printers	user feedback 0-10% NOT READY (RMF) 30-70% BUSY (RMF) user feedback
9.	Scheduling	user feedback
10.	Quality control	user feedback 0-4% jobs rerun

Evaluating Total Computer Performance for Top Management

Richard L. Fidler

Office of Procurement and ADP Management
Office of the Secretary
Department of Commerce
Washington, DC 20230

Computer performance evaluations, installation reviews and EDP audits have all focused on an individual data processing system. It is rare that any evaluation is made of all computer systems and related functions on an organization-wide basis. Consequently, there is an abundance of technical and management information for use at the installation level but a dearth of management information for use at the organization's top level.

At the Department of Commerce we are initiating a new program of installation reviews, to provide useful information not only to installation management but more importantly to the Assistant Secretary for Administration, the top Commmerce official responsible for the management of computer resources. A three-phase approach will be used. First, all data processing installations will complete an annual questionnaire about their operations. Second, several installations will be selected each year for on-site reviews. Other, more technical evaluations may be made as necessary. Finally, using the summary information provided by the bureaus and based on the installations' questionnaire responses, an annual report will be produced for the Assistant Secretary.

Key words: ADP management, computer performance evaluation.

1. Introduction

We know how to evaluate one data processing installation, but how do we evaluate all installations, and on an annual basis, for top management?

This is the question we wrestled with at the Department of Commerce. The Office of Procurement and ADP Management (OP&ADPM) is charged with evaluating all installations and making reports and recommendations to the Assistant Secretary for Administration, the Department's top ADP official.

2. What We Did in the Past

The Department first began an evaluation program in 1976. The annual review schedule was set by the

Office of Audits because ours was an integral part of their overall review of a particular office. Typically one person from our office spent a week on-site evaluating the installation. Afterwards he wrote a report and sent it to the Assistant Secretary. On occasion the services of the Federal Computer Performance Evaluation and Simulation Center (FEDSIM) were utilized for a thorough technical evaluation while the Department of Defense Computer Institute (DODCI) was called in for ADP security reviews.

With a complete turnover of OP&ADPM's top management in 1979, the new managers felt the program needed revision, for several reasons:

- The program was limited by resources to only two or three installations annually.
- The focus was on the optimization of computer utilization, thereby excluding functions such as programming and areas such as adherence to Federal Information Processing Standards (FIPS).
- It did not take into account the ADP security requirements mandated by the Office of Management and Budget (OMB) in its Transmittal Memorandum Number 1 to Circular A-71.

Therefore, management decided a new approach was needed and laid down the following conditions:

1. The evaluation should be on an annual basis.
2. It should include all data processing installations within the Department.
3. Travel funds for on-site reviews will continue to be limited.
4. All functions of the installation must be reviewed, not just computer center operations.
5. All aspects of proper management ---security, budget, programming, user satisfaction---must be evaluated, not just computer performance.
6. The various security requirements, particularly those pertaining to reviews, must be included.

3. Putting Together the New Program

We felt the easiest, cheapest and fastest way to implement such a program was to do what we did with our ADP security manual: copy an existing one from another agency. Therefore, we contacted about a dozen other agencies including the General Accounting Office (GAO) and asked if they either had such a program or knew of any other agency which did. To our disappointment and surprise we found that no agency was conducting a program similar to the one we proposed.

We found most agencies had no program at all. Those that had programs did not suit our needs. For instance, one agency had a rather extensive computer performance evaluation program using hardware and software monitors. Another had a program conducted by its Office of Audits. Another had a comprehensive program but the results did not get to and were not intended for the agency's top management. Another looked at a specific topic such as ADP security but neglected what we thought were equally vital areas. Others did it on a sporadic basis, very similar to what we had done in the past. Another agency performed a review immediately prior to the installation of any new computer system. Some had a program which evaluated computer center operations but neglected such areas as top ADP management, programming and analysis and those users who did not have their own computer but instead had terminals connected to someone else's computer. Even consolidating all these programs would still not have yielded a program aimed directly at top management.

Consequently we had to come up with a totally new program. For sources we used the various questionnaires we obtained from those agencies having some kind of review program, the audit program used by our Office of Audits, several volumes of the Faim Technical Library, magazine articles, GAO pamphlets,

conversations with other professionals and our own thoughts as to what information would be useful.

4. Objectives

The objectives of the program are to ensure that all applicable regulations, particularly in the areas of security and standards, are being adhered to, to determine how effectively our office is doing our job and what we still need to do, to disseminate knowledge of problems and their successful solutions throughout the Department, to provide our office with very basic information about all our installations and to identify problems that cannot be resolved by installation management or the various bureaus.

We want to evaluate every data processing installation in the Department. We generally are following the guidelines of OMB Circular A-71, Transmittal Memorandum Number 1, as to what constitutes a data processing installation. Since their guidelines encompass nearly everything and everyone involved with data processing, our evaluation program takes into consideration the wide variations we have in Commerce.

5. Proposed Evaluation Program

By consulting all our sources we came up with a program that has three distinct parts: an annual Commerce-wide questionnaire, on-site reviews of selected installations and summarized reports for use by the Assistant Secretary.

The first part is a questionnaire consisting of approximately 250 yes/no questions and 25 questions requiring narrative answers. The 25 narrative questions cover such items as history of the installation, unusual conditions (proximity to volcano, area prone to flood or earthquakes, etc.) and the mission of the installation.

To be completed by each data processing installation the questionnaire has sections for top ADP management, computer center management, remote programmers (not under direct control of this installation's management), other miscellaneous kinds of installations and technical and non-technical user satisfaction. Following is a sampling of questions.

For ADP management:

What is the stated mission of this ADP installation?
Discuss the reasons for any overtime incurred over the past year.

For computer center management:

What programs and procedures do you have to continuously evaluate the performance of the center's hardware, software and services?

Are console operator printouts reviewed daily to detect equipment and operator problems?

Is the equipment being used for the purpose given in the justification statement when it was acquired?

Has the fire department assigned to service this installation been notified that this is a data processing installation?

Are tapes in the tape library stored vertically?

Have turnaround standards been established for various classes of jobs?

Are supplies of critical forms stored off-site as backup?

Are procedures for using the telecommunications system documented for the users?

For systems and programming management:

Is the testing of new programs done by someone not involved in the design or programming of the new program?

Is there more than one copy of program documentation?

For terminals connected to this computer system:

Is there a portable fire extinguisher

within 50 feet of all data processing equipment?

For users:

Describe your workload (cyclical or erratic, peak times, business or scientific, etc.).

Do you maintain any manual system as a supplement to your automated system?

Have you ever received a copy of the computer center's user's manual?

The answers to all questions will first go to top management in the particular bureau. These people will summarize the information given by each installation for each of the yes/no responses and retain the completed questionnaires. These summaries will be responses to our general questions asked of each bureau's top ADP management. A general question may be "What problems have been found in implementing measures to detect fire and water hazards in the computer center and to protect the equipment from damage by these hazards?" The specific questions would ask if (1) there is plastic sheeting near the equipment, (2) fire extinguishers are present, tested regularly and easily accessible and (3) all operations personnel are instructed as to what to do in an emergency. Bureau top management consequently will have to become actively involved in this evaluation program, which they should be, of course. Bureau ADP management will also tell us how they plan to attack any problems they can solve at their level and will make recommendations on problems which can be solved only at our level.

Consequently, at headquarters level we will receive only the responses to the general questions as summarized by each bureau and the narrative questions pertaining to each individual installation.

Because these levels of departmental ADP management---installation, bureau and headquarters---are involved, each level becomes aware of the problems created by and which can be resolved at that level. This involvement by all levels of ADP management follows the philosophy that management action should be taken at the lowest possible level.

The second part will be on-site reviews, similar to what we have done in the past. Because we will have a considerable amount of information including that installation's latest responses to our questionnaire before traveling to the site, we should be able to conduct the review in a time period shorter than under the previous program. It is possible that reviews of smaller installations will be conducted over the telephone rather than by an actual visit. The selection of sites will be somewhat subjective based on importance, size, impending acquisitions and other such factors. A report for each site will be sent to the top ADP management in the bureau involved. The purpose of the on-site review is to ensure that both our questions and the installation's responses are valid.

The third part will be the production of reports for use by the Assistant Secretary. We do not intend to issue "report cards" on each installation. Since the Assistant Secretary is interested in the overall performance of all the Department's installations rather than in any single installation, we intend to produce summary reports. We will particularly highlight those problems which cannot be resolved at lower levels or which need additional emphasis at these higher levels. For instance, if we find many bureaus saying that the procurement of ADP resources is too slow, we will recommend actions to be taken at Departmental level to speed up the process.

6. Implementation

The implementation of the program will take a year or more. We have produced a draft questionnaire; the next step is to get active user involvement in putting it into usable form, i.e., adding or deleting questions, changing the method of implementation, etc. We will then publish a version for use with the testing phase in which 8 or 10 installations of varoius types will take part. The program will be further modified based on the results

of the test. At the same time the non-test installations will get a copy of the questionnaire (a) to prepare for the time they too will have to complete the it and (b) to allow for their further input. After evaluating the results of the tests, the program will be modified again and the questionnaire sent to all data processing installations for them to complete in earnest.

We hope to provide the Department's top ADP official with the kind of information needed at that level to effectively manage the Department's ADP resources. We also hope to enable all our installations to increase their effectiveness and to help them comply with Federal and Departmental requirements. We also expect to find out where this office can most effectively apply its limited resources. Since we do not have the benefit of others' experiences to draw on, we are relying heavily on involvement by users for help and guidance. With their cooperation we believe this program can be of benefit to them as well as to top management.

The Air Force Base Level

Computer Performance Management Program

John K. Graham, Jr., TSgt, USAF

Office of ADPS Management
Air Force Data Systems Design Center
Gunter AFS, Alabama 36114

The Air Force base level Computer Performance Management (CPM) program is 10 years old this year. More than 100 operating locations around the world are within the scope of the program. This paper describes the history and current status of the program. It also speculates about the future course of Computer Performance Management within the base level environment.

Key words: Air Force Data Systems Design Center; CPM Project Officer; Computer Performance Management Technical Center.

1. History

In 1969 the Air Force began converting its base level computer support from the Burroughs B263 to the Burroughs B3500 computer. The B263 was an early second generation piece of equipment with four thousand characters of main memory and no mass storage devices. The only peripherals were reader, punch, and printer. The B3500 is a third generation, multi-programming computer with disk and magnetic tape. The B3500 was supposed to provide the Air Force with trouble-free support for many years. It didn't. With the new hardware came new applications. Either the new applications or new hardware were incorrectly sized. Saturation was immediate at many locations. Compounding this problem was the lack of operations experience with multi-programmable hardware. Mix management seemed to be either run one job at a time or load the system as heavily as possible and let it thrash. Recognizing these problems, the Air Force Data Systems Design Center (see figure 1) began developing a CPM program. The scope of the program can be described in two ways. First, within the Air Force, there are 108 installations with about 120 Burroughs medium system processors. These computers support 14 Major Air Commands and Separate Operating Agencies (MAJCOM/SOA). Second, within each installation, instead of looking just at the hardware, the program also focuses on such things as software, personnel, and facilities, in short, the entire installation. Anything that can degrade a system's performance, whether it be a disk channel bottleneck, poor mix management procedures, or an inefficient air conditioner, falls under the scrutiny of the program.

In 1970 the Design Center purchased a D-7700 hardware monitor and trained analysts in its use. For the next year, the analysts evaluated different hardware monitors. The Dynaprobe D-7900 from COMTEN was selected and several were purchased. Our early CPE efforts were centered on the hardware monitors and on learning what the B3500 was capable of doing. Two travelling teams responded to crisis situations by conducting on-site studies. They also produced a series of Operations Research Reports that have since proved to be invaluable reference texts on B3500 capabilities. Although the support given individual installations was excellent, coverage was inadequate. Two teams could not respond to the needs of the entire Air Force. To solve this problem, the Design Center began developing software monitors. After several years of refinement, a monitor was developed that was the best

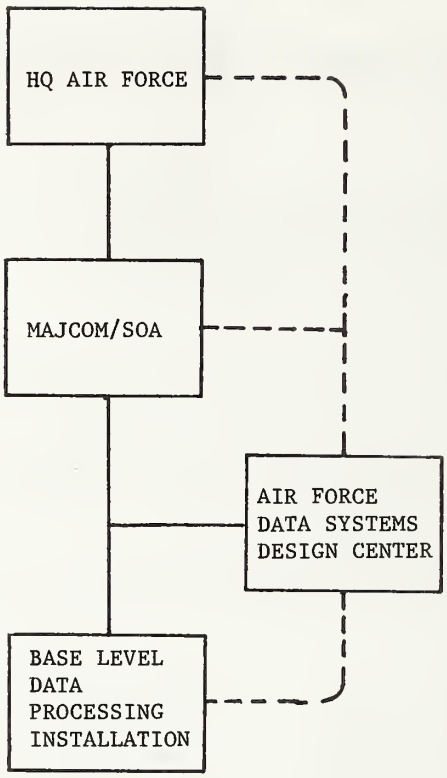


FIGURE 1. The Design Center reports to the Air Force Communications Command (a major air command). It communicates with and supports HQ Air Force, the MAJCOM/SOAs and the base level data processing installations.

trade-off between overhead and the comprehensiveness of the data collected. A great benefit was realized with the development of the software monitor. With just a little training, almost anyone with operations experience could perform their own studies.

After development of the software monitor, the direction of the CPM program began to change. While still performing studies, the teams began training personnel at all the sites they visited in the use of the monitor. A major step in the program's development was the identification, assignment, and training of "CPM Project Officers" from each Air Force MAJCOM/SOA (HQ SAC, HQ TAC, etc.). These project officers act as the managers of the CPM program for their headquarters. With the addition of project officers, the CPM program was beginning to take shape. There was, however, one last major obstacle to an effective program. The software tools and documentation for those tools were "bootlegged". That is, they were dropped off at the locations visited and were not a part of the standard release provided to the operating locations by the Design Center's Quality Control Directorate. Not only did we have a problem getting the monitor to the field but it was impossible to keep updated once it was sent out. We just couldn't remember who had which version.

In 1978 the Design Center released the software monitor, its reduction software, and the associated documentation as a portion of the standard base level software package. The standard release of the monitor removed the last major obstacle to our providing a tool for continuous use at the base level installations.

2. Tools.

The software monitor is an Air Force developed version of the Burroughs operating system. It is an event driven monitor that captures processor, disk channel, and disk device statistics. The data reduction program allows a great deal of flexibility in computations, date-time selection, and output formatting.

In addition to the software monitor, several other tools and techniques are available. One is a "snapshot" monitor that samples the active job mix. A series of tools provide reduction of the accounting log. The data collected by this method is used to compile a quarterly summary of use statistics, the Configuration Analysis and Projection System (CAPS). The Workload Analysis and Modeling System (WAMS) is a data base compiled from five years of these use statistics.

Sizing techniques are used to predict the impacts of new workloads. Through the use of sizing, it is possible to quantify new requirements and have equipment in place prior to implementation to meet those requirements.

Simulation and benchmarking are used to evaluate such things as new processors, new operating systems, and mix management procedures. A benchmark that closely represents a typical workload is being developed now. It will be used to evaluate several software and hardware options.

Models have been developed that are used in building the five-year master plan. The master plan predicts equipment requirements for the next five years. This plan is updated yearly and is used for budget requests to Congress.

3. CPM Applications.

Good tools are important, but in order to be worthwhile, they have to be applied in areas where they get results. The best results of the base level CPM program have been realized from:

- Disk balancing and tuning.
- Bottleneck analysis.
- Mix management.
- The hardware upgrade process.
- Special projects.

The application that has brought us the best return for effort expended has been disk balancing and tuning. Disk configurations at base level installations range from two channels with two devices to seven channels with nine devices. The devices are a mixture of fixed head and moving head disk. Disk balance is simply placing files on the devices so that each path to disk (channel/device) gets equal activity. This increases the probability of simultaneous I/O activity and reduces the time that the processor is idle waiting for disk I/Os to complete. Disk tuning applies only to moving head disk. It is the placing of files on each device so that the highest activity files are contiguous. This reduces read/write head movement to a minimum. When the disk system is loaded with no regard to file placement, device times range from 40 to 50 ms. On properly tuned disk, device times are reduced to a range of 29 to 36 ms. This results in greatly

reduced I/O queue times. The combined results of effective disk tuning and balance can be impressive. User response time can be decreased by 20 to 30 percent with a corresponding increase in throughput. Each Air Force base level installation is strongly encouraged to initiate a regular program of disk tuning and balance and take periodic measurements to verify the results.

When any component of a system restricts the performance of that system, a bottleneck has developed. Without detailed performance data, identification of a bottleneck is reduced almost to guesswork. A disk contention problem is sometimes seen by operations personnel as one of not having enough memory to get the job done. Had memory increases been made in these cases, it would have allowed more jobs in the mix to compete with the already saturated disk access resources. The end result would have been either no improvement or a further degradation of throughput. Accurate evaluation of a performance problem requires detailed information. As an installation develops some expertise in performance analysis, they are able to identify system bottlenecks before they degrade performance.

A computer operator can generate up to 30 percent processor overhead by mismanaging the job mix. He does this by overloading the job mix and changing priorities, causing program swapouts. One major emphasis of the CPM program has been to learn the best ways of managing the job mix, then training as many operators at as many locations as possible.

The computer upgrade process in the Air Force is rather involved. The basic philosophy is that each installation should have enough equipment to meet mission requirements without wasting resource dollars. When requesting an equipment upgrade, the installation is required to complete a CPM study that shows:

- The requested equipment is needed.
- The requested equipment will correct the problem.
- The current equipment is well managed.

This upgrade process has generated a much improved ratio of dollars spent to information processed. Upgrades are now more orderly, more predictable, and more effective.

"Special projects" covers a wide range of activities. They include evaluating new operating systems, new pieces of hardware, and evaluating the impact of new functional software. Benchmarking or simulation are usually used for these evaluations.

4. Todays Status.

A wide range of applications are served by the base level CPM program. For these applications to be well served, an effective organizational structure is necessary.

The Air Force CPM program has been formalized under direction from Headquarters Air Force (see figure 2). The program is structured with a single focal point for all performance management activities within the Air Force. The focal point is the communication link between headquarters and Computer Performance Technical Centers (CPTCs). These are the managers of the CPM program; each is responsible for one hardware type. Each Major Air Command/Separate Operating Agency has assigned a CPM Project Officer. He is responsible for providing the CPM support for his command. The CPTCs provide training and assistance to the project officers as required. The directive also establishes a requirement for upward reporting and a historical data base of performance data. This need is partially filled by the CAPS and WAMS described earlier.

The CPTC concept is working well. The structure is in place and the base level CPM Project Officers are all trained. Information is being shared and passed through the use of newsletters, working groups, seminars, and assistance visits.

The AFDSDC has developed a "CPM Guide" for base level installations. This guide is an introduction to computer performance management for operations personnel. It is also used as a text for training seminars. The guide covers:

- Developing a CPM study plan.
- Disk balancing and tuning.
- Disk file management.
- Use of the software monitor.
- Job mix management.
- The upgrade process.

Although designed as a stand-alone tool for operations personnel, the guide is most effective when used with training seminars.

Training seminars are given at irregular intervals by either the CPM project officers or CPTC personnel. The sessions last about four days. They cover the material in the CPM Guide and there are discussions on the

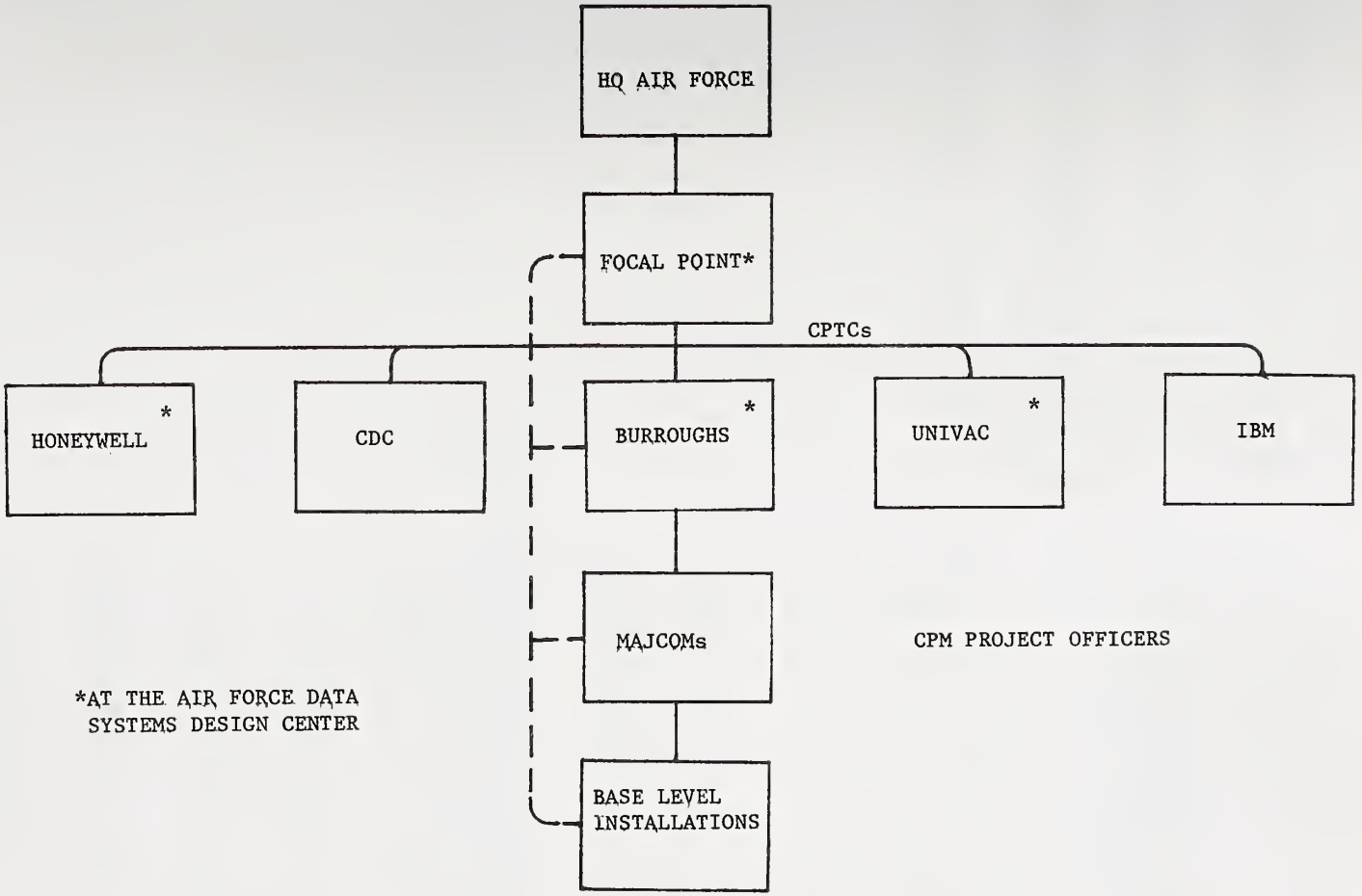


FIGURE 2. Computer Performance Technical Center (CPTC) Structure.

operating system and the Air Force developed data communications handler. Training will continue to be a major objective in the future.

5. The Future.

Major emphasis of the CPM program in the near future will be training. As the current base level hardware becomes older and more heavily used, good management practices become increasingly important. As more people are trained at the operational sites, the Design Center, as CPTC, will be less involved in the day-to-day operation of the program. The CPTC's role will be one of advisor.

The long-term future is exciting. The Phase IV capital replacement program for base level hardware is becoming closer to reality. The Phase IV program is the largest computer acquisition ever undertaken by the United States government. This will give the base level performance analysts a whole hardware series to measure, evaluate, and learn. Planning for the CPM program to support the new hardware has begun. Lessons learned in the past should prove valuable in developing this program.

In the ten years the program has been in existence, there has been one lesson that has constantly repeated itself. To be effective, a computer performance management program demands an ongoing commitment from people at all levels within the organization.

6. Glossary.

Air Force Data Systems Design Center (AFDSDC) - Analyzes, designs, develops, tests, implements, and maintains standard automated data systems. Acts as manager for hardware common to more than one MAJCOM/SOA.

Configuration Analysis and Projection System (CAPS) - A quarterly summary of machine performance statistics.

CPMe - Computer Performance Measurement. The use of specialized tools to measure system performance.

CPE - Computer Performance Evaluation. The analysis of the data collected in computer performance measurement.

CPM - Computer Performance Management. The application of evaluation results to the operation and management of Air Force hardware.

CPM Project Officer - An individual designated by a MAJCOM/SOA to act as focal point for computer performance management.

Computer Performance Technical Center (CPTC) - The focal point of the CPM program for a hardware type.

MAJCOM/SOA (Major Air Command/Separate Operating Agency) - The major breakout in the Air Force organizational structure (HQ SAC, HQ MAC, HQ TAC, etc.).

Phase IV - The capital replacement program for base level hardware.

Standard Release - The monthly release of standard software (object code) and documentation from the Design Center to installations worldwide.

Standard Software - Software common to more than one Air Force command.

CPEUG80

Prototyping/Benchmarking

RTE's - Past Is Prologue

Mitchell G. Spiegel

International Computing Company
4330 East-West Highway
Bethesda, MD 20014

This paper surveys the evolution of Remote Terminal Emulators (RTEs). Major developments in RTE technology are separated into three "generations" of products. Each generation's unique applications and features are highlighted. Recent developments are noted and a prediction of future use for RTEs is provided.

1. Introduction

Remote Terminal Emulators (RTEs) first appeared as sophisticated system drivers in manufacturing and university[1][1] laboratories more than 10 years ago. Aside from the occasional report of a study done using an RTE, the device has remained a laboratory curiosity. RTEs have not been widely used because of the high cost of their use, lack of portability, requirement for sizeable resources for stress testing, and unavailability of trained personnel. RTEs have acquired a reputation for being useful primarily in the context of stress testing systems. A literature search performed as part of the research for this paper indicates that RTEs enjoy robust application to a wide variety of problems. The RTE is entering its "third generation" of design. Whereas early RTE systems had many undesirable qualities, and "second generation" systems were not generally available, the new generation of RTEs is expected to alleviate cost, portability, resource, skill, and image problems and open many new areas of potential use. The RTE likely will become a keystone for system quality assurance efforts. (In this paper RTEs are not differentiated by the manner in which test workloads are imposed on the system under test (SUT) -- many approaches have been used to drive systems with remote terminal generated work including: central processor (CP) devices that reside in the host, also referred to as simplex drivers; front-end processor (FEP) drivers that reside in the communication control processor; message processor (MP) drivers that reside in concentrators or network switches; remote set simulator (RSS) drivers that reside in a microprocessor attached to the terminal-line interface or in terminal or cluster controllers; data pipe (DP) drivers that reside in remote job entry systems; and remote terminal emulator (RTE) drivers that reside in a separate external computer system and are capable of replicating the entire network as it is connected to the SUT.)

[1]Figures in brackets indicate the literature references at the end of this paper.

2. The "First Generation"

"First-generation" RTEs were developed mainly as debugging systems that used simplex drivers, individual terminal simulators, and/or data pipes. A few RTEs were built in the laboratories of several

computer manufacturers to test systems that supported large numbers of mixed terminal types. Reasons cited by Honeywell [2] for developing its Honeywell Communications Environment Emulator (HCEE) were to aid in the checkout and debugging of communication software; permit experimentation with software and terminal configurations (although the terminals did not actually exist); and evaluate both the operational limits and stress-load responses of on-line software. In addition, Honeywell felt that conventional test techniques (such as checking each transaction path, generating huge quantities of test data and stress testing with manned terminals) were time-consuming and impractical. More importantly, conventional tests violated a prime rule of scientific experimentation, i.e., reproducibility of test conditions.

IBM also mentioned the importance of the reproducibility of test conditions in its chronology of experience gained from its first virtual operating system [3]. The manufacturer's view of on-line system viability was and still is a complex relationship between cost, performance, and the value of system functions to the using organization. Fixed values were established for system ownership cost (measured in dollars expended over the life cycle) and function value (measured in the percentage of productivity improvement per user). The viability of the system was determined by the number of users that could be supported at acceptable levels of system performance (e.g., such service objectives as response time by application and availability). Typical on-line service objectives were specified and then compared with measures obtained by executing a series of RTE runs under controlled, reproducible conditions. Results indicated the number of users that could be supported by a given configuration. An interesting finding of IBM's virtual storage operating system development group was that, prior to the use of the measurement driver, more than two-thirds of all software errors in system releases were discovered by customers. Subsequent to the use of the measurement driver, the ratio reversed, to a point where three-fourths of all operating system software problems were detected before release by the Corporation.

Design characteristics of Honeywell's first generation RTE were as follows: The HCEE simulated up to 63 lines (8-terminal), with both asychronous and synchronous (2400 bps) terminal types. Maximum output of up to ten 100-character I/O pairs per second could be generated. The initial software was implemented on a Honeywell 1200 with 64K bytes. Resident software required about 30K bytes. By contrast, the IBM Measurement Driver handled up to 256 active asynchronous terminals on eight lines. Output was approximately 12 I/O pairs per second. The initial software was implemented on an IBM 360/40 with 128K bytes.

User exposure to drivers was limited to such tools as those supplied to users of IBM's Passenger Airline Reservation System (PARS). A unique aspect of the PARS software support accompanying the test tools was a system test methodology to assure quality at any point of the system development/operation process [4]. This methodology was an important feature because of the large number (hundreds to thousands) of terminals on-line in a typical PARS system. The RTE-like components and their relationship to the system test methodology were (1) a system test compiler (STC) used in all phases of testing, beginning with the unit test to create test records, programs, and messages; (2) a Remote Set Simulator (RSS) used to measure the behavior in a batched mode of individual or multiple terminals during application package test; and (3) a system test vehicle (STV), similar to the RSS, activated during stress tests from remote terminals in conjunction with a prerecorded scenario tape loaded on the System Under Test (SUT). All three tools were also employed during post-cutover testing to detect problems and to re-establish a baseline after changes had occurred. RTEs were made available in IBM's laboratory to perform stress testing for sizing various PARS configurations and to demonstrate the ability of a proposed configuration to meet its customer's service objectives.

The PARS stress test demonstration of performance was extended to other software product lines in the early 1970's by IBM for selected large on-line customers. The hardware/software configuration that successfully passed the customer's performance objectives served as a pilot/prototype vehicle for the general design and feasibility stages of the customer's system development cycle [5]. The system test compiler (STC) function was expanded from the PARS implementation to allow the user to specify in detail the characteristics of an application's

interface with the operating system and data base, and to specify the application's consumption of processor resources. The pilot/prototype approach was limited, however, by the lack of availability of RTEs in the user environment. Users had to form an evaluation team with the manufacturer's technical staff to develop scenarios for workload, applications, and data bases. Tests of various workload levels were performed at the manufacturer's benchmark facility. Limited test time forced many compromises on the experiment's design.

3. The "Second Generation"

The development of the minicomputer marked the beginning of the "second generation" of Remote Terminal Emulators. Whereas previous RTE implementations required manufacturer benchmark facilities, minicomputer RTEs were small enough to be installed at a customer's site. In 1972 and early 1973, two design-verification models of an RTE were developed by the MITRE Corporation under the sponsorship of the Air Force [6]. The first such model, located at MITRE/Bedford, MA, was a fixed-site system, used primarily for program and scenario development, and interfaced with the SUT through the switched telephone network.

The second design-verification model was an on-site system, used primarily for detailed emulator test and evaluation. The on-site system was conceived as a prototype RTE to be used in future ADP system procurements. The on-site system interfaced through cables directly with the SUT's communications line adapters.

The primary hardware component of both MITRE systems was a Data General Nova 800 minicomputer configured with a fixed-head disk, magnetic tape, TTY, and SUT interface unit. The system could emulate up to 16 low-speed asynchronous terminals. The on-site system had the additional capability of emulating 8 high-speed (up to 2400 bps) synchronous lines. Software included a macro pre-processor, scenario assembler, real-time executive, scenario interpreter, and data reduction program. In practice, the on-site system proved to be more difficult to relocate and return to an operating condition than had been expected, primarily because of the very complex nature of the RTE-SUT interface [7].

Custom-built RTEs were implemented for minicomputers and delivered as part of an overall system architecture in cases where the RTE was integral to the operation and maintenance of the on-line system. An example of such an RTE was the International Computing Company's Network Exerciser, delivered in 1973 to a commercial customer as part of an on-line transaction system for their clients. The RTE was part of a distributed minicomputer network, serving the vital function of determining real-time problems in the network software. Client response time requirements of less than one second, in addition to throughput rates of over 100 transactions per second, were common.

An early use of a customized RTE for evaluating time-sharing services was made at the Bell Laboratories. During the mid-seventies, Bell Laboratories spent between $50 million and $75 million on time-sharing services. To obtain satisfactory service from its time-sharing vendors, Bell constructed an emulator for a DEC PDP-11/45 system and operated it similar to MITRE's fixed-site RTE. Each time-sharing service vendor was required to pass a live-test demonstration by repeating a series of tests controlled by the RTE over a two-week trial period. The results of the tests were documented for both performance and charging algorithm baselines [8]. Services were retested at unannounced intervals (about every six to twelve weeks) to verify that the service was within contract terms.

The National Library of Medicine (NLM) used a similar approach to evaluate its MEDLINE services. The MEDLINE Simulated User System (MSUS) resided in the front-end computer (XDS940) and was limited by the number of terminals that the front-end could emulate without degrading system performance. Stress tests were performed at NLM by using multiple front-ends to drive the SUT. The NLM RTE functional and stress test capabilities were one of about a dozen RTE approaches unearthed during a survey of manufacturers and Government agencies [9,10].

In the mid-1970s, computer manufacturers upgraded their RTE capabilities at their benchmark test centers by using more efficient and flexible software. The upgrade enabled manufacturers to drive SUT's with very large networks of terminals and terminal combinations of greater variety. A rule of thumb for sizing RTE system requirements among manufacturers estimated that an RTE

one size smaller than the SUT would be able to "saturate the SUT" without falling behind the scenario rate established for the test.

Because of the availability of RTEs at manufacturers' benchmark facilities, and the large Government investment in on-line systems and time-sharing services [11], a workshop on RTEs was held jointly by the National Bureau of Standards (NBS) and the General Services Administration (GSA) in 1976. The early results of a procurement projection study were presented at the workshop [12]. The data presented indicated that the volume of procurements for on-line systems and the number of remote terminal devices would generate a need for a system-independent workload generator that could be used to evaluate vendor proposals. The related technical issues were subsequently made available in a later report [13]. In addition, a major government case study involving the use of four vendor RTEs was described [14]. GSA presented a plan for incorporating the use of RTEs in the Federal ADP procurement process [15]. The plan was closely adhered to by GSA, culminating in the successive production of three documents: (a) RTE Specifications, (b) a description of the use of RTEs in Federal procurements, and (c) a handbook [16] combining the two previous documents. This handbook presented in a compact format rules for temporary Federal Procurement Regulation 49 on the use of benchmarking and remote terminal emulators for performance validation in the procurement of ADP systems and services.

About the time the Government began its examination of RTEs, the manufacturers publicized for customer evaluation the use of such devices in complex system environments [17,18]. RTEs in manufacturer's benchmark facilities provided customers with measures of the sensitivity of a given hardware or software configuration to changes in the workload [19,20]. Customers were surveyed to determine the environment for specific industry uses of certain software packages (i.e., operating system, on-line application control system, data base management system, and network control system). The results of the survey were used to construct synthetic representations of the customer's aggregate workload. Performance measures were taken for various increments of workload to determine system sensitivity to mixtures of workloads on given hardware and software configurations.

The emergence of computer networks provided a further opportunity for second-generation RTE applications. Early networks were designed, tested, and operated with RTEs attached. Most early developments were related to digital networks. An early example was the evaluation of the packet-switched subnetwork CIGALE, the basis for the CYCLADES network which connects research centers and universities in France [21]. Studies addressed throughput rates for different line control procedures and maximum throughput and transit delays for packets transmitted under various loading conditions.

Analytics, Inc., developed a driver that was employed both as a network tester and as an RTE to evaluate a SUT. The driver, installed at one of the locations of the Worldwide Military Command and Control System [22,23], enabled the staff to reduce the time required to certify new releases of Honeywell's GCOS operating system from 200-400 man hours to about 20 man hours. The RTE was also able to drive the network with SUT output responses and to receive inputs from other network nodes. The SUT emulation capability permitted functional tests of multi-level protocols between hosts, as well as more conventional throughput and delay exercises.

Public packet-switching networks generated further functional demands for RTE-like network exercisers. In France, the TRANSPAC experimental network required a tool to provide protocol conversion, frame error detection, call establishment and clearing, and a terminal packet data generator/absorber [25]. Users were given frame trace capabilities, a list of errors caused by their software; statistics on about line errors; frame suppression capabilities to simulate errors; the ability to activate frame generation; and the flexibility to establish calls, vary messages sent, and detect error conditions. The products that resulted: REX25; ESOPE; and SIMAD; have helped users to develop interfaces with TRANSPAC. REX25 was implemented on a MITRA 125, a powerful minicomputer. ESOPE and SIMAD were implemented in a Datapoint 5500 RJE station. A follow-up emulator was implemented for the Canadian public packet-switched network, DATAPAC.

Other network testing devices also received attention during the same time period. Both IBM and Bell Telephone Laboratories developed digital devices for

testing analog networks in order to replace such complex analog test equipment as oscilloscopes. When an analog signal arrives at a modem, it is decoded into digital data, which is subsequently presented to a communication subsystem (e.g., a front-end communication processor, transmission control unit, or host computer). Measuring is done by removing the line from service or attaching analog measuring equipment to the line, modem or associated interface. The operations staff must know when to measure and often has difficulty identifying intermittent errors. Special skills are required to read analog measuring equipment.

Another approach [25] uses the extra information about the line's performance (the parameters of the analog signal as received). The analog data are passed to a line quality monitoring (LQM) program in a separate computer, which analyzes them and creates appropriate summary information. The network location of the line quality monitor computer is the same interface point normally chosen for installation of an RTE (in front of the communications front-end processor). The LQM approach offers the following advantages: (1) the line stays in service during the monitoring; (2) the data collected can be a permanent record of line performance; (3) the monitoring is continuous, allowing for identification of transient conditions; and (4) such an LQM process can be linked to a network operating system that performs dynamic routing assignments and network maintenance control. One installation's experience with the LQM approach was a prime shift circuit availability increase from 82 percent to 99 percent within one month after LQM began. Over the first three months of operation, 17 of 21 reported line problems were confirmed as correctly diagnosed. The monitoring also proved useful in diagnosing a number of problems in telecommunication software.

4. The Third Generation

The general problem of network managment has served as a unifying force to relate the RTE, LQM and other measurement and quality assurance functions. A unique approach evolved from the PARS System Test Vehicle [26]. Underlying the approach is the interrelationship between network operations and network development in order to maximize user satisfaction (i.e., to provide successful delivery of services and meet user objectives). The RTE and related network and SUT measurement applications are integral parts of network "process" control. Such project activities as network design, hardware and software configuration, and capacity planning interface with the data obtained from on-line measurement and evaluation at the network interface. The "sidestream" processor, a prototype management tool, has been developed to evaluate the automation of many labor-intensive management activities associated with large on-line environments [27]. As a separate tool, the sidestream processor can be inserted into the communications system's operational process without disturbing the on-line applications. The sidestream processor has the potential of merging RTE applications with other network management functions in a single system.

Third-generation RTEs fall into two major categories: (1) microcomputer based, economical versions of small, single-purpose systems that were previously minicomputer based and (2) multi-purpose systems, such as the sidestream processor, that integrate the RTE functions with related network management functions demanded by users and managers. Examples of the former category are under development by Analytics Inc., (e.g., a Honeywell Level 6 version of the ASE System) Computer Sciences Corporation (e.g., Data General (DG) MP-200 version of the MITRE RTE) and Logica Limited Instrumentation System (e.g., A Texas Instruments 990/10). Examples of the multi-purpose systems are under development by International Computing Company (e.g., A Univac V7700 Network Exerciser delivered in January 1980 as part of the NASDAQ system) and contemporary manufacturers' large terminal systems (e.g., IBM's Teleprocessing Network Simulator (TPNS) used on IBM 4300 series systems).

Major features of each third generation approach follow. The Analytics ASE Level 6 system is designed to handle large stress test environments by combining multiple Level 6 systems into a large parallel system. Communication interfaces are similar to the current ASE implementation.

Computer Science's approach permits an extremely portable product, weighing less than fifty pounds in total (power supply, CPU interfaces, etc.). The prototype DG MP-200 system can emulate up to 16 asynchronous terminal scripts based on the same scenario. Available protocols are only limited by the flexibility of the DG

communication, access manager and communication control hardware. Synchronous communication support (bi-synchronous) is under development.

Logica, Limited designed their Communications Environment Generator (CEG) as a hierarchical multi-microprocessor system in response to a set of requirements issued by the Central Computer and Telecommunications Agency (CCTA) of the British Government. A Texas Instruments (TI) 990/10 minicomputer and associated DX10 software serves as the "master" control in the system. The "master" is connected to up to seven TI 990/5 microcomputers, which in turn can be connected to up to nine TMS 9900 microprocessors that function as micro-programmable four-channel controllers (FCCC). RTE software functions are allocated across all three levels of the system. An initial configuration of the 990/5's and four FCCCs can drive 48 full duplex lines at 4800 bps or 36 FDX lines at 9600 bps. Throughput rates of 15 I/O pairs per second can be sustained. Protocols are currently restricted to Britain's ICL product line.

The International Computing Company Network Exerciser delivered to NASDAQ performs all of the common RTE functions - script generation, logging, data reduction time-stamping, synchronous and asynchronous line handling - while providing capability to sample the common carriers' lines and provide for network problem determination. The UNIVAC V7700 configuration is capable of handling up to 48 synchronous lines operating at 9600 bps and sustaining stress rates in excess of 50 I/O pairs per second. The software is written in a portable language (INFOS) that can be operated on five other minicomputers. An advanced third generation design is in development to incorporate other service quality assurance functions in a highly reliable "non-stop" hardware architecture.

The Naval Personnel Research and Development Center (NPRDC) obtained a RTE design for a specific workload of up to 64 interactive remote terminals under contract N6600-79-C-0406. The hardware configuration consists of a DEC PDP-11/34 with a 300 mb disc and a high performance microprocessor from Plessey called a MIPROC. The PDP-11/34 is used for compiling scripts and editing the output to readable form off-line. The PDP-11/34 and the MIPROC run the actual real-time evaluation. They are connected by an asynchronous control channel and a direct memory access channel to transfer script information from the PDP to the MIPROC and to transfer logging output from the MIPROC to the PDP. The MIPROC interfaces with the SUT to perform the most time-critical tasks.

McDonnell Douglas Automation Company has developed a simplex emulator that can drive an IBM 4300 system. The emulator is written as a CICS application, using the timing features of CICS to perform the time-critical functions. Current plans call for an extension to full-duplex emulation capability.

Announced, off-the-shelf manufacturer RTEs have recently added enhanced scripting capabilities and the ability to function in both simplex mode and as true RTEs. IBM's script generator for TPNS produces scripts from TSO, IMS and CICS trace tapes.

5. Summary

Although technology has removed many of the barriers to successful use of RTEs, some problems remain. Because of the sheltered status of RTEs during the last decade, there are very few trained personnel that can successfully accomplish an RTE study. Problems with RTE use have been documented [28,29], and the ever-present benchmark problem of lack of correct scientific method is especially prevalent in RTE environments [30]. Further impediments to RTE use are user organizations that are still structured around a production environment, rather than a service environment. The problems will be overcome slowly over the next decade. Although the 30-odd references appended to this paper represent the bulk of the published knowledge about RTEs, many ad hoc studies have been done at the manufacturer's benchmark facilities.

Since TPNS became a licensed product, about 100 of the 3,000 installations that operate medium or large scale DBMSs have installed TPNS. Certainly a small percentage, but a large number in just a few short years. I believe the emphasis on service to the user, the importance of system quality assurance, problem determination, and the continuous interest in systems design issues will enhance the demand for functions an RTE can perform at its unique position between the entire SUT and the network [31]. The RTE will, I believe, become the basis for principal

CPM/CPE products for Computer and Communications Studies and for Network Management.

References

[1] Greenbaum, H., A Simulator of Multiple Interactive Users to Drive a Time-Shared Computer System, Report MAC-TR-54 (MIT, Cambridge, MA, October 1968)

[2] Caplan, R., Pearlman, J. M., and Snyder, R., A Communications Environment Emulator, Proceedings of the Spring Joint Computer Conference, 1969, pp. 505-512.

[3] Schwemm, R., Experience Gained in the Development and Use of TSS, Proceedings of the Spring Joint Computer Conference, 1972, pp. 559-569

[4] International Business Machines Corporation, Airlines Control Program (ACP) System Documentation, Vol. B, GS-20-1435-0, PARS Testing Philosophy, Rev 1, 9/13/68.

[5] Kirrene, M., and Spiegel, M., Design for Performance, CPEUG (15th Meeting), 1979, NBS, Washington, DC, pp. 129-140.

[6] James, D.L., A Remote Terminal Emulator for Loading and Performance Measurement of On-Line Systems, M72-83, Bedford, MA: The MITRE Corporation, March 1972.

[7] DeMone, E.C., Remote Terminal Emulator (Design Verification Model), Description of Hardware, MTR-2677, Vol. 8, Bedford, MA: The MITRE Corporation, February 1975.

[8] Wright, L., and Burnette, W., An Approach to the Evaluation of Time-Sharing Systems: MH-TSS, A Case Study, Performance Evaluation Review, ACM SIGMETRICS, Vol. 5, No. 1, January 1976, pp. 8-28.

[9] Watkins, S., and Abrams, M., Survey of Remote Terminal Emulators, NBS Special Pub. 500-4, NBS, Washington, DC, April 1977, pp. 1-71.

[10] Arthur, C., Remote Terminal Emulator Development and Application Criteria, 1977 National Computer Conference, AFIPS Conference Proceedings, Montvale, NJ, 1977, pp. 733-739.

[11] Davis, R., Welcoming Address, Summary of the NBS/GSA Federal Workshop on Remote Terminal Emulators, June 21, 1976, pp. 5-7.

[12] Kiviat, P., Procurement Projection Results, Summary of the NBS/GSA Federal Workshop on Remote Terminal Emulators, June 21, 1976, pp. 36-59.

[13] Wyrick, T., Concepts and Issues Relevant to the Use of Remote Terminal Emulator in Teleprocessing Procurements, Washington, DC: General Services Administration, May 1977.

[14] McFaul, E., RTE - A Case Study at the Geological Survey, Summary of the NBS/GSA Federal Workshop on Remote Terminal Emulation, June 21, 1976, pp. 122-133.

[15] Wyrick, T., and Findley, G., Incorporating Remote Terminal Emulation into the Federal ADP Procurement Process, CPEUG (14th Meeting), NBS, Washington, DC, October 1978.

[16] General Services Administration Handbook, Use and Specifications of Remote Terminal Emulation in ADP System Requesting, Washington, DC: General Services Administration, August 1979.

[17] Bailey, L., Experimental Evaluation of an Interactive-Batch System Using a Remote Terminal Emulator, CPEUG 1976 Conference Supplement, NBS Washington, DC, November 1976, pp. 111-122.

[18] Duke, M., Testing in a Complex System Environment, IBM Systems Journal, Vol. 14, No. 4, 1975, pp. 353-365.

[19] Currie, R., An Experiment in Synthetic Benchmarking to Predict MVS Performance for Broad Mix Systems, Slides from Proceedings of SHARE 48 Conference, SHARE, Inc., Houston, TX.

[20] Palmer, S., CS-1100 - Sperry UNIVAC Communications Simulator, CPEUG 1976, NBS, Washington, DC, November 1976, pp. 197-201.

[21] Eyries, F., and Gien, M., On-Line Performance Measurement in the

CYCLADES Network, EUROCON 77, Venice, Italy, May 1977, pp. 50-55.

[22] Shirey, R., Tools for Post-Programming Software Testing in a Dispensed Multi-Vendor Environment, 18th Annual ACM Technical Symposium, June 21, 1979.

[23] Analytics, Inc., ASE Computer Operations Manual, Naval Electronic Systems Command, Contract No. N00039-75-C0145, Technical Report 1152-25-TR-02, Washington, DC, December 1977.

[24] Giraudeau, P., Description of Services Offered by REX25, ESOPE and SIMAD for the Purpose of Establishing an X.25 Module Dialogue, CCETT, Paris, France, December 1977.

[25] Bryant, P., Giesin, F., and Hayes, R., Experiments in Line Quality Monitoring, IBM Systems Journal, No. 2, 1976, pp. 124-142.

[26] Giles, H., Successful Network Management Hinges on Control, Data Communications, August 1978, pp. 33-41.

[27] Leach, J., and Campenni, R., A Sidestream Approach Using a Small Processor as a Tool for Managing Communications Systems, IBM Systems Journal, Vol. 19, No. 1, 1980, pp. 120-139.

[28] Trehan, V., Problems in Remote Terminal Emulation, CPEUG 1978 14th Meeting), NBS, Washington, DC, October 1978, pp. 27-61.

[29] Tendoklar, N., Determination of Non-Steady State Conditions in Performance Measurement Runs, CPEUG 1977 (13th Meeting), NBS, Washington, DC, October 1977, pp. 87-94.

[30] Hyman, B., Stability and Workload Definition for Time-Sharing Systems, FIPS TG-13, Document 70, 1974.

[31] Wyrick, T., Benchmarking Distributed Systems: Objectives and Techniques, ICPCI 78, Gardonne, Italy, June 19, 1978, pp. 13.

Application Prototyping: A Case Study

C. Wesley Jenkins

Budget Analysis Division
Congressional Budget Office
Washington, D.C. 20515

Accurate specification of user requirements for interactive systems is especially difficult in an environment where the demand for information is intense, short-fused and largely unpredictable.

The Congressional Budget Office was created in 1975 by an Act of Congress. Its primary mandate is to serve the Budget and Appropriation committees of both the Senate and the House of Representatives. The Act also defined a Congressional Budget process specifying a calendar of events and specific completion dates for major activities. This pacing of budgetary actions produces a highly charged environment in which CBO must be able to respond immediately to information needs with information that is both accurate and consistent.

In approaching a redesign of some of these highly visible information systems, CBO decided to follow a strategy of prototyping these systems in order to faciliate the involvement of the user, highlight the user's real needs and to demonstrate the feasibility, effectiveness and any shortfalls of the proposed system before risking either a management or design commitment.

1. Introduction

The traditional approach to systems analysis seems to have serious limitations when applied to interactive information systems that are in a state of constant change and growth. Communications among the user, analyst and manager tend to be imprecise, a detailed analysis prolongs the process to the annoyance of the users, and specifications are either ambiguous or too voluminous to read. To compound this problem, the user is often requested to 'freeze' his requirements and subsequent attempts at change are resisted. The alternative of an informal analysis only increases the probability that the user will not be satisfied by the new system.

Approaching systems analysis and design around a strategy of prototyping offers a technique that minimizes the dangers of a long formal analysis and increases the likelihood of a successful system implementation. The essence of the prototype approach is the construction, test, and demonstration of a representation (model) of the system that is to eventually be developed. This methodology allows the user to interact with this skeleton version of his system and to be more involved in the design. The analysts and programmers can experiment with alternative system and data base designs. Managers can better gauge the expected performance and cost of the new system.

CBO decided that the potential benefits seemed well worth the initial investment of resources. The following case study of the experience at CBO includes a brief description of the background, an overview of the major steps leading up to the decision to prototype and a discussion of the prototype plans. Unfortunately, the publication deadline of this journal

precludes a description of the outcome of this prototyping experience.

2. Case Study Background

2.1 Instant Systems

The appointed director of CBO, Dr. Alice Rivlin, took office in the spring of 1975. In less than six months an effective organization of almost 200 individuals had been pulled together. The Budget Analysis Division was formed to handle three major functions: (1) providing cost estimates on all bills reported, (2) keeping 'score' on the Congressional budget process, and (3) providing five-year cost projections on the Federal budget. It was quickly decided that the last two functions required computerized systems to provide the necessary information in a responsive manner. Within a few short months the necessary information systems had been developed and implemented by a handful of CBO analysts with the assistance of contract personnel. The systems were installed and run in a service bureau environment. As time had not permitted analytical studies, these systems were considered necess ary stop gap measures to get up to speed in a hurry.

2.2 Nature of Demands

Serving the information needs of Congressional committees is a unique, and often frustrating, experience for the individuals involved in data processing support. Every request is a directive that must be answered immediately. Responsiveness is absolutely critical to the credibility and viability of an organization such as CBO. When information is needed, it is often needed immediately or the value of it is lost. Flexibility is also a must, as the type of information that is in demand fluctuates from Congress to Congress, chairman to chairman, committee to committee, year to year, week to week during the budget process and often hour to hour during the Budget Committee markup sessions. The third major characteristic of the information demands on Capitol Hill is the visibility and impact of the information delivered. Accuracy, consistency, integrity and security all are critical data issues.

2.3 EDP Objectives

In 1978 the MITRE Corporation completed a study of the data processing systems in the Budget Analysis Division. By this time the 'interim' systems developed in 1975 had become a gigantic collage. For the most part they were still providing the needed information in a timely fashion, but MITRE warned of a further collapse if the systems were not redesigned. They specifically recommended that a Data Base Administration function be established and that the new design should center around the use of a Data Base Management System.

The Budget Analysis Division took several positive steps during 1979. The first step was to provide a framework of long-range objectives and strategies to guide the planned redesign and development efforts. These objectives are highlighted below.

Long-Range EDP Objectives

- o Effective and efficient information services
- o Minimize data and system maintenance
- o Improve system flexibility
- o Simplify system use
- o Improve data accountability and management
- o Improve system transferability
- o Improve data integrity
- o Improve ability to monitor system performance

3. Initial Steps

Between June 1979 and June 1980 the following steps were taken. Each was significant in the evolution towards using 'application prototyping' at CBO.

3.1 Data Management Requirements Analysis

This task was intended to identify and define the requirements for a more effective program of data management in the Budget Analysis Division. An important objective was to describe the needs from three basic perspectives: manager, user and data processing analyst. There were several significant observations made during this study that confirmed the MITRE recommendations for system redesign and improving the methods used to manage the organization's data resources.

A primary concern was with the consistency of information supplied by the Budget Analysis Division to committees and other users, both on and off Capitol Hill. Management also identified quality assurance as a major consideration. The responsibility for updating and accuracy verification for each data item should be implemented as part of an authorization scheme that would ensure that only the designated individual would be permitted to alter that data item. Other factors involved in a program of quality assurance should include the ability to maintain audit and activity records in order to monitor usage and determine ineffectiveness, to track the logical and physical growth of the data base, to monitor system performance and to assure that adequate data validation is done.

Probably the most imperative requirement was viewed as the demand for timeliness in providing the information which is needed by a Congressional committee. This means knowledge of what data is available and how to access it is critical. It also means that the organization of the data base must be convenient and facilitate the ease of retrieval and report generation. This leads directly to a related management concern for flexibility in data structure. As the committee demands for information vary radically with the issues at hand, the point in the budget cycle and any number of external factors, the need to accommodate these changing views of the data is a primary requirement of management.

Users with data accountability responsibilities each expressed concern about the need for better verification and control techniques. There is a particular need for better verification of modeling outputs. A related requirement is for a simulation or test mode when doing data projections so that the new figures can be reviewed and accepted before the data base is actually updated. In conjunction with these concerns was a heavy emphasis on the requirement for consistency and the difficulties of accomplishing it when the data is passed so frequently from system to system, machine to machine, and one file to another. This redundancy of storage and the complexity of the data combine with a lacking of documentation and standards to create a worrisome situation for the people responsible for the data.

The programmer/analysts indicated a need to either have better information on all data files, reports, programs, and procedures or to have independence of the programs and procedures from changes in the data files. The programmers spend much of their time attempting to locate a program or programs that do this or that function in order to modify it or use it to create a different version.

The outcome of this study supported the MITRE recommendations. The concerns that surfaced definitely pointed to the need for a program of data management and the use of such tools as a data dictionary and a DBMS. The ideal environment for data management should provide the tools, techniques and procedures necessary for an organization to effectively and efficiently monitor and control the allocation and utilization of its data resources. This process often involves the recording and controlling of data about data in a single authoritative source. It does not necessarily depend on physical centralization of data files, but functional integration is preferrable in order to limit the negative consequences, such as inconsistency, of application independence.

3.2 Application Requirements Analysis

After the successful completion of the data management study, plans were made to begin a thorough analysis of user requirements in the major application areas. The first proposal called for the following type of information to be brought together in a workbook: (1) current environment description, (2) data description, (3) functional requirements, (4) management requirements, and (5) standards. This traditional approach was rejected because the paperwork appeared excessive, the resources required were substantial and the users objected to such a prolonged process.

An alternative approach was proposed that called for using the structured analysis methodology. This technqiue promised to provide a rigorous description of the new system with much less paperwork. This alternative was also rejected, however, as both management and users felt that it was a case of overkill for an organization the size of Budget Analysis Division; especially considering our limited technical support staff.

At this point it was decided to ask the Federal Computer Performance Evaluation and Simulation Center (FEDSIM) for advice. A representative of FEDSIM, Mitchell Spiegel, spent a few weeks evaluating our data processing performance problems and considering our known requirements. Mr. Spiegel's recommendations basically supported the MITRE conclusions and indicated a need to begin moving to a new data processing environment. He also confirmed the suitability of a DBMS for the type of data processing requirements that exist at CBO.

3.3 Decision to Prototype

Based on the FEDSIM and MITRE conclusions about the benefits of a DBMS, CBO decided on the following course of action. The user requirements for the major applications would be briefly defined, then an evaluation of DBMS packages would be undertaken. The DBMS that appeared to best meet the CBO requirements would then be leased for a 60-day period in order to verify the conclusions of the paper evaluation and to conduct performance tests. With the decisions about data base management systems, data dictionary, ad hoc languages, development languages and report generation tools completed, the plan would be to move rapidly into application design and development.

As this plan evolved, it became clear that the 60-day test period offered an ideal opportunity to bring up a model or prototype of one of the major systems needing redesign. As the

possibilities were discussed, it became clear that this approach provided several advantages. First, it was an excellent way to involve the user, to capture his interest and to obtain specific feedback on certain techniques that were being considered. Second, this approach would allow us to demonstrate the various capabilities proposed to the managers who are actually responsible for the 'go/no go' decision. Third, performance statistics and loading tests would provide excellent information for estimating the resource utilization impact and requirement of the proposed system. Forth, the technical team would be able to test alternative system designs and data base structures.

In short, application prototyping appeared to be an excellent approach under the circumstances. CBO decided to proceed with this plan.

3.4 Initial DBMS Evaluation

The basic application requirements were documented. Major user concerns were obtained through interviews and any significant data processing problems in the current EDP environment were described. Management supplied long-range goals and strategies to guide the process. Within a short time, a profile of mandatory requirements had been put together to be used in evaluating the DBMS packages available.

CBO started with a list of some twenty DBMS products. The list was quickly reduced to seven contenders, who were invited to present their packages. The information on each contender was captured on a standard form that addressed the areas of major concern to CBO. An initial evaluation eliminated four more packages. The remaining three contenders were examined in detail and evaluated within categories that had been assigned weight factors by management. This process resulted in clearly placing one DBMS above all the rest in its ability to meet the CBO mandatory requirements.

4. Prototype

With the DBMS selection decided, steps to prepare for the prototyping were started. The choosen DBMS was leased for a 60-day period, training for the participating programmers was arranged and the DBMS was installed at a convenient service center on Capitol Hill. A detailed test plan was written, the application to be prototyped was choosen and the scope of the prototype was defined.

4.1 Test Hypotheses

As the test period is of limited duration, it is important to specify in advance the hypotheses that are going to be tested. In the case of CBO, these hypotheses relate to both the application and the use of the DBMS and data dictionary.

(1) Use of a DBMS is in concert with overall BAD EDP objectives and responsive to user requirements.

(2) An integrated or active data dictionary provides important support to efficient and effective management of EDP resources.

(3) Use of a high-level, user-oriented programming language for the development of interactive applications is an efficient and effective method for reducing system development and maintenance costs.

(4) The cost analysts in BAD will use a high-level, user-oriented language to query data bases, write simple reports and perform certain types of generic updates to data for which they are responsible.

(5) Use of a CRT improves the ease of data entry and reduces the need for hard-copy reports by providing users with a convenient method of viewing this data interactively.

It is important to either confirm these positions or document any shortcomings or problem areas during the prototype. Management and technical decisions based on these hypotheses will have a far-reaching effect on future EDP developments at CBO.

4.2 Prototype Design

Several decisions had to be made about the scope of the application prototype before an actual prototype design could be developed. CBO decided to use the five-year projections system and to load the entire data base for the test. The DBMS development language and COBOL would be used in the prototype. The DBMS would be installed under CICS and full-screen, 3270-like devices would be used. It was also decided to load the data dictionary first and to utilize it as an active part of the application prototype.

The next major task was the definition of the application functions to be performed. One goal of the prototype is to demonstrate an effective method for accomplishing interactive data updating. This will include adding new records to either the data file or the reference files. The prototype will provide for changing several fields and for record deletion. These data updating functions will provide full-screen

capabilities, interactive editing of data entered and concurrent interactive updating of the data files by multiple users.

Other techniques of updating that will be tested as part of the prototype include interactive generic updating and batch updating of the data files with external data. The generic updating is a capability currently used in the projections system. It allows the user to specify a change for certain fields within a selected set of records. The following types of change will be permitted: zero, blank, add a field, and move a field. The batch updating will show the ability of the system to accommodate the addition of data from a sequential file.

Another objective of the prototype is to demonstrate selected interactive data retrieval techniques and to display the data retrieved using full-screen terminals. There are two primary approaches to providing the users with this type of capability: one involves guiding the user through menus, tutorial aids and carefully designed parameter selections and the other is through the use of a powerful English-like query language that a non-technical user can utilize to state his own requests. The first approach is similar to techniques currently used, but it becomes more convenient to the user with full-screen speed and more fleixible with a DBMS providing the retrieval capabilities.

The prototype will be designed to provide a menu-driven query capability that offers several retrieval options. Within these options there will be choices on what data the user wishes to display. The prototype will also provide a browse capability that will allow the user more flexibility in selecting retrieval characteristics and to combine these parameters with Boolean operators or range operators. In all of these, appropriate facilities of the full-screen terminals will be utilized.

The ad hoc query language will also be tested as part of the prototype. The technical team will exercise the ad hoc facility and measure its performance. Non-technical users will experiment with the ad hoc language in order to evaluate its ease of use, ease of learning and usefulness.

Another objective of the application prototype is to test and evaluate the report generation facilities available with the selected DBMS. This will involve producing three or four of the projection reports off the prototype data base. The usefulness and efficiency of the selected Report Generator facility in creating these reports will be evaluated. At least one or two of these reports will also be produced using COBOL. This will provide us with better information on the performance differences between the two approaches, the relative programming difficulties and costs, and the relative flexibility.

Other factors that will be evaluted via the prototype include:

- o Recovery from a system failure. This will include an analysis of the automatic recovery capability from both an effectiveness and efficiency point of view.
- o Ease and cost of reconstructing the prototype data base.
- o System utilities to backup and reload the prototype data base.
- o Flexibility of DBMS to accommodate changes to data structure. This will include adding new fields, adding different record types (new logical view), changing fields from non-key to key, changing a logical view, or changing the physical attributes of a field. These will all be analyzed as to their impact on the prototype programs already written.
- o Data dictionary facilities.
- o System utilization and performance reports.
- o Levels of security.

5. Conclusion

At this moment the technical team is putting down the final design specifications for the prototype. Report and screen layouts have been completed and all the programs to be written have been identified. The actual prototype test period begins the last week in July; one week from now. We are optimistic that this application prototyping strategy will provide the information needed to assure a successful redesign of CBO's major data processing applications.

Operating System Performance Meters

Performance Evolution in a Large Scale System

Richard S. Brice
J. Wayne Anderson

Los Alamos Scientific Laboratory
Los Alamos, New Mexico

This paper documents the evolution of system performance in an operating system from its initial use by a few friendly users in October 1978 to its present state. The system, DEMOS, was developed for the Cray Research Inc. (CRI) Cray-1 computer at the Los Alamos Scientific Laboratory (LASL) by staff members in the LASL Computer Science and Services Division. Some important features of DEMOS architecture are described. It is shown how the robustness in the design permitted major reduction in overhead to be achieved with only minor software changes. Particular emphasis is placed on file system design and performance because much of the system overhead reduction occurred as a result of changes to this system component. Also, some predictions are made regarding performance resulting from proposed modifications.

Key words: System tuning; message overhead; capabilities; modular design.

1. Introduction

In October 1978, the DEMOS operating system, which was developed at the Los Alamos Scientific Laboratory, became available for use by friendly users at LASL on the Cray-1A computer. Hardware to support I/O included 7 channels, 7 disk controllers, and 13 Cray Research, Inc. (CRI) DD19 disk drives. Sustained I/O rates were expected to be in the 20-60 Mbit/s range [1]. This rate approximates the maximum for a single channel/contoller/disk (35 megabits). Initial measurements showed that the system would support only half the minimum range value at an overhead ranging from 50% to 80% of the CPU. What had gone wrong? Could it be fixed? What would the fix cost?

In this report, we analyze the DEMOS design and its underlying assumptions. We show how minor changes enabled the system to support the anticipated I/O workload with acceptable overhead, and we quantify the improvement expected from further enhancements.

In Secs. 2 and 3 we present brief overviews of DEMOS architecture and its implementation, and the file system design. Detailed descriptions can be found in Refs. 1 and 2. In Sec. 4 we present some measurements, problems, and solutions encountered early in the evolution. In Sec. 5 we present some recent (and as yet unsolved) problems and speculate on performance gains that might be achieved by alternate solutions to the problems.

2. DEMOS Architecture

The fundamental computational process in DEMOS is called a task. A task consists of a program and its associated state information, a memory area, and a link table that provides capability-like access to resources in the computing system [3].

In the simplest case the terms "task" and "user job" may be synonymous, although a user job usually consists of many tasks, active either serially or in parallel.

One designated task, called the kernel, assumes ownership of the hardware interrupt facility and access to all executable memory when the machine is deadstarted. The kernel contains hardware I/O drivers, interrupt management, the basic intertask communication mechanism, and the link creation/manipulation facility. Other system tasks are not distinguished from user tasks; however, they are able to acquire some special capabilities (links) as a side effect of the order in which they are created at system deadstart. How this happens is explained in the following descriptions of links, task communication, and the switchboard task.

2.1 System Calls

A system call allows tasks to communicate with the kernel. To make a system call, tasks place parameters in particular registers (for example, an ID defining the service desired and pointers to data areas) and then execute an interrupt instruction. The kernel fields the interrupt, performs the service if appropriate, and restarts some task. Typical system calls are move data (from one address in memory to another), send/receive message, and create/destroy link.

2.2 Links

A link is a qualified pointer to the task that created it. When creating a link (pointer to itself), a task can endow the link with properties describing the link's use and disposition. Link properties include permission for (or restriction from) duplicating the link or giving it to some other task and specification of how the link is to be used. For example, a communications link allows messages to be sent to the creating task and a data link allows access (through a system call) to the creating task's memory. The physical realization of a link also contains other detailed information that allows the creating task to determine, on receipt of a message, which of its links is being used by the message sender.

A task creates a link by first constructing a data area describing the link's desired properties and then executing the appropriate system call. The kernel responds by creating a link, placing its physical realization in the task's link table (inaccessable by the task), and returning to the task an integer representing the location of the link in the link table. A task uses the link by including this integer in a system call. The link can be used as a message or data path to the creating task. The message or data path is established when the creating task gives the link to another task. When a link is given to a task, the physical realization is removed from the giving task's link table and written into the receiving task's link table.

2.3 Task Communication

A task can communicate with another task only if it has a link to that task. The communication is straightforward. A task wishing to send a message constructs the message in its memory area in a format that has been agreed to by the receiving task. The message area includes a header (to be used by the kernel) and a data area. The header includes an integer representing a displacement into the sending task's link table. A system call then causes the kernel to gain control, select the specified link from the sending task's link table, determine which task created the selected link, and copy the data from the message area onto a queue of messages for the task that created the link. The receiving task will receive the message only by issuing a receive message system call. The receiving task can selectively receive messages from specific subsets of the links it has given out by enabling reception only on those links with certain properties. The desired properties are specified when the receiving task issues the receive message system call.

A task owning a data link to the memory area of another task can access the other task's memory area through a move data system call. The mechanism is similar to the send message system call; however, the task whose memory is being accessed is not actively involved in the transaction. The data is moved by the kernel.

2.4 Switchboard Task

Some means must exist for tasks to exchange the links required to establish initial communication. The switchboard task fulfills this function. During system initialization, following a deadstart, the kernel creates the switchboard task and forges a link to it. The forged link is given to each task as it is created. Since

all system tasks are created at deadstart prior to creation of any user tasks, system tasks are able to use the switchboard to their advantage. They do this by creating a number of links to themselves and then giving the links (along with a name by which the link should be referenced) to the switchboard. Properties of some links specify that they may be duplicated; thus, the switchboard may give copies of these links to any task that asks for them by name. Other links given to the switchboard may not be duplicated and can be given only to the first task that asks. Those nonduplicatable links that give system tasks their special privileges are obtained from the switchboard during system initialization, leaving only the nonspecial (duplicatable) links to be obtained by the user tasks. Each user task, when created, is given copies of a few standard links to the system; for example, to the switchboard and to the file system.

3. DEMOS File System

The DEMOS file system software consists of four memory resident tasks: the directory manager, request interpreter, buffer manager, and disk manager. As each of the four tasks is discussed below, the design philosophy of the DEMOS file system will emerge. Also, some potential bottlenecks and sources of system overhead will become evident.

3.1 DEMOS Directory Manager

DEMOS files have hierarchical names that are implemented through special files called directory files. Data stored in a directory file establishes a correspondence between file names and files. The directory files may be visualized as nodes in a tree. Directories point to other files, which may be directory files or data files. All leaf nodes are file descriptors that contain such information as the file's owner, classification, size, and the location on a disk of the file's data. For each file descriptor, there is a unique path through the directory tree from the system root directory to the file descriptor.

The directory manager has access to the system root directory (and thus implicitly to all files) by virtue of a special link obtained from the switchboard at system deadstart time.

Each user task, when created, is given a link to the directory manager. This link is the only communication path between user and file system guaranteed by DEMOS. The link allows the user to send messages to the directory manager. Through parameters embedded in such messages, the user may request the directory manager to create or open files, or may inquire about the properties of some file. The directory manager processes requests by using the services of the other three file system tasks. A successful create or open request causes a new link to be created and returned to the user by the directory manager. During processing of the open/create request, a correspondence between the new link and the file it represents is established by the request interpreter to support processing of future transactions on the file. The user task sends messages containing requests for file processing (for example, read, write, and seek) through the newly created link to the request interpreter.

In summary, the directory manager serves as the standard user interface to the file system, has access to all files/directories, controls file access through information contained in the directories, and processes user directory requests by obtaining services from the other file system tasks.

3.2 DEMOS Request Interpreter

The request interpreter serves as the user interface to the file system for all transactions on open files. It uses services provided by the buffer-manager and disk manager to process user requests.

Files are initially created with no data in them and no space on the disk allocated to them. As data is written to the file, blocks are allocated on a disk to hold the data. A block contains 4096 bytes and occupies a sector on the disk. Programs do not have to deal with blocks, however, and may read or write (sequentially or randomly) any number of logically contiguous bytes in the file.

The user reads or writes the file by sending the appropriate message to the request interpreter through the link created at file open time. The request interpreter updates current logical file position pointers and then requests buffers from the buffer manager to hold the data. The request to the buffer manager is made through a special link obtained by the request interpreter from the switchboard task at system deadstart.

Once the buffer manager has supplied the requested buffers, the request

interpreter moves data to/from the user space from/to the buffers. Implicit in this transaction are data links, supplied by the user and buffer-manager tasks that give the request interpreter access to user and buffer manager memory. A user request that exceeds the size of available buffer memory space is fragmented into transactions of acceptable sizes by the request interpreter and the buffer manager as buffer space becomes available. A user request for N disk blocks will be processed as some number K distinct requests within the file system, where 1 <= K <= N. Fragmentation is logically transparent to the user.

3.3 DEMOS Buffer Manager

The buffer manager supplies intermediate storage between a user task's memory and secondary storage (disks on the Cray 1A). There are several reasons for buffering I/O data. In addition to the usual reasons (for example, CPU-I/O overlap or read ahead/write behind), the buffer manager allows memory compaction (or reorganization) to proceed in parallel with I/O. Also, the buffer manager implements the interface between byte-at-a-time and block-at-a-time I/O, thus eliminating the need for this service to be replicated in each user task space. The buffer manager also serves as an adjunct to the CPU scheduler by limiting the rate at which user tasks are allowed to complete I/O transactions and, implicitly, their rate of CPU usage. A transaction that is not completely satisfied by the buffer manager is periodically retried by the request interpreter until it is satisfied. This activity is transparent to most users; however, those who have carefully balanced their CPU and I/O requirements to match the hardware capability and to overlap significantly may notice a dramatic reduction in apparent system performance; for example, CPU and I/O usage may have been serialized.

A user read request is received by the buffer manager from the request interpreter as a "fill buffers" request. If the desired data are in system buffers, those buffers are returned to the request interpreter. Otherwise, the buffer manager allocates buffers and asks the disk manager to fill them. The filled buffers are returned to the request interpreter.

A user write request causes the buffer manager to allocate empty buffers (and possibly do I/O to empty some buffers). The empty buffers are returned to the request interpreter. Filled buffers are returned to the buffer manager by the request interpreter for output to disk.

3.4 DEMOS Disk Manager

The DEMOS disk manager maintains the logical-to-physical disk file mapping tables and allocates/deallocates all disk space. Physical disk I/O requests from the buffer manager are queued by the disk manager and sent to the kernel disk driver in a order based on least latency, least seek time next. In the initial version of DEMOS, the disk manager-to-disk driver requests were block (sector) at a time (per device), because the disk hardware was oriented to this size transaction. Also the kernel disk driver did not implement request queueing.

When the kernel disk driver notifies the disk manager that a transaction is complete, the disk manager notifies the buffer manager and attempts to start another transaction.

An attempt to extend a file requires the disk manager to do housekeeping associated with disk allocation prior to initiating the user transaction. Disk space allocation maps are retained on the disks, so file extensions require the disk manager to do I/O. The disk manager uses buffer manager services in a way similar to the request interpreter for this purpose.

3.5 User/File System Communication Costs

Figure 1 illustrates the logical communication precipitated by a user I/O request to an open file.

Paths m1, ..., m8 represent message paths supported by message links; paths d1 and d2 are data paths, d1 supported by a data link and d2 implemented in hardware; paths c1 and c2 are hardware control paths, c1 being implemented by hardware command and c2 by interrupt. Paths l1 and l2 represent link creation/inquiry system calls.

The kernel has been omitted from most paths in Fig. 1 for simplicity; for example, each of the paths, mi, requires a send message system call and a receive message system call, each of which interrupts to the kernel for implementation.

In the simplest case a user I/O request requires 11 system functions, 4

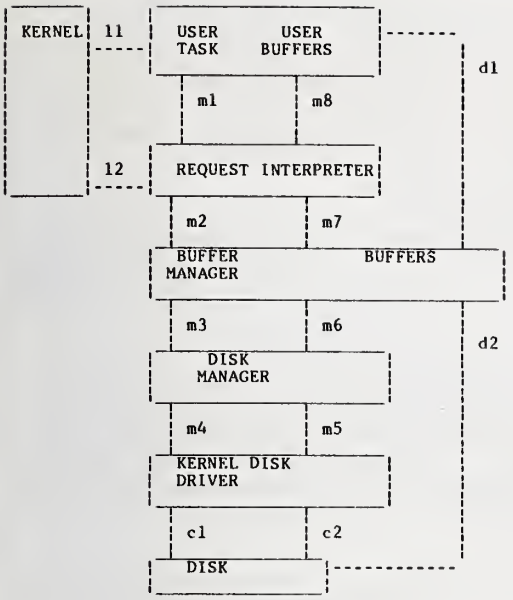


Fig. 1. DEMOS logical file system organization.

send/receive message pairs, 1 data move, and 2 link related system calls; that is, the data was found in (or moved to) the buffer manager buffers.

The simplest case involving a physical disk I/O transaction also requires 11 system functions, 4 send/receive pairs, and 3 disk interrupts (paths c1 and d2 are done by hardware). This case represents I/O between the buffer manager and the disk. Those cases including only the disk manager, kernel disk driver, and disk are considered to be part of physical I/O.

A user I/O request that requires physical disk I/O is the sum of the two above, and requires 22 system functions.

For transactions involving N disk blocks, paths m4, m5, d1, and d2 must each be traversed N times, and paths c1 and c2, 3N times. The cost in terms of system functions becomes:

Cost = 2(6 + 2N) messages + 3N interrupts
+ N data moves + 2 link calls. (1)

A transaction that is broken into K fragments by the buffer manager adds 4 x (K-1) send/receives to the cost for communication between the request interpreter/buffer manager and buffer manager/disk manager, along paths m2, m3, m6, m7. Thus the cost of a fragmented request becomes:

Cost = 2(6 + 2N + 4(K-1)) messages
+ 3N interrupts + N data moves (2)
+ 2 links.

Throughout the remainder of this section and the next we will equate the cost of all system functions. Performance data are presented later to support this simplification.

The cost (in system functions) for a one-track (18-block) I/O transaction is:

Cost (min) = 84 messages + 54 interrupts
+ 18 data moves + 2 links (3)
= 158 system functions.

Cost (max) = 228 messages + 54 interrupts
+ 18 data moves + 2 links (4)
= 296 system functions.

Clearly, the cost in CPU cycles of a system function and fragmentation introduced by the buffer manager are potential sources of serious system overhead. Figure 2 represents overhead during a sustained period of I/O transfer and illustrates potential overhead for a transaction with N = 18. In Fig. 2, the overhead is the fraction of time the CPU is busy managing I/O during the elapsed time required to perform the I/O. The x-axis represents times (in microseconds) to perform a system function, and the y-axis represents the overhead fraction. Overhead curves are shown for several values of K. Note that disk latency and positioning times are not included in the calculations.

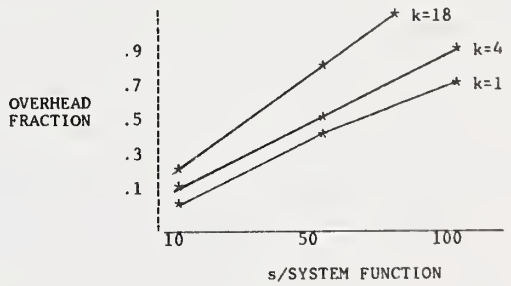


Fig. 2. Overhead in a 1-track (18-block) I/O transaction.

Recall that the overhead in Fig. 2 includes only system function overhead and excludes processing by the file system tasks. File system overhead will be incorporated in the next section.

3.6 CRAY-1A I/O Subsystem Hardware

We now describe briefly those features of the Cray I/O hardware needed to understand the discussion of sustainable I/O rates. A detailed description is found in Ref. 4.

Figure 3 illustrates the topology for the Cray I/O components used at LASL. Other topologies can be used. Only a single channel is shown in this figure. The controller has two buffers, each a disk block (512 words) in length. This allows data transfer between disk/controller and channel/controller to proceed in parallel. In this configuration, data can flow from main memory to the disk at a sustained rate of 550,000 word/s within a disk cylinder and 500,000 word/s to multiple contiguous cylinders. Data flows between main memory and controller buffer at nearly 4,000,000 word/s.

The controller signals that I/O is complete as soon as data has been transferred between main memory and controller buffer memory. For sequential sector output requests, this allows a window of approximately 1.5 sector times (1.4 ms) of elapsed time after the controller signals I/O complete for the file system to output the next physical disk block. A longer delay causes a missed disk revolution. For input, approximately 0.75 sector times (0.7 ms) of elapsed time are allowed between completion of one sector and a request to input the next without missing a disk revolution.

If we assume a value for processing time in each of the file system tasks while satisfying a request, we can determine the maximum value for an acceptable function cost, where acceptable means a system function cost that permits software to keep pace with hardware. To keep matters simple, we assume a task-processing cost equal to that of a system function (measurements confirm this within a factor of 1.5). With this assumption, maximum acceptable cost becomes:

$$\text{COST max(input)} = \frac{(700\ \mu s)}{(22 \text{ functions} + 3 \text{ system tasks})} = 28\ \mu s. \quad (5)$$

For output, the maximum acceptable cost is doubled due to the way the hardware operates. Similarly, the overhead for an unfragmented N block request can be determined.

$$\text{OVERHEAD} = 2(6 + 2N) \text{ system calls} + N \text{ data moves } 3N \text{ interrupts} + 2 \text{ links} + 3 \text{ system tasks}. \quad (6)$$

Again, equating the costs of the five functions above, overhead becomes

$$\text{OVERHEAD} = (8N + 17)Csf,$$

where Csf is the cost of a system function. Percent overhead, that is, percentage of time the CPU is busy managing the I/O transaction, is then:

$$\%\ \text{OVERHEAD} = 100 \times \frac{(\text{OVERHEAD})}{\text{ELAPSED TIME}}. \quad (7)$$

For a continuous (consecutive sectors and no missed disk revolutions) stream of I/O requests, I/O ELAPSED TIME becomes Tds x N, where Tds is the time for a disk sector to pass under the read/write heads (in our case, 926 μs).

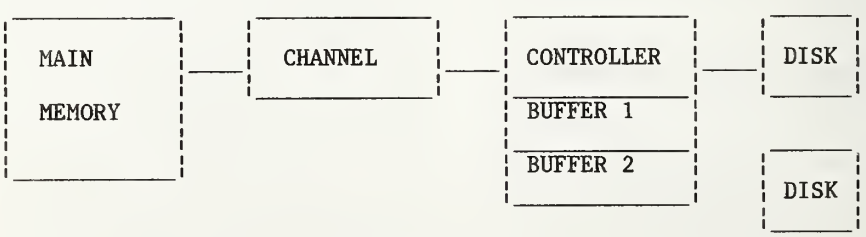


Fig. 3. Cray topology at LASL (1978).

Figure 4 is produced using values for N of 1, 18 and a range of values for Csf in Eq. (7).

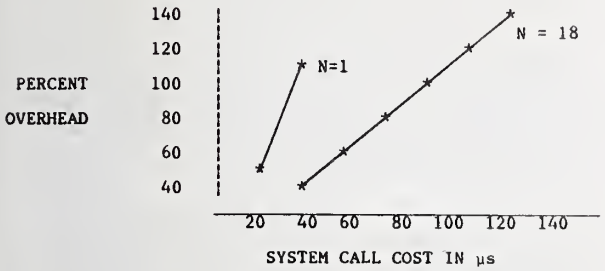


Fig. 4. Percent overhead for nonfragmented I/O transactions.

When we include a fragmentation factor, K, Eqs. (6) and (7) become:

$$\text{OVERHEAD} = (8N + 8K + 9)Csf \qquad (8)$$

and

$$\%\ \text{OVERHEAD} = 100 \times (\text{OVERHEAD})/\text{ELAPSED TIME} \qquad (9)$$

Figure 5 is a reproduction of Fig. 4 adding values for K of 3 and 6.

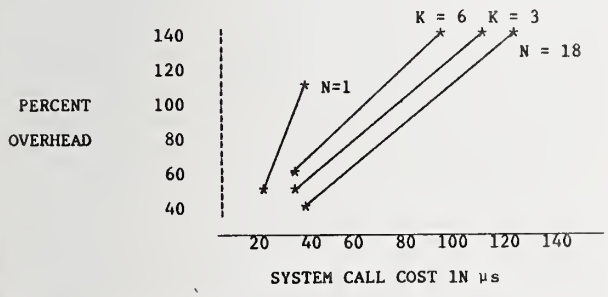


Fig. 5. Percent overhead for fragmented I/O transactions for the case N = 18.

Clearly Csf and K are important factors in both overhead and sustainable I/O rates. In the next section we present some initial measurements of Csf, K, and I/O overhead and rates.

4. Performance Data Gathering

Demos supports both centralized (kernel) and localized (system task) data collection facilities. The kernel task collects information describing all system calls and interrupts in a circular main memory buffer. Any system or user task can extract the data from the buffer. System tasks collect summary information pertinent to their operation in their local memories. These tables can be read by programs running on a computer connected to one of the Cray channels.

The kernel buffer data are used to reproduce a chronological trace of very detailed system activity at the expense of writing an enormous amount of data to disk with the attendant high cost overhead. Individual task data is used to determine parameter values associated with more global events, for example, number of physical disk I/O requests, number of active buffers. Both facilities are used to extract the data in the following sections.

4.1 Some Preliminary Results

Early measurements gave average values for costs of messages, interrupts, data moves, and file system task processing of 120, 100, 75, and 150 µs, respectively. Using these values and N=1, Eq. (6) gives:

$$\begin{aligned}\text{OVERHEAD} &= 16(120) + 1(75) + 3(100) \\ &\quad + 2(120) + 3(150) \\ &= 2985\ \mu s.\end{aligned}$$

Clearly, from Eq. (5) the I/O software cannot keep pace with the I/O hardware; that is, the minimum time between completion of sector i and a software request for sector i+1 is over three sector times. With a stream of consecutive, single-sector requests, only a single sector would be read per disk revolution without some scheme for reading alternate or every n-th sectors. However, the hardware read-ahead feature in the disk controller permits a rate of two sectors of input per disk revolution. Thus, the predicted sustainable rate for single-sector reads becomes

$$\begin{aligned}\text{Rate (in)} &= 2/18 \times \text{maximum rate} \\ &= 55{,}000\ \text{word/s}.\end{aligned}$$

From Eq. (7) the predicted percent overhead is

$$\% \text{ OVERHEAD} = 100 \times (2 \times 2985)/16667 = 36\% .$$

The dual write-behind features in the disk controller and buffer manager will allow transfer of four blocks per revolution (after the first) for single-block writes. During the first revolution, two blocks are buffered in the controller, a third is queued in the kernel, and some number (<= 15) are queued in the buffer manager. During subsequent revolutions, sufficient time exists for the block queued in the kernel and one block queued in the buffer manager to be written to disk and two more can be buffered in the controller. The predicted output rate is then

$$\text{Rate (out)} = 4/18 \times \text{maximum rate} = 110{,}000 \text{ word/s}$$

for single-block requests, and the predicted overhead is

$$\% \text{ OVERHEAD} = 100(4 \times 2985)/16667 = 72\%.$$

Measurements taken in October 1978 gave values of 53,000 word/s at 42% overhead for single-block input and 94,000 word/s at 73% overhead for single-block output.

4.2 A Simple Nonsolution

One solution to the problem seemed obvious: large user requests. For large requests (for example, one disk cylinder--180 blocks) Eq. (6) predicts:

$$\text{OVERHEAD} = 16027 \ \mu s,$$

and from Eq. (7),

$$\% \text{ OVERHEAD} = 100 \times 16027/1666667 = 93\%.$$

This seems to suggest that at least large swaps or program fetches might be done effectively, because during a full memory swap or load no user program desires to use the CPU. Such swaps and loads are common in our production environment. Measurements show that the average swap is between 200,000 and 300,000 words and a significant fraction, 30% to 40%, are over 600,000 words.

Large requests are always fragmented due to limited buffer manager buffer space (currently 18 disk blocks). Observed fragmentation factor values are in the range N/3 to N/4 for large N; that is, a 180-block request will probably be fragmented into 45 to 60 small requests within the file system. Equation (8) predicts that the resultant overhead will exceed 100% of the total CPU time. In this case, the I/O software will not be able to keep pace with the hardware, and disk revolutions will be missed.

To test the predicted effect of fragmentation and overhead on large I/O transfers, a test was run that would first fragment the available buffer memory among a number of small (3-block) requests and then attempt large (180-block) transactions. Measurements showed a maximum sustained input rate of 250,000 word/s and output rate of 140,000 word/s. To understand how overhead and fragmentation produce these rates consider the following.

Assume each 180-block request is fragmented into 60 equal sized requests of 3 blocks each. Further, note that system tasks are prioritized in increasing order from user to kernel drivers in Fig. 1. A summation of processing times required by the buffer manager, disk manager, and kernel implied by paths in Fig. 1 gives 3,120 μs required to process each 3-block request. This is slightly more than the elapsed time required to read three disk blocks (926 μs/block). On input, hardware read-ahead covers for the slight excess and allows reading to proceed for a while; the system would eventually fall behind and miss a revolution. However, no time remains for communication between buffer manager and request interpreter. When all 18 blocks (6 requests of 3 blocks each) have been transferred, the buffer and disk manager run out of work. During the resulting missed disk revolution, the request interpreter completes the transaction and produces another set of six fragmented (3-block) requests in the buffer manager buffers. This gives an effective rate of half disk speed or 250,000 word/s.

On output, hardware read-ahead is not available. The disk manager falls behind after three transfers to disk and, during the remainder of the revolution, is able to transfer two blocks to the controller buffers. The result is a rate of about five blocks per revolution or 5/18 of maximum hardware rate, that is, 150,000 word/s.

The average rate is well below the lower end of the anticipated I/O demand range of 20 to 60 Mbit/s and the overhead for these low rates is still unacceptable.

4.3 Initial Effective Solutions

A simple solution was not evident for the fragmentation problem, so effort was concentrated on reducing other overhead. One improvement was obvious and simple: increase the bandwidth along paths m4 and m5 in Fig. 1. As a result of this increase, a single message can contain a request for multiple physically-contiguous disk sectors. This did not require implementation of request queueing in the kernel but did permit a reduction in the number of required interrupts from three to one for all but the first block of a request.

For relatively large N, Eqs. (6) and (8) become:

$$\text{OVERHEAD} = 16 \text{ messages} + N \text{ data moves} + N \text{ interrupts} + 2 \text{ links} + 3 \text{ tasks} + (21 + 2N)Csf, \quad (10)$$

$$\text{OVERHEAD} = (4K + 2N + 13)Csf. \quad (11)$$

In addition some small amount of kernel code to manage messages was converted to assembly language, and the cost of a message was reduced from 120 μs to about 75 μs.

These improvements reduced average system function and overhead cost by about 25%. However, overhead would still be too high for a sequence of small requests to keep pace with the disk, and fragmentation should continue to prevent large requests from keeping pace. Tests confirmed the modest reduction in overhead, a modest increase in sustainable I/O rates, and that fragmentation could still disrupt large requests. Tests that attempted to avoid fragmentation produced more encouraging but still unsatisfactory rates.

4.4 More Complex Solutions

Several solutions to the fragmentation and overhead problems were discussed. They included:

(a) merge the three file system tasks,
(b) increase the number of buffers,
(c) modify the buffer manager to avoid fragmentation for large requests,
(d) eliminate buffers and buffer manager, that is, I/O direct to user space,
(e) treat selected I/O requests in a special way.

Solution (a) was rejected for three reasons; it would take much work, would not address fragmentation, and would not reduce overhead to an acceptable level (only by another 20-25%).

Solution (b) was politically untenable; the system was already far too large.

Solution (c) seemed attractive but all proposed schemes appeared artificial and kludgey.

Solution (d) was undesirable since the buffer manager satisfied (on the average) 57% of all I/O requests and 67% of all small requests without resorting to physical disk I/O. These figures represent percent of total requests, not percent of total data volume. Furthermore, changes to the memory manager, swapper, etc., to lock user tasks with pending I/O at a fixed location in memory appeared difficult and counterproductive to system throughput.

Solution (e) was chosen as the most desirable. The special treatment consisted of transferring the data between user memory, a kernel buffer, and the disk one block at a time. This required adding a new message path between the request interpreter and the disk manager, a buffer per channel in the kernel, and a small amount of code in each of the tasks. During data moves (done by the kernel), the current address of the task could be checked; thus no task locking was required. Only those user requests not satisfied by the buffer manager and that did not require further disk allocation were candidates for special treatment. Typically, these requests were swaps, program fetches (loads), and periodic snapshots by user codes of large tables.

For these requests, fragmentation would be eliminated and overhead could be sharply reduced. Initial task communication to set up the I/O transaction costs 14 system functions (now at a cost of slightly less than 100 μs average). Processing during the request costs an interrupt and a data move (about 175 μs); processing to clean up and terminate the request costs six system functions.

Figure 6 illustrates the predicted costs for processing an N block request using the new scheme or the standard. The standard curve shown assumes a fragmentation factor of K = N/3. The new scheme is referred to here as buffer bypass, or bypass I/O. The cost for an N block request with no fragmentation is approximately the same as the cost for an N block bypass request; only fragmentation has been eliminated, overhead remains.

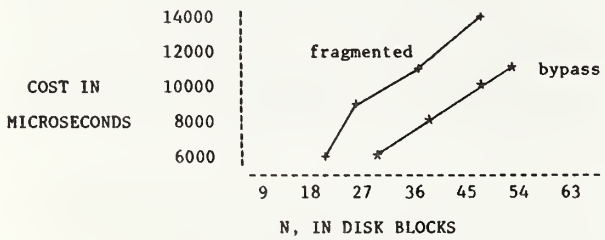


Fig. 6. Relative cost of bypass and fragmented I/O.

Figure 7 contains typical measured rates for various sized bypass requests. These measurements show that the system can now support I/O rates at the upper end of the projected range (60 X 10 bit/s). Overhead is about 20% for small requests (N ranges between 9 and 36 blocks) and decreases with increasing values of N. Overhead remains constant at about 15% once N exceeds 5 tracks (90 blocks). The overhead to support an I/O rate of 60 X 10 bit/s will be 30% if the requests are all large bypass requests. This 30% overhead reflects the fact that a program requiring this rate for a single input or output stream will have to merge input/output streams for files on two different channels.

4.5 Current Workload

Several measured 24-hour periods show that about 60% of all I/O now is treated as bypass I/O. Another 10% to 15% of the I/O workload could be transferred from standard to bypass I/O by allocating a file's disk space when the file is created rather than when it is written. As an intermediate step, the file system could be modified to allocate large blocks of disk space when a file is extended.

The average size of bypass I/O requests is 16 blocks, and for standard I/O the average size of a request is 4 blocks.

5. Remaining Problems

Figure 7 demonstrates that fragmentation and (as a byproduct) overhead are significantly reduced for large bypass I/O requests. However, rates are unacceptably low for small bypass requests, and both overhead and rates are unacceptable for standard requests. To understand the cause of low rates for small bypass requests, we again sum processing costs of the components.

The cost of finishing one bypass request and starting another is 20 system functions and two system task times. The sum of CPU costs for this activity is slightly longer than the maximum window allowed in order to continue a sequence of I/O transactions without missing disk revolutions. Even though the overhead is low for large requests, (N >= 9), the sustained I/O rate will be approximately:

Rate = maximum disk rate
x Tcount/(Tcount + 1),

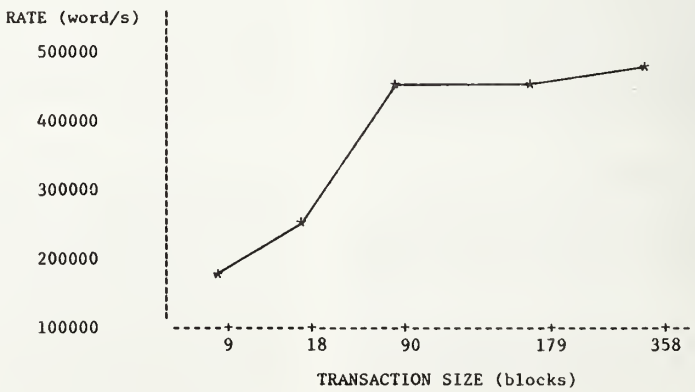


Fig. 7. Observed I/O rates for bypass requests.

where Tcount is the number of tracks in the request. This relation only states that a track will be missed between successive requests. The effective rate for 1-track requests is 1/2 x maximum rate and for full cylinder (10-track) requests, 10/11 x maximum rate.

5.1 Some Solutions

Several solutions have been discussed that should eliminate the missed revolutions between successive bypass requests. None have been implemented. Some of the solutions are:

(a) merge some or all of the file system tasks,
(b) implement single sector read-ahead in the kernel,
(c) signal output complete prior to transmitting last block to disk,
(d) increase the number of kernel buffers per channel.
(e) convert more of the kernel to assembly language and further reduce the cost of a system call or interrupt.

Solution (a) is attractive because it would eliminate some missed revolutions and would reduce overhead for standard I/O. However, a nontrivial amount of work would be required.

Solution (b) would only eliminate missed revolutions for bypass input requests. A combination of (b) and (c) would eliminate missed revolutions for bypass input and output requests. The drawback is that read-ahead unnecessarily ties up the channel and disk. Notification prior to actual I/O completion would allow hardware errors to go unreported to users.

Solution (d) is attractive because it is easy to implement and, coupled with (b), would eliminate missed revolutions for all bypass requests and for some standard requests. It would cost a small amount of trivial code and additional buffer space. It has the same drawbacks as solutions (b) and (c).

A combination of solutions (a), (b), and (e) probably represents the best approach from a performance and reliability standpoint; however, it also represents the most work and departs farthest from the original modular design. Results of (a) and (e) would be to reduce both overhead and processing time between successive requests. One current estimate suggests that message, link and interrupt costs could be reduced to about 50 μs with a moderate recoding effort. A result of (b) would be to widen the input window (see Sec. 2.6) to be equal to that of the output window, that is, 1.4 ms.

5.2 Predicted Effects of Modifications

Using the predicted cost of system functions, the cost of a standard I/O request becomes

```
Cost (standard) = 10 system calls
                + N data moves
                + (3+(N-1)) interrupts
                + 3 tasks.
```

The predicted cost of a bypass I/O request with the proposed modifications and times is

```
Cost (bypass) = 11 system calls
              + N data moves
              + (3+(N-1)) interrupts
              + 2 tasks.
```

The costs of standard and bypass I/O are both reduced to approximately the same value.

Figure 8 shows predicted percent overhead for bypass or standard requests.

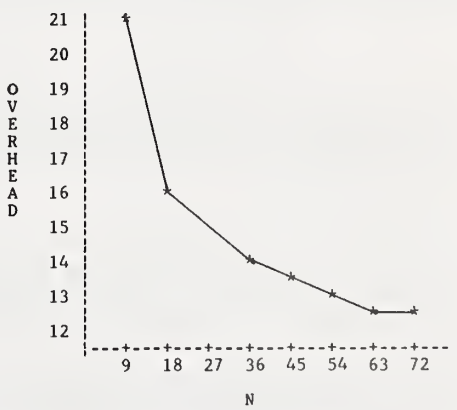


Fig. 8. Predicted percent overhead for I/O transactions.

An important point with the proposed modifications and times is that the processing done between the end of one request and start of another excludes the

costs of all but one data move and one interrupt. This places the inter-request processing time well within the 1.4-ms window for both standard and bypass requests. The software should be able to keep pace with the hardware and, as Fig. 8 shows, the overhead becomes excessive only for very small requests.

6.0 Conclusions

It continues to be our experience that when things go wrong with the performance of a system there will be numerous educated guesses as to the cause and correction required; most of them will be wrong or ineffective.

This paper provides a detailed case study of how one software project profited from a performance measurement and analysis effort that began in early design and was continued through development, testing, and production use of the software. We believe that the results reported here are typical of those for any large-scale software development project that incorporates a comprehensive performance measurement and analysis component.

Acknowledgments

We thank all members of the LASL DEMOS development staff and in particular Jim Clifford and John Montague for providing insight into DEMOS organization and operation. We also thank the CRI on-site engineers for their help in understanding the details of hardware functions.

References

1. M. L. Powell, The DEMOS File System, Proceedings of the Sixth Symposium on Operating System Principles, November 1977, 33-42.

2. F. Baskett, J. H. Howard, and J. T. Montague, Task Communication in DEMOS, Proceedings of the Sixth Symposium on Operating System Principles, November 1977, 23-31.

3. R. S. Fabry, Capability-Based Addressing, CACM, 17, 7 (July 1974), 403-412.

4. Cray Research, Inc., CRAY-1 Computer System Reference Manual, Publication 2240004 (1977).

NBS-114A (REV. 9-78)

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET	1. PUBLICATION OR REPORT NO. NBS SP 500-65	2. Gov't. Accession No.	3. Recipient's Accession No.

4. TITLE AND SUBTITLE

COMPUTER SCIENCE & TECHNOLOGY:

Proceedings of the Sixteenth Meeting of the Computer Performance Evaluation Users Group (CPEUG)

5. Publication Date: October 1980

6. Performing Organization Code

7. AUTHOR(S)

Dr. Harold J. Highland, Editor

8. Performing Organ. Report No.

9. PERFORMING ORGANIZATION NAME AND ADDRESS

NATIONAL BUREAU OF STANDARDS
DEPARTMENT OF COMMERCE
WASHINGTON, DC 20234

10. Project/Task/Work Unit No.

11. Contract/Grant No.

12. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS *(Street, City, State, ZIP)*

Same as No. 9.

13. Type of Report & Period Covered: Final

14. Sponsoring Agency Code

15. SUPPLEMENTARY NOTES

☐ Document describes a computer program; SF-185, FIPS Software Summary, is attached.

16. ABSTRACT *(A 200-word or less factual summery of most significent information. If document includes a significent bibliography or literature survey, mention it here.)*

The Proceedings record the papers that were presented at the Sixteenth Meeting of the Computer Performance Evaluation Users Group (CPEUG 80) held October 20-23, 1980, in Orlando, Florida. With the theme "CPE Trends in the 80's," CPEUG 80 focused on new applications that are expected to grow in the 80's and changes that may occur in traditional areas during the 80's. The program was divided into two parallel sessions and included technical papers on previously unpublished work, case studies, tutorials, and panels. Technical papers are presented in the Proceedings in their entirety.

17. KEY WORDS *(six to twelve entries; alphabetical order; capitalize only the first letter of the first key word unless a proper name; seperated by semicolons)*

Benchmarking; capacity planning; computer performance evaluation; computer performance measurement; computer performance prediction; computer system acquisition; CPE in auditing; installation management; on-line system evaluation; queuing models; simulation; workload definition.

18. AVAILABILITY

[X] Unlimited

☐ For Official Distribution. Do Not Release to NTIS

[X] Order From Sup. of Doc., U.S. Government Printing Office, Washington, DC 20402

☐ Order From National Technical Information Service (NTIS), Springfield, VA. 22161

19. SECURITY CLASS (THIS REPORT)	21. NO. OF PRINTED PAGES
UNCLASSIFIED	316
20. SECURITY CLASS (THIS PAGE)	**22. Price**
UNCLASSIFIED	$8.00

USCOMM-DC

U.S. GOVERNMENT PRINTING OFFICE: 1980 [illegible]

NBS TECHNICAL PUBLICATIONS

PERIODICALS

JOURNAL OF RESEARCH—The Journal of Research of the National Bureau of Standards reports NBS research and development in those disciplines of the physical and engineering sciences in which the Bureau is active. These include physics, chemistry, engineering, mathematics, and computer sciences. Papers cover a broad range of subjects, with major emphasis on measurement methodology and the basic technology underlying standardization. Also included from time to time are survey articles on topics closely related to the Bureau's technical and scientific programs. As a special service to subscribers each issue contains complete citations to all recent Bureau publications in both NBS and non-NBS media. Issued six times a year. Annual subscription: domestic $13; foreign $16.25. Single copy, $3 domestic; $3.75 foreign.

NOTE: The Journal was formerly published in two sections: Section A "Physics and Chemistry" and Section B "Mathematical Sciences."

DIMENSIONS/NBS—This monthly magazine is published to inform scientists, engineers, business and industry leaders, teachers, students, and consumers of the latest advances in science and technology, with primary emphasis on work at NBS. The magazine highlights and reviews such issues as energy research, fire protection, building technology, [illegible] conservation, pollution abatement, health and safety, and consumer product performance. In addition, it reports the results of Bureau programs in measurement standards and techniques, properties of matter and materials, engineering standards and services, instrumentation, and automatic data processing. Annual subscription: domestic $11, foreign $13.75.

NONPERIODICALS

Monographs—Major contributions to the technical literature on various subjects related to the Bureau's scientific and technical activities.

Handbooks—Recommended codes of engineering and industrial practice (including safety codes) developed in cooperation with interested industries, professional organizations, and regulatory bodies.

Special Publications—Include proceedings of conferences sponsored by NBS, NBS annual reports, and other special publications appropriate to this grouping such as wall charts, pocket cards, and bibliographies.

Applied Mathematics Series—Mathematical tables, manuals, and studies of special interest to physicists, engineers, chemists, biologists, mathematicians, computer programmers, and others engaged in scientific and technical work.

National Standard Reference Data Series—Provides quantitative data on the physical and chemical properties of materials, compiled from the world's literature and critically evaluated. Developed under a worldwide program coordinated by NBS under the authority of the National Standard Data Act (Public Law 90-396).

NOTE: The principal publication outlet for the foregoing data is the Journal of Physical and Chemical Reference Data (JPCRD) published quarterly for NBS by the American Chemical Society (ACS) and the American Institute of Physics (AIP). Subscriptions, reprints, and supplements available from ACS, 1155 Sixteenth St., NW, Washington, DC 20056.

Building Science Series—Disseminates technical information developed at the Bureau on building materials, components, systems, and whole structures. The series presents research results, test methods, and performance criteria related to the structural and environmental functions and the durability and safety characteristics of building elements and systems.

Technical Notes—Studies or reports which are complete in themselves but restrictive in their treatment of a subject. Analogous to monographs but not so comprehensive in scope or definitive in treatment of the subject area. Often serve as a vehicle for final reports of work performed at NBS under the sponsorship of other government agencies.

Voluntary Product Standards—Developed under procedures published by the Department of Commerce in Part 10, Title 15, of the Code of Federal Regulations. The standards establish nationally recognized requirements for products, and provide all concerned interests with a basis for common understanding of the characteristics of the products. NBS administers this program as a supplement to the activities of the private sector standardizing organizations.

Consumer Information Series—Practical information, based on NBS research and experience, covering areas of interest to the consumer. Easily understandable language and illustrations provide useful background knowledge for shopping in today's technological marketplace.

Order the above NBS publications from: Superintendent of Documents, Government Printing Office, Washington, DC 20402.

Order the following NBS publications—FIPS and NBSIR's—from the National Technical Information Services, Springfield, VA 22161

Federal Information Processing Standards Publications (FIPS PUB)—Publications in this series collectively constitute the Federal Information Processing Standards Register. The Register serves as the official source of information in the Federal Government regarding standards issued by NBS pursuant to the Federal Property and Administrative Services Act of 1949 as amended, Public Law 89-306 (79 Stat. 1127), and as implemented by Executive Order 11717 (38 FR 12315, dated May 11, 1973) and Part 6 of Title 15 CFR (Code of Federal Regulations).

NBS Interagency Reports (NBSIR)—A special series of interim or final reports on work performed by NBS for outside sponsors (both government and non-government). In general, initial distribution is handled by the sponsor; public distribution is by the National Technical Information Services, Springfield, VA 22161, in paper copy or microfiche form.

BIBLIOGRAPHIC SUBSCRIPTION SERVICES

The following current-awareness and literature-survey bibliographies are issued periodically by the Bureau:

Cryogenic Data Center Current Awareness Service. A literature survey issued biweekly. Annual subscription: domestic $35; foreign $45.

Liquefied Natural Gas. A literature survey issued quarterly. Annual subscription: $30.

Superconducting Devices and Materials. A literature survey issued quarterly. Annual subscription $45. Please send subscription orders and remittances for the preceding bibliographic services to the National Bureau of Standards, Cryogenic Data Center (736) Boulder, CO 80303.