COMPUTER SCIENCE & TECHNOLOGY:

COMPUTER PERFORMANCE EVALUATION USERS GROUP

CPEUG

13th Meeting

NBS Special Publication 500-18
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 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 consists of the Institute for Basic Standards, the Institute for Materials Research, the Institute for Applied Technology, the Institute for Computer Sciences and Technology, the Office for Information Programs, and the Office of Experimental Technology Incentives Program.

THE INSTITUTE FOR BASIC STANDARDS provides the central basis within the United States of a complete and consistent system of physical measurement; coordinates that system with measurement systems of other nations; and furnishes essential services leading to accurate and uniform physical measurements throughout the Nation's scientific community, industry, and commerce. The Institute consists of the Office of Measurement Services, and the following center and divisions:

Applied Mathematics — Electricity — Mechanics — Heat — Optical Physics — Center for Radiation Research — Laboratory Astrophysics[2] — Cryogenics[2] — Electromagnetics[2] — Time and Frequency[2].

THE INSTITUTE FOR MATERIALS RESEARCH conducts materials research leading to improved methods of measurement, standards, and data on the properties of well-characterized materials needed by industry, commerce, educational institutions, and Government; provides advisory and research services to other Government agencies; and develops, produces, and distributes standard reference materials. The Institute consists of the Office of Standard Reference Materials, the Office of Air and Water Measurement, and the following divisions:

Analytical Chemistry — Polymers — Metallurgy — Inorganic Materials — Reactor Radiation — Physical Chemistry.

THE INSTITUTE FOR APPLIED TECHNOLOGY provides technical services developing and promoting the use of available technology; cooperates with public and private organizations in developing technological standards, codes, and test methods; and provides technical advice services, and information to Government agencies and the public. The Institute consists of the following divisions and centers:

Standards Application and Analysis — Electronic Technology — Center for Consumer Product Technology: Product Systems Analysis; Product Engineering — Center for Building Technology: Structures, Materials, and Safety; Building Environment; Technical Evaluation and Application — Center for Fire Research: Fire Science; Fire Safety Engineering.

THE INSTITUTE FOR COMPUTER SCIENCES AND TECHNOLOGY conducts research and provides technical services designed to aid Government agencies in improving cost effectiveness in the conduct of their programs through the selection, acquisition, and effective utilization of automatic data processing equipment; and serves as the principal focus wthin the executive branch for the development of Federal standards for automatic data processing equipment, techniques, and computer languages. The Institute consist of the following divisions:

Computer Services — Systems and Software — Computer Systems Engineering — Information Technology.

THE OFFICE OF EXPERIMENTAL TECHNOLOGY INCENTIVES PROGRAM seeks to affect public policy and process to facilitate technological change in the private sector by examining and experimenting with Government policies and practices in order to identify and remove Government-related barriers and to correct inherent market imperfections that impede the innovation process.

THE OFFICE FOR INFORMATION PROGRAMS promotes optimum dissemination and accessibility of scientific information generated within NBS; promotes the development of the National Standard Reference Data System and a system of information analysis centers dealing with the broader aspects of the National Measurement System; provides appropriate services to ensure that the NBS staff has optimum accessibility to the scientific information of the world. The Office consists of the following organizational units:

Office of Standard Reference Data — Office of Information Activities — Office of Technical Publications — Library — Office of International Standards — Office of International Relations.

---

[1] Headquarters and Laboratories at Gaithersburg, Maryland, unless otherwise noted; mailing address Washington, D.C. 20234.

[2] Located at Boulder, Colorado 80302.

COMPUTER SCIENCE & TECHNOLOGY:

Computer Performance Evaluation Users Group (CPEUG)

t. Special publication 500-18

Proceedings of the Thirteenth Meeting
held at New Orleans, Louisiana
October 11-14, 1977

Editors:

Dennis M. Conti and Josephine L. Walkowicz

Conference Host:

U.S. Department of Agriculture
New Orleans, Louisiana

Sponsored by:

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

U.S. DEPARTMENT OF COMMERCE, Juanita M. Kreps, Secretary

Dr. Sidney Harman, Under Secretary
Jordan J. Baruch, Assistant Secretary for Science and Technology

US NATIONAL BUREAU OF STANDARDS, Ernest Ambler, Acting Director

Issued September 1977

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 this series should complete and return the form at the end of this publication.

National Bureau of Standards Special Publication 500-18
Nat. Bur. Stand. (U.S.), Spec. Publ. 500-18, 241 pages (Sept.) 1977
CODEN: XNBSAV

Library of Congress Catalog Card Number: 77-600040

U.S. GOVERNMENT PRINTING OFFICE
WASHINGTON: 1977

For sale by the Superintendent of Documents, U.S. Government Printing Office
Washington, D.C. 20402 - Price $4

Stock No. 003-093-01839-2

Foreword

The Computer Performance Evaluation Users Group, better known as CPEUG, needs no introduction to the growing number of computer professionals whose workaday concerns revolve around the management of the computer resources that have become indispensable to most of today's activities. In the six years of its existence, CPEUG has grown from its single-agency orientation to its present position as a national forum for active involvement for all interests engaged in computer performance evaluation.

The program for the Thirteenth Meeting of CPEUG provides an excellent illustration of the dynamic and imaginative nature of this forum. This, plus the varied interests represented in both the formal and informal activities of the Conference, provides for an effective channel for the dissemination of current information on computer performance evaluation, as well as for the detection of trends in a complex and still growing technology. The program is reproduced in the *Proceedings* and serves as the Table of Contents for the formal presentations made during the four technical sessions of the Conference. These are identified by page numbers at the right-hand margin to indicate their location in the *Proceedings*. The remaining activities of CPEUG 77 were informal sessions for which no formal papers were required but which were designed so as to provide an atmosphere conducive to frank discussions among participants of the discoveries, problems, mistakes, etc. associated with their experience in computer performance evaluation. All CPEUG activities--both formal and informal--contribute significantly to the effective use of the technology available for performance management and enhancement of the national ADP inventory.

We would like to acknowledge the assistance of all those who helped us prepare these *Proceedings*. Our special thanks go to Mrs. Brenda Ogg for her assistance during the editing process, as well as to the authors for their extra efforts in the face of tight deadlines.

Editors:
Dennis M. Conti
Josephine L. Walkowicz

Preface

The theme of this year's conference--The ADP Life Cycle--seemed fresh enough when it was adopted a year or so ago. Since then Life Cycle stock has soared. You see it everywhere--articles, books, workshops, seminars. Instead of the commencement platitudes for which this space is normally reserved, therefore, I would like to explain why the Life Cycle model continues to have a special relevance and usefulness for those of us who actively work to prevent improper selection and inefficient utilization of contemporary computer systems.

If there is one thing that can be said for the Life Cycle idea, it is that it has breadth. The fact that computer systems pass through cycles frustrates the comfortable impulse to deal with computer performance in a fragmentary, computerized, and unhistorical way. It persistently reminds us that any performance problem we are dealing with has a past from which it is possible to infer causes and chart directions; and a future before which we are obliged to be both humble and cautious in our present certitude. Having solved one set of performance problems in no way secures us a solution to The Performance Problem. Indeed the sponsoring user organization, the ADP facility, the installed configuration, and every component software system are each marking out a life cycle of their own, substantially independent of the others, and any one of them may be about to upset the performance equilibrium we may imagine we have created. In short, the cyclical view of computer systems chastens us against two of our greatest enemies: canned solutions and absolute certainty.

The notion that computer systems have in some sense a life of their own also has its value in reflecting on the mission of computer performance evaluation (CPE). The cradle-to-grave analogy suggests somewhat obliquely that any computer system is an extension of the larger social and economic life of the company or agency it serves. Clearly, the work of the performance analyst is incomplete, even for the present, if it fails to account for or to communicate with the people for whom the computing resource exists. CPE is human in another sense as well. As little as we may care to admit it, even to ourselves, CPE is far from being an exact science. Our tools are precise enough to give that impression, but we are still learning what they are good for and have found no general theories to bind our most exact observations together.

If I am correct in believing that no other part of data processing is so alien to narrowness of vision, ignorance, jargon, and technical passivity as computer performance evaluation, then the ADP Life Cycle model will do very nicely as our model. For further evidence of the point and of the enduring vitality and variety of CPE, I refer you to the papers collected inside this volume. They represent the best of CPEUG 77.

Richard F. Dunlavey
Chairman, CPEUG
October 1977

Abstract

The Proceedings record the papers that were presented at the Thirteenth Meeting of the Computer Performance Evaluation Users Group (CPEUG) held October 11-14, 1977 in New Orleans. The technical presentations were organized around the three phases of the ADP p Life Cycle: the Requirements Phase (workload definition), the Acquisition Phase (computer system and service selection), and the Operational Phase (performance measurement and prediction methods). The program of CPEUG 77 is also included and serves as a Table of Contents to the Proceedings.

Key words: ADP life cycle; computer performance evaluation; computer performance measurement; computer performance prediction; computer system acquisition; conference proceedings; CPEUG; hardware monitoring, on-line system evaluation; prediction methods; queuing models; simulation; software monitoring; workload definition.

CPEUG OFFICERS

Chairman:	Richard F. Dunlavey National Bureau of Standards Washington, DC
Vice Chairman:	Gerald W. Findley General Services Administration Washington, DC
Secretary-Treasurer:	Dennis M. Gilbert FEDSIM Washington, DC
Program Chairman:	Dennis M. Conti National Bureau of Standards Washington, DC
Arrangements:	Jules A. d'Hemecourt U.S. Department of Agriculture New Orleans, LA
Publications:	Josephine L. Walkowicz National Bureau of Standards Washington, DC
Publicity:	Caral A. Giammo Command and Control Technical Center The Pentagon Washington, DC

PROGRAM AND TABLE OF CONTENTS

PROGRAM AND TABLE OF CONTENTS

PROGRAM AND TABLE OF CONTENTS

PROGRAM AND TABLE OF CONTENTS

Thursday, October 13
(continued)

PROGRAM AND TABLE OF CONTENTS

Workshops
(continued)

A. WORKLOAD DEFINITION

FUNCTIONAL WORKLOAD CHARACTERISTICS AND COMPUTER RESPONSE TIME IN THE DESIGN OF ON-LINE SYSTEMS

J. D. Williams
J. S. Swenson

Bell Telephone Laboratories
6 Corporate Place
Piscataway, N. J. 08854

This paper presents two human factors studies which explored the relationship between user performance (data entry speed and errors) and on-line computer system response time. The studies investigated user performance with response times of zero to 45 seconds, during both data entry and on-line data correction types of work. The users experienced each response time condition for about one to two hours.

The results indicate that user performance was not degraded by long response times. In addition, the data indicates that the type of work that the user was doing had a much greater effect on user performance than did the computer response time. Finally, the results indicate that longer response times may be appropriate for certain types of work because they reduce the time that the computer waits for the user while potentially causing no decrease in system through-put.

Key words: Computer response time; human factors; on-line systems; system design; task variables; workload characteristics.

1. Introduction

With the increase in the number of computer-controlled information systems, there has been an increasing interest in user-oriented system engineering criteria. The two studies we will discuss were concerned with a prominent factor assumed to contribute to operator performance in interactive computer systems - i.e., computer subsystem (CSS) response time. As used in this report, CSS response time refers to the time lapse between the last information input by the operator of a time-sharing terminal and the subsequent initiation of a typed response by the computer at the same terminal. The designers of computer systems often place a great deal of emphasis upon "quickness" of response to operators' commands. One basic assumption that has created this short response time criterion is that long response times degrade operator performance and user attitudes towards, and acceptance of, the system.

However, this assumption is based upon a very small pool of data. The cost of providing such a short response time, in addition, is usually quite expensive in terms of computer hardware and software. Therefore, it is advantageous to determine a trade-off between hardware and software costs and operator performance. To do this, one must first accurately determine the effects of CSS response time parameters on operator performance.

The available literature contains a number of relevant articles [1,2,3,4,5,6,7,8,9][1]. None of these authors, however, addresses the question of human performance as measured by speed and errors.

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

The studies presented here were designed to explore the relationship between operator performance and mean CSS response time. The experimental tasks (functional workload characteristics)[2] used in the studies were designed to produce more reliable measures of operator performance. In addition, the apparatus utilized in the second study allowed for the measurement of operator "think time" and intercharacter typing times as well as error rates and overall typing speeds.

2. Methodology

The two studies we conducted were similar in that they were designed to investigate the effects of differing response time distributions, characterized in terms of their mean and variance ratios (the ratio of the variance to the mean), upon the typing speed and accuracy of operators. Because we felt that workload characteristics might moderate the effects of response time on operator performance, we developed two types of tasks for use in the experiment. One task (data entry) was confined to simple entry of random alphanumeric codes in a line-by-line fashion (Appendix I). The second task (data correction) required the operator to retrieve blocks of data from the computer and correct any errors that were found. A script provided the information necessary to retrieve the data but did not indicate where the errors were. The operators retyped the entire line if it contained an error (Appendix II).

As one can see from Appendices I and II, the alphanumerics used in these two tasks were quite different. In the data entry task the 10-character fields were composed of randomly ordered alphanumerics. The fields in the data correction task were of variable length, and were made up of either alphas or numerics in ascending sequential order. An error was defined as a character that was out of sequence. In both tasks the alphanumerics were arranged so that each operator typed every character an equal number of times. We shall return to these differences in task make-up (i.e., workload) in the discussion of results.

2.1 Procedures

Participants in the first study were 36 experienced typists. They worked alternately on the two types of task for one day. Each operator experienced only one of the mean response time conditions for a period of about 2.5 hours, under each of three variant conditions.

[2]By "functional workload characteristics", we mean the workload imposed upon the operator by the task, e.g., data entry, text editing, etc.

During the second study, five operators worked on the experimental tasks for five days each. They experienced each of the experimental response time distributions for periods of about 1½ hours. These operators were not experienced typists.

In the first study the operators were given an opportunity to practice on analogues of the experimental tasks for about an hour before data were collected. The first ½ day was devoted to practice in the second study.

2.2 Apparatus

The experimental tasks for the first study were performed using an EXECUPORT 300 terminal. This terminal was connected over phone lines to a Honeywell 6000 computer. An interactive system simulator program performed the following functions:

1. Acted as the computer on which the subjects performed their tasks.

2. Delayed its own responses in accordance with a predetermined distribution.

3. Recorded and time-stamped all transactions between the computer and the subject.

The equipment configuration of the second study differed in several respects. First, the participants interfaced with a Digital Equipment Co. PDP8/E computer via a Teleray 3700 CRT. This arrangement allowed for more precise timing of the response time distribution and for accurate timing of each keystroke.

In the first study the response time distributions were characterized in terms of their means and variance ratios. There were four mean response time conditions (5, 15, 30, and 45 seconds) and three variance ratio conditions (.25, 1.00, and 5.00). A baseline condition of zero mean and variance was also experienced by each participant, before and after the experimental session.

Response times in the second study were characterized in terms of their means. There were four levels: zero, low, medium, and high (0, 4.0-7.9 sec., 16.8-17.3 sec., and 24.2-30.3 sec., respectively). Variance ratio was not manipulated in this study.

3. Results

Data were analyzed using Analysis of Variance. Separate analyses were performed for each of four dependent variables: accuracy of performance, overall typing speed of performance, first-character time, and speed of performance corrected for first-character time. The results of each of these analyses are presented below.

3.1 Accuracy of Performance

In both studies the analysis of performance accuracy used as a dependent variable the number of typographical errors per character typed by each operator. Data used for this analysis were collected during the last third of each experimental condition, so that the data would reflect operator performance after the operators had become experienced with the response time distribution of the computer.

In the first study (Table 1) there was no identifiable main effect of response time variance on accuracy of performance. There

Table 1
Analysis of Variance
of Typographical Errors

SOURCE	df	MS	F
<u>Between Subjects</u>			
Response Time (R)	3	4.9634	1.9528
Variance Ratio (V)	2	.2611	0.1027
V x R	6	6.6144	2.6024*
Subjects (VR)	24	2.5416	
<u>Within Subjects</u>			
Task Type T	1	.5067	.3045
V x T	2	2.0831	1.2518
R x T	3	5.9645	3.5844*
V x R x T	6	.9362	.5626
Subjects x T (VR)	24	1.6640	

* $p < .05$

was a significant interaction between task type and response time ($F(6,24)=3.58, p < .05$) in the first study. A test of the simple effects of this interaction indicated that, in the data correction task, errors increased beyond the 15-second mean condition. The reader is advised to keep in mind that there were no samples interpolated between the 15- and 30-second conditions. These data are illustrated in Figure 1.

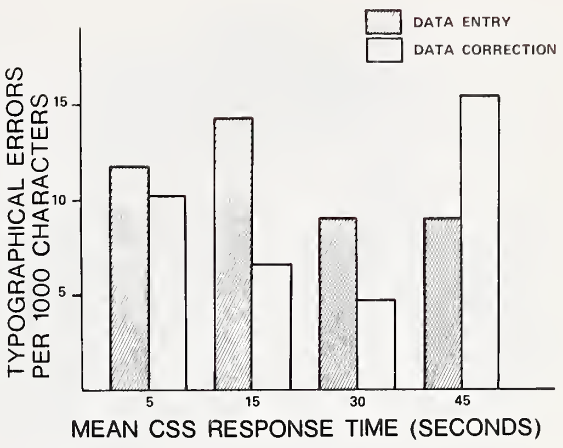


FIGURE 1 - MEAN TYPOGRAPHICAL ERRORS PER 1000 CHARACTERS IN THE FOUR EXPERIMENTAL GROUPS IN THE PILOT STUDY.

The second study dealt primarily with the response times where significant effects were found in the first study, that is, between 15 and 30 seconds. Here we found no significant effect of response time over the range investigated (0-30 seconds). There was a significant effect of task type ($F(1,4)=9.971, p<.05$), with the data entry task reflecting more errors than the data correction task (see Figure 2). The analysis of variance for these data is contained in Table 2.

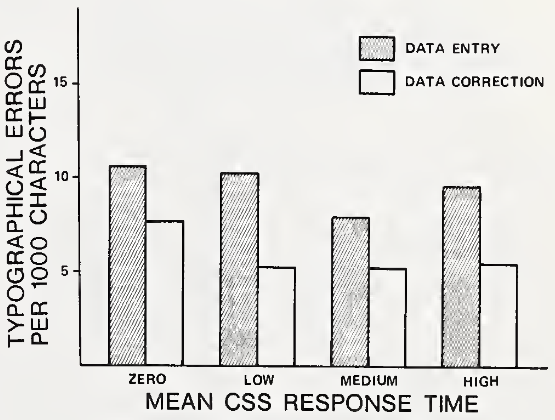


FIGURE 2 - MEAN TYPOGRAPHICAL ERRORS PER 1000 CHARACTERS IN THE DATA ENTRY AND DATA CORRECTION TASKS FOR THE FOUR CSS RESPONSE TIME GROUPS.

Table 2
Analysis of Variance
of Typographical Errors
for Data Entry
and Data Correction Tasks

SOURCE	df	MS	F
Response Time (R)	3	35.1	0.8
Operators (O)	4	362.9	9.7**
Task Type (T)	1	421.5	9.9*
R x O	12	41.0	1.1
R x T	3	8.7	0.1
O x T	4	42.2	1.1
R x O x T	12	63.8	1.7
Reps (R x O x T)	80	37.0	

** $p < .01$
* $p < .05$

3.2 Speed of Performance

Analysis of the overall speed of performance data generated during the first study (Table 3) revealed a significant main effect of task type (F(2,48)=20.638,$p<.01$) and a significant interaction between task type and response time (F(6,48)=3.827,$p<.01$). Overall speed of performance is the total measured time used by the operator from the appearance of the prompt character on the screen to the typing of the last character of the line by the operator. An analysis of the simple effects of this interaction suggested that there was no effect of response time in the data correction task, but that as response time increased in the data entry task, typing speed decreased (F=3.35,$p<.05$). Figure 3 illustrates these data.

Table 3
Analysis of Variance of
Overall Speed of Performance -
Last Third of Experimental Sessions

SOURCE	df	MS	F
Between Subjects			
Response Time (R)	3	.1840	.8063
Variance Ratio (V)	2	.0217	.0950
R x V	6	.0854	.3742
Subjects (RV)	24	.2282	
Within Subjects			
Task Type (T)	2	1.2515	20.6379**
V x T	4	.0105	.1737
R x T	6	.2321	3.8268**
V x R x T	12	.0382	.6302
Subjects x T (RV)	48	.0606	

** $p < .01$

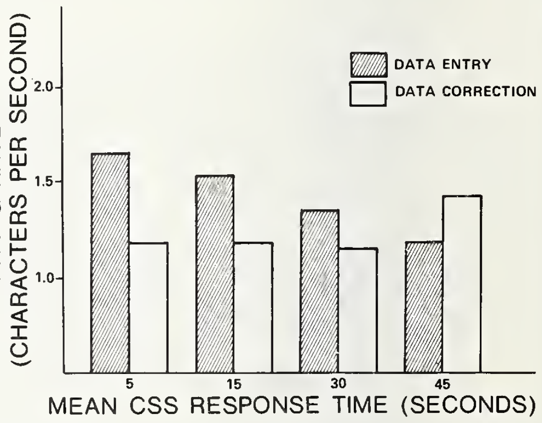


FIGURE 3 - OVERALL SPEED OF PERFORMANCE (CHARACTERS PER SECOND) IN THE DATA ENTRY AND DATA CORRECTION TASKS FOR THE FOUR EXPERIMENTAL GROUPS IN THE PILOT STUDY.

Results of the second study were, for the most part, congruent with those of the first. Here we found a significant effect of response time (F(3,12)=27.2,$p<.01$), a significant difference between the typing rate in the data entry and data correction tasks (F(1,4)=28.4, $p<.01$), and a significant interaction of response time and task type (F(3,12)=23.9, $p<.01$) (see Table 4). When this interaction was decomposed into its simple effects we found no significant effect in the data

Table 4
Analysis of Variance of
Overall Speed of Performance -
for Data entry and Data Correction Tasks
Last Third of Experimental Sessions

SOURCE	df	MS	F
Response Time (R)	3	0.3350	27.2**
Operators (O)	4	0.9574	30.3*
Task Type (T)	1	4.0022	28.4**
R x O	12	0.0123	0.3
R x T	3	0.1614	23.9**
O x T	4	0.1405	4.4**
R x O x T	12	0.0067	0.2
Reps (R x O x T)	80	0.0315	

** $p < .01$

entry task, and a significant relationship between overall typing rate and response time in the data correction task (F(3,12)=23.98, p<.01) (see Figure 4).

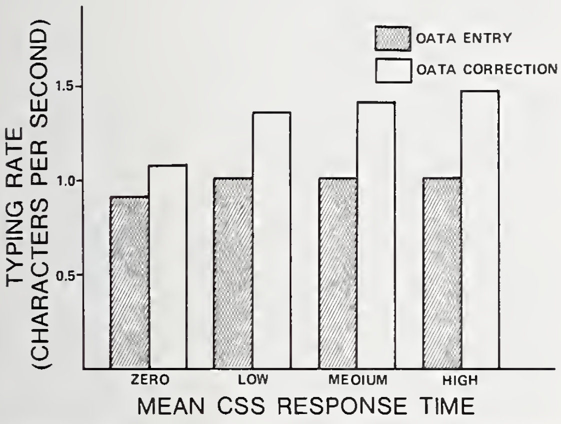


FIGURE 4 - OVERALL SPEEO OF PERFORMANCE (CHARACTERS PER SECOND) IN THE OATA ENTRY ANO DATA CORRECTION TASKS FOR THE FOUR CSS RESPONSE TIME GROUPS.

3.3 First Character Time

The reader will remember that the second study allowed us to look at the time required to type each character and further, that one of the major differences between the data entry task and the data correction task was that in the latter the line in error had to be identified by the participant. Assuming that the operators didn't stop in the middle of a line they were typing to search for more errors, but rather completed that line and then searched, the time between the computer prompt character and the first character typed (called first character time in this study), would represent the amount of time, in addition to the response time, that the participant needed to search for and prepare to correct the line in error. When we analyzed the first character times, we found a significant main effect of response time (F(3,12)=23.9, p<.01). Task type and the interaction between task type and response time were significant, (F(1,4)=42.1,p<.01 and F(3,12)=14.22,p<.01), respectively) (see Table 5). Analysis of simple effects of this interaction suggested that the response time effects were evident in the data correction task but not in the data entry task. A Newman-Keuls test indicated that the first character times were significantly longer in the zero condition than in all of the others. Apparently the operators used the system response time to search for errors since first character times were shorter in the low, medium, and high conditions. These data are illustrated in Figure 5.

Table 5
Analysis of Variance of
First Character Times for
Data Entry and Data Correction Tasks

SOURCE	df	MS	F
Response Time (R)	3	235.7	23.9**
Operators (O)	4	39.9	3.7**
Task Type (T)	1	1961.6	42.1**
R x O	12	9.8	0.9
R x T	3	107.9	14.2**
O x T	4	46.5	4.4**
R x O x T	12	7.5	0.7
Reps (R x O x T)	80	10.5	

** p <.01

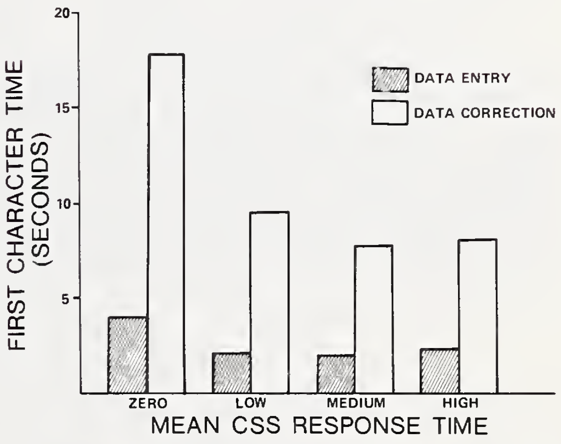


FIGURE 5 - MEAN FIRST CHARACTER TIMES (SECONDS) IN THE OATA ENTRY ANO OATA CORRECTION TASKS FOR THE FOUR CSS RESPONSE TIME GROUPS.

3.4 Speed of Performance Corrected for First Time

Realizing that the trends in the overall typing speed data reported earlier might simply be mirroring changes in first character time, we analyzed the typing rate data after the first character times for each line were removed. When these data were analyzed without the early session (sessions 1-8) included, the differences between the four response time conditions were eliminated (F(3,12)=1.25,p<.25). Task type was found to be a significant variable (F(1,4)=304.5,p<.01) indicating that the operators could key the characters for the data correction task much faster than for the data entry task, once they began typing (see Table 6).

Table 6
Analysis of Variance of
Speed of Performance Corrected
for First Character Time in the Data Entry
and Data Correction Tasks

SOURCE	df	MS	F
Response Time (R)	3	0.0710	4.2*
Operators (O)	4	1.5567	39.9**
Task Type (T)	1	21.8504	304.5**
R x O	12	0.0168	0.4
R x T	3	0.0464	1.9
O x T	4	0.0718	1.8
R x O x T	12	0.2329	0.5
Reps (R x O x T)	80	0.0139	

** $p < .01$
* $p < .05$

4. Discussion and Results

Considering the above studies, what conclusions can we draw concerning the effects of response time on operator performance? To assist us in drawing conclusions, it is very helpful to ascertain those aspects of the experiment which account for the most experimental variance, or which are the most important [10]. A number of authors have suggested that the importance of an experimental variable should be tested using strength of association statistics [11,12]. These statistics have been called "Utility Indices", and they illustrate the "practical significance" as opposed to the statistical significance of experimental variables [13,14,15,16].

Utility Indices appropriate to the designs of the respective studies were calculated [14] so that the importance of each variable could be estimated. These indices are summarized in Table 7.

Inspection of Table 7 reveals two points of interest. First, workload characteristics (task type) account for considerably more variance in this study than does any other parameter. Second, response time does appear to impact upon speed of performance. We will attempt to show that this effect is in fact a result of certain workload characteristics.

That various types of task impact on the speed and accuracy of performance is, in itself, not surprising. Consider the differences between the data entry and data correction tasks in terms of the demands they place on the operator. How might we explain the differences between the typing speeds in the data entry task and the data correction task, once the operators have begun typing?

Table 7
Proportion of Variance Attributable to Each Independent Variable

	Accuracy of Performance		Overall Typing Speed		First Character Time		Typing Speed Exclusive of First Character	
	Study 1	Study 2	Study 1	Study 2	Study 1	Study 2	Study 1	Study 2
Response Time	.0486	0	0	.111	NA	.149	NA	.0040
Variance Ratio	0	NA	0	NA	NA	NA	NA	NA
Task Type	0	.0692	.2959	.342	NA	.423	NA	.8296
Response Time x Task Type	.0863	0	.128	.044	NA	.965	NA	.0012
Response Time x Variance Ratio	.1634*	NA	0	NA	NA	NA	NA	NA
Variance Ratio x Task Type	.0056	NA	0	NA	NA	NA	NA	NA
Response Time x Variance Ratio x Task Type	0	NA	0	NA	0	NA	0	NA

*These data, indicating variance ratio effects, must be interpreted with caution since control condition data suggest that differences may be attributable to differences between subjects.

Obviously, the most significant difference between the two sets of codes is that the data entry codes were random while the codes employed in the data correction task were not. This suggests two important differences in terms of human performance. First, while there was no structure or meaning inherent in the data entry codes, the data correction codes were subsets of one of two higher level constructs, the alphabet and the set of cardinal numbers. Thus, while performing the data entry task, the operator was forced to attend specifically to each element of the code. Performance on the data correction task was not so limited. The operator was performing a much more familiar task, that is, one with which he/she had considerable experience, e.g., counting from 1 to 7. One might suggest that this task did not place as much of an attentional demand on the operator. The second difference relates to the skills the operator might bring to bear on the two tasks. In the data entry task all character sequences were equiprobable. Consequently, participants were less able to chain together common keystroke sequences, e.g., typing in bursts, and thereby increase their overall typing rate as they developed this skill.

We would suggest that these data have implications to computer system design in terms of specification of system load. That is, within the context of a model of time-sharing operational efficiency set forth by Brown and Klerer [17], those attributes of a task which allow greater speed of entry, such as we found in the data correction task, will also serve to place a greater effective load on the system. Further, if there are aspects of a task that provide for acquisition of skill, this load might be expected to increase somewhat over time.

One further implication of these studies relates to our findings with respect to first character time. The reader will remember that in the data correction task the operator had to search for errors, and that he/she had a tendency to utilize the computer delay times for searching, as evidenced by the increased first character times in the zero condition. We therefore see an attribute of the task moderating the effects of the response time of the system. At least one implication may be derived from this finding. That is, in tasks similar to the correction task we can see that all of the computer delay may not be lost to the total system, but rather may be utilized by the operator in a meaningful fashion.

5. Conclusions and Recommendations

Therefore, we would like to propose a system design philosophy which is based partially upon the results of these studies and partially upon our own systems experience. It has become clear that operators can use the computer response time delays efficiently in certain types of tasks. This means that short response times are not always necessary. We suggest that the computer system designers pay close attention to the type of work that the operators will be doing with the on-line system. The types of tasks to be performed should be studied and timed, so that distributions of times for various task elements (e.g., search times, keying time) can be ascertained.

We suggest, then, that the computer response times for those tasks be adjusted so that the mean response time is slightly less than the mean time necessary for the operators to prepare to enter the data. In addition, the distribution of the response times should be adjusted so that the operators are required to wait for the computer on no more than 10-15% of the interactions. This procedure should start us upon the road to building efficient systems which clearly take into account both the characteristics of on-line computers and the workload characteristics of the task being performed by the operators.

References

[1] Boies, S. J., & Gould, J. D. User performance in an interactive computer system, Proceedings of the Fifth Anual Princeton Conference on Information Sciences and Systems, (1971).

[2] Carbonell, J. R., Elkind, J. I., and Nickerson, R. S. On the psychological importance of time in a time sharing system, Human Factors, 10(2), 135-142, (1968).

[3] Miller, R. B. Response time in man-computer conversational transaction, IBM Technical Report, TR 00.1660-1, (January 29, 1968).

[4] Nickerson, R. S. Man-computer interaction: A challenge for human factors research, Ergonomics, 12(4), 417-501, (1969).

[5] Parson, H. M. The scope of human factors in computer-based data processing systems, Human Factors, 12(2), 1965-175, (1970).

[6] Shackel, B. Man-computer interaction--the contribution of the human sciences, Ergonomics, 12(4), 485-499, (1969).

[7] Simon, H. A. Reflections on time sharing from a user's point of view, Computer Science Research Review, Carnegie Institute of Technology, 43-51, (1966).

[8] Williams, C. M. System response time: A study of user's tolerance, IBM Technical Report, Advanced Systems Development Division, 17-272, (July 1973).

[9] Morefield, M. A., Weisen, R. A., Grossberg, M., Untema, D. B. Initial experiments on the effects of system delay on on-line problem solving, Lincoln Laboratory Technical Report, (June 25, 1969).

[10] Chapanis, A. Theory and methods for analyzing errors in man-machine systems. Annals of the New York Academy of Sciences, 1179-1203, (1951).

[11] Hayes, W. L. Statistics for psychologists. New York: Holt, Rinehart and Winston, (1965).

[12] Kirk, R. E. Experimental design: Procedures for the behavioral sciences. Belmont, California: Brooks/Cole, (1968).

[13] Gaebelein, J. W. & Sonderquist, D. R. Computational Formulae for Utility Indices. Department of Psychology, University of North Carolina at Greensboro, N. C. 27412, (August 1, 1974).

[14] Gaebelein, J. W., & Sonderquist, D. R. The utility of within-subjects variables: Estimates of strength. Manuscript accepted for publication: Education and Psychological Measurement, (1976).

[15] Dodd, D. H., & Schultz, R. F., Jr. Computational procedures for estimating magnitude of effect for some analysis of variance designs, Psychological Bulletin, 79, 391-395, (1973).

[16] Vaughn, G. M., & Corballis, M. S. Beyond tests of significance: Estimating strength of effects in selected ANOVA designs. Psychological Bulletin, 72 204-213, (1969).

[17] Brown, T. & Klerer, M. The effect of language design on time-sharing operational efficiency. International Journal of Man-Machine Studies, 7, 233-247, (1975).

APPENDIX I

```
START TASK  XXX*

PN5ZCF6OTD     QN6OYL8HØF     8ZRA9WTEBG     AB1Y2EXK5U

KTL6Z7RD1U     1VW8YGN4ØE     ØQYHFEC83N     U3SCIJM894

YHOS38FNEØ     JVB32WIXAG     3JTAV4WH18     XFØE28IYPW

LE7DGVKQ46     PCOV3SQUNI     J6KØWMIBDL     8WCGZSPVE4

19TKRZD57O     E2CI5XAB61     6ENU4K7I8M     V1HBNØWZEX

BF3XU1QØAS     2I3EVAJGBW     KP6W15Y8RT     IG2BX3WAVJ

HZCR7D1KPT     FMUL3RHZSV     4LDMZK9SI7     NE4XCBSV3O

7FHØUOL1M8     F9JUØDA6K7     QWLXTYJGØ3     M81HØNO6FL

VG3O6E2HBR     5PDTK94RZC     Q7W8JLU2M5     G8BØA6TXOH
```

APPENDIX II

DATA CORRECTION FILES

```
PRINT Ø1

ABCDEFG ZABCDEF Ø123456789Ø12 KLMYOPQRSTUVW

3456789 XYZABCDEFGH ABCDEFGHIJK PQRSTUVWXYZ

ABCDEFGH 23456789 OPQRSTUVWXYZ CDEFGHIJKLMN

HIJKLMNO XYZABCDRF 3456789Ø123 GHIJKLMNOPQR

PQRSTUVWX 456789Ø12 STUVWXYZA GHIJKLMNOPZRS

YZABCDEFG BCDEFGHIJ 3456789Ø163 TUVWXYZABCD

Ø123456789 LMNOPQRSTU GHIJKLMNOP JKLMNOPQRS


PRINT Ø2

QRSTUVWXYZ EFGHIJKLMN ABCDEFGHIJ Ø123456789

HAJKLMNOP KLMNOPQRSTU EFGHIJKLMNO 456789Ø12

QRSTUKWXY VWXYZABCD PQRSTUVWXYZAB RSTUVWXYZ

56789Ø123 MNOPQRSTUVW EFGHIJKLMNOP KLMNOPQR

ZABCDEFG 3456789Ø1234 EFGHIJKLMNOP CSEFGHIJ

HIJKLMNOPQR 56789Ø12345 9Ø1234367S9 QRSTUVW

STUVGXY 6789Ø12345678 XYZABCDEFGHIJ KLMNOPQ
```

FUNCTIONAL WORKLOAD CHARACTERIZATION

J. E. McNeece
R. J. Sobecki

U.S. Department of Agriculture
Office of Automated Data Systems
Washington, DC 20250

This paper addresses functional workload characterization in the benchmark process covering the: (1) past - several early references to standard benchmarks are reviewed and their findings summarized, as are two, large Federal computer procurements; (2) present - the current state of functional workload specification is briefly described, as is the approach currently being taken by the U.S. Department of Agriculture (USDA); and (3) future - the trends and problems uncovered in the literature gained through USDA experience, and resulting from discussions with other Government agencies and ADP equipment manufacturers are identified and discussed. Finally, those areas which need more study are outlined. These experiences and observations are presented with a view toward precipitating discussion. The main emphasis in workload characterization in this paper is on batch and on-line transaction processing utilizing a DBMS.

Key words: Benchmarking; computer procurement; computer selection; workload characterization.

1. Past

1.1. Review of References

The following briefly reviews three documents which have previously addressed various issues associated with functional workload characterization.

1.1.1. Recommendation D-14, Report of the Commission on Government Procurement--December 1972:

> D-14 - Develop and issue a set of standard benchmarks to be used as benchmarks for evaluating vendor ADPE proposals.

This recommendation was later established as an executive branch position in May 1974 as outlined in: "Proposed Executive Branch Position/Implementation for Recommendation D-14 of the Report of the Commission on Government Procurement," March 24, 1974.

This latter report raised the issue of functional workload characterization as follows:

> "A major feasibility question relates to the problem of being able to define workload characteristics in a manner suitable for selecting and modifying reference benchmarks and data to be representative of user's requirements."

Later, this paper will present an approach to functional workload characterization and its subsequent translation into standard benchmark functions or programs.

1.1.2. "Development of Standard Benchmarks," Department of the Army Pamphlet No. 18-10-2, Management Information Processing Systems Exchange, May 1973, pp. 1-8.

The above document addressed the advantages and disadvantages of using standard benchmarks to functionally characterize a workload in terms of a standard set of benchmark programs. Some of the advantages cited were:

(1) reduced preparation time and cost for users;

(2) reduced cost and response-time for suppliers;

(3) flexibility in use;

(4) wider base of users.

Some of the disadvantages cited were:

(1) initial development cost;

(2) nonuniversality;

(3) lack of user confidence.

1.1.3. Paul Oliver, et. al., "An Experiment in the Use of Synthetic Programs for System Benchmarking," Proceedings of the NCC, 1974, pp. 431-438.

The following conclusion from the above document bears on the workload characterization question:

> "Can a workload be profiled?
>
> We do not believe that it is possible to arrive at a generalized, comprehensive, and accurate model of system workloads except in the most trivial cases. We can certainly retrofit. That is, we can accept a workload definition based on the synthetic program parameters. We also believe that this need not impede the use of synthetic programs in benchmarks. In this, we strongly support the view expressed by J. C. Strauss. In a recent paper on the use of natural benchmarks, he stated that based in part on prior experience and on the difficulties encountered, 'it was felt more important that the behavior of the benchmarks be well understood and cover a broad range of important system features than that the complete benchmark series be representative of the general workload.'"

1.2. Past Major Systems Acquisitions

Two major military procurements, the World-Wide Military Command and Control Systems (WWMCCS) and the Military Personnel Center (MILPERCEN) Project 70X had very similar benchmarking approaches and were examples of the type of specifications which were prevalent in the late 1960's and early 1970's. These were a combination of resource (hardware) requirements and functional workload characterization mainly developed from historical data. The benchmark problems were similar in that the vendor had to first run all problems in a serial-mode and then run selected problems in a multiprogramming mode. Included were remote job entry demonstrations and interactive terminal demonstrations.

2. Present

There are at least four approaches being taken today in the development of workload specifications:

(1) In an open shop, one approach is to take a particular time-slice of a monthly workload (e.g., month end) and, assuming these programs are a true representation of the total workload, select them as the benchmark.

(2) Another process is to take programs and data from existing systems and use them as the benchmark. The problem with this approach is that old systems and programs may be used which are not representative of the new system.

(3) A third methodology is to quantify synthetic functions which have been identified by the user as representative of his ADP workload. (A file generator is often used in connection with this approach.) This approach is currently being employed by the U. S. Department of Agriculture (USDA) on current and future major system procurements.

(4) A fourth approach consists of some combination of the above.

Figures 1 through 4 illustrate standard forms used by USDA to implement the third approach. The following discussion describes each form.

Figure 1 - Illustrates a typical list of functions, or standard programs, that might be considered in a workload characterization effort. The list is established in an interactive manner in that for a specific workload characterization and its resulting benchmark, some new functions may need to be added and defined, while some functions of a previous benchmark may be excluded due to the nature of the processing currently being characterized. The third column in this figure identifies which of the eight quantification methods shown at the bottom of the figure is represented by the benchmark program.

Figure 2 - Depicts the input data files, their logical organization, and record sizes within each file, and cross-references the benchmark programs to the files which they use.

Figure 3 - Used to quantify events in terms of meaningful functions that occur in an agency's workload. Each event is given a code, a name, and also the volume per year. In addition, a percentage per month is given for each month for all years.

Figure 4 - As indicated in Figure 3, a code and a name are assigned to each quantifiable event as are workload projections for the system's life. Figure 4 is used to map quantifiable events into the identified functions (i.e., standard benchmark programs). Each event is broken down into ADP systems and, where necessary, subsystems. Once the events have been broken down, this figure is used to produce quantification data. Quantification data is produced at the system or subsystem level, in terms of the standard functions and quantification methods shown in Figure 1.

Discussions with other Federal agencies, who are planning major system procurements, indicate that they are using, all or part of, the previously mentioned approaches to workload characterization. In discussions with two agencies, the workload characterization was not presented to the vendors in the usual way. In one case all offerors were given a set of flow charts and data descriptions from which the vendors program the benchmark problems. The benchmark is then run using a subset of live data files. In the other case, two vendors will be chosen after a technical and cost evaluation. Each vendor will then proceed, with Government funding, to redesign and program existing systems. The new systems will then be benchmarked in phases, with award being made to the vendor with the lowest life-cycle cost.

3. Future

Benchmarks, in both the Federal and private sectors, seem to be evolving into on-line interactive demonstrations.

3.1. Problems

Some problems associated with the functional workload demonstration fall into two categories. First, problems with the data bases and secondly, problems with the programs. The problems uncovered in these areas are:

Data Bases

- Manufacturers have difficulty in converting data bases to their equipment.
- Size of data bases as presented in benchmarks are too large (e.g. 30-40 reels of magnetic tape).

Programs

- Nonstandard language constructs make conversion difficult on the part of the vendors. Thus, the major effort of the vendor's benchmark team is sometimes spent on conversion, rather than on the throughput test of the offeror's equipment.
- A major problem associated with benchmarking in general is that benchmark costs are becoming too high. Vendors are taking a careful look at Government procurements and their associated benchmarks to see what the costs will be to bid relative to the chances of a successful conclusion. Some ways in which benchmark costs could be kept down are:

  (1) standard synthetic benchmark programs;

  (2) use of file or data base generators;

  (3) demonstration of standard system functions such as payroll, inventory, etc.

  It should be noted that several ADP equipment manufacturers agree that a file or data base generator should be used where possible in creation of data bases.

3.2 Functional Workload Characterization

The following is an approach to functional workload characterization which appears promising:

(1) Identify quantifiable events which represent agency functions. These functions must be major agency program or administrative functions.

(2) Identify and define benchmark ADP operations. A benchmark ADP operation will be directly and explicitly represented in the benchmark workload mix by a synthetic program or some other workload category.

(3) The volume for each agency quantifiable event identified must be projected over the scheduled life of the computer system. Quantification for each year is required for each item.

(4) Determine, by analytical means, the relationships between quantifiable events specified in activity (1) above, and the benchmark ADP operations identified in activity (2).

(5) Select peak workload months. The objective of this activity is to identify the peak months of computer workload. This is done by tallying workload for each month from workload projection and mapping forms. Management guidance must be obtained as to the desired level of capability to support peak periods and to determine how much flattening of peaks is appropriate.

(6) Quantify peak periods. Using the data derived in activities (3) and (4), calculate the aggregate number of iterations of each benchmark operation required to perform the workload during the peak periods.

(7) Determine benchmark transaction characteristics. Here, a transaction may be defined as a coded representation of an event which triggers one iteration of one of the benchmark ADP operations.

(8) Determine data storage needs and characteristics of the data base. This activity will determine the size of the data base(s) to be stored in the object computer system, in addition to characteristics of the major data files. This also requires taking

measurements to assure that the benchmark adequately represents these data characteristics.

3.3. Areas Needing More Study

In general, the functional workload characterization approach to benchmarking needs more study by both the ADP industry and the users.

Those areas in need of further study are:

(1) Identification of a sufficient number of standard synthetic benchmark functions and programs.

(2) In adapting a benchmark to a given system, what changes will be allowed by the vendors?

(3) More ADP industry participation in development of standard benchmark programs and data base generators.

(4) In developing benchmarks for remote computing services, what type of workload characterization, synthetic or actual, gives the most accurate representation of an offeror's capabilities? What is the best way to test an offeror's network capability; i.e., concentration, lines, etc., in order to insure that adequate services can be provided?

(5) Distributed processing networks are now beginning to emerge. How can these systems be benchmarked in order to give meaningful results?

It is hoped that this paper will stimulate further discussion on the state and future of functional workload characterization.

Category of Processing	Benchmark Program	Quantification Method	Quantification
1. On-Line Retrievals-Serial Indexed (by index)	ORSIND	O	
2. On-Line Retrievals-Indexed	ORNDEX	O	
3. On-Line Updates-Serial Indexed (by index)	OUSIND	O	
4. On-Line Updates-Indexed	OUNDEX	O	
5. On-Line Data Entry	OENTXX	O	
6. On-Line Computations	OCOMPU	O	
7. Batched Retrievals-Serial Indexed (by index)	BRSIND	B	
8. Batched Updates-Serial Indexed	BUSIND	B	
9. Batched Updates-Indexed	BUNDEX	B	
10. Batched Updates of Multiple Files (by index)	BUMULT	B	
11. Batched Retrieval-Serial	BRSERL	S	
12. Batched Updates-Serial	BUSERL	S	
13. Batched Retrieval-Serial Indexed (Sequentially)	BRSEQU	S	
14. Batched Serial Edit	BRSEDIT	B	
15. Batched Retrieval-Indexed	BRNDEX	R	
16. Batched Computational	BCOMPU	I	
17. Sort	Vendor Software	S	
18. Data Base Inquiries	Vendor Software	D/R	
19. COBOL Compilations	Vendor Software	P	
20. FORTRAN Compilations	Vendor Software	P	
21. Interactive Program Development (Text Edit)	Vendor Software	L	
22. Report Generator	Vendor Software	S	

Quantification Methods:

O = On-Line Transactions

B = Batched Transactions (local and remote)

S = Records Passed on Serial Input File

L = Lines of Code for Text Editing

R = Records Retrieved via Multi-Index Relational Operators Technique

I = Iterations of Program

D = Data Base Inquiries

P = Programs Compiled, 500 Statements Average

Figure 1. Workload summary.

Data File	Logical Organization	Record Size	Using Programs
FILEØ1A	Serial Indexed - Unique Key Value	320	OUSIND BUSIND BRSEQU1
FILEØ2A, B, C	Hierarchical - 3 levels	438 MIN 686 MAX	ORNDEX BUNDEX BUMULT
FILEØ2D, E, F	Hierarchical - 3 levels	1,057 MIN 1,176 MAX	OUNDEX BUMULT
FILEØ2G, H, I	Hierarchical - 3 levels	960 MIN 1,133 MAX	BUMULT
FILEØ3	Indexed 3 Keys - Nonunique Key Values	75	BRNDEX
FILEØ5	Serial	2,913	BRSERL
FILEØ6	Serial	136	BUSERL
FILEØ7A	Serial Indexed - Unique Key Value	244	ORSIND BRSIND BRSEQU7
FILEØ8	Serial Indexed - Unique Key Values	54	BRNDEX

Figure 2. Data base characteristics.

USDA QUANTIFIABLE BENCHMARK EVENTS
AND WORKLOAD PROJECTIONS

Agency ________ Date ________

Quantifiable Event		Volume Per Year						Percent Per Month											
CODE	NAME	1977	1978	1979	1980	1981	1982	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC

Figure 3. Workload projection form.

USDA ASSIGNMENT OF BENCHMARK TO ADP OPERATIONS Agency ____________ Date ____________

Code for Quantifiable Event	ADP System or Subsystem	ADP Representative and Phone Number	Category	Disp/Freq.	1. ORSIND	2. ORNDEX	3. OUSIND	4. OUNDEX	5. OENTXX	6. OCOMPU	7. BRSIND	8. BUSIND	9. BUNDEX	10. BUMULT	11. BRSERL	12. BUSERL	13. BRSEQU	14. BSEDIT	15. BRNDEX	16. BCOMPU	17. SORT Merge	18. DBL Queries	19. Report Gen	20. COBOL	21. FORTRAN	22. IPD

Figure 4. Workload mapping form.

SOME RESULTS ON THE CLUSTERING APPROACH TO WORKLOAD MODELLING

A. K. Agrawala
J. M. Mohr

Department of Computer Science
University of Maryland
College Park, Maryland

In this paper we discuss two of the issues involved in the use of clustering techniques to characterize a computer system's workload. First we examine the results of clustering one data set with three distinctly different types of feature sets. In the second half of this paper we present results showing that the clusters that are obtained are stable and that they represent natural groupings in the workstep population.

Key words: Clustering; workload characterization.

1. Introduction

Performance evaluation is fast becoming an essential part of computer system management. However, the performance of the system depends on the specific workload it handles. It is necessary for the accurate performance evaluation of a computer system to have a good knowledge of the system's workload [1][1]. The importance of the workload to any evaluation of modelling effort is well summarized by the following: "Blessed is he who found his computer's workload. Let him ask no other blessedness." [2]

A modern computer operating in a general purpose computing environment is often shared by a large population of users, who use it to satisfy a wide variety of computational needs. Conceptually, the overall computational needs of such a user population comprise the workload of an installation. Due to the large variations in the load presented by the diverse user community, one can hardly use the whole workload in any system study. Selecting a representative test workload via a validated model offers a viable alternative.

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

This research was supported in part by the National Aeronautics and Space Administration, Goddard Space Flight Center, under Grant NASA #NAS 5-22878, and in part by the National Science Foundation, Control and Automation Branch, Engineering Division, under Grant ENG 73-04099, to the Department of Computer Science, University of Maryland, College Park, Maryland. Also the cooperation extended by the Computer Science Center of the University of Maryland during this work is gratefully acknowledged.

A framework for creating workload models with various degrees of aggregation was recently proposed [3]. The approach taken was to define the aggregation in terms of "worksteps," which are the smallest element of the workload to be used in the model. A workstep may be a job, a program, or a transaction depending upon which is the most appropriate unit for study. The characteristics of a workstep are used in creating the workload models. The basic approach suggested in [3] was to describe a workstep in terms of the system resources required to satisfy it, the location in a network from which the workstep originated and the time at which the workstep was submitted. A workload model may then consist of a probabilistic description of the workstep population.

For a centralized installation we may choose to describe a workstep only by a vector x of resource requirements where the jth component of x corresponds to the workstep's usage of the jth system resource. The workload model now consists of a probability distribution $p(x)$. For any system, such a model has to be formulated on the basis of empirical data gathered for that system. In general, $p(x)$ is likely to be a very complex distribution. A simplification proposed in [3] was to treat $p(x)$ as a mixture distribution

$$p(x) = \sum_{i=1}^{m} p(x/c_i)p(c_i) \qquad (1)$$

where the population of worksteps is divided into m classes c_i, $i=1,m$ each having a distribution $p(x/c_i)$. An advantage can only be gained by such an approach if $p(x/c_i)$ is a "well behaved" function. For example $p(x/c_i)$ may be unimodal.

In the environment of a computer system the distribution of $p(x/c_i)$ is likely to be non-parametric, so that in order to use the mixture distribution approach one may have to apply a technique such as clustering. This way the observed sample worksteps may be grouped into classes to decompose the multimodal $p(x)$ into several unimodal $p(x/c_i)$'s. Some results on the feasibility of the clustering approach were presented in [3]. Clearly, the results obtained from such an approach depend on the clustering technique used as well as on the set of features used (as components of the vector x) to describe the workstep. In this paper, we explore some questions regarding the stability of the clustering results as well as the effect of the choice of features on the final clustering obtained. The results presented are based on the same clustering technique as was used in [3]. The clustering technique is briefly described in Section 2. The effects of clustering using different feature sets is presented in Section 3. In Section 4, we present results of tests on the stability of this approach.

2. Clustering Technique

Here we briefly describe the clustering technique used. The technique used is a variant of the k-means technique discussed by Anderberg [4] and is described in detail in the appendix of [3]. The following steps are involved in this technique.

a. Transformation - It is desirable to perform a logarithmic transformation on several of the variables used as components of the vector x. For example, in some sense two jobs requiring one second and two seconds of CPU time have the same degree of similarity as two jobs requiring one minute and two minutes of CPU time.

b. Scaling - As the original scale of the components of x can vary by six orders of magnitude, an appropriate scaling is desirable. To avoid the problems caused by outliers, we scale the 98% percentile point to the value 10.0.

c. Similarity Measure - To allow for the importance of a given component of the vector x to vary depending on the class, we use as a similarity measure a weighted distance measure. The inverse of the variance along an axis is used as the weighting factor. The effect of this measure is that we get hyper-ellipsoidal clusters, rather than the hyper-spheroidal clusters that would have been obtained had we used a standard Euclidean distance.

d. Clustering Algorithm - A multipass approach is used in which, starting from a set of seed values (see Section 4.2 for the effect of varying the seed values), the points are assigned to clusters based on the weighted distance between the points and cluster means. Points far away from all of the known clusters are allowed to form their own clusters. The scheme has converged when a complete pass is made over the data and no sample points change their cluster assignments on a given pass.

3. Feature Selection for Clustering

As mentioned earlier, a workstep is described by a vector x, where each component of x represents the resources of a particular type required by this workstep. There are no inherent constraints on the dimensionality of x. A rather large number of measurements are usually available for a workstep. However, in order to make the computations and data required for clustering reasonable we have to limit the dimensionality of x. In other words, out of all features which may be used to describe a workstep, we have to select a reasonable number of them.

Pattern recognition literature has a rather large number of papers published on the topics of feature selection [5,6]. A natural question to examine here is the applicability of such feature selection techniques to the problem of workload modelling. In a typical pattern recognition problem the class membership of the samples is known *a priori*. The problem facing the analyst is to choose a set of features and a classification approach such that the class of a sample may be inferred from the data that has been extracted about a sample. However, there are no such *a priori* classes defined for the workload models presented here. Instead we are only trying to determine whether any natural groupings exist in the data [9]. But, the characteristics of a workstep defined by a subset of features change as we change the subset of features. Accordingly, the groupings are likely to change. It is not clear whether one grouping can be claimed to be significantly better than any other. Each grouping is formed based on the perspective given by the feature set. Therefore, attempting to reduce the number of features may depend on the type of grouping sought. In order to verify this we conducted a series of experiments whose results are described below.

3.1. Experimental Results

In the results presented in this paper, we used the data from the accounting log of the Univac 1108 installation at the Computer Science Center of the University of Maryland.[2] The data was obtained from a system whose configuration is described in Figure 1 and was collected for November 11, 1976. A job was used as a workstep and a total of 1342 jobs were used in this study.

At present our feature extraction routine extracts 64 different features about each job on the accounting log. Of these, we normally use 5 to 9 features for any clustering. We manually select which of the observed features are to be included in the feature set for each experiment. In order to study the effect of feature sets on clustering, we selected 3 sets of features from the data and performed three independent clustering runs using the approach described above. The results of the clustering for each of three feature sets are presented in Figures 2-5. Figures 2-4 describe, for each feature set, the number of points contained in each cluster, the mean feature values for each cluster, and the standard deviation of the feature values for each cluster. All of the feature values presented are in the scaled space. Recall that the scaling is performed such that 98% of the values of each feature lie in the range 0-10. Figure 5 contains the confusion matrices that resulted from the comparison of the output of three clustering runs (see Section 3.2.4 for a discussion of the confusion matrices). Let us analyze these results.

3.2. Analysis of Results

Three separate sets of features were extracted from the raw data set and used to cluster the worksteps. Let us consider the results obtained for each of these feature sets.

[2]The accuracy of the accounting log data was ascertained first [7,8].

3.2.1 Resource Consumption Feature Set

This set consisted of 6 features:

1) number of 512-word core blocks;
2) amount of CPU time;
3) executive request & control card charges (ER&CC charges);
4) amount of DRUM I/O;
5) amount of DISK I/O;
6) amount of TAPE I/O.

The ER&CC charges record the amount of service in 200-μsec units that the user job requested of the operating system. The user is charged for the processing of each control card and for certain functions, such as I/O that must be performed by the system. The unit used for the I/O charges is words, where the number of words charged to the user for an I/O operation is the number of words actually transferred plus the number of words that could have been transferred in an amount of time equal to the latency time plus the seek time of the device being used. All I/O to temporary files is charged as DRUM I/O while all I/O to catalogued files is charged as DISK I/O.

This feature set distinguishes between jobs solely on the basis of the total amount of hardware resources used. Therefore, we obtained clusters based upon resource usage alone. The mean and standard deviations of these clusters are presented in Figure 2. Since the data was collected in a university environment, we found a cluster of "small student" type jobs (#1). The jobs in this cluster used a relatively small amount of resources and were distinguished in that they did no TAPE I/O and virtually no DISK I/O. This cluster also contained some jobs submitted by more advanced users. These jobs typically did some trivial operation such as inquire about the status of some process, or list the contents of a file and then terminate. We also found a variety of clusters consisting of research users or students in advanced courses. These jobs used a moderate to heavy amount of all resources and consisted of repeated edits, compilations, link edits, and executions of user programs (#2,#6). Since a relatively small number of users perform TAPE I/O, we observed pairs of clusters that were differentiated only by the presence or absence of TAPE I/O (#9,#11). We also observed clusters made up of the same group of users, typically timesharing users, in which the basic difference between the jobs in the two clusters was that the jobs in one cluster used approximately twice as much of every resource, except core blocks, as the jobs in the other cluster (#8,#3). The use of the various resources is similar for the two clusters and an examination of the log entries for the job entries in the two clusters reveal that the users were basically doing similar types of cycles, such as a edit, compile, link edit, execute cycle or a data preparation, execute, data analysis cycle; the basic difference between the two clusters being the number of times through the cycle or whether the user signed on and was active for a two-hour or a four-hour session. We also found a cluster of jobs (#10) that consumed large amounts of all resources. Members of this cluster would include members of the system staff generating a new version of the EXEC or a very large research user whose job was active most of the day.

3.2.2 Percentage of Program Types Feature Set

This feature set contained five features. For the purposes of this experiment, we divided all of the programs into four categories. The five features used were the number of files assigned and the percentage of the programs in each job that came from each category. The four categories were:

1) program development;
2) data analysis programs from the system libraries;
3) general overhead programs;
4) executions of user written programs.

The program development category included all calls to the text editor, any compiler, and the linkage editor. The data analysis programs from the system libraries would include processors such as SPSS or BMD. General overhead programs would be that group of programs needed to maintain files; inquire about the status of the system, files, runs, or accounts; and to generally enable the user to do program development or data analysis. The final category included any program run by the user that was not found in any of the system libraries. The programs in these categories were termed Type 1, Type 2, Type 3, and Type 4 programs. The types of clusters obtained using this feature set was markedly different from the clusterings obtained using the hardware resource consumption feature set. If a job cycled through some sequence of programs, the clusterings based upon the "resource consumption feature set" would assign the job to one of several clusters, based upon the number of times the job cycled through the sequence of programs, while the clusterings based on the "percentage of program type feature set" would assign the job to the same cluster regardless of the number of cycles executed. The results of clustering using this latter feature are presented in figure 3.

We found four clusters (#8,#7,#3,#6) each of which had one of the four types of programs as its dominant member. This dominant program type accounted for at least 70% (#8) of the program executions and in some cases accounted for 95% (#3,#6,#7) or more of the program executions in that cluster. Most clusters showed a relatively large percentage of Type 3 or general overhead programs. This is due to the fact that no matter which of the other three types of programs dominates the cluster, certain overhead functions,such as assigning files,must be performed.

We observed a negative correlation between the occurrence of Type 2 and Type 4 programs in most clusters. This is reasonable in that the user of SPSS or a similar package is unlikely to execute user written programs. We also observed a program development cluster (#8) that consisted of roughly 70% Type 1 programs and 22% Type 4 programs, respectively.

The clusters obtained using this feature set reveal a great deal of information about the users of the systems and the types of operations they perform.

3.2.3. Rate of Resource Consumption Feature Set

This feature set consisted of five features:

1) CPU time/SUP;
2) ER&CC charges/SUP;
3) DRUM I/O/SUP;
4) DISK I/O/SUP;
5) TAPE I/O/SUP.

Univac defines the processing time of a job in Standard Units of Processing (SUP's) as the amount of time that it would have taken to run the job on a monoprogrammed, uniprocessing system with no overlap between CPU use and I/O. The SUP charges for a run are equal to the sum of its charges for CPU use, ER&CC's, DRUM I/O, DISK I/O, and TAPE I/O. The SUP is the basic unit of chargeable time on Univac 1100 series operating systems [7,8]. The feature CPU time/SUP would be the percentage of the run's chargeable time that was spent using the processor.

While the value of the features for CPU time/SUP and ER&CC's/SUP corresponds to the percentage of the run's chargeable time that was attributable to these sources, this is not quite true for the I/O counts. This is due to the fact that the I/O counts are stored on the log tapes as the number of words transferred and are converted to times for the purposes of the SUP calculation. We would have multiplied the number of words transferred by the transfer rate of the device, which would have made the three I/O related features correspond to percentages of chargeable time. This was not done, as it would have no effect on the final clusterings obtained (this operation would only entail a linear transformation of the data which would have been totally reversed by the scaling routines). The results of clustering using this feature set are presented in figure 4.

The features in this feature set in no way measured the gross amount of resources used and are similar to the features in the "program type feature set" in that if two programs differed only in the number of times they cycled through a given sequence of programs, they would both appear in the same cluster. We found several types of clusters. First, we found distinct clusters for I/O bound (#5,#6,#1), CPU bound (#4), and balanced jobs (#2). Second, in the clusters for the I/O bound jobs, we found clusters dominated by each of the three I/O device types (#5,#6,#1). Several clusters appeared with a given balance between CPU and I/O use, but these clusters were distinguished by the relative amounts of chargeable I/O to the various types of I/O devices.

This feature set can be used to produce a workload model that could be used for a performance improvement study, as this type of study is usually most concerned with the utilization rates of each of the system resources.

3.2.4 Confusion Matrices

In a confusion matrix the value at location (i,j) represents the number of samples assigned to cluster i when the data was clustered using the first feature set and that was assigned to cluster j when the data was clustered using the second feature set. Note that no correspondence between cluster numbers could be expected in independent clustering runs.

The pairwise confusion matrices for the three feature sets are presented in figure 5. From these confusion matrices it can be observed that as the feature set changes the number of clusters may change and the membership of each cluster may change. While at times a significant number of samples from one cluster run become members of the same cluster on the second cluster run, in many cases the members of one cluster are assigned to a wide variety of clusters on the second cluster run. This is not merely the result of clusters merging and splitting, but rather this phenomenon is the result of fundamental differences in the relative similarity between points because of the differences in the feature set.

From the confusion matrices of figure 5, we conclude that the clusterings based upon these three feature sets represent different groupings in the population. The type of grouping desirable for a particular study has to be the deciding factor in selecting a feature set. While the three feature sets discussed above were disjoint, we do not mean to imply that one could not or should not combine features from any two or even all three of the feature sets described above. However, some care should be taken in the selection of the feature set to insure its appropriateness to the study being conducted.

On each clustering run, we have been assuming that the clusters found represent natural groupings in the population. Before this may be assumed we must analyze the stability of the clustering approach used. We next present some results regarding the stability of our approach.

4. Stability of Clustering Results

When clustering techniques are used to group points in a multidimensional space, one has to verify that the classes or groups found by the clustering technique represent natural groupings in the data. The first step in this direction is to have enough samples to avoid certain dimensionality and sample size problems [5,6].

For the stability tests reported in this section, we used the resource consumption feature set. The clustering technique resulted in 11 clusters whose basic parameters are presented in figure 2. Several aspects of these clustering results were examined.

4.1. Unimodality of $p(x/c_i)$

An implicit assumption in the mixture distribution approach has been that $p(x/c_i)$ will yield a unimodal distribution. After clustering we get a set of samples belonging to a cluster defined in an 11-dimensional space. We would like to ascertain that its popluation has a unimodal distribution. It should be pointed out that if the parameters of the clustering technique are not adjusted properly, multimodal $p(x/c_i)$'s are quite likely.

To test the unimodality of the cluster population, we plotted the histograms of the sample point distances to the mean on a scale which would yield a horizontal straight line for a uniform distribution of sample points. One of these histograms for the clusters of the November 11, 1976, data is presented in figure 6. We note that the histogram shows a monotonically decreasing function which is what we would expect to observe if the population distribution was unimodal.

4.2. Effect of Seed Values

The clustering technique starts with a set of initial seed values for forming its clusters. If clear groupings exist, then the final clusters observed should not depend on the initial seed values. In an effort to confirm this, we made several runs on the same data with different seeds. The results of two such runs are summarized in the form of confusion matrices in figure 7.

From the results presented in figure 7, we note that less than 5% of the points changed their assignments due to different seed values. Such a difference can be explained in terms of very minor readjustments of the cluster parameters. It should be noted that this confusion matrix shows a marked contrast to the confusion matrices presented in figure 5.

Figure 8 shows the components of the individual cluster means and the distances between corresponding means for one feature set. Similar results were obtained for the other two feature sets. We note that the effect of starting the clustering routines with different seed values has almost no effect on the final clusters obtained. A change in seed points will however change the number of the cluster the points are assigned to.

4.3. Convergence

Normally we terminate a clustering run when less than 5% of the sample points change their cluster assignment from one pass to the next. Usually this process does not take more than 10 passes. To see what would happen if we let it run for many more passes, we let several runs go for up to 20 passes or until no sample points changed their cluster assignments. We observed that the number of points which changed decreased consistently, and typically after 15 to 19 passes absolute convergence was reached (when no points changed their assignements in successive runs).

4.4. Independent Test Set

All the results reported so far were based on tests performed on the same data as that used for the creation of the clusters, i.e., the training data set. If the results of the clustering were natural groupings, they should exist in data observed at other times, i.e., an independent test set of data. To confirm this, we used the data from November 12-13, 1976, as the independent test data. The changes in the workload from day to day are substantial enough that the workloads in any two non-overlapping intervals may be considered independent, they being independent samples from the same population. It should be pointed out that while the population characteristics change over a semester, they are not expected to change substantially on consecutive days. That is why in our experiments we used data from Nov. 11 for training and that from Nov. 12 and 13 for testing. Using the clusters observed from the training set as seeds, we made a pass at the test set. The results are summarized in figure 9. These results show that only a very small number of points fell outside of any of the clusters from the training set and that the means of the training set clusters moved an insignificant distance.

From the results presented in this section, we conclude that the data has natural groupings which are being uncovered by the clustering technique used and that the results are reliable and stable. Of course, an additional test to be put to the observed clusters is to check if such a grouping makes sense. Taking into account more detailed information about the sample points than that available in the vector x, we have been able to assign a meaningful description to virtually all of the clusters rather easily.

5. Concluding Remarks

In this paper, we have examined two aspects of the application of clustering techniques to the workload characterization problem. We have presented results of experiments that show that the clusters obtained from the log tape data reflect natural groupings of the sample populations and that they appear to be quite stable and reliable.

The stability of the clusters is certainly sufficient to allow their use in a workload characterization study.

In this paper we also presented the results of an experiment in which the same data was clustered using three disjoint feature sets. The results showed that three very different sets of clusters were obtained. Each of these clusterings represented the natural groupings in the multidimensional space from which they were drawn. The actual feature set used in a study, therefore, should be determined on the basis of the type of grouping desired. A better understanding of the relationship between the features and the groupings obtained may be desirable.

References

[1] Svobodova, L., *Computer Performance Measurement and Evaluation Methods: Analysis and Applications*, Elsevier Publishing Co., Inc., New York, 1976.

[2] Ferrari, D., Workload Characterization and Selection in Computer Performance Measurement, *Computer*, July/August 1972, pp. 18-24.

[3] Agrawala, A. K., Mohr, J. M. and Bryant, R. M., An Approach to the Workload Characterization Problem, *Computer*, June 1976, pp. 18-32.

[4] Anderberg, M. R., *Cluster Analysis for Applications*, Academic Press, New York, 1973.

[5] Kanal, L. N., Patterns in Pattern Recognition: 1968-1974, *IEEE Trans. on Information Theory*, Vol. IT-20, No. 6, November 1974, pp. 697-722.

[6] Agrawala, A. K., *Machine Recognition of Patterns*, IEEE Press, New York, 1977.

[7] Mohr, J. M., Agrawala, A. K., Flanagan, J. F., The EXEC-8 Log System, Part I - Description, *Computer Science Technical Report Series*, TR-434, University of Maryland, January 1976.

[8] Mohr, J. M., Agrawala, A. K., Flanagan, J. F., The EXEC-8 Log System, Part II-Error Analysis, *Computer Science Technical Report Series*, TR-473, University of Maryland, August 1976.

[9] Agrawala, A. K., Mohr, J. M., The Relationship Between the Pattern Recognition Problem and the Workload Characterization Problem, accepted for publication in *Proceedings of 1977 SIGMETRICS/CMG VIII*, Washington, D.C.

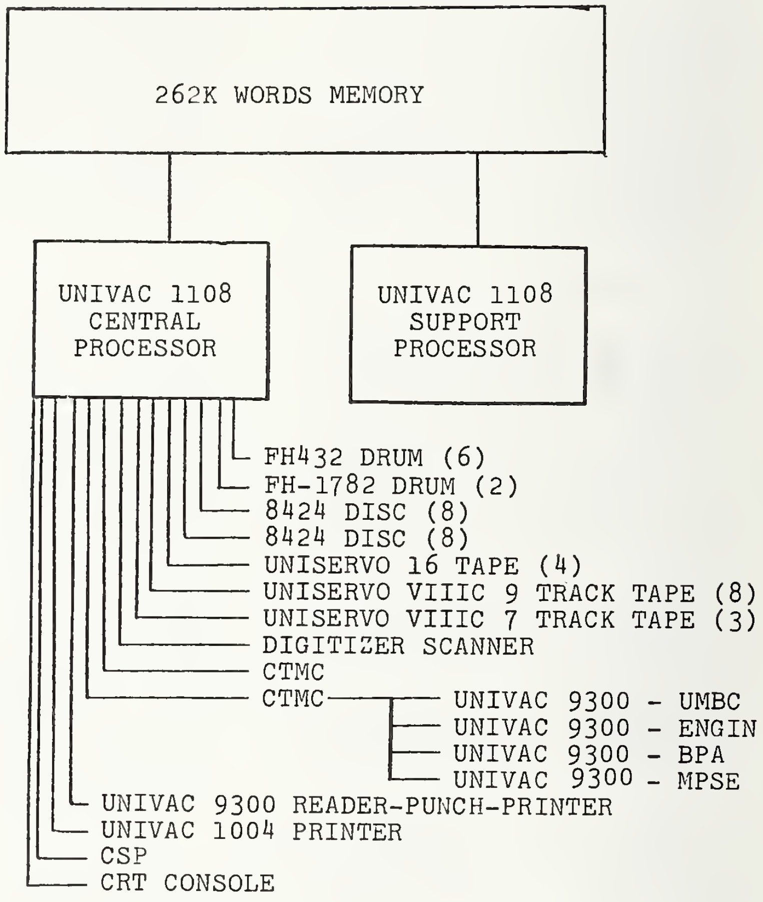


FIGURE 1. Configuration of the University of Maryland Univac 1108.

Cluster #	# Points	Core Blocks	CPU Time	ER&CC Charges	DRUM I/O	TAPE I/O	DISK I/O
1	243	0.662 (0.554)	0.061 (0.109)	2.426 (1.700)	1.509 (1.274)	0.017 (0.157)	0.245 (0.548)
2	74	3.013 (0.967)	4.482 (1.173)	7.876 (0.744)	7.769 (1.224)	6.366 (1.618)	6.650 (0.926)
3	215	3.086 (1.796)	3.793 (2.410)	6.817 (0.972)	5.922 (1.881)	0.025 (0.159)	6.072 (1.976)
4	38	3.789 (1.721)	4.042 (1.778)	6.380 (0.608)	3.852 (0.938)	5.501 (2.204)	6.326 (1.997)
5	117	4.151 (1.119)	3.567 (2.357)	4.208 (0.691)	7.005 (0.513)	0.009 (0.0976)	1.951 (1.267)
6	114	1.918 (0.540)	1.077 (0.723)	5.422 (1.142)	4.028 (1.510)	5.665 (2.029)	3.524 (1.693)
7	29	2.662 (0.666)	6.142 (1.139)	9.048 (0.999)	8.120 (1.282)	9.954 (1.441)	7.675 (1.571)
8	348	4.781 (3.340)	1.586 (1.666)	3.805 (1.117)	2.458 (1.550)	0.006 (0.0775)	4.401 (1.352)
9	60	3.312 (1.349)	7.430 (1.922)	9.178 (0.853)	9.049 (0.998)	0.099 (0.335)	8.338 (1.205)
10	18	4.479 (1.316)	9.432 (1.440)	9.646 (1.395)	9.907 (1.022)	8.782 (2.633)	8.571 (1.746)
11	31	2.874 (1.302)	7.056 (1.594)	9.421 (0.722)	8.515 (1.369)	5.223 (1.548)	8.764 (1.059)

Figure 2. Cluster Means and (Standard Deviations) for the Resource Consumption Feature Set.

Cluster #	# of Points	# of Files	% of TYPE 1	% of TYPE 2	% of TYPE 3	% of TYPE 4
1	86	0.038 (0.126)	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)
2	88	2.015 (1.120)	0.231 (0.584)	2.888 (0.584)	6.839 (0.641)	0.056 (0.240)
3	239	0.882 (0.573)	0.005 (0.0491)	0.010 (0.0867)	9.952 (0.0141)	0.008 (0.0661)
4	63	1.928 (1.078)	3.908 (1.181)	1.008 (0.994)	4.983 (0.812)	0.743 (0.647)
5	86	3.314 (1.892)	4.383 (1.194)	2.652 (1.133)	2.347 (0.824)	1.339 (0.716)
6	34	1.026 (1.253)	0.0 (0.0)	0.013 (0.0766)	0.150 (0.450)	9.812 (0.465)
7	129	0.557 (0.347)	0.0 (0.0)	9.936 (0.235)	0.039 (0.235)	0.0 (0.0)
8	142	0.512 (0.397)	8.497 (1.729)	0.080 (0.358)	0.652 (0.999)	2.194 (1.259)
9	123	1.610 (1.217)	0.049 (0.270)	5.217 (0.939)	4.497 (0.957)	0.219 (0.592)
10	28	7.464 (4.392)	0.123 (0.318)	0.728 (0.741)	8.972 (1.034)	0.173 (0.376)
11	75	5.743 (2.154)	2.034 (0.723)	1.577 (0.834)	5.315 (0.842)	1.395 (1.074)
12	10	13.746 (1.491)	2.348 (1.641)	2.224 (0.905)	4.497 (1.579)	1.455 (0.688)
13	180	1.685 (1.047)	0.008 (0.0788)	1.283 (1.625)	4.564 (1.577)	4.121 (1.422)
14	51	2.927 (1.269)	0.558 (0.815)	0.832 (0.625)	7.970 (0.530)	0.710 (0.731)

Figure 3. Cluster Means and (Standard Deviations) for Percentage of Program Type Feature Set.

Cluster #	# of Points	CPU TIME /SUP	DRUM IO/SUP	TAPE IO/SUP	DISK IO/SUP	ER&CC/SUP
1	319	1.396 (1.055)	0.349 (0.344)	0.044 (0.167)	5.623 (2.928)	5.392 (1.915)
2	124	4.745 (1.458)	2.246 (1.920)	0.076 (0.264)	2.798 (1.901)	3.949 (0.887)
3	87	1.148 (0.561)	3.750 (1.301)	0.377 (0.787)	0.944 (0.527)	7.803 (0.575)
4	73	9.113 (2.047)	0.510 (0.652)	0.095 (0.354)	0.974 (1.145)	1.372 (0.789)
5	123	2.789 (0.562)	7.896 (2.106)	0.105 (0.418)	0.763 (0.779)	5.688 (0.594)
6	120	0.932 (0.589)	0.768 (0.501)	6.309 (8.231)	1.293 (0.761)	7.724 (1.038)
7	80	1.183 (0.485)	2.177 (0.717)	0.737 (1.001)	2.512 (0.877)	7.144 (0.446)
8	374	0.144 (0.174)	0.562 (0.485)	0.115 (0.445)	0.389 (0.547)	9.572 (0.400)
9	33	2.801 (1.001)	2.978 (1.301)	2.054 (2.061)	2.906 (0.907)	5.285 (0.816)

Figure 4. Cluster Means and (Standard Deviations) for Rate of Resource Consumption Feature Set.

Rate of Resource Consumption

Resource Consumption	1	2	3	4	5	6	7	8	9
1	4	0	4	0	0	0	0	235	0
2	2	3	16	0	9	11	18	1	13
3	65	26	33	16	8	13	27	23	4
4	14	3	0	3	0	12	1	1	0
5	0	15	2	13	87	0	0	0	0
6	2	1	5	0	2	65	8	31	0
7	2	0	1	0	0	9	8	0	6
8	211	40	15	27	8	10	4	83	0
9	7	22	6	6	7	0	9	0	3
10	1	6	1	4	0	0	0	0	6
11	10	8	4	1	2	0	5	0	1

Percentage of Programs of Each Type

Resource Consumption	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	186	2	123	6	1	2	0	1	14	1	0	0	7	0
2	0	2	1	10	22	1	0	4	4	3	18	0	4	4
3	0	21	20	28	22	5	4	7	16	9	9	0	47	27
4	0	3	0	0	0	1	1	0	17	0	0	0	13	3
5	0	0	1	0	5	0	3	106	1	0	0	0	1	0
6	0	34	20	6	7	1	1	2	30	8	1	0	8	6
7	0	1	0	3	6	0	0	0	0	1	10	3	3	1
8	0	22	84	5	1	23	117	16	38	0	0	0	87	5
9	0	2	0	4	12	1	0	4	1	4	25	1	2	3
10	0	0	0	1	7	0	0	1	1	0	2	3	2	0
11	0	0	0	0	3	0	0	1	1	1	10	3	6	2

Percentage of Programs of Each Type

Rate of Resource Consumption	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	0	15	60	5	0	13	86	2	31	5	4	0	84	9
2	0	4	5	11	12	2	25	19	7	2	10	3	15	8
3	1	7	2	16	16	1	1	12	7	1	12	1	4	5
4	0	2	7	4	6	12	14	12	3	0	3	0	9	1
5	0	1	0	5	11	0	0	93	2	0	9	0	1	1
6	0	27	5	5	4	1	2	1	35	5	8	2	20	5
7	0	12	4	5	15	1	0	0	6	5	19	2	7	4
8	85	18	156	9	4	4	0	3	28	9	3	0	39	15
9	0	0	0	2	18	0	0	0	1	0	7	2	0	3

Figure 5. Confusion Matrices for Clusterings Based on Three Feature Sets.

```
    Distance  # Points
 1.   .03319     33       *********************************
 2.   .06639     18       ******************
 3.   .09958     12       ************
 4.   .13277     11       ***********
 5.   .16597      5       *****
 6.   .19916      5       *****
 7.   .23236      2       **
 8.   .26555      2       **
 9.   .29894      1       *
10.   .33194      1       *
11.   .36513      0
12.   .39832      0
13.   .43152      1       *
14.   .46471      1       *
15.   .49790      1       *
16.   .53110      0
17.   .56429      1       *
18.   .59739      1       *
19.   .63068      0
20.   .66387      0
21.   .69707      0
22.   .73026      0
23.   .76345      0
24.   .79665      0
25.   .82984      0
26.   .86304      1       *
27.   .89623      1       *
28.   .92942      0
29.   .96262      0
30.   .99581      0
31.  1.2900       0
32.  1.06220      0
33.  1.09539      0
34.  1.12858      1       *
35.  1.16178      0
36.  1.19497      0
37.  1.22817      0
38.  1.26136      0
39.  1.29455      0
40.  1.32775      0
41.  1.36095      0
42.  1.39413      0
43.  1.42733      0
44.  1.46052      0
45.  1.49371      0
46.  1.52691      0
47.  1.56010      0
48.  1.59330      0
49.  1.62649      0
50.  1.65968      1       *
```

Figure 6. Histogram of Weighted Distances Between Cluster Points and the Cluster Mean.

Cluster Number from Run 2

Cluster Number from Run 1	1	2	3	4	5	6	7	8	9	10	11
1	243	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	71	1	0	1	1
3	2	0	0	0	0	0	0	0	213	0	0
4	0	0	0	8	0	1	0	29	0	0	0
5	0	0	107	0	0	0	0	0	10	0	0
6	0	0	0	114	0	0	0	0	0	0	0
7	0	0	0	0	0	28	1	0	0	0	0
8	5	0	0	0	384	0	0	0	9	0	0
9	0	51	0	0	0	0	0	0	9	0	0
10	0	0	0	0	0	2	0	0	0	15	1
11	0	0	0	0	0	0	0	0	0	4	27

Figure 7. Confusion Matrix for Runs with Different Seed Values.

Cluster 1 (250 points)	Cluster 1 (243 points)
0.673	0.662
0.065	0.061
2.471	2.426
1.536	1.509
0.017	0.017
0.300	0.245

Distance = 0.0059

Cluster 2 (51 points)	Cluster 9 (60 points)
3.115	3.312
7.555	7.430
9.380	9.178
9.070	9.049
0.116	0.099
8.443	8.338

Distance = 0.1804

Cluster 3 (107 points)	Cluster 5 (117 points)
4.093	4.151
3.583	3.567
3.140	4.208
6.968	7.005
0.000	0.009
1.717	1.951

Distance = 0.0645

Cluster 4 (122 points)	Cluster 6 (114 points)
1.928	1.918
1.204	1.077
5.485	5.422
4.043	4.028
5.816	5.665
3.667	3.524

Distance = 0.0637

Cluster 5 (384 points)	Cluster 8 (348 points)
4.817	4.781
1.526	1.586
3.795	3.805
2.428	2.458
0.006	0.006
4.397	4.401

Distance = 0.0024

Cluster 6 (31 points)	Cluster 7 (29 points)
2.559	2.662
6.342	6.142
9.022	9.048
8.186	8.120
10.067	9.954
7.797	7.675

Distance = 0.0822

Cluster 7 (72 points)	Cluster 2 (74 points)
3.026	3.013
4.466	4.482
7.849	7.876
7.773	7.769
6.434	6.366
6.527	6.560

Distance = 0.0069

Cluster 8 (30 points)	Cluster 4 (38 points)
4.290	3.789
4.366	4.042
6.469	6.380
3.703	3.852
3.703	5.501
4.678	6.326

Distance = 1.087

Cluster 9 (241 points)	Cluster 3 (215 points)
3.308	3.086
3.958	3.793
6.706	6.817
6.043	5.922
0.026	0.025
6.088	6.072

Distance = 0.1037

Cluster 10 (20 points)	Cluster 10 (18 points)
4.924	4.479
8.662	9.432
9.504	9.646
9.624	9.907
7.792	8.782
8.602	8.571

Distance = 1.872

Cluster 11 (29 points)	Cluster 11 (31 points)
2.951	2.874
7.086	7.056
9.427	9.421
8.551	8.515
5.302	5.223
8.641	8.764

Distance = 0.1482

Figure 8. Means and Distances Between Corresponding Clusters Obtained Using Different Seed Values.

Cluster #	# Points	Core Blocks	CPU Time	ER&CC Charges	Drum I/O	Disk I/O	Tape I/O	Distance Moved Weighted	Distance Moved Euclidean
1	592	0.651	0.066	2.608	1.540	0.016	0.328	0.0345	0.0620
2	166	2.580	4.365	7.674	7.692	6.129	6.390	0.255	0.0745
3	555	3.127	4.251	6.807	6.035	0.018	6.103	0.0597	0.0612
4	86	3.354	3.935	6.296	3.941	5.643	6.022	0.197	0.0740
5	342	3.783	4.029	4.295	6.952	0.006	2.031	0.0926	0.0780
6	278	1.588	0.970	5.310	3.714	5.374	3.306	0.259	0.0780
7	82	2.475	6.146	8.991	8.344	9.767	7.745	0.154	0.0466
8	950	4.263	1.611	3.791	2.396	0.002	4.220	0.0637	0.0714
9	122	2.947	7.438	9.280	9.139	0.086	8.392	0.113	0.0508
10	35	3.976	9.097	9.629	10.085	9.737	8.809	0.352	0.1510
11	71	2.556	6.755	9.300	8.478	5.202	8.750	0.185	0.0589

Number outside of all clusters = 8, Number of samples = 3287

Figure 9. Results of Using Training Set Clusters on the Test Data.

WORKLOAD CHARACTERIZATION AND PERFORMANCE MEASUREMENT FOR A CDC CYBER 74 COMPUTER SYSTEM

Captain Jonathan R. Bear
Captain Thomas E. Reeves

Air Force Institute of Technology
Wright-Patterson AFB OH 45433

Characterizing the workload of a computer system is directly related to interest in evaluating and predicting the performance of a computer system. Developing an accurate description of the workload is a major requirement which should precede the performance evaluation of a computer system.

This paper reports on research aimed at using existing data collected by an operating system for accounting purposes to characterize the workload of the system. The workloads presented in this paper are for a Control Data Corporation CYBER 74 computer system located at Wright-Patterson Air Force Base, Ohio.

The observed workloads obtained from the accounting data are subjected to numerical derivations using cubic spline computer subroutines in generating cumulative frequency distributions and continuous probability density functions. Tables and graphs are presented for the derived functions. Four performance measures are given for the CYBER 74 system. The measures are total jobs executed per day, CPU utilization, an economic measure called computer resource unit (CRU) and turnaround time for jobs. Histograms, tables, and graphs are used to illustrate these performance measures.

Key words: Accounting data; computer performance; computer resource unit; probability density function; statistics; turnaround; workload.

1. Introduction

Measurement and evaluation of computer performance is an area which is growing rapidly due to the economic necessity to derive maximum utilization from computer facilities [1, 25][1]. Applications of performance evaluation in the computer science field are many and diverse. For example, the evaluation of computer performance could be involved with the selection of new computer systems, with the design of applications and equipment, and with the examination and improvement of existing systems [5, 20]. The choice of a specific evaluation approach will depend upon the objectives of a particular

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

study, but, basically, the methodology for a computer performance evaluation involves the observation and measurement of a computer system model while a set of jobs, or workload, is being processed. Computer performance evaluation, then, can be divided into two focal points, the computer system and its workload.

There are various definitions of workload. The term workload as used in this paper refers to the specific computer programs which must be processed by the computer system in order to satisfy user requests [13]. Workload may be contrasted to other software programs, such as compilers and operating system alogrithms, which are considered to be part of the system overhead. The computer system is defined as the physical machine itself, all associated peripheral equipment, the workload, overhead, and the man-machine interface.

Due to the expense and time involved in measuring the workload of an existing computer system, several authors, such as Watson, Sreenivasan, and Esposito, have suggested a cost-effective approach to this problem [2, 27, 12]. Such an approach would involve using computer system accounting data, the information which is collected about resource usage in order to charge cost to users. Analysis of such data could be a valuable tool in assessing computer performance. Since the data is already available, this approach also offers the advantage of being relatively inexpensive [29].

2. CDC CYBER 74 Computer System

The CYBER 74 computer used in this study is structured as a part of a larger computer network, the Aeronautical Systems Division (ASD) computer system, which also contains a CDC 6600 computer. At selected times, the CYBER 74 can process a portion of the workload from the CDC 6600, and vice versa. In addition to the central site facilities, the ASD computer system has the capability for interactive access by 40 teletype terminals and 9 remote batch terminals.

The organizations which can directly access the CYBER 74 computer system are listed below:

(1) Air Force Institute of Technology (AFIT)
(2) Air Force Flight Dynamics Laboratory (AFFDL)
(3) Air Force Avionics Laboratory (AFAL)
(4) Aeromedical Research Laboratory (AMRL)
(5) Air Force Wright Aeronautical Laboratory (AFWAL)
(6) Air Force Human Reliability Laboratory (AFHRL).

The CYBER 74 normally operates continuously with all its remote batch and teletype terminals; however, an occasional backlog of jobs at the batch terminals has forced the Computer Center to disconnect the teletype terminals for short periods. In an attempt to provide adequate turnaround time for smaller jobs, the policy of the Computer Center is to limit the central memory field length to 120,000 octal words between the hours of 0800 to 1630 [2]. In addition, the interactive terminals and remote batch terminals are not operative from 2400 to 0800.

The CYBER 74 computer, a more recent variety of the earlier CDC 6600 series, is a multiprogrammed file-oriented machine which can be divided into three parts: a single central processor (CPU), 20 peripheral processors [6, 9, 14], and a central memory of 131K 60-bit words. The CPU has a high speed arithmetic and control unit which fetches, decodes and executes instructions sequentially. The average execution time for each instruction (major cycle) is one microsecond [10].

2.1 CYBER 74 Operating System

The Network Operating System/Batch Environment (NOS/BE) monitors the CYBER 74 computer system. A variety of modes of service is supported: batch, interactive, and graphics.

A system monitor, JANUS, which is located in one of the PPUs, schedules the total operation of the system, including CPU and PPU requests. Since there are no hardware interrupts in the system, this PPU schedules all requests for system resources. The remaining PPUs are assigned a variety of tasks in executing system and user jobs [7]. For example, INTERCOM, which controls the scheduling of all interactive terminal requests, is another system program located in a PPU.

In order to provide a multiprogramming capability, the CYBER 74 can simultaneously store 15 jobs in central memory. These jobs reside in 15 variable length partitions, or control points; the size of a control point is called its central memory field length [8].

Although a maximum of 15 active control points are available at any one time, fewer points are actually available for user jobs since a number of system support programs occupy several control points. JANUS occupies one control point, INTERCOM occupies two, graphics one, and two control points are occasionally used for communication with other computer systems.

The 15 jobs holding control points can compete for the use of the CPU, although only one job can actually be executed by the CPU at any one time. A control point may be in one of five states: (1) executing with the CPU, (2) waiting for the CPU, (3) waiting for a PPU activity to complete, (4) waiting for some operator action, or (5) rolled out [7].

The priority given a job depends upon what state of processing it is in. Initially, a job receives a priority depending upon its type. Batch is the lowest in the hierarchy with a priority of 1000. All interactive job commands and requests for execution have a 3000 priority. Graphics has the highest priority which is 6000; an exception is that operator assigned priorities may range as high as 7000. Once a job enters the CYBER 74, it passes through several stages before execution is completed; at each state, priorities are increased by an "aging factor" which is based upon the amount of time spent waiting. In this way, lower priority jobs can eventually compete on an equal basis for execution with jobs of the same type.

It is interesting to note that although INTERCOM and graphics hold only three of the ten available control points for jobs, they could use significantly more than 30 per cent of the CPU's time, since they have a higher priority than batch. As a result, interactive and graphic requests can and do affect the performance of batch jobs.

3. Data Collection System

A software program, Dayfile, which is maintained during normal processing by the Network Operating System/Batch Environment (NOS/BE) operating system collects the accounting information for the CYBER 74 computer system resources. This program is primarily used to charge users for computer resources, but the measurements obtained may also be helpful in assessing computer system performance and workload measurement. The Dayfile data are saved on magnetic tapes and stored on a regular basis as a record of each day's activities. Unlike most software monitors which are active for only short periods of time, the Dayfile program is executed continuously by the operating system [29]. Three categories for the data are:

(1) Identification data which includes the job name, name of person submitting the job, origin of the job and account number.

(2) Computer system resource requests and usage which include CPU time, PPU time, number of tape drives requested and used, I/O time, CRU's generated, and rollout time.

(3) Initiation and termination time which include time of day a job entered and exited a control point, and time a job entered the input queue.

3.1. CLARA

The Computer Load and Resource Analysis (CLARA) System is a software program which extracts specific information from the Dayfile data to a much smaller data base. These data are stored on magnetic tapes and may be processed to obtain additional information. Data from the CLARA tape are well suited for use in constructing a workload characterization and in evaluating the performance of the CDC CYBER 74 in that a wide range of parameters are available.

In order to be valid, the CLARA data must be analyzed carefully so that such problems as large numbers of system initializations and erroneous data are eliminated to avoid distorting the results. The time period investigated must be long enough to take into consideration the stability of the workload and operating system. Boeing Computer Service which supports CLARA, recommends a statistical base of about one month in order that the Dayfile data provide an adequate representation of system and job characteristics. The days and weeks of the time period under investigation may be analyzed individually once a base of sufficient time length has been obtained.

3.2. Statistical Package for the Social Sciences

The Statistical Package for the Social Sciences, a validated and tested computer program, was selected to calculate the workload statistics presented in this paper. The SPSS system of computer programs represents more than ten years of designing and programming, and has been valuable to social scientists, statisticians, and computer specialists [21]. Use of this package offers the advantage of reducing the amount of coding effort needed to process the CLARA parameters.

4. Workload Characterization

A variety of techniques may be used to evaluate the performance of a computer system. Such techniques include analytical modeling, simulation, and experiments on the existing system. For these techniques the representative[2] workload should be stable and reproducible in order to maintain consistency during performance evaluation; it should be flexible in order that the characteristics of the representative workload may be varied [26]. This is necessary since there can be a wide variation in actual workloads [3]. In addition, it may be necessary to represent the wide variations in the actual workload by separating it into different classes based upon specific workload characteristics. A potential pitfall, however, in simulating the actual workload is that the representative workload may not adequately represent the actual workload; it may not be possible to design certain characteristics of the actual workload into the representative workload [30].

An alternate method that can be used in separating the workload into different classes is to use the clustering techniques described by Agrawala and Mohr [2]. This approach generates joint probability density functions. Thus, workload parameters such as CPU, I/O, and memory used would give a joint probability density function where the parameters are considered simultaneously.

In this paper several workload classes are characterized by the type and amount of resources which are used by a computer system to satisfy requests from users' programs [28]. For example, some workload characteristics are central processor unit (CPU) time, input/output (I/O) and memory requirements. The values of these job parameters can be used to quantify the representative workload by analyzing the distribution of demands upon individual system resources.

4.1. Workload Characteristics

The methodology for characterizing the CYBER 74 workload uses statistical techniques to measure and categorize the accounting data by type and amount of resources. A two step approach is used in this paper.

First, each day of accounting data is characterized by resource usage; a number of variables are selected from the accounting data as representative of the actual workload. These variables[3] reflecting the suggestions of several authors, appear appropriate for a workload characterization [12, 27]. The parameters in table 1 were used to characterize the workload.

Table 1. Parameters Used in Workload Characterization (All measurements of time are in seconds)

Name	Description
CPTIME	The central processor time needed to process a job.
PPTIME	The peripheral processor time needed to process a job.
TIMEIO	The total input/output time needed to process a job.
TOTCOST	The total job cost in terms of Computer Resource Units (CRU's).
KWS	The memory usage in kilo-word seconds.
CPOT	The total control point occupancy time which is the time a job enters a control point until it leaves a control point for the last time.
CMLOC	The number of central memory locations occupied by all jobs including this job at the time this job left a control point.
CPLOC	The number of control points occupied by all jobs including this job at the time this job left a control point.

[2]A representative workload should faithfully represent the real workload.

[3]The variables used depend on the objective of the study being done.

ROLLOUT	The total time that a batch job was rolled out for processing magnetic tapes.
LATIME	The total time between this batch job and the last batch job to arrive into the Input Queue.
INQTIME	The time a batch job spends in the Input Queue.
TAPEREQ/ TAPEUSED	The number of tapes requested/ used per job.

Second, the variations in the workload among the different days are categorized by separating the days of data into two classes based upon CPU time and memory usage. Although other classes could be formed, this paper focuses upon these two classes as an initial effort to charcterize the CYBER 74 workload.

Graphical plots are used to consolidate data in an easily readable fashion. Frequency histograms are used to illustrate the usage pattern of a variable or its frequency of occurrence over a range of values. In addition, a time-series pattern is plotted in order to display variable usage through time. The time series plots are used as an indication of an increase or decrease in the components of the workload.

5. Performance Measurements

Measuring the effect of workload variations upon the performance of a computer system is a complex task because there is no generally accepted measure of performance. The main difficulty can be traced to the absence of acceptable criteria upon which the performance can be optimized [19]. One approach to selecting an appropriate measure of performance involves investigating the mission of operational objectives of an organization [4]. An effort should be made to select a performance measure consistent with those objectives. However, a problem exists when goals of the organization are broad and not specifically stated: finding a performance measurement which is universally acceptable may be difficult or impossible.

The overall mission of the Aerospace Systems Division (ASD) computer center is stated below.

(1) To provide centralized computer support to the AFSC/AFIT organizations.

(2) To develop improved computer techniques and procedures.

(3) To program and operate the general purpose digital, analog, and hybrid computers.

(4) To act as the ASD focal point for computer support, acquisition, and utilization.

The ASD mission as stated above is relatively broad, and it would be exceedingly difficult to select a single adequate measure of performance. Therefore, turnaround time and an economic measure called Computer Resource Unit were selected as performance measures for CYBER 74.

5.1. Turnaround

Turnaround has been defined as the time from when a batch job enters the computer system to the time when output from the job is received [17, 25]. This measure includes the man-machine interaction, which is outside the scope of this paper. Therefore, in this paper, turnaround time will represent the machine turnaround time, which is the time between a batch job entry into the input queue and the last time the job leaves a control point.

5.2. Economic Performance

A measure of economic performance can be derived from the Computer Resource Utilization (CRU) algorithm. The ASD Computer Center uses a measure of resource usage, the CRU, to charge usage since it is a weighted measure of CPU time, I/O time, and central memory occupancy time. The total cost of a job is calculated by charging $0.06 for each CRU generated by a user's program. A more general description of economic and accounting measures is discussed in Reference 16.

The Computer Center algorithm for computing CRU's is described in an ASD letter dated 14 June 1976, "Charges for Use of the ASD Computer Center's CDC 6600 and CYBER 74 Computers." The algorithm is as follows:

$$\text{Number of CRUs} = a_1\text{CPTIME} + a_2\ \text{TIMEIO} + a_3 \sum_{i=1}^{n} (\text{CPTIME}_i + \text{TIMEIO}_i)\text{CM}_i$$

where CPTIME	=	the total CPU time needed to process the job in seconds.
TIMEIO	=	the total I/O time used by the job to include total tape channel and disk access time in seconds.
$CPTIME_i$	=	the CPU time during the ith stage of processing.
$TIMEIO_i$	=	I/O time during the ith stage of processing.
CM_i	=	number of central memory locations (octal) required during the ith stage of processing.
n	=	number of stages of processing.
a_1	=	CPTIME cost percentage ratio per second = .3598/sec.
a_2	=	TIMEIO cost percentage ratio per second = .5418/sec.
a_3	=	CM percentage cost ratio per 100,000 (octal)-word second = .3936/100K-word sec.

6. CYBER Workload Trend Analysis

The available CLARA accounting data encompassed more than six months of background information which was recorded on magnetic tape. This historical data was the base for an investigation for workload and performance trends on both a long term (days) and short term (hours) basis. The results of the trend[4] data were then used to select a time period and specific days for further investigation.

Four performance measures were examined for a six month investigation period which included normal weekdays only. These performance measures, total jobs executed per day, CPU utilization per day, machine turnaround time per day, and average CRU's per hour for each day were selected because they were representative of the available accounting data. Weekdays were selected since a majority of the complaints about slow response time and turnaround time occurred then.

The following schematic diagram shows one way of viewing the relationships between the four performance measures that were examined.

$$\begin{tikzcd}
& \text{COST (CRU)} & \\
\text{LOAD (no. of Jobs)} \arrow[r] & \text{COMPUTER FACILITY} \arrow[u] \arrow[r] \arrow[d] & \text{SERVICE (Turnaround)} \\
& \text{UTILIZATION (CPU)} &
\end{tikzcd}$$

The observations below were made from figures 1-4; table 2 lists the days which were used in the trend analysis.

(1) The total jobs executed per day (fig 1) ranged from 850 to 1700. Since the INTERCOM jobs were relatively constant, the total number appeared to be a function of the number of batch jobs.

(2) The CPU utilization (fig 2) showed an overall increasing trend which began on the 37th day (23 June 1976) of the investigation and continued until the 95th day (15 September 1976). Utilization ranged from 41 per cent to 88 per cent.

(3) The machine turnaround time, the time from job entry into the input queue to the final exit from the control point (fig 3), varied from a daily average of 0.25 (9 July) to 1.70 hours per job (7 October).

(4) The average CRU's per active hour (fig 4) ranged from zero to 1200 CRU's per hour for INTERCOM jobs. Batch jobs, more variable in production, ranged from 2600 to 5500 CRU's per hour. In order to normalize the measure, CRU's produced per hour were measured rather than CRU's produced per day.

Further, trend analysis over longer periods of time may also confirm weekly, monthly, and quarterly cycles; the trend data appears to be periodic. For example, the workload upon the computer system appears to be heavier at the beginning of a week. On a quarterly basis, the number of jobs tends to

[4]Trend is used here as a tendency in the data to increase or decrease over time and not as a functional trend or as statistical trend analysis.

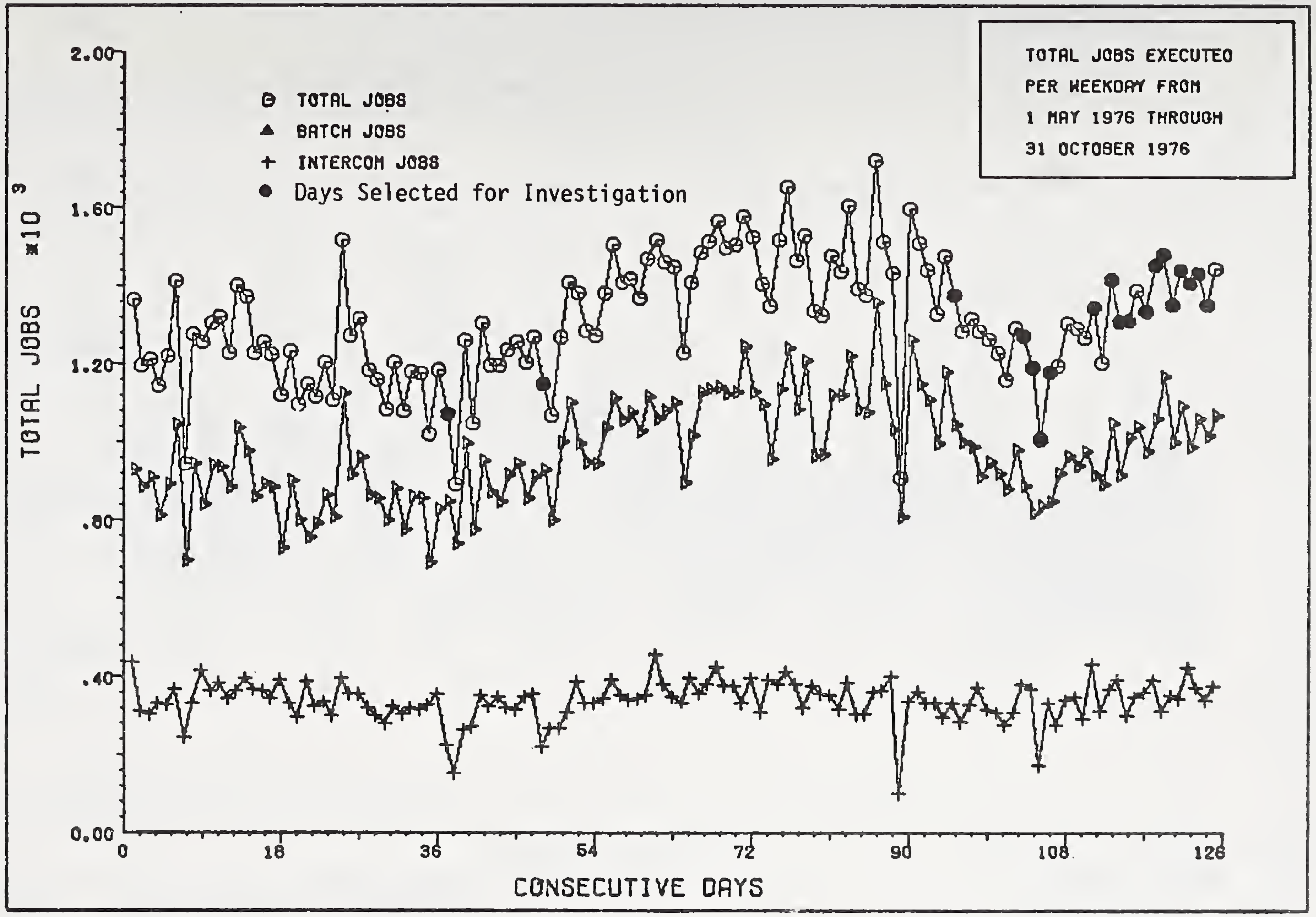


Figure 1. Trend analysis for total jobs per day.

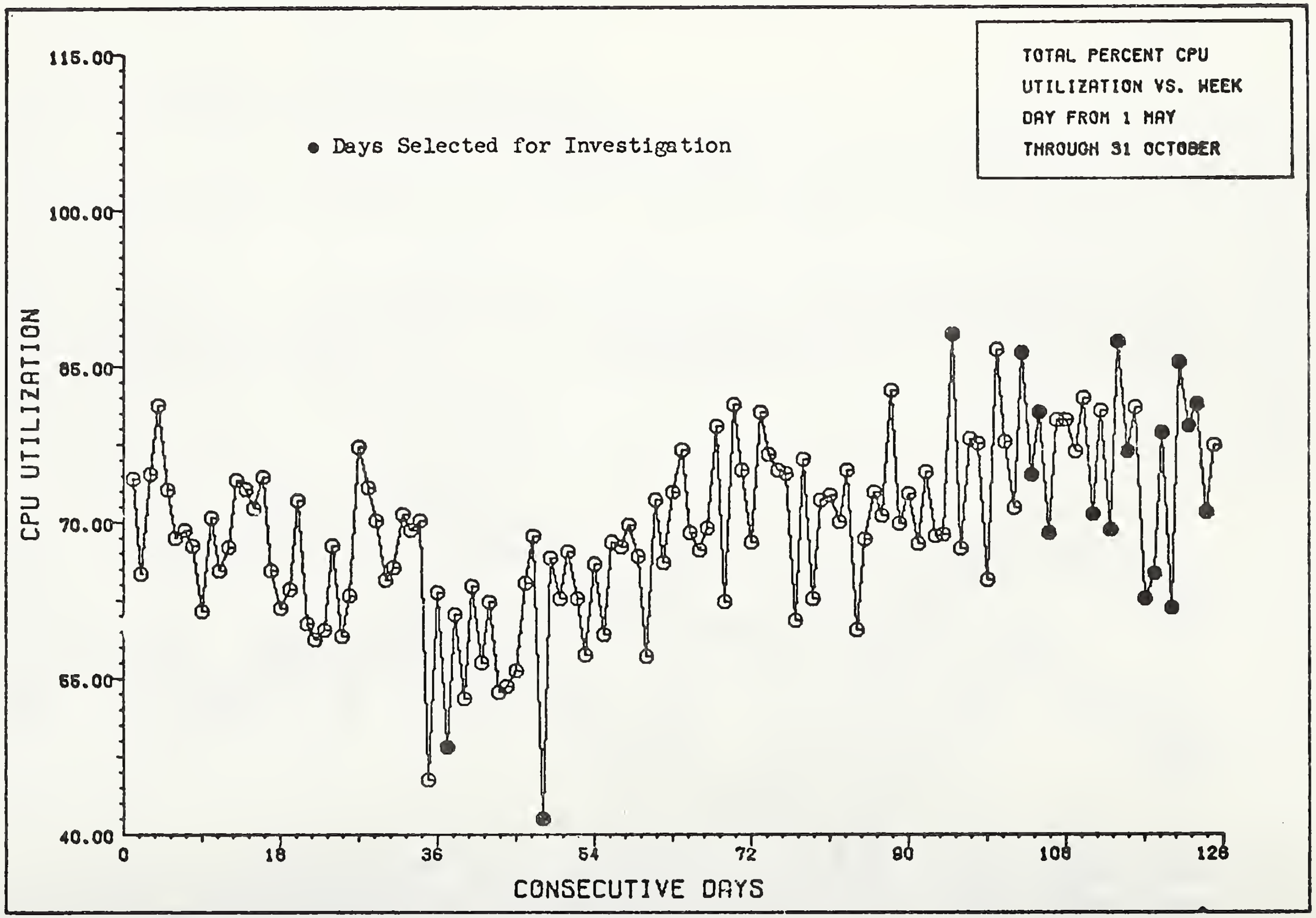


Figure 2. Trend analysis for CPU utilization.

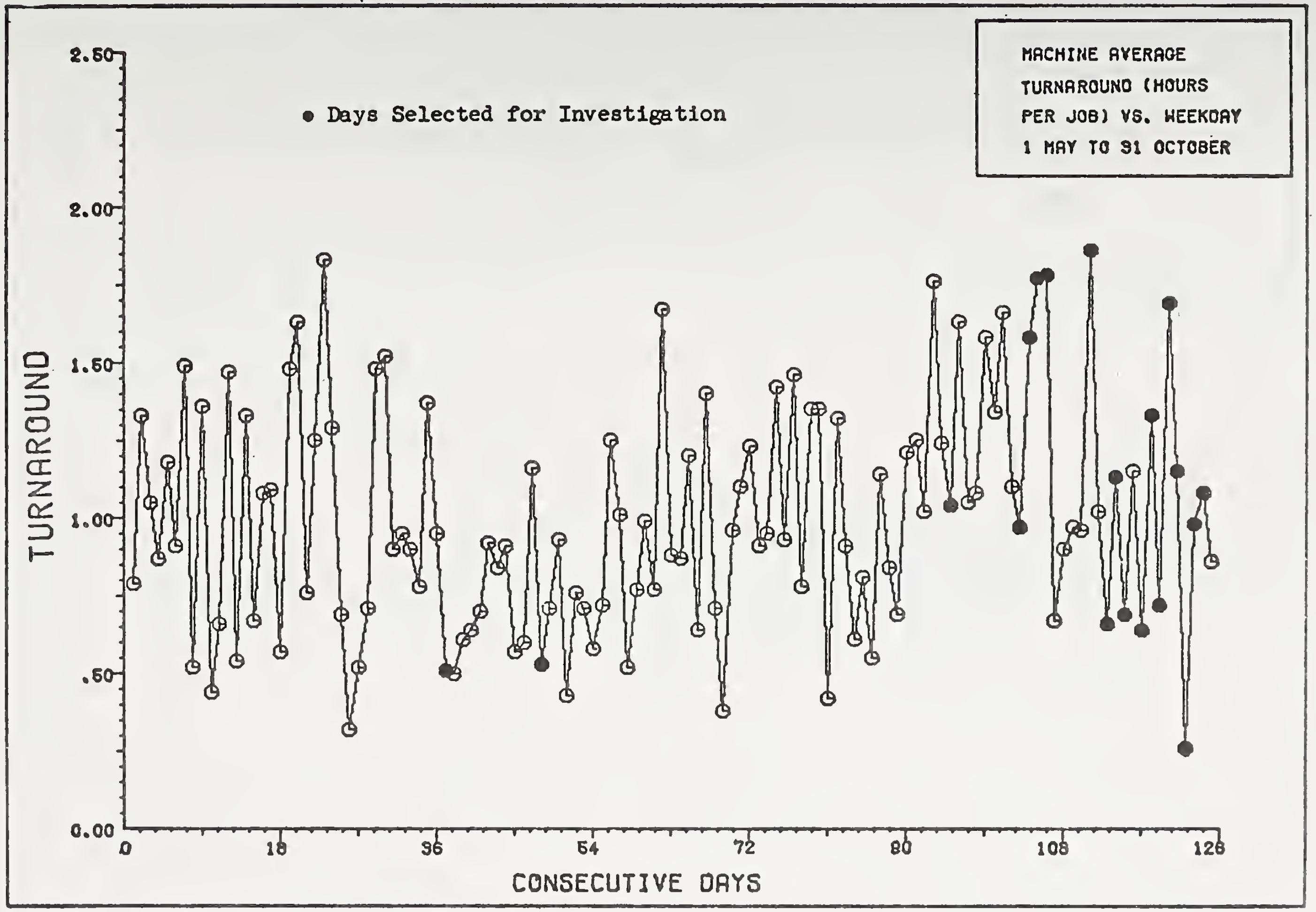


Figure 3. Trend analysis for turnaround.

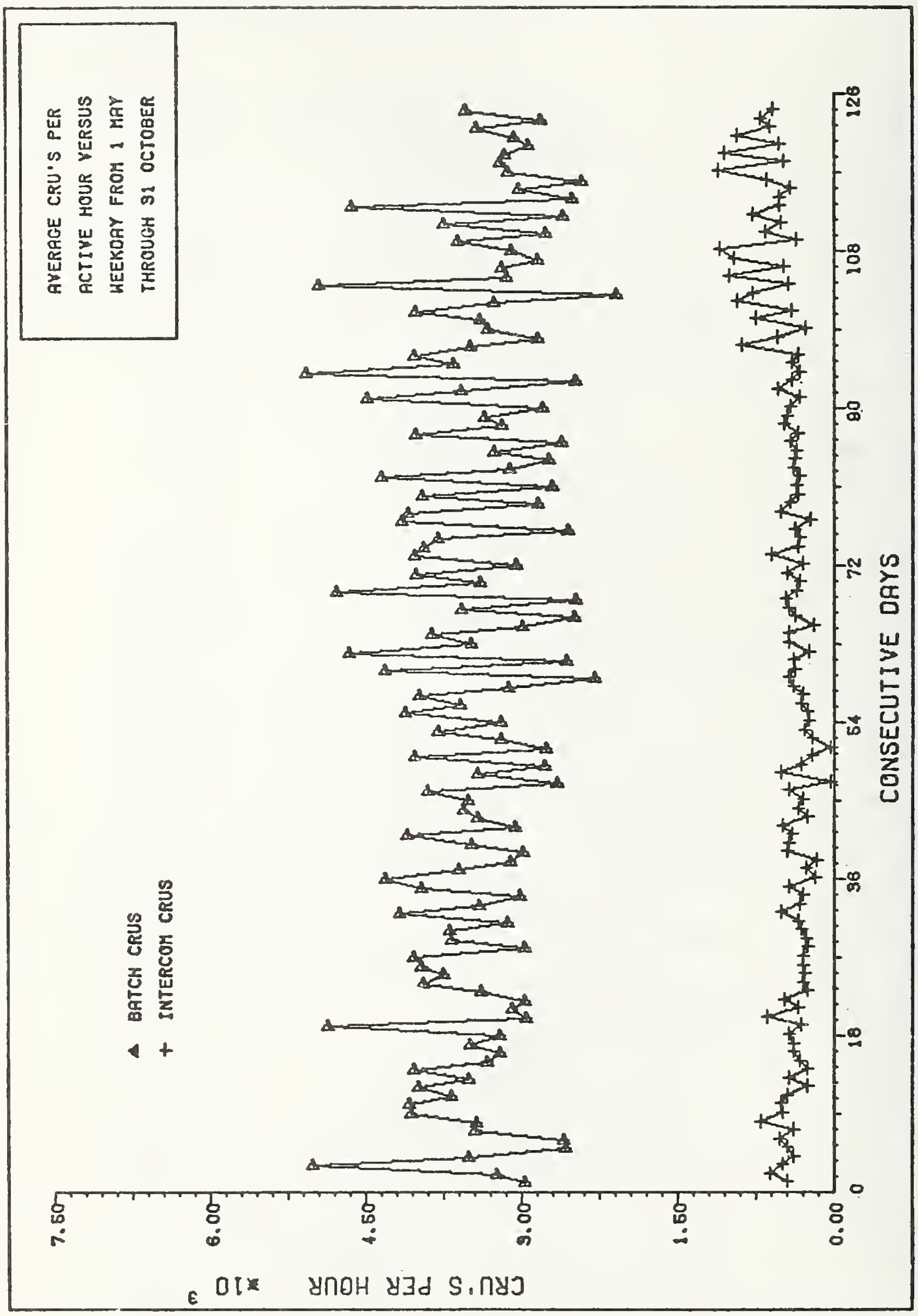


Figure 4. Trend analysis for CRU's per hour.

Table 2. Trend Analysis Days and Dates

Days	Dates	Days	Dates
1-5	1 to 5 May	64-68	2 to 6 Aug
6-10	10 to 14	69-73	9 to 13
11-15	17 to 21	74-78	16 to 20
16-20	24 to 28	79-83	23 to 27
21-24	1 to 4 Jun	84-85	30 to 31
25-29	7 to 11	86-88	1 to 3 Sep
30-34	14 to 18	89-92	7 to 10
35-39	21 to 25	93-97	13 to 17
40-42	28 to 30	98-102	20 to 24
43-44	1 to 2 Jul	103-106	27 to 30
45-48	6 to 9	107	1 Oct
49-53	12 to 16	108-112	4 to 8
54-58	19 to 23	113-116	12 to 15
59-63	26 to 30	117-121	18 to 22
		122-126	25 to 29

Table 3. Days Scheduled for Investigation

Day	Date	Day	Date
1	27 September	10	18 October
2	28 September	11	19 October
3	29 September	12	20 October
4	30 September	13	21 October
5	7 October	14	22 October*
6	11 October	15	25 October
7	12 October	16	26 October
8	13 October	17	27 October
9	14 October	18	28 October*

*Excluded from further investigation as non-representative.

increase and decrease as the (academic) quarter begins and ends at the Air Force Institute of Technology (fig 1).

Several days were selected for further investigation as seen in Table 3. The number of days was limited to 18 due to the amount of data to be processed. The days were selected to be representative of average performance[5] as well as extremes in performance. In order to minimize the effects of changes in the computer system, such as changes in the operating procedures and computer center personnel the days were generally grouped within a 30-day period. The investigation period ranged from the 103rd day (27 September 1976) through the 125th day (28 October 1976).

It should be noted that the variation in the INTERCOM CRU's generated from the 96th through the 125th day in figure 4 was caused by a change in the accounting method for this parameter rather than by a change in the performance measurement of the CRU's per hour.

In order to eliminate nonrepresentative data, the criterion was established that a day would not be included in the investigation if malfunctions of the central computer site resulted in idle time for that day which comprised four per cent or greater of total active time; none of the days were excluded due to the extremely low number of jobs recorded on the CLARA tape. The remaining 16 days were used as a data base for the investigation.

During the period of exploratory investigation, the entire 24-hour day was studied; however, it was necessary to narrow this time period because the maintenance of the computer system, which is periodically scheduled at midnight for several hours, was biasing the results obtained. For example, the number of CRU's generated per day varied widely depending upon the number of active hours. A second approach was attempted which focused upon the 0800 to 2400 time period. Several problems were encountered, and it was decided to reduce the period of study to a more specific workload, the hours from 0800 to 1600, for the following reasons:

(1) The largest number of users are active on the CYBER 74 during normal working hours from 0800 to 1600.

(2) Since the INTERCOM activity is relatively constant during the day and tends to be reduced in the evening, the daytime period is more appropriate for measurement of the CYBER 74 multiprogramming mode, including batch and INTERCOM.

(3) In an attempt to give the smaller user a faster response and turnaround time during the day, the ASD central site withholds until 1600 each day, those jobs requiring a large number of system resources.

(4) The majority of complaints from users occur during the 0800 to 1600 time period, and relate to slow turnaround and response times.

Unless stated otherwise, the remaining discussion of the results will refer to the 0800 to 1600 time period for the 16 investigation days.

7. Workload Characteristics

For the investigation days, the accounting data was grouped on low, medium, and high days. The workload characterizations were separated into two job classes based on CPU time and memory used. The results of this separation are summarized in tables 4 and 5.

Figures 5 and 6 illustrate sample probability density functions for the two job classes. A general observation for the two job classes is that the workload for the CYBER 74 generally is distributed bimodally.[6] One group, which comprises approximately 80 per cent of the jobs has small KWS requested or small CPU time. The other group is composed of about 20 per cent of the jobs and consists of larger jobs which consume large amounts of CPU time or memory.

The memory term, kilo-word seconds (KWS), is calculated from the central memory term (CM) and the constant (a_3) given in the costin

[5]Determining average performance is not a simple problem. It is really a multivariate problem. Average performance here was taken from observations of the trend data.

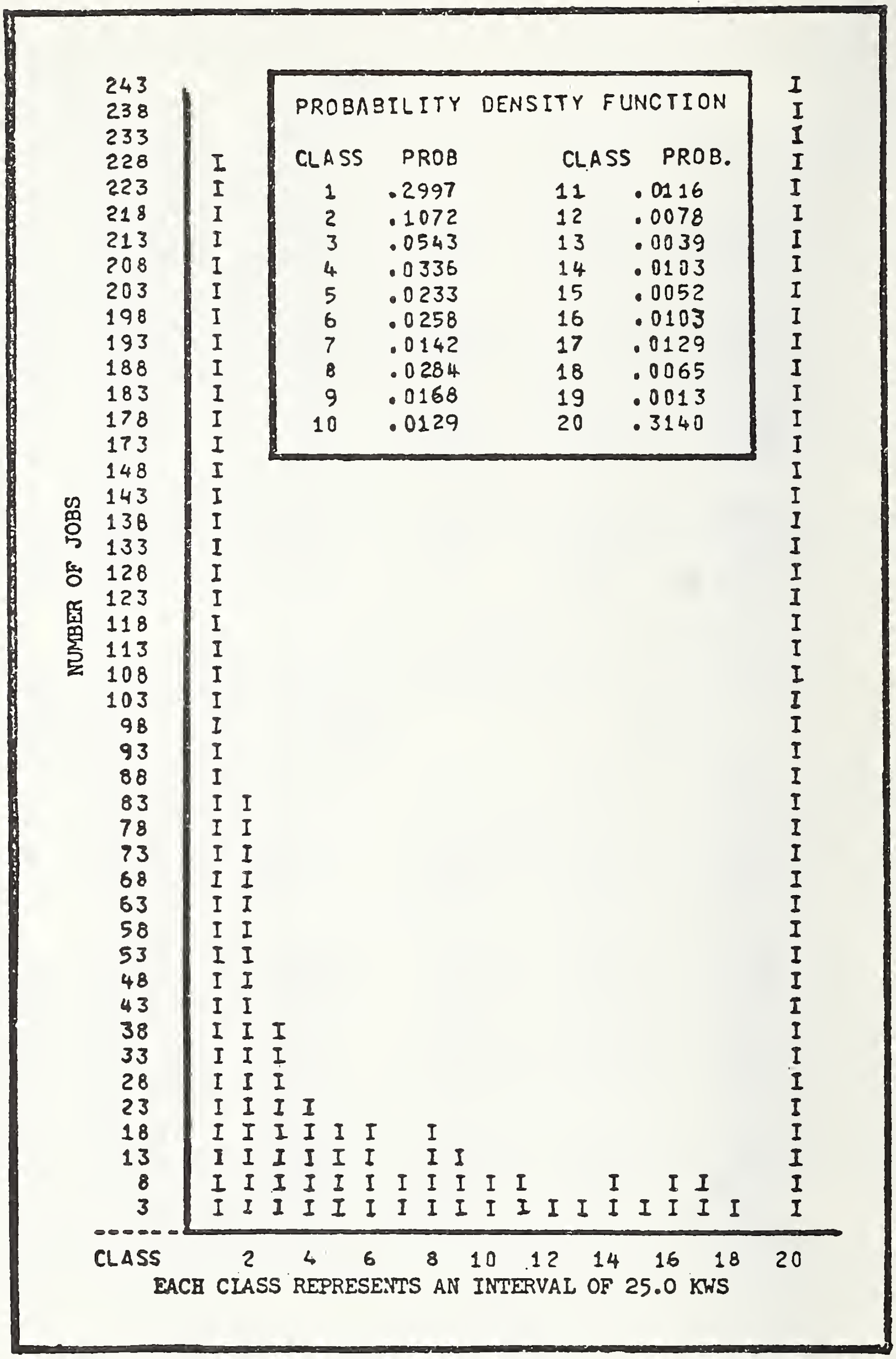
[6]The intervals chosen for the probability density function may have caused the appearance of a bi-modal distribution.

Table 4. KWS Workload Characterization

WORKLOAD PARAMETERS (AVG VALUE PER JOB)	Low KWS Days		Medium KWS Days		High KWS Days	
	25 OCT	14 OCT	19 OCT	27 OCT	29 SEP	13 OCT
CPTIME	24.28	22.60	23.12	23.26	35.17	34.10
PPTIME	78.36	62.36	34.53	68.21	95.72	73.83
TIMEIO	13.95	15.49	22.95	23.53	1397.72	17.13
TOTCOST	24.08	29.96	31.43	33.40	43.55	35.81
KWS	648.56	701.93	888.88	1022.81	1397.78	1187.30
CPOT	1045.16	777.24	863.50	1234.32	1158.486	1176.29
CMLOC	191314	124030	129792	76375	202640	129083
CPLOC	90.32	94.85	97.25	95.27	92.19	91.95
ROLLOUT	22.13	17.37	27.50	16.83	61.29	41.82
IATIME	40.06	44.50	49.52	42.22	49.92	41.49
INQTIME	129.74	576.55	1310.99	645.79	799.85	345.60
TAPEREQ	.052	.062	.048	.050	.078	.074
TAPEUSED	.052	.058	.047	.050	.076	.071

Table 5. CPU Workload Characterization

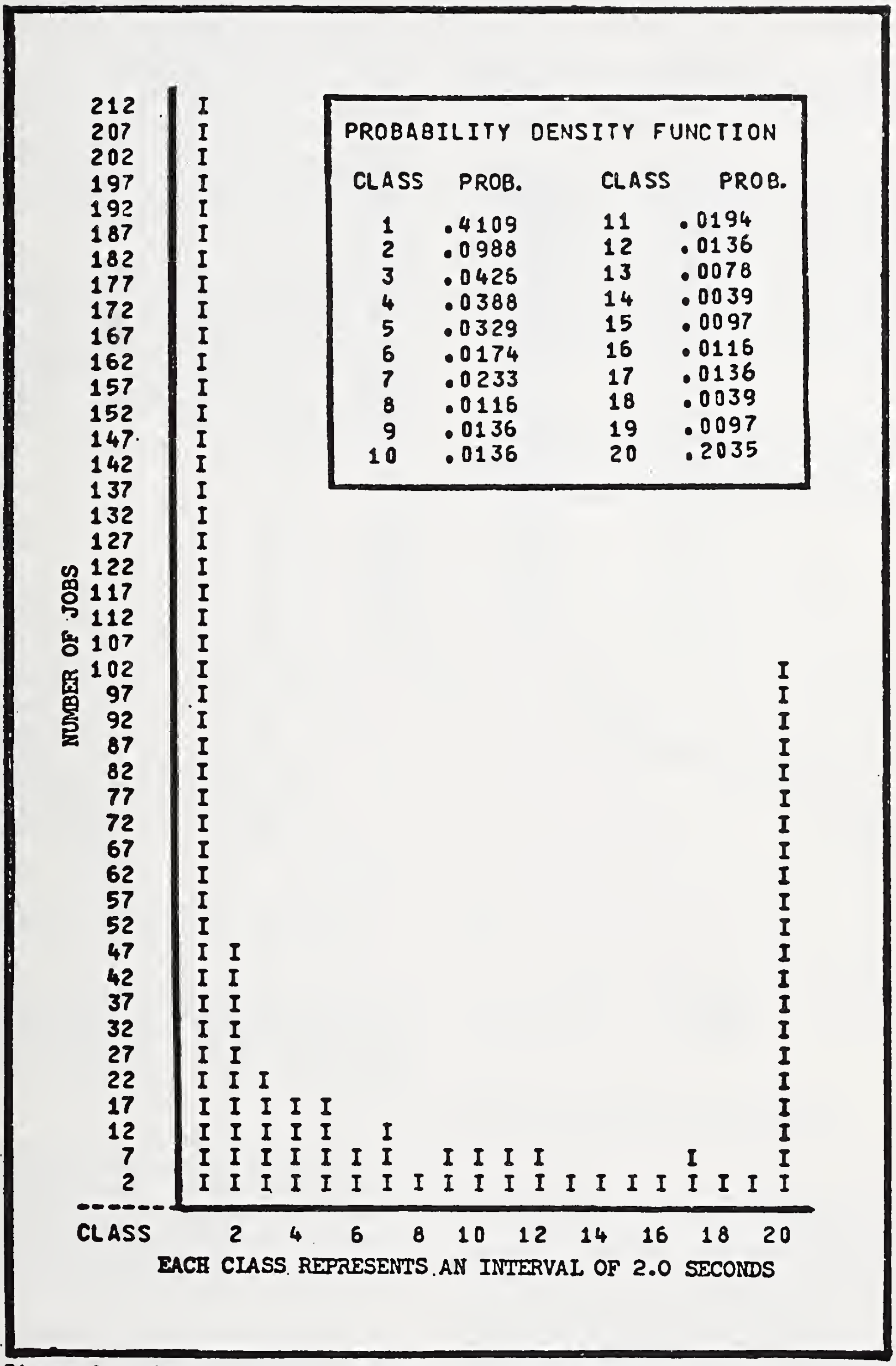
WORKLOAD PARAMETERS (AVG VALUE PER JOB)	Low CPU Days		Medium CPU Days		High CPU Days	
	11 OCT	20 OCT	7 OCT	21 OCT	29 SEP	27 SEP
CPTIME	21.67	22.13	29.31	29.68	35.17	35.53
PPTIME	38.36	48.45	98.03	72.02	95.72	56.62
TIMEIO	17.98	24.02	881.05	16.34	1397.72	808.17
TOTCOST	26.75	31.00	31.46	29.44	43.55	27.66
KWS	767.43	834.04	881.25	826.16	1397.78	816.16
CPOT	903.66	1080.86	804.03	964.20	1158.486	760.25
CMLOC	142858	187128	149777	146110	202640	166231
CPLOC	88.20	95.44	61.21	96.42	92.19	67.57
ROLLOUT	21.00	22.33	22.46	16.17	61.29	16.19
IATIME	47.28	46.04	65.35	50.23	49.92	44.88
INQTIME	598.58	1651.24	808.29	831.63	799.85	322.38
TAPEREQ	.060	.057	.059	.056	.078	.073
TAPEUSED	.053	.053	.055	.053	.076	.067



PROBABILITY DENSITY FUNCTION

CLASS	PROB	CLASS	PROB.
1	.2997	11	.0116
2	.1072	12	.0078
3	.0543	13	.0039
4	.0336	14	.0103
5	.0233	15	.0052
6	.0258	16	.0103
7	.0142	17	.0129
8	.0284	18	.0065
9	.0168	19	.0013
10	.0129	20	.3140

Figure 5. Distribution of KWS for 13 October.



PROBABILITY DENSITY FUNCTION

CLASS	PROB.	CLASS	PROB.
1	.4109	11	.0194
2	.0988	12	.0136
3	.0426	13	.0078
4	.0388	14	.0039
5	.0329	15	.0097
6	.0174	16	.0116
7	.0233	17	.0136
8	.0116	18	.0039
9	.0136	19	.0097
10	.0136	20	.2035

Figure 6. Distribution of CPU time for 29 September.

algorithm in section 5.5. It is a good representation of memory used by job requests.

8. Obtaining Continuous Probability Functions

A technique for taking a series of observations and obtaining probability density functions from the observations is presented.

The customary way of organizing data derived from observations is to display them as a frequency distribution, which shows the number of times the variable falls in different intervals. Table 6 on next page is an example of this technique. The data used in this section is for 13 October 1976, which was the day having the highest CRU's generated.

A series of straight line segments drawn through the points in the frequency table (fig 7) can be taken as an approximation to the cumulative distribution function.

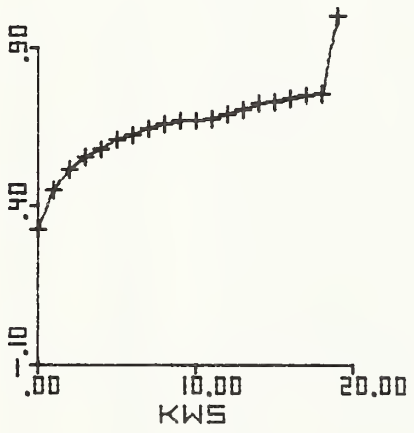


Figure 7. Cumulative distribution function

The respective probability function can be derived by plotting the slope of the cumulative distribution function. The computer subroutine package called International Mathematical and Statistical Libraries (IMSL) was used to derive the probability density functions. The subroutine ICSSCU from the IMSL package will give a smooth cubic spline function along a given set of data points.

A smoother interpolating function can usually be produced by mechanical means such as a French curve or by forcing a flexible elastic bar to pass through the desired points. The mathematical analog of this flexible elastic bar is the cubic spline function. A cubic spline with knots $x_0, x_1, \ldots x_n$ is a piecewise cubic function with values $f(x_i)$ specified at the knots and such that first and second derivatives of the spline are continuous on $[x_0, x_n]$. The natural cubic spline is a cubic spline with the additional conditions that the second derivatives vanish at x_0 and x_n.

The subroutine DCSEVU also from the IMSL package will evaluate the first derivative of the cubic spline function. Using the values of table 6 as the given set of data points, the subroutines ICSSCU and DCSEVU were called in succession. The plot of the derived probability density function is given in figure 8.

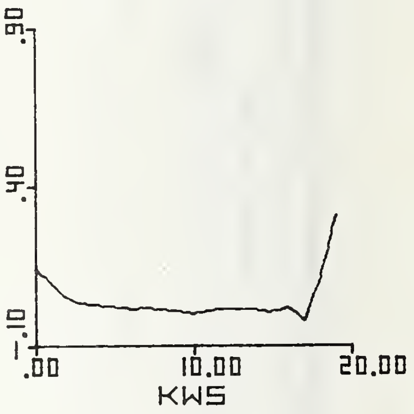


Figure 8. Probability density function

The first observation that can be made from the two graphs is the occurrence of a hump in the data at the end point interval, which contains all occurrences greater than 20 KWS.

The first analysis consisted of making tests to see if the large requests for KWS occurred during certain time periods. The results of the tests indicated that large requests were not made during any specific time period other than those occurring randomly during the 0800-1600 time period.

The next analysis consisted of breaking the information according to the amount of KWS required into two sections. Table 7 gives the intervals for the frequency distribution when divided into two sections. Figure 9 shows the cumulative distribution graphs.

Table 6. Frequency distribution for KWS of memory used

MEMORY (KWS)	NUMBER OF JOBS	RELATIVE FREQUENCY	PROBABILITY DENSITY	CUMULATIVE DISTRIBUTION
0	149	.3253	.3253	.3253
1	57	.1245	.1245	.4498
2	30	.0655	.0655	.5153
3	18	.0393	.0393	.5546
4	12	.0262	.0262	.5803
5	14	.0306	.0306	.6114
6	6	.0131	.0131	.6245
7	10	.0213	.0213	.6463
8	7	.0153	.0153	.6616
9	3	.0066	.0066	.6681
10	1	.0022	.0022	.6703
11	2	.0044	.0044	.6747
12	7	.0153	.0153	.6900
13	7	.0153	.0153	.7052
14	8	.0175	.0175	.7227
15	3	.0066	.0066	.7293
16	5	.0109	.0109	.7402
17	5	.0109	.0109	.7511
18	2	.0044	.0044	.7555
19	112	.2445	.2445	1.0000

Table 7. Frequency distribution for KWS of memory used when separated into two sections

MEMORY (KWS)	NUMBER OF JOBS	RELATIVE FREQUENCY	PROBABILITY DENSITY	CUMULATIVE DISTRIBUTION
0	149	.4269	.4269	.4269
1	57	.1633	.1633	.5903
2	30	.0860	.0860	.6762
3	18	.0516	.0516	.7273
4	12	.0344	.0344	.7622
5	14	.0401	.0401	.8023
6	6	.0172	.0172	.8195
7	10	.0287	.0287	.8481
8	7	.0201	.0201	.8682
9	3	.0036	.0036	.8768
10	1	.0029	.0029	.8797
11	2	.0057	.0057	.8854
12	7	.0201	.0201	.9054
13	7	.0201	.0201	.9255
14	8	.0229	.0229	.9434
15	3	.0036	.0036	.9570
16	5	.0143	.0143	.9713
17	5	.0143	.0143	.9857
18	2	.0057	.0057	.9914
19	3	.0036	.0036	1.0000
20	34	.3119	.3119	.3119
40	27	.2477	.2477	.5596
60	18	.1651	.1651	.7248
80	3	.0275	.0275	.7523
100	5	.0459	.0459	.7982
120	8	.0734	.0734	.8716
140	2	.0183	.0183	.8899
160	3	.0275	.0275	.9174
180	1	.0092	.0092	.9266
200	2	.0183	.0183	.9450
220	1	.0092	.0092	.9541
240	2	.0183	.0183	.9725
260	0	.0000	.0000	.9725
280	1	.0092	.0092	.9817
300	1	.0092	.0092	.9908
320	0	.0000	.0000	.9908
340	0	.0000	.0000	.9908
360	0	.0000	.0000	.9908
380	0	.0000	.0000	.9908
400	1	.0092	.0092	1.0000

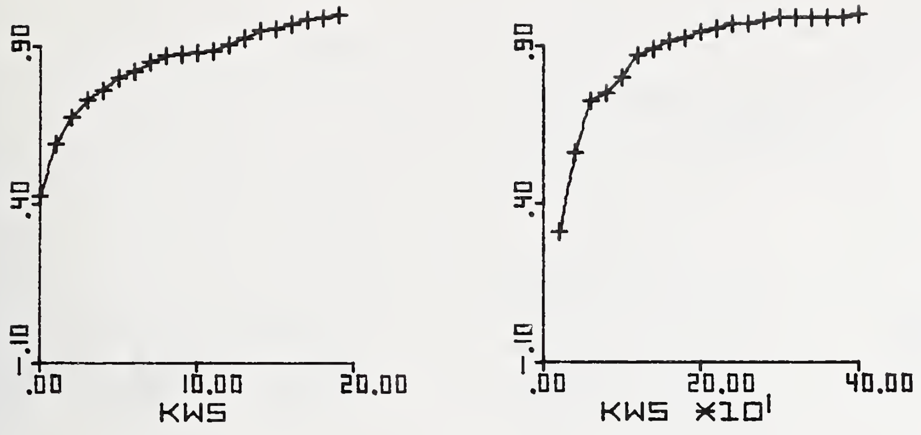


Figure 9. Cumulative distribution functions separated

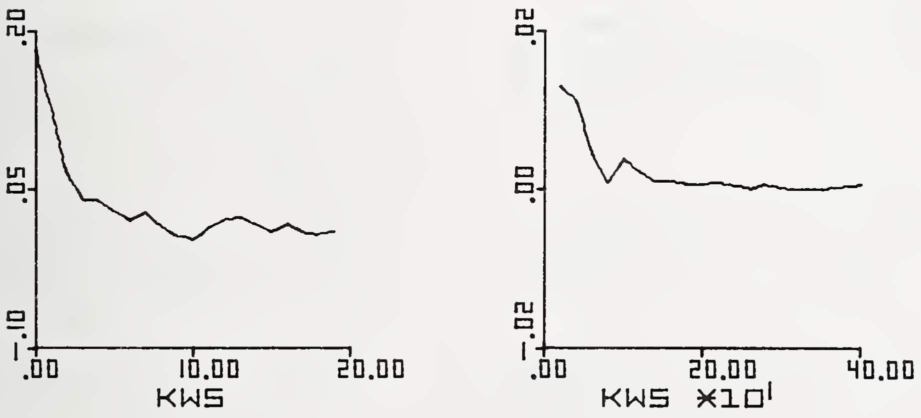
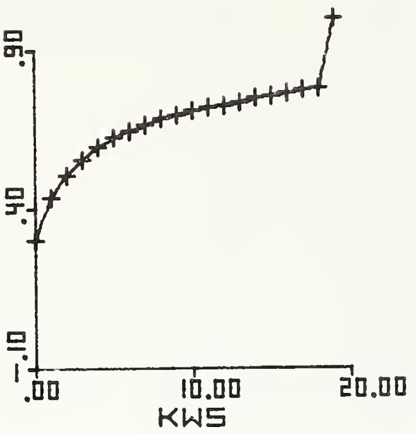


Figure 10. Probability density functions separated



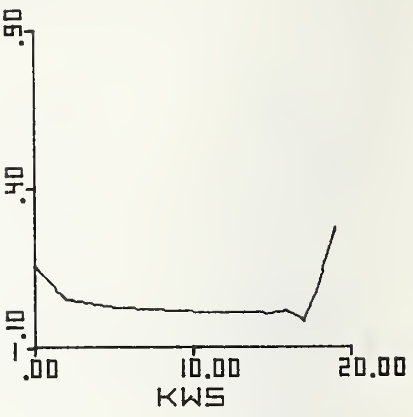
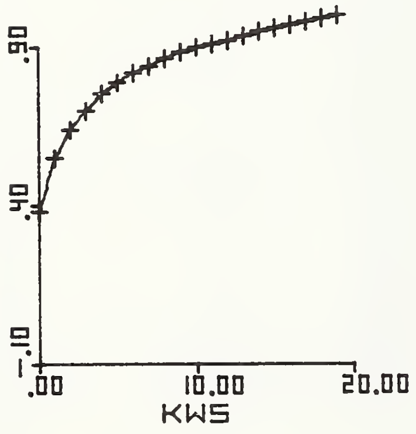


Figure 11. Cumulative distribution and probability density functions for total period not separated into two sections



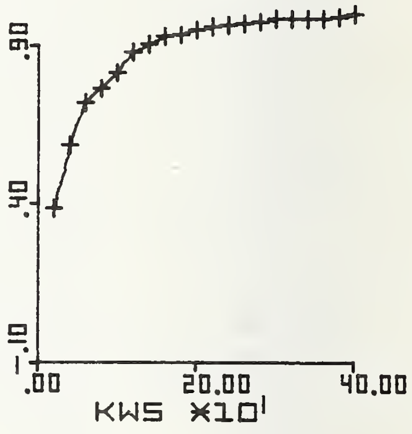
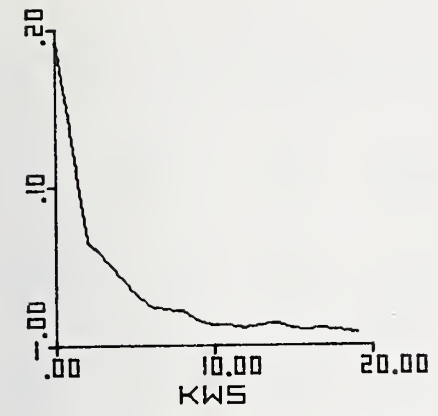


Figure 12. Cumulative distribution functions separated into two sections



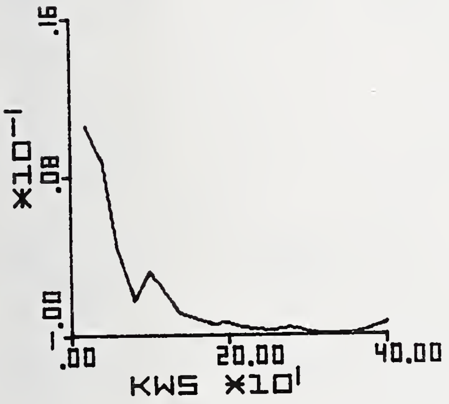


Figure 13. Probability density functions separated into two sections

The respective probability density functions are given in figure 10.

Often, there is interest in the general trend of a workload parameter over the total period under investigation rather than just for a specific day. The same technique and interval sizes used for figures 7-10 were applied to the total 16-day period under investigation. Figures 11-13 show the results for the total 16-day period. The graphs (fig 12 and fig 13) for the 16-day period illustrate there can be some inaccuracies if one specific day's information (fig 9 and fig 10) is used to model the total period. Graphs for some other specific days were plotted. Those graphs were more inaccurate than the day given in figures 9-10 as a representation of the total period.

These probability density functions could now be used in a simulation model or mathematical model that is evaluating performance of the CYBER 74 when KWS of memory used is a parameter being considered. If a particular mathematical function was being used as the probability density function, it should now be plotted and compared to the functions generated from the observed data. This would assure that the assumed mathematical function is a good approximation to what is actually occurring in the real model.

9. Performance Measurement

An analysis of the turnaround time and CRU generation performance measures was conducted for each day in the time interval 0800-1600.

9.1. Turnaround

The first performance measure is based upon the average turnaround per batch job for the eight-hour period from 0800 to 1600. The range of values is shown in table 8.

The average machine turnaround time (ITURN) ranged from 1189.74 seconds (19.8 minutes) to 3176.835 seconds (53.0 minutes) for jobs which were basically the same size in terms of CPTIME, PPTIME, TIMEIO, and KWS. For all of the investigation days, the average turnaround time for a batch job was 28.3 minutes.

The slower turnaround could be a result of the computer system being saturated with jobs. As an example, on slower turnaround days, the time spent by a job in the input queue waiting for processing and the number of control points filled for a job were larger than for jobs on days with faster turnaround. Faster turnaround appeared to be proportional to faster throughput time and inversely proportional to CRU's per hour (fig 14). Future investigation into INTERCOM turnaround time and response time may further explain these relationships.

The average turnaround per day for all the days investigated was 28.3 minutes.

Table 8. Turnaround Summary

WORKLOAD PARAMETERS (AVERAGE VALUES)	Slow Turnaround Days		Medium Turnaround Days		Fast Turnaround Days	
	30 SEP	20 OCT	28 SEP	19 OCT	21 OCT	14 OCT
CPTIME	29.15	22.13	34.43	23.12	29.60	22.60
PPTIME	83.78	48.45	66.50	34.53	72.02	62.36
TIMEIO	1056.61	24.02	839.96	22.95	16.34	15.49
TOTCOST	36.52	31.00	32.49	31.43	29.44	24.96
KWS	1056.70	834.04	839.98	888.88	826.16	701.93
CPOT	1103.85	1080.86	1142.59	863.50	964.20	777.24
CMLOC	180053	187128	51228	129792	146110	124030
CPLOC	91.83	95.44	94.86	97.25	96.42	94.85
ROLLOUT	42.15	22.33	41.63	27.50	16.17	17.37
IATIME	101.36	46.04	69.83	49.52	50.23	44.50
INQTIME	1533.99	1651.24	1218.00	1310.99	931.63	576.55
TAPEREQ	.097	.057	.093	.048	.056	.062
TAPEUSED	.094	.053	.092	.047	.053	.058
PERFORMANCE MEASURES (AVERAGE VALUES PER HR UNLESS STATED OTHERWISE)						
JOBS/8 HRS	639	825	728	899	942	936
TURNAROUND/JOB	3176.84	3081.54	2726.35	2544.44	1752.68	1189.73
THRUPUT	79. 8	103.13	91.00	112.38	117.75	117.00
BATCH JOBS	43.00	62.38	50.38	63.88	76.38	72.25
INTERCOM JOBS	36.00	42.00	40.25	44.50	39.88	44.00
CPU UTIL	62.00	61.60	79.30	68.90	79.20	67.70
PPU UTIL	182.20	136.30	164.80	105.40	143.50	149.80
CRUS	2825.90	3126.80	2718.30	3422.80	3216.70	2644.10
BATCH CRUS	1362.10	1529.00	1633.70	1913.60	2177.70	1776.80
INTERCOM CRUS	1463.80	1597.80	1084.60	1509.50	1039.00	867.30

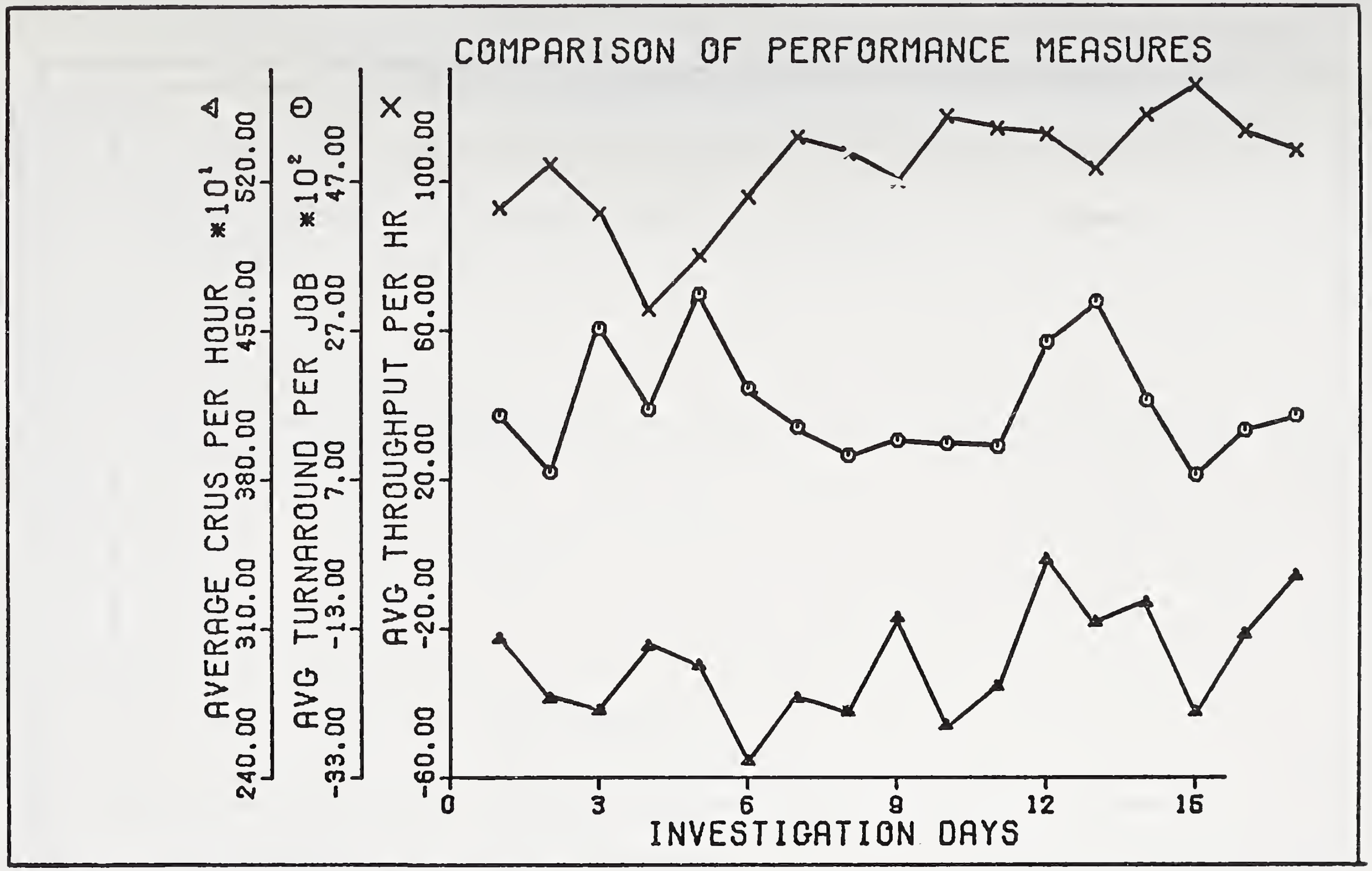


Figure 14. Comparison of CRU's, turnaround, and throughput.

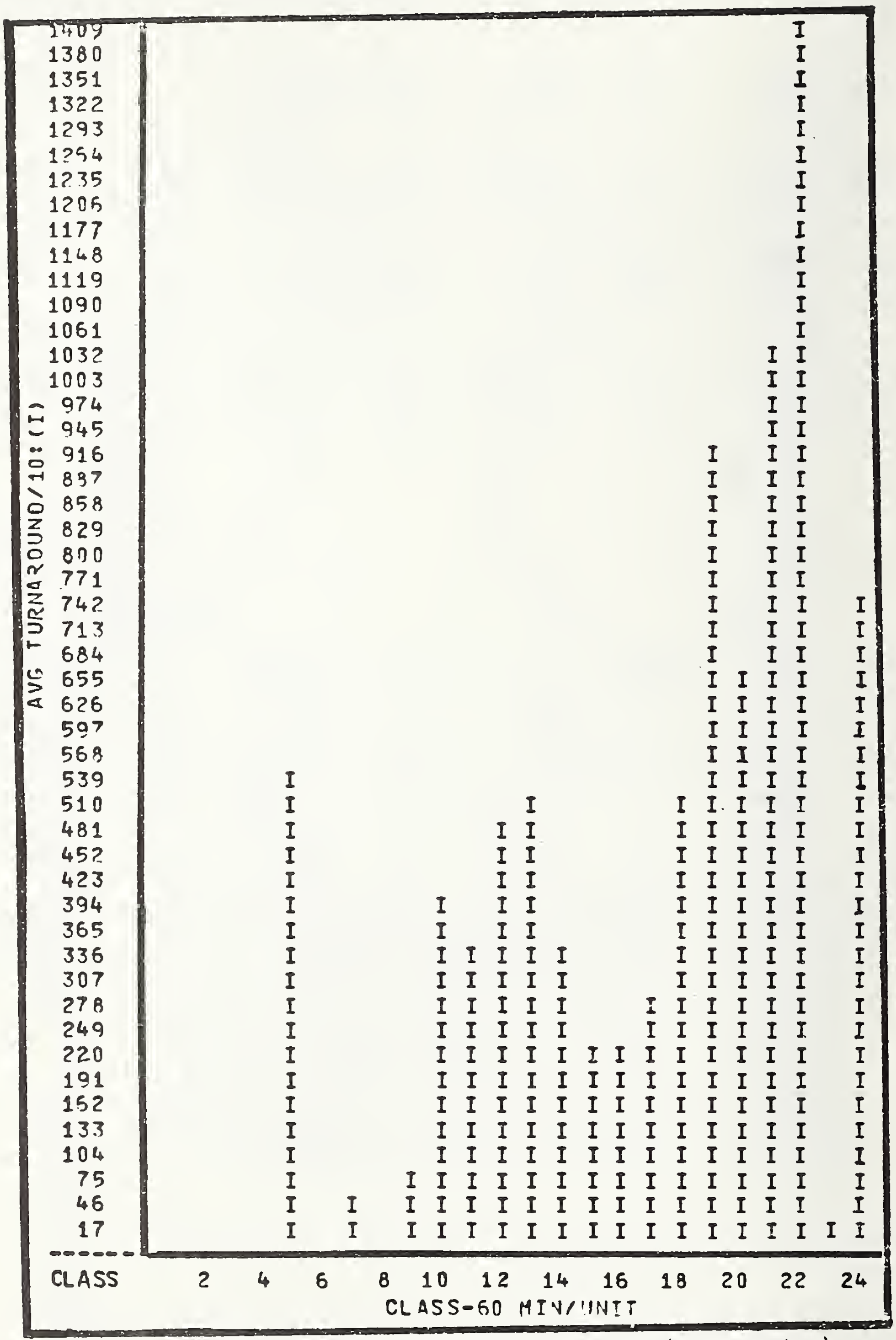


Figure 15. Average turnaround per hour for 24 hours (30 September).

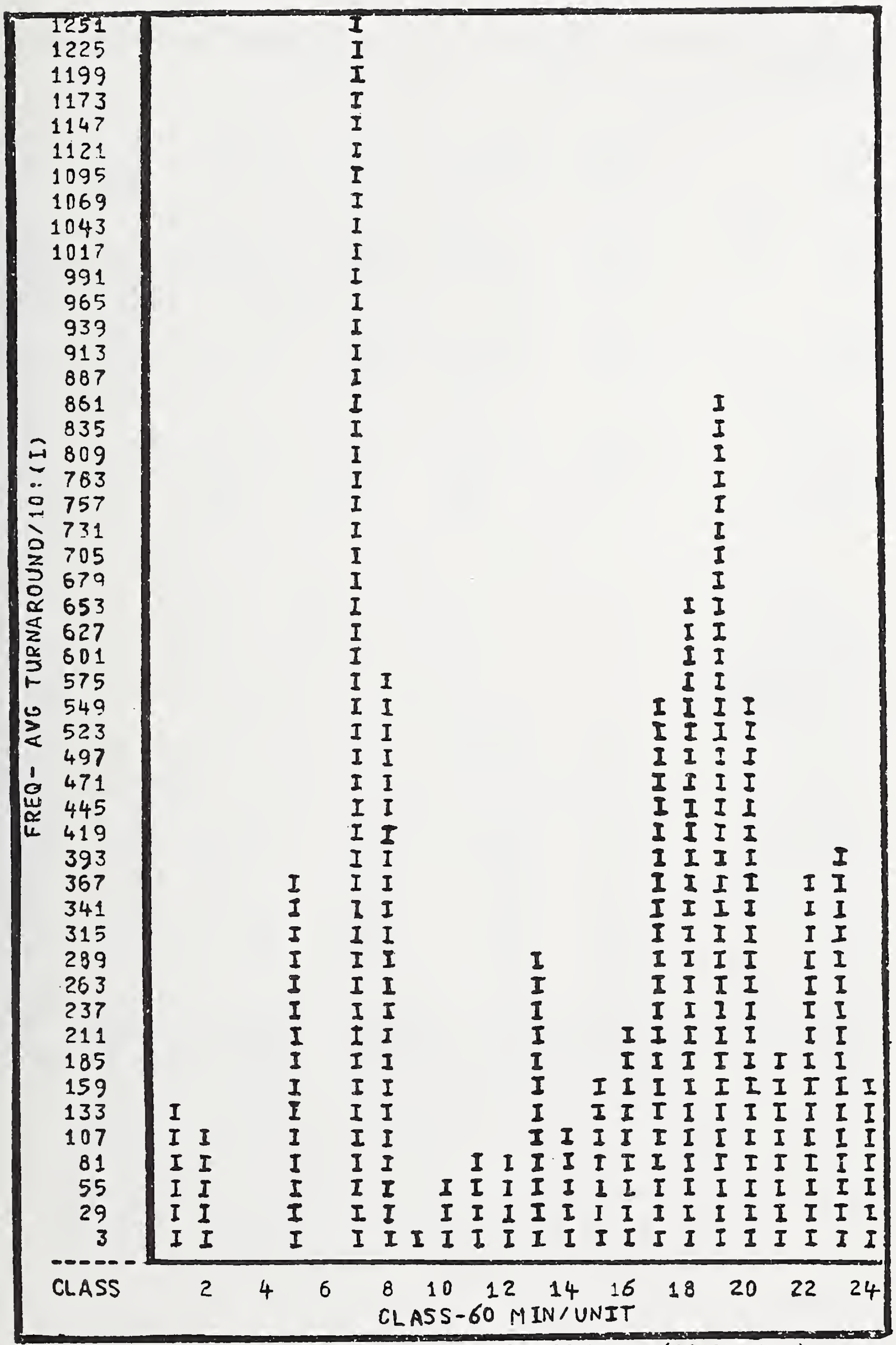


Figure 16. Average turnaround per hour for 24 hours (14 October).

Table 9. CRU Summary

WORKLOAD PARAMETERS (AVERAGE VALUES)	Low CRU Days 7 OCT	Low CRU Days 14 OCT	Medium CRU Days 29 SEP	Medium CRU Days 26 OCT	High CRU Days 19 OCT	High CRU Days 27 OCT
CPTIME	27.31	22.60	35.17	22.89	23.12	23.26
PPTIME	98.03	62.36	95.72	33.71	34.53	68.21
TIMEIO	881.05	15.49	1397.72	18.65	22.95	23.53
TOTCOST	31.46	29.96	43.55	28.02	31.43	33.40
KWS	881.25	701.93	1397.78	806.09	888.58	1022.81
CPOT	804.03	777.24	1158.486	933.11	863.50	1234.32
CMLOC	149777	124030	202640	174853	129792	76735
CPLOC	61.21	94.85	92.19	97.81	97.25	95.27
ROLLOUT	22.46	17.37	61.29	18.10	27.50	16.83
IATIME	65.35	44.50	49.92	54.34	49.523	42.22
INQTIME	808.29	576.55	799.85	525.54	1310.99	645.79
TAPEREQ	.059	.062	.078	.031	.048	.050
TAPEUSED	.055	.058	.076	.031	.047	.040
PERFORMANCE MEASURES (AVERAGE VALUES PER HR UNLESS STATED OTHERWISE)						
JOBS/8 HRS	764	936	525	904	899	863
TURNAROUND/JOB	1912.84	1189.73	1638.52	1355.74	2544.41	1567.16
THRUPUT	95.50	117.00	65.63	113.00	112.38	107.88
BATCH JOBS	50.13	72.25	40.75	67.50	63.88	68.00
INTERCOM CRUS	43.63	44.00	26.50	44.63	44.50	38.50
CPU UTIL	66.90	67.70	65.70	70.00	68.90	72.90
PPU UTIL	202.70	149.80	166.00	102.60	105.40	191.30
CRUS	2477.90	2644.10	3017.60	3073.30	3422.80	3345.10
BATCH CRUS	1680.00	1776.80	1849.80	1804.00	1913.60	2059.20
INTERCOM CRUS	797.90	867.30	1167.80	1269.30	1509.50	1285.90
FIGURE OF MERIT	.000589	.001693	.002678	.000587	.000324	.000460

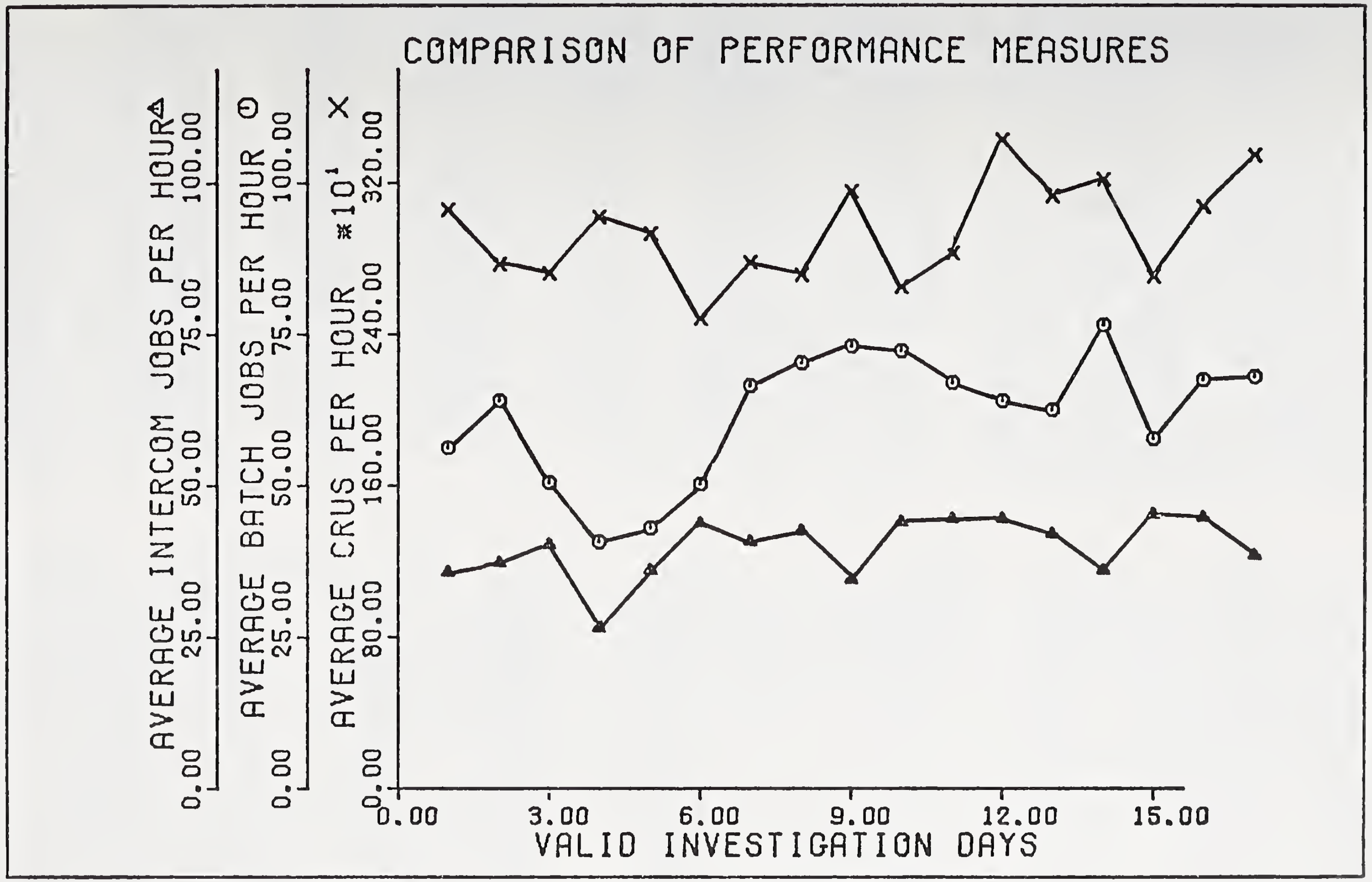


Figure 17. Comparison of INTERCOM, batch, and CRU's.

In figures 15 and 16, the extremes of average turnaround per day are presented for comparison. Slowest average turnaround (52.9 minutes) was on 30 September and fastest average turnaround (19.8 minutes) was on 14 October.

The backlog on the turnaround in both histograms reflects the number of jobs which enter the computer system at the 0800, 1200, and 1600 periods.

9.2. CRU's

As may be seen in table 9, there is a wide range of total CRU's for the eight-hour period on the various days investigated.

For the overall investigations, average CRU's per hour for the CYBER 74 computer system was 3001.21; the range of CRU's was from 2477.93 to 3422.78. In terms of this class of workload, the workload parameter, CPLOC (control points) appeared to be related to CRU's. The days with high CRU's were the ones where more control points were assigned.

As seen in figure 17, the number of CRU's appears to depend upon the number of batch jobs and upon the number of CRU's generated by the batch jobs. INTERCOM jobs and INTERCOM CRU's are relatively constant.

10. Conclusions

In this paper, the workload for a number of investigation days was analyzed by several measurement tools. The techniques and tools used are general and could be applied to any computer workload study.

A brief description of the normal operations of the target computer system was given. Particular attention was given to the assignment of priorities to various job classes in the system.

The measure of a computer system's performance can be given as how effectively and efficiently the system uses its resources to accomplish its objectives. The objective function as perceived by the management of the ASD Computer Center for the CYBER 74 was presented. The objectives were relatively broad, and it would have been exceedingly difficult to select a single adequate measure of performance. Thus, memory used, CPU utilization, computer resource units used, and total number of jobs executed per day were selected as performance measures.

A description of the CLARA software program used to collect accounting data for the computer system was given. Problem areas such as elimination of incorrect data and the number of investigation days to be used were considered.

The accounting data parameters from CLARA provided an adequate source of information from which selective parameters were chosen. The parameters were then treated as univariant functions and time series plots were presented for the parameters. The parameters were also subjected to numerical derivations to produce cumulative distribution and probability density functions. Tables and figures were used extensively to illustrate the workload parameters and performance measures.

References

[1] Agrawala, A.K., J.M. Mohr, and R.M. Bryant. "An Approach to the Workload Characterization Problem." Computer, 9:18-32 (June 1976).

[2] Agrawala, A.K., and J.M. Mohr, "A Model for Workload Characterization," Proceedings of Symposium on Simulation of Computer Systems, 3, 9-18 (1975).

[3] Bard, Y. "Performance Criteria and Measurement for a Time-Sharing System." IBM Systems Journal, 10: 193-216 (1971).

[4] Bell, T.E., B.W. Boehm, and R.A. Watson. Computer Performance Analysis: Framework and Initial Phases for a Performance Improvement Effort. Rand Technical Report R-549-1-PR. Santa Monica: Rand Corp., November 1972.

[5] Bell, T.E. Computer Performance Analysis: Measurement Objectives and Tools. Rand Technical Report R-584-NASA/PR. Santa Monica: Rand Corp., February 1971.

[6] Blitt, William J. et al. The Advanced Logistics System CYBER 73: A Simulation Model. Masters Thesis. Air Force Institute of Technology, Wright-Patterson Air Force Base, Ohio: December 1975. AD/A 019841.

[7] Conti, D.W. "A Case Study in Monitoring the CDC 6700 - A Multi-Programming, Multi-Mode System." Proceedings of Eighth Meeting of CPEUG, 115-118. (4-7 December 1973).

[8] Control Data Corporation. Scope Integrated Scheduler Tuning Guide Programming Systems Bulletin. Pub. No. 60493500. Sunnyvale, CA.: CDC, 1975.

[9] Control Data Corporation. 6000 Series Introduction and Peripheral Processor Training Manual. Publication Number 60100000. Minneapolis, Minnesota: Control Data Corporation.

[10] Dodson, P.O. Control Data Corporation's CYBER 70: Procedures for Performance Evaluation. Masters Thesis. Air Force Institute of Technology, Wright-Patterson Air Force Base, Ohio: May, 1974.

[11] Drummond, M.E. "A Perspective on System Performance Evaluation." IBM Systems Journal. 8, No. 4: 252-263. (1969).

[12] Esposito, J.E. Statistical Analysis to Determine Digital Computer Workload Characteristics. Bedford, Massachusetts: Mitre Corporation, June 1974. AD 786061.

[13] Esposito, J.E. and K. Sreenivasan. A Semi-Empirical Approach for Evaluating Computer System Performance. ESD-TR-73-297. L.G. Hanscom Field, Massachusetts: Deputy for Command and Management Systems, December 1973. AD773312.

[14] Gray, M.A. "A Performance Study of the Multiprocessor Transition in a Large Computer System." Proceedings of Twelfth Meeting of CPEUG, 33-39. (8-12 November 1976).

[15] Gudes, E. and C. Sechler. "Measures for Workload and their Relation to Performance Evaluation." Proceedings of CPEUG: 115-121 (23-26 September 1975).

[16] Hamlet, R.G. "Efficient Multiprogramming Resource Allocation and Accounting." Communications of the ACM, 16: 337-343 (June 1973).

[17] Hellerman, Herbert and Thomas F. Conroy. Computer System Performance. New York: McGraw-Hill Book Company, 1975.

[18] Johnson, R.R. "Needed: A Measure for Measure." Datamation, 16:22-30. (15 December 1970).

[19] Kimbleton, S.R. The Role of Computer System Models in Performance Evaluation. TR-72-4. Ann Arbor: The University of Michigan, Michigan, Department of Industrial Engineering, 30 April 1972. AD 746881.

[20] Lucas, H.C., Jr. "Performance Evaluation and Monitoring." Computing Surveys, 3: 79-91 (September 1971).

[21] Nie, Norman H., et al. Statistical Package for the Social Sciences. New York: McGraw-Hill Book Company, Inc., 1975.

[22] Noe, J.D. and G.J. Nutt. "Validation of a Trace-Driven CDC 6400 Simulation." Proceedings AFIPS Spring Joint Computer Conference, 40:749-758. Montvale, N.J.: AFIPS Press, 1972.

[23] Pleuler, E.F. Design of a Computer System Performance Measurement Program. Masters Thesis. Air Force Institute of Technology, Wright-Patterson Air Force Base, Ohio: December 1974.

[24] Shetler, A.C. and T.E. Bell. Computer Performance Analysis: Controlled Testing. Rand Technical Report R-1436-DCA. Santa Monica: Rand Corp., April 1974.

[25] Sreenivasan, K. On the Use of Accounting Data in the Evaluation of Computer System Performance. Bedford, Mass.: Mitre Corp., January 1974.

[26] Sreenivasan, K. and J.E. Esposito. Experiments with the Burroughs B3500 Computer Using a Synthetic Workload. Bedford, Mass.: Mitre Corp., January 1975. AD/A 004804.

[27] Sreenivasan, K., et al. Construction and Application of Representative Synthetic Workloads. Bedford, Mass.: Mitre Corp., August 1973. AD 769865.

[28] Svobodova, L. Computer Performance Measurement and Evaluation Methods: Analysis and Applications. New York: American Elsevier Publishing Company, Inc., 1976.

[29] Watson, R.A. Computer Performance Analysis: Applications of Accounting Data. Rand Technical Report R-573-NASA/PR. Santa Monica: Rand Corp., May 1971.

[30] Wood, D.C. and E.H. Forman. "Throughput Measurement Using A Synthetic Job Stream." Proceedings AFIPS Fall Joint Computer Conference, 39: 51-56.

B. COMPUTER SYSTEM ACQUISITION

SELECTION OF ADPS FOR THE AIR FORCE ACADEMY: A CASE STUDY

Robert E. Waters

Air Force Computer Acquisition Office
Hanscom Air Force Base
Bedford, Mass. 01731

This paper reviews the competitive acquisition of a large scale ADP system capable of supporting numerous interactive terminals while concurrently processing a large batch workload. The system was designed to act as an educational tool for the cadets while simultaneously supporting research in aerospace mechanics, applied mathematics and other related areas.

Emphasis is placed on the examination of the techniques used in workload analysis and benchmark development associated with this acquisition. This paper cites examples of sound approaches used in these two areas and also discusses a major shortcoming in the benchmarking of teleprocessing systems.

Key words: Air Force Academy; benchmark tests; development of benchmarks; live test demonstrations; selection of ADPS.

1. Introduction

As is the case with most other endeavors, the successful acquisition of a large scale computer system relies on careful preparation and proper execution. Both of these qualities were present during the procurement of such a system for the United States Air Force Academy. Although this acquisition took place in 1971, many of the problems encountered at that time and their related solution are still appropriate today.

One of the more significant problems at that time, which remains with us, is the development of a benchmark test that is both representative and practical. Striking a balance between these two often conflicting factors is difficult indeed. Quite often in an attempt to insure representativeness one loses sight of the practicality of the test. Conversely, in an effort to develop a practical, less costly benchmark test, the representative quality of the test is often compromised. This paper will track the development of the Air Force Academy benchmark test and comment on the techniques used to bring about a benchmark that has been considered both practical and representative.

2. Background

The Air Force was among the first to recognize the computer as an effective management tool and has committed itself to the use of computers for a myriad of applications. An obvious consequence of this fact was the realization that a need existed for Air Force Academy Cadets, who would be the Air Force leaders of the future, to have a thorough understanding of and appreciation for computers. To this end the Air Force Academy has included computer sciences in its curriculum and has instituted a policy of supporting other academic studies with the currently available computer resources.

In the late 1960s it was apparent that the computer system then installed at the Air Force Academy was becoming inadequate for the increasingly sophisticated workload applications. Upon receiving authorization from Headquarters Air Force to upgrade their computer capability, the Air Force Academy joined with the Air Force's centralized computer acquisition office to develop the Request for Proposal and the related benchmark test. This team appoach is profitable because it brings together people who possess extensive experience in the evaluation and selection of computer systems with those people who have the most thorough knowledge and understanding of the particular application.

A segment of that team concentrated on the development of the Live Test Demonstration. Prior to developing a series of benchmark problems a workload analysis was required. But even before this could be undertaken, the team needed a thorough understanding of the operational environment.

Data processing in the academic environment is characterized by large numbers of student users submitting small programs for debugging of either syntax or logic errors. Once a program is running correctly, it is considered finished and seldom executed again. However, another workload dimension is present at the Air Force Academy. The computer system has to support the analytical activities of research associates and Academy students and faculty. These activities are mainly in the areas of aerospace mechanics, applied mathematics and chemistry. The programs used in this research area are generally large and require extensive access to library routines and utilize the full power of the programming languages. Most programs of this type are submitted for processing in a batch or remote batch mode. The operational approach conceived for the new system would adequately support sixty interactive terminals while concurrently processing the large scientific applications in the background.

3. Workload Quantification

The benchmark development team began with an examination of the volume of data handled by the then current system. By reviewing outputs from the time and accounting routines and by analyzing the programs and exercises which instructors had included in their lessons plans, the team was able to determine the volume, content, and size of transactions entering and leaving the system. Further analysis revealed the types of Input/Output devices used and the traffic load at various times of the day. Similar investigations into the experiments then being conducted provided the volume, size, and content of programs and data used in the areas of research.

Since the activities of the cadets and the research analysts are fairly closely controlled, the projected workload throughout the anticipated system life of the computer was easy to calculate. A direct result of this calculation was the determination of the numbers and capabilities of the various Input/Output devices that would be needed for the anticipated system. The basic hardware requirements of most all peripheral devices were obtained through this exercise.

4. Workload Categories

With this statistical information in hand and with the knowledge that the system would be operating in batch, real time, and interactive modes of processing in a multiprogramming environment, the team began to formulate a scenario for the benchmark test. The first step in this process was the establishment of workload categories. Four categories were identified. Category One included housekeeping and general utility routines. (Batch processing with a high concentration on Input/Output operations.) Category Two represented the data processing requirements to support beginning and intermediate courses in computer programming. It was established that approximately 800 cadets would participate in this workload category. The programs were small and often contained numerous errors. Of primary importance in this category was a requirement for a relatively rapid turnaround. (Interactive processing.) Category Three represented the data processing requirements to support advanced courses in computer sciences and other courses in the sciences and in engineering. The student-generated programs covered the full scope of data processing applications. (Interactive processing of a more sophisticated nature.) Category Four represented the requirements needed to support the research and analysis activities. The programs in this category were mathematical in nature, generally large in size, and required extensive use of library routines. (Batch processing with a high concentration on computation.) Thus, the team had established workload categories which covered the various types of computer resource usage encountered at the Academy.

Once the workload categories were established, it was necessary to determine the relationship of each category to the whole. That is, the establishment of what percentage each category was of the total workload.

5. System Characteristics

It was now necessary for the team to conduct a detailed examination of the anticipated telecommunications network. This included a determination of the need for and capabilities of multiplexers or concentrators; the type of terminals to be employed; baud rates to be used; and the manner in which the terminals were to be connected to the network.

During this phase of the workload analysis a careful examination of external influences was also undertaken. Careful consideration was given to how these influences affected the volume of traffic flow throughout a given day. An identification was made of those periods of the day which were associated with "peaks and valleys."

When all the significant factors of the workload characteristics had been identified and correlated, when the various segments of the workload had been placed in categories and the data volumes had been assigned to the appropriate categories, then a profile of the workload began to emerge. This profile of the workload served as the model used in the development of the Live Test Demonstration.

With this workload profile in hand the team searched the vast reservoir of existing programs to find valid representatives of the four workload categories. Those members of the team most familiar with the overall application chose programs which most closely represented the type and content of the programs within each category. Once the representative programs were selected, the number of iterations of each program was manipulated until the proper percentage relationship among the various categories was obtained. In some instances alterations in the volumes of data were necessary in order to reflect the proper amount of peripheral activity. However, when doing so, care was taken to insure that the content and format of the data were not altered.

Each program selected for inclusion in the benchmark test was assigned specific hardware devices for program handling and for input/output handling. Programs were arranged in a specific entry sequence and were assigned a specific introduction time. The device assignments, the sequencing and introduction times of the programs were established to reflect the structure of the telecommunications network, the characteristics of the workload and the normal "peaks and valley" associated with the Academy's computer usage.

The normal workday at the Academy at that time was sixteen hours. The team chose to use a 2-hour benchmark test to represent that workload. Thus, the volumes and processing activities had to be scaled down to one eighth of the actual figures.

In consort with these activities a determination was made as to the hardware configuration necessary to process the benchmark test. This configuration was established based on the need for a representative number of devices, tempered by the cost to the vendor to assemble and operate the equipment. The more significant components needed at the benchmark test included: CPU, main memory, immediate access storage (as required), magnetic tape units (as required), two line printers, six remote terminals, and two remote batch stations.

6. Testing and Review

Each selected program was tested and "debugged" to insure that it compiled and executed in the manner expected. Two sets of input data were developed and tested. The first set, the sample data, was distributed to the various vendors subsequent to the release of the Request for Proposal. The second set, the live test data, contained minor modifications which did not alter the execution time or the logic flow but caused obvious changes to the output products. This second set of data was retained by the Air Force and presented to the vendor on the day of the Live Test Demonstration.

At this point a review was made of the benchmark model to insure that it properly reflected the more significant characteristics of the actual workload. The benchmark model was subjected to the same type of workload analysis as was the projected workload. This exercise produced a workload profile of the model which was compared to the previously developed profile of the actual workload. Results of this comparison revealed a need for minor adjustments to the benchmark model.

7. Documentation

Once it had been determined that the benchmark model had attained an acceptable level of representativeness, it was time to begin the development of the benchmark material. Care was taken to insure that the benchmark material was accurate and complete. The benchmark material included programs, program listings, record formats, narratives of the programs and data in the appropriate medium.

In addition to the benchmark material, a thorough description of the benchmark test was included in the Request for Proposal. Tables were developed which specified by program, the frequency of the program's occurrence during the timed mix, quantities of data involved, device assignment, response times (where appropriate) and program introduction time during the timed mix.

The care and thoroughness spent in the development of the benchmark documentation greatly aided the vendors' ability to understand the demands of the benchmark test. This understanding, in turn, reduced the potential for slippage in overall schedule, and reduced the resources and preparation costs expended by the vendors.

8. Summary

A review of this project reveals a number of factors which go a long way toward insuring a proper benchmark test.

a. A team made up of individuals who are familiar with the application in question and specialists familiar with computer acquisition procedures.

b. A gathering of statistical information concerning the volumes of data entering, being processed, and leaving the system.

c. An estimation of the workload projected over the anticipated system life.

d. An examination of the manner in which the data is manipulated and the environment in which this is done.

e. The identification of categories of workload and the establishment of the relationship of one to another.

f. The selection of programs to properly represent the workload of each category.

g. Thorough testing and "debugging" of each program used in the benchmark test.

h. Review, adjustment and a final review of the benchmark mix to insure it represents the actual workload as closely as possible.

i. A detailed description of the benchmark test in the Request for Proposal designed to instruct the vendor of what is expected of him.

j. Complete and thorough documentation of the program and data to be used in the benchmark test.

In general, if one follows the techniques outlined above, the possibility of an aborted project or a vendor protest is greatly reduced and the possibility of acquiring a system capable of accomplishing the workload is considerably enhanced.

9. Epilogue

Although much effort was expended to create a benchmark that was practical and representative, one serious shortcoming was experienced. The Request for Proposal called for the proposed system to be capable of supporting sixty remote terminals concurrently. For the sake of practicality and economy the benchmark called for only six remote terminal devices and two remote batch stations. Once the selected system was installed, however, it became immediately apparent that the delivered software could not support the remote devices in the required manner. Much time and effort was spent by both vendor and Academy personnel to resolve this problem. It was a situation not cherished by either party.

A more accurate measurement of the proposed systems' telecommunication capabilities could have been accomplished had a remote terminal emulator been available for use at the benchmark. Such a device would have brought to light at the time of the benchmark the kinds of shortcomings which were discovered only after the system was installed. Unfortunately, remote terminal emulators were not available at the time of this acquisition. They are available today and it is recommended that consideration be given to these devices in any benchmark test designed to represent a workload environment which utilizes a large number of remote terminals.

VALIDATION - ALL IMPORTANT IN BENCHMARKING

L. Arnold Johnson

Federal COBOL Compiler Testing Service
Department of the Navy
Washington, D.C. 20376

This paper discusses validation of benchmarks associated with Government procurement of data processing equipment. It is not uncommon, after a procurement, to find that the computer system obtained did not have the processing capability or capacity required by the Request for Proposal (RFP), even though the vendor successfully passed a benchmark. In many cases this insufficiency can be attributed to inadequate validation of the benchmarking process. The benchmark is a tool for validating the capability of a vendor's computer system and there are certain characteristics which the benchmark must have. This paper describes these characteristics and some techniques for incorporating them into a benchmark. Benchmark preparation, benchmark execution, software configuration are four areas which may affect the representativeness of the benchmark. Improper validation of the software configuration for the benchmark or the optimization applied to benchmark programs, for example, may result in a benchmark implementation which would be uncharacteristic of the production workload.

The approach taken to benchmark validation can greatly affect the ease and timeliness with which the benchmark demonstration can be performed. A well-defined set of audit procedures and a multiple-step validation, with appropriate contingency plans are essential to reduce benchmark reruns and vendor protests. An approach for developing benchmark audit procedures is presented, together with an outline of the steps in the benchmark validation process.

Key words: Benchmarking; benchmark audit procedures; benchmarks associated with government procurement of computers; validation.

1. Reason for Benchmark Validation

Considerable work and research are being done on techniques which can be used for benchmarking and how to represent a predicted workload through benchmarking for computer selection/procurement [1][1] Validation of the benchmarking process on the other hand, is given much less attention. If the validation process is not properly conducted, misrepresentation of the benchmark workload can occur and a less than adequate computer system may be selected.

Goff's article, "The Case for Benchmarking" [2], suggests that benchmarking is the only available means of evaluating large and complex ADP systems by a common standard (as opposed to other approaches such as simulation). This view is also expressed in a Federal Government procurement regulation that in effect says that simulation shall not be used as the sole selection criterion for selection of computer systems. Kiviat's article, "A

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

Challenge to Benchmarking," (which is appendix E of reference [1]), suggests that there are sources available for testing the representativeness of benchmarks and that a fairly good feeling on just how representative the benchmark is of the predicted workload may be obtained. One is led to conclude from this, that the means for determining the credibility/capability of a vendor's computer system will be via benchmarking, and the benchmark will be a valid representation of the production workload. Normally the benchmark is executed on the vendor's computer system at the site of his choosing. This means that the benchmark material is given to the vendor in advance and he is the one who prepares the computer system environment, puts together the components of the benchmark to represent the workload and executes the benchmark. This is accomplished by the vendor based on his interpretation of the benchmark instructions provided with the benchmark material.

No matter how well the benchmark has been modeled to represent a predicted workload, its representativeness can be easily lost in the benchmark conversion, the vendor preparation of the workload mix, or the hardware/software that is used for execution of the benchmark. Generally, the benchmark is short in duration and only representative of a portion of the total monthly, weekly, or even daily user processing requirements. Thus a small deviation of the workload in the benchmark from the benchmark requirements can have a much larger impact on the computer system's ability to meet the installation's total processing requirements. The truth is that this situation can and does occur, and unfortunately is often not detected until after the computer system is procured. Proper validation during the benchmarking process can reduce the possibility that an inadequate computer system will be selected.

The purpose of benchmarking is to have the vendor demonstrate the ability of his proposed computer system to meet the requirements set forth by the Request for Proposal (RFP). Validation of the benchmarking process serves to preserve the integrity of the benchmark and the workload it represents. Benchmark integrity is maintained only if: (1) all processing is performed; (2) all processing is performed correctly; (3) the hardware/software environment is as defined in the benchmark instructions; and (4) the character of the benchmark in terms of optimization, coding style, etc. is consistent with the installation's production environment.

Another important reason for validating the vendor benchmark implementation is to ensure that each vendor has an equal opportunity for providing his most competitive computer system. Vendor protests and procurement delays can result if each vendor is not processing an equivalent workload or is not forced to adhere to the benchmark requirements.

2. Validation Begins with Benchmark Design

In order to adequately validate a benchmark implementation, certain benchmark qualities are necessary. It is difficult to include these qualities in a benchmark without having planned for them when the benchmark was designed. There is an advantage to both the vendor and the benchmark validation team, if the benchmark has these qualities. One task of the vendor is to convert the benchmark material for use on his system. If the benchmark design is lacking in certain qualities it is likely that the vendor will have to expend more of his resources to merely get the benchmark into execution. In addition to the required conversion of the benchmark material for use on his system, the vendor typically applies whatever optimization that is possible to the benchmark. By either the conversion or the optimization process the vendor may have changed the representation of the benchmark. It is the job of the benchmark validation team to ensure that any modifications to the benchmark material by the vendor are proper in terms of the processing requirements. If the benchmark does not include qualities which enable one to evaluate the correctness of the benchmark implementation, then the task of validating is at best difficult. The vendor must also be able to verify that the benchmark implementation is correct. If it is not correct, it could affect the workload being represented and cause the vendor to waste time and money in sizing an inadequate computer system. The following paragraphs discuss some qualities which are desirable in benchmarks.

2.1 Auditability

Auditability is the ability of the benchmark to provide the information necessary for evaluating whether the benchmark implementation was correct. That is, being able to determine from the benchmark execution that all the required processing was performed, that all the processing was performed correctly and that the benchmark implementation adhered to the requirements imposed by the benchmark instructions. It is convenient for the benchmark validation team if the auditing process is as automated as possible. When benchmark programs are self-auditing (ability of a program to test its own processing results, for example, against some predetermined value), it reduces the work the benchmark validation team must do. Also, benchmark systems which have predesigned checkpoints for determining whether the benchmark execution is correct have been found to be very useful. Information resulting from execution of the benchmark such as the program displaying file record counts, hash totals or producing summary reports add to the benchmark's auditability. Frequent checkpoints of this type are advantageous from the viewpoint of the computer vendor as well as the auditor, since the vendor may have to do some debugging on the benchmark.

2.2 Portability

Portability is the capability of benchmark programs to be implemented on each computer vendor's system with minimal conversion effort and expense. This means that few modifications, if any, to the benchmark programs are required to make the programs operational on each vendor's computer system. The objective of a competitive procurement is to obtain a capable computer system at the lowest price. Nonportable benchmarks cause the computer vendor to direct his resources to conversion instead of sizing the computer system to satisfy the RFP requirements. The expense which a vendor incurs on the benchmark demonstration is eventually reflected in the prices which he can afford to offer the procuring agency. Less than portable benchmarks can have a serious effect on the benchmark with respect to benchmark representativeness. The more modifications the vendor must make to the benchmark in order to implement the benchmark the more difficult it is for the benchmark validation team to verify that the workload represented by the benchmark has not been affected, e.g., optimization has not destroyed the representativeness of the benchmark.

For a benchmark to be portable, standard programming languages need to be used and machine dependencies need to be kept to a minimum. The machine and language features which cause nonportable benchmarks are beyond the scope of this paper. There is literature available that discusses the cause of nonportable benchmark programs and data, and what can be done to make benchmarks portable [3].

2.3 Repeatability

Repeatability is the ability of the benchmark to produce the same execution results on each vendor's computer system. Benchmarks which produce different results on different computer systems raise doubt as to whether the benchmark was implemented correctly or whether all the intended processing was actually performed. It is recognized within the computer community that the exact results may not be possible among computer systems due to differences in numerical precision. Nevertheless, the validation team needs to know whether different results are truly due to precision. It is therefore desirable if the benchmark can be executed on different computer systems to see how repeatable the benchmark is before it is officially released to the vendors.

2.4 Predictability

Predictability is the ability of the benchmark to produce an expected result from a given set of data [4]. One technique that is often used in benchmarking to reduce the possibility that the benchmarked computer system may be tuned to a given set of data is to change some part of the benchmark before the benchmark test begins (making <u>substantial</u> changes to the benchmark of course is not a good practice). If the results that are produced when a change is made to the benchmark cannot be predicted, then its value as a validation tool is questionable.

2.5 Modifiability

Modifiability is the ability of the benchmark to be easily identified and updated. It may be necessary during the period of the benchmark implementation or for validation purposes to change the benchmark programs or data. There must be a method by which these changes can be communicated to the computer vendor. Techniques such as line numbers in sequential order on each source line within the program, name of the program embedded in the program in a common location, or file name and unique record number embedded in each record of the file can simplify benchmark modification. If the vendor has questions regarding the benchmark this quality is useful in that communication.

2.6 Conformity

Conformity is the ability of the benchmark, as delivered to the vendor, to be consistent with the software/hardware requirements defined in the RFP. If, for example, the RFP requires that a Standard COBOL compiler be provided, then those benchmark programs used in the benchmark should be written in Standard COBOL and contain no vendor extension.

3. What to Validate

A clear understanding of the role benchmarking will take is important in validation of a computer system against the RFP requirements. The purpose for which benchmarking is being used influences the validation approach which is taken and the items which need to be checked. A combination of contractual obligation and benchmarking is the best practice to verify the credibility of a vendor's claim. There are some RFP requirements which cannot be adequately validated by benchmarking. Computer system reliability, for example is a requirement which can be found in many RFP's. There is a question as to how this can be done through benchmarking. To verify that a computer system will remain up and available for processing, say 95% of the time, in a normal 2-hour benchmark would not be meaningful. A much longer time frame would be required to determine if the vendor can comply with such a requirement. System reliability is a requirement that will exist throughout the life of the computer system, and therefore, it might be better if this were a contractual obligation with reasonable penalties for noncompliance.

There are basically two types of benchmark demonstrations. One type of benchmark demonstration is for the purpose of testing the capability of a specific feature/function of the computer system often referred to as a functional demonstration [5]. The other type of benchmark demonstration is one which tests the ability of the computer configuration to process a given workload, within given time constraints, often referred to as a benchmark mix demonstration [5]. Functional demonstrations are typically expensive and time consuming for the vendor and provide little benefit toward validating RFP requirements except under certain conditions. Normally, functional demonstrations are used to verify those RFP requirements about which there remains a question as to whether some part of a prospective system is capable of meeting a requirement.

A benchmark mix demonstration is useful for establishing the ability of a computer system to process a given workload. This type of demonstration involves the collection of programs, data and work requirements which together form a workload quantity that is executed on the benchmark computer system.

The specific items which must be checked for the benchmark validation process vary with each benchmark. These items can be categorized into four general areas. They are benchmark preparation, benchmark execution, hardware configuration and software configuration.

3.1 Benchmark Preparation

As discussed earlier in this paper the techniques by which the benchmark is prepared for execution can impact the representativeness of the benchmark. The purpose of validating benchmark preparation is to ensure that the integrity of the benchmark was not lost in the vendor's conversion or in developing the workload for the benchmark. Occasionally, the vendor, in preparing the benchmark for use with fewest or lowest-priced hardware resources, will apply optimization to the benchmark which would be uncharacteristic of the workload in a production environment. There are some items which can be checked to verify that the vendor's preparation approach has not violated any benchmark instruction requirement and is characteristic of its use in the production environment:

(a) the modifications made to the benchmark programs by the vendor;

(b) the changes or reformatting of the data by the vendor from that originally provided;

(c) the optimization techniques applied to the benchmark programs;

(d) the function and use of the software developed specifically for the benchmark.

3.2 Benchmark Execution

Even though the vendor correctly prepares the benchmark for execution, the way in which the benchmark is executed can affect the representativeness of the benchmark workload. The validation team needs to check that the execution ground rules by which the benchmark was modeled are also maintained by the vendor. Some items which can be checked to ensure proper benchmark execution are:

(a) the compiler or other software options used are consistent with the modeled benchmark;

(b) the data files have been preloaded to the correct device type;

(c) the data files resulting from benchmark execution are assigned to the correct peripheral device type;

(d) the benchmark job tasks were initiated in the proper sequence;

(e) the starting status of all hardware components was correct;

(f) the job queues were properly loaded before start of the benchmark test;

(g) the execution priority between different processing types, e.g. time-sharing vs. batch processing is consistent with the benchmark requirements;

(h) all output produced by any previous benchmark execution has been purged from the system;

(i) the benchmark execution adheres to the time requirements imposed by the benchmark instructions;

(j) the jobs entered into the benchmark test were from the proper location/device.

3.3 Software Configuration

The software configuration used for the benchmark can affect the size and power of the computer system required to perform the benchmark. The computer vendor has many options and versions of operating systems, compilers, etc. available to him. Given a stable workload he can tailor the software configuration to process the workload in a very efficient manner requiring only minimal amount of computer resources. Since a benchmark test rarely requires all of the functional capability defined in the RFP, it is highly conceivable that a "stripped-down" version of the operating system software and its components would be adequate to execute the benchmark. However, if the benchmark process is to be representative of a production workload, <u>all</u> of the software capabilities required by the RFP should be present.

Language compilers, or the resulting programs produced by them, play an important role in most computer installations. Most Government data processing installations require the use of high-level languages (namely, COBOL or FORTRAN), and it is common to provide benchmark programs in these languages. Current Federal Property Management Regulations (FPMR) require that COBOL compilers offered as a result of the requirements set forth in a RFP must comply with one of the four levels of Federal Standard COBOL [6]. Although there is no similar regulation for FORTRAN, it is common to find Government RFP's requiring American National Standard FORTRAN [7].

Also, there is a FPMR which requires that all COBOL compilers brought into the Federal inventory be validated [8]. Again, there is no similar regulation for FORTRAN; however the Navy in practice requires that the FORTRAN compilers also be validated as part of their procurement process. The Department of the Navy maintains the COBOL Compiler Validation System (CCVS) [9] and a FORTRAN Compiler Validation System (FCVS) [10] which are used to test a given compiler's conformance to its respective language specification. Validation services are also provided upon request.[2]

If the vendor is required to deliver compilers which conform to their respective language standards, then it is proper that the same compilers should be used for benchmark purposes. A case in point is a Government procurement in which a COBOL compiler used for the benchmark did not fully conform to the standard language specifications. Later when it was brought into conformance with the language standard, the memory requirements and the execution times of the compiled programs increased substantially. In this instance the compiler was brought into compliance following the installation of the computer system. It is quite likely that if the compiler that met the requirements of the RFP was used in the benchmark, it would have required a larger system than was actually bid. In fact, the system resulting from this procurement required equipment upgrades above and beyond that originally planned when

[2]Information on the FCVS or the CCVS can be obtained from the Director, Federal COBOL Compiler Testing Service, Department of the Navy, Washington, D.C. 20376.

the system was initially purchased.

Current Government regulations require that COBOL compilers be validated only before being brought into the Federal inventory. A good policy, however, might be to have the compiler validated before the vendor's benchmark demonstration so the validation team could evaluate the compiler being used in the benchmark.

To test all computer system software used in a benchmark test for compliance with the RFP requirements would be an impossible task. Typically, some combination of vendor software certification and software testing is used. A software certification is useful when it is impractical to test the system software for compliance to RFP requirements. Certainly, where practical, the software most affecting the performance of the system being benchmarked should be tested.

3.4 Hardware Configuration

For the benchmarking environment to be representative of the production environment, the hardware configuration for the benchmark should be the same as that proposed by the vendor. There are reasons however which do not make this possible. The benchmark workload is a subset of the production workload, thus a full complement of computer configuration proposed may not be necessary in order to execute a benchmark test. Nevertheless, one must verify that the smaller benchmark configuration would be representative of the processing environment on the production system.

As in validating the software configuration, testing all hardware components for functional compliance with the RFP requirements would also be a very lengthy and difficult process. Typically some combination of vendor hardware certification and physical inspection is used for validating the hardware configuration for compliance with the benchmark instructions.

The vendor should be asked to provide a configuration schematic and list of hardware components of the benchmark system for the validation team to review. Some items which can be checked by the validation team are:

(a) the hardware components configured for the benchmark are the same as listed on the configuration list;

(b) any difference between the benchmark configuration and the proposed configuration are permissible in accordance with benchmark instructions or Government approved waivers;

(c) any extra equipment on the configuration floor is not operational or part of the benchmark configuration.

4. Audit Procedures

Quite often the difference between a well-run office operation and one which is mediocre, is that the well-run office has documented operating procedures for all workers to follow. The same is true in benchmarking. For the vendor to plan for the benchmark demonstration, he needs to know what is expected of him. Also the benchmark validation team members need to know their responsibilities and the things that need to be checked for the validation. This information can best be provided by means of well-documented audit procedures.

Good audit procedures for use in benchmarking will, first, define the ground rules by which the benchmark will be conducted and, second, provide a detailed description of each step in the validation process [11]. The audit procedures should describe the audit material that the vendor must provide and how this material will be used for the validation. It is a good practice to have validation forms and checklists which supplement the audit procedures. The reason for validation forms and checklists is to aid the validation team members in the mechanics of performing the validation.

It is convenient if the validation forms identify the location and include a copy of the exact data content that is to be verified. This may either be on the form itself or it may be a sample listing attached to the form. If the data which will be used in verification is kept with the validation forms or checklists and not included in the audit procedures, then the audit procedures, with a sample of the validation forms/checklists, can be given to the vendor. This will preclude the need to develop separate audit procedures for the vendor and allow both the validation team and the vendor to work from the same procedures.

Having ground rules for the benchmark demonstration is a vital part of the audit procedures. Ground rules define those items which are critical to smooth operations of the benchmark test demonstration, but are not an identifiable step in the validation process. Some of the ground rules which are commonly defined in benchmarking are:

(a) the composition of vendor and validation teams which would be permitted in the immediate vicinity of the benchmark demonstration configuration;

(b) the conditions or events which will be considered the beginning of the benchmark test; i.e., when timing clocks for the tests will start;

(c) the conditions or events which will be considered the completion of the benchmark test; i.e., when timing clocks for the benchmark test will stop;

(d) the procedure that will be followed if program or equipment failure occurs during the benchmark test;

(e) the restart procedures that the validation team will expect the vendor to follow in the event the benchmark test must be restarted;

(f) the persons who will be the "vendor spokesperson" and the "validation team spokesperson" during the benchmark test.

The procedural steps of the audit procedures should describe the activities that are expected to occur during the benchmark test. A logical approach is to divide the audit procedures for the benchmark test into pre-execution, execution and post-execution sections. Within each of the three sections the steps that will be followed by the validation will be described. Where practical, a validation form/checklist is useful for each validation step.

The following is a list of some common steps that can be found in benchmark validation.

4.1 Pre-execution

(a) Audit of the Benchmark Hardware Configuration. This step would involve a walk-thru and inspection of the hardware configuration. Also, any required vendor hardware certification would be signed at this time by the vendor.

(b) Audit of the Benchmark Software Configuration. This step would be an inspection of the list of system software on the benchmark system provided by the vendor. Also, any required vendor software certification statements would be signed at this time.

(c) Change Benchmark Programs/Data. This step would involve having the vendor incorporate minor changes, for validation purposes, into the benchmark and verify that the changes were applied correctly.

(d) Audit of the Programs/Data. This step would be an inspection of a directory list of all data files and programs catalogued on the system.

4.2 Execution

(a) Starting the Timing Clocks. This step would involve the designated team members starting the timing clocks at the moment that the test is to begin.

(b) Audit of the System Consoles. This step would involve the monitoring of the console operator activities and verifying that all benchmark jobs/tasks initiated and terminated in the required test time.

(c) Audit of Special Events. This step would involve the monitoring of special output or timing requirements that will result during benchmark execution; e.g., response times, messages, job initiation at specific times or job execution sequences.

(d) Stopping the Timing Clocks. This step would involve the designated team member stopping the timing clocks at the moment the benchmark test is completed.

4.3 Post-execution

(a) Produce the Audit Material. This step would involve gathering the data resulting from execution of the benchmark and executing any programs for producing the required audit material.

(b) Audit the Benchmark Execution Results. This step would involve the checking of the files, record counts, program reports, system accounting data and other data resulting from execution of the benchmark.

5. Multiple-step Validation

Benchmarking is a process whereby the computer vendor demonstrates the ability of his computer system to execute the benchmark within the required time constraints, and the procuring agency validates that the benchmarking process is correct. Because the computer vendor is required to implement the benchmark on his system he has an opportunity to become familiar with its characteristics. The benchmark validation team, on the other hand, does not have an opportunity to participate in the implementation of the benchmark on each vendor's computer system. Therefore, the validation team must rely on technical literature and information provided by the vendor to familiarize themselves with the vendor's benchmark implementation approach.

Often, the first in-depth review of the vendor benchmark implementation by the validation team occurs during the site visit to the computer vendor for the benchmark demonstration. This is insufficient for the benchmark team to adequately validate a benchmark implementation. Discovery of incorrect benchmark implementation at this point would very likely cause the benchmark demonstration to be postponed and require the benchmark validation team to revisit the vendor's computer site.

A validation approach which permits the validation team to review each vendor's benchmark implementation prior to the on-site visit is the best practice. Possibly a two-step validation where the first step is a review of the vendor's benchmark implementation before the site visit and the second step is another review and validation at the on-site benchmark demonstration. The accepted benchmarking practice is to evaluate the benchmark results at the vendor's demonstration site so that an indication can be given the vendor whether the benchmark passed or failed prior to the validation team's departure.

The Navy in one of its recent procurements used a multiple-step validation approach which resulted in a "smooth" benchmark demonstration by most benchmarking standards. The first step was a review of the vendor's benchmark implementation when the vendor's proposal was submitted. The second step was a discussion at vendor oral presentations of the implementation discrepancies found in the first step. The third step was a final review at the vendor's on-site benchmark demonstration. There were provisions in the validation plan for further checks following the site visit but this step was not required for any of the three vendors benchmarked. The first step, by far, was the most beneficial. For the first review, the vendor was required to provide the validation team with: (1) all execution results from the benchmark; (2) all the audit material that would be used by the validation team; (3) a hardware configuration schematic of the benchmark configuration; (4) an explanation for each change made to the benchmark material; and (5) an explanation of any special software used in the benchmark that was not reflected in the technical documents supplied by the vendor. A review of this information revealed benchmark implementation discrepancies which if found at the benchmark demonstration would certainly have required another visit to the vendor's benchmark site for some vendors. Also, this review caused a dialogue between the validation team and vendor which answered many questions regarding the benchmark.

A multiple-step validation has some very important advantages particularly for the benchmark validation team. Having a "dress rehearsal" before the site visit enables the validation team to fine-tune the audit procedures and the validation forms/checklists. One may find that because of the unique nature in which audit material is provided by a vendor's computer system, the validation forms or procedures may need to be modified to be useful.

Another important advantage is the information provided for the "dress rehearsal" serves as excellent training material for the benchmark validation team members. FIPS PUB 42-1 suggests that a trial benchmark prior to release of the benchmark material to the vendor serves as a valuable training exercise [5]. This approach has merit but it is both untimely and incomplete as a training mechanism for the validation team. With the trial benchmark approach the time period between validation team training and benchmark demonstration is too long a period to be effective. Also, this approach does not enable the validation team members to become familiar with the format of the audit material that will be provided by each vendor. An early review by the validation team can be beneficial for the vendor as well. If the first review step is properly planned, the vendor has an opportunity to correct any benchmark discrepancies and resize his computer configuration before the benchmark demonstration.

There are those who may consider this first validation step as additional work and expense which is required of the vendor. Indeed this is partially true but it also forces the vendor to check-out his benchmark implementation early, thus leaving more time for him to size his most competitive computer system.

6. Summary

Validation associated with vendor benchmarking is often not really given the consideration that it should until after the benchmark has been modeled. How well the benchmark does in verifying that the vendor's proposed computer system is adequate is in part due to the qualities which have been incorporated into its design.

The ability of the benchmark to retain its representation through the vendor's implementation is largely the result of the requirements placed on its implementation. A modeled benchmark may be very representative of the predicted workload, but if the benchmark, as converted by the vendor, is not run under the environment characteristic of the production workload, then it may cause a computer system to be selected that is not adequate for the production workload.

Procurement delays and vendor protests are of concern to most and the possibility of their occurrence can be reduced if a good validation plan is in effect. A multiple-step validation and well-documented audit procedures serve to provide this end.

References

[1] Benwell, N., Benchmarking Computer Evaluation and Measurement, John Wiley & Sons, Inc., New York, (1975).

[2] Goff, N.S., The Case for Benchmarking, Computers and Automation, 22, 5, (1973), p. 23-25.

[3] Baird, G.N., Johnson, L.A., System for Efficient Program Portability, Proceedings of the National Computer Conference, (1974), p. 423-429.

[4] Oliver, P., Baird, N., Cook, M., Johnson, A., and Hoyt, P., An Experiment in the Use of Synthetic Programs for System Benchmarking, Proceedings of the National Computer Conference, (1974), p. 431-438.

[5] Federal Information Processing Standards Publications 42-1, Guidelines for Benchmarking ADP Systems in the Competitive Procurement Environment, U. S. Department of Commerce, National Bureau of Standards, (1977).

[6] Federal Information Processing Standards Publication 21-1, COBOL, U. S. Department of Commerce, National Bureau of Standards, (1975).

[7] American National Standard FORTRAN, X3.9-1966, American National Standards Institute, New York, (1966).

[8] U.S. Government Federal Property Management Regulation 101-32.1305-1a, Validation of COBOL Compilers, Federal Register, Vol. 40, No. 221, (November 14, 1975).

[9] Baird, George N. and Cook, Margaret M., Experiences in COBOL Compiler Validation, Proceedings of the National Computer Conference, (1974), p. 417-421.

[10] Hoyt, Patrick M., The Navy Fortran Validation System, Proceedings of the National Computer Conference, (1977), p. 529-537.

[11] Handbook for Preparation of Benchmark Instructions, Department of the Navy, ADP Equipment Selection Office, Software Development Division (1976).

C. ON-LINE SYSTEM EVALUATION

DETERMINATION OF NON-STEADY STATE CONDITIONS IN PERFORMANCE MEASUREMENT RUNS

Nandakumar N. Tendolkar

International Business Machines Corporation
Poughkeepsie, New York 12602

An analytical technique is proposed for determining whether a performance measurement run of an interactive computer system is in steady state. The technique is illustrated by applying it to some measurements of a benchmark workload processed by IMS/VS under OS/VS2 (MVS). An approach is suggested to determine the length of measurement runs required to predict system transaction rate with a given reliability.

Key words: Interactive computer system; performance measurement; renewal theory; statistical analysis; steady state.

1. INTRODUCTION

In the past, computer performance measurement runs were controlled and set up by a limited set of measurements and by gut-feeling. In such cases, the application of the results of a small set of measurements to unmeasured systems often led to unpredictable results. This paper describes an analytical technique that was developed to control performance measurement runs of an interactive computer system, and which was successfully used in detecting non-steady state conditions. The technique is based on the application of renewal theory and statistics and can be applied to the analysis of performance measurement runs of many typical interactive computer systems.

An interactive computer system is a computer system in which a large number of users simultaneously communicate with the system through terminals. Each user sends a transaction to the system for processing. The system processes it and sends a response back to the terminal. Examples of such interactive systems are the Time Sharing Options of OS/VS2 (MVS) and the IMS/VS systems.

In evaluating the performance characteristics of an interactive computer system, the principal components are defined. These are usually the hardware configuration, the operating system, and the program product that interfaces with the users. The hardware configuration is described by the CPU, the memory size, the channels, and the number of terminals.

The objective of performance measurement is to study the various system characteristics as a function of the load on the system. The system characteristics that are of interest are response time, CPU, channel and device utilizations, instructions executed, bytes transferred from and to the direct access storage devices, paging rates, etc. The load on the system consists of a certain set of transactions sent by the user and the rate at which these transactions are sent. Each transaction type needs a specified amount of processing from the system.

The outcome of a performance measurement run is a collection of statistics about the system characteristics and user load during a time interval (the measurement interval) in which a certain number of terminals are interacting with a certain hardware configuration, operating system, and program product.

The majority of performance measurement runs are made to establish a steady-state functional relationship between the system charactertistics and the user load. An example of such a relationship is the mean response time as a function of the load (the number of terminals or the transaction rate). The problem then is to be able to determine whether a steady state was obtained during a measurement run.

Suppose S_T stands for a particular system characteristic (for example, transaction rate) measured during the time interval (T, T+ΔT). We define a system to be in steady state if there is some time T' such that for

$$T \geqslant T', \frac{dS_T}{dT} = 0$$

(that is, S_T is not a function of time for $T \geqslant T'$). In this case, the statistic S_T reaches a steady or stable value. Depending upon the purpose of measurement, one may define a set of such statistics that have to reach steady values to make the measurement a valid measurement. This aspect of measurements has been discussed by Hyman [1][1] and Wyrick [2]. In this paper we present a method of determining whether a measurement run for an interactive computer system has reached a steady state with respect to the transaction rate.

It is not clear whether, for any general case of performance measurement of a real interactive computer system, one can derive the necessary and sufficient conditions that indicate the system is in a steady state. In most real system measurements one does not have an a priori criterion that the system would meet if it were operating in a steady state. One approach that is commonly used both in simulation experiments and in measurements is to divide the measurement interval into a number of segments and to study various statistics for each interval [3]. The problem with this approach is that one must know how much variation should be allowed in the observed statistics from one interval to another and how many intervals should be observed before we are satisfied that the system is in steady state.

An analytic methodology to solve this problem for a specific set of measurements that were planned for studying IMS/VS performance characteristics is presented here. In addition, a methodology is suggested to estimate the probability that a system will end up in a non-steady state if it was in a steady state during the measurement.

2. SYSTEM CHARACTERISTICS

A certain set of performance measurements was planned to study IMS/VS performance characteristics on IBM System/370 Model 158 running under OS/VS2 (MVS). A specific data base was created, and the set of transactions (messages) that the system would process was defined. TPNS [4], running in another processor, was used to send transactions to the IMS/VS system (see fig. 1). The TPNS system was generated in such a way that the host system communicates logically with a fixed number of terminals during a run. The number of terminals is varied from run to run, but is fixed during a run.

The terminals go through a cyclic activity of sending a transaction, getting the response, and sending the next transaction. The flow chart of the user behavior (terminal) is shown in figure 2. The behavior of each terminal is key to the derivation of the conditions for steady state.

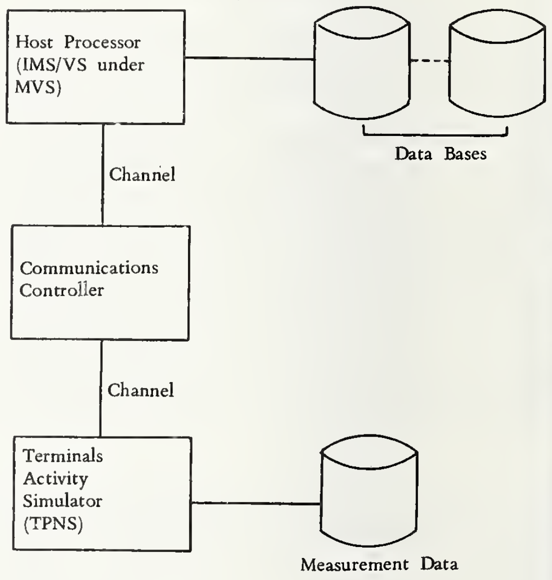

Host Processor (IMS/VS under MVS)
Data Bases
Channel
Communications Controller
Channel
Terminals Activity Simulator (TPNS)
Measurement Data

Figure 1. Performance measurement system set-up

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

A terminal sends a transaction to the system at time T. TPNS then generates a number ΔT at random, with ΔT representing the intermessage delay. For this system, ΔT was uniformly distributed between 75-125 seconds. If the response is received by the terminals at time T', then there are two following possibilities (see fig. 2):

1. If $T' < T+\Delta T$, then the next message from the terminal would be sent at time $T+\Delta T$.

2. If $T' \geqslant T+\Delta T$, then the next message from the terminal would be sent at time T'.

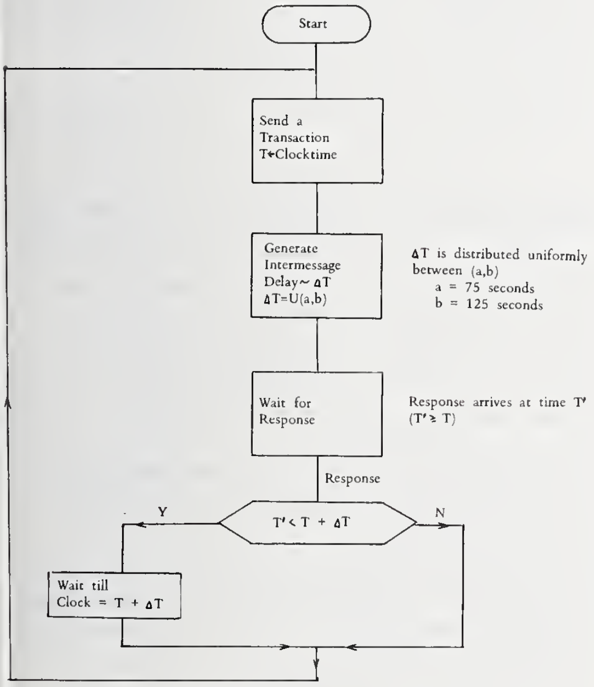

Start
Send a Transaction T←Clocktime
Generate Intermessage Delay~ ΔT ΔT=U(a,b)
ΔT is distributed uniformly between (a,b) a = 75 seconds b = 125 seconds
Wait for Response
Response arrives at time T' (T' ≥ T)
Response
T' < T + ΔT
Y
N
Wait till Clock = T + ΔT

Figure 2. Behavior of a terminal

In the first case, the intermessage time would be ΔT; in the second case, it would be T'-T. Note that T'-T is the response time for the transaction.

Before each measurement run begins, the TPNS system is generated to simulate a fixed number of terminals. The times at which the terminals send their first messages are randomized. Hence, the times at which a terminal sends a transaction (message) to the system are independent of the times at which other terminals send messages to the system.

In a certain time interval, t, if M_t is the total number of transactions received by the host system, then we define the average transaction rate (λ_t) during the interval t as M_t/t. One condition that the system must satisfy in steady state is that λ_t should be independent of t for large values of t and should approach λ, the steady state transaction rate.

The mathematical model discussed here deals with deriving the steady state distribution of M_t. M_t is a random variable.

The distribution of M_t leads us to specify the lower and upper limits for M_t. The lower and upper limits of M_t are used to check whether a given measurement run is in steady state. We postulate that if a system is in steady state, then M_t will always be between the lower and upper limits for M_t given by the mathematical model.

3. MATHEMATICAL MODEL

N = No. of terminals

M_t = No. of transactions sent by all terminals in time t

D = Intermessage time, a random variable distributed uniformly between (a,b)

$$\mu = \text{mean of } D = \frac{a+b}{2}$$

$$\sigma^2 = \text{variance of } D = \frac{(b-a)^2}{12}$$

The mathematical model derives the distribution of M_t. N and D are specified for a given measurement.

M_t is the sum of N independent and identically distributed random variables Θ_t, where Θ_t is the number of transactions sent by a single terminal in time t.

3.1 Derivation of the Distribution of Θ_t

The intermessage time, D, has a minimum value of (a), where (a) is 75 seconds. For the system at hand, the response time was much smaller than 75 seconds. Hence, the time between two consecutive messages from a terminal is D. Since the intermessage times are independent, identically distributed, random variables, all with distribution U(a,b), the process of sending messages from a terminal to the host system is an ordinary renewal process [5]. Θ_t is the number of renewals in time t.

From [5], Θ_t is asymptotically normally distributed with mean t/μ and variance $\sigma^2 t/\mu^3$.

It should be noted that the fact that the response time is always less than intermessage time is critical for Θ_t to be a renewal process.

3.2 Derivation of Distribution of M_t

Let

$$\mu_1 = t/\mu \text{ and } \sigma_1^2 = \frac{\sigma^2 t}{\mu^3}$$

M_t = The sum of N independent, identically distributed, random variables (iidrv) each with distribution Θ_t.

The distribution of Θ_t is normal (μ_1, σ_1^2) and the sum of N iidrv with normal distribution (μ_1, σ_1^2) is a normal distribution [6] with mean $N\mu_1$, and variance $N\sigma_1^2$. Hence, M_t is distributed normally with mean μ_2 and variance σ_2^2 where

$$\mu_2 = N\mu_1 \text{ and } \sigma_2^2 = N\sigma_1^2$$

Substituting for μ_1 and σ_1^2 we get:

$$\mu_2 = \frac{Nt}{\mu} \text{ and } \sigma_2^2 = \frac{Nt\sigma^2}{\mu^3}$$

3.3 Derivation of the Limits for M_t

From the properties of the normal distribution [7] we get:

Probability $(\mu_2 - 3\sigma_2 \leq M_t \leq \mu_2 + 3\sigma_2) = 0.9974$.

We define the upper limit and the lower limit for M_t in such a way that the probability that M_t would be outside these limits is extremely small; that is, one would be 99.74% confident that M_t would be between these limits.

Let:

LL = lower limit for M_t

UL = upper limit for M_t

then:

$$LL = \mu_2 - 3\sigma_2$$

$$UL = \mu_2 + 3\sigma_2$$

The interval (LL, UL) is also known as the 99.74% confidence interval.

In summary, for an N-terminal measurement run, the number of transactions in t seconds in steady state should be between LL and UL. Note that LL and UL are functions of N, t, a, and b.

4. PRACTICAL CONSIDERATIONS

The model constructed above is for a system where the intermessage delay is uniformly distributed between (a,b). Since the behavior of the terminals is simulated by generating random numbers uniformly between (a,b), the actual numbers may not have the exact characteristics of a uniform distribution. The measurement intervals are small, about thirty minutes long, and a typical terminal sends about eighteen messages in this interval. Hence, no rigorous tests could be made to check the real distribution of the intermessage time. Moreover, even if the distribution of intermessage times could be determined in one case, we cannot be sure it will hold for another case. Hence, it is clear that some method is required to take into account the deviation of the actual measurement from a theoretical specification.

A number of measurements were made to estimate the true distribution of the intermessage time. The parameters of interest were the mean and the variance of the intermessage time. It was found that the mean of the intermessage time was 101 seconds, a value very close to the theoretical 100 seconds. The mean intermessage time did vary from run to run, but the ratio of variance to mean square (σ^2/μ^2) stayed constant and close to its theoretical value of 1/48. It can be easily verified that for a uniform distribution from 75 to 125 seconds, the ratio of variance to mean square is 1/48.

Another practical consideration is to determine the value of t, the duration of a measurement interval. The renewal formula holds for large values of t, since it is the asymptotic distribution.

To determine the value of t that we could use, two hours of terminal activity was simulated where the intermessage time from each terminal was a uniform distribution between 75 and 125 seconds. The two-hour interval was broken into (120÷t) t-minute intervals. The number of transactions in each interval was compared to the upper and lower limits for M_t.

It was found that for $t \geq 10$ minutes, the number of transactions in time t was always between LL and UL. Hence, we concluded that if we let t be ten minutes, the renewal formula would hold.

Hence, to adapt the model to practice, we decided to divide a measurement run into K intervals, each of length ten minutes. For example, a thirty-minute run gives three ten-minute intervals, and K would be 3. We also note that:

$$\mu_2 = \frac{Nt}{\mu}$$

= mean total transactions in a t-second interval

$$\sigma_2{}^2 = \frac{Nt}{\mu}\left(\frac{\sigma^2}{\mu^2}\right) = \mu_2 \frac{\sigma^2}{\mu^2}$$

= variance of the number of transactions from N terminals in time t.

But, earlier it was mentioned that σ^2/μ^2 is 1/48 for the system. Hence,

$$\sigma_2{}^2 = \frac{Nt}{\mu} \; \frac{1}{48}$$

$$= \frac{\mu_2}{48}$$

From this and the formulae for LL and UL, we get:

$$LL = \mu_2 - 3\sqrt{\frac{\mu_2}{48}}$$

$$UL = \mu_2 + 3\sqrt{\frac{\mu_2}{48}}$$

The theoretical value of μ_2 is Nt/μ. But adjustments should be made to take into consideration the variation in μ from run to run. To do that, a run is divided into several intervals each of length t. We can estimate $\mu 2$ by $\overline{M}$, where $\overline{M}$ is the mean number of transactions in time t. $\overline{M}$ is given by:

$$\overline{M} = \frac{\text{Total transactions received in the run time}}{\text{number of intervals of length t}}$$

The lower and upper limits for the number of transactions in time t are:

$$LL = \overline{M} - 3\sqrt{\frac{\overline{M}}{48}}$$

$$UL = \overline{M} + 3\sqrt{\frac{\overline{M}}{48}}$$

To calculate $\overline{M}$, the run should have three or more intervals of length t. Since the minimum value of t is ten minutes, the measurement should be thirty minutes or longer.

The method of detecting whether a measurement was in a steady state is as follows:

1. The measurement is divided into K intervals each of length ten minutes. The intervals are non-overlapping.

2. The number of transactions received by the system in interval i is, say, M_i for i=1, 2, ...K. This statistic is available from a message log tape that TPNS produces.

3. Calculate $\overline{M}$. $\overline{M}$ is given by $\frac{\Sigma M_i}{K}$.

4. Calculate LL, UL.

$$LL = \overline{M} - 3\sqrt{\frac{\overline{M}}{48}}$$

$$UL = \overline{M} + 3\sqrt{\frac{\overline{M}}{48}}$$

5. If $UL \geq M_i \geq LL$ for all i, then the measurement run was in steady state. Otherwise, the run was not in a steady state.

6. The mean transaction rate for the run in steady state is $\overline{M}$ transactions per ten minutes.

5. APPLICATION TO REAL MEASUREMENTS

The theory developed here was used to control the performance measurement runs for IMS/VS system running under MVS on an IBM system/370 Model 158. The results of four independent measurement runs are shown here to illustrate the application of the theory. This section contains no performance analysis of IMS/VS running under MVS. No conclusions can be drawn on the basis of the four runs mentioned here regarding the true transaction rate (number of transactions sent to the system for processing per unit time) that any real IMS/VS system can sustain. Transaction rate was measured only for the particular IMS/VS system under study. See Table I for the data collected during the measurement runs.

The four runs represent four combinations of number of terminals and main storage size. Each run was thirty minutes long. There was a fifteen-minute warm-up period before the thirty-minute measurement interval. The warm-up time was required to start the required number of terminals. No data was collected during the warm-up period. The TPNS tape was processed to determine the number of transactions sent by the terminals to the IMS/VS system in each ten-minute interval of a run. Using the formulae developed in the previous section, the values of $\bar{M}$,UL, and LL were calculated for each run. A check was made to determine if $LL \leq M_i \leq UL$ for each interval i for each run.

From Table I it follows that for runs 1, 2, and 3 the conditions for steady state were satisfied. In run 4, the analysis showed that the system was not in a steady state.

Table I. Results of Measurements

RUNID	1	2	3	4
No. of terminals	152	320	384	438
Memory (MB)	3	4	4	3
Trans./10 min. (M_i)				
Interval 1	902	1916	2282	2531*
Interval 2	904	1890	2264	2486*
Interval 3	905	1902	2289	2222*
Average ($\bar{M}$)	904	1903	2278	2413
UL	917	1922	2299	2434
LL	891	1884	2257	2392
Conclusion	Steady state	Steady state	Steady state	Not steady state
Steady state trans. rate (per sec.)	1.506	3.172	3.797	----

* The interval where the number of transactions per ten minutes is not between UL and LL.

It should be noted that, besides checking whether a run is in steady state or not, one must check other statistics that determine the validity of the run. One such parameter is the frequency of each transaction type.

Having determined that a run was not in steady state, we must determine why the run did not reach a steady state. There could be many reasons, such as:

1. Telecommunication line failure.

2. Number of terminals too high for the particular system to handle. The transaction rate is initially high but starts falling as queues develop.

3. Errors in measurement procedures.

4. Hardware or software failure.

If these causes can be eliminated in a particular case, then we may conclude that the system did not reach a steady state because of an imbalance between the workload (number of terminals) and the system capacity. In this case, the run should be further analyzed to determine the cause of transaction queue build-up. Tuning the system may help alleviate the problem in this case.

When we analyze a measurement run that is of finite time duration, we can reach one of the following conclusions: the transaction rate was steady during the run, or the transaction rate was not steady during the run. In the latter case, if a fixed number of terminals are interacting with the system, then theoretically the queues can take a finite number of values. It is therefore possible that if the measurement was continued longer, a steady state might eventually be reached. It is also possible that the transaction rate may keep oscillating. A longer measurement may help determine the true behavior of the system in this case.

6. PROJECTING SYSTEM TRANSACTION RATE

If the system is in steady state during the measurement interval, we can state its throughput in terms of the observed steady state mean transaction rate. The question is, how sure are we that the system can sustain the transaction rate indefinitely? The question can be answered only in a probabilistic way. The measurement is of finite duration, and only a finite set of load conditions can be observed. There is a probability that, at some point in time beyond the measurement interval, queues could develop in the system that cause the transaction rate to depart from the steady state value. The measured (steady state) transaction rate is an estimate of the true system transaction rate. We would like to know how reliable this estimate is.

Suppose there is a certain probability 'p' that the measured system would have an interval where the transaction rate is outside the steady state limits. We would like to estimate this probability. If 'p' is close to 0, then the measured transaction rate is a very reliable indicator of the true system transaction rate for the measured environment. If 'p' is close to 1, then the measured transaction rate is not a reliable indicator of the true system transaction rate.

One possible approach to estimating 'p' is as follows. Let the number of intervals measured be 'k'. The measurement indicated that the system was in a steady state in all of the 'k' intervals. 'p' is the probability that an interval will be found where the transaction rate is not between the specified limits (UL, LL). 'p' is the unknown that we must determine. The problem is analogous to 'k' Bernoulli trials [7] in which the number of successes is zero. Let 'E' be the event that the system is in steady state. P(E) is the probability of the event 'E'. Then,

$P(E) = (1-p)^K$

= Probability that transaction rate was within specified limits in all 'K' intervals.

Let E' be the event that in one or more intervals the transaction rate is outside the limits of steady state.

$P(E') = 1-P(E)$

$= 1-(1-p)^K$

If $P(E') \geq 0.95$, then we can be at least 95% confident that E' would occur.

There is a value pc, $0 \leq pc \leq 1$ such that:

$1-(1-pc)^K = 0.95$

For $p > pc$, $P(E') \geq 0.95$ and hence, E' must occur. Since, in the real measurement E' did not occur, we reject the hypothesis (with 95% confidence of being right) that $p > pc$.

'pc' is given by the following equation:

$pc = 1-(1-0.95)^{1/K}$

$= 1-(0.05)^{1/K}$

We will call (0,pc) the 95% confidence interval for 'p'. 'pc' is the maximum probability that the system would not sustain the measured transaction rate in an interval.

For our system, an interval was ten minutes long. Table II shows the values of pc as a function of the number of intervals and the total measurement run time.

Table II. Maximum Probability of Not Sustaining the Measured Transaction Rate

No. of Intervals	Run Time (Minutes)	Maximum Probability (pc)
1	10	0.95
2	20	0.78
3	30	0.63
4	40	0.53
5	50	0.45
6	60	0.39
9	90	0.28
15	150	0.18
18	180	0.15
36	360	0.08
60	600	0.05

The values of pc have been plotted in Figure 3. Using Table II and figure 3, one can select the run time that guarantees a certain reliability in predicting the true system throughput. For example, to ensure that we are 80% confident that the measured transaction rate represents the true transaction rate, the run should be 150 minutes long. Figure 3 also shows that the reliability of predicting true system throughput dramatically improves initially with increase in run time. Beyond thirty intervals (300 minutes) the improvement is very slow.

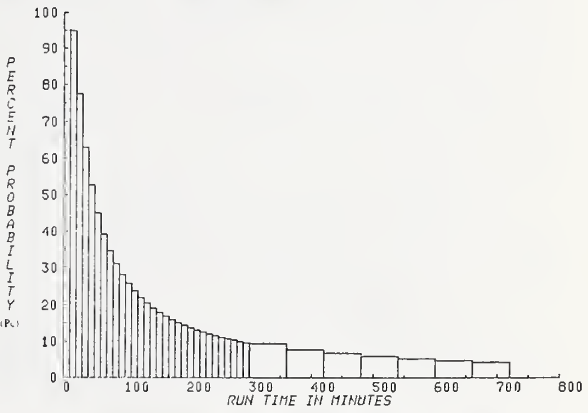


Figure 3. Maximum probability that an interval would be in non-steady state vs. run time

7. SUMMARY

We have presented an analytic method for determining whether a performance measurement run for an interactive computer system is in steady state. The method was successfully used to control performance measurement runs for an IMS/VS system. The question of projecting whether a certain transaction rate can be sustained indefinitely based on steady state measurements was also looked at. We derived a relationship between the number of intervals measured and the 95% confidence estimate of the maximum probability that the system cannot sustain the throughput in some interval beyond the measured intervals. An important result was to be able to select the measurement run duration needed to ensure a certain reliability of estimating the true transaction rate for the measured environment.

The techniques described here can be applied to control measurement runs of an interactive computer system. Prior to the availability of the technique, one had to use graphical techniques, making subjective decisions based on the graph to determine if steady state was achieved. We hope the analytical techniques will provide an alternative method that can be used in many practical measurements.

References

[1] B. Hyman, *Stability and Workload Definition for Time Sharing Systems,* Presentation to Federal Information Processing Standards Coordinating and Advisory Committee, Task Group 13, July 30, 1975.

[2] Thomas F. Wyrick, *Concepts and Issues Relevant to the Use of Remote Terminal Emulation in Teleprocessing Procurements,* (Federal Computer Performance Evaluation and Simulation Center, Washington, D.C., May 1977), p. 46.

[3] G. Gordon, *System Simulation,* (Prentice-Hall, 1969), Chapter 15.

[4] Teleprocessing Network Simulator (TPNS), Program Product (5740-XT4), IBM Corporation, White Plains, N.Y.

[5] D. R. Cox, *Renewal Theory,* (Methuen and Co., London, 1962), p. 40.

[6] W. Feller, *An Introduction to Probability Theory and Its Applications,* Vol. 2, (John Wiley and Sons, 1966), p. 46.

[7] B. W. Lindgren and G. W. McElrath, *Introduction to Probability and Statistics,* (The Macmillan Company, N.Y., 1959), p. 253.

CAPTURING TERMINAL TRAFFIC USING A HARDWARE MONITOR

Thomas M. Marter

Computer Sciences Corporation
400 Army-Navy Drive
Arlington, VA 22202

This paper presents a detailed technical discussion of the HIS 6000 hardware monitoring project performed under subtasks 3 and 5 of task 398 of Contract Number DCA 100-74-C-0002. This paper presents a statement of the problem, a conceptual overview of the DATANET 355 (and specifically the HSLA), the equipment selected, the probe points used and their development, and the overall measurement strategy for the terminal data capturing effort. Some additional discussion is presented on direction for continuing work in this area.

Key words: Data input bus; data output bus; DATANET 355; high-speed line adaptor; HIS 6080; interactive terminal; probe point development; programmable monitor; stimulator; terminal data.

1. Background

The Command and Control Technical Center (CCTC)[1] at the Pentagon is a field activity of the Defense Communications Agency (DCA) and provides technical assistance in support of the National Military Command System (NMCS). Under the guidelines of the World-wide Military Command and Control System (WWMCCS), the CCTC has undertaken a conversion from CDC and IBM mainframes to Honeywell (HIS) systems. Until 1975, all Computer Performance Evaluation (CPE) efforts were directed at the IBM/360s at the CCTC, with only nonrecurring projects performed on the HIS system. It became apparent at that time that an evaluation of the performance of both the current and planned WWMCCS configurations at the CCTC was required.

It had been established that the CCTC management was faced with three major problems:

a. Configuration Variability - The system at that time was a dual-processor HIS 6080. It would become a triplex system by 1 July 1975 and was projected to become a quadruplet system in FY77.

b. System Behavior - The mix of batch and interactive workloads, especially in terms of the peak and fluctuating interactive demands impact on the system, had to be examined.

c. DATANET 355 Behavior - A greatly expanded terminal network was planned for the projected system and its impact on that system was not known; it was also not known if the DATANET 355 front ends would prove to be a bottleneck on that system.

The emphasis of all planning towards the CPE effort would be focused on the correction of these major problems, and a joint effort was initiated between the CCTC, Computer Sciences Corporation (CSC) and the Federal Computer Performance Evaluation and Simulation

[1]Formerly known as the National Military Command Systems Support Center (NMCSSC).

Center (FEDSIM), with all hardware monitoring support being the responsibility of CSC from probe point development through data reduction.

The host system in question was the Force Control HIS 6080 system at the CCTC, The Pentagon, Washington, D.C. This system, as shown in figure 1, was a triplex 6080 consisting of three Central Processor Units (CPUs), four Storage Control Units (SCUs) with 576K of memory, three Input/Output Multiplexors (IOMs) with 52 active channels, and three DATANET 355s acting as front-end processors supporting 42 terminals plus remote line printers and data links. The operating system in use was WWMCCS/GCOS 6.2 with approximately 1,000 batch jobs a day. Terminals operated under TSS, under the direct program access (DAC) mode of GCOS, or in some cases under the Worldwide Data Management System (WWDMS). The Transaction Processing System (TPS) was not used.

2. Objectives

From a hardware monitoring point of view, there were two basic objectives to the CPE effort. The first was to generate a resource utilization profile of the host system. The second objective was to capture all data passing through the High-Speed Line Adaptor (HSLA) of the DATANET 355s. This is something that had never been done, and so there was little assurance that it could be done. Even those who proposed the idea had grave doubts about its feasibility, but it was selected because it was the best if not the most practical alternative. The necessity of capturing these data is treated in the FEDSIM working paper "Phase I-NMCSSC Network Study Technical Development Plan," and the reader is referred to it for a detailed discussion. For the purposes of this paper, the following discussion will suffice.

The acceptance of simulation of a proposed system configuration for acceptability is not questioned; the results of the simulation will reflect whether it was performed well or badly. To simulate a batch/interactive system, two ingredients are necessary: a synthetic job stream to simulate the batch workload, and a terminal emulator to simulate the interactive workload. Successful capturing of the terminal data would ultimately determine whether or not the terminal emulation would reflect the real world. In actuality, a stimulator[2] was chosen as opposed to an emulator, but the fact remained that certain metrics about the interactive terminals were required for the proposed study. These necessary metrics included mean input message length, mean output message length, expected interaction time, and expected time between interaction. All of these were needed for each channel. In addition, a command usage distribution was required.

There are several ways to obtain this information. One is to attempt to interview all users and get the information from them. Even if all users could be contacted immediately after their terminal session, the reliability of the data could be questioned. At the CCTC, the number of users is large, and the ability of a user to remember all he did during a session is questionable. Additionally, there is no way that response times and message lengths can be measured using this method.

Another alternative is observation, i.e., watching over the shoulder of the user. However, it has been shown that being observed has significantly altered the think time of users. Some begin making mistakes because they are being watched, while others alter their behavior in order to make no mistakes. The observers, trying to keep track of message lengths and timing interactions, alone can distract and cause the user to alter his behavior.

The normal solution is to give each user a form or questionnaire to complete during and after his terminal session. This is acceptable if the user is not distracted by the questions, if the user is conscientious about answering, and if no subjective questions are used, such as "How was response time?" This method can work if the user keeps track of all his message lengths and interaction times, but if he does, his behavior must be altered, and therefore the experiment is not valid.

The solution, therefore, if it can be done, is to capture all terminal traffic for prime shifts for a given number of days. These data can then be submitted to data reduction software, which can obtain the required metrics and command distributions.

3. Discussion

It was known that the HSLA of the DATANET 355 operates at a burst mode of 50,000 bits/sec. It transfers data as eight-bit parallel modified ASCII with parity. There are two physically separate data paths for input and output, and up to 32 subchannels

[2]See "Phase I-NMCSSC Network Study Technical Development Plan" for a discussion on the selection of a stimulator.

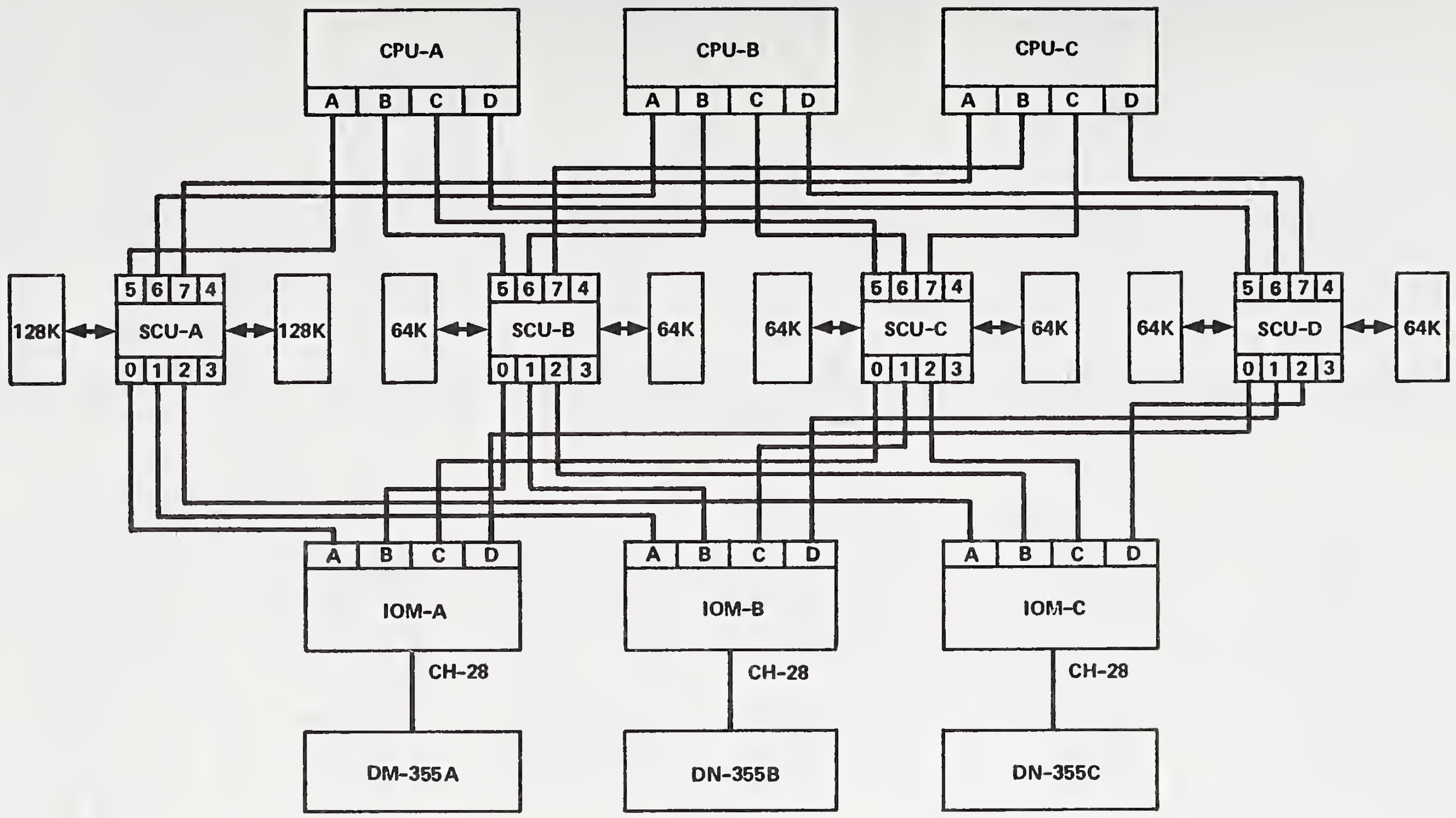


Figure 1. Force Control HIS 6080 System

can be serviced. Two eight-bit paths had to be captured (parity was ignored) and some means had to be found to identify the source or target of the data, i.e., subchannel. If the subchannel number could be appended to the data, then two 13-bit[3] data paths had to be captured. Also, it was not known if any vendor store function software would perform the required function, so a monitor which could be programmed was required.

3.1 Equipment Selection

Although the speed of the data handler and minicomputer software were important, a critical factor was tape speed. The DYNAPROBE-8000's 25 ips/1600 bpi/9-track drive is not unique, but its "hot write" capability is. It does not have to stop in order to write an IRG but can write IRG "on the fly." The expected data rate made this highly desirable. In addition, the D-8000 allowed two 16-bit and one 14-bit store operations. The monitor was built around a PDP-11, a minicomputer very familiar to the two key analysts involved. If reprogramming the minicomputer would become necessary, this would be an important factor. These reasons and the economics of dealing with one vendor and one measurement philosophy justified the choice of the DYNAPROBE-8000. The D-8000, with its PDP-11 and KSR-33, was selected along with its own 1600 bpi/9-track tape drive. The 1600-bpi tape drive was chosen for two reasons: (1) twice as much data could be collected on one tape, and (2) the data reduction would not have to be done on the host system since an IBM 360/67 was available, meaning that the OS/360 software could be run on a virtual OS machine under CP-67.

Additionally, the DYNAPROBE monitor family was chosen because of the single/quad probe and quad concentrator philosophy. This setup has always proven itself to be very convenient and efficient.

3.2 Strategy - Interactive Terminal Data Capturing

The interactive terminal data capturing strategy was not straightforward; in fact, it was nonexistent. As far as could be determined, no one had used a hardware monitor to capture all data passing through a front-end processor to a system mainframe. A parallel front-end was once used to capture all data on an airlines reservation system where a lockout loop, which was believed to be input data dependent[4], was occurring, but this approach was not feasible at the CCTC. No DATANET 355 was available and parallel communications lines were impossible.

The DATANET 355s at the CCTC are used as front-end network processors and communication subsystems to the HIS 6000. The DATANET 355 is a programmable stored program processor and interfaces to the HIS 6000 either through the SCM or IOM. The input/output structure is bus oriented. Up to 16 channels can be provided for on the I/O bus with a data transfer rate of 500,000 words (up to 32 bits) per second. Trying to capture the needed data at this bus would be an impossible task, and determining which data were terminal data would be formidable if not impossible.

The DATANET 355 configuration shown in figure 2 reveals that all high speed subchannels (and therefore all interactive devices) are connected to the DATANET 355 through the HSLA. The HSLA provides connections for up to 32 lines (subchannels) at speeds of 75 to 50,000 bps. This speed seems much more feasible for data capture and the only alternative closer to the data is the subchannel itself. Study of the subchannel showed that capturing the data at that point would be simple; however, a single monitor would be required for each of the three subchannels[5] enabled on the three DATANET 355s. This approach was not economically feasible. The solution had to be within the HSLA.

Each DATANET 355 at the CCTC is configured with only one HSLA, which supports 32 subchannels. The HSLA provides an interface between the subchannel units (8 bits wide) and the DATANET 355s I/O bus (32 bits wide) that is 32 bits wide, though the actual data flow is only 8 bits wide.

The problem for the analyst was to break down the HSLA, find the eight bit path(s), and determine the best place to capture it. At first the only sources of information available were the device itself, a Honeywell Hardware Manual, "DATANET 355 Front-End Network Processor," Number BS03, Rev. 1,

[3]Eight data bits plus 5 subchannel number bits. Five bits are needed to represent 32 subchannel numbers (0-31 or 2^5).

[4]The author could not locate the original source. The results and solution are now known.

[5]Assuming only traffic on the lines and a strobe of the LOGICAL AND of all the lines, then 8 bits in plus 8 bits out equals 16 bits. A single D-8000 can accept 48 inputs.

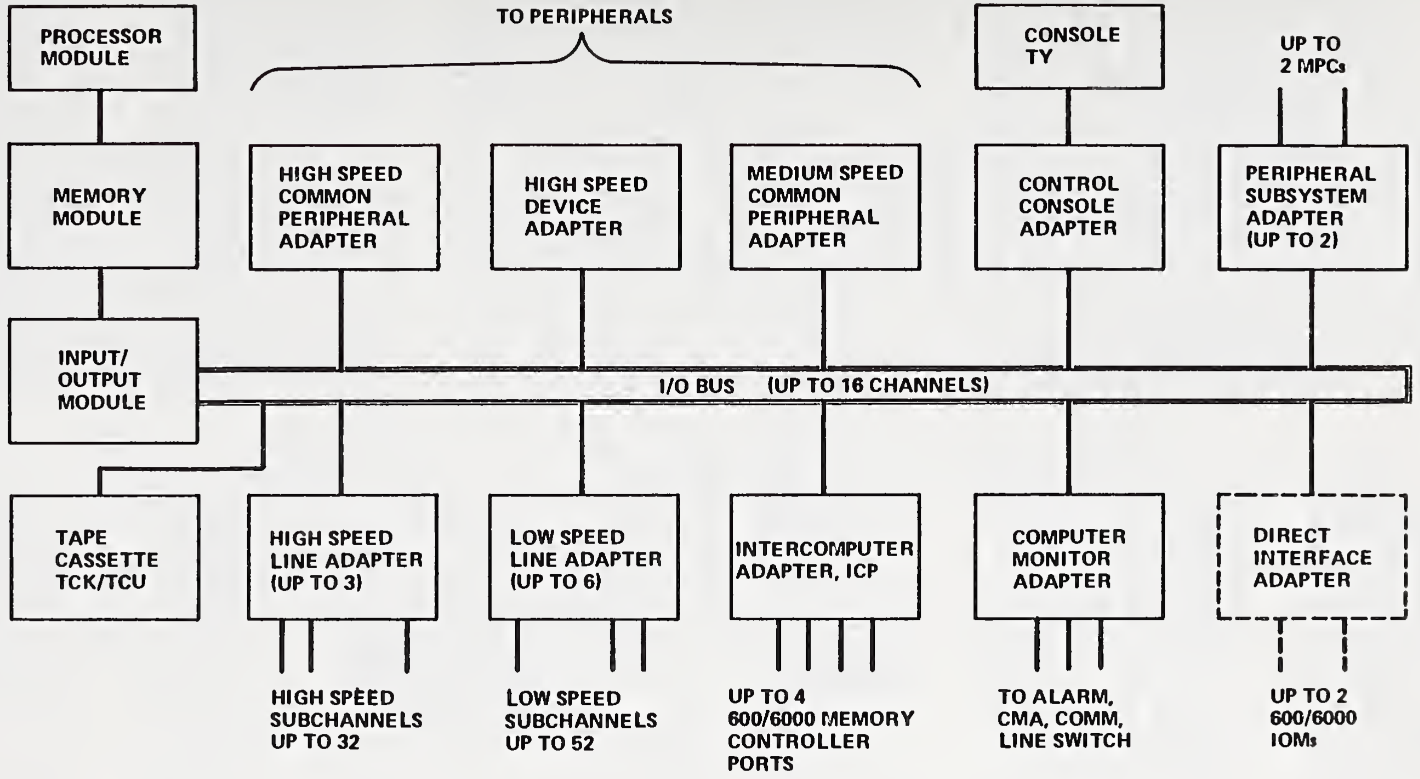

PROCESSOR MODULE
TO PERIPHERALS
CONSOLE TY
UP TO 2 MPCs
MEMORY MODULE
HIGH SPEED COMMON PERIPHERAL ADAPTER
HIGH SPEED DEVICE ADAPTER
MEDIUM SPEED COMMON PERIPHERAL ADAPTER
CONTROL CONSOLE ADAPTER
PERIPHERAL SUBSYSTEM ADAPTER (UP TO 2)
INPUT/ OUTPUT MODULE
I/O BUS (UP TO 16 CHANNELS)
TAPE CASSETTE TCK/TCU
HIGH SPEED LINE ADAPTER (UP TO 3)
LOW SPEED LINE ADAPTER (UP TO 6)
INTERCOMPUTER ADAPTER, ICP
COMPUTER MONITOR ADAPTER
DIRECT INTERFACE ADAPTER
HIGH SPEED SUBCHANNELS UP TO 32
LOW SPEED SUBCHANNELS UP TO 52
UP TO 4 600/6000 MEMORY CONTROLLER PORTS
TO ALARM, CMA, COMM, LINE SWITCH
UP TO 2 600/6000 IOMs

Figure 2. DATANET 355 FNP Configuration

December 1973, and the DATANET 355 logic diagrams. After studying these sources for some time, no immediately obvious way to capture the data presented itself. All internal registers and busses were 32 bits wide and, although strobes drove these bits in ordered groups, the function of the strobes and the purpose of the bit groups were not intuitively obvious.

One thing did become immediately obvious from the logic diagrams. The subchannel number, at least in part, was controlled by a scanner, or ring counter, with five output lines which were ultimately decoded to 32 separate enable functions. These five lines appeared to be the point at which the subchannel number could be captured, but the knowledge that the DATANET 355 was a multiple level interrupt driven device did not coincide with a scanner. The probe points were a beginning.

As an experiment, these probe points were connected and the output of the probes was wired directly to the display register of the D-8000. By observing this display and the subchannel number display on the maintenance panel of the HSLA, it was determined that the displays were in exact synch. Further study showed that the subchannel number could be generated in one of two ways. The first is by an interrupt of the scanner. Each of up to five levels of requests would be serviced for that subchannel without refreshment of the interrupt register, and then the scanner would continue. This is how the subchannel gets data to the HSLA. The second is when a subchannel number is obtained from a Peripheral Control Word (PCW). This is how the HSLA gets commands and data to the subchannel. A valid subchannel number was present in the Scanner Address Switch register when the strobe ¢CMDCTRL was not present. If this strobe was present, a command was present in the switch register. When successful data were finally captured, it was determined that this strobe was never present when data were being transferred, and therefore the strobe was not needed. If an analyst wished to determine which channel (device) was most active on the DATANET 355, the inverse of this strobe would be very useful.

Using the HSLA maintenance panel to verify the subchannel number brought attention to the fact that there was provision for displaying "DATA" on the panel for both an Input Display and an Output Display. If a meaningful strobe could be defined, this could prove to be a useful way to capture the data. The logic diagrams seemed to reinforce the original premise and a good signal was present which gated the lines to the light drivers. Let it suffice to say that a great deal of time was spent on this theory with absolutely no results.

At this time the Honeywell customer engineers pointed out the Product Engineering Specifications microfiche for the HSLA. The logic diagrams suddenly became clear. A functional diagram of the HSLA is shown in figure 3. The diagram shows that the common denominator between input and output is the data input switch register and the data output switch register (please note that the presence of ¢CMDCTRL at both the Scanner Address Switch and the Data Output Switch denotes the presence of a command, not data). Again the observation method was used. Probes were attached to each of the registers in turn, and the D-8000 display was compared with the maintenance panel display. They were in exact synch.

In the case of the output data switch, it can contain data, the Base Address Word, or a character control character. On the output side it can be data, a Peripheral Control Word, a Base Address Word, or a Character Control Character. Strobes still had to be defined which would indicate that data were present. Additional study and analysis of the $CON strobe (CONNECT) and $ANS strobe (ANSWER) and their families was begun. The first likely candidate was $SCANS, which was defined as "a strobe bus line which indicates to the subchannel the completion of the subchannel initiated interrupt - answer cycle and presence of data on the DOBUS if a data request was made." A strobe with a more succinct definition was necessary to fulfill the needs of the study.

Staring at a microfiche reader for long periods of time is not highly tolerable. Between microfiche sessions, the input and output switch registers were probed and meaningful output data were gleaned from the data captured, especially the output of the GCOS module VIDEO, to the operator console CRT. Input was not very obvious and was buried in reams of synch frames from the synchronous terminals (VIPs) and nondata characters, if it was there at all.

Further perusal of the Product Engineering Specifications turned up ¢SCDATI, which was defined as "a command bus line requiring the subchannel to present the received data character on the DIBUS," and ¢SCDATI which was defined as "a command bus line informing the subchannel that the requested data character is present on the DOBUS." Examination of the logic diagrams confirmed that ¢SCDATO was exactly what was needed, but ¢SCDATI did

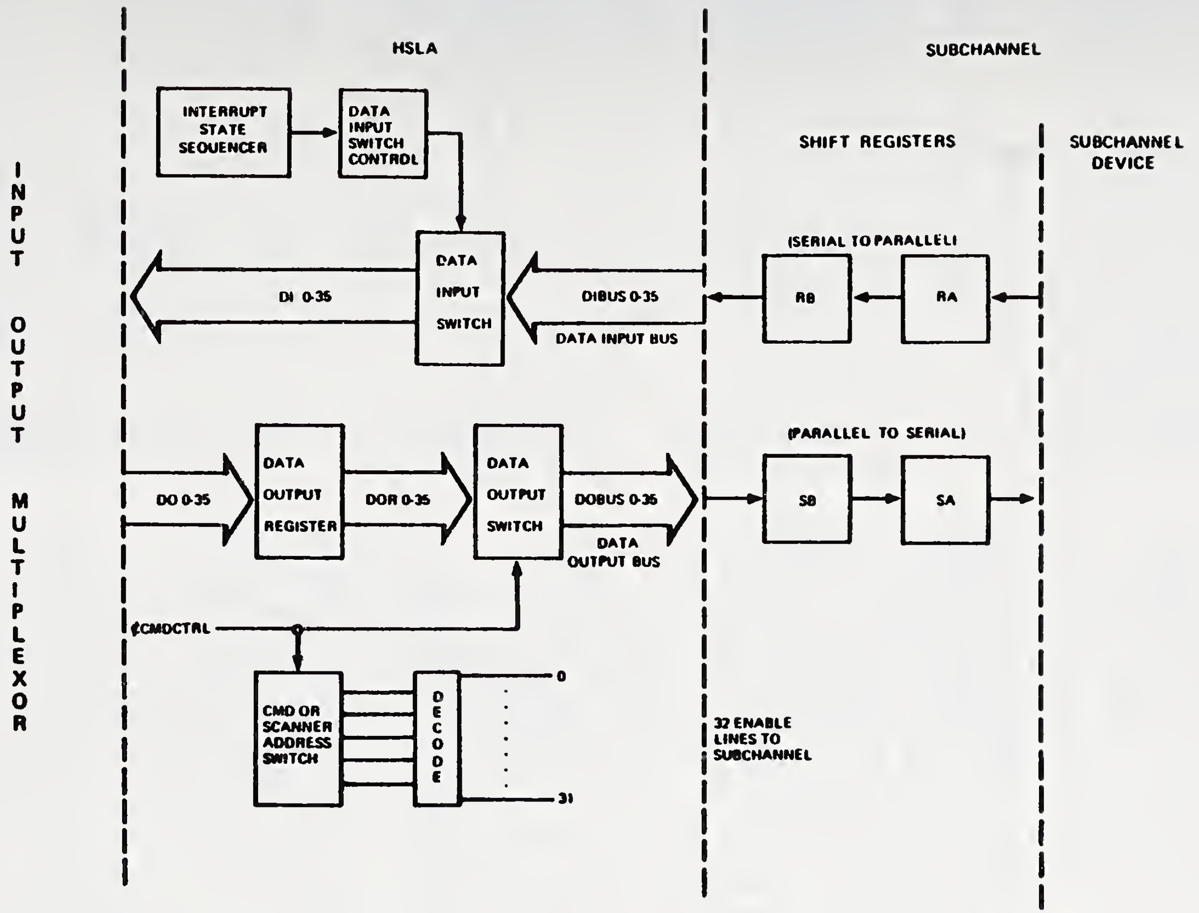


Figure 3. Functional Diagram of the HSLA

not appear to be logically correct for what was needed. The output strobe was used to capture the DOBUS and everything worked perfectly. A number of other strobes were postulated for input but none worked. Finally, ¢SCDATI was tried and, except for a tendency to append an erroneous character to the end of messages, it worked. The verification of input data was not as easy as the verification of output data due to the small volume, but test messages typed at a terminal on an almost dead machine were immediately evident.

The normal source for $CON and $ANS and their family of strobes is a nominal 100 ns positive going strobe. Whether the source of the strobe is a 100 ns strobe when online, a logic edge when offline, or a standard strobe from the recycle logic, pulse forming logic forms a standard strobe of 35 ± 5 ns for distribution. During the development of these data capturing strobes, Tektronix scopes proved to be invaluable in determining the attributes of a signal on a given pin and its attributes at the logic panel of the D-8000. A D-416 monitor was used to determine the frequency of strobes and the frequency of data captures.

3.3 Interactive Data Collection

It had now been shown that a D-8000 type of hardware monitor could capture the interactive terminal data at the HSLA of a DATANET 355. There were still a number of questions that needed to be answered, in particular, could all the data to and from all terminals on a DATANET 355 be captured during prime time.

The selection of a programmable monitor provided two options for the usage of the D-8000. The first option would be to use the monitor as is, using the vendor software to dump the data directly to tape as quickly as possible. The advantage of this option was that the data could be handled at a high expected arrival rate. The possible problem area was that the arrival rate may be too high even for this option. The second option would be to reprogram the minicomputer to massage and compress the data at the minicomputer and then write it to tape. The advantage of this option was that the amount of data is greatly reduced. Synch characters make up a large percentage of the data and can be easily eliminated. With additional programming effort, the VIDEO display to the operators CRT could be eliminated since it is a known value and easily quantified. The possible problem area would be that the arrival rate is too high to allow program execution time.

Because of strict time constraints, it was decided that direct capturing of the data was the only real option, and the system was so implemented. The D-8000 had no problems capturing all the data when attached to DATANET 355s A and B. DATANET 355 C caused some data to be lost due to the speed of the data link type devices on its subchannels. Approximately 9 million bytes of data would fill a tape in 1 to 2 hours of monitoring. Approximately 60 tapes were filled in a 10-day period. A data reduction software package called SNORE was developed by CSC to digest these tapes into the required FEDSIM metrics.

3.4 DATANET 355 Core Availability Monitoring

One metric was requested by the FEDSIM which also was something that had never been done, but because of the effort put forth on the rest of the study, and the presence of a usable alternative, its solution was allowed to wait until last. It is an interesting metric and solution, and was related to the terminal data and therefore should be documented.

The FEDSIM wanted to know dynamically how much core was available in the DATANET 355 during the prime shift. In the communications region of GRTS, word 652_8 (.CRBUF) contains the number of words of free core. Using the PEEK console command of GRTS, the analyst could periodically interrupt the DATANET 355, "peek" at word 652_8, and record the contents. This method can become boring and can be very time consuming.

The D-8000 still had some measurement capacity available in that the COMMAND register was not being used. It was decided that if a hook could be placed in GRTS so that the contents of .CRBUF could be placed where the monitor could see it when a NOP1 pulse was present, the problem could easily be solved. Analysis of the DATANET 355 showed a number of registers available to the programmer that could also be probed. It was decided that the accumulator side of the AQ register could easily be used and a patch was applied. Due to a number of other patches preexisting in GRTS, the frequency of NOP1 pulses was higher than expected. The non-valid pulses were accompanied by register contents of zero, so no invalid data were involved. The solution to the problem was to divide the strobe by 100 and therefore capture only every hundredth occurrence. By ignoring zero data during data reduction, a good core availability histogram could be produced, with sufficient sample to allow good confidence. Capturing every hundredth occurrence also prevented the COMMAND register from interfering with the terminal data capture since it has highest priority in the D-8000.

The probe points developed, their function names, logic sheet origin, and logic levels are given in table 1. The application layouts showing logic used and register assignments for the D-8000s are available upon request.

Table 1. D-8000 Probe Points

Function	Sheet	Point	Level
OUTPUT			
DOBUS28;000	D-16	D-WC14	0
DOBUS29;000	C-13	C-WG14	0
DOBUS30;000	D-5	D-WC03	0
DOBUS31;000	D-5	D-WC04	0
DOBUS32;000	D-5	D-WC05	0
DOBUS33;000	D-5	D-WC06	0
DOBUS34;000	D-5	D-WC07	0
DOBUS35;000	D-5	D-WC08	0
OUTPUT STROBE			
¢SCDATO;000	B-9	B-WD04	0
INPUT			
DIBUS28;000	D-13	D-WC15	0
DIBUS29;000	D-13	D-WC17	0
DIBUS30;000	D-14	D-WC18	0
DIBUS31;000	D-14	D-WC19	0
DIBUS32;000	D-14	D-WC20	0
DIBUS33;000	D-14	D-WD00	0
DIBUS34;000	D-14	D-WD01	0
DIBUS35;000	D-14	D-WD02	0
INPUT STROBE			
¢SCDATI;000	B-8	B-WG07	0
SUBCHANNEL No.			
ASCADD0;010	C-19	C-WH00 (2^0)	0
ASCADD1;010	C-19	C-WH02 (2^1)	0
ASCADD2;010	C-19	C-WH03 (2^2)	0
ASCADD3;010	C-19	C-WH04 (2^3)	0
ASCADD4;010	C-19	C-WH05 (2^4)	0
FREE BUFFER			
RA041P ZY0C	PJ-16	PJ-TP21 (2^{13})	+5
RA051P ZY0C	PJ-19	PJ-TP23 (2^{12})	+5
RA061P ZY0C	PJ-22	PJ-TP24 (2^{11})	+5
RA071P ZY0C	PJ-25	PJ-TP25 (2^{10})	+5
RA081P ZY0C	PJ-28	PJ-TP26 (2^9)	+5
RA091P ZY0C	PK-4	PK-TP13 (2^8)	+5
RA101P ZY0C	PK-7	PK-TP14 (2^7)	+5
RA111P ZY0C	PK-10	PK-TP15 (2^6)	+5
RA121P ZY0C	PK-13	PK-TP16 (2^5)	+5
RA131P ZY0C	PK-16	PK-TP17 (2^4)	+5
RA141P ZY0C	PK-19	PK-TP18 (2^3)	+5
RA151P ZY0C	PK-22	PK-TP19 (2^2)	+5
RA161P ZY0C	PK-25	PK-TP20 (2^1)	+5
RA171P ZY0C	PK-28	PK-TP21 (2^0)	+5
STROBE			
DNOP1P ZY01	PH-10	PH-TP01	+5

4. Results

All the data necessary to script the stimulator was obtained. The data reduction software produced reports on mean input message length, mean output message length, expected interaction length, and expected time between interaction. In addition, a TSS subsystem command usage distribution was obtained. The stimulator was scripted and various projected systems were configured at a test site and a batch load was placed on these systems using a synthetic job stream and a timesharing load was obtained from the stimulator. The synthetic job stream was obtained from a system used at the CCTC through which a data base is maintained on Statistical Collection File output. Based on activities, pertinent information is stored in a highly condensed version of the accounting data. This historical data base is then used for generation of the job streams.

The configurations, batch workloads, and timesharing workloads could be varied in different ways so that deficiencies in the projected configuration could be revealed and possible variations of the configuration could be proposed. If the configuration was not found to be deficient, then the two workloads could be varied upward in order to determine a projected life cycle for the projected system configuration.

4.1 Recommendations

If and when this experiment is undertaken again, the following would be considered advisable for the interactive data collection. Dedicate some software development time to reprogramming the minicomputer to at least ignore synch characters. Additionally, to be efficient and economical, the software should be capable of ignoring specific subchannels, e.g., VIDEO display or data link devices. Less captured data means far less data reduction software execution.

This reprogramming can take the form of rewriting the vendor supplied software or the user can write their own version of software. In either case, it is felt that the effort will be well justified. At the CCTC plans are being formulated for this effort.

4.2 Further Efforts

The Computer Sciences Corporation and the CCTC are preparing to try an experiment whereby this method of data capturing at the HSLA can be used to determine response time on the timesharing system at the CCTC. In this case, response time is defined as the total (wall clock) time that elapses between the last terminal key stroke of a user and the first character transmitted back to the terminal by the HIS 6080 through the DN-355.

Terminal messages on the HIS 6000 are terminated by an ETX (end of transmission) character and are begun by an STX (start of transmission) character. If the ETX character can be detected in the incoming data stream and the subchannel number can be captured and time-stamped, and likewise the STX character from the outgoing data stream can be detected and the subchannel number can be captured and time-stamped, then data reduction software can be used to analyze the response time.

At the CCTC this will be tried using a known scenario on a known terminal only until it can be proven to be feasible. Under these conditions, response time can be related to a particular TSS subsystem command. When used for the system, in general, only a general (average) response time can be conclusively obtained. Both the micro and macro techniques should prove highly useful.

Work for the CCTC performed by the Mitre Corporation has shown that the number of retries on the network at the CCTC are almost nonexistent on the terminal lines.

Using a D-8000 with its dual-function store-mode software, the original experiment with a single subchannel number can be implemented by specifying the same value for the monitored high and low limit when initializing the software. The value will be the 13-bit combination of subchannel number and the ETX or STX character. In addition, a buffer size of one will be defined as the store buffer. Thus each time the value is found it will be immediately written to tape and thus it will be time-stamped.

Using a D-7900, the same experiment can be done using the logic shown in figure 4. These strategies have never been used before but there is a high confidence level of success. The number of terminals monitored on the D-7900 will be defined by the amount of logic available at the plugboard. This has not been fully studied but the expected level of monitoring is eight terminals per monitor.

This Technical Paper is based on work performed under the direction of the Chief for Computer Performance Evaluation by Computer Sciences Corporation under Contract Number DCA 100-74-C-0002.

Bibliography

[1] Honeywell Information Systems, Inc., DATANET 355 Front-End Network Processor, Order Number BS03, Rev. 1, December 1973.

[2] Federal Computer Performance and Simulation Center, FEDSIM/NMCSSC Customer Agreement, FEDSIM Project NA-505-081-DCA, Washington, D.C., 30 May 1975.

[3] Federal Computer Performance and Simulation Center, NMCSSC Network Study Technical Development Plan, Working Paper for FEDSIM Project NA-505-081-DCA, Washington, D.C., October 1975.

[4] Federal Computer Performance and Simulation Center, External System Measurements, Informational Letter for FEDSIM Project Number NA-505-081-DCA, 22 December 1975.

[5] Command and Control Technical Center, Special Numeric Operations Report Executive (SNORE), Technical Report TR-109-76, Washington, D.C.

[6] Command and Control Technical Center, Application of Hardware Monitors to the Triplex HIS 6080, Technical Report Number TR 110-76, Washington, D.C., 31 October 1976.

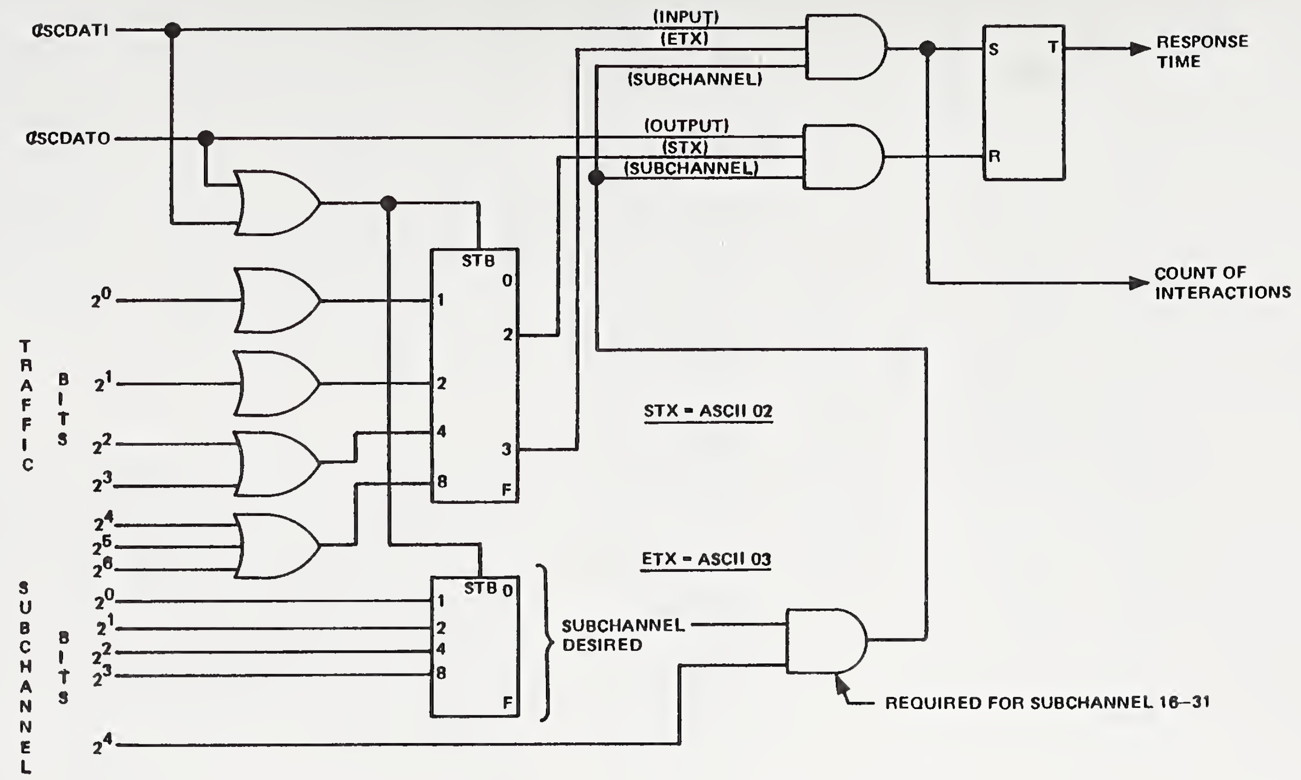


Figure 4. Terminal Response Measurement Logic

A NEW TOOL FOR MEASURING RESPONSE TIME

Dr. Gary Carlson

Director, Computer Services
Brigham Young University
Provo, Utah

Dormovil Ferreira

Director, CODEG
Goianoa, Goias, Brazil

A new software monitor was recently announced that measures the response time of up to 100 terminals for over 9 hours with only 32 seconds of CPU time and 8K bytes of core overhead. The data reported includes the minimum, maximum, average, standard deviation, and distribution of response times by terminal, by hour. Various other reports can be obtained.

1. Importance of Response Time

As on-line systems developed from batch operations, the use of interactive terminals in a teleprocessing environment has increased at an enormous rate. In fact, there are approximately 900,000 terminals installed in the United States now and perhaps as many as 1.2 million terminals installed world-wide [4][1].

Carlson, in an earlier paper [4] expressed the importance of response time as follows:

> "When we consider that there is a human sitting at each one of these terminals for 6 to 12 hours a day, the enormous room for either saving human effort or wasting it becomes readily apparent."

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

With this human-terminal interaction, it is often the terminal that can determine the effective rate at which people are working. This is similar to many man-machine interactions, but in most previous systems, like assembly lines, the machine rate is set to approach the limits of human performance and the human is asked to match the machine. With computer terminal systems it is just the opposite. We ask the human to perform at his maximum rate and hope that the machine can respond and keep up with the human. If response times are excessively slow or highly variable, the machine then impedes the performance of the human operator."

There are several things besides response time that should be measured to improve the overall performance of teleprocessing systems. This includes job turn-around time, system job holding or delay time, rate of actual transaction throughput, volume of transaction throughput, both overall and by operator, system availability, and others. A further discussion of the criteria that should be considered in measuring of remote services is given in Grubb and Cotton [9].

All of these activities need good measurement procedures, and results. We choose response time for the first major measurement effort since it seemed to be the most widely discussed by our users of all of the other measures. We were often told by our customers that response time "sure is slow today." By the time we would go over with a stop watch, things always seemed to be better. It seemed that an easy to use, reliable measure of response time could tell us where and when problems of poor service were happening.

2. Measuring Response Time

In order to reduce terminal response time we must first be able to measure it, and this can be done with a:

1) hand held stop watch;
2) hardware monitor;
3) software monitor.

For some purposes any of these three techniques may be satisfactory. However, to perform concurrent measurements of response time of all terminals on the system, stop-watch techniques are not adequate. It would be necessary to have as many people performing measurements as there are terminals on the system. For some installations this number could run into the thousands.

Measurements of response time with a hardware monitor provide the "truth" down to a microsecond range. However, the number of terminals that can be measured at any given time is limited by the number of probes and counters available in the monitor (usually 16) and by the distance of the terminals from the hardware monitor. The sensor probe that is connected to the terminal cannot be located more than 200 feet from the hardware monitor.

The National Bureau of Standards has developed a sophisticated Network Measurement Machine (NMM) that uses a minicomputer to measure the level of service at the terminal [7].

Measurement of response time by software monitors gives results that are accurate within a few hundredths of a second and are therefore adequate for most tuning efforts. However, most software monitors have given only partial, or fragmentary information.

A new software monitor called TRT (Terminal Response Time) has been developed to perform concurrent measurements of all response times of interactive terminals operating in IBM OS and VS operating systems. At present TRT only measures terminals that are in the local mode. Developments are under way to measure remote terminals. Measurements are made at the CPU by trapping on terminal I/O processing. The definition of response time used is the time interval from when the enter key is hit until the screen or typewriter starts to be rewritten. For local terminals the line and controller delays are insignificant in human terms -- at most only a few milliseconds. We anticipate essentially the same behavior with remote terminals. Some preliminary measures show a maximum line and controller delay in remote operations of only 250 milliseconds. Further information on remote operations can be found in Grubb [8]. The trapping occurs at a very fundamental system level and seems to work for all kinds of access methods and all kinds of TP control systems. The monitor is designed to measure terminal response time by frequency of occurrence and arrange the data into a number of intervals in a form convenient to analyze, and to provide the average and standard deviation. The monitor provides the following information at the end of each run:

- terminal identification;
- total number of responses (by terminal);
- average response time (by terminal);
- maximum response time (by terminal);
- minimum response time (by terminal);
- standard deviation (by terminal);
- frequency of occurrence of response times for each class interval specified by the user (by terminal);
- cumulative percent of the frequency of occurrence for

each class interval with an automatic identification of the 80% mark (by terminal).

The software monitor was designed in a modular fashion, and is structured in such a way that it runs under OS or VS. Terminals can be grouped in any desired way.

3. Response Time Distribution Curve

Many studies related to terminal response time have been conducted with the objective of determining the average response time. However, detailed analysis of data shows that the average can be misleading in studying response time because it hides a lot of information. The average alone cannot provide any information concerning response time variability which has been of great concern to terminal users [4]. Therefore, any tool designed to measure terminal response time must give the distribution of response time, as well as provide measurements of central tendency and variability.

4. Overhead

Whenever a software monitor of any kind is used, one must be careful about the possible excessive overhead on the system. Hardware monitors were used to measure all activity in the system while the new TRT software monitor was running. At first we were rather surprised and a little worried when we found that almost 500 instructions were executed for every terminal response. That is, to intercept the fact that a terminal "enter" key was hit, to measure the time, and to record the activity takes almost 500 instructions. Upon closer scrutiny, however, it was found that in the IBM OS system if you hit the "clear screen" key, there are over 6,600 instructions executed. Further analysis showed that the TRT software monitor only used 38 instructions to determine if the interrupt was caused by a terminal. If some other interrupt has occurred, processing by the software monitor ends there. This leads then to the very small overhead that often an entire 8-to 9- hour day can be monitored with less than ½ a minute of CPU time.

The other overhead consideration is core used and all software monitors have to use some. The modular design of TRT has allowed us to put all critical functions in 8K bytes of core.

5. Accuracy

The accuracy of any monitor must be carefully examined to make sure that we are measuring what is truly expected. Many measurements have been taken comparing the TRT software monitor with various hardware monitors and a hand-held stop watch. In all cases the measurements are essentially identical. The results of a typical measurement are shown in Table 1. From these results it can be seen that the average and standard deviation are essentially identical with all three techniques. In other words, this is evidence that if a person sits at a terminal with a stop watch to measure response time, he will get identical results by using the Questronics hardware monitor or the TRT software monitor. The interesting feature of the TRT software monitor is that it gets simultaneous measures of all terminals on the system and gives the distributions of response times as well.

6. Some Early Results

Continuous measurements performed on 74 interactive terminals (IBM 3277) connected to an IBM 360/65 installed at Brigham Young University show that the average number of entries per day at this installation is about 436 per terminal day, or 54 entries/terminal hour, with an average response time of less than 2 seconds. The distribution of response times for 2 sets of terminals is given in Figure 2 and 3. Computer installations differ widely in their terminal response time requirements, and the response time needed is determined by the type of teleprocessing application used by each installation [6].

7. Some Early Conclusions from the Terminal Response Time Measurements

The monitor has been in use for over a year now and some interesting and in some cases surprising concepts seem to be emerging. Some of these preliminary and tentative ideas are:

1. There is a great deal of variability among response times of terminals on the same system. Average terminal response times measured on one 360/65 OS MVT system, range from 0.9 seconds up to 27.2 seconds. This does mean that asking for the average response time of "terminals" on a system is a misleading question. The question must be more specific, asking what is the response time for a given terminal. This does vary somewhat during the day, but not nearly as much as one might expect.

2. System tuning may be a poor approach to improvement of response time. Otherwise, how could we have such high variability in response times of different terminals on the same configured system during the same time of measurement. Rather than system tuning it appears that particular programs will have to be tuned, both in terms of what they do and how they do it, as well as more careful study of the interaction between jobs.

3. The response time of terminals in the computer center is generally much better than the users' response time. This may point out that we service ourselves best. This is probably no surprise to some people, but it was to us as we were initially struck by this phenomenon. This can be shown by looking at Figure 2 which shows an average response time for Computer Services terminals of 0.57 vs. a user department shown in Figure 3, with an average response time of 1.65 seconds. Not all user departments are that bad, but these seem to be fairly typical numbers.

4. Users can and do adapt to poor response time. This adaptation is enhanced if Computer Services provide adequate "hand holding" where we help the user feel that we are interested in his well-being even though we have poor response time. Expression of sympathy and sorrow and shared frustration seems to help the user feel okay about poor response time. There is a fairly high cost in people to do this and there is a very large cost in wasted human effort waiting for response time.

5. The distributions and average response times are initially not believed by the responsible programmers. Programmers always think that response time is better than it actually is. This is probably an expression of the same phenomenon that a programmer always thinks he can write a program in at least half of the time it will actually take. As response time measurements are gathered you need to be prepared for disbelief by the programmers who wrote the programs.

6. The terminal keyboard operators' performance for the same function can vary in the number of entries from one operator to another by 50%. This means that operators on identical terminals, using the same program and doing the same function put out significantly differing amounts of work. This should come as no surprise, but the differences seemed larger than we expected. It does raise a question if keyboard operators should have an incentive plan or piece rate similar to other kinds of production workers.

These are some of the preliminary insights. These came rather quickly and easily. We are optimistic that further insights can be obtained which can help Computer Services do a better job in terms of truly serving the user.

8. Future of Response Time Measurements

The development and installation of the terminal response time monitor has been very encouraging. The next developments will be aimed at developing a cookbook procedure on how to improve response time. Once the data on what is actually occurring is available, then specific steps can be taken to improve the response time. This cookbook approach will include such things as balancing channel activities, careful distribution of disk files to minimize disk and channel contention, and specific techniques on improving the applications programs so that they use less computing resources or at least use the resources in less competition with the other system activities. This list will grow as experience is gained in making changes that do bring about improvements in response time.

The next step planned after this is to make this cookbook part of the automatic "diagnostics" that the monitor could provide. In other words, we hope that the terminal response time monitor can be run to measure the response time and then highlight the terminals that most need attention and what should be done to try to improve the response time.

The third stage for the future is to then incorporate the automatic cookbook procedures into program fixes so that the response times will be automatically improved by automatic changes of programs, data set allocations, and other systems tuning that can bring about improvement of response time. We realize this ultimate goal is optimistic and may take a while to get there, but we are confident that the tools now are in hand to make it possible.

References

[1] Carlson, Gary. "What Causes Slow Response Time on the DEC-10 and the IBM-360 - A Use of Factor Analysis." Proceedings of Computer Science and Statistics Eighth Symposium on the Interface, UCLA , February 13-14, 1975, p. 466.

[2] Carlson, Gary. "A User's View of Hardware Performance Monitors or How to Get More Computer for Your Dollar," Proceedings of the IFIP Congress 71. August 23-28, 1971, Ljubljana, Yugoslavia. TA-5, p. 128-132.

[3] Carlson, Gary. "Practical Economics of Computer Monitoring," in Proceedings of IFIP Administrative Data Processing Conference, Amsterdam, Holland, May 2-5, 1972. Published in Management Informatics, Vol. 1, No. 6, p. 251-256.

[4] Carlson, Gary. "Measuring Response Time of Interactive Terminals," EDP Performance Review, Vol. 3, No. 8, August 1975, p. 1-5.

[5] Martin, James."Telecommunications and the Computer." IBM Systems Research Institute, Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1976, p. 71-73.

[6] Chang, J. H. "Terminal Response Times in Data Communications Systems," IBM Journal of Research and Development, Vol. 19, No. 3, May 1975, p. 272-282.

[7] Rosenthal, Robert, Rippy, Don E., and Wood, Helen M. "The Network Measurement Machine--A Data Collection Device for Measuring the Performance and Utilization of Computer Networks," NBS publication TN-912, April 1976.

[8] Grubb, Dana S. "Data Communications System Throughput Performance Using High Speed Terminals on the Dial Telephone Network," NBS publication TN-779, May 1973.

[9] Grubb, Dana S. and Cotton, Ira W. "Criteria for the Performance Evaluation of Data Communications Services for Computer Networks," NBS publication TN-882, September 1975.

Response Number	Stop Watch	Questronics	TRT Software Monitor
1	1.20	0.90	
2	2.00	1.60	
3	2.70	2.40	
4	3.80	3.40	
5	1.30	0.80	
6	2.00	1.60	
7	2.80	2.40	
8	3.80	3.40	See Figure 1 for distribution of response times.
9	1.10	0.70	
10	1.40	1.10	
11	1.80	1.30	
12	2.00	1.60	TRT does not give each individual response time.
13	2.50	2.20	
14	3.00	2.60	
15	3.50	3.10	
16	4.10	3.70	
17	4.90	4.50	
18	5.70	5.30	
19	1.40	1.10	
20	2.20	1.80	
Average	2.66	2.28	2.31
Minimum RT	1.10	0.70	0.79
Maximum RT	5.70	5.3	5.28
Std. Dev.	1.29	1.28	1.22

Table 1. Comparative Measurements of Response Time

```
                                              ******

                              **********************************************************
                              ***                                                    ***
                              ***   T E R M I N A L    R E S P O N S E    T I M E    ***
                              ***                                                    ***
                              ***          S O F T W A R E    M O N I T O R          ***
                              ***                                                    ***
                              **********************************************************


*********  RUN STARTED AT  22.18.39 SATURDAY  19 FEB 77  ****************  RUN STOPPED AT  22.28.39 SATURDAY  19 FEB 77  *********

*********  SAMPLING PERIOD NUMBER   1 -     0 HOURS AND   .10 MINUTES - FROM 22.18.39. TO 22.28.39

                                                                         NUMBER OF RESPONSES FOR EACH TIME INCREMENT (INCREMENTS IN SECONDS)

TERMINAL TERMINAL  TOTAL  AVERAGE  MINIMUM  MAXIMUM  STANDARD  0.00   0.51   1.01   1.51   2.01   2.51   3.01   3.51   4.01   4.51
ADDRESS     ID    NO.RESP  R. T.    R. T.    R. T.  DEVIATION  TO     TO     TO     TO     TO     TO     TO     TO     TO     AND
                                                               0.50   1.00   1.50   2.00   2.50   3.00   3.50   4.00   4.50   OVER

  C6C                20     2.31     0.79     5.28     1.22      0      3      3      4      3      1      3      1      0      2
                                                 CUM PERCENT     .     15     30     50     65     70     85 **  90      .    100

     23   RECORDS PRINTED
```

Figure 1. Experimental Measure of 20 Response Times

```
INTERNATIONAL COMPUTER MONITORING INC  ***                                                   ***
CORPORATE HEADQUARTERS                 ***   T E R M I N A L   R E S P O N S E    T I M E    ***
3289 MOHAWK CIRCLE                     ***                                                   ***
PROVO - UTAH                           ***         S O F T W A R E   M O N I T O R           ***
801-377-6826                           ***                                                   ***
                                       *******************************************************

*********  RUN STARTED AT   9.01.11 THURSDAY  28 JUL 77  ****************  RUN STOPPED AT  17.01.15 THURSDAY  28 JUL 77  *********

*********  SAMPLING PERIOD NUMBER   1 -   8 HOURS AND    0 MINUTES - FROM 09.01.11. TO 17.01.11

                                                                 NUMBER OF RESPONSES FOR EACH TIME INCREMENT (INCREMENTS IN SECONDS)
                                                                 -------------------------------------------------------------------
TERMINAL  TERMINAL   TOTAL   AVERAGE  MINIMUM  MAXIMUM  STANDARD  0.00   2.01   4.01   6.01   8.01  10.01  12.01  14.01  16.01  18.01
ADDRESS      ID     NO.RESP  R. T.    R. T.    R. T.    DEVIATION  TO     TO     TO     TO     TO     TO     TO     TO     TO    AND
                                                                  2.00   4.00   6.00   8.00  10.00  12.00  14.00  16.00  18.00  OVER

  04D    ADMISSIONS  1359    1.44     0.03    14.83      1.10     1027    305     19      5      1      1      0      1      0      0
                                                    CUM PERCENT     75     98 **  99     99     99     99      .    100      .      .

  052    ADMISSIONS  1118    1.37     0.03     8.90      0.92      896    209      9      3      1      0      0      0      0      0
                                                    CUM PERCENT     80 **  98     99     99    100      .      .      .      .      .

  053    ADMISSIONS  1444    1.69     0.03    10.69      1.22      971    406     55     10      1      1      0      0      0      0
                                                    CUM PERCENT     67     95 **  99     99     99    100      .      .      .      .

  055    ADMISSIONS   565    2.19     0.03    13.80      1.43      283    272     54      3      2      0      1      0      0      0
                                                    CUM PERCENT     50     89 **  98     99     99      .    100      .      .      .

  056    ADMISSIONS  1076    1.82     0.03     8.49      1.25      662    366     35     11      2      0      0      0      0      0
                                                    CUM PERCENT     61     95 **  98     99    100      .      .      .      .      .

  05F    ADM COUNS    241    1.46     0.03     9.00      1.03      184     53      3      0      1      0      0      0      0      0
                                                    CUM PERCENT     76     98 **  99      .    100      .      .      .      .      .

  06F    ADMISSIONS   312    1.93     0.03    14.88      1.64      189    103     11      6      1      1      0      1      0      0
                                                    CUM PERCENT     60     93 **  97     99     99     99      .    100      .      .

USER TOTALS - NUMBER OF ENTRIES  6115 AVERAGE RESPONSE TIME   1.65
```

Figure 2. Sample TRT Output for Computer Center Terminals

```
PROPRIETARY PROGRAM OF I. C. M.          ***************************************************************
INTERNATIONAL COMPUTER MONITORING INC    ***                                                         ***
CORPCRATE HEADQUARTERS                   ***    T E R M I N A L    R E S P O N S E    T I M E        ***
3289 MOHAWK CIRCLE                       ***                                                         ***
PROVO - UTAH                             ***           S O F T W A R E    M O N I T O R              ***
801-377-6826                             ***                                                         ***
                                         ***************************************************************


********* RUN STARTED AT   9.01.11 THURSDAY  28 JUL 77  ***************** RUN STCPPED AT  17.01.15 THURSDAY  28 JUL 77  *********

********* SAMPLING PERIOD NUMBER   1 -    8 HOURS AND     0 MINUTES - FROM 09.01.11. TO 17.01.11

                                                                       NUMBER OF RESPONSES FOR EACH TIME INCREMENT (INCREMENTS IN SECONDS)
                                                                       -------------------------------------------------------------------
TERMINAL  TERMINAL    TOTAL   AVERAGE  MINIMUM  MAXIMUM  STANDARD      0.00    2.01   4.01   6.01   8.01   10.01  12.01  14.01  16.01  18.01
ADDRESS      ID      NO.RESP  R. T.    R. T.    R. T.    DEVIATION     TO      TO     TO     TO     TO     TO     TO     TO     TO     AND
                                                                       2.00    4.00   6.00   8.00   10.00  12.00  14.00  16.00  18.00  OVER

  058    COMP SERV    1583    0.39     0.03      9.27     0.41          1554     21      6      1      1      0      0      0      0      0
                                                          CUM PERCENT  98 **   99     99     99     100     .      .      .      .      .

  06C    COMP ROOM    2327    0.48     0.03     10.87     0.53          2248     65     10      1      2      1      0      0      0      0
                                                          CUM PERCENT  96 **   99     99     99     99    100      .      .      .      .

  08C    COMP SERV    1020    0.58     0.03     13.90     0.84           979     26      8      2      3      1      1      0      0      0
                                                          CUM PERCENT  95 **   98     99     99     99     99    100      .      .      .

  080    AAP           667    0.63     0.03     14.78     0.79           643     17      4      1      1      0      0      1      0      0
                                                          CUM PERCENT  96 **   98     99     99     99      .      .    100      .      .

  092    ADM DEV      1314    0.47     0.03     10.52     0.38          1292     20      1      0      0      1      0      0      0      0
                                                          CUM PERCENT  98 **   99     99      .      .    100      .      .      .      .

  094    COMP SERV    1209    0.81     0.03     22.15     1.07          1115     70     13      5      2      2      1      0      0      1
                                                          CUM PERCENT  92 **   98     99     99     99     99     99      .      .    100

  095    COMP SERV     966    0.87     0.03     28.98     1.08           877     68     10      7      3      0      0      0      0      1
                                                          CUM PERCENT  90 **   97     98     99     99      .      .      .      .    100

  096    AAP           732    0.68     0.03     41.95     0.94           748     27      5      0      0      0      0      1      0      1
                                                          CUM PERCENT  95 **   99     99      .      .      .      .     99      .    100

  098    ADM RJE      2050    0.54     0.03     27.22     0.82          1992     41     11      3      0      0      0      0      0      3
                                                          CUM PERCENT  97 **   99     99     99      .      .      .      .      .    100

USER TOTALS - NUMBER OF ENTRIES 11918 AVERAGE RESPONSE TIME   0.57
```

Figure 3. Sample TRT Output Admissions Department

D. PERFORMANCE MEASUREMENT

COMPARATIVE STUDY OF TASK DISPATCHING ALGORITHMS IN AN IBM MVT/ASP ENVIRONMENT

S. Elmer Freeman and F. Cary Green

Georgetown Computer Support Project
Office of Management and Computer Systems
National Oceanic and Atmospheric Administration
Washington, D. C. 20035

The study presents the results of a comparison of three dispatching algorithms available in the IBM MVT/ASP environment. The three algorithms investigated are: MVT/ASP with standard dispatching; MVT/ASP with dynamic dispatching; and MVT/ASP using APG from the LSPS package. The comparisons were made using the synthetic benchmark approach described in the NBS publication "Use of Synthetic Benchmarks for Estimating Service Bureau Processing Charges." A relationship for the synthetic benchmark in terms of elapsed running time is also developed in addition to those described in the NBS publication. A discussion of the dispatching algorithms as well as the procedure used for construction of the benchmark are described.

Key words: Benchmarking; dispatching algorithms; synthetic benchmarking.

I. Introduction

Keeping I/O bound tasks running while maintaining an acceptable level of service to all CPU bound tasks is a desirable feature of most any general purpose operating system. Historically several schemes have been proposed (time slicing, etc.) and implemented to achieve these goals. More specifically, three dispatching[1] algorithms for task sequencing are available to users of IBM's OS/MVT operating system. The standard MVT approach, a priority method, relies on an a priori knowledge of the task's CPU and I/O characteristics and ignores the fact that these characteristics may change during the life of the task. In the early 1970's HASP and ASP (one of which was usually an integral part of an OS/MVT system) incorporated a monitor (called the Dynamic Dispatcher in ASP) which determines whether tasks are CPU or I/O bound and reorders the dispatching queue placing I/O bound tasks closer to the beginning of the sequence. Finally, in 1975, through the efforts of SHARE (the IBM users group) Automatic Priority Grouping (APG) of the IBM Large System Program Support package (LSPS) was made available to the general user community. APG not only reorders the dispatching sequence periodically but also assures that there exists an optimal percentage of CPU bound tasks. To maintain this percentage, the dispatcher varies the criteria for CPU and I/O boundedness as the workload changes. Investigation of operating system performance using these three approaches has revealed some dramatic results in terms of throughput and resource utiliza-

[1]Dispatching - the process of allocating the CPU resource to a task in an OS/MVT environment.

tion. The following discussion details these results and explains the construction and running of the necessary benchmarks and the operation of the algorithms themselves.

II. Algorithm Descriptions

In the basic MVT operating system tasks are scheduled on a priority basis. The source of the priority may be from one of two job control language parameters; in either case assignment of priority is made at the beginning of task execution. This may allow performance advantages in an installation where job characteristics are well known and the priority scheme is directly related to these characteristics. However, in many installations either job characteristics are not known or are subject to significant change during execution. These shortcomings have prompted development of at least two other methods of dynamically measuring CPU utilization and rearranging the dispatching sequence to keep I/O bound tasks active and CPU tasks from "hogging" the processor.

In the Dynamic Dispatcher an attempt is made to distribute the CPU resource more equitably. The Dynamic Dispatcher is implemented as a system task which periodically inspects CPU usage of tasks in a certain priority range. Both the interval between inspection and the range of priorities are specifiable by the installation system programmer. Usage of the CPU resource is maintained according to the following equation:

$$h_{t,n} = CPU_{t,n} + h_{t-1,n} - H_t/N \qquad (1)$$

$$\text{where } H_t = \sum_{i=1}^{N} (CPU_{t,i} + h_{t-1,i});$$

N	=	the actual number of tasks being monitored during the time interval;
$h_{t-1,n}$	=	the history of CPU utilization for task n taken at the previous time interval;
$CPU_{t,n}$	=	the amount of CPU time used for task n during time interval t.

The $h_{t,n}$ values are measures of the current and most recent CPU usage characteristics tempered by the average usage for all tasks within the priority range. A low $h_{t,n}$ value is indicative of low CPU usage by a task. A high $h_{t,n}$ value indicates high processor usage by the task [7][2].

During processing at the end of each inspection interval, $h_{t,n}$ is calculated and the task dispatching sequence is reordered by sorting tasks in order of increasing $h_{t,n}$. Thus, tasks using smaller amounts of I/O will have a higher priority relative to CPU bound tasks. Newly initiated tasks have a zero value for $h_{t-1,n}$. Tasks continuing to require small amounts of the CPU resource will have negative $h_{t,n}$ values. To insure a quick response by the algorithm to a sudden change in CPU utilization by a task, there exists a lower bound below which a task's $h_{t,n}$ value will not be allowed. Tasks which use the Attach feature of OS/MVT to create other tasks are not addressed by the Dynamic Dispatcher.

APG is similar to the Dynamic Dispatcher in that "it dynamically determines the relative I/O or CPU boundedness ... and establishes dispatching priority ... on that basis" [3]. As in the Dynamic Dispatcher only those tasks within the priority range specified by the system programmer are addressed. Further, tasks within the range are segregated into I/O bound tasks which have a favored, higher position in the dispatching sequence and CPU bound tasks which have a less favorable position near the bottom. If the task to be dispatched is within the APG range, then it is dispatched for a specified time interval. If the task relinquishes the processor before expiration of its interval, it is marked I/O bound and, if previously I/O bound, remains in that position in the dispatching queue. If a task was previously CPU bound it is now repositioned as the last task in the I/O bound section of the APG range (such tasks are positioned last under the presumption that, since they have recently been CPU bound, they are likely to be so again). However, if the time interval expires before the task relinquishes control of the processor, the task is marked CPU bound (if it wasn't already) and is placed at the bottom of the CPU bound portion of the APG range. This methodology also effects a round robin algorithm for continuously CPU bound tasks.

The aforementioned dispatching interval for APG is dynamically determined in order

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

to maintain an optimal ratio of task switches caused by tasks going CPU bound to total task switches. Periodically, the recent history concerning task switches is examined and if the ratio of tasks going CPU bound to total task switches is less than this optimal ratio, then the interval is "too long" and is shortened by a fixed amount. Conversely, if the ratio is greater than optimal the interval is "too short" and it is increased by a fixed amount. For MVT systems there appears to be nothing in the literature known to us to suggest whether the optimal ratio should be determined analytically or empirically. In any event, in the original coding of APG the ratio was 1:2 and experimentation with several ratios up through 1:8 has indicated that 1:2 is at least as good as any other for this installation.

The range of job priorities to be addressed by APG is specifiable by the system programmer as is the amount by which the interval is changed. APG is implemented as a change to the OS/MVT control program.

III. Benchmark Construction

Resource-oriented synthetic benchmarks were first introduced in 1969 by Buchholz [1]. Buchholz described a parameter-driven PL/I program which he characterized as "a well-behaved exerciser of system features ...". Since then, resource-oriented synthetics have been investigated extensively and the Buchholz program has been used as the basis for many resource-oriented synthetic benchmarks.

A well-travelled and modified Fortran version of the original Buchholz synthetic program was selected for this study. The program performs as a generalized file maintenance program operating on two ordered files, a master file and a detail file. The program sequentially reads the master file until a record is found which matches the current detail record. Upon detection of a match, a compute-bound kernel is executed. An attempt is then made to match the next detail record. This process is continued until the end of the master file is reached. Input parameters are chosen to control the master and detail records, the type of compute-bound kernel activity to be performed following a master-detail match, and the number of times this kernel activity is to be performed for each match. Note that this program models all I/O channel activity via disk I/O. The methodology used to calibrate the synthetic program on the NOAA IBM 360/65 is identical to that used by Conti [2].

Calibration of the synthetic program with respect to CPU time, EXCP[3] counts, and elapsed time was accomplished by running the program over a wide range of NMAS (number of master records) and NCPURP (number of CPU loop repetitions per detail record) values as input; and the CPU time, EXCP counts and elapsed time on NOAA's IBM 360/65 were noted, as shown in Table 1. The NDET (number of detail records), NPASS (number of passes), and ITYPE (type of CPU kernel activity) parameters were held constant at 12, 1, and 1 (integer arithemetic), respectively. Using the model:

$$\text{CPU time} = X_1 \cdot \text{NMAS} + X_2 \cdot \text{NCPURP} + X_3$$

linear regression analysis of the data in Table 1 produced the following relationship between CPU time and the NMAS and NCPURP parameters:

$$\text{CPU time} = .0042 \cdot \text{NMAS} + .0018 \cdot \text{NCPURP} + .5816 \tag{2}$$

Analysis of the EXCP counts in Table 1 produces the following relationship between EXCP counts and NMAS values:

$$\text{EXCP'S} = 3 \cdot \text{NMAS} + 54 \tag{3}$$

This count includes 12 EXCP's for the synthetic SYSIN and SYSOUT[4] activity. Further, the DCB[5] parameter values for the four temporary data sets used by the program are:

BLKSIZE	LRECL	RECFM
800	32768	VS

For use of the synthetic benchmark program in this study a relationship was also established between elapsed time in terms of CPU time and EXCP counts (this relationship was established by running each job stand-alone). Using the

[3]EXCP - a measure of the number of I/O block transfers on IBM systems.

[4]SYSIN and SYSOUT - unit record I/O on IBM systems.

[5]DCB (Data Control Block) - control block containing file description characteristics on an IBM system.

model:

$$\text{Elapsed time} = X_1 \cdot \text{CPU} + X_2 \cdot \text{EXCP} + X_3$$

linear regression analysis of the data in Table 1 produced the following relationship:

$$\text{Elapsed time} = 1.0046 \cdot \text{CPU} + .0119 \cdot \text{EXCP} + 5.4439 \quad \text{seconds} \qquad (4)$$

From the inverse of eqs (3) and (2), the proper values of NMAS and NCPURP needed for the synthetic program to duplicate a prescribed CPU time and EXCP count can be determined:

$$\text{NMAS} = (\text{EXCP} - 54)/3 \qquad (5)$$

$$\text{NCPURP} = (\text{CPU} - .5816 - .0042 \cdot \text{NMAS})/.0018 \qquad (6)$$

Two synthetic job mixes were prepared in order to compare the performance of the three dispatching strategies. The first job mix was composed of twenty three-part jobs. Each job used the same total CPU time and EXCP counts. However, each of the three job-parts exhibited different behavior. Each job-part was described in terms of the following CPU boundedness definition:

$$\text{CPU boundedness} = \text{CPU time}/\text{Elapsed time}$$

Furthermore, the following definitions were assumed:

CPU bound	- CPU boundedness $\geq$ .75
Mixed	- .75 > CPU boundedness > .25
I/O bound	- .25 $\geq$ CPU boundedness

Each job was constructed using all three CPU boundedness types, i.e. CPU bound, mixed, and I/O bound. The specific values selected were:

TYPE	BOUND	ELAPSED	CPU	EXCP	NMAS	NCPURP
CPU	.79	37.13	29.24	204	50	16000
MIXED	.44	71.48	31.29	2754	900	16000
I/O	.12	46.24	5.55	3054	1000	500

Using all permutations of the three different part behaviors, six jobs each having identical overall characteristics but exhibiting differing CPU boundedness during execution were obtained.

The second benchmark consisting of fifty jobs was developed from a random sample of the NOAA IBM 360/65 SMF[6] data from a representative month of processing using a random number generator with the uniform distribution to select both the day of the month and a particular job within that day. The CPU time and EXCP counts for each of the fifty jobs were used in eq (4) to obtain an estimated stand-alone run time. If the estimated stand-alone run time exceeded thirty minutes, a scaled synthetic job of thirty minutes or less elapsed time was then constructed. This random sample produced a surprisingly representative benchmark in that the average CPU time and EXCP counts for the selected month were almost identical to those produced by the benchmark.

For purposes of comparing the three dispatching strategies core was not considered as a constraint for either of the job mixes.

IV. Results

In order to discuss the results of the two benchmarks some terminology must first be developed.

Initiator/terminator - a part of the job scheduler. In the MVT configuration of the control program, the initiator/terminator selects a job from the input work queue, allocates resources required to perform a step of the job, loads and transfers control to the program that is to be executed to perform the job step, and terminates the job step when execution of the program is completed.

Initiator capacity time - the amount of time (in seconds) available to one initiator for execution of jobs.

Total system capacity - the sum of individual initiator capacity times.

Benchmark completion time - the termination time of the last active job in a benchmark job mix.

[6]SMF (System Management Facilities) - resource utilization and accounting information produced by IBM OS operating systems.

Remaining initiator capacity time - the difference between benchmark completion time and the time when a given initiator completes processing its last job.

Remaining system capacity - the sum of individual remaining initiator capacity times for a given benchmark.

Absolute benchmark completion time - the benchmark completion time for the benchmark under any one of the three dispatching strategies.

For purposes of benchmarking the three dispatching strategies, six initiators were chosen (the average number of active initiators at this installation).

For synthetic job mix 1 absolute benchmark completion time is 2130 seconds after benchmark initiation. Therefore, total system capacity is 12780 seconds. Figure 1 illustrates the results of running synthetic job mix 1 under all three dispatching strategies. Note that the APG system has 11.3% more remaining system capacity than the OS/MVT system and 6.1% more than the Dynamic Dispatcher system. Figure 3 presents the system active CPU time distribution for the three dispatching strategies. Note here that APG returns 12% of the CPU resource to the problem state over the OS/MVT system and 7% more than the Dynamic Dispatcher system.

For synthetic job mix 2 absolute benchmark completion time is 6060 seconds after benchmark initiation. Therefore, total system capacity is 36360 seconds. Figure 2 illustrates the results of running synthetic job mix 2 under all three dispatching strategies. Note that the remaining system capacity for the APG system versus the remaining capacities for the OS/MVT and Dynamic Dispatcher systems exceeds even that which was found in the running of synthetic job mix 1. Figure 4 presents the system active CPU time distribution for this benchmark. Note here that the pronounced difference in problem program state CPU is not seen as it was in job mix 1. However, the system CPU figures dropped significantly with the use of APG versus the other two strategies; time did not permit an investigation of this behavior.

Although the results presented are based on a statistically small sample, subsequent measurements of this installation's operational environment while running APG support these results. Since installing APG at this installation we have experienced an increase in the number of jobs processed per unit of time and have seen an increase in the amount of CPU time spent in the problem program state.

V. Concluding Remarks

Three methods of task sequencing have been presented and compared. It has been demonstrated that several performance metrics are dependent upon the choice of task sequencing algorithm; clearly, the APG algorithm is superior in these areas. While APG is certainly not the ultimate solution to the problem of optimal task sequencing, it has provided exceptional benefits to this installation.

We would like to thank Mr. Fred Long of the National Oceanic and Atmospheric Administration for his constructive evaluation of this paper and for his many useful suggestions.

References

[1] Buchholz, W., "A Synthetic Job for Measuring System Performance," *IBM Systems Journal*, 8:4 (1969), 309-318, 6 refs.

[2] Conti, D., "Use of Synthetic Benchmarks for Estimating Servic Bureau Processing Charges," *NBS Technical Note 920*, National Bureau of Standards, Washington, D. C., July 1976, 13 refs.

[3] International Business Machines Corp., "LSPS Documentation", April 1971.

[4] International Business Machines Corp., "IBM System/360 Operating System: Introduction," Form GC28-6534.

[5] International Business Machines Corp., "The HASP System Documentation for IBM Type 3 Program, HASP-II, Version 3.0," No. 360D-05.1.014, February 1971.

[6] International Business Machines Corp., "IBM System/360 and System/370 ASP, Version 3, Asymmetric Multiprocessing System, Logic Manual," Form GC20-1403, September 1973.

[7] Strauss, J., "An Analytic Model of the HASP Execution Task Monitor," *Communications of the ACM*, 17:12 (December 1974), 679-685, 9 refs.

APPENDIX A

TABLES

TABLE 1

SYNTHETIC PROGRAM CALIBRATION
(NDET=12,NPASS=1,ITYPE=1)

NMAS	NCPURP	CPU TIME (SEC)	EXCP	ELAPSED TIME (SEC)
12	0	0.67	90	7.47
50	0	0.50	204	8.92
100	0	1.00	354	11.12
500	0	3.47	1554	26.98
900	0	4.89	2754	42.92
1000	0	5.17	3054	46.22
12	500	1.70	90	7.65
50	500	1.62	204	10.53
100	500	1.97	354	11.70
500	500	4.19	1554	28.87
900	500	5.44	2754	43.23
1000	500	5.55	3054	46.25
12	1000	2.92	90	8.66
50	1000	2.49	204	10.78
100	1000	2.97	354	12.64
500	1000	4.72	1554	27.87
900	1000	5.57	2754	43.42
1000	1000	6.00	3054	49.03
12	4000	7.85	90	13.80
50	4000	8.15	204	15.97
100	4000	8.07	354	17.97
500	4000	10.07	1554	33.35
900	4000	11.79	2754	49.98
1000	4000	11.19	3054	54.23
12	16000	29.29	90	35.44
50	16000	29.24	204	37.13
100	16000	29.72	354	39.25
500	16000	31.52	1554	55.10
900	16000	31.29	2754	71.48
1000	16000	34.79	3054	75.32
12	90000	161.29	90	168.37
50	90000	161.14	204	170.13
100	90000	161.35	354	172.11
500	90000	163.25	1554	187.88
900	90000	165.20	2754	203.62

TABLE 2

SYNTHETIC BENCHMARK JOB MIX 1

JOBNAME	STEP 1	STEP 2	STEP 3
A	IO	MIXED	CPU
B	IO	CPU	MIXED
C	MIXED	IO	CPU
D	MIXED	CPU	IO
E	CPU	IO	MIXED
F	CPU	MIXED	IO
G	IO	CPU	MIXED
H	IO	MIXED	CPU
I	MIXED	IO	CPU
J	MIXED	CPU	IO
K	CPU	IO	MIXED
L	CPU	MIXED	IO
M	IO	MIXED	CPU
N	IO	CPU	MIXED
O	MIXED	IO	CPU
P	MIXED	CPU	IO
Q	CPU	IO	MIXED
R	CPU	MIXED	IO
S	IO	MIXED	CPU
T	IO	CPU	MIXED

TYPE	SYNTHETIC INPUT PARAMETERS (NDET=12,NPASS=1,ITYPE=1)	
STEP	NMAS	NCPURP
CPU	50	16000
MIXED	900	16000
IO	1000	500

TABLE 3

EXPECTED VS. ACTUAL CPU TIMES, EXCP COUNTS, AND ELAPSED TIMES FOR JOB MIX 1 FOR NOAA 360/65

TYPE	CPU TIME (SEC)		ELAPSED TIME (SEC)		EXCP'S	
STEP	EXPECTED	ACTUAL	EXPECTED	ACTUAL	EXPECTED	ACTUAL
CPU	29.59	29.24	37.25	37.13	204	204
MIXED	33.16	31.29	69.65	71.48	2754	2754
IO	5.68	5.55	47.36	46.25	3054	3054

TABLE 4

PARAMETER SPECIFICATIONS FOR SYNTHETIC JOB MIX 2

JOB NAME	SYNTHETIC INPUT PARAMETERS (NDET=12,NPASS=1,ITYPE=1) NMAS	NCPURP
SIM10	3328	46911
SIM11	29	164
SIM12	3570	84113
SIM13	342	1656
SIM14	7041	146036
SIM15	2113	21963
SIM16	2878	8517
SIM17	25	729
SIM18	25	729
SIM19	1801	1024
SIM20	550	60
SIM21	341	12770
SIM22	1280	5023
SIM23	306	7846
SIM24	1210	17409
SIM25	29	164
SIM26	3225	29368
SIM27	5623	34334
SIM28	307	2849
SIM29	1070	22174
SIM30	5171	22055
SIM31	411	4273
SIM32	1350	415
SIM33	6445	247416
SIM34	1454	1284
SIM35	6802	73811
SIM36	4718	41996
SIM37	1000	38454
SIM38	376	6577
SIM39	64	83
SIM40	619	2676
SIM41	617	58231
SIM42	16285	138917
SIM43	237	2451
SIM44	1280	1
SIM45	18270	95369
SIM46	132	12702
SIM47	897	3139
SIM48	1349	13751
SIM49	98	1114
SIM50	6110	18192
SIM51	8679	61087
SIM52	445	13638
SIM53	964	64088
SIM54	724	765
SIM55	29	1831
SIM56	98	5559
SIM57	2426	6238
SIM58	341	3875
SIM59	29	1275

TABLE 5

EXPECTED VS. ACTUAL CPU TIMES, EXCP COUNTS, AND ELAPSED TIMES FOR JOB MIX 2 FOR NOAA 360/65

JOB	CPU TIME (SEC)		ELAPSED TIME (SEC)		EXCP'S	
NAME	EXPECTED	ACTUAL	EXPECTED	ACTUAL	EXPECTED	ACTUAL
SIM10	99.00	99.70	224.35	254.73	10038	10038
SIM11	1.00	1.57	8.13	8.50	141	147
SIM12	166.98	166.84	301.28	335.81	10764	10764
SIM13	5.00	4.89	23.32	28.32	1080	1080
SIM14	293.02	293.07	551.82	620.60	21177	21177
SIM15	48.99	48.92	130.74	129.68	6393	6393
SIM16	28.00	27.42	136.96	134.66	8688	8688
SIM17	2.00	2.40	8.99	9.13	129	129
SIM18	2.00	2.19	8.99	9.32	129	129
SIM19	9.99	10.67	80.42	79.14	5457	5457
SIM20	3.00	3.05	28.74	28.61	1704	1704
SIM21	25.00	24.72	43.38	43.48	1077	1077
SIM22	15.00	15.27	66.85	65.98	3894	3894
SIM23	15.99	17.04	33.07	33.20	972	972
SIM24	37.00	37.97	86.45	85.58	3684	3684
SIM25	1.00	1.65	8.13	8.78	141	147
SIM26	66.99	67.45	188.52	187.70	9729	9729
SIM27	86.00	86.97	293.22	290.42	16923	16923
SIM28	7.00	7.37	24.08	24.15	975	975
SIM29	44.99	45.07	89.48	88.52	3264	3264
SIM30	62.00	62.15	252.98	251.90	15567	15567
SIM31	10.00	10.24	30.81	30.75	1287	1287
SIM32	7.00	5.19	61.31	60.25	4104	4104
SIM33	473.00	467.04	711.35	706.40	19389	19389
SIM34	9.00	10.52	67.04	66.11	4416	4416
SIM35	162.01	161.42	411.67	411.24	20460	20460
SIM36	95.99	95.52	270.95	268.07	14208	14208
SIM37	74.00	74.22	116.13	115.40	3054	3054
SIM38	14.00	14.00	33.57	33.68	1182	1182
SIM39	1.00	0.80	9.38	9.41	246	246
SIM40	8.00	8.54	36.22	36.14	1911	1911
SIM41	107.99	107.24	136.60	135.92	1905	1905
SIM42	319.03	320.87	907.96	915.18	48909	48909
SIM43	5.99	5.79	20.56	21.31	765	765
SIM44	5.96	6.19	57.77	57.46	3894	3894
SIM45	248.98	256.19	908.45	917.75	54864	54864
SIM46	24.00	23.90	34.91	35.35	450	450
SIM47	10.00	11.15	48.16	47.54	2745	2745
SIM48	31.00	30.70	85.39	85.41	4101	4101
SIM49	3.00	4.40	12.60	12.77	348	348
SIM50	58.99	59.55	283.47	295.40	18384	18384
SIM51	146.99	146.04	463.59	467.70	26091	26091
SIM52	27.00	26.44	49.10	49.32	1389	1389
SIM53	119.99	118.14	161.04	159.87	2946	2946
SIM54	5.00	3.85	36.96	36.38	2226	2226
SIM55	4.00	4.54	11.14	11.82	141	147
SIM56	11.00	11.35	20.64	21.03	348	348
SIM57	22.00	22.04	114.80	113.38	7332	7332
SIM58	8.99	9.24	27.29	27.15	1077	1077
SIM59	3.00	3.55	10.14	10.71	141	147

APPENDIX B

RESULTS

FIGURE 1

TOTAL SYSTEM CAPACITY 12780 SECONDS

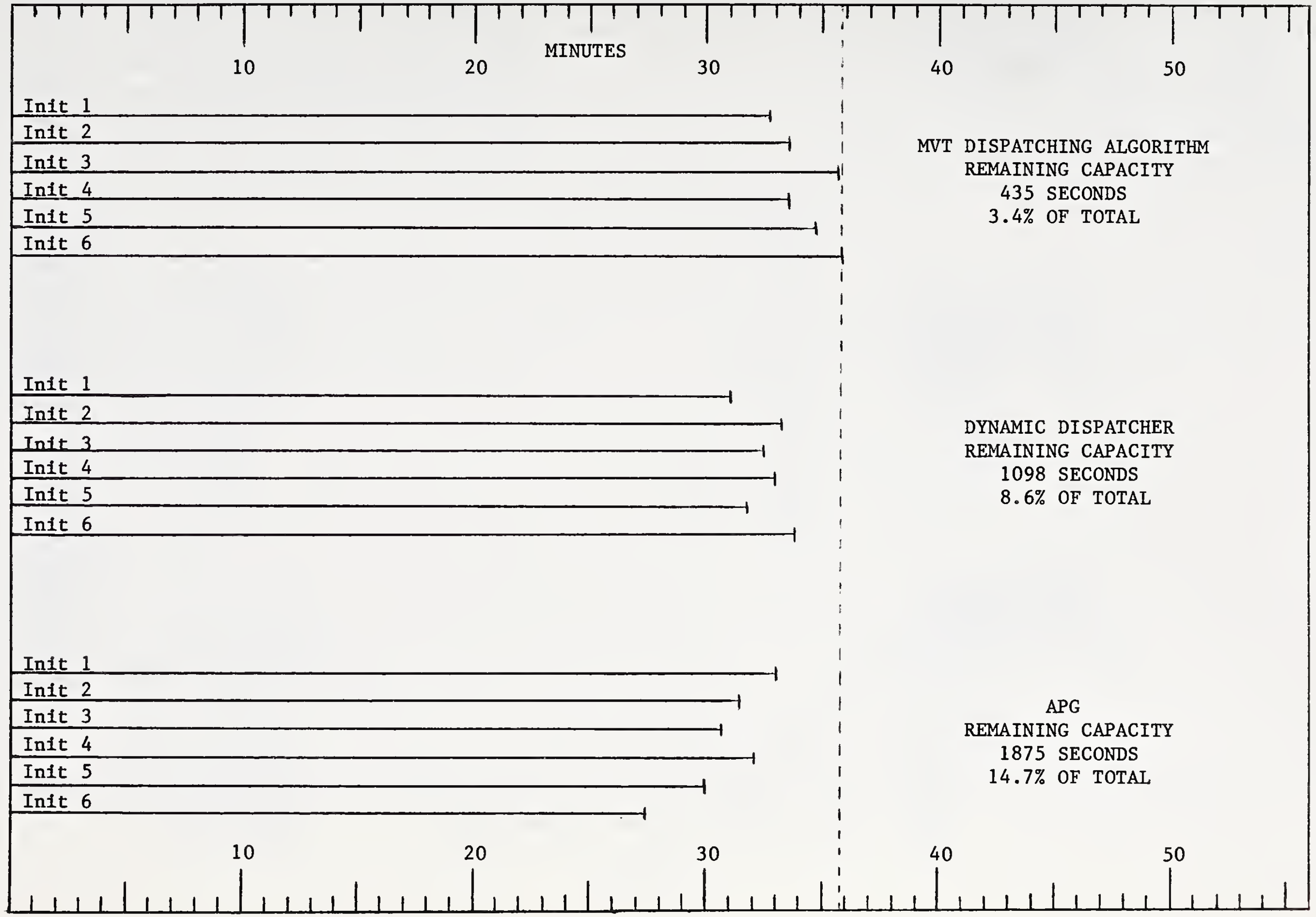
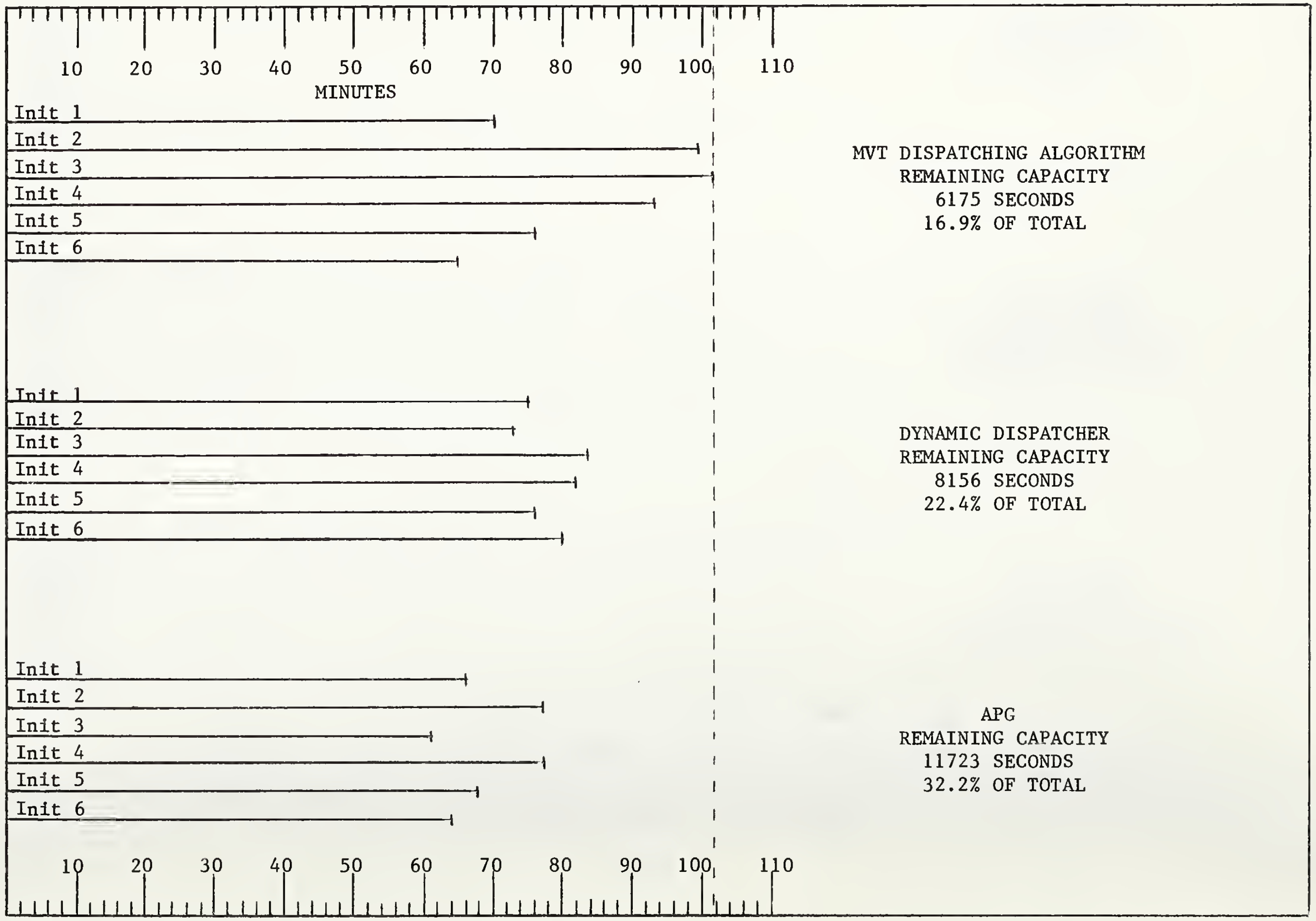
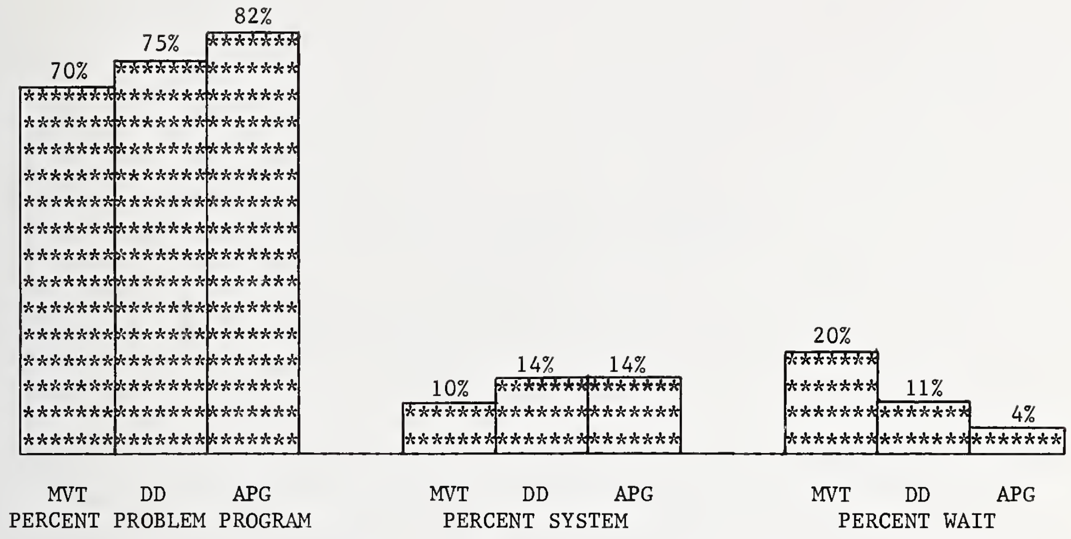


FIGURE 2

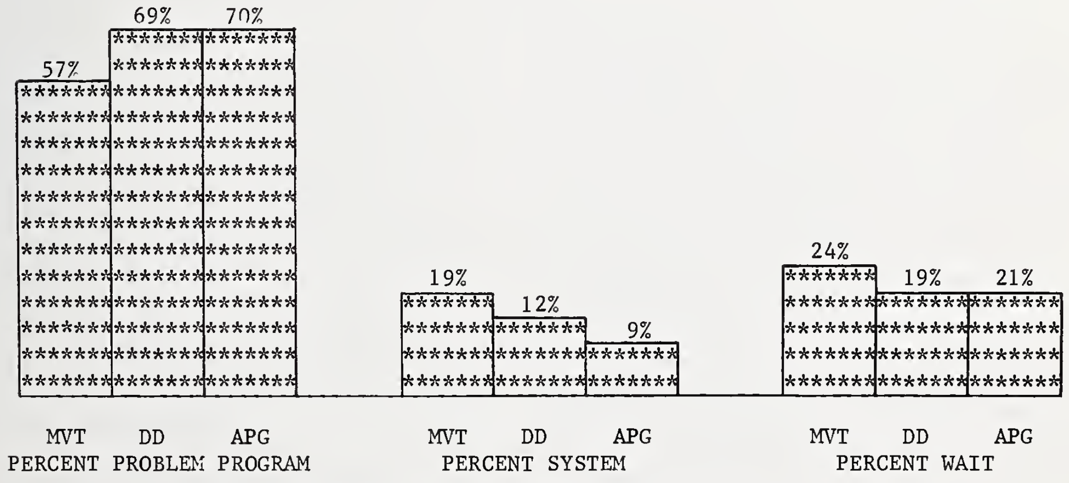
TOTAL SYSTEM CAPACITY 36360 SECONDS





-- SYSTEM ACTIVE CPU TIME DISTRIBUTION --
SYNTHETIC BENCHMARK JOB MIX 1[a]

FIGURE 3



-- SYSTEM ACTIVE CPU TIME DISTRIBUTION --
SYNTHETIC BENCHMARK JOB MIX 2[a]

FIGURE 4

---

[a] Each asterisk (*) represents 5% on the vertical scale.

COMPUTER PERFORMANCE COMPARISONS

Wayne Douglas Bennett

Navy Regional Data Automation Center
Washington Navy Yard
Washington, D.C. 20374

This paper enumerates the necessary and sufficient conditions under which system accounting log data collected over one time interval of normal operations may be statistically compared to that over a second interval, obviating the need for benchmark experiments. It is determined that since major modifications to hardware and system software, as well as shifts in workload character occur at identifiable points in time, the actual operational environment may, in fact, be considered sufficiently "controlled" to permit the inspection of one random variable and its effect on system performance over two carefully selected intervals. Included is an account of such a comparison as performed at NARDAC Washington DC which uses these concepts to investigate the effect of a modification to the system scheduling algorithm on an HIS-6000 computer.

Key words: Availability; basis statistics; offered load; performance.

1. Introduction

System performance is measured via a set of statistics commonly thought to reflect the multiprogramming efficiency of the system. This set includes such measures as average job turnaround and queue times as well as throughput and system multiprogramming depth. When one or more of these values shifts over time, very little information is available regarding the cause(s) of the shift(s). Certainly, a decrease in job turnaround could easily be attributed to fewer jobs, shorter jobs, or perhaps even a new release of the operating system.

This discussion is an attempt to enumerate a set of "basis statistics" which, when computed along with the performance measures, yields a more complete picture of why performance has changed. In fact, it is determined that computation of these basis statistics is a prerequisite to adequate performance analysis. An example is given here in which an attempt is made to determine the effect of a modification to the system scheduler on an HIS 6000 single CPU system. The performance variables of particular interest were defined as turnaround time and throughput. However, all available performance statistics (such as input-queue time and activity-queue time, etc.) were also computed, in order to insure that all possible effects on the system could be analyzed. In this context, performance variables are defined as statistics resulting from events whose timing depends upon the entire multiprogramming environment. Performance variables are actually "effects" whose "causes" shall be labelled "basis statistics." The latter can be categorized into three broad areas:

System Availability
Offered Workload
Site Parameters

The thesis presented here asserts that meaningful comparison of performance statistics depends upon the extent to which normal operation at the site approaches a controlled environment. That is to say that the intervals of time used for the comparison must be selected in a fashion that allows only one of the basis statistics to vary across the two intervals. Clearly, the rest cannot remain constant. However, simple statistical tests (such as the two-tailed test of normal probabilities) requiring only means and standard deviations of each variable can be used to determine whether any basis statistic has changed significantly over the two intervals. When similar tests are used on performance variables, the causes for any significant shift in these is immediately identifiable as the one basis statistic which was allowed to vary.

2. Time Intervals

Let there be two distinct periods of time, p(1) and p(2). Sometime between these two periods, one variable is known or thought to have varied (in this instance, the system scheduler). It is desired that system performance be measured over p(1) and p(2) in order to determine whether the system's multiprogramming efficiency (as expressed in terms of job turnaround time and throughput) has improved as a result of this single modification. However, the intervals of interest must be constructed carefully. First, each period may not be of interest in its entirety. Perhaps only weekday-prime-time statistics are desired. Let the sum of all such intervals of interest over the period p(1) be called I(1). Further, over this sum of time, the operating system may or may not have been up and available for general use. In fact, each subsystem of the operating system (such as time-sharing) has associated with it a set of intervals during which it is up. Let SS be the set of all software systems at the site, including the operating system. Suppose there are Ns such elements, in the set SS, and let the matrix S (size: 2*Ns) be defined where each element, s(i,j), represents the total time over the interval I(i) that the software system ss(j) is both up and available for general use. All statistics of interest must now be calculated over each "interval" included as an entry of the matrix S. Some statistics will require a further breakdown of these time intervals. Over any s(i,j), the configuration can be expected to change often (whenever an operator assigns or releases a device from the system), and so there exists a set of smaller intervals of time, t(i,j,k), (subintervals of s(i,j)) over which the configuration remains constant and such that:

$$\sum_{k} t(i,j,k) = s(i,j) \qquad (1)$$

3. Basis Statistics

3.1. System Availability

Availability has two aspects:

Software Up Time
Hardware Up Time

Software up time has been completely characterized by the matrix S. The matrix H will now be constructed to reflect the total amount of available hardware at the site over each of the s(i,j). Let DD be the set of all device types (of which there are Nd) at the site. Then d(i,j,k,m) can represent the number of devices of type dd(m) that were configured over the interval t(i,j,k). Thus,

$$h(i,j,m) = \sum_{k} (t(i,j,k) * d(i,j,k,m)) \qquad (2)$$

yields the total number of device hours available over the interval s(i,j) for device type dd(m). Finally, an availability matrix, A (size: 2*Ns*Nd) can be defined as follows:

$$a(i,j,m) = h(i,j,m)/s(i,j) \qquad (3)$$

where a(i,j,m) is the average number of devices of type dd(m) that were available over the interval s(i,j). The variance of each component can be computed if an H2 matrix is constructed, such that:

$$h2(i,j,m) = \sum_{k} (t(i,j,k) * (d(i,j,k,m)**2)) \qquad (4)$$

Then the components of Var(A) can be computed:

$$var(a(i,j,m)) = (h2(i,j,m)/s(i,j)) - (a(i,j,m)**2) \qquad (5)$$

3.2. Offered Workload

Let the volume of work entering the system be represented by the matrix V (size: 2*Ns), where v(i,j) is the number of jobs entering the system via ss(j) over the interval I(i). If SS is composed of:

ss(1):	Operating System Executive
ss(2):	Time-Sharing Executive
ss(3):	Transaction Processing Executive

then all normal batch jobs will fall under ss(1). Once V is established, it is clear that a useful statistic can be composed in a matrix called E (size: 2*Ns):

$$e(i,j) = v(i,j)/s(i,j) \qquad (6)$$

where e(i,j) is the average number of jobs entering the system via ss(j) over I(i), per unit of time. Unlike V, E is independent of the relative interval lengths, I(1) and I(2). In order to arrive at VarE (the variance matrix corresponding to the matrix E), it is necessary to compute each component from the variance of the inverse of the inter-arrival time for jobs of type ss(j) over the interval I(i).

The character of the offered load is best described by computing the average job requirements for each job type over each of the s(i,j). The set RR includes all user required processing by device type, all core and file space requirements, and all other user defined job parameters (such as initial urgency) affecting job scheduling and processing. If the number of all such requirements is Nr, then let r(i,j,k,n) represent the rr(k)th job requirement of the nth job entering the system via ss(j) during s(i,j). The work matrix, W (size: 2*Ns*Nr), can be defined as follows:

$$w(i,j,k) = [\sum_{n=1}^{v(i,j)} r(i,j,k,n)] \;/\; v(i,j) \qquad (7)$$

where w(i,j,k) represents the mean rr(k)th requirement for all jobs of type ss(j) over I(i). VarW can be constructed from R2, V, and W just as VarA was constructed from H2, S, and A.

3.3. Site Parameters

Site parameters are non-valued, but must, nevertheless, be accounted for. These include system software modifications, changes in working hours and other hard-to-qualify parameters affecting the system. The status of each such parameter must be determined for both time periods, so that they may be compared by inspection while the formally constructed matrices are compared statistically.

4. Basis Comparison

4.1. Overview

We can consider the A matrix as the system attribute matrix, since each component represents average hardware availability per unit of software time. Likewise the W matrix can be thought of as the job attribute matrix. By fixing the first index in each of W and A, a two dimensional plate of each matrix is obtained. Let the plate obtained by fixing the first index of W to one be called W1, and let the similar plate in A be called A1. Likewise setting this index to two yields W2 and A2.

It is desired that W1 be compared to W2 and A1 be compared to A2 in order to determine whether the average job requirements and system availability averages for I(1) are significantly different from those for I(2).

Two kinds of comparisons will be performed:

Individual component comparisons
Scaled sum comparisons.

4.2. Individual Comparisons

Individual component comparison involves only the mean and variance of each component. Each respective pair of entries from W1 and W2 can be compared if a Z statistic is computed from the respective means and variances:

```
Let   D(j,k)  = W(1,j,k) - W(2,j,k)
      E(j,k)  = VarW(1,j,k)/v(1,j)
      F(j,k)  = VarW(2,j,k)/v(1,j)
Then ZW(j,k)  = D(j,k)/((E(j,k,)+F(j,k))
                **.5)
```

Similar computations yield the ZA matrix.

For the alternative that one normal population has shifted from another, tests based on the difference in sample means are generally used. ZW(j,k) becomes a random variable with standard normal distribution if the population means are equal, that is, under the null hypothesis. Against the alternative W(1,j,k) = W(2,n,k) (where inequality represents significant difference) a two-sided critical region $|Z| > K$ would be used at the p confidence level. When Z falls in this critical region, the null hypothesis is rejected. These values can be computed directly using $|Z|$:

$$p = 2*(1-\int_{-\infty}^{|Z|} (1/\sqrt{2\pi})e^{-1/2*t^2}\,dt) \qquad (8)$$

P gives the confidence level at which the alternative can be asserted (e.g., if p=1.97 then it can be stated that at the 95+% confidence level, a shift in mean is likely to have occurred). The corresponding K and p values are tabulated in most statistics handbooks in a table entitled "two-tailed probabilities for the standard normal distribution" and can also be derived from the tabulated percent of the standard normal distribution.

It is clear that if two or more entries in the ZW and ZA matrix exceed the tabulated K value at the user selected confidence interval p, then, should a later performance statistic comparison for the two periods indicate a shift in some performance mean (such as turnaround time), that shift can never be attributed to a single identifiable cause. Suppose, for instance, that the mean CPU time per batch job over I(1) is substantially smaller than that same mean over I(2). If no other batch job requirement or hardware availability mean has shifted across I(1) and I(2), no site parameter has changed status, and the mean batch job turnaround time decreases across I(1) and I(2), then that decrease is attributable to the decrease in CPU time for batch jobs. If, however, the job scheduling algorithm was also modified sometime between I(1) and I(2), then any decrease in batch job turnaround time is not uniquely attributable to either the decrease in CPU time or the modification in the scheduler. Neither would there exist justification for attributing the change in performance to both CPU time and the algorithm. It is highly likely that one of these is the cause. However, if neither is isolated through careful construction of the intervals I(1) and I(2), few conclusions can be drawn.

4.3. Scaled Sum Comparisons

It is often the case that although performance measures have shifted, no individual ZW or ZA entries exceed the tabulated K values and the status of all site parameters has remained constant. This is generally the result of at least one of three causes:

1. A variable has been omitted from one of the three groups of basis statistics.

2. The user selected critical region is too small (i.e., the significance level should be decreased).

3. Although no single basis variable has shifted, the interaction of two or more such variables may indeed cause a shift from I(1) to I(2).

Causes (1) and (2) are easily corrected. Cause (3), however, requires further comparisons. Certainly it is tedious to try all N-way combinations to determine if any have shifted over time. However a simple check can be performed summing all job requirements for each of the Ns job types in order to determine whether any of the

Ns * ((2**Nr) - (Nr+1)) (9)

possible interactions of job requirements have indeed shifted. Likewise each of the Ns columns of the A matrix can be checked in order to determine whether any of the

Ns * ((2**Nd) - (Nd+1)) (10)

possible interactions of hardware availability averages have shifted over time. The job requirements matrix W shall be used as an example. The procedure to be outlined can be used not only for the column-wise Nr and Nd-way interaction of the W and A matrices, but any N-way interaction of variables.

It would be optimal if the job requirement entries in any given column of the W matrix summed to a recognizable quantity. However, this is not the case. In fact, such requirements as "slave CPU seconds" and "number of tapes" are not even summable items, considering the fact that the latter will be swamped by the former. Obviously some scaling is required. A problem immediately encountered is the choice of scale. If the data is normalized using the standard score formula:

$$Xi = (Xi-mean)/(Standard\ deviation) \quad (11)$$

then all normalized addends will have zero as means, their sums will have zero as means and there will never exist a difference in means across I(1) and I(2). However one <u>can</u> proceed using the method of <u>reductio ad absurdum</u>. If the individual components do not have significantly different means across I(1) and I(2), then we shall assume that the two "samples" are actually from the same population. This statistical procedure, analogous to its logical counterpart, calls for seeking a contradiction that will invalidate the assumption. If the samples are from the same population, then a single vector of scaling factors can be derived from the job requirements over I(1) alone. This will lead to a summed mean of zero for the period I(1). However, since the I(2) sample is not <u>identical</u>, using the same scaling vector on I(2) job requirements will yield a summed mean different from zero. These two estimates can then be compared in the same manner as were the individual components. If there exists a significant difference, the contradiction has been found. At the outset, a single population was assumed and the conclusion of significance tests contradicted this assumption.

5. Application

As stated, this procedure was used successfully in a recent effort at NARDAC Washington DC in order to determine the effect of modifications to the scheduling algorithm on a Honeywell single CPU system. The modification involved the construction of an additional queue as well as a redefinition of the urgency setting algorithm incorporated in the scheduler. These changes were implemented and tested during the period 13 September 1976 to 31 October 1976.

The intervals p(1), p(2), I(1) and I(2), as well as the sets DD, SS and RR were defined as follows:

p(1) = 7 June 1976 to 13 September 1976
p(2) = 1 November 1976 to 31 December 1976

The I(i) were defined as all prime shifts (0600-1600) that occurred in P(i) on weekdays during which at least four hours were spent both up and busy.

SETS:

<u>SS</u> (Software Systems)

GCOS OS
TIME-SHARING
TRANSACTION PROCESSING

<u>DD</u> (Device Types)

CPU
CORE
181 DISKS
191 DISKS
7-TRK TAPES
9-TRK TAPES
CARD READER
CARD PUNCH
TERMINALS
PRINTER

<u>RR</u> (Job Requirements)

CPU HOURS
CHANNEL HOURS
DISK HOURS
TAPE HOURS
NUMBER OF TAPES
NUMBER OF DISKS
PRINT LINES
ACTIVITIES
CPU LIMIT
IO LIMIT
CORE LIMIT

The selection of these P(i) was predicated in part on the interim period of testing which could not overlay either P(i). Had a larger P(1) been selected (by starting P(1) prior to June 1976), configuration modification occurring at the end of May 1976 would have caused there to be a decided shift in the A matrix, thus negating the possibility that performance variable shifts were uniquely assignable to the scheduler change.

For these sets the necessary matrices were constructed and it was found that at the .05 significance level, the only shift in basis statistics across the two intervals was, indeed, the status of the scheduler.

The following performance variables of interest were calculated:

Throughput
Turnaround Time/JOB

Input-Queue Time/JOB
Activity Queue Time/ACTIVITY
Swap Time/ACTIVITY
Core Time/ACTIVITY
Multiprogramming Effect/JOB
Useful Processing Per Hour

All performance variables remained constant (to the .05 level of significance) across I(1) and I(2) with the exception of two, input queue time per job and activity-queue time per activity. Input-queue time represents that amount of time prior to the assignment of a program number (catalogued time) plus all time in queue prior to the loading of the first activity or task. This statistic was found to be some 30% lower over I(2) than over I(1). However, activity-queue time, which represents the amount of time between tasks, increased in the order of 50%. Further, the job type that was especially hard hit was batch while jobs spawned via the time-sharing subsystem showed a small improvement.

It was immediately evident that the addition of a queue reduced the time jobs spent catalogued. However, it was equally clear that the new algorithm favored time-sharing spawned jobs to the detriment of batch jobs. An additional, tentative conclusion drawn from this analysis was that the system was in fact heavily loaded, thus negating even the possibility of a scheduling algorithm change having a beneficial effect on throughput time and turnaround. The performance statistics clearly pointed to such a conclusion, and subsequent software monitoring of core and the central processor verified this.

6. Conclusions

The argument can be made that most system statistics are not normally distributed and that, therefore, tests based on normal curves do not apply. However, it should be noted that the variance can be used in a meaningful manner even in non-parametric statistics. Further, any test, even one based upon non-parametric statistics, could replace the one given here. In addition to having yielded reasonable results in the above application, the method of two-tailed normal probabilities is one of the most widely used and easily understood. An effort was made here simply to distinguish basis statistics or "causes" from performance statistics or "effects."

The conditions for meaningful performance analysis are stated as follows:

IF there exists a pair of intervals I(1) and I(2) such that the following statements are true:

1. There exists exactly one basis statistic X such that it is the variable of interest and its value has significantly shifted sometime between I(1) and I(2).

2. No other variable affecting system performance has significantly shifted across the entire interval I(1) + I(2).

3. No N-way interaction between the basis statistics (with the exception of those involving X) has significantly shifted across the entire interval I(1) + I(2).

THEN:

A statistical comparison of performance measures across I(1) and I(2) will yield meaningful results and the cause for any shift found to exist can be assigned to the variable of interest X.

A STUDY ON THE EFFECTIVENESS OF DATA BLOCKING IN AN MVS ENVIRONMENT

R. E. Paulhamus
G. E. Ward

Corporate Computer Center
Comptrollers Department
American Telephone and Telegraph Company

In a production oriented computer center with IBM equipment, blocking of program data can be very advantageous yielding reduced resource utilization. This paper outlines the theory of data blocking, derives equations for two system variables, and interprets sample data in light of these equations. Regression tests help to show the similarity of theoretical and practical results.

Key words: blocking factor; data record; multiple regression; processing time; regression model.

1.0 Introduction

Efficient blocking of program data in an IBM 370 MVS environment can be even more important in terms of performance than in pre-MVS operating systems. Frequently, however, this area is often overlooked in performance and tuning studies. This paper presents investigative work being done in the area. Most of the arguments, reasoning and formulae in the paper also apply to other than MVS operating systems. Where differences exist, we will identify the special considerations for MVS.

The objectives in conducting this research were twofold:

1) to increase thruput of the CPU by reducing the processing time of jobs;
2) to increase utilization of space on direct access secondary storage devices by using more of the device's capacity for storing data.

It was found that effective blocking of program data files provided significant steps in achieving these goals.

We began the study by interrogating the Systems Management Facility (SMF) data, the log file from an IBM system, and quickly became convinced that data sets were not being blocked efficiently by programmers. It was decided that an operating system option, invokable by programmers, should be made available to automatically block new data sets at a utilization factor for the particular device of 95 percent or better. This decision was made as it was felt that information on the efficiency of specific blocking factors and the characteristics of various storage devices was not common knowledge to most programmers.

2.0 Theory of Data Blocking

Before examining the theory and reasoning behind data blocking, it is necessary to define some terms to be used in this paper. First of all, the term record will refer to a logical unit of information. Records are created, manipulated, updated, and destroyed by programmers. External to the program, however, these records are the responsibility of the operating system and in particular, the access methods (e.g., BSAM, QISAM, TCAM). When transferred between main memory and secondary storage, these data records are moved in blocks. A block is a physical unit of data containing one or more logical units (records). Thus the number of blocks of data transferred, or EXecute Channel Program (EXCP) instructions in IBM terminology, may or may not equal the number of program reads and writes. At one extreme (no blocking), there is one physical transfer (EXCP) for every associated program read or write and a block is equivalent to a record. The system overhead for performing an EXCP is high but relatively independent of blocksize. To reduce this overhead, therefore, records can be grouped in one area of memory called a buffer which is an area in core equal to the blocksize; a physical read (write) is then only required when the buffer is empty (full). The amount of core required for a program is relatively unimportant in an MVS environment; therefore buffers can become quite large without adding significant overhead. The process of storing data is performed by the access method in queued methods (e.g. QSAM, QISAM) and by the program in basic access methods (e.g. BSAM, BISAM). The number of records in a block is the blocking factor for the file. When data is blocked, the number of EXCP's is reduced by a factor approximately equal to the blocking factor. Thus, blocking will reduce the total overhead of issuing EXCP's and therefore decrease the processing time of a job. This would help us achieve our first objective. Since most accounting packages charge for EXCP count, blocking can also reduce job costs.

Processing time can also be saved at the storage device end. Assuming that this device is either a fixed (e.g. an IBM 2305) or moveable (e.g. an IBM 3330, 2314, 3350) head device, actual transfer of data is fast compared to the preparation time of the device. This preparation includes head movement for moveable head devices and average rotation delay. Thus fewer transfers of data will result in less total preparation time. Another situation which must be considered is the possibility that other users may be accessing data on the same device. Thus, moveable head devices may spend considerable time alternating between data sets. It is obvious that blocking will also reduce this time.

Blocking of data sets can increase the utilization of the secondary storage space. This utilization is the percentage of the space used to store program data. There is an "inter-block gap," IBG, between each block of data to allow the device some rotation time to sense data. For short unblocked records, more space may be required for the IBG's than for the actual data. The IBG space will be reduced by a factor equal to the blocking factor. Details of the importance of this IBG will be presented in the next section with efficiency calculations.

3.0 Calculations

As stated in the introduction, we were interested in decreasing the elapsed time of jobs and increasing utilization of secondary storage space. We noted that both of these objectives were affected by making blocksizes of program data more efficient. We will now proceed to verify this dependence. We are interested in fixed-length records stored on moveable and/or fixed-head devices. We found the blocking of variable length records to be effective but the proof was not as straightforward.

3.1 Processing Time

The first calculation of interest is the processing time of a task. This factor, PT, is comprised of two components: the CPU time, C, and the non-overlapped input/output time, I/O. We must distinguish between "processing" and "elapsed" time of a job in an MVS environment. This is necessary due to the fact that swap time (time when a job is made ineligible for computer resources due to excessive system activity) can become significant due to a poorly tuned MVS system. When this occurs, swap time may overshadow the CPU and non-overlapped I/O time and totally distort any calculations. Since this is something we do not intend to impact, we do not include the swap time in the processing time calculations.

Thus,

$$PT = C + I/O \qquad (1)$$

but, I/O can be divided into two subcomponents as:

$$I/O = t_1 + t_2 \qquad (2)$$

where t_1 = non-overlapped I/O time for the data sets we are blocking;

t_2 = non-overlapped I/O time for other data sets.

Now

$$t_1 = (t_{11} + b/TR)E \qquad (3)$$

where t_{11} = the sum of the preparation time for a data transfer to a device. This will include average head movement time and average rotation time. Thus, t_{11} can be considered a constant;

b = blocksize;

TR = transfer rate for the device;

E = number of EXCP's to this data set(s).

Equation (3) states that the total non-overlapped I/O time for the blocked data set(s) is the product of the number of EXCP's and the sum of the time required to do one EXCP. This time equals the preparation time plus the time required to transfer a block of data.

But

$$E = \{BT/b\} \qquad (4)$$

where

BT = total bytes transferred during the program execution;

b = blocksize;

{} = the next greater integer function.

Now, substituting all the above into eq. (1), we have

$$PT = C + (t_{11} + b/TR)*\{BT/b\} + t_2 \qquad (5)$$

3.2 <u>Secondary Storage Utilization</u>

As the second calculation, we define the utilization efficiency, U, of the storage device.

We can begin with the equation:

$$U = \frac{SU}{SA} * 100 \qquad (6)$$

where

SU = space used for program data;

SA = capacity of a device track.

Now

$$SU = [SA/B]*b \qquad (7)$$

where

B = space required for a block of program data;

b = blocksize;

$[\,]$ = greatest integer function.

B can be further defined as

$$B = b + IBG + KL + KO \qquad (8)$$

where

b = blocksize;

IBG = interblock gap;

KL = key length;

KO = key overhead.

KO and KL variables are nontrivial only if the records have keys associated with them as in an indexed sequential file. In this situation each block of data has an additional overhead of one key, of length KL, indicating the last record in the block, and 56 bytes of overhead, KO. If the records have no keys, however, both KL and KO are zero. To express this in PL/1 type statements:

```
If Key Length = 0
    Then Key Overhead = 0;
    Else Key Overhead = 56;
```

Thus the space utilized, SU, can be written as:

$$SU = \left[\frac{SA}{b + IBG + KL + KO}\right] * b \qquad (9)$$

Again, by substitution in eq. (6), we have

$$U = \frac{\left[\frac{SA}{b + IBG + KL + KO}\right] * b}{SA} * 100 \qquad (10)$$

or

$$U = \left[\frac{SA}{b + IBG + KL + KO}\right] \frac{*b}{SA} * 100$$

These two results, eqs. 5 and 10, are theoretically correct. They can now be used to compare with the models resulting from regression tests run on the actual data to determine if any common factors exist. This will be done in Section 7.0, Results.

4.0 Environment

All investigative studies and tests were conducted on an IBM 370/168 computer with four megabytes of real memory and a production size complement of peripheral devices such as disks, drums, tapes, teleprocessing gear and unit record equipment. The operating system is the MVS system from IBM, Release 3.7, with several of the available selectable units. The system normally operates in one of two modes. During prime shift, TSO and IMS are online along with IMS and development batch jobs. Between the hours of 6 p.m. and 6 a.m. production batch jobs for several departmental systems, batch portions of the online systems and system backup and maintenance are performed. This environment provided the data on data set blocking for production and development jobs, along with subsystem components such as compilers, linkage editors, utilities, etc.

5.0 Experiment Design

In the study, we initially had to demonstrate the degree to which blocking was being ignored, and to make some sound hypotheses for what sort of improvement could be expected simply by blocking data. The first step was to watch the system log file for several weeks. A program was designed and implemented to process this file and simply list all data sets accessed along with the record length, blocksize and an indication of how much I/O activity was issued to this file. The I/O activity is simply the number of physical reads or writes to the file (EXCP's). A sample of this output can be seen in Exhibit 1 of the Appendix.

In compiling this data, we found that in several of the production systems, blocking was nonexistent. Other files were blocked with ten records, regardless of the record size. Still others were blocked efficiently for 2314 disks but not for the 3330's we use. This was what had been expected.

We now had to gain a feel for the type of improvement that could be expected by blocking. For this part of the study, several runs were made of various programs representing the typical jobs in the shop. These jobs were run in a stand-alone environment with various blocking factors and results plotted. The four types of jobs were:

1) an IBM utility;
2) a compile;
3) a CPU bound production job;
4) an I/O bound production job.

Four composite plots were then created for the jobs. These were:

1) processing time versus blocking factor;
2) I/O activity versus blocking factor;
3) total CPU time versus blocking factor;
4) secondary storage used versus blocking factor.

At this point, multiple regression tests were run against the data to determine what behavior could be found for each job, and to see if similar behavior was being exhibited by all or certain types of jobs. Finally, an attempt was made to interpret the data in light of the theoretical results of Section 3.

6.0 Data

The data used in the study included the processing time and I/O activity from the standalone runs. The composite plots described in the last section are seen in exhibits 2-5. It is readily apparent that all jobs are affected similarly by blocking factor. The curves are more pronounced than we had anticipated. This indicated that even a very small blocking factor produces significant results. The rate of decrease in processing and CPU time is an inverse relation which is much better than a simple linear relation.

Exhibit 5 illustrates how the number of tracks needed to store the same amount of data decreases as the blocking factor is increased. This general decrease is due to the elimination of the interblock gaps when larger blocksizes are used. However, we see that the decrease is not monotonic but rather a saw-tooth effect with several local minima and maxima. This effect is caused by the requirement that each track of a device contain an integral number of blocks. A simplified example will serve to illustrate this point. If the capacity of a track is one hundred (100) units and a blocksize of twenty (20) units is specified, two hundred (200) units of data may be stored on two tracks. However, if a blocksize of forty (40) is used in the same circumstance, three tracks will be required. This phenomenon causes the saw-tooth graph until the blocking factor results in half track blocking. After that point, only one block will fit on a track and the graph is monotonic decreasing to the point of full track blocking.

We then used this data in a regression analysis, attempting to fit a curve to the observed data. In almost all cases it was observed that a model of:

$$E(Y) = a + \frac{b}{X}$$

provided a very good fit. In this model, Y is the observed dependent value (e.g., processing time, CPU time, I/O activity) and X is the blocking factor. This equation says that processing time, for example, is basically an inverse function of the blocking factor. This one term usually generated an R^2 value of .95 or above, which indicates that the model fits the data quite well. In some cases, we saw that a model of:

$$E(Y) = a + \frac{b}{X^2}$$

was found for processing time. This implies that the processing time decreases at the rate of:

$$\frac{1}{\text{(blocking factor)**2}}$$

An observed excellent fit for I/O activity to the model:

$$E(Y) = a + \frac{b}{X}$$

is understandable since the relation between the number of blocks (i.e., EXCP's) and the blocking factor is nearly an inverse.

The processing time usually had the lowest R^2 of the three regressions for each job. This is also understandable since this value is the least repeatable of the three. Variations are caused by unpredictable supervisory states, especially when in a job mix as opposed to stand-alone.

New data was generated by normalizing figures for processing time, CPU time and I/O activity and summing the three. A summary of the models found to fit this new data for each job type is seen in Figures 1-2.

7.0 Results

The basic result was the achievement, with repeatable, statistically significant data, of the two objectives stated in the introduction. We showed that:

1) Processing time, CPU time and I/O activity can be drastically reduced simply by blocking program data. The rate of decrease was even better than expected, being on the order of

$$\frac{1}{\text{blocking factor}} .$$

2) Secondary storage utilization could be increased by using full-track, half-track or third-track blocking.

The procedure to achieve these goals is not complex but rather simple. In fact, a program can be designed to automate the blocking of temporary and/or new data sets in the system.

It proves interesting, at this point, to investigate the manner in which actual data tracks the theoretical results derived earlier in Section 3.1 of the paper. The two results were calculations for processing time and secondary storage utilization. The equation for processing time, eq. 5, is basically of the form:

$$PT = C + I + O$$

where C = CPU factor;

I = I/O factor for blocked data sets;

O = I/O factor for other data sets.

Now, the BT/b factor in eq. 5 can be rewritten as:

$$(R*LR)/b \text{ or } R*(LR/b)$$

where R = number of records in the data set;

LR = size of each record.

But (LR/b) is nothing more than

$$\frac{1}{\text{blocking factor}}$$

and this was the common factor in all of the regression models shown in Figure 1. The constant in these models can be attributed to the CPU time factor. Thus it would appear that the empirical data correlates very well with our hypothesis, eq. 5.

The second calculated result for secondary storage utilization is seen in eq. 10 in Section 3.2 If we were to substitute actual values for the variables in eq. 10 and construct a plot, it would look very similar to Exhibit 5. As was mentioned in Section 6.0, the saw-tooth effect of these plots is due to the fact that each track must contain an integral number of blocks. This fact is depicted exactly by the greatest integer function found in eq. 10.

Thus not only have we collected data which achieved the two objectives of the study; we also illustrated the high correlation of this data with the derived calculations.

Although this was the primary result, there were two other points worth mentioning. First, the importance and effects of blocking data in MVS is very similar to previous IBM operating systems. Calculations, graphs and so forth tend to substantiate this view.

The second point addresses the question of whether increased blocking continues to provide increased performance. Is full-track blocking always warranted? The response to both of these queries would seem to be negative. By studying the data in exhibits 2-5,

Job Type	Equation	R^2
IBM utility	$0.0803 + \frac{2.8842}{BF}$	.9881
Compile	$1.6968 + \frac{1.2474}{BF}$	.9826
I/O Bound	$0.2560 + \frac{2.6533}{BF}$	.9957
CPU Bound	$0.6434 + \frac{2.663}{BF}$	.9870

Figure 1. One Term Equation Regression Model.

Job Type	Equation	R^2
IBM utility	$0.0976 + \frac{2.8508}{BF} + 0.0\ BF^3$	.9890
Compile	$1.6424 + \frac{1.3465}{BF} + 0.0007\ BF$	.9980
I/O Bound	$0.2922 + \frac{2.1698}{BF} + \frac{0.5339}{BF**2}$	.9997
CPU Bound	$0.6730 + \frac{1.725}{BF} + \frac{0.6243}{BF**2}$	.9946

Figure 2. Two Term Equation Regression Model.

one can easily see the applicability of the '80-20' rule. In other words, 80 percent of the benefits are achieved through 20 percent of the effort, or in this case blocking. For all blockings, the processing time, CPU time and I/O activity all drop sharply with increasing blocking factor to a point. That point appears to be in the area where the blocking factor is around twenty. After that, the decrease in resource utilization is minimal for blocking factors up to full-track. Similarly for space utilization, the number of tracks required drops quickly and then tends to level off. Thus recommendations for blocking factors should probably be in the 10 to 20 range for maximum benefit. In a paging environment, one should also consider the page size of the system. For example, if the page size is 2048 bytes, it would not be profitable to use a blocking factor of 20 if this results in a blocksize of 2400 bytes. Rather, blocksize should be close to, but not larger than, some multiple of the system page size. Keeping this in mind may result in savings due to minimal paging activity.

8.0 Future Work

This study, although already profitable, is far from complete. We hope to develop and implement a resident module and collect data on such measurable items as device utilization, turnaround times, processing time, CPU and I/O activity, and channel activity on packs with scratch space. This resident module would also automatically assign efficient blocking factors to newly created data sets. We would also like to collect data on the job mix effect for our sample jobs and watch the system log for paging, swapping and service rate changes as the blocking factor is varied.

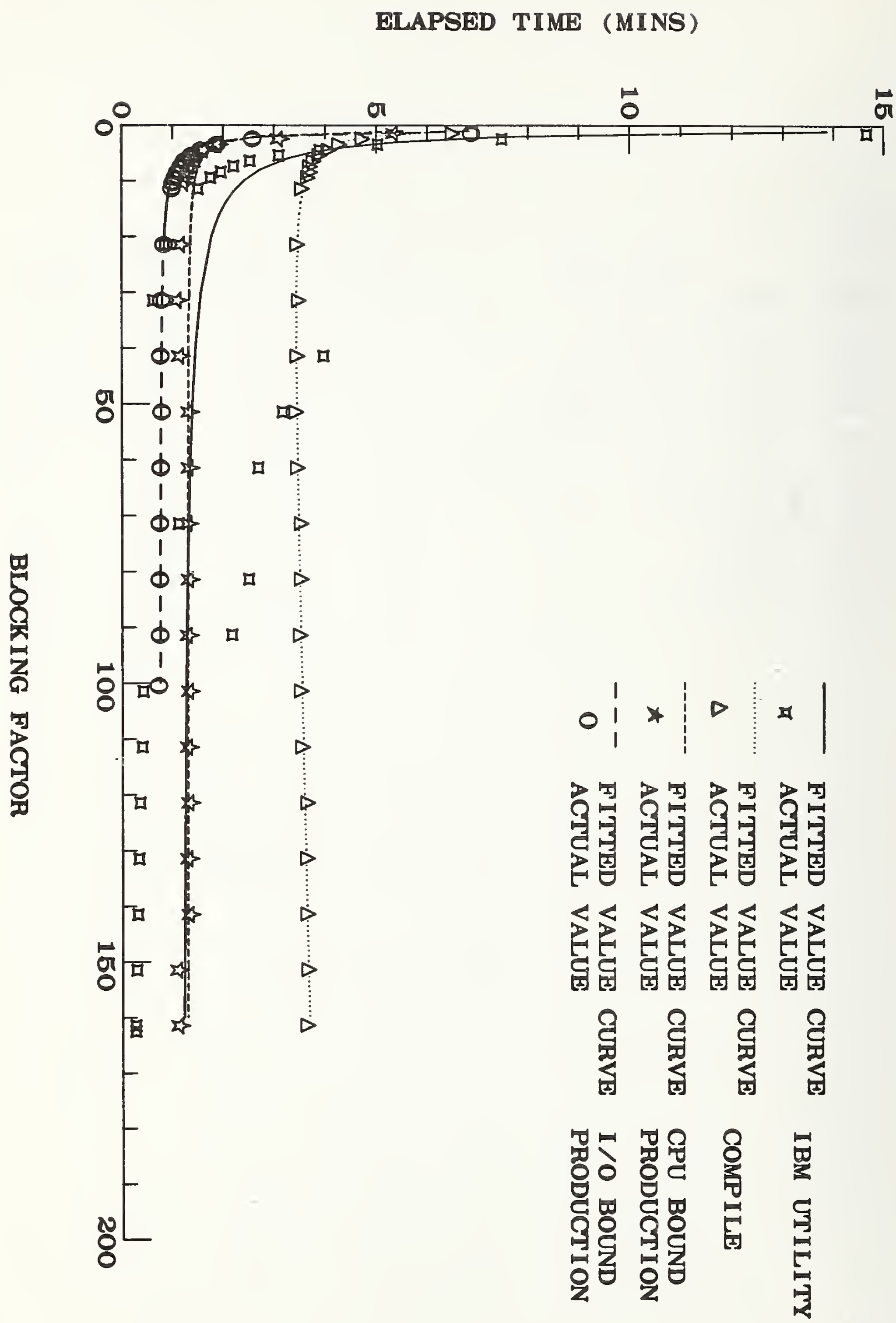
It is felt that substantial savings in resource usage can be obtained on any system through a conscientious program for implementing efficient blocking of program data. We hope to develop and prove the worth of such a program.

APPENDIX

JOBNAME	VOLUME ID	DATASET NAME	RECFM	LRECL	BLOCK SIZE	EXCPs
TL700000	TAST66	MRISDCH.PSB010.DATA	FB	80	1680	36,987
TL700000	TAST66	IMSEAW.WUMPUS.CLIST	VB	255	1680	36,984
TL700000	TAST66	ASCRAP.SL03TSTC.DATA	FBS	97	4850	36,981
TL700000	TAST66	X092476.X151727.SR03NRB	F	80	80	36,971
VR160000	044765	CEN2.VR03C25	VB	3434	6872	933
IMSPROCS	FFFFFF	SYS77021.T063648.RV000.IMSPROCS	F	80	80	137
VR160000	044749	CEN2.VR03B45	VBS	4000	4000	822
TLDMP006	PUB006	TLPUB006	U	00	13030	2,804
VR170000	044737	CEN2.VR03B21	V	3434	3438	6,778
VR170000	044763	CEN2.VR03C21	V	3434	3438	6,762

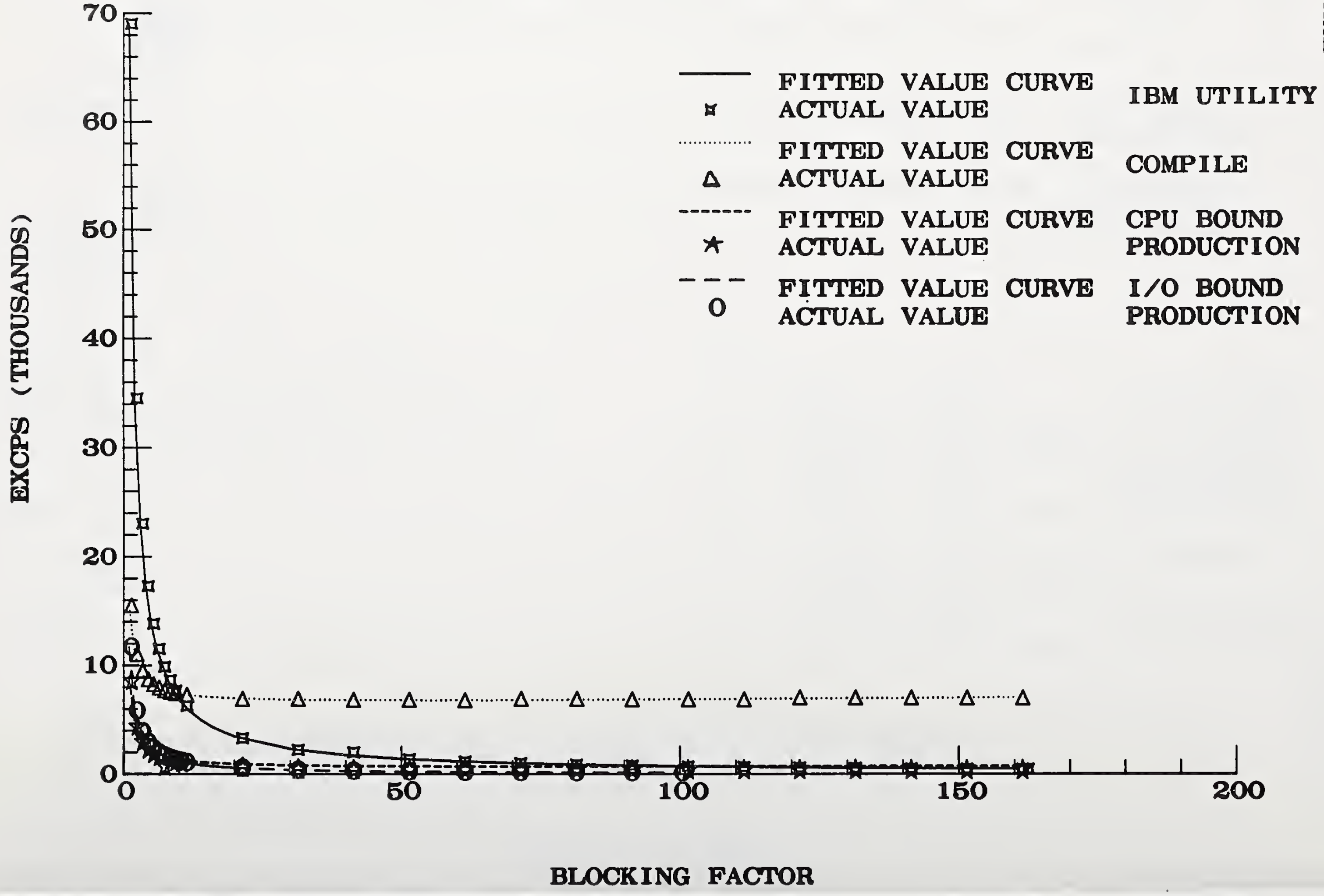
EXHIBIT 1

EXHIBIT 2



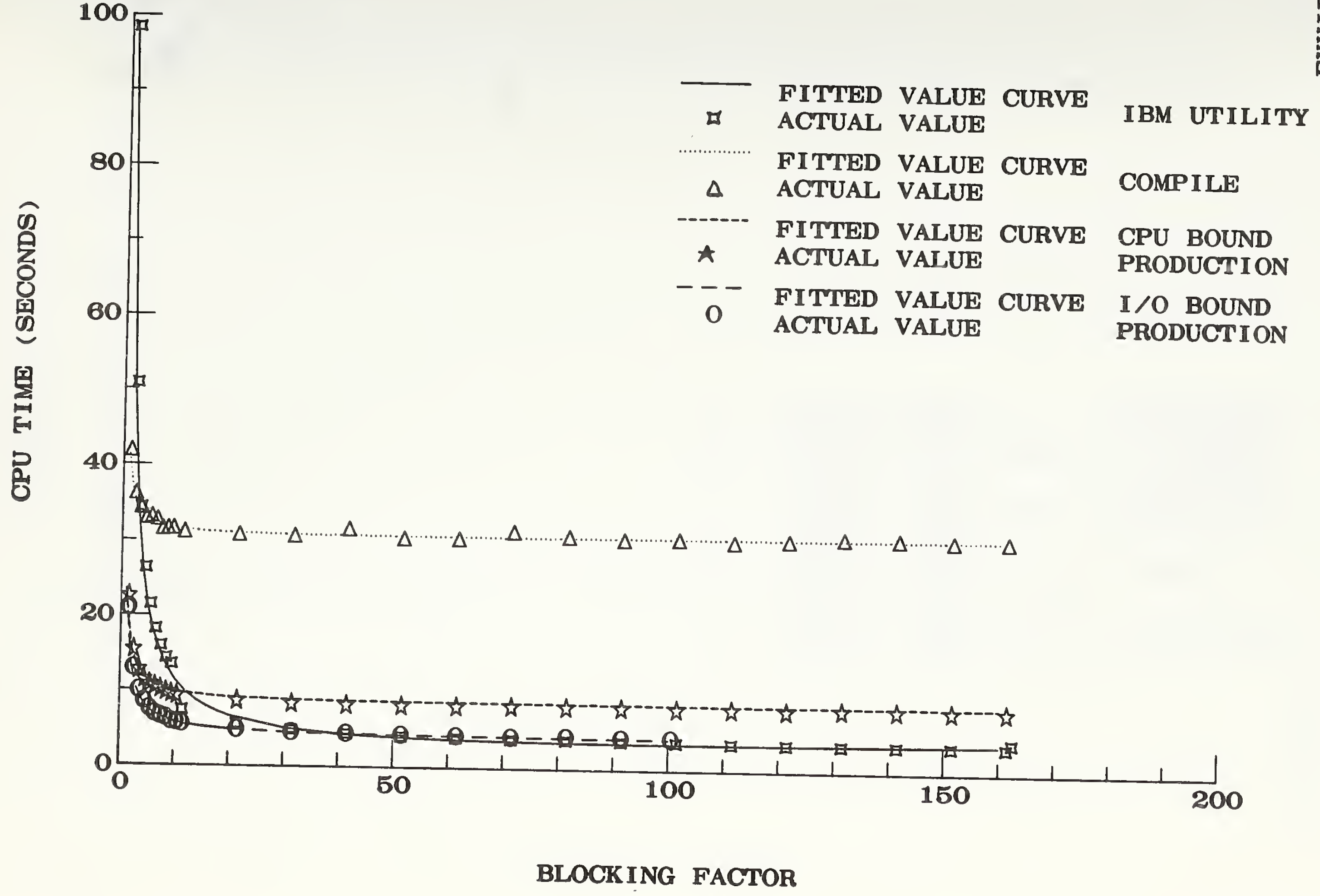
ELAPSED TIME (MINS)
15
10
5
0
0
50
100
150
200
BLOCKING FACTOR
FITTED VALUE CURVE
ACTUAL VALUE IBM UTILITY
FITTED VALUE CURVE
ACTUAL VALUE COMPILE
FITTED VALUE CURVE
ACTUAL VALUE CPU BOUND PRODUCTION
FITTED VALUE CURVE
ACTUAL VALUE I/O BOUND PRODUCTION

EXHIBIT 3



EXCPS (THOUSANDS)
0
10
20
30
40
50
60
70
0
50
100
150
200
BLOCKING FACTOR
FITTED VALUE CURVE
ACTUAL VALUE
IBM UTILITY
FITTED VALUE CURVE
ACTUAL VALUE
COMPILE
FITTED VALUE CURVE
ACTUAL VALUE
CPU BOUND PRODUCTION
FITTED VALUE CURVE
ACTUAL VALUE
I/O BOUND PRODUCTION

EXHIBIT 4



CPU TIME (SECONDS)
100
80
60
40
20
0
0
50
100
150
200
BLOCKING FACTOR
FITTED VALUE CURVE
ACTUAL VALUE
IBM UTILITY
FITTED VALUE CURVE
ACTUAL VALUE
COMPILE
FITTED VALUE CURVE
ACTUAL VALUE
CPU BOUND PRODUCTION
FITTED VALUE CURVE
ACTUAL VALUE
I/O BOUND PRODUCTION

EXHIBIT 5

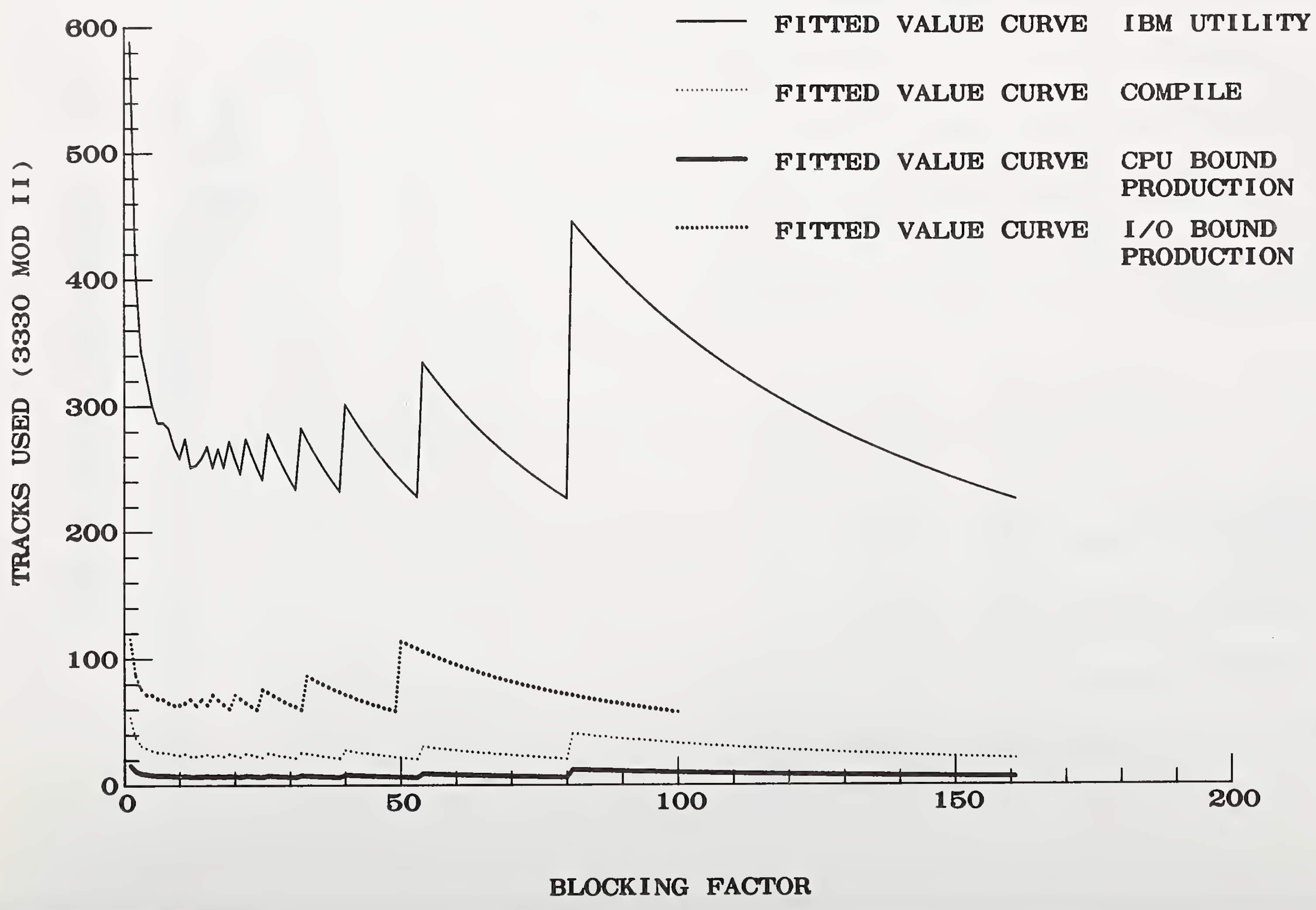
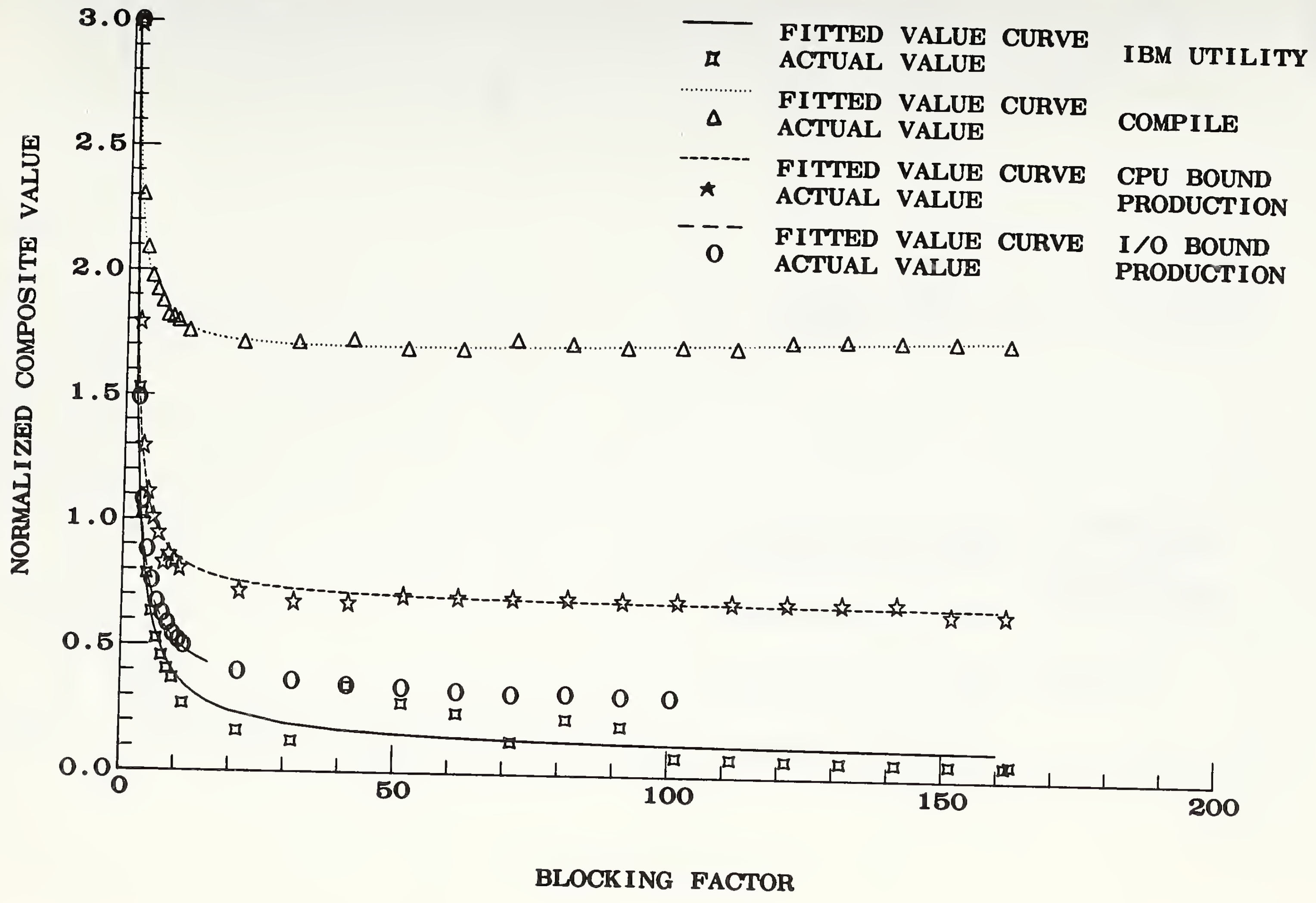

TRACKS USED (3330 MOD II)
600
500
400
300
200
100
0
0
50
100
150
200
BLOCKING FACTOR
FITTED VALUE CURVE IBM UTILITY
FITTED VALUE CURVE COMPILE
FITTED VALUE CURVE CPU BOUND PRODUCTION
FITTED VALUE CURVE I/O BOUND PRODUCTION

EXHIBIT 6



NORMALIZED COMPOSITE VALUE
0.0
0.5
1.0
1.5
2.0
2.5
3.0
0
50
100
150
200
BLOCKING FACTOR
FITTED VALUE CURVE
ACTUAL VALUE
IBM UTILITY
FITTED VALUE CURVE
ACTUAL VALUE
COMPILE
FITTED VALUE CURVE
ACTUAL VALUE
CPU BOUND PRODUCTION
FITTED VALUE CURVE
ACTUAL VALUE
I/O BOUND PRODUCTION

A New Methodology For Computer System Data Gathering

R. A. Orchard

Bell Laboratories
Piscataway, New Jersey 08854

Many computer system monitoring, data gathering, and reduction efforts ignore unbiased sampling techniques. The approaches generally taken are expensive and can make no scientifically based statement about the accuracy of the data gathered or consequent data reduction. The approach outlined in this paper attempts to correct these inadequacies by using the theory of random sampling.

Several new techniques are introduced for obtaining optimal error bounds for estimates of computer system quantities obtained from random samples. A point of view is taken (boolean variable random sampling) which makes it unnecessary to have any a priori knowledge of the population parameters of the phenomena being sampled. It is expected that the techniques introduced will significantly reduce monitoring overhead for computer systems while increasing the quality of the data gathered.

Key words: boolean random sampling; computer system monitoring; data gathering.

I. INTRODUCTION

Large investments of time, hardware, and software are generally required for data gathering and reduction by computer system analysts and those responsible for computer system performance and accountability. This paper presents a rather obvious, but not currently practiced, methodology for data gathering which should significantly reduce the amount of data gathered and increase its quality.

Many data gathering efforts in the computer system area ignore techniques which would yield objectively estimable sampling error. The approaches generally chosen are expensive and can make no scientifically based statement about the accuracy of the data or consequent data reduction. The rationale for the data gathering may not stand on sound objective criteria. The approach outlined in this paper attempts to correct these inadequacies by using random sampling.

The use of random sampling will allow one to justify the cost of a data gathering effort in terms of the quality of the data gathered. It is expected that the implementation of the random sampling techniques outlined will significantly reduce monitoring overhead for computer systems. Several new techniques are introduced for obtaining optimal error bounds for estimates of computer system quantities obtained from random samples.

II. CURRENT SAMPLING APPROACHES TO DATA GATHERING

Current sampling techniques in use in the computer system field today are characterized by one or more of the following:

a. fixed periodic (uniform) sampling rate;

b. an assumption that the underlying phenomenon is random;

c. absence of statistical confidence levels on the quantities measured.

The presence of c. makes us suspect that in many instances the rationale for the data gathering methodology is not motivated by considerations from statistical sampling theory. In these cases, the "Christmas mail" philosophy is probably closer to the rationale; i.e., if we cover the problem with enough data points, we should see some result. In the limit, this technique is guaranteed to work since one has all the points.

Item b. is an interesting note and has some possible overtones of a psychological nature. When human beings are confronted with complexity and cannot discern simple patterns at work, they tend to be borne out in fact. One can mathematically test for "randomness." In other situations, complex patterns and correlations of a deterministic nature are at work and the phenomena cannot be characterized mathematically as random. Unfortunately, more often than not, analysts do not test for randomness.

Others may make distributional assumptions and, if so motivated, proceed to at least a verification that the population reasonably fits the assumed distribution. They then speak about a random variable with the given distribution. Often they are not aware of the fact that one could have a normal or exponential distribution of a variable without that variable being a random variable. If one is working with purely descriptive aspects of the distribution, the misclassification of the variable may not affect the end result of the analysis. If, however, the randomness of the variable is central, serious error may ensue.

It is known that if the underlying phenomenon being sampled is not random then fixed periodic sampling may be a statistically biased sampling technique. The variance associated with it will depend heavily on the deterministic components present in the system data.

Increases in the sampling rate may not decrease sampling variance and may yield unpredictable performance. If the underlying phenomenon is random then fixed periodic sampling is equivalent to the methodology outlined in the next section and the equations presented will hold. In this case, the results of this paper will provide a sound basis for the choice of the sampling rate and a new perspective on variable sampling, namely, representation in terms of boolean variable sampling.

There are individuals with sharp insight into computer systems who may design monitors or other data gathering mechanisms which yield what are considered "good" results. The difficulty here is that the accuracy of results varies from monitor (individual) to monitor (individual), generally bias is present, and *no objective measure of sampling error or bias exists.*

Statistically speaking, as accepted as hardware monitor results are, confidence intervals cannot be placed on their results unless one assumes the data stream being monitored is random, every state change is being recorded, or one has a priori knowledge of the phenomenon being sampled; i.e., by what criteria do we validate a hardware monitor's results? There are examples of hardware monitor results being in error because of synchronization with internal system logic.

The sampling technique we propose for use in computer system data gathering and analysis is characterized by the following:

a. *No Assumptions* concerning the data being monitored or gathered are made. The independence and randomness necessary for objectively estimating sampling error and sample size are provided by the sampling method itself.

b. No a priori knowledge or estimates of actual mean values or standard deviations are required.

c. The accuracy depends on sample size.

d. There is no sample bias.

e. One has some measure of control over the cost of a data gathering effort. Sampling error can be minimized for a given cost or cost for a given sampling error.

f. The approach tends to stimulate a rational, organized process of data gathering.

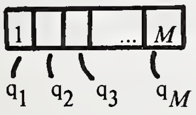
An area where the proposed sampling scheme may be useful is in the collection of system-wide computer system data (e.g., computer networks, corporate or agency collections of data centers and computer systems). In many instances a well controlled sampling effort may turn out to be cheaper and faster than a census or partial census and meet accuracy requirements.

III. BOOLEAN REPRESENTATION OF COMPUTER SYSTEM VARIABLES

It is our conjecture that for a computer system, most of the quantities of interest can be expressed, for sampling purposes, as a suitable function of attribute valued variables. Such variables can be viewed as a boolean variable, i.e., for a given instance of the variable either the attribute is present (=1) or absent (=0). Examples of such variables are any bit positions or combination of bit positions in a computer system to which an attribute has been assigned (CPU busy, channel n busy, task A active, etc.).

Many quantities can be represented as a boolean function of time. If, for example, the quantity of interest is job elapsed time, then the presence or absence of a suitable indication pertaining to a job name in a computer system can be taken as the variable of interest. The number of times the job is found "elapsing" divided by the total number of times the variable was checked in a time period T can be used to estimate the time the job was elapsing in time period T. Similarly, the number of times the CPU busy bit is a one *and* the channel busy bit is a one, divided by the number of times the and'ed condition was checked is an estimator of CPU-channel overlap. In general, extremely complex situations can be modeled from a boolean data gathering perspective. A typical instance of this might be the estimation of average queue length in a particular queue.

Suppose a queue can hold at most M items. To the ith slot on the queue we assign a boolean occupancy variable q_i:



$$q_i = \begin{cases} 0 & \text{if } \textit{slot i is not occupied} \\ 1 & \text{if } \textit{slot i is occupied} \end{cases}$$

If during a period of time T these variables are sampled N times and their averages calculated, we will find

$$\bar{q}_i = \frac{1}{N} \sum_{i=1}^{N} q_i,$$

$\bar{q}_i =$ relative frequency with which a queue length greater than or equal to i elements occurred.

$f_o = 1 - \bar{q}_1 =$ relative frequency with which an empty queue occurred.

$f_i = (\bar{q}_i - \bar{q}_{i+1}) =$ relative frequency with which exactly i elements occur in the queue, $1 \leq i \leq M-1$.

$f_M = \bar{q}_M =$ relative frequency with which exactly M elements occur in the queue.

It is easily seen that:

$$\sum_{i=0}^{M} f_i = 1.$$

Based on a random sample, we have determined the distribution of the elements in the queue.

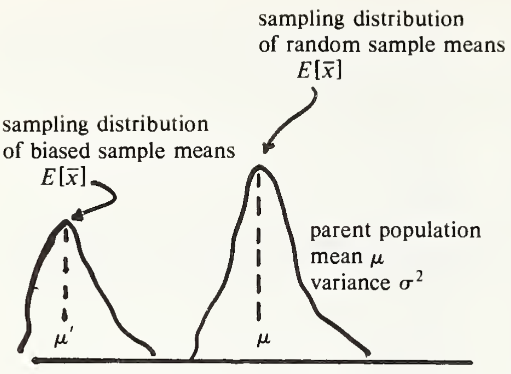
An estimate of the average queue length is:

$$\bar{L} = \sum_{i=1}^{M} i \cdot f_i.$$

All of the quantities given above can be estimated with high mathematical confidence from a sample which, by today's standards, would be a small amount of data gathered. In addition, we have developed techniques which require no a priori estimate of sample average or sample standard deviation and yet yield the "tightest" confidence intervals on the actual occurring estimate of a quantity.

IV. PROPOSED RANDOM SAMPLING APPROACH

A simple random sample is a sample of the parent population in which each element of the population had an equal probability of occurring in the sample. We refer to a simple random sample as a random sample. As a consequence of the Central Limit theorem, it is known that the sampling distribution of the sample mean of a random sample asymptotically approaches the normal distribution with increasing sample size. This property is independent of the properties of the parent population from which the sample is drawn.



Distributions of Random and Biased Sample Means

Figure 1

Figure 1 indicates that the mean of the random sampling distribution is identical to the parent population mean. In other words, if random samples of size n were repeatedly drawn from the same parent population and the sample averages calculated, the random variable $\bar{x}$ would take on different values and the $E[\bar{x}] = \mu$. The variance of the random sampling distribution is $\sigma_{\bar{x}}^2 = \frac{\sigma^2}{n}$, where σ^2 is the parent population variance.

Another sampling technique, which is biased, is also shown in Figure 1. The sample means $\bar{x}$ from repeated samples of size n in this case are not distributed about μ but rather about some point μ'. That is, $E[\bar{x}] = \mu'$. $|\mu' - \mu|$ is the *sampling bias* of the nonrandom sampling technique. Random samples have no sampling bias. If $\bar{x}$ is an instance of a mean calculated from a sample, then $|\bar{x} - E[\bar{x}]|$ is called the *sampling* error. For the random sampling technique, $|\bar{x} - \mu|$ is then the sampling error.

An approximate large sample confidence interval for the true mean of the parent population is given by:

$$\bar{x} - k\sigma_{\bar{x}} < \mu < \bar{x} + k\sigma_{\bar{x}} \tag{1}$$

where k is the σ limit on the standard normal distribution giving 100 (1-α)% confidence.

	Confidence	α
$k = 1.645$	90%	.10
$k = 1.96$	95%	.05
$k = 2.575$	99%	.01
$k = 3.01$	99.9%	.001

Alternately we may write (1) as

$$\mu - k\sigma\bar{x} < \bar{x} < \mu + k\sigma_{\bar{x}} \tag{2}$$

The precision on a sample mean $\bar{x}$ is then

$$\bar{x} \pm k\sigma_{\bar{x}}$$

or since $\sigma_{\bar{x}} = \frac{\sigma}{\sqrt{n}}$,

$$\bar{x} \pm \frac{k\sigma}{\sqrt{n}} \tag{3}$$

σ, the standard deviation of the parent population, is generally unknown and the sample standard deviation

$$s = \sqrt{\frac{\sum_i \left(x_i - \bar{x}\right)^2}{n}}$$

is usually taken as an estimator for σ for large n.

Hence an estimate of the accuracy of $\bar{x}$ as an estimate of μ is:

$$\bar{x} \pm \frac{ks}{\sqrt{n.}}$$

But this is known after the fact (after the sample is taken), and depends on the accuracy of s as an estimator for σ. We do not take this approach.

Suppose we wish to impose an accuracy requirement on our estimate and want the corresponding sample size. If it is required that an absolute tolerance of a units be maintained then from (3):

$$n = \left(\frac{k\sigma}{a}\right)^2 \tag{4}$$

At this point our interest is restricted solely to boolean variables (populations) so that $\bar{x}$ is an *estimator for a proportion (utilization).* In this case if μ is the parent population mean then σ is easily shown to be $\sqrt{\mu(1-\mu)}$.

Equation (4) then becomes,

$$n = \left(\frac{k}{a}\right)^2 \mu(1-\mu). \qquad (5)$$

Since we do not know a priori the actual mean μ of the parent population, we choose the sample size that will guarantee the tolerance a for all μ. The maximum of $\mu(1-\mu)$ occurs at $\mu = .5$, so that:

$$n = \left(\frac{k}{2a}\right)^2 \qquad (6)$$

is the desired sample size.[1] For utilization to be accurate within .01 absolute error at 99% confidence,

$$n = \left(\frac{2.575}{2.(.01)}\right)^2$$

$$= 16577$$

samples must be taken.

If the tolerance a in (5) is not an absolute tolerance but rather a relative tolerance then a is of the form $b\mu$. In this case (5) becomes,

$$n = \left(\frac{k}{b\mu}\right)^2 \mu(1-\mu)$$

$$= \left(\frac{k}{b}\right)^2 \frac{1-\mu}{\mu}. \qquad (5.1)$$

Since $\frac{1-\mu}{\mu}$ is a strictly decreasing function as $\mu \rightarrow 1$, if (5.1) is evaluated at a particular μ^* then the tolerance specified by b will be valid for $\mu \geq \mu^*$. For a utilization to be accurate within 1% relative error for utilization $\geq$ .1 at 99% confidence,

$$n = \left(\frac{2.575}{.01}\right)^2 \frac{(1-.1)}{.1}$$

$$= 596,757$$

samples must be taken. This sample size would produce absurdly low relative error bounds at higher utilization. Fixing n at 596,757 and solving for b as a function of μ in (5.1) we find,

$$b = k\sqrt{\frac{1-\mu}{(596,757)\,\mu}}$$

and the relative errors at 30%, 60% and 90% utilization are respectively $\frac{1}{2}$%, $\frac{3}{10}$% and $\frac{1}{10}$%.

V. CHOOSING ERROR CRITERIA AND DEVELOPING ERROR BOUNDS

Two possible approaches to choosing reasonable error criteria are developed in this section. Both use an absolute error condition.

Method 1

For utilizations within desired ranges, specify the absolute error constraint to be met and the confidence level desired.

interval	*absolute error*	*confidence k value*
$\mu_1 \leq \mu < \mu_2$	a_1	k_1
$\mu_2 \leq \mu < \mu_3$	a_2	k_2
.	.	.
.	.	.
$\mu_j \leq \mu < \mu_{j+1}$	a_j	k_j
.	.	.
.	.	.
$\mu_n \leq \mu \leq \mu_{n+1}$	a_n	k_n

The sample size ensuring all the tolerances will be met at the appropriate confidence levels is given by

$$n = \max\{n_j\}$$

where

$$n_j = \left(\frac{k_j}{2a_j}\right)^2 \mu_j^*(1-\mu_j^*),$$

$$\mu_j^* = l.u.b.\{\mu \mid \mu_j \leqslant \mu \leqslant \mu_{j+1}\} \qquad (7)$$

1. Tables are given in Appendix A.

If more than one variable (utilization) is being estimated, say $\mu^1, \mu^2, \ldots, \mu^m$, then for each μ^i, an n_{μ^i} is calculated using (7). The sample size needed is then:

$$n = \max\left\{ n_{\mu^i} \right\}$$

It is clear that if the sample size so calculated is considered to be too high, then the μ^i yielding the maximum value n_{μ^i} is the candidate for re-evaluation. Returning to (7) the jth interval producing n_{μ^i} can be altered in absolute error or confidence to bring down the sample size. It is possible to automate the choice of an appropriate sample size covering several sample variables. Operations research techniques can be useful here.

Method 2

In this method it is first noted that $\mu = .5$ maximizes the sample size in (5). Hence, if an absolute tolerance of a utilization units is specified, there may be some difficulty in estimating the accuracy at low utilizations. A statement that the tolerance at .02 is $.02 \pm a$ may not be too meaningful if a is .01. However, the sample size calculated in (6) is a worst case sample size since we are assuming no a priori knowledge. If the actual utilization is less than or greater than $\mu = .5$ then more samples than necessary have been taken and one should see a better tolerance than a in effect. Solving for a in (5) yields

$$a = k\sqrt{\frac{\mu(1-\mu)}{n}}. \tag{8}$$

A specification of $a = .01$ for all μ at 99% confidence yields by (6), 16577 samples. Substituting this for n and 2.575 for k in (8) results in an expression for the tolerance as a function of μ at this sample size and confidence.

$$a = .02\sqrt{\mu(1-\mu)} \tag{9}$$

(In general, if a^* is the absolute tolerance at a given k and n^*, then:

$$a = 2a^*\sqrt{\mu(1-\mu)} \tag{9.1}$$

and $a \leq a^*$ for all μ.)

At $\mu = .02$ the tolerance is not .01 but $a = .005$. Rather than make a tolerance statement $\bar{x} \pm .01$ for all sample means, we will derive an error expression as a function of the actual sample estimate obtained.

Sample derived error bounds

Let an absolute error criterion of a^* be imposed for all μ and let n^* be the sample size calculated by (6) then (8) becomes

$$a = k\sqrt{\frac{\mu(1-\mu)}{n^*}} \tag{10}$$

and $a \leq a^*$ for all μ.

Suppose $\bar{x}$ is an actual sample mean obtained from a random sample of size n^* then,

$$\bar{x} - a^* < \mu < \bar{x} + a^*$$

with the confidence level associated with k. If $\bar{x} + a^* < .5$ then an estimator of μ (i.e., $\bar{x}$) cannot be in error by more than a where a is calculated from (10) with $\mu = \bar{x} + a^*$.

If $\bar{x} - a^* > .5$ then an estimator of μ (i.e., $\bar{x}$) cannot be in error by more than a where a is calculated from (10) with $\mu = \bar{x} - a^*$.

If neither of the two conditions hold then μ may take on the value $\mu = .5$. Combining these results we have the following.

If $\bar{x} < .5 - a^*$ then

$$\pm k\sqrt{\frac{(\bar{x}+a^*)(1-(\bar{x}+a^*))}{n^*}} \tag{11}$$

is the appropriate approximate error on $\bar{x}$.

If $\bar{x} > .5 + a^*$ then

$$\pm k\sqrt{\frac{(\bar{x}-a^*)(1-(\bar{x}-a^*)}{n^*}} \tag{11.1}$$

is the appropriate approximate error on $\bar{x}$.

Otherwise the error interval is:

$$\bar{x} \pm a^*.$$

The argument could be repeated on the new confidence interval for $\bar{x}$ given by (11) or (11.1). The process will converge on the tightest error interval that can be placed around $\bar{x}$ [2].

2. Note that we are solving a fixed point problem for a.

One iteration is generally sufficient for most accuracy needs.

At a 99% confidence level with $a^* = .01$, 16577 samples are required. If a sample mean of $\bar{x} = .005$ were recorded, the error interval would be $.005 \pm .01$. Using (11) a refined error interval would be $.005 \pm .002$. If (11) were reapplied using .002 as the a^*, a more refined error interval of $.005 \pm .0016$ would occur.

Another application would show no change in the fourth digit place and the tightest error interval to four places about $\bar{x} = .005$ based on n^* samples is now available. If $\bar{x}$ were .05, one application of (11) yields $.05 \pm .005$ to three places. Reiteration does not change the result to three places.

Combined approaches using both method 1 and method 2 are possible. Rather than tighten an error bound on a particular sample mean $\bar{x}$, one can use an approach similar to method 2 to derive a higher confidence level for a particular sample mean. Solving (10) for k would yield:

$$k = \sqrt{\frac{n^*}{\mu(1-\mu)}}\ a. \qquad (12)$$

If k^* is the k used when n^* was determined for a fixed tolerance a in (6) then for all μ in (12) $k \geq k^*$.

It is also possible to use (11) or (11.1) to drop the error bound on a given sample mean $\bar{x}$ from $\pm a^*$ to $\pm a$. If a is not the minimum error bound achievable, then the unwanted difference between a and the tightest bound can be used to increase the confidence in a. The new value of k can be calculated from (12) with

$$\begin{aligned} \mu &= \bar{x} + a \quad \text{if } \bar{x} < .5 - a \\ \mu &= \bar{x} - a \quad \text{if } \bar{x} > .5 + a \end{aligned} \qquad (13)$$

Alternately, if one simply stays with the error bound a for which n^* was computed, (12) given the substitutions in (13) will calculate the higher confidence levels for utilizations not equal to .5. Clearly, if one is willing to accept a larger error bound and/or confidence level at the 50% utilization level, then a smaller sample size can be used. As the utilization being sampled approaches 0% or 100%, the error bounds and confidence levels improve from those specified at the 50% level.

VI. ENSURING RANDOM SAMPLING

All of the results of the previous sections require that the sampling be carried out in a random manner. Suppose that we have analyzed our problem and, based on cost and accuracy considerations, arrived at a sample size n and a time period T for which results are to be reported.

There are many methods by which one can sample a parent population. The choice and study of a particular method for a given set of circumstances is the subject of sampling theory. Since we have in view the development of the most generally applicable sampling methodology for computer systems, our interest is primarily in those methods which are based ultimately on simple random sampling of boolean variables. This will allow the automation of computer system data gathering and reduction with no assumptions on the underlying computer system phenomena.

Two methods of random sampling are given. The first, simple random sampling, is the method on which all sample sizes, confidence statements, and error bounds mentioned in this paper are based. The second method is stratified sampling with simple random sampling, of sample size one, within each stratum. This technique in general yields higher accuracy than simple random sampling (and in no case lower accuracy). If the second method is used, then the error bounds developed in previous sections are conservative bounds. We make no attempt in this paper to improve on the error bounds in the case where stratified sampling is used.

Simple Random Sampling

Case (i)

Let the parent population size be N with individuals numbered 1 to N.

Let n be the sample size desired. Generate n integer uniform random numbers on the discrete interval $[1,N]$. The n items in the parent population corresponding to the n uniform numbers constitute the random sample.

Case (ii)

Let T be a time interval and $\{x(t) \mid t \in T\}$ the parent population. Let n be the sample size desired. Generate n uniform random numbers in the interval T. The collection $\{x(t') \mid t'$ is one of the generated n random numbers$\}$ is the desired sample of size n.

In both cases, if a duplicate random number is generated it may be discarded in favor of a nonduplicate number (sampling without replacement).

Stratified Sampling With Random Sampling of Size One Within Strata

Case (i)

Let the parent population size be N. Let n be the sample size desired. Form n strata of size N/n. We leave it to the reader to adjudicate a nonintegral division since there are several reasonable ways of handling this situation. Assuming there are now m units in a given stratum, generate an integer random number on $[1,m]$ say i. The ith element of the stratum is placed into the sample. Repeating the procedure for all strata yields the random sample of size n.

Case(ii)

Let T be a time interval and $\{x(t) \mid t \in T\}$ the parent population. Let n be the sample size desired. Form n strata (subintervals) on T, each of length τ. In each stratum generate a random time t′ from a uniform distribution $[0,\tau]$. The point x(t′) is then placed in the sample. Repeating the procedure for all strata yields the desired sample of size n.

Comparison of the Two Sampling Techniques

If a sample must be obtained in time (e.g., we are monitoring a process), then simple random sampling requires that n random points in time be generated, ordered, and stored before the process begins. Hence, additional memory is required for the data gathering effort. Stratified sampling obviates the need for the storage of sample times since the next sampling time can be calculated (generated) "on the fly". There are several subtleties which may have to be considered in any actual implementation of the scheme (e.g., the amount of time needed to generate and sample the next point, etc.).

Previously, it was mentioned that the stratified sampling method yielded in general a lower error on our sample estimates. To see this consider a stratified sample as in the following figure:

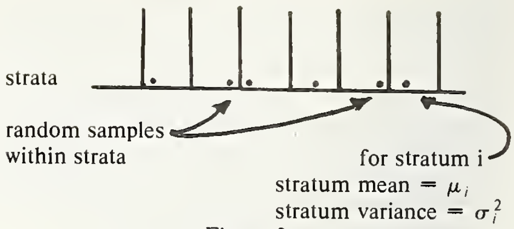


Figure 2

For simple random sampling, we saw that the error bound on the sample mean took the form $k\sigma/\sqrt{n}$ where σ^2 is the population variance. It is known that for the stratified sampling method outlined above, the error bound on the sample mean is

$$\frac{k}{n}\sqrt{\sum_{i=1}^{n}\sigma_i^2}$$

where σ_i^2 is the population variance in the ith stratum.

Let the error bound for simple random sampling be:

$$E_1 = \frac{k}{\sqrt{n}}\cdot\sigma \tag{14}$$

and for stratified sampling:

$$E_2 = \frac{k}{\sqrt{n}} \cdot \frac{\sqrt{\sum_i \sigma_i^2}}{\sqrt{n}}. \tag{15}$$

Let the total parent population size be N. If b_{ij} denotes the jth element in the ith stratum, m elements per stratum with n strata, σ^2 can be written:

$$\begin{aligned}\sigma^2 &= \frac{1}{N}\sum_{i=1}^{n}\sum_{j=1}^{m}\left[(b_{ij}-\mu_i) + (\mu_i-\mu)\right]^2\\ &= \frac{1}{N}\sum_{i=1}^{n}\sum_{j=1}^{m}(b_{ij}-\mu_i)^2 + \frac{1}{N}\sum_{i=1}^{n}\sum_{j=1}^{m}(\mu_i-\mu)^2\\ &= \frac{m}{N}\sum_{i=1}^{n}\sum_{j=1}^{m}\frac{(b_{ij}-\mu_i)^2}{m} + \frac{m}{N}\sum_{i=1}^{n}(\mu_i-\mu)^2.\end{aligned}$$

Hence,

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}\sigma_i^2 + \frac{1}{n}\sum_{i=1}^{n}(\mu_i-\mu)^2$$

$$\geq \frac{1}{n}\sum_{i=1}^{n}\sigma_i^2$$

or

$$\sigma \geq \frac{\sqrt{\sum_i \sigma_i^2}}{\sqrt{n}}$$

equality holding precisely when $\mu_i = \mu$ for all i. That is, when *each and every stratum average is identically the population average.* Hence, from (14) and (15)

$$E_2 \leq E_1$$

Since the condition for equality is a rare event, sampling by stratification is to be preferred where possible. It also represents a methodology similar in form to systematic sampling. Therefore a minimum of work is entailed in converting existing systematic sample based data gathering to a random sample basis.

VII. APPLICATIONS OF BOOLEANIZATION IN COMPUTER SYSTEM AREAS

The use of the boolean variable random sampling techniques outlined in previous sections is of such broad applicability that a complete enumeration of where they may be applied in computer system analysis is precluded. However, whenever these techniques are used to replace current data gathering technique, an increase in data quality and a decrease in cost should ensue. Many quantities are gathered on a census basis (i.e., every value of the variable is captured). A well controlled random sampling plan may be preferred, if the quality of the census data is in question or the census cost too high. This may be especially useful in gathering computer system reliability and benchmarking data.

Computer system variables capable of being randomly sampled, and therefore accurately estimated, include:

- utilizations (channel, CPU, problem program state, etc.)
- average queue lengths and queue distributions
- arrival rates to queues and hence average waiting times
- overlaps between *n* variables
- single thread characteristics of a job run in a multiprogrammed environment
- event triggered monitoring may also be carried out on a sample basis. Based on a random variable, one chooses to trace an event or not.

Census reporting and information gathered to support a variety of computer indices may be gathered at a decrease in cost and possibly an increase in the reliability of the data. Since the method of sampling is random, a certain level of data integrity is insured. Current system wide indices for computer systems should be re-evaluated from the perspective of boolean random sampling.

VIII. CONCLUSION

It is extremely unlikely that current data reduction and reporting facilities available to the reader state confidence intervals on the quantities reported. Unfortunately, sampling theory is often neglected in computer science programs at the university level. Hopefully, this paper will stimulate further investigation of the quality, quantity, and necessity of current data gathering efforts. Computer system overhead for software monitors can be significantly reduced.

APPENDIX

TABLES FOR THE DESIGN OF RANDOM SAMPLING EXPERIMENTS

CONFIDENCE LEVEL

99.0% (K=2.575)

ABSOLUTE ERROR ON QUANTITY	SAMPLE SIZE	AVERAGE INTERSAMPLE TIME FOR REPORT INTERVAL OF				
.01	16577.	1 MIN	5 MIN	10 MIN	30 MIN	60 MIN
		0.004 SEC.	0.018 SEC.	0.036 SEC.	0.109 SEC.	0.217 SEC.
		2 HR	4 HR	8 HR	16 HR	24 HR
		0.007 MIN.	0.014 MIN.	0.029 MIN.	0.058 MIN.	0.087 MIN.
.02	4144.	1 MIN	5 MIN	10 MIN	30 MIN	60 MIN
		0.014 SEC.	0.072 SEC.	0.145 SEC.	0.434 SEC.	0.869 SEC.
		2 HR	4 HR	8 HR	16 HR	24 HR
		0.029 MIN.	0.058 MIN.	0.116 MIN.	0.232 MIN.	0.347 MIN.
.03	1842.	1 MIN	5 MIN	10 MIN	30 MIN	60 MIN
		0.033 SEC.	0.163 SEC.	0.326 SEC.	0.977 SEC.	1.955 SEC.
		2 HR	4 HR	8 HR	16 HR	24 HR
		0.065 MIN.	0.130 MIN.	0.261 MIN.	0.521 MIN.	0.782 MIN.
.04	1036.	1 MIN	5 MIN	10 MIN	30 MIN	60 MIN
		0.058 SEC.	0.290 SEC.	0.579 SEC.	1.737 SEC.	3.475 SEC.
		2 HR	4 HR	8 HR	16 HR	24 HR
		0.116 MIN.	0.232 MIN.	0.463 MIN.	0.927 MIN.	1.390 MIN.
.05	663.	1 MIN	5 MIN	10 MIN	30 MIN	60 MIN
		0.090 SEC.	0.452 SEC.	0.905 SEC.	2.715 SEC.	5.429 SEC.
		2 HR	4 HR	8 HR	16 HR	24 HR
		0.181 MIN.	0.362 MIN.	0.724 MIN.	1.448 MIN.	2.172 MIN.

ABSOLUTE ERROR OF .01 AT 99.0% CONFIDENCE FOR ALL U
YIELDS THE FOLLOWING ERROR BOUND AS A FUNCTION OF U

U	E	U	E	U	E	U	E	U	E
.0	.0	.200	.0080	.400	.0098	.600	.0098	.800	.0080
.005	.0014	.205	.0081	.405	.0098	.605	.0098	.805	.0079
.010	.0020	.210	.0081	.410	.0098	.610	.0098	.810	.0078
.015	.0024	.215	.0082	.415	.0099	.615	.0097	.815	.0078
.020	.0028	.220	.0083	.420	.0099	.620	.0097	.820	.0077
.025	.0031	.225	.0084	.425	.0099	.625	.0097	.825	.0076
.030	.0034	.230	.0084	.430	.0099	.630	.0097	.830	.0075
.035	.0037	.235	.0085	.435	.0099	.635	.0096	.835	.0074
.040	.0039	.240	.0085	.440	.0099	.640	.0096	.840	.0073
.045	.0041	.245	.0086	.445	.0099	.645	.0096	.845	.0072
.050	.0044	.250	.0087	.450	.0099	.650	.0095	.850	.0071
.055	.0046	.255	.0087	.455	.0100	.655	.0095	.855	.0070
.060	.0047	.260	.0088	.460	.0100	.660	.0095	.860	.0069
.065	.0049	.265	.0088	.465	.0100	.665	.0094	.865	.0068
.070	.0051	.270	.0089	.470	.0100	.670	.0094	.870	.0067
.075	.0053	.275	.0089	.475	.0100	.675	.0094	.875	.0066
.080	.0054	.280	.0090	.480	.0100	.680	.0093	.880	.0065
.085	.0056	.285	.0090	.485	.0100	.685	.0093	.885	.0064
.090	.0057	.290	.0091	.490	.0100	.690	.0092	.890	.0063
.095	.0059	.295	.0091	.495	.0100	.695	.0092	.895	.0061
.100	.0060	.300	.0092	.500	.0100	.700	.0092	.900	.0060
.105	.0061	.305	.0092	.505	.0100	.705	.0091	.905	.0059
.110	.0063	.310	.0092	.510	.0100	.710	.0091	.910	.0057
.115	.0064	.315	.0093	.515	.0100	.715	.0090	.915	.0056
.120	.0065	.320	.0093	.520	.0100	.720	.0090	.920	.0054
.125	.0066	.325	.0094	.525	.0100	.725	.0089	.925	.0053
.130	.0067	.330	.0094	.530	.0100	.730	.0089	.930	.0051
.135	.0068	.335	.0094	.535	.0100	.735	.0088	.935	.0049
.140	.0069	.340	.0095	.540	.0100	.740	.0088	.940	.0047
.145	.0070	.345	.0095	.545	.0100	.745	.0087	.945	.0046
.150	.0071	.350	.0095	.550	.0099	.750	.0087	.950	.0044
.155	.0072	.355	.0096	.555	.0099	.755	.0086	.955	.0041
.160	.0073	.360	.0096	.560	.0099	.760	.0085	.960	.0039
.165	.0074	.365	.0096	.565	.0099	.765	.0085	.965	.0037
.170	.0075	.370	.0097	.570	.0099	.770	.0084	.970	.0034
.175	.0076	.375	.0097	.575	.0099	.775	.0084	.975	.0031
.180	.0077	.380	.0097	.580	.0099	.780	.0083	.980	.0028
.185	.0078	.385	.0097	.585	.0099	.785	.0082	.985	.0024
.190	.0078	.390	.0098	.590	.0098	.790	.0081	.990	.0020
.195	.0079	.395	.0098	.595	.0098	.795	.0081	.995	.0014

ABSOLUTE ERROR OF .02 AT 99.0% CONFIDENCE FOR ALL U
YIELDS THE FOLLOWING ERROR BOUND AS A FUNCTION OF U

U	E	U	E	U	E	U	E	U	E
.0	.0	.200	.0160	.400	.0196	.600	.0196	.800	.0160
.005	.0028	.205	.0161	.405	.0196	.605	.0196	.805	.0158
.010	.0040	.210	.0163	.410	.0197	.610	.0195	.810	.0157
.015	.0049	.215	.0164	.415	.0197	.615	.0195	.815	.0155
.020	.0056	.220	.0166	.420	.0197	.620	.0194	.820	.0154
.025	.0062	.225	.0167	.425	.0198	.625	.0194	.825	.0152
.030	.0068	.230	.0168	.430	.0198	.630	.0193	.830	.0150
.035	.0074	.235	.0170	.435	.0198	.635	.0193	.835	.0148
.040	.0078	.240	.0171	.440	.0199	.640	.0192	.840	.0147
.045	.0083	.245	.0172	.445	.0199	.645	.0191	.845	.0145
.050	.0087	.250	.0173	.450	.0199	.650	.0191	.850	.0143
.055	.0091	.255	.0174	.455	.0199	.655	.0190	.855	.0141
.060	.0095	.260	.0175	.460	.0199	.660	.0189	.860	.0139
.065	.0099	.265	.0177	.465	.0200	.665	.0189	.865	.0137
.070	.0102	.270	.0178	.470	.0200	.670	.0188	.870	.0135
.075	.0105	.275	.0179	.475	.0200	.675	.0187	.875	.0132
.080	.0109	.280	.0180	.480	.0200	.680	.0187	.880	.0130
.085	.0112	.285	.0181	.485	.0200	.685	.0186	.885	.0128
.090	.0114	.290	.0182	.490	.0200	.690	.0185	.890	.0125
.095	.0117	.295	.0182	.495	.0200	.695	.0184	.895	.0123
.100	.0120	.300	.0183	.500	.0200	.700	.0183	.900	.0120
.105	.0123	.305	.0184	.505	.0200	.705	.0182	.905	.0117
.110	.0125	.310	.0185	.510	.0200	.710	.0182	.910	.0114
.115	.0128	.315	.0186	.515	.0200	.715	.0181	.915	.0112
.120	.0130	.320	.0187	.520	.0200	.720	.0180	.920	.0109
.125	.0132	.325	.0187	.525	.0200	.725	.0179	.925	.0105
.130	.0135	.330	.0188	.530	.0200	.730	.0178	.930	.0102
.135	.0137	.335	.0189	.535	.0200	.735	.0177	.935	.0099
.140	.0139	.340	.0189	.540	.0199	.740	.0175	.940	.0095
.145	.0141	.345	.0190	.545	.0199	.745	.0174	.945	.0091
.150	.0143	.350	.0191	.550	.0199	.750	.0173	.950	.0087
.155	.0145	.355	.0191	.555	.0199	.755	.0172	.955	.0083
.160	.0147	.360	.0192	.560	.0199	.760	.0171	.960	.0078
.165	.0148	.365	.0193	.565	.0198	.765	.0170	.965	.0074
.170	.0150	.370	.0193	.570	.0198	.770	.0168	.970	.0068
.175	.0152	.375	.0194	.575	.0198	.775	.0167	.975	.0062
.180	.0154	.380	.0194	.580	.0197	.780	.0166	.980	.0056
.185	.0155	.385	.0195	.585	.0197	.785	.0164	.985	.0049
.190	.0157	.390	.0195	.590	.0197	.790	.0163	.990	.0040
.195	.0158	.395	.0196	.595	.0196	.795	.0161	.995	.0028

ABSOLUTE ERROR OF .03 AT 99.0% CONFIDENCE FOR ALL U
YIELDS THE FOLLOWING ERROR BOUND AS A FUNCTION OF U

U	E	U	E	U	E	U	E	U	E
.0	.0	.200	.0240	.400	.0294	.600	.0294	.800	.0240
.005	.0042	.205	.0242	.405	.0295	.605	.0293	.805	.0238
.010	.0060	.210	.0244	.410	.0295	.610	.0293	.810	.0235
.015	.0073	.215	.0246	.415	.0296	.615	.0292	.815	.0233
.020	.0084	.220	.0249	.420	.0296	.620	.0291	.820	.0231
.025	.0094	.225	.0251	.425	.0297	.625	.0290	.825	.0228
.030	.0102	.230	.0252	.430	.0297	.630	.0290	.830	.0225
.035	.0110	.235	.0254	.435	.0297	.635	.0289	.835	.0223
.040	.0118	.240	.0256	.440	.0298	.640	.0288	.840	.0220
.045	.0124	.245	.0258	.445	.0298	.645	.0287	.845	.0217
.050	.0131	.250	.0260	.450	.0298	.650	.0286	.850	.0214
.055	.0137	.255	.0262	.455	.0299	.655	.0285	.855	.0211
.060	.0142	.260	.0263	.460	.0299	.660	.0284	.860	.0208
.065	.0148	.265	.0265	.465	.0299	.665	.0283	.865	.0205
.070	.0153	.270	.0266	.470	.0299	.670	.0282	.870	.0202
.075	.0158	.275	.0268	.475	.0300	.675	.0281	.875	.0198
.080	.0163	.280	.0269	.480	.0300	.680	.0280	.880	.0195
.085	.0167	.285	.0271	.485	.0300	.685	.0279	.885	.0191
.090	.0172	.290	.0272	.490	.0300	.690	.0277	.890	.0188
.095	.0176	.295	.0274	.495	.0300	.695	.0276	.895	.0184
.100	.0180	.300	.0275	.500	.0300	.700	.0275	.900	.0180
.105	.0184	.305	.0276	.505	.0300	.705	.0274	.905	.0176
.110	.0188	.310	.0277	.510	.0300	.710	.0272	.910	.0172
.115	.0191	.315	.0279	.515	.0300	.715	.0271	.915	.0167
.120	.0195	.320	.0280	.520	.0300	.720	.0269	.920	.0163
.125	.0198	.325	.0281	.525	.0300	.725	.0268	.925	.0158
.130	.0202	.330	.0282	.530	.0299	.730	.0266	.930	.0153
.135	.0205	.335	.0283	.535	.0299	.735	.0265	.935	.0148
.140	.0208	.340	.0284	.540	.0299	.740	.0263	.940	.0142
.145	.0211	.345	.0285	.545	.0299	.745	.0262	.945	.0137
.150	.0214	.350	.0286	.550	.0298	.750	.0260	.950	.0131
.155	.0217	.355	.0287	.555	.0298	.755	.0258	.955	.0124
.160	.0220	.360	.0288	.560	.0298	.760	.0256	.960	.0118
.165	.0223	.365	.0289	.565	.0297	.765	.0254	.965	.0110
.170	.0225	.370	.0290	.570	.0297	.770	.0252	.970	.0102
.175	.0228	.375	.0290	.575	.0297	.775	.0251	.975	.0094
.180	.0231	.380	.0291	.580	.0296	.780	.0249	.980	.0084
.185	.0233	.385	.0292	.585	.0296	.785	.0246	.985	.0073
.190	.0235	.390	.0293	.590	.0295	.790	.0244	.990	.0060
.195	.0238	.395	.0293	.595	.0295	.795	.0242	.995	.0042

ABSOLUTE ERROR OF .04 AT 99.0% CONFIDENCE FOR ALL U
YIELDS THE FOLLOWING ERROR BOUND AS A FUNCTION OF U

U	E	U	E	U	E	U	E	U	E
.0	.0	.200	.0320	.400	.0392	.600	.0392	.800	.0320
.005	.0056	.205	.0323	.405	.0393	.605	.0391	.805	.0317
.010	.0080	.210	.0326	.410	.0393	.610	.0390	.810	.0314
.015	.0097	.215	.0329	.415	.0394	.615	.0389	.815	.0311
.020	.0112	.220	.0331	.420	.0395	.620	.0388	.820	.0307
.025	.0125	.225	.0334	.425	.0395	.625	.0387	.825	.0304
.030	.0136	.230	.0337	.430	.0396	.630	.0386	.830	.0301
.035	.0147	.235	.0339	.435	.0397	.635	.0385	.835	.0297
.040	.0157	.240	.0342	.440	.0397	.640	.0384	.840	.0293
.645	.0166	.245	.0344	.445	.0398	.645	.0383	.845	.0290
.050	.0174	.250	.0346	.450	.0398	.650	.0382	.850	.0286
.055	.0182	.255	.0349	.455	.0398	.655	.0380	.855	.0282
.060	.0190	.260	.0351	.460	.0399	.660	.0379	.860	.0278
.065	.0197	.265	.0353	.465	.0399	.665	.0378	.865	.0273
.070	.0204	.270	.0355	.470	.0399	.670	.0376	.870	.0269
.075	.0211	.275	.0357	.475	.0399	.675	.0375	.875	.0265
.080	.0217	.280	.0359	.480	.0400	.680	.0373	.880	.0260
.085	.0223	.285	.0361	.485	.0400	.685	.0372	.885	.0255
.090	.0229	.290	.0363	.490	.0400	.690	.0370	.890	.0250
.095	.0235	.295	.0365	.495	.0400	.695	.0368	.895	.0245
.100	.0240	.300	.0367	.500	.0400	.700	.0367	.900	.0240
.105	.0245	.305	.0368	.505	.0400	.705	.0365	.905	.0235
.110	.0250	.310	.0370	.510	.0400	.710	.0363	.910	.0229
.115	.0255	.315	.0372	.515	.0400	.715	.0361	.915	.0223
.120	.0260	.320	.0373	.520	.0400	.720	.0359	.920	.0217
.125	.0265	.325	.0375	.525	.0399	.725	.0357	.925	.0211
.130	.0269	.330	.0376	.530	.0399	.730	.0355	.930	.0204
.135	.0273	.335	.0378	.535	.0399	.735	.0353	.935	.0197
.140	.0278	.340	.0379	.540	.0399	.740	.0351	.940	.0190
.145	.0282	.345	.0380	.545	.0398	.745	.0349	.945	.0182
.150	.0286	.350	.0382	.550	.0398	.750	.0346	.950	.0174
.155	.0290	.355	.0383	.555	.0398	.755	.0344	.955	.0166
.160	.0293	.360	.0384	.560	.0397	.760	.0342	.960	.0157
.165	.0297	.365	.0385	.565	.0397	.765	.0339	.965	.0147
.170	.0301	.370	.0386	.570	.0396	.770	.0337	.970	.0136
.175	.0304	.375	.0387	.575	.0395	.775	.0334	.975	.0125
.180	.0307	.380	.0388	.580	.0395	.780	.0331	.980	.0112
.185	.0311	.385	.0389	.585	.0394	.785	.0329	.985	.0097
.190	.0314	.390	.0390	.590	.0393	.790	.0326	.990	.0080
.195	.0317	.395	.0391	.595	.0393	.795	.0323	.995	.0056

ABSOLUTE ERROR OF .05 AT 99.0% CONFIDENCE FOR ALL U
YIELDS THE FOLLOWING ERROR BOUND AS A FUNCTION OF U

U	E	U	E	U	E	U	E	U	E
.0	.0	.200	.0400	.400	.0490	.600	.0490	.800	.0400
.005	.0071	.205	.0404	.405	.0491	.605	.0489	.805	.0396
.010	.0099	.210	.0407	.410	.0492	.610	.0488	.810	.0392
.015	.0122	.215	.0411	.415	.0493	.615	.0487	.815	.0388
.020	.0140	.220	.0414	.420	.0494	.620	.0485	.820	.0384
.025	.0156	.225	.0418	.425	.0494	.625	.0484	.825	.0380
.030	.0171	.230	.0421	.430	.0495	.630	.0483	.830	.0376
.035	.0184	.235	.0424	.435	.0496	.635	.0481	.835	.0371
.040	.0196	.240	.0427	.440	.0496	.640	.0480	.840	.0367
.045	.0207	.245	.0430	.445	.0497	.645	.0479	.845	.0362
.050	.0218	.250	.0433	.450	.0497	.650	.0477	.850	.0357
.055	.0228	.255	.0436	.455	.0498	.655	.0475	.855	.0352
.060	.0237	.260	.0439	.460	.0498	.660	.0474	.860	.0347
.065	.0247	.265	.0441	.465	.0499	.665	.0472	.865	.0342
.070	.0255	.270	.0444	.470	.0499	.670	.0470	.870	.0336
.075	.0263	.275	.0477	.475	.0499	.675	.0468	.875	.0331
.080	.0271	.280	.0449	.480	.0500	.670	.0466	.880	.0325
.085	.0279	.285	.0451	.485	.0500	.685	.0465	.885	.0319
.090	.0286	.290	.0454	.490	.0500	.690	.0462	.890	.0313
.095	.0293	.295	.0456	.495	.0500	.695	.0460	.895	.0307
.100	.0300	.300	.0458	.500	.0500	.700	.0458	.900	.0300
.105	.0307	.305	.0460	.505	.0500	.705	.0456	.905	.0293
.110	.0313	.310	.0462	.510	.0500	.710	.0454	.910	.0286
.115	.0319	.315	.0465	.515	.0500	.715	.0451	.915	.0279
.120	.0325	.320	.0466	.520	.0500	.720	.0449	.920	.0271
.125	.0331	.325	.0468	.525	.0499	.725	.0447	.925	.0263
.130	.0336	.330	.0470	.530	.0499	.730	.0444	.930	.0255
.135	.0342	.335	.0472	.535	.0499	.735	.0441	.935	.0247
.140	.0347	.340	.0474	.540	.0498	.740	.0439	.940	.0237
.145	.0352	.345	.0475	.545	.0498	.745	.0436	.945	.0228
.150	.0357	.350	.0477	.550	.0497	.750	.0433	.950	.0218
.155	.0362	.355	.0479	.555	.0497	.755	.0430	.955	.0207
.160	.0367	.360	.0480	.560	.0496	.760	.0427	.960	.0196
.165	.0371	.365	.0481	.565	.0496	.765	.0424	.965	.0184
.170	.0376	.370	.0483	.570	.0495	.770	.0421	.970	.0171
.175	.0380	.375	.0484	.575	.0494	.775	.0418	.975	.0156
.180	.0384	.380	.0485	.580	.0494	.780	.0414	.980	.0140
.185	.0388	.385	.0487	.585	.0493	.785	.0411	.985	.0122
.190	.0392	.390	.0488	.590	.0492	.790	.0407	.990	.0099
.195	.0396	.395	.0489	.595	.0491	.795	.0404	.995	.0071

CONFIDENCE LEVEL

95.0% (K=1.960)

ABSOLUTE ERROR ON QUANTITY	SAMPLE SIZE	AVERAGE INTERSAMPLE TIME FOR REPORT INTERVAL OF				
.01	9604.	1 MIN	5 MIN	10 MIN	30 MIN	60 MIN
		0.006 SEC.	0.031 SEC.	0.062 SEC.	0.187 SEC.	0.375 SEC.
		2 HR	4 HR	8 HR	16 HR	24 HR
		0.012 MIN.	0.025 MIN.	0.050 MIN.	0.100 MIN.	0.150 MIN.
.02	2401.	1 MIN	5 MIN	10 MIN	30 MIN	60 MIN
		0.025 SEC.	0.125 SEC.	0.250 SEC.	0.750 SEC.	1.499 SEC.
		2 HR	4 HR	8 HR	16 HR	24 HR
		0.050 MIN.	0.100 MIN.	0.200 MIN.	0.400 MIN.	0.600 MIN.
.03	1067.	1 MIN	5 MIN	10 MIN	30 MIN	60 MIN
		0.056 SEC.	0.281 SEC.	0.562 SEC.	1.687 SEC.	3.374 SEC.
		2 HR	4 HR	8 HR	16 HR	24 HR
		0.112 MIN.	0.225 MIN.	0.450 MIN.	0.900 MIN.	1.349 MIN.
.04	600.	1 MIN	5 MIN	10 MIN	30 MIN	60 MIN
		0.100 SEC.	0.500 SEC.	1.000 SEC.	2.999 SEC.	5.998 SEC.
		2 HR	4 HR	8 HR	16 HR	24 HR
		0.200 MIN.	0.400 MIN.	0.800 MIN.	1.599 MIN.	2.399 MIN.
.05	384.	1 MIN	5 MIN	10 MIN	30 MIN	60 MIN
		0.156 SEC.	0.781 SEC.	1.562 SEC.	4.686 SEC.	9.371 SEC.
		2 HR	4 HR	8 HR	16 HR	24 HR
		0.312 MIN.	0.625 MIN.	1.249 MIN.	2.499 MIN.	3.748 MIN.

ABSOLUTE ERROR OF .01 AT 95.0% CONFIDENCE FOR ALL U
YIELDS THE FOLLOWING ERROR BOUND AS A FUNCTION OF U

U	E	U	E	U	E	U	E	U	E
.0	.0	.200	.0080	.400	.0098	.600	.0098	.800	.0080
.005	.0014	.205	.0081	.405	.0098	.605	.0098	.805	.0079
.010	.0020	.210	.0081	.410	.0098	.610	.0098	.810	.0078
.015	.0024	.215	.0082	.415	.0099	.615	.0097	.815	.0078
.020	.0028	.220	.0083	.420	.0099	.620	.0097	.820	.0077
.025	.0031	.225	.0084	.425	.0099	.625	.0097	.825	.0076
.030	.0034	.230	.0084	.430	.0099	.630	.0097	.830	.0075
.035	.0037	.235	.0085	.435	.0099	.635	.0096	.835	.0074
.040	.0039	.240	.0085	.440	.0099	.640	.0096	.840	.0073
.045	.0041	.245	.0086	.445	.0099	.645	.0096	.845	.0072
.050	.0044	.250	.0087	.450	.0099	.650	.0095	.850	.0071
.055	.0046	.255	.0087	.455	.0100	.655	.0095	.855	.0070
.060	.0047	.260	.0088	.460	.0100	.660	.0095	.860	.0069
.065	.0049	.265	.0088	.465	.0100	.665	.0094	.865	.0068
.070	.0051	.270	.0089	.470	.0100	.670	.0094	.870	.0067
.075	.0053	.275	.0089	.475	.0100	.675	.0094	.875	.0066
.080	.0054	.280	.0090	.480	.0100	.680	.0093	.880	.0065
.085	.0056	.285	.0090	.485	.0100	.685	.0093	.885	.0064
.090	.0057	.290	.0091	.490	.0100	.690	.0092	.890	.0063
.095	.0059	.295	.0091	.495	.0100	.695	.0092	.895	.0061
.100	.0060	.300	.0092	.500	.0100	.700	.0092	.900	.0060
.105	.0061	.305	.0092	.505	.0100	.705	.0091	.905	.0059
.110	.0063	.310	.0092	.510	.0100	.710	.0091	.910	.0057
.115	.0064	.315	.0093	.515	.0100	.715	.0090	.915	.0056
.120	.0065	.320	.0093	.520	.0100	.720	.0090	.920	.0054
.125	.0066	.325	.0094	.525	.0100	.725	.0089	.925	.0053
.130	.0067	.330	.0094	.530	.0100	.730	.0089	.930	.0051
.135	.0068	.335	.0094	.535	.0100	.735	.0088	.935	.0049
.140	.0069	.340	.0095	.540	.0100	.740	.0088	.940	.0047
.145	.0070	.345	.0095	.545	.0100	.745	.0087	.945	.0046
.150	.0071	.350	.0095	.550	.0099	.750	.0087	.950	.0044
.155	.0072	.355	.0096	.555	.0099	.755	.0086	.955	.0041
.160	.0073	.360	.0096	.560	.0099	.760	.0085	.960	.0039
.165	.0074	.365	.0096	.565	.0099	.765	.0085	.965	.0037
.170	.0075	.370	.0097	.570	.0099	.770	.0084	.970	.0034
.175	.0076	.375	.0097	.575	.0099	.775	.0084	.975	.0031
.180	.0077	.380	.0097	.580	.0099	.780	.0083	.980	.0028
.185	.0078	.385	.0097	.585	.0099	.785	.0082	.985	.0024
.190	.0078	.390	.0098	.590	.0098	.790	.0081	.990	.0020
.195	.0079	.395	.0098	.595	.0098	.795	.0081	.995	.0014

ABSOLUTE ERROR OF .02 AT 95.0% CONFIDENCE FOR ALL U
YIELDS THE FOLLOWING ERROR BOUND AS A FUNCTION OF U

U	E	U	E	U	E	U	E	U	E
.0	.0	.200	.0160	.400	.0196	.600	.0196	.800	.0160
.005	.0028	.205	.0161	.405	.0196	.605	.0196	.805	.0158
.010	.0040	.210	.0163	.410	.0197	.610	.0195	.810	.0157
.015	.0049	.215	.0164	.415	.0197	.615	.0195	.815	.0155
.020	.0056	.220	.0166	.420	.0197	.620	.0194	.820	.0154
.025	.0062	.225	.0167	.425	.0198	.625	.0194	.825	.0152
.030	.0068	.230	.0168	.430	.0198	.630	.0193	.830	.0150
.035	.0074	.235	.0170	.435	.0198	.635	.0193	.835	.0148
.040	.0078	.240	.0171	.440	.0199	.640	.0192	.840	.0147
.045	.0083	.245	.0172	.445	.0199	.645	.0191	.845	.0145
.050	.0087	.250	.0173	.450	.0199	.650	.0191	.850	.0143
.055	.0091	.255	.0174	.455	.0199	.655	.0190	.855	.0141
.060	.0095	.260	.0175	.460	.0199	.660	.0189	.860	.0139
.065	.0099	.265	.0177	.465	.0200	.665	.0189	.865	.0137
.070	.0102	.270	.0178	.470	.0200	.670	.0188	.870	.0135
.075	.0105	.275	.0179	.475	.0200	.675	.0187	.875	.0132
.080	.0109	.280	.0180	.480	.0200	.680	.0187	.880	.0130
.085	.0112	.285	.0181	.485	.0200	.685	.0186	.885	.0128
.090	.0114	.290	.0182	.490	.0200	.690	.0185	.890	.0125
.095	.0117	.295	.0182	.495	.0200	.695	.0184	.895	.0123
.100	.0120	.300	.0183	.500	.0200	.700	.0183	.900	.0120
.105	.0123	.305	.0184	.505	.0200	.705	.0182	.905	.0117
.110	.0125	.310	.0185	.510	.0200	.710	.0182	.910	.0114
.115	.0128	.315	.0186	.515	.0200	.715	.0181	.915	.0112
.120	.0130	.320	.0187	.520	.0200	.720	.0180	.920	.0109
.125	.0132	.325	.0187	.525	.0200	.725	.0179	.925	.0105
.130	.0135	.330	.0188	.530	.0200	.730	.0178	.930	.0102
.135	.0137	.335	.0189	.535	.0200	.735	.0177	.935	.0099
.140	.0139	.340	.0189	.540	.0199	.740	.0175	.940	.0095
.145	.0141	.345	.0190	.545	.0199	.745	.0174	.945	.0091
.150	.0143	.350	.0191	.550	.0199	.750	.0173	.950	.0087
.155	.0145	.355	.0191	.555	.0199	.755	.0172	.955	.0083
.160	.0147	.360	.0192	.560	.0199	.760	.0171	.960	.0078
.165	.0148	.365	.0193	.565	.0198	.765	.0170	.965	.0074
.170	.0150	.370	.0193	.570	.0198	.770	.0168	.970	.0068
.175	.0152	.375	.0194	.575	.0198	.775	.0167	.975	.0062
.180	.0154	.380	.0194	.580	.0197	.780	.0166	.980	.0056
.185	.0155	.385	.0195	.585	.0197	.785	.0164	.985	.0049
.190	.0157	.390	.0195	.590	.0197	.790	.0163	.990	.0040
.195	.0158	.395	.0196	.595	.0196	.795	.0161	.995	.0028

ABSOLUTE ERROR OF .03 AT 95.0% CONFIDENCE FOR ALL U
YIELDS THE FOLLOWING ERROR BOUND AS A FUNCTION OF U

U	E	U	E	U	E	U	E	U	E
.0	.0	.200	.0240	.400	.0294	.600	.0294	.800	.0240
.005	.0042	.205	.0242	.405	.0295	.605	.0293	.805	.0238
.010	.0060	.210	.0244	.410	.0295	.610	.0293	.810	.0235
.015	.0073	.215	.0246	.415	.0296	.615	.0292	.815	.0233
.020	.0084	.220	.0249	.420	.0296	.620	.0291	.820	.0231
.025	.0094	.225	.0251	.425	.0297	.625	.0290	.825	.0228
.030	.0102	.230	.0252	.430	.0297	.630	.0290	.830	.0225
.035	.0110	.235	.0254	.435	.0297	.635	.0289	.835	.0223
.040	.0118	.240	.0256	.440	.0298	.640	.0288	.840	.0220
.045	.0124	.245	.0258	.445	.0298	.645	.0287	.845	.0217
.050	.0131	.250	.0260	.450	.0298	.650	.0286	.850	.0214
.055	.0137	.255	.0262	.455	.0299	.655	.0285	.855	.0211
.060	.0142	.260	.0263	.460	.0299	.660	.0284	.860	.0208
.065	.0148	.265	.0265	.465	.0299	.665	.0283	.865	.0205
.070	.0153	.270	.0266	.470	.0299	.670	.0282	.870	.0202
.075	.0158	.275	.0268	.475	.0300	.675	.0281	.875	.0198
.080	.0163	.280	.0269	.480	.0300	.680	.0280	.880	.0195
.085	.0167	.285	.0271	.485	.0300	.685	.0279	.885	.0191
.090	.0172	.290	.0272	.490	.0300	.690	.0277	.890	.0188
.095	.0176	.295	.0274	.495	.0300	.695	.0276	.895	.0184
.100	.0180	.300	.0275	.500	.0300	.700	.0275	.900	.0180
.105	.0184	.305	.0276	.505	.0300	.705	.0274	.905	.0176
.110	.0188	.310	.0277	.510	.0300	.710	.0272	.910	.0172
.115	.0191	.315	.0279	.515	.0300	.715	.0271	.915	.0167
.120	.0195	.320	.0280	.520	.0300	.720	.0269	.920	.0163
.125	.0198	.325	.0281	.525	.0300	.725	.0268	.925	.0158
.130	.0202	.330	.0282	.530	.0299	.730	.0266	.930	.0153
.135	.0205	.335	.0283	.535	.0299	.735	.0265	.935	.0148
.140	.0208	.340	.0284	.540	.0299	.740	.0263	.940	.0142
.145	.0211	.345	.0285	.545	.0299	.745	.0262	.945	.0137
.150	.0214	.350	.0286	.550	.0298	.750	.0260	.950	.0131
.155	.0217	.355	.0287	.555	.0298	.755	.0258	.955	.0124
.160	.0220	.360	.0288	.560	.0298	.760	.0256	.960	.0118
.165	.0223	.365	.0289	.565	.0297	.765	.0254	.965	.0110
.170	.0225	.370	.0290	.570	.0297	.770	.0252	.970	.0102
.175	.0228	.375	.0290	.575	.0297	.775	.0251	.975	.0094
.180	.0231	.380	.0291	.580	.0296	.780	.0249	.980	.0084
.185	.0233	.385	.0292	.585	.0296	.785	.0246	.985	.0073
.190	.0235	.390	.0293	.590	.0295	.790	.0244	.990	.0060
.195	.0238	.395	.0293	.595	.0295	.795	.0242	.995	.0042

ABSOLUTE ERROR OF .04 AT 95.0% CONFIDENCE FOR ALL U
YIELDS THE FOLLOWING ERROR BOUND AS A FUNCTION OF U

U	E	U	E	U	E	U	E	U	E
.0	.0	.200	.0320	.400	.0392	.600	.0392	.800	.0320
.005	.0056	.205	.0323	.405	.0393	.605	.0391	.805	.0317
.010	.0080	.210	.0326	.410	.0393	.610	.0390	.810	.0314
.015	.0097	.215	.0329	.415	.0394	.615	.0389	.815	.0311
.020	.0112	.220	.0331	.420	.0395	.620	.0388	.820	.0307
.025	.0125	.225	.0334	.425	.0395	.625	.0387	.825	.0304
.030	.0136	.230	.0337	.430	.0396	.630	.0386	.830	.0301
.035	.0147	.235	.0339	.435	.0397	.635	.0385	.835	.0297
.040	.0157	.240	.0342	.440	.0397	.640	.0384	.840	.0293
.645	.0166	.245	.0344	.445	.0398	.645	.0383	.845	.0290
.050	.0174	.250	.0346	.450	.0398	.650	.0382	.850	.0286
.055	.0182	.255	.0349	.455	.0398	.655	.0380	.855	.0282
.060	.0190	.260	.0351	.460	.0399	.660	.0379	.860	.0278
.065	.0197	.265	.0353	.465	.0399	.665	.0378	.865	.0273
.070	.0204	.270	.0355	.470	.0399	.670	.0376	.870	.0269
.075	.0211	.275	.0357	.475	.0399	.675	.0375	.875	.0265
.080	.0217	.280	.0359	.480	.0400	.680	.0373	.880	.0260
.085	.0223	.285	.0361	.485	.0400	.685	.0372	.885	.0255
.090	.0229	.290	.0363	.490	.0400	.690	.0370	.890	.0250
.095	.0235	.295	.0365	.495	.0400	.695	.0368	.895	.0245
.100	.0240	.300	.0367	.500	.0400	.700	.0367	.900	.0240
.105	.0245	.305	.0368	.505	.0400	.705	.0365	.905	.0235
.110	.0250	.310	.0370	.510	.0400	.710	.0363	.910	.0229
.115	.0255	.315	.0372	.515	.0400	.715	.0361	.915	.0223
.120	.0260	.320	.0373	.520	.0400	.720	.0359	.920	.0217
.125	.0265	.325	.0375	.525	.0399	.725	.0357	.925	.0211
.130	.0269	.330	.0376	.530	.0399	.730	.0355	.930	.0204
.135	.0273	.335	.0378	.535	.0399	.735	.0353	.935	.0197
.140	.0278	.340	.0379	.540	.0399	.740	.0351	.940	.0190
.145	.0282	.345	.0380	.545	.0398	.745	.0349	.945	.0182
.150	.0286	.350	.0382	.550	.0398	.750	.0346	.950	.0174
.155	.0290	.355	.0383	.555	.0398	.755	.0344	.955	.0166
.160	.0293	.360	.0384	.560	.0397	.760	.0342	.960	.0157
.165	.0297	.365	.0385	.565	.0397	.765	.0339	.965	.0147
.170	.0301	.370	.0386	.570	.0396	.770	.0337	.970	.0136
.175	.0304	.375	.0387	.575	.0395	.775	.0334	.975	.0125
.180	.0307	.380	.0388	.580	.0395	.780	.0331	.980	.0112
.185	.0311	.385	.0389	.585	.0394	.785	.0329	.985	.0097
.190	.0314	.390	.0390	.590	.0393	.790	.0326	.990	.0080
.195	.0317	.395	.0391	.595	.0393	.795	.0323	.995	.0056

ABSOLUTE ERROR OF .05 AT 95.0% CONFIDENCE FOR ALL U
YIELDS THE FOLLOWING ERROR BOUND AS A FUNCTION OF U

U	E	U	E	U	E	U	E	U	E
.0	.0	.200	.0400	.400	.0490	.600	.0490	.800	.0400
.005	.0071	.205	.0404	.405	.0491	.605	.0489	.805	.0396
.010	.0099	.210	.0407	.410	.0492	.610	.0488	.810	.0392
.015	.0122	.215	.0411	.415	.0493	.615	.0487	.815	.0388
.020	.0140	.220	.0414	.420	.0494	.620	.0485	.820	.0384
.025	.0156	.225	.0418	.425	.0494	.625	.0484	.825	.0380
.030	.0171	.230	.0421	.430	.0495	.630	.0483	.830	.0376
.035	.0184	.235	.0424	.435	.0496	.635	.0481	.835	.0371
.040	.0196	.240	.0427	.440	.0496	.640	.0480	.840	.0367
.045	.0207	.245	.0430	.445	.0497	.645	.0479	.845	.0362
.050	.0218	.250	.0433	.450	.0497	.650	.0477	.850	.0357
.055	.0228	.255	.0436	.455	.0498	.655	.0475	.855	.0352
.060	.0237	.260	.0439	.460	.0498	.660	.0474	.860	.0347
.065	.0247	.265	.0441	.465	.0499	.665	.0472	.865	.0342
.070	.0255	.270	.0444	.470	.0499	.670	.0470	.870	.0336
.075	.0263	.275	.0477	.475	.0499	.675	.0468	.875	.0331
.080	.0271	.280	.0449	.480	.0500	.670	.0466	.880	.0325
.085	.0279	.285	.0451	.485	.0500	.685	.0465	.885	.0319
.090	.0286	.290	.0454	.490	.0500	.690	.0462	.890	.0313
.095	.0293	.295	.0456	.495	.0500	.695	.0460	.895	.0307
.100	.0300	.300	.0458	.500	.0500	.700	.0458	.900	.0300
.105	.0307	.305	.0460	.505	.0500	.705	.0456	.905	.0293
.110	.0313	.310	.0462	.510	.0500	.710	.0454	.910	.0286
.115	.0319	.315	.0465	.515	.0500	.715	.0451	.915	.0279
.120	.0325	.320	.0466	.520	.0500	.720	.0449	.920	.0271
.125	.0331	.325	.0468	.525	.0499	.725	.0447	.925	.0263
.130	.0336	.330	.0470	.530	.0499	.730	.0444	.930	.0255
.135	.0342	.335	.0472	.535	.0499	.735	.0441	.935	.0247
.140	.0347	.340	.0474	.540	.0498	.740	.0439	.940	.0237
.145	.0352	.345	.0475	.545	.0498	.745	.0436	.945	.0228
.150	.0357	.350	.0477	.550	.0497	.750	.0433	.950	.0218
.155	.0362	.355	.0479	.555	.0497	.755	.0430	.955	.0207
.160	.0367	.360	.0480	.560	.0496	.760	.0427	.960	.0196
.165	.0371	.365	.0481	.565	.0496	.765	.0424	.965	.0184
.170	.0376	.370	.0483	.570	.0495	.770	.0421	.970	.0171
.175	.0380	.375	.0484	.575	.0494	.775	.0418	.975	.0156
.180	.0384	.380	.0485	.580	.0494	.780	.0414	.980	.0140
.185	.0388	.385	.0487	.585	.0493	.785	.0411	.985	.0122
.190	.0392	.390	.0488	.590	.0492	.790	.0407	.990	.0099
.195	.0396	.395	.0489	.595	.0491	.795	.0404	.995	.0071

E. PREDICTION METHODS

THE USE OF A VALIDATED EVENT MODEL IN A COMPREHENSIVE PERFORMANCE EVALUATION OF AN ON-LINE MINICOMPUTER SYSTEM

S. G. Gangwere Jr., J. R. Hosler, L. H. Stewart

TRW
Hawthorne, CA

A performance evaluation technique is described that has been successfully applied to several on-line systems implemented on a NOVA minicomputer with a disk file. The technique is based on the use of a system event model, validated by direct, hardware-assisted measurement of system behavior. The event model is necessary for successful evaluation of system limits and for prediction of the effect of various design changes on system behavior. The fidelity required of the model is such that the hardware and software actions of the disk file system must be simulated in detail; considerations of disk modelling are examined.

Key words: Credit authorization system; critical resource; disk modelling; event model; resource dependency; measurement; modelling; system capacity; validation.

1. Introduction

1.1 Objectives and Scope

This paper describes a technique used in the detailed analysis of the performance characteristics of TRW credit authorization systems. These systems are real-time, on-line transaction processing communication systems, implemented on a minicomputer with a moving-arm disk. The minicomputer is responsible for the management of the communication network and the disk files, as well as for the execution of the credit authorization algorithm.

System performance evaluation combines analysis, measurement, and predictive event modelling. Each technique is examined separately for its utility and limitations as an evaulation tool. Special emphasis is given to the use of an event modelling technique that draws on measurement techniques for validation. An illustrated example shows the application of the modelling technique to a specific credit authorization system, and the conclusions drawn from the use of a validated model of the system.

1.2 The Performance Evaluation Program

TRW Communication Systems and Services has an active Performance Evaluation Program, charged with detailed performance analysis of all the minicomputer and microprocessor-based systems in TRW's retail and financial product lines, including the credit authorization systems. The Performance Evaluation Program has been underway since early 1976. During that time, it has performed detailed evaluations of six systems. The program is successful from a business viewpoint as well as from a technical one.

1.3 The Study and its Objectives

The study exemplified in this paper was performed during the summer of 1976. The study objective was simple but comprehensive: to discover the operating limit of the system, that is, the transaction throughput capacity. The preliminary results showed that the throughput capacity was unacceptably low; consequently, the study was broadened to include an investigation of design alternatives that might improve system throughput.

The study resulted in a thorough understanding of the operating behavior of the system. The system event model has subsequently been used to examine design changes and further refinements to the same system; a contractual performance level commitment has been made by TRW based on the study results.

2. System Description

The system studied is a typical department store credit authorization system; the structure of the system is shown in figure 1.

The communication lines are interfaced to the minicomputer by a conventional teleprocessing interface device, which delivers one interrupt to the processor for each character. The disk is interfaced by a conventional direct-memory-access ("DMA") controller, which permits overlapped seek operations on the drives.

The keyboard inquiry terminals are connected to the half-duplex, multidrop communication lines via communications interface ("CI") devices located in the stores. The terminals themselves are fully buffered. Figure 2 shows a segment of a typical communication line.

A "breakaway" communications strategy is used. When an inquiry message is complete in the terminal's buffer, the operator presses the "send" key at the terminal. The CI is constantly scanning keypads; when a send key is noted, the CI stops scanning, locks onto that terminal, and awaits a "poll" message from the central system. When one is received, the CI routes the message from the terminal onto the line and "breaks away" to resume scanning. When the system response is transmitted to the CI, it contains the terminal address as well as the CI address.

2.1 Software Overview

Within the minicomputer, the software is responsible for supervising the communications process. An inquiry message normally includes at least an inquiry type, an account number, and a purchase amount. When such an inquiry message is completely assembled, the system accesses the disk file to obtain the account record that is the subject of the inquiry. A decision is made to OK or reject credit. In the normal "OK" case, the system must update and rewrite the disk file, and transmit the appropriate message to the originating terminal. If the credit request is not automatically accepted, the system transmits full particulars of the situation to an authorization clerk at a CRT terminal for a manual decision. At the conclusion of any transaction, an audit trail record is written to magnetic tape. Figure 3 shows the processing sequence.

2.2 Disk Subsystem

The disk files in the subject system are hash-organized, with a single level of disk-resident index (see figure 4).

The program accesses the index file by a conventional hash technique. The index file contains a pointer to every record in the data file. This organization preserves disk space, by permitting dense packing within the large data file. It requires a minimum of two disk accesses per transaction.

2.3 Communications Subsystem

The communication manager within the system is driven by periodic interrupts. Its execution is transparent to the application program shown previously. Essentially, a program of the form shown in figure 5 is attached to each active line.

One significant feature of this picture is that the communication line to which the program is attached becomes idle while the program is "getting" a communications buffer. This has a significant effect on system performance, and must be accurately simulated by the system model.

3. Performance Evaluation Techniques

One of the results of the 18-month performance evaluation experience is a clear understanding of the applicability and scope of the three fundamentally different evaluation techniques and tools: analysis, measurement, and event modelling. The material below summarizes the utility of each to the performance program and outlines the way in which they are combined to effect a comprehensive understanding of a system.

3.1 Analysis

Conventional queuing analysis is adequate for a reasonably accurate description of the behavior of the communication lines, terminals, and other equipment, under a variety of load conditions. TRW has successfully sized communications configurations for some years, by the application of a standard analytic model. Queuing theory also provides a description of the behavior of hashed files, and offers rules of thumb for the evaluation of the observed behavior of other queued facilities. Additionally, analytic approximations of the behavior of various system elements are included in the system event models.

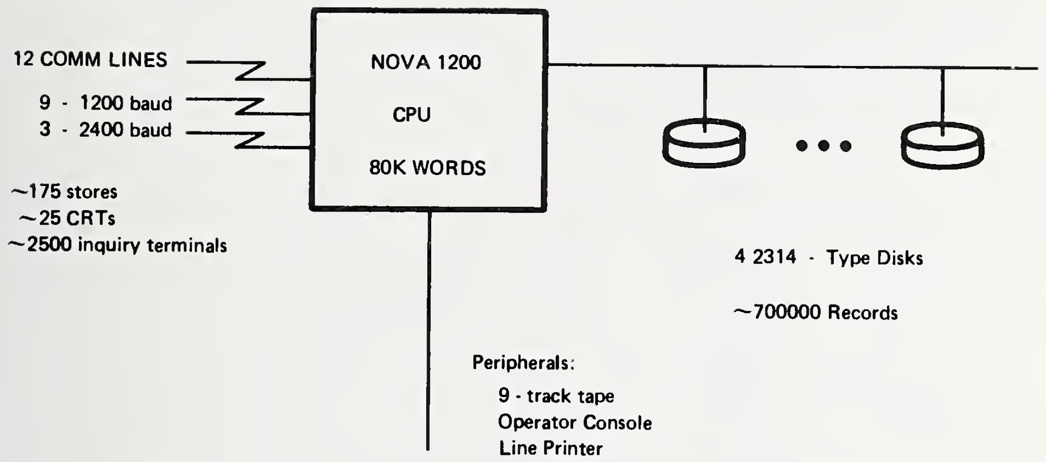


Figure 1. Structure of Study System

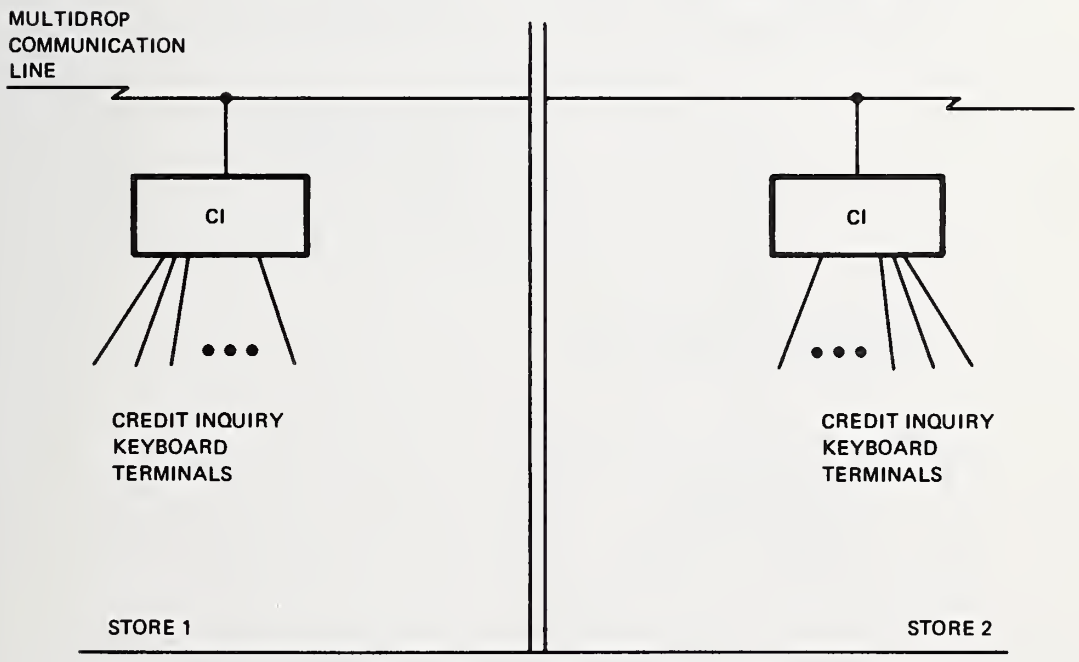


Figure 2. Segment of Typical Communication Line

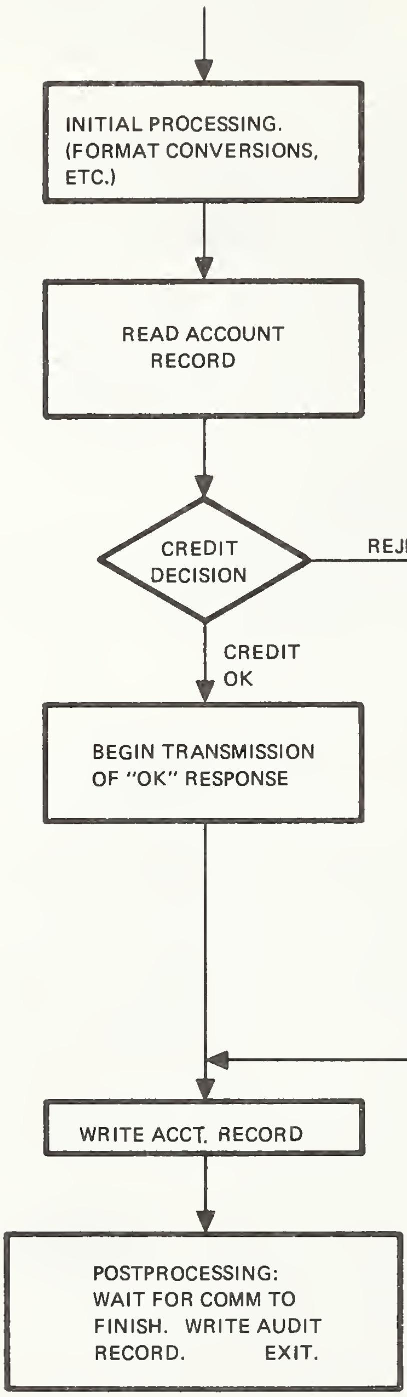

INITIAL PROCESSING. (FORMAT CONVERSIONS, ETC.)
READ ACCOUNT RECORD
CREDIT DECISION
REJECT
CREDIT OK
BEGIN TRANSMISSION OF "OK" RESPONSE
BEGIN TRANSMISSION OF "CALL CREDIT OFFICE" RESPONSE
BEGIN TRANSMISSION TO AUTHORIZATION CLERK
WRITE ACCT. RECORD
POSTPROCESSING: WAIT FOR COMM TO FINISH. WRITE AUDIT RECORD. EXIT.

Figure 3. Simplified Transaction Processing Sequence

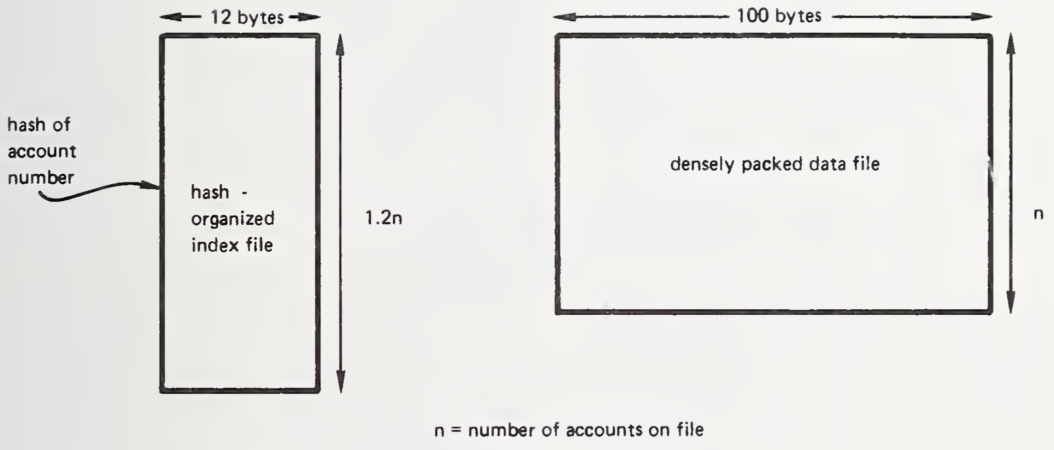


Figure 4. Disk File Organization

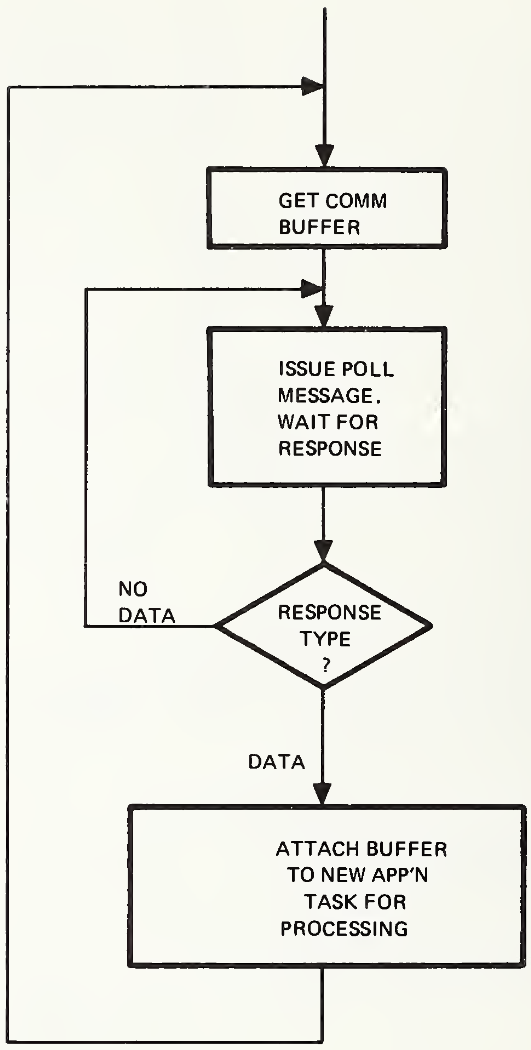

GET COMM BUFFER
ISSUE POLL MESSAGE. WAIT FOR RESPONSE
RESPONSE TYPE ?
NO DATA
DATA
ATTACH BUFFER TO NEW APP'N TASK FOR PROCESSING

Figure 5. Communication Line Driver Program

Analysis, however, leaves us short of our goal of determining the response characteristics of each system, under varying loads. The physical and logical elements that contribute to this response characteristic include (and are connected by) a relatively complex software algorithm -- far too complex for accurate analytic modelling within our budget and schedule constraints.

Figure 6 illustrates this point. It shows a simplified version of the basic flow for a single transaction, relating it to the use of the major system resources -- the CPU, the disk system components, the disk buffer pool, and the communications buffer pool. The response time as a function of load in the subsystem shown is highly complex, because the availability of the different resources is influenced by the availability and servicing times of each of the others. Because the CPU resource is acquired and dismissed several times during the service time of the disk buffer, for example, its waiting and service times are reflected in the service time of the disk buffer. The same interaction occurs in the case of the communications buffer.

The communications buffer also affects the basic service rate of the CPU resource. This results from the unique relationship between the communications buffer pool and the polling function. When the buffer pool is exhausted, the polling function for a line requires nonzero time to acquire a new communications buffer. During this time, the line lies fallow, contributing no interrupts. Consequently, the total CPU overhead drops slightly.

In addition, the physical behavior of the disk resource is complex. Disk throughput improves to some extent with increasing load, because the drives attached to a controller can perform seek operations simultaneously. When a substantial number of drives (three or more) are attached to the controller, this has the effect of lowering the contribution of the arm facility to the deterioration of system responsiveness with increasing load. In such a configuration, the system behavior is more dependent on the disk channel itself than on the disk arms.

These and other complexities persuade us that development and verification of an analytic model would cost far more than an event model. We suspect that an analytic model of even such a simple system would become so complex that we could never have developed the confidence in its behavior that we have in the behavior of the event model.

3.2 Measurement

The term "measurement" used in conjunction with the performance evaluation program has come to mean direct measurement of system behavior, usually by hardware-assisted or direct hardware means. We use three fundamentally different types of measurement, as described below.

3.2.1 Lab Measurement

Measurement under laboratory conditions is logistically the simplest measurement technique. Normally, at the outset of a system evaluation process, a lab configuration is set up similar to the one in figure 7.

The system need not have a real communications environment, or even real files. It normally has a single communication line with a single CI and terminal attached. This type of configuration is used to measure the basic system operating parameters: the behavior of the disk system, the execution times of the important programs (interrupt codes, transaction-processing codes, etc.), and the communications system behavior.

A lab system provides controllable, repeatable experimentation, involving only a single computer. However, the system is not in a real environment; it is not subjected to realistically high overhead, does not have realistically large files, and is not subjected to a realistically large load. Transactions are entered into the system one at a time, by hand, through the attached inquiry keyboard. Although this configuration permits basic measurements, it hardly permits accurate conclusions to be drawn about the behavior of the system in the real world.

3.2.2 Field Measurement

The opposite of the lab conditions above are provided by measurements of the field customer environment. Whereas the lab conditions lack reality, the field provides the actual loads, files, and configurations. Field measurements are very useful; they provide accurate data points along the responsiveness-load curve. Unfortunately, there are several drawbacks to field measurements. The largest is that the experiments are neither controllable nor repeatable, and are difficult to analyze. It is hard to know precisely what conditions prevailed during the measurement. This situation is complicated by the fact that a field measurement activity must be performed in a remote location, to which people and instrumentation gear must be shipped days in advance -- all on the customer's premises.

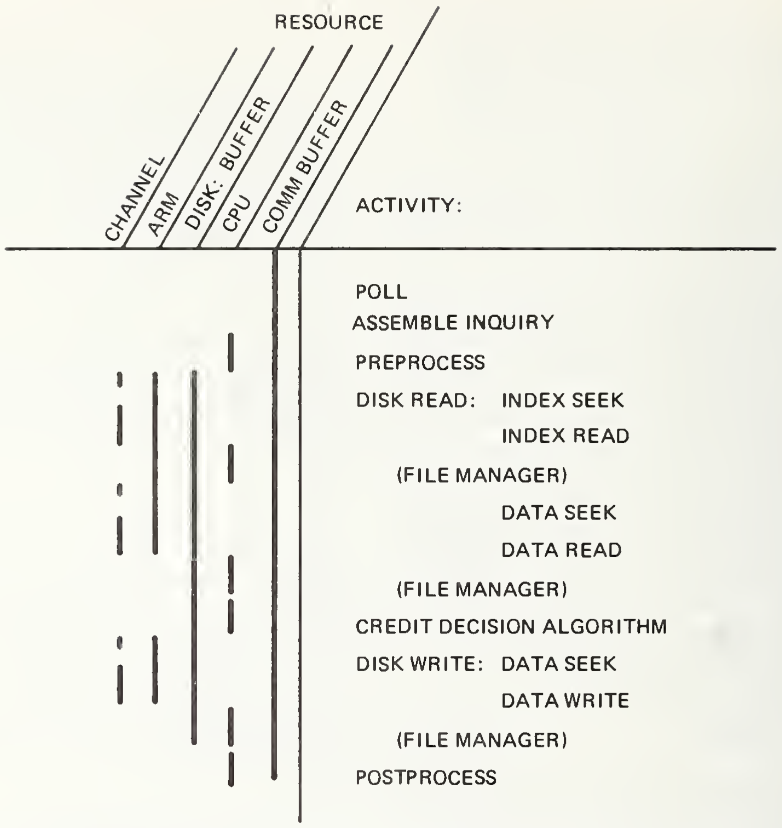


Figure 6. Resource Usage in a Simplified Transaction Flow

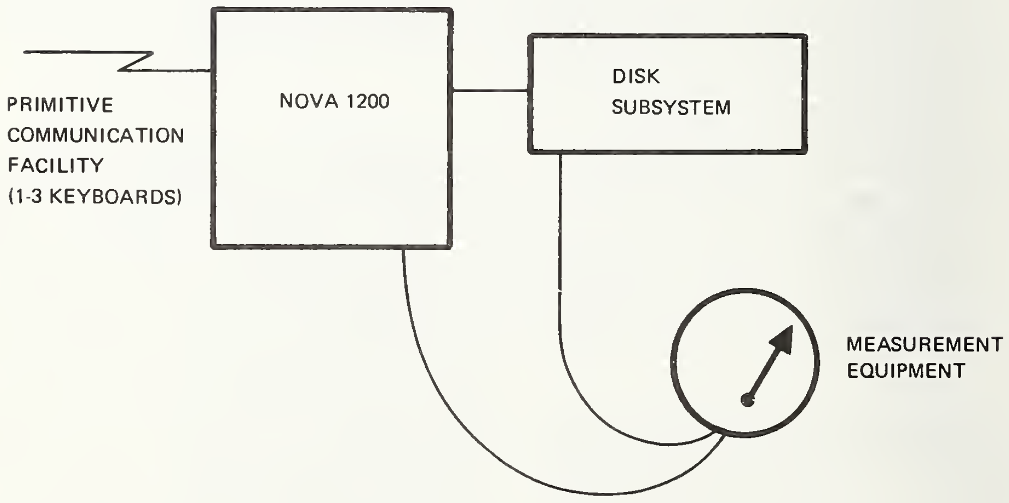


Figure 7. Lab Measurement Configuration

The utility of field measurements is high enough, however, that we tolerate the difficulties in order to obtain the data. For experimental verification of predicted system behavior, the behavior of the system in the field is irreplaceable. Further, the existence of empirical performance information permits new insight into any overall system behavior problems that occur during instrumented periods.

3.2.3 Measurement with Simulators

The third type of measurement is depicted in figure 8.

In this kind of configuration, the subject system is driven by another computer, which simulates the communications environment, provides a controllable load to the subject system, and observes its response. This technique combines the benefits of the lab situation with those of the field. It permits controllable, repeatable experiments, using real files, high loads, and nontrivial communications configurations.

Simulator configurations are the working backbone of the performance measurement program. However, they are subject to two fundamental shortcomings:

1. The communications environment in the real world is very complex; it involves various foreign equipment -- lines with diverse characteristics attached to many kinds of terminal devices with individually peculiar features. Consequently, the simulated environment can never precisely duplicate the actual environment. The small differences between the real and simulated environments are significant, especially where they hurt most - at high load levels.

2. The degree to which the simulator and the subject system limit each other is unknown. At very high load levels, when the combination is saturated, it is difficult to infer anything about the limit behavior of the subject system in the field.

These difficulties combined preclude the use of simulator configurations to determine the operational limits of real field systems directly. However, the overall utility of the simulator configuration is enormous; it provides the basic tool for acquiring data for validation of the system event model.

3.2.4 Fundamental Limitations of Measurement Techniques

The three measurement types described above have individual advantages and shortcomings. From them, it is possible to obtain a clear picture of the behavior of an existing system under achievable loads. However, several questions that must be addressed during the evaluation of a system cannot be answered from such a picture, no matter how clear. Measurement techniques can be used only on systems that exist. This leads to the following issues:

1. The effect of design changes cannot be evaluated by a measurement procedure.

2. The operating limit of most systems cannot be assessed by a measurement procedure. In some cases, field-measured loads provoke system behavior that is characteristic of the system operational limit -- but when this occurs, it is a lucky coincidence. We have been unable to accurately predict system limits by driving simulated lab environments to the brink.

3. The identity of the critical resource cannot be determined from even a very thorough measurement exercise. This is a consequence of the complex interaction of the different system facilities, as shown in figure 6. When the system approaches its limit, the resources that appear to be most heavily loaded may not be the ones whose behavior most strongly affects the system. This subject is illustrated in the discussion of resource dependency, using the event model example.

3.3 Predictive Event Modelling

The third major technique employed by the performance evaluation program is the use of a detailed, high-fidelity event model, for each system studied. The term _event model_ as used in this paper means a software realization of an idealized version of a modelled system. An event model defines two basic entity types:

1. _Resources_, which represent the facilities of the subject system (for example, refer to figure 6);

2. _Tasks_, which represent transactions and various internal system program activities.

Tasks contend for resources in the model, as they do in reality; the contention is resolved by a queue and scheduling

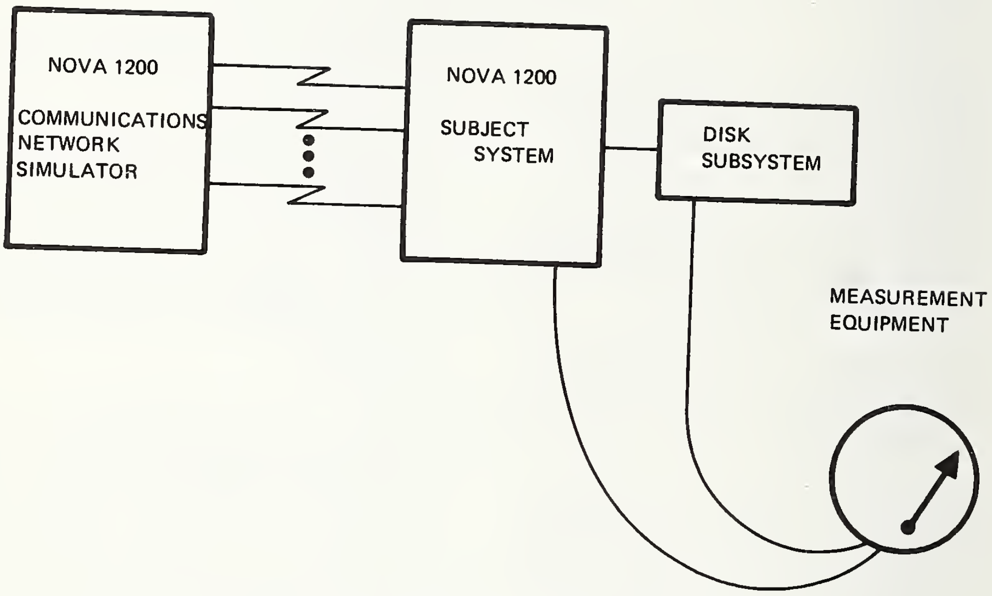


Figure 8. Simulator Measurement Configuration

algorithm associated with each resource.

The utility of an event model lies in its behavior: in mimicking the inner workings of a subject system, the model offers a detailed insight into the behavior of the system itself. The actions of the model are highly visible and mirror the inner workings of the system, which are otherwise difficult to observe.

The level of detail at which a system is modelled is not fixed. For each system, the level has been just high enough that the observed behavior of the model could be made to correspond closely to the measured behavior of the simulated system in the lab. The examples presented later in this paper show a model with about 20 resources, where each inquiry requires the modelling of about 15 events. The simplicity of this model is one of its attractive features. The modelling technique, when combined with a measurement program for model validation, offers a verifiable predictor of system behavior under conditions that cannot be realized readily in the lab. The measurement process has shortcomings -- it is unable to determine limits, detect resource dependencies, or examine design alternatives. These difficulties are all overcome through the use of a validated model. Additionally, the implementation cost of a model the size of ours is attractively low. The initial version of our model for the system exemplified in this paper is written in about 1500 lines of BASIC. A subsequent version has been implemented on the NOVA in about the same number of lines of an in-house higher-order language. Moreover, models for different systems share a basic structure and quite a bit of code; a typical "new" model for a credit authorization system now requires the development of only about 500 lines of new code.

The arguments against the use of event models for circumstances like ours are less than convincing. The strongest of these arguments is that the model shares complexity and "unknowability" with the system it depicts. In practice, this is unimportant. We understand the behavior of our models thoroughly, and from them have learned much about the behavior of the systems themselves. Our strategy was to keep the model as simple as possible, consistent with our requirements for fidelity; this tends to minimize the complexity argument. Further, the simplicity of our models has enhanced our confidence in their reliability, and has minimized the cost of their preparation.

The most persuasive argument for event modelling was based on our conviction that a high-fidelity analytic model of the systems we were investigating would be extremely complex -- so much so that we would never develop the intuitive confidence in its behavior that we have for the event models.

The keystone of any modelling effort, of course, is careful validation of the model. In our case, the measurement laboratory provides ample opportunity for verification of model behavior. Subsequent sections define our validation techniques in detail, with illustrations from actual system validation exercises. Our confidence in the behavior of our family of models is based entirely on earned trust: the models have been individually and painstakingly validated. This necessity was not obvious to us at the outset of our effort. We suspect that unvalidated models have contributed to the poor standing that event modelling efforts apparently hold within the CPE community.

3.4 Evaluation Strategy

Figure 9 shows the evaluation strategy that combines model execution with the measurement program. After the initial development of the basic system description, the model is implemented; a measurement configuration is used to determine the basic operating parameters of the system, which are supplied to the model. The model is then executed in a variety of configurations and under a variety of loads, all of which must be realizable either in the lab or in the field. The data drawn from these model executions is compared with the behavior of the system under the modelled conditions. Discrepancies are examined in detail, and the model is altered appropriately - usually by redevelopment of the system description at a more detailed level. When the discrepancies have been reduced to an acceptable level, the model is used for predictions of system behavior under conditions that cannot be realized in the lab.

This procedure is directed toward determining the transaction throughput capacity of a system. After the exercise is complete, this information is used in conjunction with standard queuing formulae to determine the response time characteristics of the keyboard entry terminals attached to the system, as a function of system load. (The latter part of the performance evaluation activity is not included in this paper.)

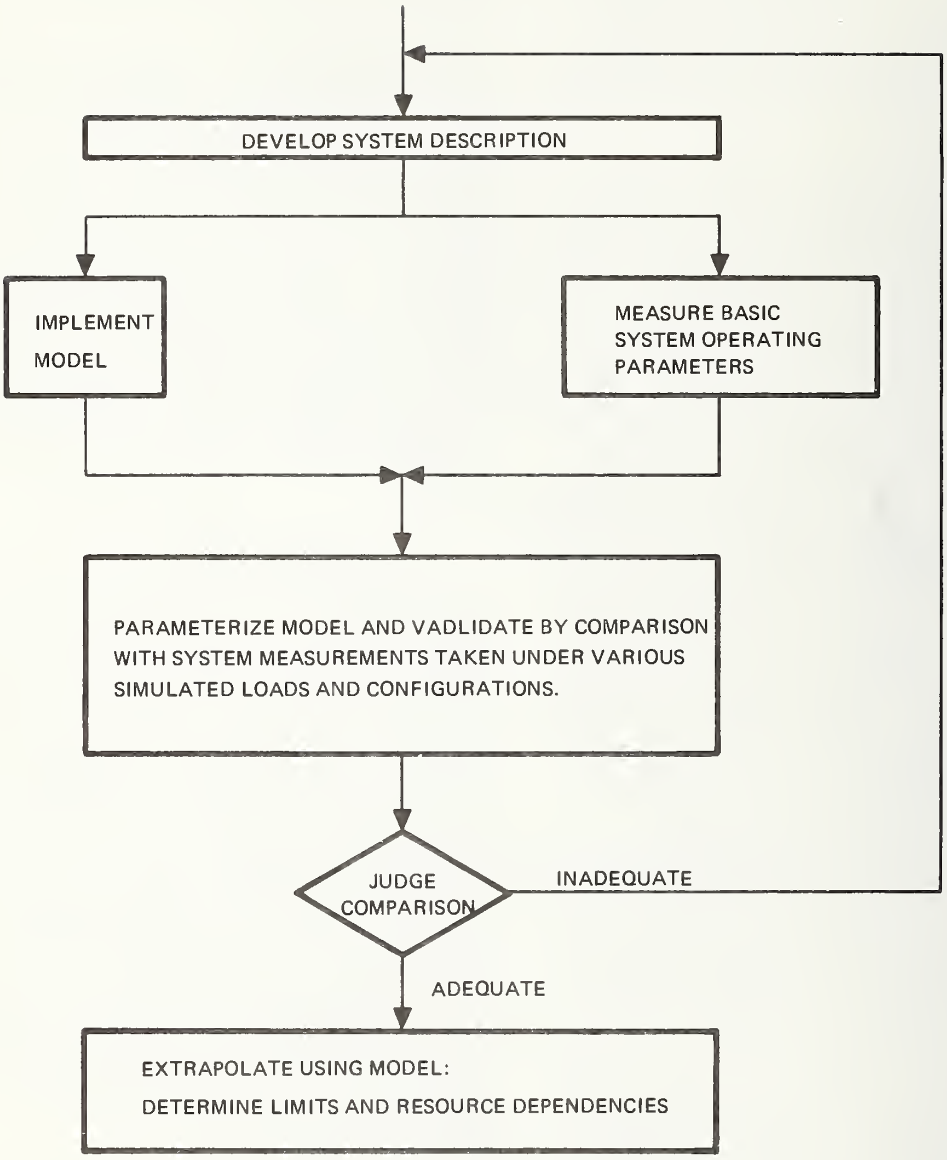

DEVELOP SYSTEM DESCRIPTION
IMPLEMENT MODEL
MEASURE BASIC SYSTEM OPERATING PARAMETERS
PARAMETERIZE MODEL AND VADLIDATE BY COMPARISON WITH SYSTEM MEASUREMENTS TAKEN UNDER VARIOUS SIMULATED LOADS AND CONFIGURATIONS.
JUDGE COMPARISON
INADEQUATE
ADEQUATE
EXTRAPOLATE USING MODEL:
DETERMINE LIMITS AND RESOURCE DEPENDENCIES

Figure 9. System Evaluation Strategy

4. The Evaluation Process: An Illustrated Example

This section illustrates the evaluation process with reports from model runs. All data refer to the evaluation of the study system.

4.1 Basic Measurements

The basic operational parameters of the system include disk operation times, execution times of the application programs, and the CPU overhead load measurements. They were all made in a lab configuration, using a TRW-built address comparator and a DYNAPROBE 7900 to determine all time intervals and extents. This configuration is depicted in figure 10.

4.2 Disk System Measurements

Because the disk controller permits overlapped seek operations, it was necessary to measure the operating characteristics of the disk system first. These measurements included seek time, synchronization delay time, overall channel and arm utilization, and the maximum rate for random disk accesses. This phase of the project yielded numerous surprises. For example, the measured mean seek time of the disk is 42 milliseconds -- not 35 milliseconds as the manufacturer had suggested. This difference was explained by an analytic error in the computation of the mean on the part of the disk drive manufacturer. The published data for seek time as a function of number of tracks moved proved to be correct; only the dynamic average was inaccurate.

Additionally, the disk controller used an unexpectedly large amount of channel-active time for each operation. This problem was traced to a synchronization delay, necessary for almost every operation on any drive, and amounting to about half a revolution per operation.

The results of the first phase of the process had thus already invalidated the system description to which the model was built. It was necessary to alter the model to reflect the synchronization delay, as well as the longer seek time.

The model described, as well as several others built subsequently, includes a detailed model of the actions of the disk system - to the level of accurate depiction of arm motions, rotational (and synchronization) delays, and channel utilization. We have become convinced that a high-fidelity model of a real-time disk-based system must necessarily include detailed disk descriptions. After an initial validation exercise, the disk model was further altered: the drives had to be independently positioned rotationally as an initial condition, and the arms had to be given random initial positions. The latter has only a small effect. The effect of the random rotational positioning, however, is significant - especially when pack-to-pack data motion dominates processing.

4.3 Additional Measurements

In addition to measurements of the disk system, the execution times of the significant programs were measured, by direct application of the DYNAPROBE/comparator configuration shown in figure 10. These measurements included the following:

1. "Clock" processing time, that is, the time required to process the periodic interrupt that drives the communications management function.

2. Interrupt response program times for communications interrupts.

3. Execution times for each application module modelled.

Also, various fundamental hardware-related measurements were made during all experiments as a cross-check on other results: the total interrupt-disable time, the total channel utilization, the total arm-in-motion time, and others.

4.4 Validation Phase

The validation phase consists of executing the newly parameterized model side-by-side with the system, with identical loads and configurations. For the subject study, the validation phase took about three weeks; it was only the second validation we had done. Validation is a highly time-consuming and frustrating process. The lab equipment is placed in a simulator configuration. Measurement involving two computers which must cooperate, and which are connected to a maze of measurement gear, leads to difficult logistic problems that lengthen the total time substantially.

The validation phase measured two different configurations at various load levels, comfortably below the system limit, so that interference from the simulator could be held to a minimum. The selected configurations used three and six connected lines. For validation criteria, two kinds

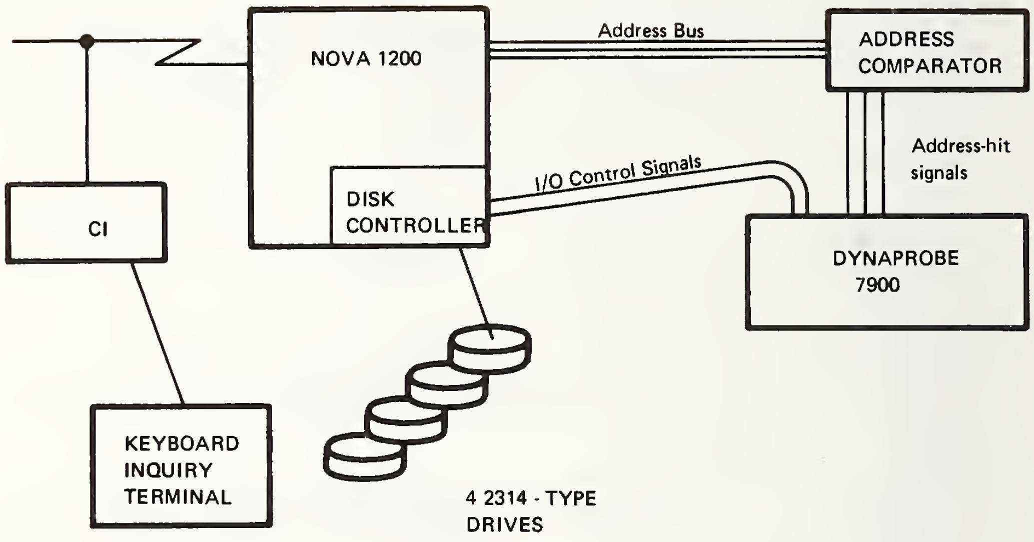


Figure 10. Basic Measurement Configuration

of measurements were used:

1. "External" measurements: for example, the "response" time displayed by the system in each experiment. The response time is taken in measurement and by the model to be the time elapsed between the moment when a message is completely assembled in the processor ready for processing and the moment when the first output character is transmitted to the waiting terminal.

2. "Internal" measurements: the validation exercise measured and compared the total channel-busy time, the communications buffer service time, the disk buffer service time and other "internal" measurements reflecting overall system activity.

The validation was eventful; the model had to be made fundamentally more complex to behave like the real system. The system description presented earlier in this paper, particularly of the communications subsystem, was developed in response to discrepancies between the lab system and the original model version; the original version supplied the application processor with a single exponential stream of inquiries. This had provided a fundamental inaccuracy, since in the real system the communication system causes lines to become occasionally idle, thus lowering total system overhead. Because the system, in the final analysis, is heavily CPU-bound, the overhead is a determining factor in total capacity.

Figure 11 shows a report from a validation run of the final version of the model. It shows a load of two transactions per second and a configuration of three lines. The report is in four sections: response time data, communication line-associated data, resource utilization data, and miscellaneous disk system data. The first and last report sections are self-explanatory. The second and third sections are explained by figures 12A and 12B, respectively.

Figure 11 shows that the system is behaving "well" -- that is, the load is well below the system limit. This is evidenced by the low standard deviation of response times and by the low facility utilization figures. Validation was performed on mean response time, "dynamic buffer" (communications buffer) service time, channel facility utilization, and disk buffer service time. The validation is satisfactory, as examination of figure 11 shows.

4.5 Limit Predictions Phase

The hard work of a modelling effort comes in the validation phase. After confidence in the fidelity of the model grows, the measurement phases can end, with their difficulties, and the model can be used for a variety of investigations - limit predictions, alternative design investigations, critical resource identification, and others.

The limit prediction phase looks for the saturation transaction throughput capacity and the operating transaction throughput capacity of the system. The former is the highest load level at which the system is able to operate, regardless of response times; the latter is the highest level at which the system is able to execute with reasonable times. Naturally, the latter limit is not absolutely fixed; however, investigation of the response time versus load characteristics of the system usually makes the operating point obvious - at the "knee" of the response time curve.

A report from a limit run with the model is shown as figure 13. This report shows a steady-state condition with the system absolutely saturated; its output rate is about 7.4 transactions per second, as measured on successive reports. This rate is the saturation transaction throughput capacity of the system. The situation depicted by this report could never be achieved in reality. Most transactions are "timed out" (refer to "NUM LOST" in figure 13). The system response time as seen from keyboard terminals would be erratic and very long. However, the absolute throughput limit of a system is an excellent criterion for comparison of different systems.

In figure 14, both response time and inter-poll delay time are plotted as a function of system load. "Inter-poll delay time" is the mean time during which a line lies idle, following receipt of an inquiry. This delay results from exhaustion of the dynamic buffer pool, and strongly affects the response time of the system as seen from the inquiry terminals.

Inspection of figure 14 shows the safe operating limit of the system to be about 5 or 5.5 transactions/second. Above this level, the system is no longer able to absorb the peaks in the load; response time distribution becomes erratic, and some very long response times may occur.

Figure 15 shows a report from execution of the model at about five transactions/second -- near the safe operating limit.

```
                  RESPONSE TIME DATA

TOTAL TIME =    16.000 SEC
NUMBER OF TRANSACTIONS SPAWNED =    32
AVERAGE SPAWNING RATE =  2.00/SEC
MEAN RESPONSE TIME =   .28 SEC ———————————————Ⓐ
MAX. RESP. TIME FOR TIMES BELOW 15.00 SEC =   .66 SEC
STANDARD DEVIATION OF RESP. TIMES =   .102 SEC
75 PERCENTILE RESPONSE TIME =   .33 SEC
95 PERCENTILE RESPONSE TIME =   .50 SEC
```

```
                       LINE PERFORMANCE

LINE   NUM   TRANS   NUM    NUM    NUM     MAX    NUM   MAX    MEAN
NUM   TRANS  /SEC    LOST   COMP   IN SYS  IN SYS WAIT  WAIT   DELAY

  1     13    .81     0      13      0       2     0     1      .000
  2      7    .43     0       7      0       2     0     1      .000
  3     12    .75     0      12      0       4     0     1      .000

TOTAL   32   2.00     0      32      0             0            .000
NUMBER OF COMPLETIONS/SEC =  2.00
```

```
                          RESOURCE PERFORMANCE

RESOURCE     OPS.    MEAN     MEAN    MEAN    MEAN    MAX       %      PEAK
  NAME       /SEC    SERV     WAIT    QUEUE   QLNTH   QUEUE  FACILITY DEMAND
                     TIME     TIME    LNGTH   @DISP   LNGTH    LOAD

CPU         12.00    .018     .002     .03     .20      3      22.1      1
CHANNEL     12.00    .010     .002     .02     .14      2      12.9Ⓒ    1
DRIVE1        .93    .056     .000     .00     .00      0       5.3      1
DRIVE2       1.31    .056     .000     .00     .00      0       7.3      1
DRIVE3       2.06    .048     .000     .00     .00      0       9.9      1
DRIVE4       1.68    .055     .000     .00     .00      0       9.3      1
SYS#BUF1      .31    .228⎫    .025     .00     .20      1       7.1      1
SYS#BUF2      .43    .220⎬Ⓑ  .000     .00     .00      0       9.6      1
SYS#BUF3      .68    .199⎪    .000     .00     .00      0      13.7      1
SYS#BUF4      .56    .220⎭    .075     .04     .33      2      12.3      1
DTBUF#PAIR   2.00    .299     .000     .00     .00      0       4.7      6
AUX#BUF      2.00    .314     .000     .00     .00      0       6.2      6
DYN#BUF      2.18   1.658Ⓓ   .000     .00     .00      0      24.1      9
```

```
    96 CHANNEL OPERATIONS, OF WHICH  52.08% REQUIRED SYNC
    63 NON-NULL SEEKS, AVERAGING  66 TRACKS AND 41.0 MILLISEC
```

Measured Values

A.	Mean Response Time	.282 sec.
B.	Mean Disk Buffer Service Time	.220 sec.
C.	Channel Facility Utilization	13.1%
D.	Dynamic (comm) Buffer Service Time	1.6 sec.

Figure 11: A Validation Run

```
                    LINE PERFORMANCE

LINE   NUM    TRANS   NUM    NUM    NUM     MAX     NUM   MAX    MEAN
NUM    TRANS  /SEC    LOST   COMP   IN SYS  IN SYS  WAIT  WAIT   DELAY

  1     13     .81     0      13      0       2      0     1      .000
  2      7     .43     0       7      0       2      0     1      .000
  3     12     .75     0      12      0       4      0     1      .000
  .
  .
  .
```

(one entry per comm line)

Number inquiries

Inquiry rate

Number inquiries timed-out due to slow polling

Number inquiries completed

Number inquiries now being processed

Maximum value of "num in sys"

Number inquiries now awaiting poll

Maximum value of "Num Wait"

Inter-poll delay time this line

Each column shows the data for one line characteristic:

(Bottom line shows totals for all lines)

```
TOTAL    32     2.00     0     32      0              0             000
NUMBER OF COMPLETIONS/SEC =   2.00
```

Figure 12A. Key to Section 2 of Model Report

```
                              RESOURCE PERFORMANCE

RESOURCE      OPS.     MEAN     MEAN     MEAN     MEAN      MAX       %       PEAK
  NAME        /SEC     SERV     WAIT     QUEUE    QLNTH    QUEUE  FACILITY  DEMAND
                       TIME     TIME     LNGTH    @DISP    LNGTH    LOAD

CPU          12. 00    . 018    . 002    . 03     . 20       3     22. 1       1
CHANNEL      12. 00    . 010    . 002    . 02     . 14       2     12. 9       1
DRIVE1         . 93    . 056    . 000    . 00     . 00       0      5. 3       1
DRIVE2        1. 31    . 056    . 000    . 00     . 00       0      7. 3       1
DRIVE3        2. 06    . 048    . 000    . 00     . 00       0      9. 9       1
DRIVE4        1. 68    . 055    . 000    . 00     . 00       0      9. 3       1
SYS#BUF1       . 31    . 228    . 025    . 00     . 20       1      7. 1       1
SYS#BUF2       . 43    . 220    . 000    . 00     . 00       0      9. 6       1
SYS#BUF3       . 68    . 199    . 000    . 00     . 00       0     13. 7       1
SYS#BUF4       . 56    . 220    . 075    . 04     . 33       2     12. 3       1
DTBUF#PAIR    2. 00    . 239    . 000    . 00     . 00       0      4. 7       6
AUX#BUF       2. 00    . 314    . 000    . 00     . 00       0      6. 2       6
DYN#BUF       2. 18   1. 658    . 000    . 00     . 00       0     24. 1       9
```

(One entry for each resource)

Resource Request Rate

Mean resource service time

Mean queue residence time

Mean queue length

Mean queue length at time of resource allocation

Maximum queue length

Resource utilization

Largest number of units in use simultaneously

Figure 12B. Key to Section 3 of Model Report

```
                    RESPONSE TIME DATA

TOTAL TIME =    90.000 SEC
NUMBER OF TRANSACTIONS SPAWNED =   689
AVERAGE SPAWNING RATE =   7.65/SEC
MEAN RESPONSE TIME =   2.17 SEC
MAX.  RESP.  TIME FOR TIMES BELOW 15.00 SEC =   4.39 SEC
STANDARD DEVIATION OF RESP.  TIMES =     .664 SEC
75 PERCENTILE RESPONSE TIME =   2.56 SEC
95 PERCENTILE RESPONSE TIME =   3.44 SEC


                        LINE PERFORMANCE

LINE    NUM    TRANS    NUM    NUM    NUM      MAX    NUM   MAX    MEAN
NUM    TRANS   /SEC    LOST   COMP   IN SYS  IN SYS  WAIT  WAIT   DELAY

  1     222     2.46    156     59      1        3      6    11     1.435
  2     180     2.00    114     59      1        3      6    12     1.435
  3     205     2.27    139     58      2        3      6    11     1.433
  4     226     2.51    157     61      1        3      7    11     1.413
  5     200     2.22    132     60      2        3      6    13     1.418
  6     203     2.25    135     59      2        3      7    10     1.426
  7     201     2.23    131     63      1        3      6    12     1.375
  8     220     2.44    150     62      1        3      7    13     1.381
  9      39      .43      0     38      1        4      0     1      .027
 10      34      .37      0     34      0        3      0     1      .004
 11     238     2.64    170     60      2        3      6    13     1.418
 12     224     2.48    156     60      2        3      6    11     1.407

TOTAL 2192    24.35   1440    673     16              63            1.264
NUMBER OF COMPLETIONS/SEC =   7.47


                          RESOURCE PERFORMANCE

RESOURCE      OPS.     MEAN     MEAN     MEAN     MEAN     MAX       %      PEAK
  NAME        /SEC     SERV     WAIT     QUEUE    QLNTH    QUEUE  FACILITY DEMAND
                       TIME     TIME     LNGTH    @DISP    LNGTH    LOAD

CPU          45.07     .021     .044     2.02     2.18       6      98.6       1
CHANNEL      45.04     .013     .008      .39      .61       3      61.6       1
DRIVE1        5.08     .072     .000      .00      .00       0      37.0       1
DRIVE2        6.02     .071     .000      .00      .00       0      43.3       1
DRIVE3        5.98     .073     .000      .00      .00       0      43.8       1
DRIVE4        5.43     .072     .000      .00      .00       0      39.4       1
SYS#BUF1      1.70     .455     .426      .72     1.31       5      77.4       1
SYS#BUF2      2.01     .444     .737     1.48     2.10       6      89.4       1
SYS#BUF3      2.00     .445     .815     1.63     2.32       6      89.1       1
SYS#BUF4      1.81     .449     .818     1.48     2.31       7      81.4       1
DTBUF#PAIR    7.58    1.145     .000      .00      .00       0      86.9      10
AUX#BUF       7.58    1.314     .886     6.72     6.74      10      99.7      10
DYN#BUF       6.84    2.190    1.448     9.91     9.72      10      99.9      15
CRT#BUF        .83    4.534     .016      .01      .05       1      62.9       6

 2027 CHANNEL OPERATIONS, OF WHICH  99.85% REQUIRED SYNC
 1351 NON-NULL SEEKS, AVERAGING  68 TRACKS AND 41.4 MILLISEC
```

Figure 13. Saturation Limit (NOVA 1200)

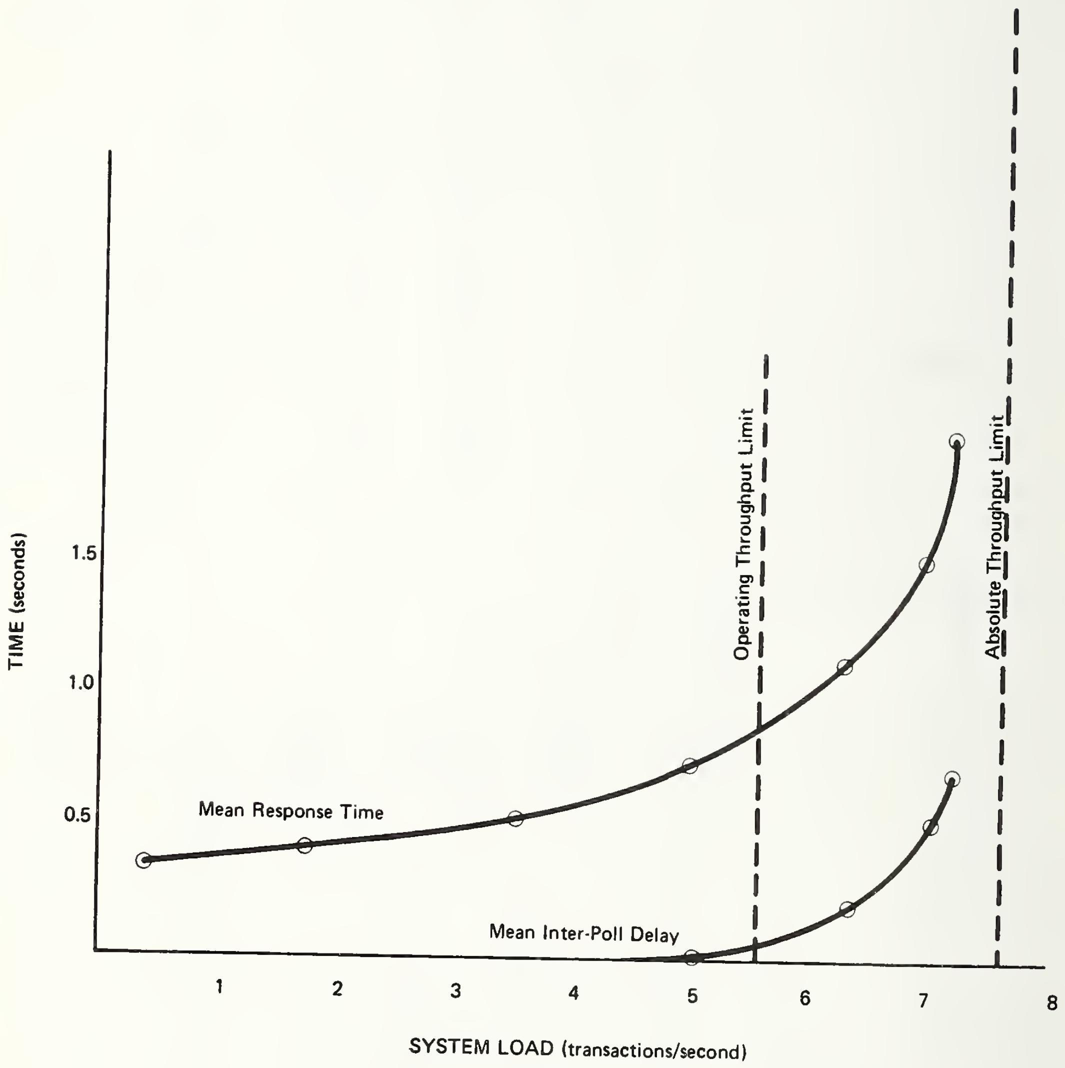


Figure 14. Response Time vs. Load (NOVA 1200)

```
                    RESPONSE TIME DATA

TOTAL TIME =    50.000 SEC
NUMBER OF TRANSACTIONS SPAWNED =  253
AVERAGE SPAWNING RATE =  5.05/SEC
MEAN RESPONSE TIME =   .47 SEC
MAX. RESP. TIME FOR TIMES BELOW 15.00 SEC =  1.39 SEC
STANDARD DEVIATION OF RESP. TIMES =   .186 SEC
75 PERCENTILE RESPONSE TIME =   .56 SEC
95 PERCENTILE RESPONSE TIME =   .87 SEC


                        LINE PERFORMANCE

LINE   NUM   TRANS   NUM   NUM   NUM    MAX   NUM   MAX   MEAN
NUM   TRANS  /SEC    LOST  COMP  IN SYS IN SYS WAIT WAIT  DELAY

  1     32     .64     0    31     1      3     0     1     .025
  2     29     .58     0    27     2      5     0     1     .033
  3     18     .36     0    18     0      2     0     1     .015
  4     20     .40     0    20     0      2     0     1     .000
  5     23     .46     0    23     0      3     0     1     .016
  6     19     .38     0    18     1      2     0     1     .025
  7     21     .42     0    21     0      2     0     1     .000
  8     19     .38     0    19     0      2     0     1     .017
  9      9     .18     0     9     0      1     0     1     .000
 10      1     .02     0     1     0      1     0     1     .000
 11     26     .52     0    26     0      3     0     1     .000
 12     36     .72     0    36     0      3     0     1     .003

TOTAL  253    5.05     0   249     4            0           .013
NUMBER OF COMPLETIONS/SEC =  4.98


                          RESOURCE PERFORMANCE

RESOURCE    OPS.    MEAN    MEAN    MEAN    MEAN    MAX      %     PEAK
  NAME      /SEC    SERV    WAIT    QUEUE   QLNTH   QUEUE FACILITY DEMAND
                    TIME    TIME    LNGTH   @DISP   LNGTH   LOAD

CPU        30.04    .024    .018     .56     .85      4     73.0       1
CHANNEL    30.10    .012    .004     .14     .31      3     37.9       1
DRIVE1      3.81    .060    .000     .00     .00      0     23.2       1
DRIVE2      3.48    .062    .000     .00     .00      0     21.6       1
DRIVE3      3.94    .062    .000     .00     .00      0     24.7       1
DRIVE4      3.81    .063    .000     .00     .00      0     24.3       1
SYS#BUF1    1.27    .315    .068     .08     .40      2     40.4       1
SYS#BUF2    1.15    .318    .067     .07     .34      2     36.9       1
SYS#BUF3    1.32    .320    .079     .10     .39      2     42.3       1
SYS#BUF4    1.27    .306    .058     .07     .37      2     39.1       1
DTBUF#PAIR  5.05    .382    .000     .00     .00      0     19.3       8
AUX#BUF     5.05    .509    .000     .00     .00      0     25.7       8
DYN#BUF     5.05   2.452    .014     .07     .11      3     82.7      15
CRT#BUF      .24   8.741    .000     .00     .00      0     34.9       3

  752 CHANNEL OPERATIONS, OF WHICH  80.58% REQUIRED SYNC
  503 NON-NULL SEEKS, AVERAGING  67 TRACKS AND 41.0 MILLISEC
```

Figure 15. Operating Point Limit (NOVA 1200)

The behavior of the model at this load level is acceptable: response time is low, and the low standard deviation of response time shows that the load level is below the "knee" of the curve. However, some signs show that the system is nearing its limit. The processor has a wait time nearly as long as its service time, and is 73% utilized. Also, the 82% load on the dynamic buffer pool ("DYN#BUF") is responsible for the 13-millisecond inter-poll delay time, which affects response time at the inquiry keyboards.

4.6 Resource Dependency Determination

Casual investigation of figures 13 and 15 may suggest that the critical resource is the dynamic buffer pool, because this resource has higher facility utilization than any other. However, this is misleading; it is caused by the complex interlocking of the utilization of the different resources in this system. The resource that really counts is the NOVA 1200 CPU itself. To demonstrate this, we ran the model in two altered conditions - with a greatly enlarged dynamic buffer pool, and with an infinitely fast CPU. Reports from these two runs are shown as figures 16 and 17. In the first report, with an essentially infinite dynamic buffer pool, response time actually deterioriates somewhat. Examination of figure 14 shows that the response time for 6 transactions/second should be about .7 seconds; in the report of figure 16, the response time is 1.02 seconds. This result is caused by the previously described nature of the communications system: when no dynamic buffer is available, a line lies idle until a buffer becomes available. When lines become idle, CPU overhead drops somewhat, because there are no interrupts to be processed for that line. The increased buffer pool does not help; it hurts.

The second run, figure 17, shows the effect of an infinitely fast processor. Comparison of this with the previous report should leave no doubt as to the major point: the CPU is what is hurting performance in the system. Additionally, it should be clear that measurement alone, no matter how well done, is inadequate to detect the real resource dependencies in an existing system.

The results of the critical resource identification exercise suggest the replacement of the NOVA 1200 with a faster processor -- a NOVA 800 or NOVA 3. The NOVA 800 executes instructions in about 60% of the time taken by the NOVA 1200. To examine the effect of the NOVA 800 in the system, the model was re-parameterized to reflect the operating speeds of the 800, and limit determination runs were made. Figure 18 shows the relationship of load to response time for the NOVA 800; figure 19 shows the reasonable operating point of a NOVA 800 system - about 9 transactions per second. These values have been validated by lab experiments with a NOVA 800-based system.

5. Summary

This paper has described a performance evaluation of an existing credit authorization system, using a detailed system event model, validated by laboratory measurements of system behavior. The system was found to be limited to about 5 transactions per second; the primary limiting factor was found to be the NOVA 1200 CPU itself. Replacement of the NOVA 1200 with the faster NOVA 800 raises the transaction throughput capacity to about nine transactions per second.

This condensed information belies the difficulty of its derivation and its importance to the relationship between TRW and its customer. The evaluation required almost three calendar months and about six man months to complete. In total, TRW's performance evaluation staff has now done six complete evaluations, including four with a validated model. We have gained the following insight into the difficulty and cost of these activities:

1. The logistics of a model validation exercise are very difficult. The multiple-machine configuration is a difficulty in itself; the checkout problems can be lengthy. A locked measurement lab is required to prevent delays due to logistic interference. Every step is slow.

2. Models of disk systems present particular problems. The disk activity must be modelled in extensive detail to achieve high-fidelity results in a model of a disk-based system. This requires careful validation measurement, which complicates the logistic problem, because several drives and controllers must be available as part of the measured configuration. Furthermore, measurements of disk controller peripherals require that probes be attached at probe points that must be devised in-house for non-standard disk controllers.

3. Evaluation of existing software is difficult at best because of inadequate documentation and unwarranted complexity. Additionally, even the

best existing software is hard to instrument as an afterthought.

The combination of these difficulties lends an unfortunate aura of unschedulability and unmanageability to the evaluation activity. Nonetheless, it appears to us that the value of the information yielded is so high that we must continue to proceed, essentially as above, for many existing TRW systems.

```
                   RESPONSE TIME DATA

TOTAL TIME =    60.000 SEC
NUMBER OF TRANSACTIONS SPAWNED =   372
AVERAGE SPAWNING RATE =   6.19/SEC
MEAN RESPONSE TIME =   1.02 SEC
MAX. RESP. TIME FOR TIMES BELOW 15.00 SEC =   3.40 SEC
STANDARD DEVIATION OF RESP. TIMES =    .569 SEC
75 PERCENTILE RESPONSE TIME =   1.31 SEC
95 PERCENTILE RESPONSE TIME =   2.19 SEC


                       LINE PERFORMANCE

LINE   NUM    TRANS    NUM    NUM    NUM     MAX    NUM   MAX    MEAN
NUM    TRANS  /SEC     LOST   COMP   IN SYS  IN SYS WAIT  WAIT   DELAY

  1     41      .68     0      41      0       4      0     1      .000
  2     38      .63     0      38      0       4      0     1      .000
  3     44      .73     0      44      0       5      0     1      .000
  4     28      .46     0      27      1       3      0     1      .000
  5     35      .58     0      33      2       4      0     1      .000
  6     36      .60     0      34      2       3      0     1      .000
  7     28      .46     0      28      0       3      0     1      .000
  8     34      .56     0      34      0       4      0     1      .000
  9      8      .13     0       8      0       1      0     1      .000
 10      7      .11     0       7      0       2      0     1      .000
 11     31      .51     0      31      0       4      0     1      .000
 12     42      .70     0      42      0       3      0     1      .000

TOTAL  372    6.19      0     367      5              0            .000
NUMBER OF COMPLETIONS/SEC =   6.11


                         RESOURCE PERFORMANCE

RESOURCE     OPS.    MEAN    MEAN    MEAN    MEAN    MAX      %       PEAK
  NAME       /SEC    SERV    WAIT    QUEUE   QLNTH   QUEUE FACILITY  DEMAND
                     TIME    TIME    LNGTH   @DISP   LNGTH   LOAD

CPU          36.90   .024    .049    1.81    2.07     6      91.7       1
CHANNEL      36.95   .013    .006     .22     .42     3      48.9       1
DRIVE1        4.28   .068    .000     .00     .00     0      29.1       1
DRIVE2        4.63   .065    .000     .00     .00     0      30.1       1
DRIVE3        5.01   .068    .000     .00     .00     0      34.1       1
DRIVE4        4.55   .067    .000     .00     .00     0      30.8       1
SYS#BUF1      1.43   .469    .339     .48    1.01     4      67.3       1
SYS#BUF2      1.54   .449    .406     .63    1.22     5      69.6       1
SYS#BUF3      1.68   .447    .485     .81    1.37     4      75.3       1
SYS#BUF4      1.51   .452    .274     .41     .97     3      68.6       1
DTBUF#PAIR    6.19   .832    .000     .00     .00     0      51.6      10
AUX#BUF       6.19  1.019    .043     .27     .38     5      63.2      10
DYN#BUF       6.11  2.667    .000     .00     .00     0      16.4      24
CRT#BUF        .28  8.038    .000     .00     .00     0      37.9       5

 1108 CHANNEL OPERATIONS, OF WHICH  96.48% REQUIRED SYNC
  738 NON-NULL SEEKS, AVERAGING  72 TRACKS AND 42.6 MILLISEC
```

Figure 16. Large Dynamic Buffer Pool (NOVA 1200)

```
                RESPONSE TIME DATA

TOTAL TIME =    60.000 SEC
NUMBER OF TRANSACTIONS SPAWNED =  357
AVERAGE SPAWNING RATE =  5.94/SEC
MEAN RESPONSE TIME =   .23 SEC
MAX. RESP. TIME FOR TIMES BELOW 15.00 SEC =   .94 SEC
STANDARD DEVIATION OF RESP. TIMES =   .112 SEC
75 PERCENTILE RESPONSE TIME =   .27 SEC
95 PERCENTILE RESPONSE TIME =   .45 SEC


                    LINE PERFORMANCE

LINE    NUM    TRANS    NUM    NUM    NUM    MAX    NUM   MAX    MEAN
NUM    TRANS   /SEC    LOST   COMP   IN SYS IN SYS WAIT  WAIT   DELAY

  1     37      .61      0     37      0      2      0     1     .000
  2     26      .43      0     26      0      2      0     1     .009
  3     31      .51      0     31      0      3      0     1     .000
  4     46      .76      0     44      2      3      0     1     .002
  5     31      .51      0     31      0      2      0     1     .000
  6     35      .58      0     35      0      2      0     1     .000
  7     28      .46      0     28      0      2      0     1     .000
  8     41      .68      0     41      0      2      0     1     .000
  9      8      .13      0      8      0      1      0     1     .000
 10      9      .15      0      9      0      1      0     1     .000
 11     32      .53      0     32      0      2      0     1     .000
 12     33      .55      0     33      0      2      0     1     .005

TOTAL  357     5.94      0    355      2             0           .001
NUMBER OF COMPLETIONS/SEC =   5.91


                      RESOURCE PERFORMANCE

RESOURCE     OPS.    MEAN    MEAN    MEAN    MEAN    MAX      %      PEAK
  NAME       /SEC    SERV    WAIT    QUEUE   QLNTH   QUEUE FACILITY DEMAND
                     TIME    TIME    LNGTH   @DISP   LNGTH   LOAD

CPU         35.58    .000    .000     .00     .00     1       .0       1
CHANNEL     35.61    .012    .006     .22     .32     3     44.5       1
DRIVE1       4.30    .062    .000     .00     .00     0     26.9       1
DRIVE2       4.51    .064    .000     .00     .00     0     29.3       1
DRIVE3       4.30    .064    .000     .00     .00     0     27.8       1
DRIVE4       4.69    .066    .000     .00     .00     0     31.2       1
SYS#BUF1     1.43    .187    .030     .04     .27     2     26.9       1
SYS#BUF2     1.51    .193    .035     .05     .28     2     29.3       1
SYS#BUF3     1.43    .194    .025     .03     .23     2     27.8       1
SYS#BUF4     1.56    .199    .072     .11     .50     3     31.2       1
DTBUF#PAIR   5.94    .235    .000     .00     .00     0     14.0       7
AUX#BUF      5.94    .235    .000     .00     .00     0     14.0       7
DYN#BUF      5.83   1.941    .001     .00     .02     2     75.4      15
CRT#BUF       .31   6.532    .000     .00     .00     0     34.4       4

  1068 CHANNEL OPERATIONS, OF WHICH  66.66% REQUIRED SYNC
   709 NON-NULL SEEKS, AVERAGING  68 TRACKS AND 41.2 MILLISEC
```

Figure 17. Infinitely Fast Processor

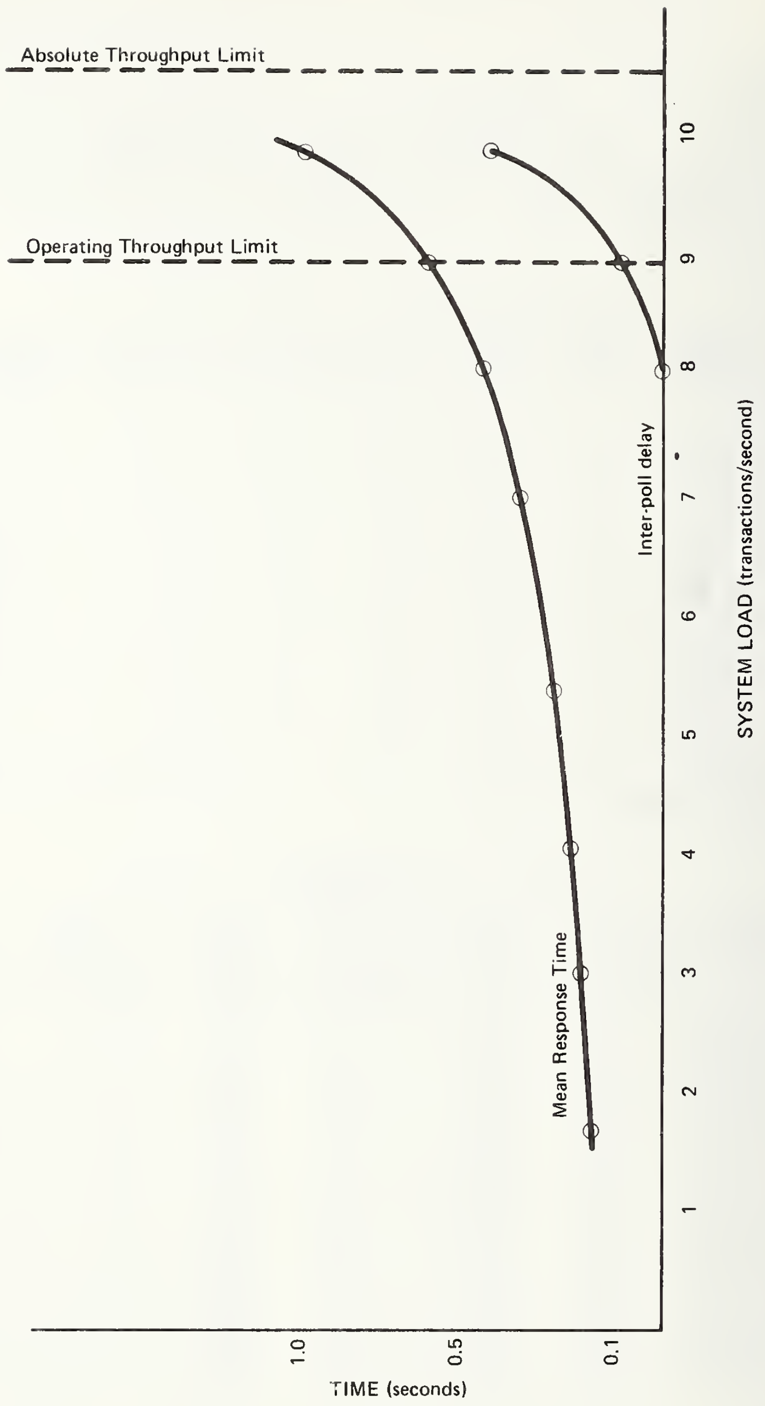


Figure 18. Response Time vs. Load (NOVA 800)

```
                    RESPONSE TIME DATA

TOTAL TIME =    70. 000 SEC
NUMBER OF TRANSACTIONS SPAWNED =  630
AVERAGE SPAWNING RATE =  9. 00/SEC
MEAN RESPONSE TIME =    . 57 SEC
MAX. RESP. TIME FOR TIMES BELOW 15. 00 SEC =  1. 91 SEC
STANDARD DEVIATION OF RESP. TIMES =    . 298 SEC
75 PERCENTILE RESPONSE TIME =    . 70 SEC
95 PERCENTILE RESPONSE TIME =  1. 20 SEC


                      LINE PERFORMANCE

LINE    NUM    TRANS    NUM    NUM    NUM     MAX    NUM   MAX    MEAN
NUM    TRANS   /SEC    LOST   COMP   IN SYS  IN SYS  WAIT  WAIT   DELAY

  1      68      . 97     0     66      1       3      1     1      . 099
  2      60      . 85     0     59      1       3      0     1      . 104
  3      60      . 85     0     59      1       3      0     1      . 100
  4      64      . 91     0     64      0       3      0     2      . 129
  5      70     1. 00     0     70      0       3      0     1      . 141
  6      65      . 92     0     64      1       4      0     1      . 097
  7      55      . 78     0     55      0       3      0     1      . 124
  8      60      . 85     0     60      0       2      0     1      . 121
  9      12      . 17     0     11      1       2      0     1      . 000
 10      10      . 14     0      9      1       1      0     1      . 000
 11      52      . 74     0     50      2       3      0     1      . 109
 12      55      . 78     0     54      1       4      0     1      . 096

TOTAL   631     9. 01     0    621      9              1            . 108
NUMBER OF COMPLETIONS/SEC =   8. 87


                        RESOURCE PERFORMANCE

RESOURCE    OPS.    MEAN     MEAN     MEAN      MEAN     MAX      %      PEAK
  NAME      /SEC    SERV     WAIT     QUEUE     QLNTH    QUEUE FACILITY DEMAND
                    TIME     TIME     LNGTH     @DISP    LNGTH   LOAD

CPU        53. 48   . 013    . 009     . 49      . 74      5     73. 9      1
CHANNEL    53. 50   . 013    . 012     . 68      . 85      3     70. 9      1
DRIVE1      7. 27   . 079    . 000     . 00      . 00      0     57. 7      1
DRIVE2      6. 74   . 078    . 000     . 00      . 00      0     53. 0      1
DRIVE3      6. 30   . 079    . 000     . 00      . 00      0     50. 2      1
DRIVE4      6. 44   . 080    . 000     . 00      . 00      0     51. 7      1
SYS#BUF1    2. 42   . 306    . 335     . 81     1. 50      6     74. 3      1
SYS#BUF2    2. 25   . 299    . 230     . 52     1. 08      4     67. 6      1
SYS#BUF3    2. 09   . 310     134      . 28      . 75      4     65. 1      1
SYS#BUF4    2. 15   . 306     170      . 36      . 92      3     66. 0      1
DTBUF#PAIR  9. 00   . 523     000      . 00      . 00      0     47. 1     10
AUX#BUF     9. 00   . 594    . 001     . 01      . 02      2     53. 4     10
DYN#BUF     8. 80  1. 607    . 113     . 99     1. 28      6     94. 3     15
CRT#BUF      . 34  6. 465    . 000     . 00      . 00      0     36. 9      4

  1872 CHANNEL OPERATIONS, OF WHICH  96. 31% REQUIRED SYNC
  1244 NON-NULL SEEKS, AVERAGING  67 TRACKS AND 41. 2 MILLISEC
```

Figure 19. Operating Point Limit (NOVA 800)

APPROXIMATE EVALUATION OF THE EFFECT OF A BUBBLE MEMORY IN A VIRTUAL MEMORY SYSTEM

W. T. K. Lin
A. B. Tonik

Sperry Univac
Blue Bell, PA 19422

This paper uses an approximate queuing model to evaluate how a bubble memory will affect a virtual memory paging system. Specifically, we compare the throughput of two systems: one uses bubble memory, the other uses a fixed-head disk. We found that if we replace the fixed-head disk by a bubble memory, we can reduce the main memory size by at least two million bytes and still maintain the same system throughput. Although we make several assumptions about the model, these assumptions have been shown to be quite accurate by various authors.

Key words: Bubble memory; performance evaluation; system modeling; virtual memory system.

1. Introduction

Bubble memory is not just a research object anymore; it will be a real product on the market very soon. But is it worthwhile putting a bubble memory into a system? This paper will use an approximate queuing model to evaluate the bubble memory in a virtual memory system. A fixed-head disk system, the SPERRY UNIVAC 8405, will be used for comparison with bubble memory being worked on by a SPERRY UNIVAC Research group.

2. Mathematical Model

The mathematical model used to compare bubble memory and a fixed-head disk is shown in figure 1. There are three stations in this closed queuing model. The first station S_0 consists of the CPU and main memory, S_1 consists of either bubble memory or fixed-head disks, and S_2 is mass storage of slower speed, such as a moveable-arm disk. There are m processes circulating in this system; i.e., the multiprogramming level is m. After being processed by the CPU for an average of $1/u_0$ seconds, a process will generate a page fault. This missing page has the probability of P_1 of being found in S_1, and P_2 of being found in S_2. That means a process will leave S_0 for S_1, if the missing page resides in S_1, or S_2 if the missing page is in S_2. The missing page in S_1 will, on the average, take $1/u_1$ seconds to be brought into S_0. The missing page in S_2 will, on the average take $1/u_2$ seconds. We assume the service times in all three stations are exponentially distributed. We also assume that there is only one CPU, one I/O channel for each storage level. Then, from the result of a paper by Gecsei and Lukes [1][1], we derive the following equations:

$$X_i = P_i X_0 \quad \text{for } i = 1,2 \tag{1}$$

$$X_1 + X_2 = X_0 \tag{2}$$

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

$$Q_0 + Q_1 + Q_2 = m \quad (3)$$

$$U_i = X_i/u_i \quad (4)$$

where:

X_i: throughput of station S_i in processes per second
Q_i: queue length at station S_i, including the one being served
U_i: utilization of station S_i
u_i: average service rate of station S_i.

From these equations we want to find the CPU utilization U_0 which is equivalent to the system throughput. Since we assume there is only one CPU, one I/O channel per storage level, and exponential service time, we can approximate each station by a M/M/1 queue. Gecsei and Lukes [1] have shown that this is a good approximation. Therefore, from Queuing Theory we know:

$$Q_i = U_i/(1-U_i). \quad (5)$$

Before we solve these equations, let us check which variables are unknown variables, and which are given parameters.

Multiprogramming level m and secondary storage access times u_1 and u_2 are given system parameters. Average CPU service time between page faults $1/u_0$, and probabilities P_1, P_2 can be computed by using the linear model proposed by Saltzer [2]. In that paper it is suggested that the number of memory references between page faults is proportional to the size of the main memory, i.e.,

$$n = c\ S_0 \quad (6)$$

where c is some constant, S_0 is main memory size, n is the number of memory references between page faults. Since part of the memory is used for resident system software, it is not pageable. Thus we modify eq (6) to eq (7):

$$n = c\ (S_0 - M) \quad (7)$$

where M is the size of non-pageable main memory (see [3]). Actually, n is the number of memory references to the pageable part between page faults. For the number of references between page faults to any part of the main memory, we have to add to n two numbers. One is the number of references to the non-pageable part of the main memory, which is approximately 25 percent of n; the other is a fixed overhead per page fault of about 2500 references. Therefore:

$$n = c\ (S_0 - M)\ (1 + 25\%) + 2500. \quad (8)$$

If we assume there are r memory references per instruction executed (measurements within UNIVAC indicate r can range from 1.5 to 2.0), and the CPU speed is v seconds per instruction, then the time between page faults will be:

$$1/u_0 = nv/r$$

$$= (vc\ (S_0 - M)\ (1 + 25\%) + 2500\ v)/r \quad (9)$$

By extending the linear model to the fixed-head disk (or bubble memory), we can compute the average number of memory references between page faults from the fixed-head disk to moveable-arm disk, i.e.,

$$n_1 = c\ S_1$$

where S_1 is the size of the fixed-head disk (or bubble memory). Therefore:

$$P_1/P_2 = (1/n - 1/n_1)/(1/n_1) \quad (10)$$

$$P_1 + P_2 = 1. \quad (11)$$

By solving (10) and (11), we obtain P_1 and P_2.

By substituting eq (1) into eq (4) and eq (4) into eq (5) and eq (5) into eq (3), we have an equation with X_0 as the unknown. A value for X_0 can be obtained from which, by eq (1), X_i can be obtained. Then the utilization of the various levels is obtained by using eq (4).

3. Comparison Between Bubble Memory and a Fixed-Head Disk

The model developed in the last section is used to compare a bubble memory being developed by UNIVAC with a UNIVAC 8405 fixed-head disk. The results are shown in figures 2 through 4. The values used for the parameters are as follows:

$r = 2$
$c = 1/1950$
$M = 3 \times 10^6$ bits
$v = 10^{-6}/3$ seconds/instructions
$1/u_2 = 4 \times 10^{-2}$ seconds.

These graphs show how the effective CPU utilization varies as the size of main memory varies for different levels of multiprogramming. The parameter m is the multiprogramming level. The main memory size is expressed in units of 10^7 bits.

The difference between figures 2 and 3 is the number of modules of 8405. In figure 2 the system has 4 units and in figure 3, six units. It can be seen that there is very little increase in utilization (throughput) by adding more 8405 memory units (except in the case of very large main memory sizes). Figure 4 has the bubble memory connected to the system through a regular I/O channel of 3 megabytes per second transfer rate. The size of the bubble memory is 6×10^8 bits, about twice the size of the fixed-head disks in figure 3. The reason for using twice as much bubble memory is because the price of bubble memory per bit is about half of the 8405.

Let us try to compare a system with 8405's and a system with bubble memory. We could take two systems with the same size main memory. However, it is obvious that the system with bubble memory would have more throughput (because the bubble memory is faster). According to figures 2 through 4, the system with bubble memory has about two times the throughput of the system with 8405's. This is almost the ratio of the total access times of those devices (this ratio is not surprising considering that the devices are the limitation of the systems). A more interesting comparison is to compare systems with the same throughput. This means that we should pick the CPU utilization of, for example, 0.25 on all figures. The first observation is that the different systems would have different sizes of main memory. Consequently, the multiprogramming level in the different systems should be different. Let us look at m = 9 for the system with six 8405's and m = 5 for the system with bubble memory. From figures 3 and 4, we obtain the following:

CPU Utilization	8405 Main Memory (m=9)	(Bubble) Main Memory (m=5)
.25	50M	26M
.20	40M	21.5M
.15	32M	16.5M

A system with bubble memory could have about 15 million bits less of main memory and still maintain the same throughput. This is about two million bytes. At 1.7 cents per bit, the savings to the users are about $250K. This is out of a system cost to the customer between $1.5M and $2M. This comparison also shows that at lower multiprogramming (m = 5 versus m = 9), the bubble memory system can still achieve the same throughput. However, in a transaction-oriented system, response time as well as system throughput will be important. This paper only addresses a batch virtual memory system. A separate analysis of a transaction system with bubble memory will be published at a later time.

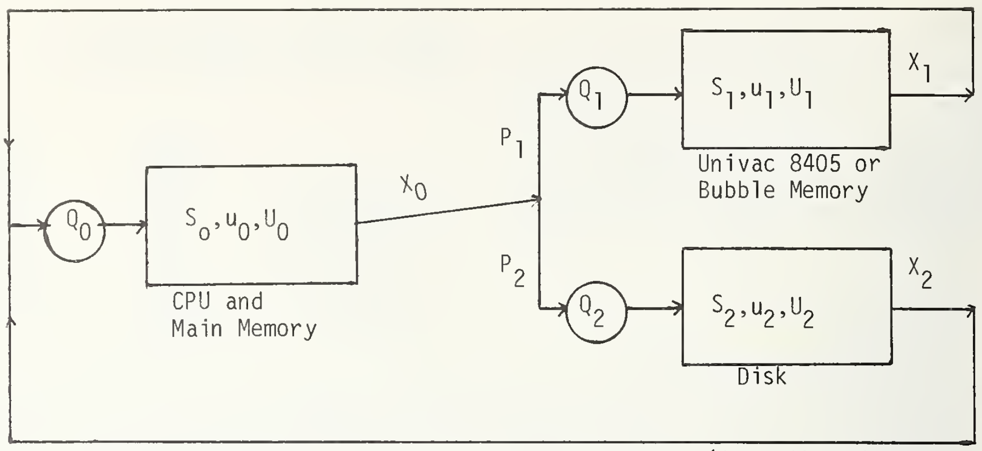
References

[1] Gecsei, J. and Lukes, J. A., A model for the evaluation of storage hierarchies, *IBM Systems Journal*, 13, 163 (1974).

[2] Saltzer, J. H., A simple linear model of demand paging performance, *Communications of the ACM*, 17, 181 (1974).

[3] Sekino, A., Throughput analysis of multiprogrammed virtual-memory computer systems, *Proceedings of SIGME Symposium*, 1973, pp. 47-58.

[4] Kleinrock, L., *Queueing Systems*, Vol. 1, (John Wiley & Sons, New York, 1975).



Q	=	Queue length; no. of jobs waiting for service (including the one being serviced)
S	=	Storage level
1/u	=	Average service time (I/O or CPU time between page faults)
X	=	Throughput
P	=	Probability of a missing page found in a storage level
U	=	Utilization of a storage level (fraction of time it is used)

Figure 1.
Queuing Model

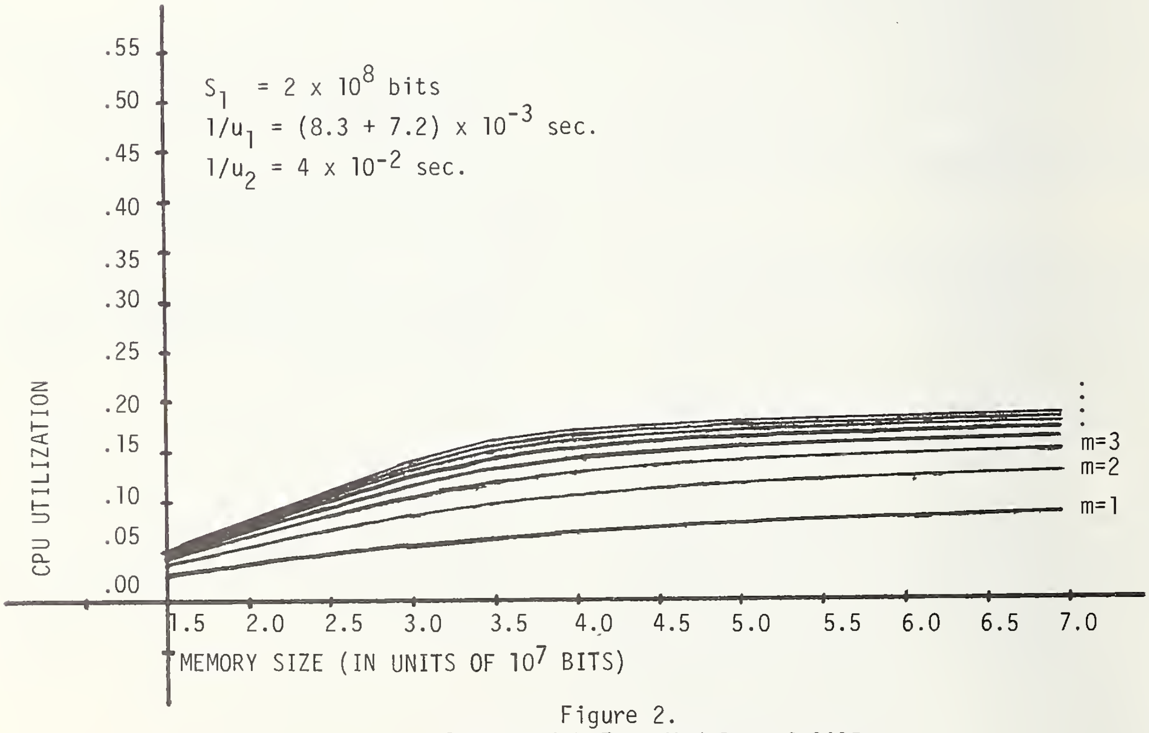


Figure 2.
System with Four Modules of 8405

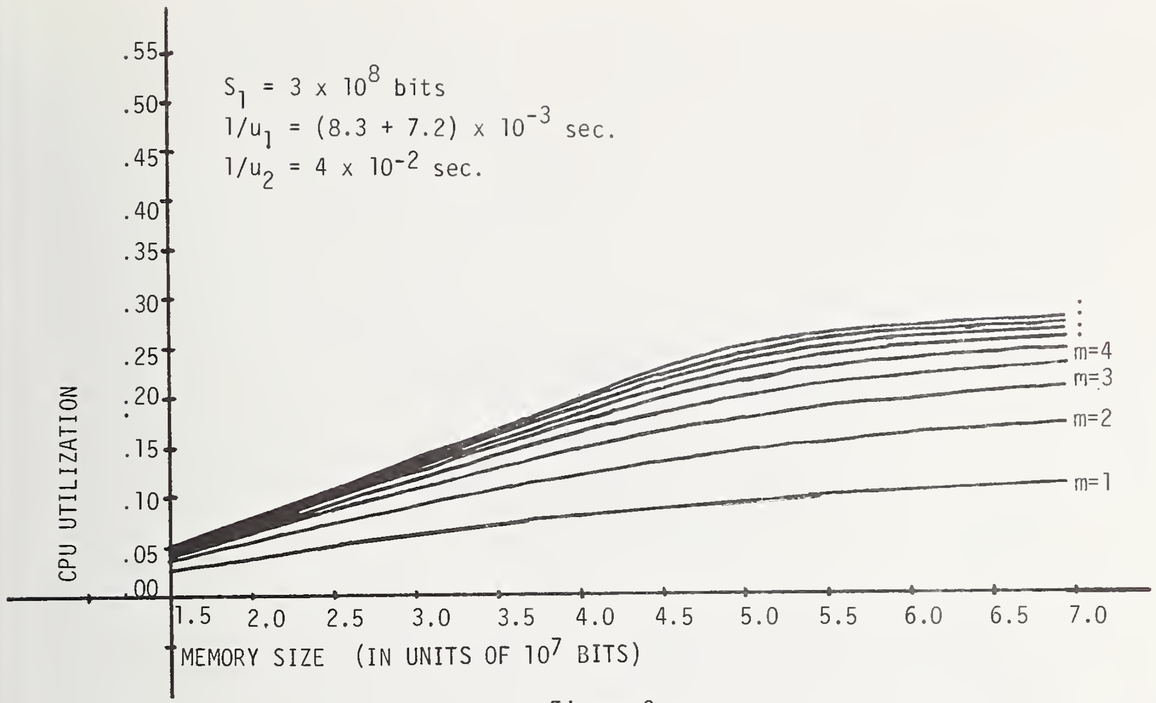


Figure 3.
System with Six Modules of 8405

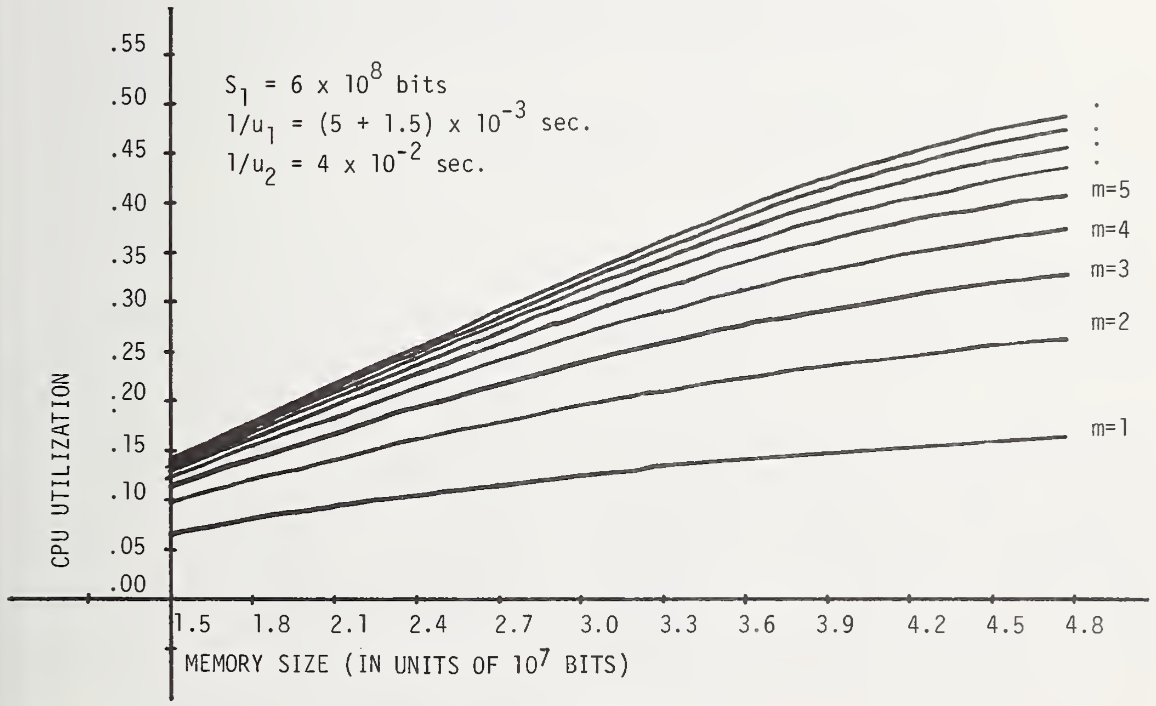


Figure 4.
System with 6×10^8 Bits Bubble Memory and
3M byte/second Channel

THE USE OF MEASURED UTILIZATIONS IN QUEUING NETWORK ANALYSIS

James Bouhana

Academic Computing Center
University of Wisconsin
Madison, WI 53706

For closed queuing networks having simple, load-independent servers, computationally efficient equations are derived relating several interesting network properties to server utilization. The equations permit measured utilizations to be used in computing the probability of a specific number of jobs at a server, the expected number of jobs at a server, and the mean queue length at a saturated server. The expressions in the derived equations require far less knowledge about the detailed stochastic parameters of a network than is needed in classical queuing network analysis.

Key words: Operational analysis; queuing networks; queuing theory; utilization.

1. Introduction

In recent years, queuing theory has proved increasingly useful in computer performance studies. Visualizing the components of a computing system as an interconnected network of servers and associated queues has enabled analysts to quantify the queuing delays that result when jobs compete for service in a multiprogramming environment. In particular, analysts are often interested in determining the expected apportionment of jobs and the mean lengths of the queues that form at each server.

In classical queuing network analysis, the determination of job apportionment and mean queue lengths proceeds by way of considering the equilibrium joint probability distribution of jobs in the network. The resulting mathematical expressions are in terms of the detailed stochastic parameters of the network. A brief exposition of the mathematical solution of a certain type of queuing network is given in Section 3.

In this paper, our orientation is that of determining queuing network properties by using easily-measured operational data. Although operational data has been the traditional raw material of performance analysis, its application to queuing systems has only recently been considered [1][1]. As delineated by Buzen [2]:

----

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

The operational method ... is based on a set of concepts that correspond naturally and directly to observed properties of real computer systems. ... This method is based on a set of definitions and working assumptions that are intended to reflect the viewpoint of individuals engaged in empirical studies of computer system performance.

The operational data required for our analysis are the measured utilizations of each server at each level of multiprogramming (henceforth called "mix level" for brevity). We will derive formulas relating certain network properties to measured utilizations. The derivations are presented in Section 4.

Next, some definitions and assumptions are stated which describe the type of queuing network models we will be considering.

2. Queuing Network Models

An example of a queuing network model is shown in figure 1. The example is the central-server model, discussed in [3], in which the central server represents the CPU and each of the peripheral servers represents an input/output device.

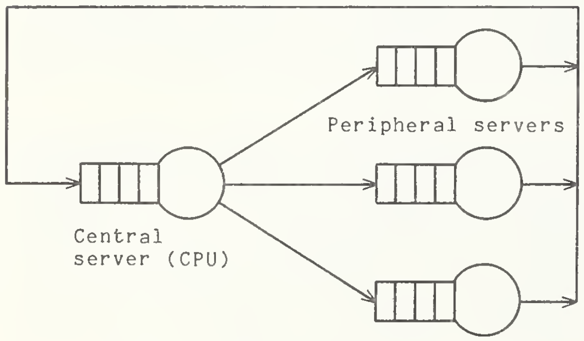


Figure 1: Central server model.

Associated with each server is a "waiting room" or queue which temporarily contains jobs that arrive at a server and find that the server is busy. The network is assumed to be closed in the sense that a fixed number of jobs travel among the various servers. (The assumption of closure will not prevent the analysis of actual systems in which the mix level varies, as we shall see later.) The routing of jobs through the network is specified by directed paths connecting pairs of servers. Associated with each path emanating from a server is a number specifying the probability that the path will be taken by a job that has just completed service at that server. The sum of the probabilities of all paths emanating from a server must equal one. The routing probabilities do not vary with time nor with the number of jobs at any server (i.e., they are both time and load independent). Each server is assumed to be simple in the sense that only one unit of that server is present at each circle (e.g., multiple CPU's are disallowed, unless each is modelled as a separate server). All jobs are assumed to behave identically in their routing through the network. Jobs are also identical with respect to the amount of service that they require from each server. The mean amount of time it takes for a server to fulfill a job's request for service is independent of the number of jobs in the server's queue (the "load-independent" assumption). The mean service time at a specific server is also independent of the load at any other server (the "homogeneity" assumption, which is equivalent to saying that the servers behave the same within the network as they would if they were isolated from the network and given the same workload). Finally, each server is assumed to have a per-request service time distribution that is exponential, with different servers having possibly different exponential distributions.

The models considered in the following sections are more general than the central server model in that the servers may be arbitrarily connected, as long as it is possible to travel from any server to any other server in the network. Additional generality will be noted in Section 3.2, as one of the major assumptions (that of exponentially distributed service times) is relaxed.

3. Mathematical Solution

The mathematical solution of closed queuing networks having simple, load-independent, exponential servers is presented in this Section. A complete understanding of the solution itself is not critical to understanding the results produced in Section 4. The solution is presented here for completeness and to establish notational conventions used in subsequent derivations.

For a network having M servers and containing N jobs, a state of the network is described by an M-tuple $n = (n_1, n_2, \ldots, n_M)$ where each n_i gives the number of jobs present at the i-th server (n_i includes the jobs that are enqueued, if any, and the one that is in service, if any). Each n_i is a non-negative integer, and the sum of all M of them equals N.

A queuing network is considered solved if one can determine the equilibrium probability distribution of states. Gordon and Newell [4] showed that the equilibrium probability distribution of states for the type of closed network discussed in this Section is:

$$P(n_1, \ldots, n_M) = \frac{1}{G(N)} \prod_{i=1}^{M} (X_i)^{n_i}$$

where the X_i are such that the M equations:

$$t_j X_j = \sum_{i=1}^{M} t_i X_i p_{ij} \qquad (1 \le j \le M)$$

are satisfied ($1/t_j$ is the load-independent mean service time of the j-th server, and p_{ij} is the constant probability associated with the directed path--if any--connecting the i-th server to the j-th server). The quantity G(N) is a normalizing constant selected such that the probabilities of all states sum to one. That is,

$$G(N) = \sum_{n \epsilon A} \prod_{i=1}^{M} (X_i)^{n_i}$$

where A is the set of all states.

3.1 Derived Statistics

In practice, a performance analyst is rarely interested in the probability of occurrence of a specific state. More commonly, one seeks expressions for server utilizations, probability of a specific number of jobs at a server, and mean number of jobs at a server. Buzen [3] showed that these statistics can be derived from the equilibrium solution. His results, which we will use later, are:

<u>Minimum number of jobs</u>. The probability that there are at least k jobs present at the i-th server (both enqueued and in service) when there are N jobs in an M-server network is:

$$P(n_i \ge k, N) = (X_i)^k \frac{G(N-k)}{G(N)} \qquad (1)$$

Server utilization is simply given by the probability that there is at least one job at the server, or:

$$P(n_i \ge 1, N) = (X_i) \frac{G(N-1)}{G(N)} \qquad (2)$$

<u>Exact number of jobs</u>. The probability that there are exactly k jobs at the i-th server is:

$$P(n_i = k, N) = \frac{(X_i)^k}{G(N)} (G(N-k) - (X_i)G(N-k-1)) \qquad (3)$$

Mean number of jobs. The mean number of jobs both enqueued and in service at the i-th server is:

$$E(n_i, N) = \sum_{k=1}^{N} (X_i)^k \frac{G(N-k)}{G(N)} \qquad (4)$$

3.2 A Generalization

The "product form" network solution given in Section 3 was derived by Gordon and Newell [4] under the assumption that the per-request service time of each server is exponentially distributed. Using operational arguments, Denning and Buzen [5] have recently shown that the product form equation is valid for general service time distributions--provided that the interpretation of "probability of a specific state" is replaced by the operational counterpart of "fraction of time spent in a specific state." The change in interpretation poses no difficulty; it is, in fact, wholly consistent with our operational orientation.

The results of Denning and Buzen permit generalizing the derived statistics of Section 3.1 to networks having non-exponential servers. In eqs (1)-(4), only the terms X_i are directly related to the service time distribution. In Section 4.1, we show that even these terms cancel completely in the derived expressions. Thus, the results which will be subsequently derived are, in an operational sense, truly independent of both the mean and the form of the service time distribution.

4. Measured Utilizations

Suppose that for a computing system with simple homogeneous load-independent servers, one empirically measures server utilizations at each mix level. If at some mix level N, the i-th server is observed to have delivered a total of T_i hours of service in an observation period of length T hours, then the utilization of that server is T_i/T. We will denote the utilization of the i-th server when there are N jobs in the mix by $U_i(N)$. In the notation of eq (2), $U_i(N) = P(n_i \geq 1, N)$.

In practice, server utilizations can be determined by a hardware or a software monitor. Alternatively, one could use a recently-developed algorithm for determining mean server utilizations at each mix level from accounting log data [6]. Either of the first two suggested methods is generally more accurate than using accounting log data. However, use of accounting data avoids the overhead usually incurred by monitoring, and the data is readily available on most systems.

In the remainder of this Section, an equation relating the network solution parameter G(N) to utilization will be derived. From eq (2), we have the following relationships for the i-th server at mix levels of 1, 2, ..., N:

$$G(1) = \frac{(X_i)G(0)}{U_i(1)} = \frac{X_i}{U_i(1)}$$

$$G(2) = \frac{(X_i)G(1)}{U_i(2)} = \frac{(X_i)^2}{U_i(1)U_i(2)}$$

$$\vdots \qquad \vdots \qquad (5)$$

$$G(N) = \frac{(X_i)G(N-1)}{U_i(N)} = \frac{(X_i)^N}{\prod_{j=1}^{N} U_i(j)}$$

4.1 Derived Statistics

By substituting the formulation for G(N) given in eq (5) into the expressions given in Section 3.1, we arrive at expressions formulated solely in terms of utilizations, as follows:

Minimum number of jobs.

$$P(n_i \geq k, N) = \prod_{j=N-k+1}^{N} U_i(j) \qquad (6)$$

Exact number of jobs.

$$P(n_i = k, N) = \left[\prod_{j=N-k+1}^{N} U_i(j) \right] \cdot (1 - U_i(N-k)) \qquad (7)$$

Mean number of jobs.

$$E(n_i, N) = \sum_{k=1}^{N} \prod_{j=N-k+1}^{N} U_i(j) \qquad (8)$$

a. Computational Aspects

We first consider the number of additions and multiplications required for the evaluation of eq (8). Assuming that the utilizations of M servers at mix levels ranging from 1 to N have been determined and are stored in an M x N array, the computation of the mean number of jobs at any one server can be accomplished with N-1 additions and multiplications. Denoting the M x N array by UTIL, an ALGOL routine which evaluates eq (8) for the i-th server and stores the result in a one-dimensional, M-element array JOBS is:

```
TEMP := UTIL[I,N];
JOBS[I] := TEMP;
FOR K := 2 STEP 1 UNTIL N DO
    BEGIN
    TEMP := TEMP*UTIL[I,N-K+1];
    JOBS[I] := JOBS[I] + TEMP;
    END;
```

Note that in the process of computing JOBS[I], all of the product forms appearing in eqs (6)-(7) are also computed. Thus, the given computational routine may be straightforwardly embellished to yield $P(n_i \geq k, N)$ and $P(n_i = k, N)$ for all values of k ranging between 1 and N. Then, the total number of additions and multiplications required for evaluating eqs (6)-(8) for any one server is N (the one extra operation results from the additional multiplication and subtraction appearing in eq (7)).

5. Saturation

Aside from its pragmatic applications, eq (8) is interesting in a theoretical sense, since it yields simple expressions for queuing behavior under conditions of server saturation. In this Section, we develop and comment upon expressions for $E(n_i, N)$ at saturation. Throughout this Section we assume that there is a unique server that becomes saturated before any other server.

If some server, say the i-th, becomes 100% utilized at some mix level, say S, then its utilization will remain at the value of 1 for all mix levels above S. The determination of the mean number of jobs at the i-th server for any mix level N where N > S takes the form:

$$E(n_i, N) = \sum_{k=1}^{N-S+1} \prod_{j=N-k+1}^{N} U_i(j) + \sum_{k=N-S+2}^{N} \prod_{j=N-k+1}^{N} U_i(j) \qquad (9)$$

Since $U_i(j) = 1$ for all $j \geq S$, all of the product forms in the left term of eq (9) equal one; also, the upper limit of the product form in the right term need only extend to S-1, yielding:

$$E(n_i, N) = N - S + 1 +$$

(Equation is continued on next page)

$$\sum_{k=N-S+2}^{N} \prod_{j=N-k+1}^{S-1} U_i(j) \qquad (10)$$

Expanding the sum of products in eq (10) shows that it can be written in a form independent of N, as follows:

$$E(n_i, N) = N - S + 1 + \sum_{k=1}^{S-1} \prod_{j=S-k}^{S-1} U_i(j) \qquad (11)$$

Equation (11) states that the mean number of jobs at a saturated server grows linearly and equally with the mix level beyond saturation. Since the amount of growth is in fact equal to number of jobs beyond the saturation level of the server, we can deduce that the first server to become saturated is the one at which jobs will become enqueued. Thus, the first server to become saturated exclusively captures--in a statistical sense--all jobs entered after saturation, and that server may rightly be viewed as the bottleneck of the network.

5.1 Mean Queue Lengths

Suppose, as before, that the i-th server is the first server to become saturated and that saturation occurs at the mix level of S. Setting N = S in eq (11) yields:

$$E(n_i, S) = 1 + \sum_{k=1}^{S-1} \prod_{j=S-k}^{S-1} U_i(j) \qquad (12)$$

Equation (12) is intuitively meaningful, since the 1 on the right hand side can be interpreted as representing the job (or, more precisely, different jobs at different times) that is always keeping the server busy. The sum of products in eq (12) represents the mean queue length (exclusive of the job in service) of the i-th server at saturation.

Note that the sum of products in eq (12) is of a familiar form, and the equation can be re-written as:

$$E(n_i,S) = 1 + E(n_i, S-1)$$

which states that the mean queue length of a server at its saturation level is equal to the mean number of jobs at that server when the mix level is one less than the saturation level.

6. Level of Detail

Equations (6)-(8) are somewhat surprising, since they reveal the wealth of probabilistic information embodied in such a macroscopic-level quantity as measured utilizations. After a moment or two of reflection, much of the surprise vanishes as one realizes that utilizations themselves are macroscopic-level manifestations of the detailed stochastic parameters that describe a network model of a computing system. So, one should not be too surprised that a mathematical connection can be made relating certain aspects of network behavior to utilizations. Equations (6)-(8) expose some of the connections.

However, one element of surprise lingers, which is that the connection can be made in a way that makes no reference whatsoever to the stochastic quantities required in the classical mathematical solution of a queuing network. For example, one need not know the mean service times of the servers. Also, the transition probabilities are not required. In fact, one need not even know the manner in which the servers in the system are interconnected, other than knowing that it is possible for a job to get from any server to any other server. Finally, one need not even specify the number of servers in the network.

The presence of other servers, the manner in which the network is interconnected, the routing probabilities, and the mean service times are all reflected in the utilizations that each server experiences at various mix levels! Thus, it may well be that utilizations constitute a natural bridge connecting the two realms of operational and stochastic analysis of queuing networks. Additional research exploring the mathematical topography of the utilization bridge is needed. Any further contributions can only serve to enrich our knowledge of both methods of analysis.

References

[1] Buzen, J. P. "Operational Analysis: The Key to the New Generation of Performance Prediction Tools," Proc. IEEE Fall Compcon '76, (Sept., 1976), Washington, D.C., pp. 166-171.

[2] Buzen, J. P. "Fundamental Laws of Computer Systems Performance," Proc. IFIP-ACM Sigmetrics International Symposium on Computer Performance Modeling, Measurement, and Evaluation, Cambridge, Mass., (March, 1976), pp. 200-210.

[3] Buzen, J. P. "Computational Algorithms for Closed Queueing Networks with Exponential Servers," Comm. ACM, Vol. 16, No. 9, (Sept., 1973), pp. 527-531.

[4] Gordon, W. J. and G. F. Newell "Closed Queuing Systems with Exponential Servers," Operations Research, Vol. 15, No. 2, (April, 1967), pp. 254-265.

[5] Denning, P. J. and J. P. Buzen "Operational Analysis of Queuing Networks," Proc. Third International Symposium on Modelling and Performance Evaluation of Computer Systems, Bonn, W. Germany, (Oct., 1977).

[6] Bouhana, J. "A Family of Mix Characteristics Curves," Proc. 12-th CPEUG Meeting, (Nov., 1976), San Diego, Calif., pp. 181-188.

APPLICATIONS OF QUEUING MODELS TO ADP SYSTEM PERFORMANCE PREDICTION: A WORKSHOP SUMMARY

Mitchell G. Spiegel

Federal Computer Performance Evaluation
and Simulation Center (FEDSIM)
Washington, DC 20330

The objective of this presentation is to summarize the state-of-the-art of queuing models from the position of Workshop Chairman of a recently completed workshop on the "Applications of Queuing Models to ADP System Performance Prediction." Over sixty queuing model practitioners gathered together at the National Technical Information Service (NTIS) on 7-8 March 1977. Topics discussed were divided into four general areas: (1) Application of Queuing Models to Feasibility and Sizing Studies; (2) Application of Queuing Models to System Design and Performance Management; (3) Queuing Model Validation; and (4) New Queuing Model Implementations. Eighteen speakers provided a broad viewpoint of applied queuing model issues.

Key words: Computer-communication networks; configuration analysis; model validation; performance management; performance prediction; queuing networks; sizing studies; system design.

INTRODUCTION

A workshop was held on the Applications of Queuing Models to ADP System Performance Prediction on 7-8 March 1977 at the National Technical Information Service in Springfield, VA. Topics were divided into four general areas: (1) Application of Queuing Models to Feasibility and Sizing Studies, (2) Application of Queuing Models to System Design and Performance Management, (3) Queuing Model Validation and (4) New Queuing Model Implementations. Mr. Philip J. Kiviat, Chairman, SIGMETRICS, made the welcoming remarks. As Workshop Chairman, I provided a historical overview of queuing model use which traced the development of the application of queuing models to ADP system performance prediction through the 20th century, while setting the stage for each speaker's talk.

The first queuing model applications were to busy signal problems in the early telephone network. One of the early uses of queuing formulas to model a computer system was for the SABRE airline reservation system. These early uses employed queuing tables. In the early 1960's, fixed form queuing models appeared within the internal algorithms of packaged simulators. Today, queuing model facilities with graphical types of languages and imbedded approximation techniques allow the analyst to describe and solve almost any queuing network.

FEASIBILITY AND SIZING STUDIES

The speakers on the first day of the workshop concentrated their talks on queuing model applications. A complete list of the speakers and their topics is provided in Appendix 1. The first speakers, Capts. Askland and Davis, discussed the use of a queuing model for an Air Force Base Level Burroughs 3500 Sizing. The model was used on hundreds of combinations of configurations and workloads. The presenters estimated that over 138,000 experiments had been performed. The model employed was that of Avi-Itzhak/Heyman [1][1]. Two implementations were described: (1) a version to solve for response time and (2) fixing response time and service rate to find maximum tolerable traffic. Performance data on the operating system was obtained by graphing measurements of the ratio of system I/O's to user I/O's for many different mixtures of workload. The mechanics of testing the adequacy

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

of a particular configuration was to test the largest configuration within a class of CPU and accept or reject it by experimentation. This approach eliminated unsatisfactory classes immediately.

The next speaker, Capt. Pearson, described the in-depth analysis of a small communications network model for the Navy. The normal method of modeling a host computer was reversed; the network terminated at the host. The host was represented as a single arrival/departure point for traffic. The network consisted of ten geographical sites with 500 interactive terminals. The model was constructed using the Analytic Solution for Queues (ASQ). Model limitations forced the experimenter to accept a loss of accuracy by consolidating different attributes of the network.

One limitation of the ASQ model was a maximum of three job types. Five different experiments were run using the same sub-network structure, while altering the use of the job-type feature. Representing a class of work, i.e., time-sharing, as a job type was less desirable than allocating a job type to a physical network location containing heterogeneous traffic, because detailed information about specific nodes was lost. (The most accurate representation is a job type for each physical location and kind of work.)

Another experiment tested the use of Norton's theorem to collapse an arbitrary network to a single node with a queue-dependent service rate. This feature is thought to be critical to managing the analysis of a large network. Experiments with sub-networks demonstrated the validity of the "collapse" approach. One other observation concerned the flow of job types. Results were not reliable unless all job types passed through one node somewhere in the network. This omission is common for beginning users of a network model.

Dr. Wang closed the morning session with a talk on the sizing of a computer for use in a switching network. His opening remarks addressed a general network design methodology. He proposed a cyclic process consisting of four stages: (1) definition of network functions, (2) distribution of network intelligence, (3) structuring of network architecture, and (4) design of network to cost/performance constraints. "Performance" related to the configuration before, during, or after the structuring process. He recommended setting performance constraints, minimizing the cost of a sub-network within the constraints established, and relaxing the constraints slightly to test the sensitivity of the design to performance parameters.

The sizing study Dr. Wang described was part of a system redesign problem. The existing network served 20,000 terminals, 14,000 of which were in the continental United States. The workload consisted of cyclic transactions and source data entries for various destinations. Response time performance was the principal measure of service used by subscribers and was contractually guaranteed. Dr. Wang began the analysis by partitioning response time into six component parts stretching from terminal to host. His approach to sizing evaluated the slowest component first (in this instance, the low-speed line). He then attempted to eliminate the intervening level of processing in the network (network and data processors) by multiplexing low-speed lines directly to the host computer. Predicted response times were insensitive to the type of application, which was an unsatisfactory situation. By replacing the multiplexor with a data processor containing frequently accessed data, the desired response time mix was obtained. Performance was predicted using a number of subsystem models with discrete capability to evaluate the dynamics of line protocols, concentrators, and network component interactions. The subsystem results were fed to a queuing network model of the entire computer-communications net.

The Monday morning panel discussed the imprecise measure of response time produced by most queuing models. None of the presenters saw this as a problem, because users have little or no conception of response time at all, or its distribution. Management has been receptive to rough predictions relating response time to economic value. "Price/ Performance" data is much sought after for major system design and financing decisions. Panelists also felt that an undue amount of attention is paid to system response time without regard for the design implications on the human operators in the user-computer loop.

SYSTEM DESIGN AND PERFORMANCE MANAGEMENT

Dr. Wang reopened the afternoon session with a second presentation describing the use of a queuing model in detailed design trade-offs for a 2,500-terminal network. The workload modeled consisted of over 30 transaction types. Terminals were connected in a star configuration to a network interface minicomputer. The minicomputer was joined to other minicomputers in a loop with a host computer. Messages could travel in either direction in the loop, with the minicomputer acting as a packet switching node. The queuing model predicted the number of intermediate nodes and loop line utilization maximums required to meet all response time design goals.

Mr. Berners-Lee of ICL described how a manufacturer successfully used a queuing model internally within his organization. A model following Jackson's theory, known as the "football model," was constructed and validated carefully for throughput and elapsed time predictions over long periods of time (12 hours of mixed work on a system). The model is now used to plan enhancements to the product line. System analysts are trained in configuration balancing in order to assist customers to obtain the best system performance possible.

While calibrating the model, International Computers Ltd. technicians observed a close fit between the number of file accesses and the storage space occupied by the file. The access pattern conformed to Zipf's law (access traffic is proportional to the logarithm of the number of unique addresses on the file). Further, the use of isolated server theory (Norton's theorem) worked well in a hierarchical model. It was also thought that competitive sizing would be difficult because of the problem of translating the performance of an application from one machine to another. This proved not to be the case. Once a model of a competitive operating system was obtained, it was relatively simple to forecast the impact of an application conversion to the competitive vendor's equipment and software [2]. Finally, a good long-term model of system performance was obtained by developing an aging cycle for file activities. Using the state transitions, analysts predicted various types of file traffic over time, e.g., retrieval, archiving.

Ms. Dowdy used queuing models to configure a system, by balancing work between a general purpose processor and an array processor. Measured data from benchmarks were used to calibrate the initial closed queuing network model. Subsequent predictions were validated against actual measured data and found to follow the projections closely.

Dr. Bronner indicated the importance of queuing models to understanding complicated system environments. Queuing models can answer most management questions easily and quickly and are proving to be a very popular tool. He has incorporated queuing models into a capacity planning methodology that addresses the management of computer resources [3]. By using the capacity planning process, the loading, utilization, and response of the system's resources are monitored and analyzed by a group of capacity planners. Most of the measurement tools used (i.e., SMF) were not developed with capacity planning as an objective. Although the data are incomplete, an organized structure for tracking and analysis, using existing tools, offers insights into the operation of an installation. The type of projection usually made by the analyst is a trend line, using time series analysis. The trend line helps to correlate interacting factors among workload, resource use, and user service levels. Detailed analysis of subsystem measurement data (e.g., TSO trace, CICS analyzer) is required for model development. When data are not available, approximations are required. One example of an approximation is to double the data transfer time to account for command data passed back and forth over data channels. Dr. Bronner recommended the use of an analytically-oriented person to enhance the modeling and predictive skills of the capacity planning group. This person can provide reasonable loading and service level predictions for new applications, changes in existing workload, or system configuration modifications.

Queuing models were recommended once confidence in established measured parameters had been obtained. His opinion was that the problem in complex computer systems analysis was not with the availability of analytics, but a lack of understanding of system relationships. Most prediction requirements can be handled by treating the computer/communications system as a single system queue. Refined analysis employs hierarchical queuing networks in place of the single system queue point.

Mr. Day described the use of a queuing model as part of a customer assistance package in the pre-sale, order-sizing, and post-sale environments. In the pre-sale environment, the package operates in a "choice" phase. The customer describes the number and location of stations in his network, traffic, and transaction types. The package output consists of a prediction of the size and type of system required. The model contains pre-measured values for the service times of all critical paths through the software operating system and related subsystems.

Mr. Day responded to a challenge by members of the audience. He claimed to be able to generate non-trivial response time distributions without excessive run time. His methodology consisted of computing all the permutations and combinations of event service time probabilities in such a sequence that when the tail of the distribution containing high response times is reached, he truncates the calculation in accordance with a delta corresponding to the percentage of all combinations desired. Very little accuracy is lost with large reductions in run time.

The afternoon speakers ended the discussion of applications of queuing models with ideas about how to get performance out of applications during their initial design. The panelists agreed that performance does take a back seat to error-free execution, except when it is mandatory to the acceptance of the work. Analysts have a poor track record for getting a performance discipline into the application design cycle. The results are saturated systems, in as little time as six months following installation, and far in advance of the projected peak workload conditions. Users have had good queuing models available to them for some time, but they persist in doubting the validity of the parameters and the models' world view of the system instead of applying the models to the application design process.

The critiques submitted by attendees were very receptive to the sessions of the first day. The first session, on applications of queuing models to feasibility/sizing, "illustrated the problem domain and pointed toward the need for improvement in support and directions for future research and development efforts" for one attendee. A presenter wrote "I got some insight into how others are using queuing models." A user's comment was: "Very good overview of the state-of-the-art in applying queuing network models to real world problems." The second session, on applications to design, implementation, and tuning, contained presentations by three mainframe vendors. Several users were pleasantly surprised at "computer manufacturers really trying to get control of matters with novel approaches."

MODEL VALIDATION

The speakers on the second day dealt with the issues of queuing model validation and new queuing model implementations. A complete list of the speakers and their topics is provided in Appendix 1.

Cmdr. Rose discussed the various uses of queuing network models and the problems encountered with current measurement devices. He obtained the current version of the Buchholz FORTRAN jobstream from FEDSIM and conducted many carefully controlled benchmark experiments at the IBM Gaithersburg facility that had been heavily instrumented for this purpose [4].

Problems with model validations result primarily from the fact that typical measurement probe points and commercially available monitors often preclude obtaining the proper queuing network model parameters. This situation does not necessarily mean that achieving the capability is difficult, but reflects the absence of a requirement in the monitors' design specifications to obtain model parameters. For example, data processing centers have historically been provided measurement devices so that they can determine bottlenecks and equalize CPU and I/O processor utilizations. The installations typically measure "CPU waiting and only channel 1 busy" and "CPU busy and no channel busy," but these measurements do not meet the requirements of an analytic model.

An analytic model of a computer system can be used to predict performance improvement that might result from a proposed modification of the existing system. Cmdr. Rose hypothesized situations in which a manager considered upgrading his system and wanted to compare estimated performance improvements to anticipated costs. Cmdr. Rose basically ran a benchmark program on the baseline system and validated the model. After validation, he used the model to predict CPU and I/O channel utilizations for the upgraded or reconfigured system. The benchmark was rerun on the reconfigured system, and measured utilizations were compared to the model's predicted values. Adding a channel to an IBM 370/155, reallocating files and adding two channels to an IBM 370/155-2, and adding a channel to an IBM 370/168-1 resulted in less than 10% variances between predicted and measured utilizations. The model overestimated these computers' utilization by approximately 20% when main memory was doubled.

Experiments with large CPU quantum in the system worked better with a processor-shared (PS) discipline than first-come-first-served (FCFS) disciplines. When smaller numbers of jobs were being processed, the reverse was true. The reason appears to be the high level of interrupts generated by a jobstream with many concurrent jobs. The high activity level effectively creates a processor-shared environment.

Contrary to the expectation of the audience, there is no requirement for random access device motion time (seek and latency), to calibrate the model. Station balance was accurately reflected by using channel service time only. The missing time may have had an impact on predicting the average memory use and throughput level.

Mr. Berners-Lee described the validation process for the Jackson model he had discussed previously. As the multiprogramming level increased, the model predicted that all devices would increase their utilization in proportion. Measured data conformed remarkably well to these predictions. As an aside, he mentioned that allowing for non-integer values of multiprogramming level seemed to overcome problems described by Cmdr. Rose, as well as compensating for unknown or low-level effects. A discovery he made showed that parts of the operating system were multiprogrammed with other parts, as well as with the user's job. Accounting for this effect also improved the accuracy of the model.

Mr. Wood chose a computer subsystem, network polling control, applying workload data to both queuing model approximations and discrete simulation techniques. He found that the use of actual line transmission speed as the service time for a line discipline consistently gave erroneous estimates of specific response time values. However, the overall shape of the response time curve produced by the queuing model tracked against the simulation results, if line service time was adjusted to account for polling delay and polling list management discipline.

The adjustment process was not straightforward. The shape of the response time vs utilization curve appeared to resemble a high order polynomial (i.e., cubic), whose number of saddle points and radius of curvature depended heavily on the actual values of polling delay time, polling discipline, and buffer space for transmission blocks [5,6].

Mr. Wallack took a different position with respect to validation of a closed queuing network model [7]. He contended that satisfactory accuracy cannot be obtained for performance predictions without including random access device activity in the model. He ran several benchmarks, and may have encountered a file structure that was more sensitive to the inclusion of device level service times. Reaction from the audience included speculation that for a highly utilized system, channel utilization would be the determining factor of response time delays. For lightly loaded systems, device service time anomalies would not affect system throughput. Another observation was that predictions of response time performance for individual applications requiring frequently utilized random access devices would be inaccurate.

NEW IMPLEMENTATIONS

Mr. Giammo began the afternoon session of queuing model implementations with a discussion of new approaches to general closed form solutions for queuing models [8]. He described the problem that few queuing formulae accept a product form solution for traffic intensity. Using a product form solution, he obtained a limiting distribution. It was not necessary to know the number of cycles between stations or what the per-visit time to a station was. Required was the aggregate amount of work to be performed at each station. In this concept, a job became an artificial work unit for which individual characteristics were no longer required. The change of classes by jobs, and the amount of time a job spends in a class were similarly irrelevant pieces of data. The macro approach to solutions for queuing network structures facilitated the evaluations of entire applications. The resource consumptions of an application were added to the aggregate totals to determine the input on system performance. Once the system performance prediction was established, applications or individual user performance were derived by reversing Norton's collapse theorem. To estimate the percentage contribution by each user or application, the network was expanded into all of its component sub-networks, rescaling and allocating traffic and service time data to conform to the sub-network structure.

Dr. Browne summarized the benefits of queuing model implementations. As an overall prediction of system performance, queuing models can achieve accuracies of well below 10% error. They also excel at describing the performance of subsystems (i.e., direct access storage), and exploring the importance of service time parameters for relative and comparative analysis. At this time they are not effective for analyzing operating system heuristic algorithms. Their biggest value is enabling the analyst to gain enough knowledge of the workings of a system to understand the performance experienced by users and adjust it to meet a wide spectrum of service levels. Dr. Browne described the most recent enhancements to the ASQ queuing model implementation [9]. These enhancements consisted of a hierarchical decomposition technique for the system and a method of including the restrictions imposed by a finite size executable memory on system performance indices.

Dr. Sauer provided information on a hybrid approach to performance prediction employing both analytical models (QNET4) and discrete simulation (APLOMB). APLOMB was used to verify the simplistic assumptions in the QNET4 models. QNET4 was used as a subsystem model technique to provide input to the APLOMB model of the overall system. No connection between the models was provided. The user must integrate the output from each solution technique to produce an overall result [10].

Dr. Agrawala discussed studies of traffic obtained from terminals to study the durability of the Poisson process as an acceptable approximation to the distribution of user traffic. Invariably, the distribution measured converged to a Poisson process whenever more than ten terminals were configured for the system and were active over the same time period [11]. Another experiment performed by Dr. Agrawala examined various ways of representing the operating system in the model. The most accurate representation of operating system activities treated parallel processes in the operating system as separate activities at the job level. This finding matched results of experiments performed by Mr. Berners-Lee.

Dr. Goldberg wrapped up the afternoon session by describing the use of a queuing model implementation in three distinctly different environments. Without providing any detailed results, he relayed the customer's satisfaction with the use of the queuing model to meet the performance product ojectives [12].

The critiques of the second day's sessions were equally favorable to those of the first day. The session on validation of queuing models was the overwhelming choice as the most popular session. Most attendees were surprised at the amount of validation which had been performed, and they were pleased to hear the successes the speakers achieved as well as the unusual accuracy of many of the results. Perhaps one fitting summary comment was "these sessions are still necessary from an educational viewpoint, but there is now enough strong evidence to show that the practitioner can obtain reliable results." Another reviewer said "I was happy to see the efforts that have taken place. The validation of specific theoretical models should continue, as well as the study of the validity of our assumptions, many of which still seem to be based on folklore."

The picture of queuing model use in the 20th century for ADP system performance prediction was clarified by the workshop. During the mid-1960's, analysts were concerned with the throughput of primarily batch systems. The packaged simulators used queuing models to predict the effect of multiprogramming on scheduling. Subsystem analysis focused on the random access device. Analysts of communication systems using single server queuing models to predict system response time treated the host as the hub of a star network. Some studies were done of the behavior of operating system dispatchers in time-sharing environments. The growth of system and application software in the early 1970's changed the primary congestion point of the system from I/O to the CPU. Queuing models were popularized by people studying how functions could be moved out of the host processor and into the network. The models were used in the operations phase of the system life cycle to answer more detailed design and configuration questions. This trend increased the concern about accuracy of these models. Fortunately, the measurement of system performance data was beginning in earnest. The quantity of data provided a basis to calibrate the models and determine their accuracy.

This present information technology is heading toward the development of complex computer-communications networks. Today, there are many permutations and combinations of alternative system structures capable of meeting the requirements of an organization. Queuing network models will be an increasingly valuable tool for the study of current and future systems, because they can be used to quickly search through many possible system approaches and find those solutions which are most cost-effective.

ACKNOWLEDGEMENTS

Thanks to Gerald D. Stocks, Registration Chairman, Linda Glaza, Terry Hammer and Sylvia Mabie for their excellent efforts to successfully organize the workshop.

REFERENCES

[1] B. Avi-Itzhak and D. P. Heyman, "Approximate Queuing Models for Multiprogramming Computer Systems," JORSA 21-6, Dec, 1973, pp. 1212-1230.

[2] C. M. Berners-Lee, "Four Years Experience with Performance Methodology for System Planning," proceedings of Computer Performance Evaluation session, EUROCOMP 1976, Sept 1976, pp. 165-187.

[3] L. Bronner, "An Introduction to Capacity Planning," IBM Washington System Center Technical Bulletin GG-22-9001-00, January 1977.

[4] C. A. Rose, "Measurement and Analysis for Computer Performance Evaluation," PHD Dissertation, George Washington University, Sept 1975, available from University Microfilm, Ann Arbor, MI.

[5] J. P. Bricault and I. Delgavis, "An Analysis of a Request-Queued Buffer Pool," IBM Systems Journal, Vol. 5, No. 3, 1966, pp. 148-157.

[6] J. H. Chang, "Terminal Response Times in Data Communications Systems," IBM Journal of Research and Development, Vol. 19, No. 3, May 1975, pp. 272-282.

[7] D. P. Gaver, "The Construction and Fitting of Some Simple Probabilistic Computer Models," Naval Postgraduate School Technical Report No. 55GV75011, January 1975.

[8] T. Giammo, "Extensions to Exponential Queuing Network Theory for Use in a Planning Environment," distributed at the Queuing Model Workshop, unpublished.

[9] R. M. Brown, J. C. Browne, and K. M. Chandy, "Memory Management and Response Time," Communications of the ACM, Vol. 20 No. 3, March 1977, pp. 153-165.

[10] C. H. Sauer, M. Reiser, and E. A. MacNair, "A Package for Solution of Generalized Queuing Networks," Proceedings of NCC 77, Vol. 46, pp 977-986.

[11] A. Agrawala and R. N. Brown, "On the Behavior of Users in the MEDLINE System," University of Maryland, Department of Computer Science Technical Report, May 1973.

[12] J. P. Buzen, "Principles of Computer Performance Modeling and Prediction," Infotech State-of-the-Art Report on Performance Modeling and Prediction , April 1977.

APPENDIX 1: FINAL AGENDA

MONDAY, 7 MARCH

Welcoming remarks - Mr. Philip Kiviat
USAF, FEDSIM, Washington, DC

I. INTRODUCTION

Mr. Mitchell Spiegel
USAF, FEDSIM, Washington, DC
Historical Overview of Queuing Model Applications

II. APPLICATION OF QUEUING MODELS TO FEASIBILITY AND SIZING STUDIES

Capt. Ed Askland/Capt. Larry Davis
USAF, AFDSDC, Montgomery, AL
Air Force Base Level Burroughs 3500 Sizing

Capt. Sam Pearson
USAF, FEDSIM, Washington, DC
A Communications Network for the Navy

Dr. Lindsay Wang
SAI, Arlington, VA
Feasibility Study of a Brokerage Network

Panel Discussion

III. APPLICATION OF QUEUING MODELS TO SYSTEM DESIGN AND PERFORMANCE MANAGEMENT

Dr. Lindsay Wang
SAI, Arlington, VA

Design of a Distributed Network

Mr. Conway Berners-Lee
ICL, London, UK
George III System Monitoring and Modeling Utility

Ms. Ann Dowdy
Mobil Oil, Houston, TX
Use of Queuing Models for Configuring Large Scale Systems

Dr. Leeroy Bronner
IBM, Gaithersburg, MD
Capacity Planning with Queuing Models

Mr. John Day
Burroughs, Goleta, CA
Designing Large Scale Systems Applications Using MAIDENS

Panel Discussion

TUESDAY, 8 MARCH

IV. QUEUING MODEL VALIDATION

Cmdr. Cliff Rose
NAVY, NELC, San Diego, CA
Problem of Validation and Inadequacy of Measurement Tools

Mr. Conway Berners-Lee
ICL, London, UK
Validating Queuing Models that Predict Operating System Performance

Mr. Jim Wood
R.L. Deal, Rosslyn, VA
Queuing Model Validation of a Communications Network Using a Discrete Simulator

Mr. Barry Wallack
DCA, Washington, DC
Validation of Analytical Models for H6000

Panel Discussion

V. NEW QUEUING MODEL IMPLEMENTATIONS

Mr. Tom Giammo
HEW, Baltimore, MD
New Theory and Its Implication for Queuing Model Implementations

Dr. Jim Browne
University of Texas, Austin, TX
Current ASQ Implementation

Dr. Charles Sauer
IBM, Yorktown Heights, NY
RESQ (QNET4)

Dr. Ashok Agrawala
University of Maryland, College Park, MD
Workload Characterization for Queuing Models

Dr. Robert Goldberg
BGS Systems, Lincoln, MA
BGS Systems BEST/1 Computer Performance Evaluator

Panel Discussion

A SIMULATION STUDY OF INITIATOR/TERMINATOR POLICY IN OS/MVT

E. Fiegl and N. Schneidewind

Naval Postgraduate School
Monterey, California 93940

An initiator is a task in IBM's OS/MVT which selects a job for execution and attempts to obtain the main storage, file space and devices which are necessary for job execution. A related task, the terminator releases these resources upon completion of the job. The order in which job classes are served by an initiator from the job queue is determined by the order of assigning job classes to an initiator. Initiators and their job class assignments are specified by the operator with start initiator commands at the console. The number of initiators started and the job class assignments significantly influence system performance. The number of initiators corresponds to the maximum degree of multiprogramming available. However, too many initiators could be detrimental to system performance because an initiator and terminator consume main storage during the execution of their functions. If the job input rate is low and there are many initiators, resources will be wasted. On the other hand, if the job input rate is high and there is an insufficient number of initiators, a large job queue will develop.

Key words: IBM OS/MVT; initiator policy; job scheduling; simulation.

1. Overview

The model simulates the main functions of the IBM OS/MVT Job Management routines. In general, the overall structure of the operating system is reflected in the structure of the model, but since OS/MVT is a very complex system some simplifications and limitations are necessary. They are described in the following parts.

The purpose of the model is to test different initiator strategies under certain job loads and operating conditions. To achieve this goal the number of Initiators (up to 15) and their associated job classes (up to 8 per Initiator) can be varied during the simulation. The model also allows important system parameters (size of main memory, input spool space, number of I/O devices, etc.) to be entered. By varying these parameters, the simulation program can be tailored to a certain extent to a given environment.

The job stream used during the simulation is generated by a job-generating module. This module can be modified to allow generation of job streams with different characteristics. Some statistical routines collect and print statistical and performance data upon user request.

A special problem is the simulation of the time used by Job Management routines, by other system tasks, and by the different user jobs. The basic time measurement in the model is elapsed step run time. This is the wall clock time counted from the beginning of step initiation to the end of step termination. The elapsed job run time is the sum of all elapsed step run times of a job. Included in this time is the CPU time used by the job, the time waiting for I/O, as well as the time used by the Job Management routines and other system tasks. Since the elapsed step run time has a range of one second to several minutes, one second is used as the basic time unit in the simulation model.

The programming language PL/1 was chosen as the simulation language for several reasons:

1) it is a block-structured language;
2) it allows good data structuring;
3) it is easy to use for I/O routines;
4) it is well supported at the Naval Postgraduate School (NPS), where the model was developed.

In addition, PL/1 allows nearly unrestricted variable names. This makes the program more readable and self-documenting.

A functional overview of the simulation model is given in Figure 1.

2. Supervisor Module

The supervisor module initializes and drives the entire simulation program. When it calls the initialization and modification routines, the user may enter the following parameters:

1) system modifications
   a) main memory (high address)
   b) main memory (low address)
   c) number of disk drives
   d) number of tape drives
   e) amount of input spool space
   f) amount of public direct access space)
2) run parameters
   a) number of jobs to be read
   b) simulation time
   c) job stream modifications
3) Initiator modifications
   a) number of active initiators (up to 15)
   b) associated job class(es) for each initiator
4) trace parameters
   a) simulation trace
   b) map of main memory usage
   c) statistics gathering

After these parameters are entered the timer module gets control. This module checks the simulation time table, which contains the times when the Reader and each active Initiator need attention. The timer always calls the next module, which is responsible for updating the attention time. This process is terminated when the simulation time or the input job stream is exhausted, whichever comes first.

At the end of each simulation step the user has the choice to stop or restart. If restart is chosen he may run the simulation with the same or new parameters.

3. Reader Module

It is assumed that the Reader is active during all simulation steps and that it resides in the upper part of the dynamic area in main memory. The user must note the amount of core used by the Reader when entering the main memory high address parameter.

During the initialization phase the job generating module places the requested number of jobs and their characteristics into the input job stream and also sets the time of the first job arrival into the simulation time table. When the Reader is called by the timer module it takes the next job from the input stream and enqueues it according to its class and priority into one of the job input queues. Then the Reader determines the time of next job arrival and places this time into the simulation time table as its new attention time.

If the input spool space is exhausted, the reading and enqueuing of jobs is delayed until another job terminates and enough spool space becomes available. Since the supervisor ends the simulation run after the requested number of jobs has been read, the Reader will never be called when the job stream has been exhausted.

4. Initiator Module

The Initiator module simulates the functions of job selection, waiting for work, region management, device allocation, data set allocation, direct access space allocation, step termination, and job termination. All information necessary to perform these functions is maintained in an Initiator table. Since the dimension of this table is 15 it is possible to run 15 Initiators concurrently. Each Initiator updates its time of next attention in the simulation time table.

4.1. Job Selection and Waiting for Work

An Initiator can be associated with up to 8 different job classes. To find the next job the input queues are searched in the order in which the classes were assigned to Initiators by the user.

If there is no job of the appropriate

class in the queues, the Initiator releases its region and is put into a "wait for work" state. This state is kept until a new job arrives. Then the Initiator gets a new region of a pre-defined minimum size and the queues are searched again. If a job is found it is dequeued and associated with its Initiator for further processing.

4.2. Region Management

When a job is selected the region size of its first step is determined and the region currently used by the Initiator is released. The new region is allocated from the top of the dynamic area in main memory. If insufficient continguous core is available, the Initiator is placed in a "wait for core" state. It is activated again for a new region allocation when another job ends and some core is released. The region management routines are called at the beginning of each job step.

It is assumed that the size of the dynamic area is fixed during the simulation. However, the user must set the upper and lower addresses. He can account for the size of the system queues by setting the appropriate lower address. He must also account for the amount of storage used by system tasks, by Reader(s), Writer(s) and other permanent programs by setting the appropriate upper address.

4.3. Device Allocation

In general, the allocation of I/O devices and I/O channels is not simulated in the model. Most devices are physically shareable (data cell, disks) or are made shareable using spooling techniques (card reader, printer, plotter). Evaluation of system logs has shown that normally all requests to such devices can be satisfied by the system immediately. The time overhead required for selection, allocation, and spooling is included in the elapsed run time of each job step.

However, this simplification is not valid in the case of tape drives and disk drives with removable disk packs. The allocation of these devices sometimes requires operator interaction or causes long additional waiting times until a requested tape or disk drive becomes available.

The device allocation routines handle the tape and disk requests of each job step. If a device is not available, an operator interaction is simulated. The operator answer could be "cancel" or "wait." In the first case the whole job is abended; in the second case the Initiator is put into a "wait for device" state. Whenever another job terminates the device allocation routines are activated again until all outstanding device requests can be satisfied for the current job step. In order to avoid long waiting times, all jobs which request more devices than are installed in the system are abended.

The type of operator answer ("cancel" or "wait") and his response time are drawn from a probability distribution.

The number of tape and disk drives can be set by the user, thus tailoring the simulation model to his needs.

4.4. Data Set Allocation

Only those data set allocations are of interest which require operator interaction, thus causing additional waiting time. It is assumed that for every requested tape and disk drive an appropriate volume has to be mounted. By placing the Initiator into a "wait" state, the data set allocation routines simulate the time needed by the operator to perform the mounting.

Independent of mounting requests are verification requests. Some data sets require an operator response to verify that a user is authorized to access a data set. This case is also simulated by the data set allocation routines. Since, for the model, the operator response time is of greater interest than the reason for an operator interaction, this case could also be used to account for any additional operator request which is otherwise not covered (channel separation request, etc.).

The response time for mounting disks, mounting tapes, or answering other requests is drawn from a probability distribution as described earlier for the device allocation routines. Also the possibility of job cancellation is included in the model.

4.5. Direct Access Space Allocation

Only the allocation of temporary space on public direct access devices is simulated. The total amount of public space within the system can be set by the user. If a space request of a job step cannot be satisfied the space allocation routine checks if there are other job steps active. If not, the current job will be abended, since its request can never be granted. Otherwise the Initiator is placed into a "wait" state until another job ends which might release some temporary space on public direct access devices.

4.6. Step and Job Termination

At the end of a step all requested disks and tapes are released and given back to the system. Temporary space on public direct access devices, however, is kept until job termination. If there is another step to process, control is given to the region management routines to start the next step.

At normal job termination as well as in case of job abending all system resources (tapes, disks, public direct access space, and input spool space) are released. The Initiator table is cleared and a new job can be selected. Job termination is also posted to the Reader which might be waiting for input spool space and to other Initiators which are in a state of waiting for system resources.

5. Writer Module

Spooled system output only is assumed for this simulation. This means that the Writer works independently from and concurrently with the Reader and Initiators. Since the amount of overhead due to multiprogramming with the Writer is already included in the elapsed job run times, no Writer function has to be simulated. However, as mentioned earlier the user must deduct the core size used by the Writer from the top of the dynamic area in main memory.

6. Statistical Module

Several statistical routines gather Initiator performance data. These data are maintained in a statistical table which can be written on a file upon user request. This file must then be processed and evaluated by a separate evaluation program.

As a second choice the user can request a simulation trace. Similar to the logs at the operator's console, all important events (job starting, job termination, initiator waiting for work, mount requests, etc.) are printed out. As a third choice a map showing the utilization of main memory at the end of each simulation step can be printed.

7. Source of Data

To drive the simulation model certain information about input job stream characteristics, system configuration, operator response times, etc., were necessary. To gather these data four main sources were used.

The first source was the IBM 360/67 computer center at the Naval Postgraduate School. An overview of the hardware configuration is given in Table 1; the job class definitions and the priority policy are listed in Table 2. With the installation of the HASP spooling system in September/October 1976, the Quickrun class was replaced by the input class 0, which was restricted to jobs using certain catalogued procedures only, using up to 180K of core and up to 20 seconds of CPU time. These restrictions were nearly the same as for the old Quickrun class, but since some changes were made in the catalogued procedures, about half of all old class A jobs together with nearly all old Quickrun jobs would now qualify for the new class 0. For the validation runs the characteristics of the old Quickrun class were simulated.

The second source was data collected from the System Management Facility (SMF) routines. These routines gathered statistics about every job processed by the computer system. Contained therein were: job name, job class, job priority, job arrival time and date, job starting time and date, job completion code, number of input cards, number of job steps, requested and used core per step, CPU time per step, elapsed time per step, sysout records per step, etc.

Only SMF data of the period from February to August 1976 were usable for the purpose of this study. Before this time period a completely different job class and priority specification was in effect. After this period some parameters used for the simulation model were no longer recorded due to the change to the HASP spooling system. SMF tapes of April, May, and August 1976, containing data of about 75,000 jobs, were therefore evaluated.

As a third source the complete set of system logs of August 1976 was available and used to extract certain parameters. These parameters included the number of Initiators and their associated job classes, the number of other system tasks active at the same time, and upper and lower addresses of the dynamic area in main memory.

Since the SMF tapes did not provide information about usage of tapes and disks, the system logs were also used to count the number of tape and disk mount requests and to evaluate data such as operator mounting

IBM 360/67 at NPS

No.	Unit	Description
2	2067-2	Processing Unit
1	2167-4	Configuration Control Unit
2	1052-7	Console Typewriter
2	2860-2	Selector Channel
2	2870-1	Multiplexor Channel
3	2365-12	Processor Storage (256K Bytes each)
5	MM365-12	Core Storage (256K Bytes each) - Lockheed
1	2820-1	Drum Control
1	2301-1	Drum Storage (4M Bytes)
3	2841-1	Disk Control
8	2311-1	Disk Drives (7.25M Bytes each)
1	2314-1	Disk Unit (8 Drives, 29M Bytes each)
2	5314	Disk Control - Plotter
16	4314	Disk Drives - Plotter (20M Bytes each)
1	2321-7	Data Cell (400M Bytes)
1	3803-1	Tape Control
5	3420	Tape Drives
1	2803-1	Tape Control
2	2402-1	Tape Unit (2 Drives each)
1	2821-1	Control Unit
1	2821-2	Control Unit
2	1403-N1	Printer
1	2501-B2	Card Reader
1	2540-1	Card Reader/Punch
1	110	Plotter Control - CALCOMP
2	765	Plotters - CALCOMP
1	2702-1	Transmission Control Unit (30 Ports)
24	2741	Communication Terminals
8	----	Video Display Units (assorted vendors)
1	2250-1	Graphic Display Unit
1	Tek4012	Display Terminal - Tektronix
1	Tek4610	Hard-Copy Device - Tektronix
1	PIX	Paradyne PIX/Remote Job Entry
1	2701	Data Adapter Unit (with PDA)

Table 1. System Hardware at NPS

times, operator response times to other system requests, and the number of job cancellations by the operators.

Last, but not least, the operators themselves and other members of the computer center staff at NPS provided some valuable input for the collection and evaluation of system parameters.

Effective 1 February 1976

Job Class Definitions

Class	Region	Time	Tape/Jobstep
Q	QUICKRUN		none
A	180K	20s	none
B	180K	2m	≤ 2
C	250K	5m	≤ 2
D	250K	5m	≤ 2
E	350K	5m	≤ 2
F	400K	30m	none
J	>400K	>30m	none
K	>400K	>30m	any

Comments:

1. Execution in each class will be on First-come First-served (FCFS) basis.

2. Classification scheme ignores SYSOUT and SYSDA requirements. Printing priority is considered separate from execution priority and is based on the actual number of lines generated.

Table 2. Job Class Definitions at NPS

8. Job Stream Characteristics

Very important for an effective simulation run were parameters which characterized the input job stream. Members of the computer center staff and students had analyzed the input job stream at the NPS computer center. But since these studies were based on jobs rather than steps, as required by the simulation model, these studies were not usable and a new evaluation had to be made.

Most of the job characteristics were extracted from the SMF tapes. When working with these tapes a few problems arose. There was no class D job observed and the number of J and K class jobs was very small. In addition some jobs in undefined job classes were present. An explanation for this was that the operators used to start one Initiator with an undefined job class. Then they reset jobs from classes J and K and the very few jobs from class D, and selected these manually for initiation. For the simulation model, classes D, J, K, and all undefined classes were collected into one class K.

Also some jobs used more core than allowed by their job class. The explanation again was that operators reset jobs from one class to another. During the evaluation these jobs were filtered out and added to job class K.

Elapsed time and core used were not recorded for the Quickrun jobs. Since these jobs had the same time and core restrictions as class A jobs (core up to 180K, CPU time up to 20 sec.), the class A distribution was assumed.

To obtain relatively stable, but still representative data the time period from 10 a.m. to 5 p.m. each day was selected. The data collection was further restricted to those days with more than 500 job arrivals within this time. 47 days of the months of April, May, and August 1976 met these requirements. Figure 2 gives a histogram of job arrivals in April 1976. With this approach untypical conditions which occur at night and on weekends and holidays were eliminated. Although the time of observation covered only about 15% of the total hours within the three-month period, 25,532 or more than one third of all job arrivals were included.

To obtain the distribution of job arrivals a 2-hour period in each month was selected at random and the arrivals per minute were counted. As shown in Table 3 in each job class the distribution was very close to a Poisson distribution. In fact, the observed values easily passed a 95 percentile chi-square test to match the theoretical values. Thus for the job arrivals in the simulation model a Poisson distribution with exponentially distributed interarrival times was used. The mean job arrival rate was 9.294 jobs per minute.

Other parameters evaluated from the SMF tapes were distribution of job classes, number of steps per job, number of input cards per job, core used per step, and elapsed time per step. Histograms are given in Figures 3 through Figure 7.

It was felt that the distribution of elapsed step time might be approximated by a Gamma or possibly a Weibull distribution. Although a great amount of work was spent to match the observed values with those theoretical functions, no relationship could be found.

In the simulation model the amount of public direct access space was one input parameter. The storage of system output records was only one part of this space, but other data were not available. An evaluation of the job completion codes, however, showed that within the observed time periods no job abended because of lack of public direct access space. Thus for the simulation runs no public direct access space was requested.

The number of disk and tape mount requests and the number of other system requests were evaluated from the system logs of August 1976. Only the total number of requests could be counted, but information about the associated job classes was not available. For the simulation model it was assumed that the probability of requests was the same for all job classes.

	Class A		Class B		Class C	
i	F_o	F_{th}	F_o	F_{th}	F_o	F_{th}
0	67	66.41	88	87.43	105	104.15
1	39	39.29	26	26.69	13	14.75
2	11	11.62	6	4.38	2	1.05
3	2	2.29	0	0.46	0	0.05
4	1	0.34	0	0.04	0	0.00
5	0	0.04	0	0.00	0	0.00

	Class E/F		Class K		Class QR	
i	F_o	F_{th}	F_o	F_{th}	F_o	F_{th}
0	118	118.01	109	109.49	106	105.02
1	2	1.97	11	10.04	12	14.00
2	0	0.02	0	0.46	2	0.93
3	0	0.00	0	0.01	0	0.04

F_o: observed number of 1-min. intervals with i arrivals

F_{th}: theoretical number of 1-min. intervals with i arrivals assuming Poisson distribution

Table 3. Distribution of Job Arrivals

9. Operator Response Times

The system logs were also used to evaluate the operator volume mounting times, their response times to other system requests, and the number of jobs cancelled by the operators because a request could not be satisfied. One problem for the evaluation was the fact that the system requests had no time stamps. In most cases the time could be estimated within a 10-second range from other system messages with time stamps just above and below the request messages. For tape and disk mounts there were also no direct operator answers on the system logs, but in a certain number of cases the actual mounting time could be estimated from other system messages. Here again only those cases were evaluated where the estimation could be made within a 10-second time range. Using this approach a total of about 700 operator response times could be used. A histogram of the probability distribution per job step, separated into the cases for tape mount, disk mount, and other system requests, is given in Figure 8.

The relatively high probability of short reaction times to tape and disk mount requests came from the fact that the requested volumes were already pre-mounted and the devices had only to be varied on-line.

The number of jobs cancelled by the operators because a request could not be satisfied could be counted exactly: 49 jobs or 1.31% out of 3,735 jobs.

10. System Parameters

The following system parameters were used to tailor the simulation model to the environment at the Naval Postgraduate School as it was available to the user during the time period April to August 1976:

1) number of tape drives: 9;
2) number of disk drives: 3;
3) amount of input spool space: 45,000 card images;
4) amount of direct access space: 100 records;
5) main memory (high address): 1140K;
6) main memory (low address): 140K.

The actual number of disk drives in the system was much higher (see Table 1), but without special arrangements only three were free for general users. Also only five of the eight core boxes were routinely available for OS/MVT.

Assuming a mean of 300 input cards per job the amount of 45,000 card images was equivalent to the current system spooling capacity of about 150 jobs.

As mentioned earlier the direct access space was not used as a parameter for the simulation runs. Thus the number of 100 records had no meaning.

The upper and lower addresses of main memory were the bounds of the dynamic area. These bounds varied depending on the load on the system. The values of the bounds used were mean values observed from the system logs.

11. Validation

In order to show the usefulness and validity of the simulation model it was parameterized to match the characteristics of the computer center installation at the Naval Postgraduate School. The parameters used for input job stream characteristics, system configuration, and operator response times were mostly the same as described in the previous chapter. The outcome of the simulation runs could be compared with data observed from the actual system.

An unexpected problem arose when searching for console log data which could be compared with simulation results. Within August 1976, the only month for which both SMF tapes and system logs were available, there were 15 days which qualified for use in the model (more than 500 job arrivals in the period from 10 a.m. to 5 p.m.). At first this seemed to be a sufficient number of days to choose from, but a more detailed examination showed that none of these days could be used. For each day there was either system down time, or the operators held the queues up to 50 minutes, or both. In addition, the operators reset up to 40 jobs daily from one class into another or changed job priorities. The longest continuous time interval without down time, or queue hold, or with few resets was 4.5 hours. It was observed from 10:00 a.m. to 2:30 p.m. on August 16, 1976. This was a rather short time period for validation purposes, but for lack of better data it had to be used.

The job arrival rate (1.2407 jobs per min.) and the job class distribution (see Table 4) within this time interval differed significantly from the values observed over the three-month period. The appropriate modification in the simulation model was made.

Class:	A	B	C	E	F
Prob.:	.3403	.1940	.1045	.0179	.0149
Class:		K	QR		
Prob.:		.0716	.2567		

Table 4. Distribution of Job Classes (Validation Runs)

Table 5 shows the usage of Initiators and their associated job classes during the validation runs. This set-up differed only in two minor points from the actual usage.

Class 0 in the validation runs represented the old Quickrun class and class K was used in the validation runs instead of class M.

	TIME			
Initiator	10:00	12:00	12:18	14:30
1	OAB	OAB	OAB	OAB
2	OAB	OAB	OAB	OAB
3	OABC	OABC	OABC	OABC
4	OABC	OABC	OABC	OABC
5	OABCE	OABCE	OABCE	OABCE
6	K	KABFEC	KABFEC	KABFEC
7	-	-	AB	AB

Table 5. Initiator Usage (Validation Runs)

Forty validation runs with different input job streams were made. A comparison between the actual values and the mean values from the simulations is given in Table 6. More jobs were started in some classes than arrived because the queues were partly filled with jobs which had arrived during the previous hour.

The ratio of jobs started to jobs arrived observed from the evaluation runs was very close to the actual ratio for the job class 0 (=Quickrun) and for the total. Good results were also obtained for classes A, B, and C. Since the sample size for classes E and F was small the results were meaningless. Class K results were not representative since in the actual system class M was used for K class jobs and these jobs were selected by the operator.

Due to lack of more usable data no further comparison against actual system performance could be made. The small sample size available for this kind of validation did not allow a definitive statement about the accuracy of the results.

Numerous additional validation runs have been made to check individual components of the model (region management, device allocation, etc.) and to test boundary conditions (limitation in number of devices, core size, etc.). All of these runs showed the expected results.

However, one unusual result was observed. Although the job arrival distribution generated by the simulation model closely approximated the desired distribution for a sample size of 10,000 jobs, the job arrival rate for the first 600 jobs was always too high for a given seed. To overcome this anomaly a new feature was added to the model. Upon user's request the seed for the random number generator was modified by the value of the computer clock. It was then possible to change the seed at random. When this feature was used in additional simulation runs the unusual statistical pattern was no longer observed.

Since the future use of the simulation model is to compare the relative merits of different Initiator strategies rather than to predict absolute performance, it was sufficient to assure that the principal characteristics of the Job Management functions were reasonably well simulated. The results so far demonstrate the correct functioning of the simulation model.

Bibliography

1. Afifi, A. A. and Azen, S. P., Statistical Analysis, A Computer Oriented Approach, Academic Press, 1972.

2. Barron, D. W., Computer Operating Systems, Chapman and Hall Ltd., 1971.

3. Browne, J. C., Lan, J. and Baskett, F., "The Interaction of Multiprogramming Job Scheduling and CPU Scheduling," Proceedings of AFIPS FJCC, Vol. 41, Part 1, p. 13-22, 1972.

4. Colin, A.J.T., Introduction to Operating Systems, American Elsevier Inc., 1971.

5. Cuttle, G. and Robinson, P.B., Executive Programs and Operating Systems, American Elsevier Inc., 1970.

6. Flores, I., Computer Programming System/360, Prentice-Hall, 1971.

7. Flores, I., Job Control Language and File Definition, Prentice-Hall, 1971.

8. Flores, I., OS/MVT, Prentice-Hall, 1973.

9. Freeman, P., Software Systems Principles, Science Research Associates, 1975.

10. Hoare, C.A.R. and Perrot, R.H., Operating Systems Techniques, Academic Press, 1972.

11. IBM System/360 Operating System: Introduction, 5th ed., IBM, 1972.

12. IBM System/360 Operating System: MVT Guide, 6th Ed., IBM, 1974.

13. IBM System/360 Operating System: MVT Job Management, Program Logic Manual, 10th ed., IBM, 1971.

14. IBM System/360 Operating System: MVT Supervisor, 7th ed., IBM, 1972.

15. IBM System/360 Operating System: Operator's Reference, OS Release 21.7, 4th ed., IBM, 1974.

16. Katzan, H., Jr., Computer Organization and the System/370, Van Nostrand Reinold, 1971.

17. Katzan, H., Jr., Operating Systems, Van Nostrand Reinold, 1973.

18. Katzan, H., Jr., Information Technology, Petrocelli Books, 1974.

19. Madnick, S.E. and Donavan, J.J., Operating Systems, McGraw-Hill Book Co., 1974.

20. Sayers, A. P., editor, Operating Systems Survey, Auerbach Publishers, 1971.

21. User's Manual, W. R. Church Computer Center, 2nd ed., Naval Postgraduate School, Monterey, California, 1974.

Actual Data:

	Classes							
	A	B	C	E	F	K	QR	Total
A:	114	65	35	6	5	24	86	335
S:	126	63	32	0	3	9	86	319
R:	1.105	.969	.914	.000	.600	.375	1.000	.9522

Validation Results:

	Classes							
	A	B	C	E	F	K	QR	Total
A:	127	69	40	5	7	29	77	354
S:	131	71	39	3	1	20	77	343
R:	1.027	1.019	.968	.621	.157	.711	1.008	.968

A: Number of jobs arrived

S: Number of jobs started

R: Ratio jobs started to jobs arrived

Table 6. Validation Results

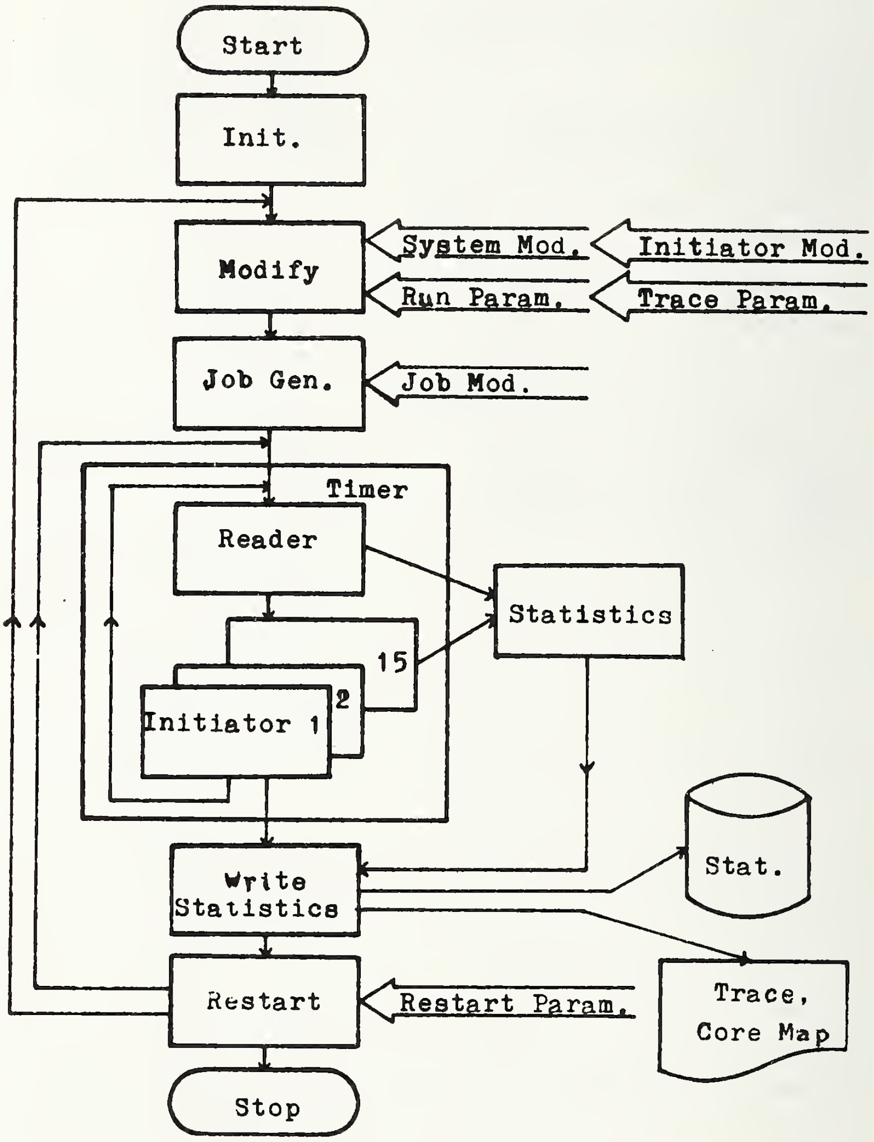
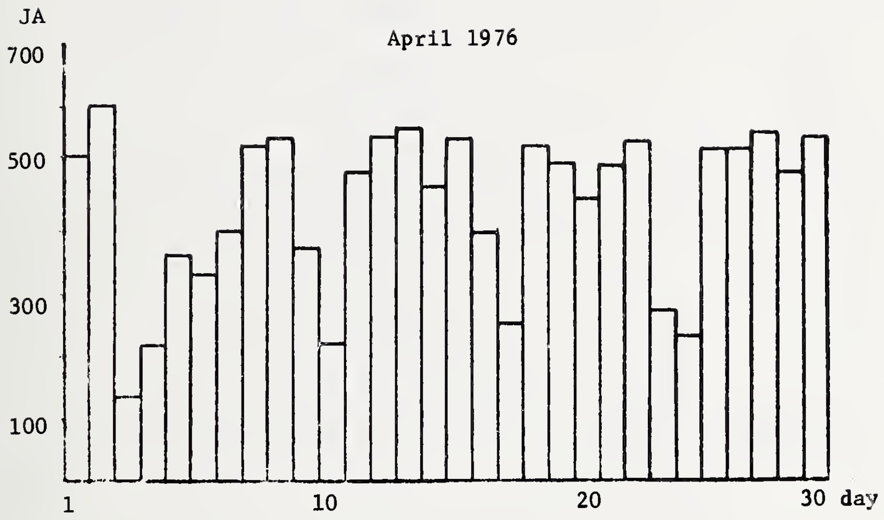


Figure 1. Structure of the simulation model.



JA: Number of job arrivals between 10 a.m. and 5 p.m.

Figure 2. Job arrivals (April 1976).

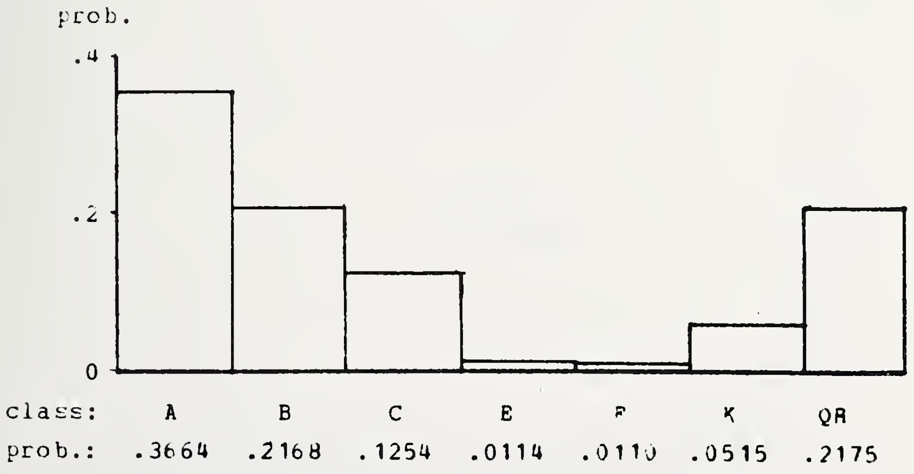


Figure 3. Job class distribution.

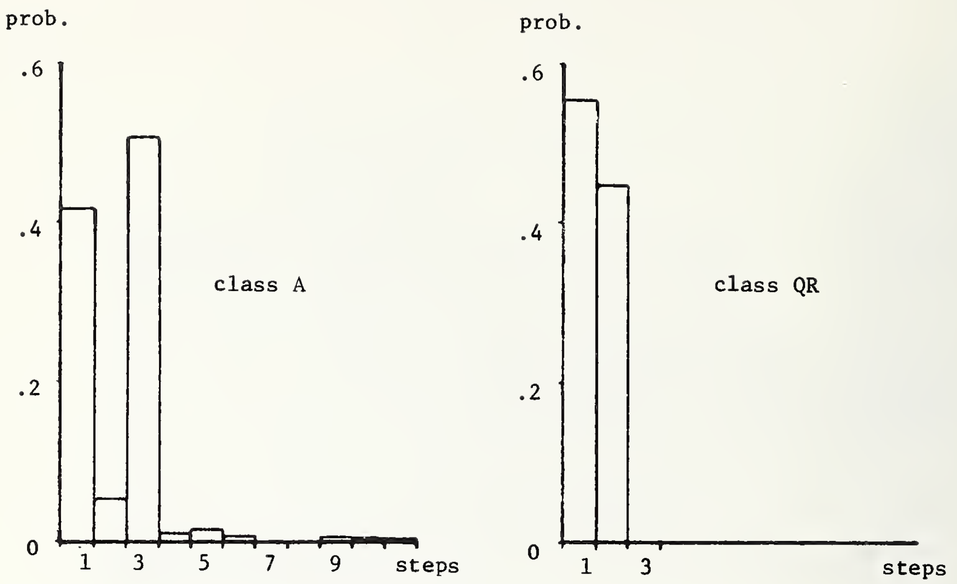


Figure 4. Histogram: job steps per class.

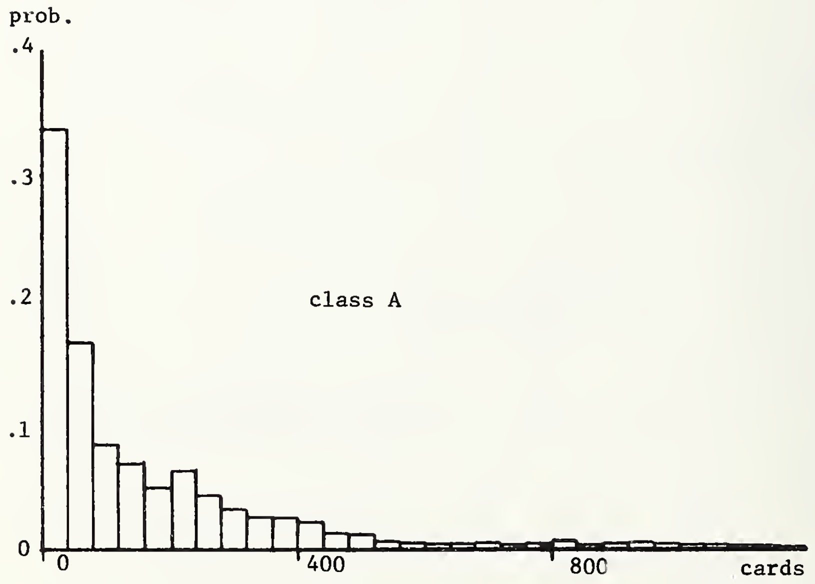


Figure 5. Histogram: input cards per job.

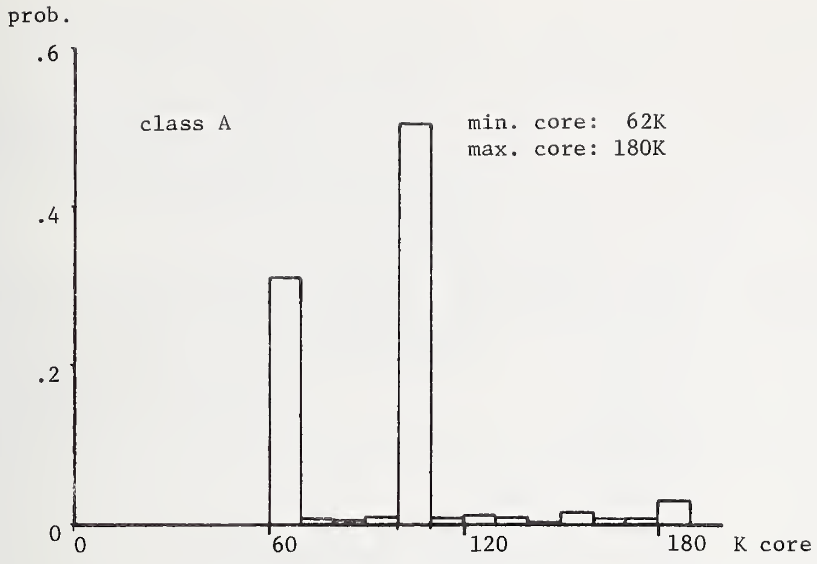


Figure 6. Histogram: core used per step.

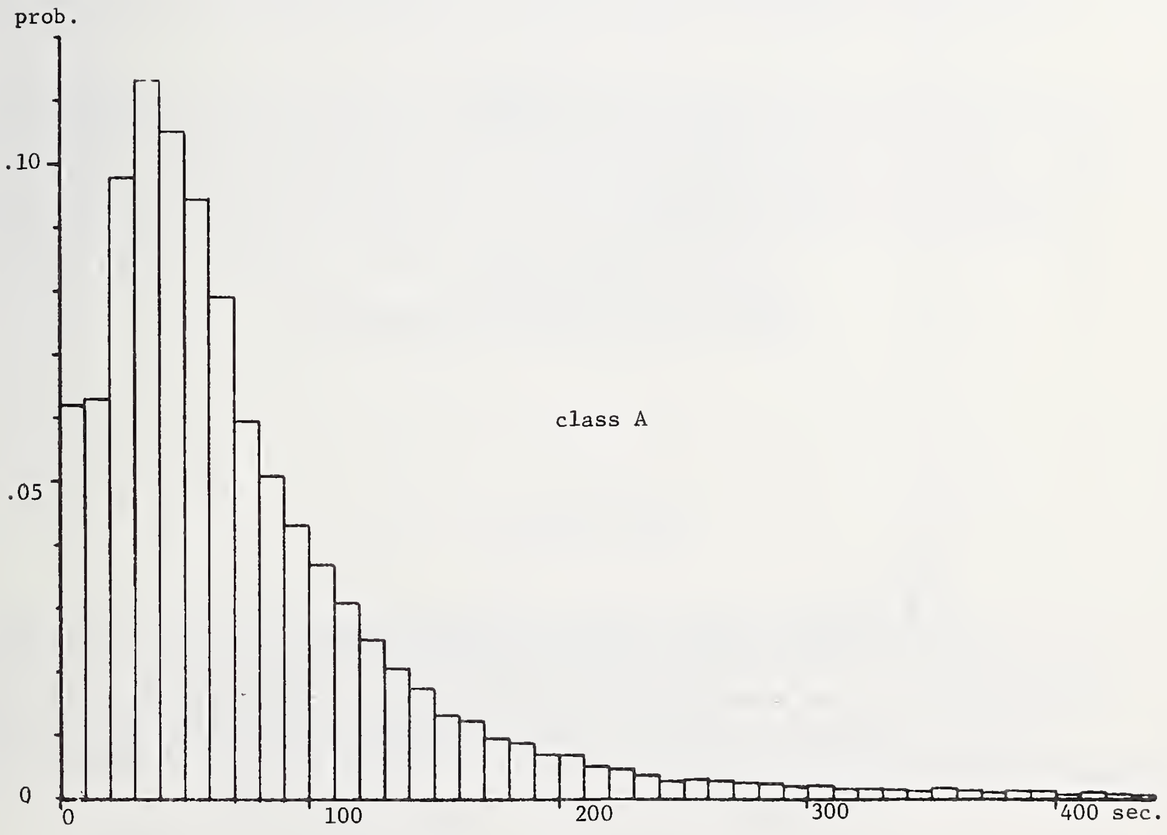


Figure 7. Histogram: elapsed step run time.

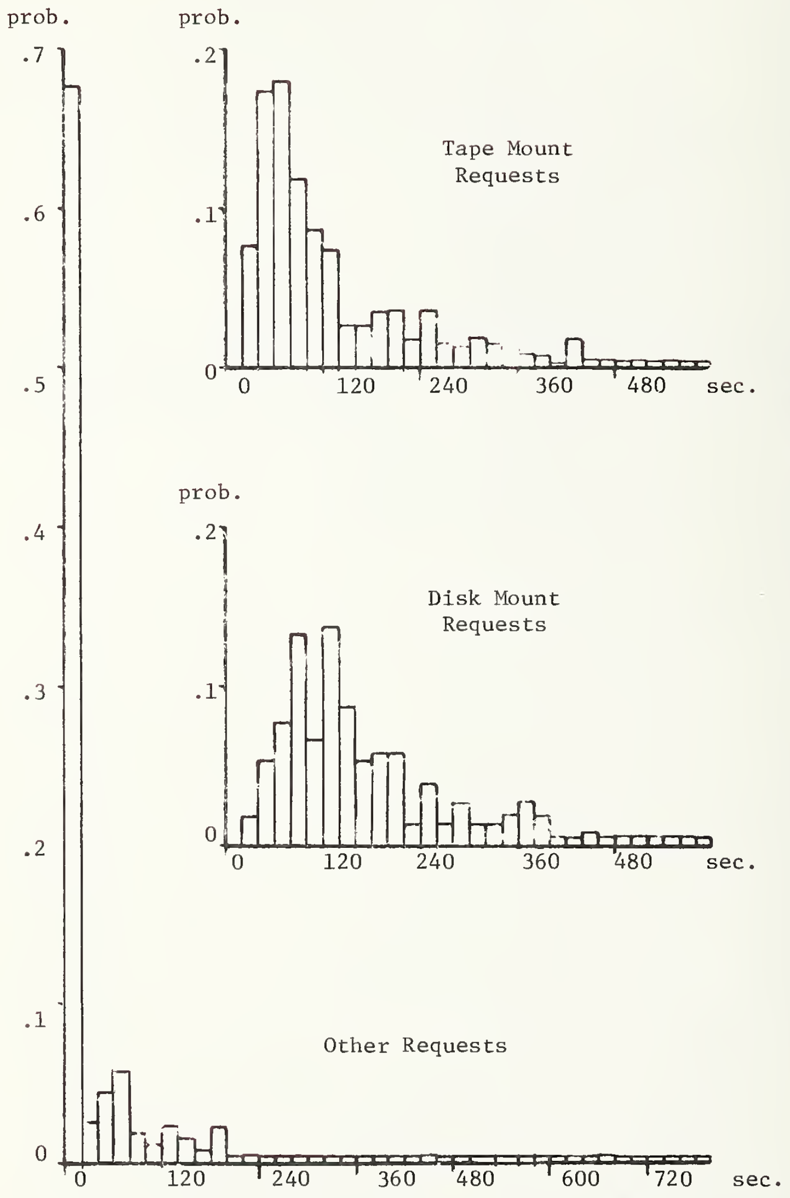


Figure 8. Histogram: operator response times.

NBS-114A (REV. 7-73)

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET	1. PUBLICATION OR REPORT NO. NBS SP 500-18	2. Gov't Accession No.	3. Recipient's Accession No.
4. TITLE AND SUBTITLE COMPUTER SCIENCE & TECHNOLOGY: Computer Performance Evaluation Users Group CPEUG 13th Meeting			5. Publication Date September 1977
			6. Performing Organization Code
7. AUTHOR(S) Editors: Dennis M. Conti and Josephine L. Walkowicz			8. Performing Organ. Report No.
9. PERFORMING ORGANIZATION NAME AND ADDRESS NATIONAL BUREAU OF STANDARDS DEPARTMENT OF COMMERCE WASHINGTON, D.C. 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

Library of Congress Catalog Card Number: 77-600040

16. ABSTRACT *(A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here.)*

The Proceedings record the papers that were presented at the Thirteenth Meeting of the Computer Performance Evaluation Users Group (CPEUG) held October 11-14, 1977 in New Orleans. The technical presentations were organized around the three phases of the ADP Life Cycle: the Requirements Phase (workload definition), the Acquisition Phase (computer system and service selection), and the Operational Phase (performance measurement and prediction methods). The program of CPEUG 77 is also included and serves as a Table of Contents to the Proceedings.

17. KEY WORDS *(six to twelve entries; alphabetical order; capitalize only the first letter of the first key word unless a proper name; separated by semicolons)* ADP life cycle; computer performance evaluation; computer performance measurement; computer performance prediction; computer system acquisition; conference proceedings; CPEUG; hardware monitoring; on-line system evaluation; prediction methods; queuing models; simulation; software monitoring; workload definition.

18. AVAILABILITY [✓ X] Unlimited	19. SECURITY CLASS (THIS REPORT) UNCLASSIFIED	21. NO. OF PAGES 241
☐ For Official Distribution. Do Not Release to NTIS		
[✓ X] Order From Sup. of Doc., U.S. Government Printing Office Washington, D.C. 20402, SD Cat. No. C13.10:500-18	20. SECURITY CLASS (THIS PAGE) UNCLASSIFIED	22. Price $ 4.00
☐ Order From National Technical Information Service (NTIS) Springfield, Virginia 22151		

ANNOUNCEMENT OF NEW PUBLICATIONS ON COMPUTER SCIENCE & TECHNOLOGY

Superintendent of Documents,
Government Printing Office,
Washington, D. C. 20402

Dear Sir:

Please add my name to the announcement list of new publications to be issued in the series: National Bureau of Standards Special Publication 500-.

Name ____

Company ____

Address ____

City ________________ State ____________ Zip Code __________

(Notification key N-503)

☆ U. S. GOVERNMENT PRINTING OFFICE : 1977 244-785/6557

NBS TECHNICAL PUBLICATIONS

PERIODICALS

JOURNAL OF RESEARCH reports National Bureau of Standards research and development in physics, mathematics, and chemistry. It is published in two sections, available separately:

• Physics and Chemistry (Section A)

Papers of interest primarily to scien orking in these fields. This section covers a br ıge of physical and chemical research, wit ır emphasis on standards of physical measur , fundamental constants, and properties of m ıssued six times a year. Annual subscription: D c, $17.00; Foreign, $21.25.

• Mathematical Sci Section B)

Studies and com ıs designed mainly for the mathematician and etical physicist. Topics in mathematical statis ıeory of experiment design, numerical analysi oretical physics and chemistry, logical design programming of computers and computer syst nort numerical tables. Issued quarterly. Annual s scription: Domestic, $9.00; Foreign, $11.25.

To begin publication as a single journal, July-August, 1977

DIMENSIONS/NBS (formerly Technical News Bulletin)—This monthly magazine is published to inform scientists, engineers, businessmen, industry, teachers, students, and consumers of the latest advances in science and technology, with primary emphasis on the work at NBS. The magazine highlights and reviews such issues as energy research, fire protection, building technology, metric conversion, 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, $12.50; Foreign, $15.65.

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 world-wide program coordinated by NBS. Program under authority of National Standard Data Act (Public Law 90-396).

NOTE: At present the principal publication outlet for these 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. N.W., Wash. D. C. 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 purpose of the standards is to establish nationally recognized requirements for products, and to 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 **above** *NBS publications from: Superintendent of Documents, Government Printing Office, Washington, D.C. 20402.*

Order **following** *NBS publications—NBSIR's and FIPS from the National Technical Information Services, Springfield, Va. 22161.*

Federal Information Processing Standards Publications (FIPS PUBS)—Publications in this series collectively constitute the Federal Information Processing Standards Register. 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, $25.00 ; Foreign, $30.00.

Liquified Natural Gas. A literature survey issued quarterly. Annual subscription: $20.00.

Superconducting Devices and Materials. A literature survey issued quarterly. Annual subscription: $30.00 . Send subscription orders and remittances for the preceding bibliographic services to National Bureau of Standards, Cryogenic Data Center (275.02) Boulder, Colorado 80302.

U.S. DEPARTMENT OF COMMERCE
National Bureau of Standards
Washington, D.C. 20234

OFFICIAL BUSINESS

Penalty for Private Use, $300

POSTAGE AND FEES PAID
U.S. DEPARTMENT OF COMMERCE
COM-215

U.S.MAIL

SPECIAL FOURTH-CLASS RATE
BOOK