COMPUTER SCIENCE & TECHNOLOGY:

COMPUTER PERFORMANCE EVALUATION USERS GROUP

CPEUG
15th Meeting

NBS Special Publication 500-52
U.S. DEPARTMENT OF COMMERCE
National Bureau of Standards

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

COMPUTER SCIENCE & TECHNOLOGY:

Computer Performance Evaluation Users Group (CPEUG)

Proceedings of the Fifteenth Meeting
held at San Diego, California
October 15-18, 1979

Editor:
James E. Weatherbee

Conference Host:
Naval Personnel Research
and Development Center
San Diego, California

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
Luther H. Hodges, Jr., Under Secretary
Jordan J. Baruch, Assistant Secretary for Science and Technology
NATIONAL BUREAU OF STANDARDS, Ernest Ambler, Director

Issued October 1979

Reports on Computer Science and Technology

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

National Bureau of Standards Special Publication 500-52

Nat. Bur. Stand. (U.S.), Spec. Publ. 500-52, 240 pages (Oct. 1979)
CODEN: XNBSAV

Library of Congress Catalog Card Number: 79-600123

U.S. GOVERNMENT PRINTING OFFICE
WASHINGTON: 1979

For sale by the Superintendent of Documents, U.S. Government Printing Office
Washington, D.C. 20402 - Price $5.50

Stock Number 003-003-02118-1

CPEUG 79

FOREWORD

Those of us involved in the many aspects of computer performance evaluation can hardly think of doing our tasks without the knowledge and expertise gained at some computer conference. For many of us some of the knowledge came from past CPEUG meetings, and from papers published in CPEUG proceedings. A quick look at this year's program will show, I am sure, that the 79 Conference will continue to broaden your knowledge and expertise in CPE technology.

The theme of CPEUG 79 is "The Expanding Scope of CPE." For a few days, we will stop, learn, listen, think, and discuss where we are, and where we are going with performance technology. When we think about computer performance, such issues as computer and network performance, tools and techniques, workload modeling and forecasting, benchmarking and simulation immediately come to mind. But other performance issues are frequently overlooked. CPE in auditing -- How management views CPE -- CPE and the human factors -- CPE in systems design and software: these and other issues and directions in computer performance evaluation are the focus on this year's conference.

For better or worse, computing is now involved in every type of Government activity. Computers make decisions for management -- they issue large sums of money -- they do clerical and auditing work and they even monitor other devices and machines. Everybody is seeking for better performance tools while some are asking, what are the CPE technician's responsibilities in these important areas? To help answer this question we established the theme for the '79 Conference as our objective -- to point out new directions for our profession.

As we discuss the expanding scope of CPE during this year's conference you will see that the "good old days" of the strictly technical hardware performance conferences are gone. Performance management, improving productivity of people, programs, systems, and software, along with evaluating and improving hardware performances -- these are the issues of today.

CPEUG was founded in 1971 by the United States Air Force. It is now sponsored by the National Bureau of Standards as part of its mission to develop guidelines and standards for improving the utilization and procurement of Federal ADP resources. Fourteen prior conferences in eight years were held to promote more efficient and effective performance of the Government's valuable computer resources.

We welcome you to CPEUG 79, and wish you an interesting and worthwhile experience while you are here.

My sincere thanks go to the many people whose efforts have gone into bringing you this fine conference, and especially to those who have dedicated their time, talent, and effort to serve on the 79 Committee. Their names appear elsewhere in these proceedings.

Harry J. Mason, Jr.
Chairman, CPEUG
October, 1979

CPEUG 79

PREFACE

The theme of this year's Conference, "The Expanding Scope of CPE," is more completely stated as "The Roles of CPE in an Expanding Technology." The consistent reduction in the cost of semiconductor components is having a dramatic effect on commercial computers. Small, fast, and cheap computer mainframes, together with high bandwidth digital communications channels at moderate cost, make extensive use of distributed processing more attractive. Under these circumstances, there is increasing debate in the ADP community concerning the continued economic viability of the large central computer installation. This debate is important to CPE practitioners, as CPE has been associated traditionally with large central computers. The economies of scale achieved in other areas by large central computers have been applicable to CPE. Hardware monitors, CPE software, and qualified analysts have been used most effectively when their cost could be amortized over a large amount of equipment and a large base of applications and customers. Similarly, some of the most impressive successes of CPE have been efforts whose results could be magnified by frequency of use (such as optimizing a critical loop in a low-level operating system module) or by number of affected applications. How will CPE, traditionally associated with large central computers, change in an era of smaller and cheaper hardware and improved digital communications? The debate on this subject will likely continue for some time, and it is critical to the discipline of CPE. One of CPEUG's lasting contributions may well be providing forums for this debate (both formal and informal). The questions posed here are met directly in this year's keynote address and keynote panel, and they are present in the remainder of the program.

The technical program this year is divided into two major sections, corresponding to two views of CPE. The first section of the program focuses on the application of CPE to installation management, while the second section focuses on the methodologies and tools of CPE as a technical discipline. In a sense, the first section addresses goals or ends, while the second section addresses means. The session on Workload Analysis and Capacity Planning forms a bridge between these major sections. The subjects covered in this session are of increasing interest and concern to installation managers, yet they require some very sophisticated technical methods.

The technical program this year can best be viewed in the context of the technical programs of preceding Conferences. In this regard, I will note a few key areas of comparison. The session on CPE in Auditing is new, although individual papers and tutorials on this subject have been presented at past Conferences. When viewed together, the papers in the second section of the program display a continued and increasing emphasis on rigorous and quantitative methodology in designing CPE experiments and in analyzing the data from these experiments. Several papers reflect a growing concern with the end user of computer services. In general, this year's program represents an evolution based solidly on previous Conferences. If this observation can be applied to the questions raised above, it suggests an evolution in the discipline of CPE to meet a technology which often seems to be undergoing revolutionary change.

The technical program represented here is the work of many hands. The names associated with these hands appear elsewhere in the Proceedings -- members of the Conference Committee, session chairpersons, authors, and tutors. The Conference referees merit special recognition. These individuals worked against almost impossible deadlines and contributed in a very important way to the overall quality of the technical program.

John Bongiovanni
Program Chairman

CPEUG 79

ABSTRACT

The Proceedings record the papers that were presented at the Fifteenth Meeting of the Computer Performance Evaluation Users Group (CPEUG 79) held October 15-18, 1979 in San Diego, California. With the theme "The Expanding Scope of CPE," CPEUG 79 focused on changes in CPE techniques that will occur in an era of increased use of distributed processing techniques. The program was divided into two parallel sessions with one session devoted to the presentation of technical papers on previously unpublished work and the other devoted to tutorials and case studies. The technical papers fall into one of two general categories, (1) application of CPE in installation management and (2) methods and tools of CPE as a technical discipline, and are presented in the Proceedings in their entirety. Summaries of several of the tutorials and case studies are also presented.

Key words: Computer performance evaluation; computer performance measurement; computer performance prediction; computer system acquisition; conference proceedings; CPEUG; CPE in auditing; installation management; hardware monitoring; on-line system evaluation; prediction methods; queuing models; simulation; software monitoring; workload definition.

CPEUG 79

CPEUG ADVISORY BOARD

CPEUG 79

CONFERENCE COMMITTEE

° CONFERENCE CHAIRMAN	Harry J. Mason, Jr. U.S. General Accounting Office Washington, DC
° PROGRAM CHAIRMAN	John J. Bongiovanni, Jr. Air Force Data Services Center Washington, DC
° PUBLICATIONS CHAIRMAN	James E. Weatherbee FEDSIM/MV Washington, DC
° REGISTRATION AND ARRANGEMENTS	Margaret Maulin DOD Computer Institute Washington, DC
° PUBLICITY CHAIRMAN	Judith G. Abilock The Mitre Corporation McLean, VA
° FINANCE CHAIRMAN	Carol B. Wilson National Bureau of Standards Gaithersburg, MD
° LOCAL HOST	Mark Underwood Navy Personnel Research and Development Center San Diego, CA

REFEREES

Steve Dorman
United Airlines
San Francisco, CA

Larry Frazier
Air Force Data Systems Design Center
Montgomery, AL

Lloyd Hasche
Strategic Air Command
Omaha, NB

Steve Hunt
Headquarters Air Force
Washington, DC

James Mulford
International Computing Company
Dayton, OH

John Peterson
FEDSIM/AY
Washington, DC

Mitchell Spiegel
FEDSIM/NA
Washington, DC

CPEUG 79

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

TABLE OF CONTENTS (continued)

CPE In Auditing

DATA PROCESSING INFORMATION UTILIZATION
AN AUDIT PERSPECTIVE

Michael J. Thibault

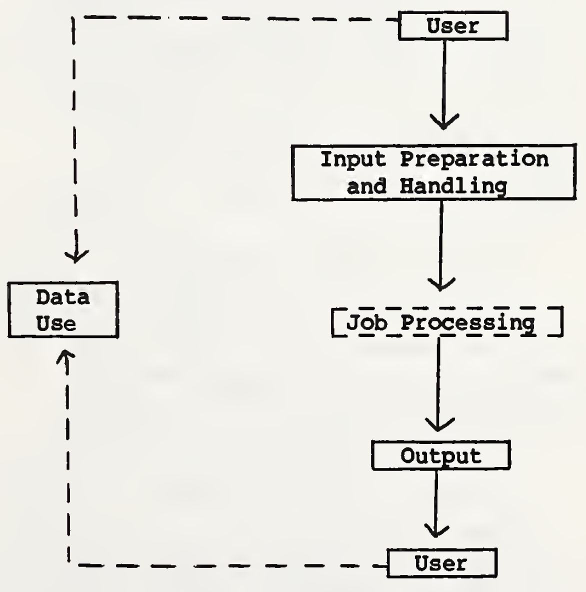
Defense Contract Audit Agency
San Francisco Region
San Francisco, CA

This paper outlines an audit approach used to evaluate ADP support of the user population and covers subjects such as (i) measurement of user satisfaction, (ii) data center responsiveness to user information requirements, and (iii) distribution methods for computer based information, i.e., on-line availability vs. hardcopy distribution. The paper's intent is to show that audit disciplines can disclose user needs. Specific review methods, sampling criteria and actual audit results are presented supporting this premise. Through user oriented reviews non data processing professionals can be effective ADP auditors.

1. Introduction

One of the questions often facing the Defense Contract Audit Agency auditor is how to quickly and comprehensively determine the efficiency and effectiveness of the ADP organization in light of the facts that (i) many DCAA auditors have been professionally educated as accountants rather than data processors, and (ii) a large majority of audit assignments are of short duration without substantial advance knowledge of the particular ADP organization being reviewed. It is the premise of this paper that audit disciplines applied in a user-oriented audit approach is one good method for evaluating the strenghts and weaknesses of an ADP organization.

A definition of data processing performance was recently presented by a top authority in the ADP field as the "ability to consistently meet user requirements." In this context user-oriented reviews certainly merit consideration. It may be helpful to visualize user-oriented reviews as shown below where audit emphasis is on the solid lines.



In essence a user oriented review provides the opportunity to develop an audit opinion as to the efficiency and effectiveness of the data processing organization without the need for specialized knowledge to evaluate the data center itself. Furthermore, there are increased potential savings from such a review as audit coverage is not limited to the data center. DCAA's experience to date indicates that identified savings for these reviews are primarily in the user population where ADP efficiencies result in labor savings rather than equipment savings. Therefore, a two pronged audit approach, within the data center and outside the data center, is being advocated. Within the data center are such areas as equipment acquisition, performance measurement and other installation performance criteria. Outside the data center, the subject of this paper, the emphasis is placed on the end user and as a total will be referred to as Data Processing Information Utilization. From an audit viewpoint we have divided Data Processing Information Utilization into the three following areas:

a. Report Utilization

b. Automation of Manual Tasks

c. On-line Information Requirements

Separating the audit into the three areas shown will measure (i) user satisfaction with existing system output (Report Utilization), (ii) user satisfaction with the systems development process (Automation of Manual tasks), and (iii) the manner in which ADP information is being distributed to users (On-line Information Requirements). While the above titles and approaches are intentionally simple, later examples will show that often simple approaches result in complex results.

2. Report Utilization

A survey of ADP report utilization is usually the quickest method to identify inadequate support provided to ADP users. The following steps are useful in determining whether improvements are desirable.

2.1 Sample ADP Reports

In most instances there will be a summary catalogue of available reports that can be used as a basis for sampling. We have found that interviews are more desirable than questionnaires because a user who participated in system development will be reluctant to provide negative comments. However, during an interview it will be obvious if the user cannot interrelate the report with the job function. Suggested interview topics include:

2.1.1 Job Description

This may seem an unnecessary step but is in fact one of the most important. A key audit objective is to determine if data processing reports efficiently and effectively contribute to job accomplishment. What are the day to day information requirements for the user to be effective in the job?

2.1.2 An Explanation of the Report and its Purpose

If a user cannot effectively explain the purpose and use of the data then its value is questionable. "How is this data used? Can you walk me through one of the line items and explain how it assists you in your job?" If a report isn't useful, for whatever reason, chances are that needed information is provided by another system. Where there are parallel information systems, significant savings can usually be attained by eliminating the inefficient or unnecessary system.

2.1.3 Frequency of Reports

Information value is sometimes overrated. Reports received daily may only be required weekly or monthly, thus freeing valuable computer resources for other priorities.

2.1.4 Report Distribution

Often a real need for data is identified but its distribution exceeds its useful range. If a report is processed, copied, and distributed to several users and required by only a few then avoidable costs are associated with this unnecessary distribution.

2.1.5 Input Requirements

Substantial savings can be realized by eliminating or reducing input or verification requirements. For sampled reports, user input should be defined and logically evaluated. Unnecessary input results in inefficient labor and equipment utilization.

2.1.6 Report Output

User information needs can often be satisfied using less expensive methods such as microfiche.

2.1.7 Report Accuracy

Accuracy of data is critical for objective user decision making. In addition, accuracy is an integral part of data security.

2.2 Evaluate Internal Reviews

Most organizations require periodic internal ADP evaluation. In several instances we found that questionnaires are distributed to users to determine the quality of services provided. Effectiveness of such reviews will affect the scope of audit work.

2.3 Determine ADP Service Request Procedures

It is essential that the users have methods providing feedback to the data centers. Most authoritative sources recommend there be a centralized location where user problems can be reported and documented (user service desk). This source often contains a wealth of information and will identify consistent problems. Is this data summarized and monitored to provide information for corrective actions and have corrective actions been taken?

2.4 Summarization of Results

Presentation of results can often be clarified and enhanced by a summary matrix. In addition, it will provide the auditors with a better understanding of whether problems are isolated or consistently experienced. In the following example, organizational areas with ADP problems are annotated and would be referenced to supporting documentation. Consistent occurrence would indicate overall ADP system problems rather than isolated instances. For example, the lack of timeliness is adversely affecting about half the users and would indicate a need for detailed evaluation of the cause of this condition.

MATRIX SCHEDULE OF SUMMARY CONDITIONS
ADP REPORT UTILIZATION

Organization	Timeliness	Accuracy	Usefulness	Availability	Frequency	Input	Documentation	Etc.
Engineering	X		X		X			
Material Branch								
Prod. Control		X						
Mat. Control	X		X		X		X	
Receiving	X			X	X			
Quality Assurance			X					
Manufacturing								
Machinery				X				
Welding	X				X			
Assembly								
Etc.								

2.5 Examples of Conditions Identified During Initial Reviews

2.5.1 Unnecessary Input Requirements

ADP input was required from all trades on all assemblies regardless of involvement. Review disclosed that a large portion of the time certain crafts did not have work in a particular assembly. Modifying ADP input requirements to exclude input where no work was scheduled resulted in reduced ADP input, verification and file maintenance.

2.5.2 Report Distribution

Sample results showed that only six percent of the ADP reports were unnecessary. However, eighteen percent of all distributed copies were unnecessary.

2.5.3 Parallel Information Systems

Review disclosed similar and redundant production performance management and labor variance systems. Modified systems resulted in greatly reduced ADP input, verification and reporting while enhancing available summary data.

2.5.4 Report Usefulness

Computer based planning data were not current and complete. As a result, a parallel manual system was necessary to furnish this data. Modifying report structure and improving timeliness eliminated the duplicate system.

2.5.5 Input Requirements

Performance reporting was required on an exception basis. Analysis showed that over fifty percent of reported items were unchanged from prior periods. Excluding unchanged items eliminated half of the input requirements.

2.5.6 ADP Report Availability

Quality Assurance summary data was automated and distributed to QA personnel but not to production personnel. Distributing this data to production personnel enabled the contractor to eliminate the manual record keeping necessary to log quality assurance schedules and completions.

2.5.7 Report Timeliness

Production performance system reports, scheduled for distribution by Monday P.M., were in fact not available until Thursday. As a result, various crafts had to manually summarize data for Tuesday/Wednesday producton meetings.

2.5.8 Contractor Reviews

On a sample basis, an internal ADP report utilization questionnaire was distributed to ADP users. Sample results showed that over fifty percent of the users requested data additions, deletions, cancellations, and other changes. These modifications were subsequently performed. However, no further analysis was scheduled. Based on audit recommendations, the contractor scheduled a comprehensive evaluation which resulted in substantial system improvements.

2.5.9 Report Frequency

Data processing support was provided at a functional level by designated computer interface personnel. Evaluation showed that they were doing a good job of providing necessary ADP support. However, our sample of the end users disclosed that the weekly distribution was too frequent and that users unanimously agreed that biweekly and monthly distribution would be just as effective while reducing output by half.

2.5.10 Report Output

Most output distribution was hardcopy. Sample results disclosed that over 24 percent of the user population preferred microfiche due to lower cost, ease of use and permanent storage advantages.

3. Automating Manual Tasks

In the rush to develop sophisticated data systems, ongoing requirements for file maintenance, data gathering and analysis can be neglected. A survey or sample of administrative responsibilities will aid disclosure of potential adverse conditions.

3.1 Define Work Areas or Job Functions Having Extensive Administrative Requirements

Administrative responsibilities are often performed by non-administrative personnel from other organization (i.e., manufacturing). Therefore a survey of work areas using observations and/or interviews will disclose total administrative effort. As an example, at one location a survey showed that 79 production personnel were, in fact, performing material related responsibilities.

3.2 Where Significant, Comprehensively Evaluate Administrative Practices

This is usually done in the form of a structured interview. Pertinent data includes (i) ADP reports presently received, (ii) primary record keeping-file maintenance responsibilities, (iii) desired information presently not received, (iv) definition of staff assignments and time required, and (v) suggested changes or improvements that would improve job performance.

3.3 Examples of Conditions Identified

3.3.1 Test Data Control

As part of the QA process, over 30 employees maintained compartment test data. These data were summarized and updated three times monthly; over half the employee time was spent performing clerical functions. Automating test data and placing them on-line enabled the contractor to eliminate those clerical tasks.

3.3.2 Material Requirements

Electrode requirements had historically been determined by applying a given percentage factor to steel requirements. This method often resulted in surpluses or shortages. Stock status was accomplished by periodic physical inventories and manual record keeping. Status data was usually outdated and resulted in either (i) uneconomical local purchases, or (ii) less efficient welding methods with other electrodes. Automating material requirements responsibilities and developing better statusing methods enabled the organization to substantially reduce surplus and shortage conditions.

3.3.3 Personnel Qualifications

A good deal of manual effort was devoted to maintaining welder qualification data. Automating such data eliminated most manual record keeping and file maintenance.

3.3.4 Engineering Status

Complex engineering personnel statusing was necessary to support various program manpower requirements. Review showed that up to forty similar systems were maintained manually throughout the organization. In addition, time lags in distributing data from subtiers to higher organizational levels resulted in extensive reconciliation effort. Automating engineering status activities eliminated most manual preparation, copying and distributing while improving the timeliness and accuracy of necessary data.

4. On-line Information Requirements

A key decision in todays ADP environment is the selection of best means to distribute data processing information. Many organizations presently make use of on-line ADP information storage, retrieval and system updating. We have found this approach improves user support and satisfaction. While there is no specific formula for determining when an ADP system can best be geared for on-line output as opposed to hardcopy, an objective appraisal of existing, planned and potential systems will often disclose how the user can best be supported. It is after such an appraisal that technical considerations, such as the correct system network, will be most effective. Information developed by the auditor can then be provided to systems analysts and other responsible personnel and be included as part of the overall systems development decision making process. The auditor's objective is to be assured that all the users needs are being considered and are fully visible to the ADP organization. The following steps will assist in satisfying these objectives.

4.1 Determine Existing ADP Configuration and On-line Capability

Defining existing systems as well as equipment limitations will enable the auditor to limit the scope of review to realizable on-line configurations.

4.2 Determine Current and Planned On-Line Applications

Evaluating the current short and long range plans will enable the auditor to determine whether potential on-line applications are being effectively considered as part of the systems development process.

4.3 Evaluate Existing On-Line Applications

If these are current systems on-line a survey of these systems should include:

a. Criteria used to place systems on-line

b. Qualitative and quantitative studies

c. Periodic system evaluations

d. Planned changes

e. User participation and satisfaction

   (i) Frequency of use

   (ii) Suggested modifications or changes

   (iii) Recommended additions

4.4 Survey of Sample Systems that are Not On-line

Our experience to-date indicates that after appropriate research, the primary potential for payback lies with identifying existing batch systems which can be more economically placed on-line. We have found two criteria which most often disclose whether a system should be placed on-line.

4.4.1 Size of System Data Base and Occurrence of Information Retrieval

Large data bases having high volumes of data requests point towards the need for on-line information retrieval. Reviews to date have disclosed difficulties in maintaining the accuracy and timeliness of large dynamic batch systems. If information developed during the report utilization review indicates general user dissatisfaction with completeness and timeliness of data, placing the system on-line may eliminate those conditions.

4.4.2 Distribution and Documentation of System Information

Batch information, initially developed for limited applications, and subsequently expanded and distributed to a wide variety of end users, is susceptible of being placed on-line. If a system is unwieldy, data must often be supplemented by manual record keeping, updating and distribution. In addition, if there are few documentation requirements, and informational needs are for statusing and visibility (such as material and scheduling information), on-line applications can result in substantial administrative labor savings. Savings generally include information filing and maintenance, manual recording of data changes between distributions and information distribution costs.

4.5 Example of Conditions Identified

4.5.1 Material Inventory Status

Review disclosed instances where material inventory data was logged and maintained up to thirty times to provide necessary material status visibility. Lack of timeliness of the current batch system was for the most part responsible. Placing the material inventory system online and interactively updating the data base precluded most manual statusing and saved significant manual labor.

4.5.2 Industrial Relations Status

Substantial ADP hardcopy data was being distributed to Industrial Relations representatives throughout several divisions and programs. Review showed that about 85 percent of all data was received, analyzed and used at eight locations. Placing the data on-line through the use of CRTs enabled the organization to eliminate most hardcopy distribution while saving substantial time previously expended by organizations in information exchange.

5. Summary

This papers intent was not to advocate that internal reviews of data centers have been misdirected. On the contrary they are vitally necessary. But from an audit viewpoint, the paper does advocate that if one of the primary purposes of the audit is to measure whether user needs are being met, a user oriented approach is worthwhile. Further, this approach does not require years of data processing experience. In summary, a two step audit or review approach may be desirable, including (i) a user oriented review measuring user satisfaction with existing systems, and (ii) a data center review which reflects the results of the user satisfaction review.

AUDITING AN IMS SYSTEM

Capt. C. L. Gausche and W. J. Schwinden

Boeing Computer Services
P.O. Box 24346
Seattle, WA 98124

This paper provides a structured approach to data collection, analysis, and reporting when conducting a performance audit of an IBM Information Management System (IMS). Performance auditing, analogous to the traditional definition of an operational audit, is concerned with: verifying planned results of a change, developing a current baseline, and reporting this data in management usable terms. The approach of the paper is to provide the performance auditor with information concerning: a performance auditing procedure, various data collection tools, applications of performance analysis techniques, and how gathered data and subsequent analysis can be structured into a logical flow. The performance audit steps are amplified by using a case study. A reporting format, used extensively by Boeing Computer Services (BCS) Company, is also included.

Key words: Auditing; capacity planning; computer performance evaluation; Information Management System (IMS); management reporting; performance analysis; structured approach.

1. Introduction

In the operation of a data center there are a number of questions that are of prime interest; how efficiently are the system resources being used? how close to capacity are these resources being operated? how effective is the data security? are the computer center costs adequately covered and equitably distributed? etc.

The purpose of a performance audit is to address the first two questions: resource usage efficiency and operating capacity levels. Unlike a traditional audit, which is concerned with fiscal accountability, the computer performance audit is concerned with resource accountability. That is, how and to what extent are the various computer resources (CPU, memory, I/O devices, etc.) being used? As a vehicle for presenting the concepts and approaches, an audit of an IBM 370/168 processing an IMS workload is used.

The techniques and approaches which are used are not limited to IMS systems or even to on-line systems. These techniques have been successfully applied to systems running batch only, batch with TSO, and combined batch, TSO and IMS.

The objectives of this paper are to present a structured approach for auditing a computer system and to provide a specific application of the concepts and approaches. The paper is organized in the following manner:

1. Introduction and overview

2. General discussion of computer performance auditing

3. Data sources and data reduction techniques

4. Procedure for conducting an audit

5. Case study

6. Summary, conclusions, and areas for development.

2. Concepts of Computer Audits

There are several approaches to the auditing of computer systems. For the purposes of this paper, auditing is defined as the process of reviewing, verifying, capacitizing, recommending, and reporting. That is, it is the process of reviewing the current state of the system, verifying the results of any system changes, determining system capacity levels, providing required recommendations, and preparing the necessary reports.

The review process is the prime function of the audit. In this process the usage of the system and all of its components are determined. In the verification phase the quantification of any changes to the system, such as additional hardware or a new release of software, are made. Capacitizing is the process of determining the capacity limits for each of the system components and the per cent of capacity at which each component is operating during a given time frame. Specification of capacity limits is usually obtained from other sources; special studies, vendor data, published papers, etc. The operating capacity levels are obtained directly from collected data. Recommendations are based on the results of the previous processes. The primary emphasis is to provide management with the basis for effective decision making. After these phases are completed, a report to management presenting the findings and recommendations is necessary. In traditional terminology, this complete process is an operational audit.

The remainder of the paper discusses the audit procedure in detail and provides an illustrative case study.

3. Sources of Data and Data Analysis

Auditing an IMS system requires the use of various data collection techniques. The basic tools are software monitors, hardware monitors, and accounting data. Software monitors are used for obtaining detailed data on the internal operations of the system. First are IMS statistical summary reports, such as Boole and Babbage's Control/IMS. These reports provide information on transactions/hour, transaction existence time, transaction mix, Message Processing Region (MPR) usage, and data base access (DL/1 calls) activities. (For the non-IMS oriented, an MPR is equivalent to a batch job stream processor. Messages are processed serially by an MPR, but a system can have a number of concurrently operating MPRs.) A second source of information is the Resource Measurement Facility (RMF). The types of information gained from this report are paging rates, device and channel busy percentages, IMS Control Region Activity, and queueing activity. This information complements hardware monitor data or, in cases where hardware monitors are not available, can be used alone. The third type of software monitor is the IMS DC monitor. Information from this monitor is used to explore various I/O relationships between IMS and Multiple Virtual Storage (MVS). Whereas other monitors can be run continously, this monitor is run for short time periods due to high overhead.

Hardware monitor data, although not always available, is very useful because it can monitor the system continuously without inducing its own load. In addition, it can continue running despite any system failures. Hardware monitor data is useful for showing CPU and channel busy over a representative week and finding peak hour in terms of CPU busy. This data is also used to validate various software monitor reports.

Accounting data is useful in determining shift loading profiles. The profile is expressed in terms of system components (CPU, paging, I/O activity) or in terms of a work unit, such as BCS's Computer Resource Unit (CRU). A data reduction program, such as BCS's SARA, greatly simplifies the activity. The shift profile identifies the weekly loading cycle and the workload mix for each shift.

Data analysis utilizes three tools: statistical analysis, system modeling, and graphic representations. The statistical analysis consists of mean and standard deviations for the system variables, correlation analysis between these variables, and, where applicable, curve fitting techniques. System modeling techniques frequently involve queueing network models. In some cases a single server queueing analysis may be sufficient; however, as the perception of the system becomes more detailed, more sophisticated models may be required. Although graphic representation of data is generally thought of as a reporting tool, it is very useful as an analytical tool. The graphic representation of shift loading is extremely helpful in identifying peak periods and trends.

4. Generalized Procedure

This section presents a structured approach to conducting a performance audit. A structured approach assists the analyst in separating the mechanical aspects of the audit from the analysis. By identifying the mechnical aspects, the audit can be conducted in an efficient manner and development of tools to automate these functions is possible.

This structured approach separates the auditing process into seven steps:

(The first three steps are the review and verifying processes presented in Section 2.)

1. Define the hardware and software configuration and operating environment.

2. Time-series analysis.

3. Cross-section analysis.

(The next two steps are the capacitizing process of an audit.)

4. Statistical analysis.

5. **System modeling.**

(The last processes are recommenaing and reporting.)

6. Recommendation formulation.

7. Report publication.

The first step in the audit is understanding the system configuration, hardware, **software, and operating environment**. Hardware configuration consists of the CPU, memory size, and quantity, capacity, and interconnection of I/O devices. Software configuration consists of the version of the operating system, the version of IMS and any unique patches or updates. Operating environment identifies the reliability, availability and service level requirements. Reliability requirements specify how sensitive the operation is to system outages. Availability requirements define the hours of operation, maintenance and service needs, and any other factors that could influence the available system time. Service level requirements specify the level of service expected by the user. Service levels are determined by the workload mix: jobs/hour for batch, response time for on-line applications, etc.

After the analyst has described the complete system, the second step of the audit begins. This step is concerned with looking at the system over a period of time. In an analytical sense, the independent variable is time and the dependent variables are performance service levels.

For IMS, two measures are tracked and plotted on a weekly and monthly basis; response time and number of transactions per hour. Both of these data elements are available from IMS Statistical reports. Changes in configuration and capacity are superimposed on these charts so a complete picture of the long term workload is available. After several audits a history of other performance indicators can also be established. A more detailed breakdown of these items is contained in the case study.

The next step in the time-series analysis is the narrowing of the perspective from a long-term time-series approach to a general performance interval. This interval is a time period such as month-end closing for a company accounting system, seasonal peak demand periods and so forth. For the IMS application a representative week was selected. The interval is chosen by looking at three sources of data; either the hardware or software monitors, the IMS statistical summary report, and accounting data for the months under study. Hardware or software monitors supply the data for the studied interval, i.e., CPU and channel utilization. The IMS monitor provides transaction load and response time statistics. Once a general interval is identified, a shift profile is developed showing the most used periods and the composition of the workload.

Next in the sequence is the narrowing of the perspective to a specific performance interval. A specific performance interval is also application dependent. It could be average or peak day in the general interval, prime or time critical shift for an on-line operation, etc. For IMS a representative day during the general interval was choosen. A good method of showing usage for a representative day is by MPR. Each MPR is tracked by hour of the day and graphed. Along with a graph by MPR during the day, a graph of transaction type is generated. Transaction type data are contained in the IMS statistical summary report and each type has its associated statistics. From this data the following can be determined:

1. The most frequent transaction types in terms of per cent of total transactions. (Generally, 20% of the transactions will account for 80% of the work.)

2. For these transactions, the resource utilizations in terms of MPR and DL1 CPU utilization, DL/1 calls, response time, and the extent of MPR activity.

From the specific performance interval a peak demand period is defined. This period, generally a peak hour or shift depending upon the application, is used to determine the resources necessary to meet the service level requirements. The peak demand interval is used for the cross-section analysis.

Step 3 in the audit process takes a cross-section view of the data. Time-series analysis can be thought of as a horizontal view of the data. Cross-section data is then a vertical cut with the time variable held fixed. Information obtained from this analysis includes resource usage, service times and workload data.

Resource usage is the demand on CPU, memory, and I/O devices during the peak hour. CPU utilization during this period can be obtained with either a hardware monitor or a software monitor. Memory usage, however, generally requires a software monitor such as RMF. This analysis focuses on paging activity since paging reflects memory utilization. I/O activity is broken into two catagories: channel activity and device activity. Channel activity is obtained from either hardware or software monitors. Device utilization is usually obtained from the software monitors since device queueing activity is also available.

Service times, in an IMS environment, are a measure of MPR efficiency. The overall service time, for all MPRs combined, is computed by multiplying MPR usage by the sample interval time, usually 3600 seconds, and then dividing by the total number of transactions during the hour. The equation is:

$$t_s = \frac{\text{MPR usage X interval seconds}}{\text{total transactions}}$$

Workload during peak hour is measured by the number of transactions processed. Transaction mix is analyzed by looking at the following set of transaction statistics: IMS CPU usage, DL/1 CPU usage, DL/1 calls, memory paging, IWAITS, and EXCPS.

After the time dependent analysis has been completed, the next step is to perform the necessary statistical analysis. Up to this point, basic statistical analysis (mean, standard deviation, regressions, etc.) have been used to determine the representative week and day. In this step, correlation analysis and curve fitting techniques are used. The process here is to determine the relationship between service level indicators and system resource usage. The concern is whether service level indicators (response time, etc.) are an adequate measure of resource usage. It is quite possible that a service level indicator is not sensitive to a critical resource. This effect is common in a mixed workload environment, such as batch and IMS. For example, a particular disk drive may be saturated and have excessive queueing, but IMS response time may be well within specifications. The second concern is to identify the critical system resource. The critical system resource is that resource which would constrain the system if the workload were to increase to capacity. This analysis leads into the next step, system model application.

The fifth step in the audit procedure is the application of a system model. In this step there is a definite trade-off between the amount of time expended in applying a model and the degree of precision required. In general, detail precision is not a requirement. The fluctuations of the workload from day-to-day and the difficulty in predicting future workloads soon negate even the most detailed of systems models. Experience with IMS shows that a single server queueing model is sufficiently precise for auditing even though IMS is a complicated multi-process, multiple server queueing system.

The approach was to take the service time, as calculated in Step 3, to generate an expected value for response time. The equation is:

$$t_r = \frac{t_s}{(1-\rho)}$$

where ρ = (MPR usage)/No. MPR's

The equation was then scaled so that its graph would indicate the related transaction rate. A specific example is contained in the case study.

Several recommendations generally result from the audit procedure. These recommendations should be structured so management can make timely and cost effective decisions. Thus, the sixth step in the audit process is frequently the most complicated to analyze, the most intricate to quantify, and the most difficult to costing. Whereas, in the capacitizing step a simple queueing model may suffice, a detailed complicated model may be required in order to fully analyze the alternatives. Alternatives will range from managerial actions, to changes in hardware, software or operations, to acquisition of new equipment. The initial problem is to determine the right balance between the depth of analysis required and the timeliness of the audit. To answer this question is usually an iterative process between auditors, analysts, and management decision makers. Possible alternatives must be discussed, reviewed, and filtered before a viable set is selected for detailed analysis. The end-product of this step should be a set of alternatives, the cost and benefits of each alternative and a specific recommendation.

The last step in the audit process is to prepare a formal report. As with any documentation effort, it is disastrous to leave this effort until last. The report should be written as the audit progresses. Emphasis should be placed on the report since it is the only effective and long-lasting means for communicating the audit results.

There are various formats for the audit report. The format developed is both workable for the analyst and acceptable by management. The outline for this format is:

Part 1 - Introduction
Part 2 - Summary of Findings and Recommendations
Part 3 - System Description
Part 4 - Workload Profile
Part 5 - Detailed Analysis of Findings

The Introduction provides such data as reasons for and scope of the audit. The Summary is an executive level review of the report. Trend charts and other graphic representations are used extensively in this section. In the last part of the Summary the recommendations are presented to management. The System Description provides information on the environment by describing the hardware configuration and software revision levels. The Workload Profile is partitioned into time-series and cross-section data. The time-series analysis is used to identify trends and to select peak periods of operation. The cross-section analysis is used to quantify resource usage by analyzing peak period system operation. The Detail Analysis section provides support data for the summarized findings and recommendations. It shows resource use and discusses any unique areas considered important by management or the auditor.

Figure 1. is a flowchart summarizing the audit procedure. The five functions of an audit are identified in the left-hand column and the seven steps of the procedure are indicated in the center column. As with any procedure, strict application may not be desired. However, the general application insures an acceptable end product. The structure increases the effectiveness of the analyst and defines areas that can be automated. For example, data reduction necessary for selecting representative weeks and days, construction of shift profiles, statistical analysis and modeling are all candidates for automation.

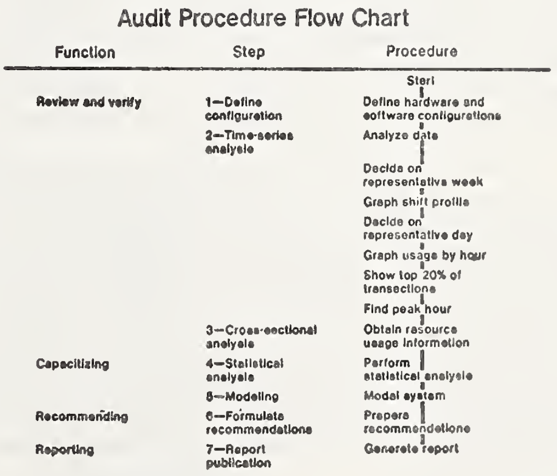


Figure 1. Audit Procedure Flowchart

The next section is a case study that presents a specific application of computer performance audit concepts and techniques.

5. Case Study

In September of 1978, a 7th megabyte of storage was added to a 370/168 system operating IMS. The IMS system was used in a large aircraft manufacturing environment for production planning. One month after the memory installation, an audit was conducted to determine the impact on the transaction processing capability in terms of transaction rate and response time. The following case study is presented in sequence of development rather than as presented in the final audit report.

The first step in the audit defined the hardware and software configurations. Figure 2. is the diagram of the hardware configuration at the time of the audit. The software configuration consisted of:

MVS operation system, Release 3.7
JES2
MS Release 1.1.4 at PTF Level 3

The next step, time-series analysis, started by reviewing the operation of the system to-date. Figure 3. is the historical performance graph at the time of the audit. The performance service level is defined as a 15 second response time requirement for 80% of all transactions. The 80% requirement eliminates the occasional long transaction that tends to skew the total average statistic. This chart shows that the system was operating at capacity, 5700 transactions/hour, prior to the addition of the memory.

The third step was to select a specific week and then develop a shift profile. For this application, the third week in October was selected. Figure 4. is a graph of the shift profile. Although the evening shift is the busiest in terms of total CRUs, the prime shift is the most critical because of the stringent response time requirements. (Evening shift processes Batch MPRs (BMPs), which update the same data bases as the on-line transactions, but without the response time requirement.)

From the representative week a representative day was selected. Figure 5. presents the total transactions by hour and by MPR for the entire prime shift of the representative day. The peak hour, 10 to 11 a.m., is easily identified. Although the peak hour is in the morning, the peak sustained load occurs in the afternoon.

The first step in the cross-section analysis was to perform a transaction analysis. Figure 6. is a graphic representation of the analysis. These 15 transaction types account for 76% of the total transactions and 83% of the data base accesses (DL/1 calls).

The results of the cross-section analysis of the performance statistics are presented in Figure 7. For comparison purposes, performance statistics for six periods are included.

This completes the review and verify function. Next is the capacitizing function with the first step being statistical analysis.

Statistical analysis began with the data contained in Figure 7. The important statistics are 80% response time values, pages (SMF), peak hour CPU, service time, and pages/second (RMF). The 80% response time value identifies 'out-of-specification' periods. The EXCPs/transaction and peak hour CPU show that neither of these resource components were constrainted. The service time values closely track the 80% response time. The system constraint is identified by comparing the pages (SMF) with either service time or 80% response time. A close look at pages/second (RMF) shows that system paging reaches its capacity at about 70 pages/second. Since paging was running at 65 pages/second in October, some capacity still remains. The next statistical analysis technique was a correlation analysis of system variable relationships. Figure 8. is a table of the correlation analysis results. As expected, the relationship between variables of interest was very high.

The system modeling step immediately followed the statistical analysis. Figure 9. is a graph of the system models for the previous audit and this audit. The equations for a single server queueing model are given on the graph. The equations were developed using the transaction service time, as given in the table in Figure 7. The equation is then scaled and plotted, as in Figure 9. The data points on the plots were obtained directly from the daily IMS performance report. These values are the 80% response times for the average hour during prime shift. Once the model is plotted, the maximum capacity is determined by graphing where, in terms of transactions/hour, the 15 second limit is reached.

The last step was to update the historical performance graph with the new capacity level. Figure 10. is this graph with the capacity level, after the 7th

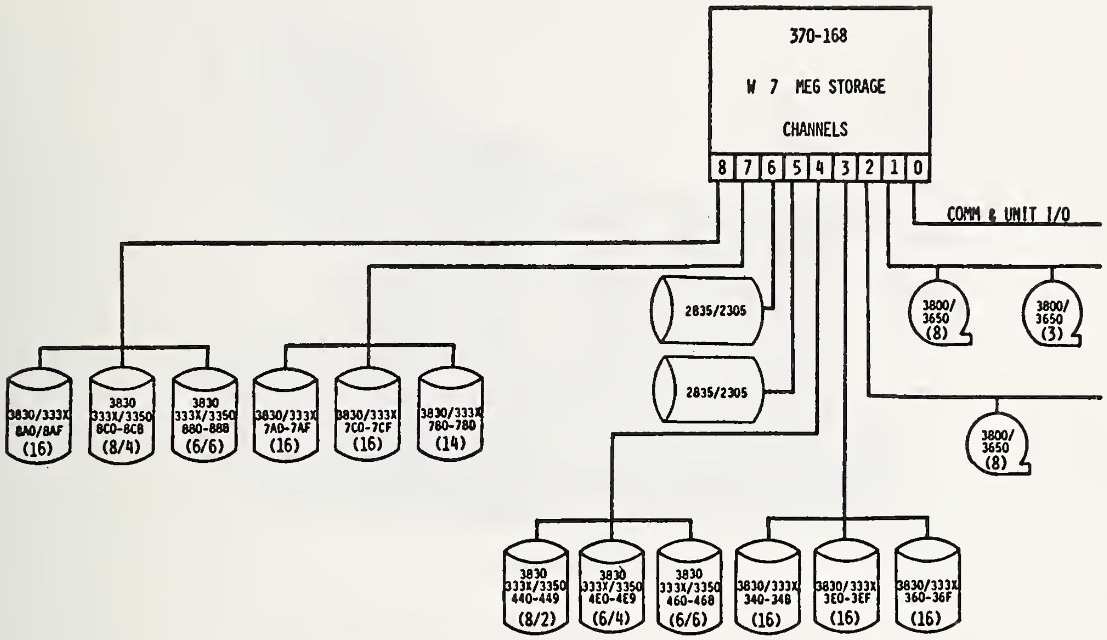


Figure 2. Hardware Configuration

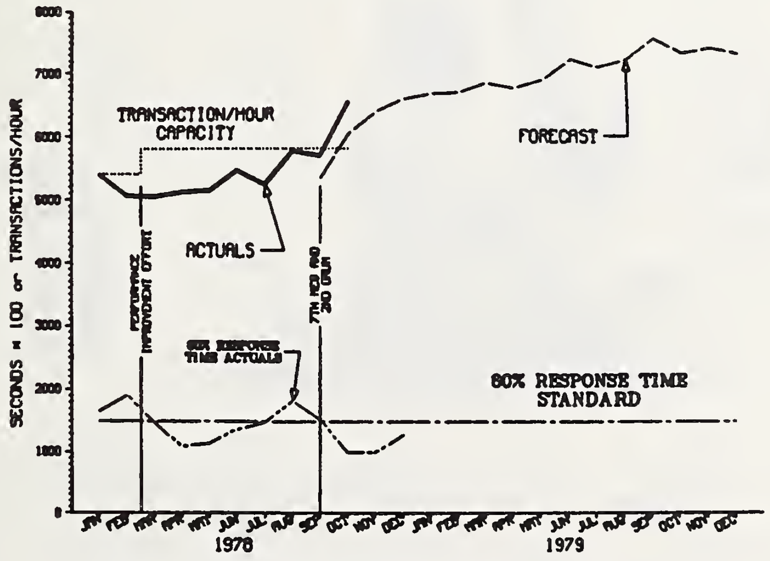


Figure 3. IMS Historical Performance Chart
Before Audit

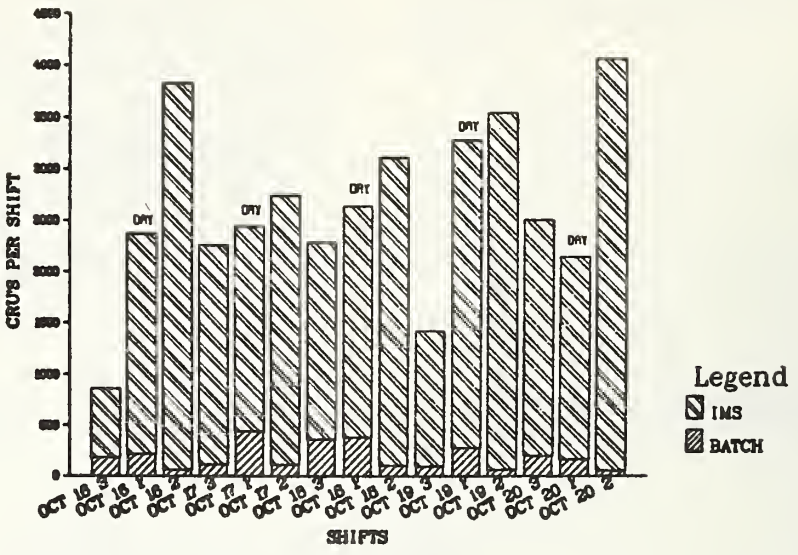


Figure 4. IMS Production Shift Loading Profile

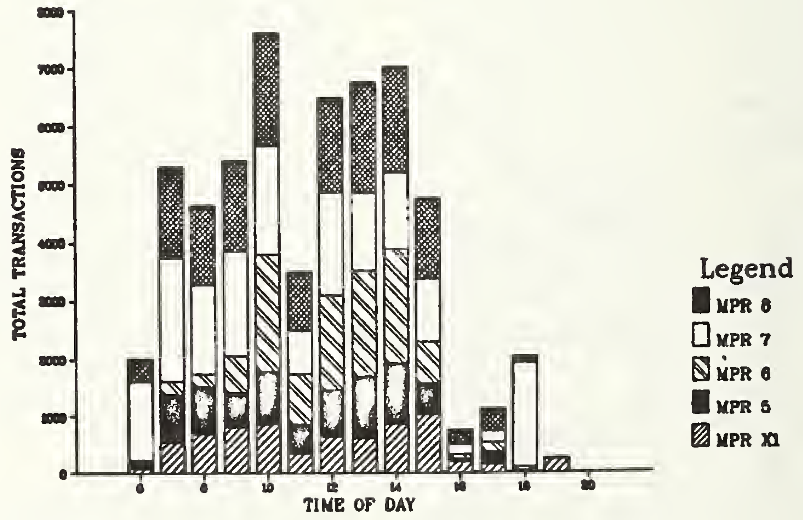


Figure 5. IMS Transactions By MPR By Hour

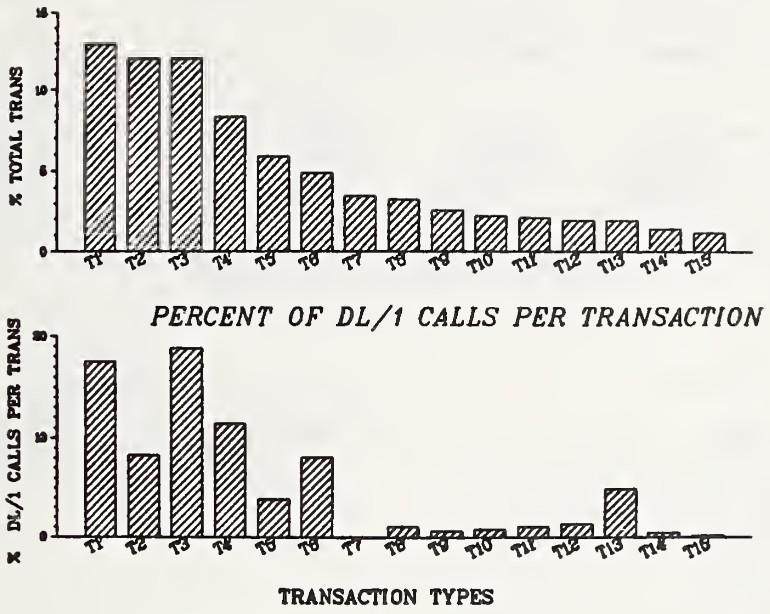


Figure 6. Percent of Total Transactions

	IMS 1.0.1 Jan, 77	IMS 1.1.3 Jun, 77	IMS 1.1.4 Dec, 77	IMS 1.1.4 Feb, 78	IMS 1.1.4 Apr, 78	IMS 1.1.4 Oct, 78
Transactions						
Prime Shift	31,383	38,091	48,993	44,252	49,279	51,148
Peak Hour	4,400	4,460	5,444	4,917	5,778	7,575
Response Time						
Average	23.0 sec	55.0 sec	9.4 sec	48.5 sec	32.8 sec	32.8 sec
80%	---	6.0 sec	8.0 sec	20.0 sec	14.0 sec	9.0 sec
Per Transaction						
MPR CPU	---	---	47.4 ms	86.1 ms	84.3 ms	78.3 ms
DL/1 CPU	---	---	---	---	---	64.7 ms
DL/1 calls	33.8	32.8	34.7	24.2	22.6	24.7
Pages (SMF)	11.4	---	8.8	19.9	18.7	11.0
IWAITS	27.7	---	---	---	7.7	12.4
EXCPs	---	---	7.5	---	9.2	8.1
CRUs	---	.128	.130	.110	.108	.105
IWAITs/DL1 call	0.82	---	---	---	0.34	0.41
Peak Hour CPU	60.2%	---	75.1%	---	67.0%	80.5%
CPU sec/transaction	0.493	---	0.497	---	0.417	0.382
Total MPR Usage	2.65	1.51	2.15	3.02	2.67	3.66
Service Time	2.21 sec	1.33 sec	1.58 sec	2.21 sec	1.96 sec	1.79 sec
Pages/Sec (RMF)	---	---	---	---	71	65

Figure 7. IMS Performance Statistical Summary

	IMS CPU	IMS DL/1	IMS CRU	RMF CPU	RMF EXCP
IMS CPU	1.000	---	0.993	0.876	---
IMS DL/1	---	1.000	0.989	---	0.920
IMS CRU	0.993	0.989	1.000	---	---
RMF CPU	0.876	---	---	1.000	---
RMF EXCPs	---	0.920	---	---	1.000

Figure 8. IMS Correlation Matrix

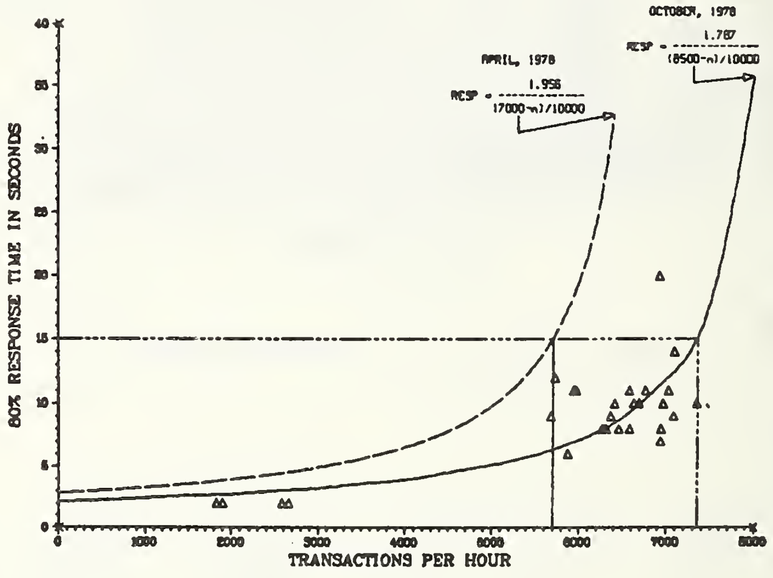


Figure 9. IMS Transactions VRS 80% Response Time

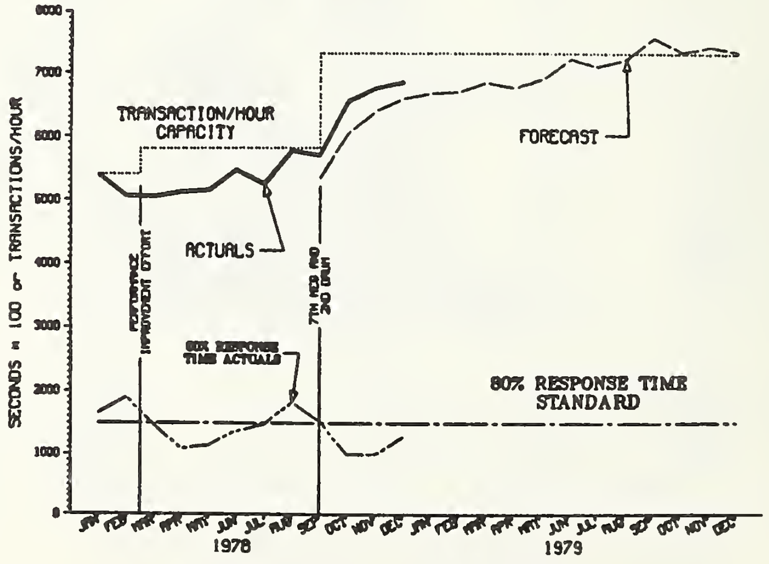


Figure 10. IMS Historical Performance Chart After Audit

megabyte of memory, at 7300 transactions/hour.

Normally, this audit would contain a recommendation to add more real memory to the system. However, it was not made because a system upgrade to a 370/3033 was planned for the first quarter of 1979. The upgrade was necessary to meet requirements for new applications. Thus, the primary benefits of this audit were to provide a better understanding of the system and to establish a baseline for determining improvements achieved with the new system.

A final report was published with Figure 10, Historical Performance graph; Figure 8, Statistical Summary Table, and Figure 9, System Model graph all included in the Executive Summary.

6. Conclusion

This paper has presented an overview of computer auditing, a procedure for auditing an IBM computer using IMS, and a case study of one unique IMS system. It is important to note that while judgment is a very important ingredient in auditing, a standardized procedure provides the analyst with a structured environment. This environment encourages the analyst to spend more time analyzing and making reliable recommendations and comparatively less time searching for an approach.

The point should be made that this is not a textbook approach to auditing, but rather an approach that has evolved from actual experience. This approach has been used for auditing two dissimilar IMS systems, a VM/370 running CMS system, a special purpose on-line manufacturing system which is run on dual 370/168's, and on a combined batch, TSO, and IMS system. Although the procedure has been used extensively, it is not, and probably never will be, finalized. It will continue to evolve as the systems change and the analysis tools become more sophisticated.

The immediate needs are for automating data reduction and for more system model development. The automation effort concentrates on more rapid breakdown of RMF data into graphic displays. The modeling efforts fall into two catagories: better statistical models and more accurate queueing network models. Statistical analysis improvements concentrate on utilizing cluster analysis and developing appropriate curve-fitting techniques. A prime concern in queueing network modeling is determining the trade-offs between stochastic and operational analysis models.

Finally, although additional information could enhance the audit, it is functionally complete. The greatest benefit of the basic procedure is its structure - structure in data collection, structure in data analysis, and structure in data reporting.

References

[1] Boyse, John W. and Warn, David R., "A Straightforward Model for Computer Performance Prediction", Computing Surveys, VII (June, 1975), pp. 73-93.

[2] Bronner, LeeRoy, Capacity Planning; An Introduction, IBM Technical Publication, GGC22-9001, January, 1977.

[3] Ferrari, Domenico, Computer Systems Performance Evaluation, Prentice-Hall, New York, 1978.

[4] Ferrari, Domenico, "Workload Characterization and Selection in Computer Performance Measurement", Computer, July/August, 1972, pp. 18-24.

[5] Hughes, P.H., and Moe, G., "A Structured Approach to Computer Performance Analysis", National Computer Conference, 1973.

[6] Svobodva, Liba, Computer Performance Measurement and Evaluation Methods: Analysis and Applications, Elsevier North-Holland, New York, 1976.

Computer Systems Acquisition

COMPUTER SYSTEMS ACQUISITION

Dennis M. Gilbert

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

The growing diversity, complexity, and sophistication of ADP systems and the increasing costs of total systems have caused corresponding increases associated with acquiring these systems. Spurred, in part, by a series of Federal Regulations, a recent parallel trend has been the growing sophistication of the ADP System buyer. There is a new awareness of the importance of the fact that the user must state requirements so that they not only reflect actual needs but also are in a form that the vendor can understand and in a form such that vendor responses can be evaluated fairly.

To satisfy these needs, more vigorous tools, techniques, and approaches than were adequate a few short years ago are now necessary. Questions of how to evaluate interactive and distributed systems, how to establish fair procedures which encourage competition, and whether the evaluation process is cost effective must be looked at anew. In this session, we will examine a set of quantitative approaches that addresses these questions and the ADP acquisition process.

This session's first paper, "Applications and Limitations of the Cost-Value Technique for Competitive Computer Selection," by Barbour, Holcombe, Harris and Moncrief, describes the Cost-Value technique -- presented by Joslin in Computer Selection (1977) but still little known, little used, little understood -- and compares it with other methods for evaluating vendor ADP system proposals. The technique is demonstrated using exercises from a recent simulated DOD procurement and analysis project at the Navy Postgraduate School. While the technique was found to be superior to others being examined and worthy of consideration, the study nonetheless identified difficulties in implementing the methodology in a real market environment.

Significant costs and technological and administrative complexities are associated with the computer selection process. It would be of much practical help to the buyer to determine which information will contribute to the selection process and what costs and values are associated with that information. In the second paper of this session, "Computer Selection: To Measure or Not to Measure," Mamrak explores the applicability of statistical decision theory to aid the buyer in making such cost effectiveness decisions. The paper clarifies such concepts as "mandatory," "desirable," "measurable," and "non-measurable" as applied to performance evaluation criteria. The author shows that the economic question of whether to measure the proposed system or not is only applicable to desirable and measurable criteria; other categories either must be measured or cannot be measured.

In the third paper, Spooner complements his previous efforts with "Benchmarking Interactive Systems: Modeling the Application." He describes a modeling complex under development which permits the benchmarking of clusters of multiple minis proposed by multiple vendors. The complex consists of: synthetic code written in a higher order language representing the application, the proposed hardware and software, a vendor developed "Vendor Interface Package (VIP)," and a separate benchmark driver. A specific methodology for defining "Natural Flows" and resource consumption is presented. The author describes the flexibility of the technique to allow for varying levels of detail and varying vendor options. The author contends that the flexibility, while imperfect, is less unfair than the available alternatives.

Following the presentations, authors will participate in a panel discussion on computer system acquisition. Panelists will be asked to identify areas in which additional quantification tools could aid in the acquisition process. They will be asked to comment on the relative applicability of their approaches to Federal Government and non-Government environments. They will also be asked to address how their approaches facilitate or constrain the specification of user requirements and how the approaches contribute to a satisfactory user/ADP interface.

APPLICATIONS AND LIMITATIONS OF THE COST-VALUE TECHNIQUE FOR COMPETETIVE COMPUTER SELECTION

Richard E. Barbour,Jr. Craig S. Harris
James O. Holcombe Wm. C. Moncrief

Department of Computer Science
Naval Postgraduate School
Monterey, CA 93940

The cost-value technique, as proposed by E. O. Joslin in Computer Selection, was investigated in a simulated DOD system procurement exercise. The cost-value technique was found to be favorable over less formalized and rigorous approaches such as specification fulfillment, cost only, cost-effectiveness selection, etc. The cost-value method requires detailed research on the part of both the decision-maker and the end user into the needs and intended applications of the system prior to issuance of a Request for Proposals (RFP). They must analyze and define all aspects of system applications and segregate these into mandatory and desirable features, assigning a dollar value to each. With a more thorough definition of system requirements, the system proposals are driven more by the purchaser's needs than by "marketing assumptions" made to enhance the vendor's product. In order for any system to be considered for selection under this method, the system must meet all mandatory features. Qualifying systems are then judged on the basis of their cost-value relative standing in meeting the desirable features. This approach provides to all vendors the criteria for selection, thus establishing a fair and competitive basis for system proposals. Proposal relevancy is therefore encouraged with respect to organizational needs. The selection experiment showed that employing the cost-value technique in a real market environment was subject to several difficulties not common to other techniques. The problems encountered and the limitations of the technique will be discussed in the context of competitive computer procurement.

Key words: Computer evaluation; computer performance; computer procurement; computers; computer selection; cost-value; cost-value technique; EDP selection; evaluation methodology; performance evaluation; procurement.

1. Introduction

From an industry which was once dominated by one supplier (IBM) and one buyer (the Federal Government) the electronic data processing (EDP) market in America has become extremely competitive. In 1978, Datapro Research Corporation published information profiles of 925 companies that offered EDP products and services [1][1]. With such a varied range of equipment from which to choose, the data processing manager of today faces an incredible dilemma attempting to select a suite of hardware that best meets

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

the organization's processing needs. "The people who sell computers are specialists. Not only are they specialists in computing and data processing; they are specialists in selling and the presentation of their wares, and in the associated negotiations on commercial and contractual terms. In the very least they are specialists in the sense that selling is their job and they do it all the time.

The people who buy computers, even when they are themselves professionals in computing, are rarely equally specialized, or trained in buying" [2].

The suppliers' representatives are selling all the time, whereas a data processing activity may only acquire upgraded or additional hardware once every 8 - 10 years in the government sector [3]. "The unfortunate results of this unequal contest are only too easy to see. In countless computer installations it has come to be accepted that no configuration as proposed by the supplier will ever do the required work and that the customer will have practically no redress. The typical pattern of supplier and prospective customer exchanges leaves a great deal of the initiative with suppliers" [2].

Ultimately during the EDP selection process there comes a time when the data processing manager must evaluate the vendors' submitted proposals and select the one system in which the organization's capital is to be invested. Unless the manager has taken adequate measures to dominate the procurement, and to aggressively evaluate the proposed systems, the final selection decision may be a faulty one. The buyer must be able to evaluate the system exclusive of the marketing bias.

Traditionally there have been several evaluation methodologies employed in selecting one hardware item over another in a competitve procurement. The methods include sole source, overall impression of the vendor, cost only (low bidder), weighted scoring, cost-effectiveness ratio, and others. One technique which has not been widely published is the cost-value technique. This technique was employed in a simulated EDP procurement and analysis project and was found to greatly facilitate the buyer's control over the competitive procurement process. The purpose and scope of this paper is to describe the cost-value technique and to demonstrate its use, since there exists very litte literature on the subject and the potential of the technique with respect to EDP equipment selection is perceived to be significant.

2. Background

The cost-value technique was developed as a selection methodology which translates a set of specifications for a desired computer system into two fundamental categories: mandatory and desirable features. By definition mandatory features are those items that are essential to the implementation of the activity's needs and objectives. Desirable features are only those items which would make the completion of the activity's mission easier. Hence, a proposal will receive no consideration if it fails to meet any one of the mandatory requirements. The absence of some desirable feature from a vendor's proposal would invoke some penalty, however, it would continue to be considered in the process of selecting the most advantageous proposal. This enables a procuring activity to validate and objectively choose among several vendor proposals. The technique was introduced in this context by Joslin [4]. The cost-value technique is superior to other selection methodologies because it examines both mandatory and desirable features, establishes understandable relative values between these items and simultaneously incorporates system life costing into the decision process. Other commonly used evaluation techniques such as "cost-only" and "weighted-scoring" do not provide these facilities (without extensive and awkward modifications).

The cost-value technique combines the simplicity of the cost-only technique with the realism of the weighted-scoring technique. The result is a technique superior to both. It is superior to the cost-only technique in that it considers the relative merit of the elements of a proposed system in addition to viewing the overall cost of the system and its ability to meet the mandatory requirements. It is superior to the weighted-scoring technique in that it establishes meaningful relationships between the items of value and the system's cost while at the same time incorporating system life costing.

This technique and several others were employed as part of a simulated DoD procurement and analysis project at the Naval Postgraduate School. One purpose of the project was to evaluate alternative computing systems designed to satisfy specific requirements (described in Figure 1). Seven teams analyzed alternative computer systems, using different evaluation techniques in selecting their final computer system.

The teams' selections were evaluated primarily on the quality of system design, the ability of their selected system to meet

1. Main purpose - scientific processing (numeric floating point calculations). This processing would consist of an average of 40 jobs/day (standard deviation = 8), each job averaging 6 x 10**8 floating point operations (standard deviation = 1 x 10**8). These jobs would have small main memory requirement (average 30 Kbytes/job, standard deviation = 15 Kbytes) and I/O would be negligible.
2. Secondary purpose - data base handling. Data base management routines would be run on an average of four times per day (standard deviation = 1) to update four data bases which the system would maintain on line (1 update/data base/day). Each update requires creation of a backup copy on removable media. The average size of the data bases was 5 Mbytes (standard deviation = 2.5 Mbytes).
3. A normal supportive/development system was required, capable of time-shared operation with less than 8 terminals. Hardcopy output was required, and a graphics capability was highly desirable. An operating system was needed to support the described job mix along with providing a reasonable number of utilities, assemblers, and language processors.
4. The initial budget request is $125,000.

Figure 1. Full System Specifications

the stated system requirements, and the quality of the selection technique employed. The teams were competitive in their challenging of evaluation techniques and design features. Evaluation techniques used by other teams included sole source, cost-only, overall impression, and weighted scoring. At the conclusion of the experiment, a joint meeting of all teams was held to discuss and evaluate the various methodologies used. The cost-value technique, under fire, was found to be substantially superior to other selection methods utilized.

While the cost-value selection technique was noted to be superior, other methodologies employed are briefly described below for contrast.

2.1 Sole Source

This methodology will be required in some applications due to a single vendor possessing the required piece of equipment or software capability. On the other hand, the typical reason encountered in applying this method, software compatability, may not be a valid justification for remaining with a single vendor. Software conversion and established vendor relationships should not normally be the driving selection criterion of the system.

2.2 Overall Impression

Two of the groups' techniques boiled down to subjective judgement. Their feelings as to the good or bad points of each individual proposal swayed their decision. In short, the administrators took a vote on the technical merits.

2.3 Cost-only (or "low-bid")

Two other groups' selection methodology resulted in determining which vendor's system could perform the job and selecting the cheapest one. Note, however, that often a small price differential can result in a system which can far out perform the low bid system.

2.4 Weighted-scoring

Although the group using this technique attempted to preassign points to items required and desired, no meaningful relationships between points awarded for performance and points awarded for low cost existed. The group's selected system, based upon the proposal that earned the most points, was open to challenge with respect to several aspects.

2.5 Cost/effectiveness Ratio

This evaluation technique, although not specifically utilized by any of the competing groups is still generally used throughout DoD activities. This evaluation technique is really a subcategory of the weighted-scoring technique, except that here, by dividing the cost category by the effectiveness category, the user can select systems with the lowest ratio of cost

	Vendor A		Vendor B
a. System cost	System 1	System 2	System 3
site preparation	---	---	---
transportation/installation	---	---	1,500
training	2,000	2,000	3,040
equipment	147,605	159,395	159,197
maintenance contract	73,800	79,700	63,680
total cost	223,405	241,095	227,417
b. system value			
graphics	---	---	19,500
space	---	---	---
software	25,000	25,000	25,000
equipment interface	10,000	10,000	10,000
vendor support	1,875	1,875	3,750
equipment delivery	6,000	6,000	10,000
total value	42,875	42,875	68,250
c. total cost value			
system costs	$223,405	$241,095	$227,417
system value	42,875	42,875	68,250
system cost value over 5 year life span	180,530	198,220	159,167

Figure 2. Cost-value Summary

to effectiveness. This technique suffers from the same deficiency as the weighted-scoring method in that the establishment of a meaningful relationship between cost and effectiveness is quite difficult, as is the actual definition of "effectiveness".

3. Theory of Cost Value Defined

The cost-value technique recognizes the necessity for comparatively evaluating the desirable features offered by various computer systems which meet certain minimum system performance standards. The ability of the proposed systems to perform the functions specified as mandatory requirements are not evaluated, but are validated. If it is found that a vendor's system cannot meet the minimum levels of performance, the proposal is eliminated from further consideration. At this point it is possible to compare desirable features offered in the proposals to determine whether the claimed desirable features are important in themselves or are mere incidental elements. For example, a 60-nanosecond memory and a 1,000-cards-per-minute card reader are not important in themselves. More important and desirable than the specific operating characteristics of these devices is how these features affect the overall performance of the system. The value of every desirable feature which is considered to be important must be studied.

One distinguishing feature of the cost-value technique is the assignment of value (in dollars) to the desirable features. By assigning a value to the desirable features, a common denominator is defined which allows all useful features offered to be compared. These comparisons are relative to each other and not to the total cost of the system. Although the values assigned to the various desirable features will be a function of the requirements for the system to be procured, a value, when realistically assigned, can be understood, examined, discussed, (and modified if necessary) independently of all others.

The cost-value technique takes the total cost of a proposed system as quoted by the vendor and then deducts from it the total value of all the desirable features included in that proposal. The difference represents the derived cost of satisfying the mandatory requirements stated in the specification package. The system having the lowest derived cost for satisfying the mandatory requirements becomes the system selected, since the values of the desirable features offered were taken into consideration in calculating the derived cost.

In the cost-value technique, the proposals are scored or ranked by a method referred to as the cost-value accounting scheme. Actually, this is cost and value accounting, since some of the values and costs used, although stated in dollar terms, may not involve actual expenditures. (Figure 2 illustrates such an accounting scheme.)

This ranking could also be looked at from a value-to-cost ratio basis. The results will be the same if value is considered in its full sense as value of mandatory requirements plus the value of the desirable features offered, and cost is considered to be the total cost for the package over the estimated life of the system.

4. Using the Cost Value Technique.

4.1. Determining values

To avoid any bias or appearance of bias on the part of the selecting/purchasing agent, the appraisal of desirable features should be completed as an intregal part of the development of the RFP in order to be fair to the vendors and users alike. The cost-value technique's approach to the desirable features (those likely to be offered by the vendors above and beyond the mandatory requirements) is to appraise them determining whether they are worthy of inclusion in the evaluation, and if so, determining the dollar value of these features. Full participation (if possible) of the end user in this process is extremely valuable. Where the criteria are not available at RFP time, vendors have no alternatives but to make assumptions as to what is needed. These assumptions may or may not be valid.

By pre-establishing the mandatory requirements and desirable features, all concerned know the "rules of the game," i.e. the criteria upon which selection will be based. Thus the user and the vendor are aware of what to expect from one another upon reviewing final proposals.

The procuring/selecting agent must take the lead in the value determination process. Although the end users generally understand their technical problems better than the administrator, they sometimes have a myopic view of the value of systems and tend to "want everything". Hence, they alone may not be able to accurately and independently assess the value of each of the desirable features. Some fiscal guidance is usually required.

A simple listing of features which may be considered as desirable must first be established. Values are then assigned to those items which are of importance to the system's anticipated operation. Figures 3 and 4 contain sample listings of these features which one might consider at this point in the process.

4.2 Evaluation Templates

The value of a desirable feature may be established independently of the proposals for predetermined ranges which are created to show value for varying amounts of each of the desirable features being evaluated. These predetermined ideas of worth may be referred to as "evaluation templates". Evaluation templates incorporate two generalized cost-value concepts which form the basis for assigning a value to a particular desired characteristic.

The first concept is that of establishing limits. Assessing the value of any given item is a difficult task. The logical starting place is the cost of the item. If the item is competitively available, its value should never greatly exceed its market cost. For example, a utility costing \$3000 should be valued at approximately that amount or less if it is categorized as a desirable feature. If the cost of having a mathematical subroutine written by a software consulting group is \$10,000, it would be reasonable for a user with little use for this subroutine to establish a value of only \$500 for its availability. If on the other hand, only one vendor can supply a critical subroutine, its value is almost indeterminate. However, this case should not arise in cost-value assessment for two reasons:

1. If the item is critical, it should be listed as a mandatory requirement and should not require value assessment.
2. If the item is critical and can be procured from only one vendor, then the full selection should have been handled as a sole-source procurement, again making value assessment unnecessary.

The second concept is that of diminishing values. The prevailing thought behind most evaluation templates created is that, as more of an item becomes available, the incremental benefit from inclusion of additional units of that item decreases.

These evaluation templates are the practical criteria employed by the selection group in assessing the relative worth of particular desirable chacteristics. Techniques used to develop these templates run the entire spectrum from simple ratios of cost-vs-expansion to in-depth statistical analysis of job loads and/or turn-around times.

1. Costs
    - One-time costs
        - Site preparation
            - Electrical
                - Air conditioning (cooling, heating, and humidity control)
                - Power supply (including all wiring)
            - Space for equipment
                - Facilities (walls, ceiling, painting, draperies)
                - False flooring (including bracings)
        - Equipment installation
        - Equipment transportation
        - Vendor support
            - Personnel (analysts, programmers, operators, instructors)
            - Training (including transportation, living costs)
            - Existing programs
            - Backup facilities
            - Machine time (checkout)
            - Documentation
            - Program and data conversion
    - Continuing costs
        - Procurement of computer system equipment (falls in one-time costs category if system is purchased)
            - Central processor and associated equipment (console, floating point option, real time option, etc.)
            - Peripheral computer equipment: on-line or off-line (remote-inquiry device, card reader, printer, etc.)
            - Auxiliary equipment
                - Keypunch machines and other data-creating devices
                - Printers, sorters, collators, etc.
        - Operation and manitenance of all electrical equipment
        - Personnel (manager, analysts, programmers, operators, etc.)
        - Program development
        - Supplies (magnetic tape, printer paper, cards, etc.)
        - Indirect cost for space used

2. Vendor's support of system
    - Program assistance
        - development, writing, converting, emulation training, analysts, programmers, operators, managers, users
    - Maintenance offered
        - backup availability
        - program testing
        - hours, shift schedules, location
        - existing software maintenance plans
        - operating system maintenance plans
            - schedulers, I/O control, memory allocation, etc.
            - sort, merge, system simulators or emulators, COBOL, FORTRAN, report generator, etc.
        - documentation
        - personnel loaned
            - analysts, programmers, operators, users

Figure 3. Costs and Vendor Support [5]

1. Equipment characteristics
    - Speed
    - Time required to complete applications specified
    - Instructions
        - add time (fixed and floating)
        - multiply time (fixed and floating)
        - divide time (fixed and floating)
        - move
        - other instructions
    - Peripheral equipment
        - printer (lines per minute)
        - card reader (cards per minute)
        - card punch (cards per minute)
        - magnetic tape units (characters per second)
        - ias (characters per second, average)
        - other equipment (through all other peripheral equipment listed)
    - Capacity
        - storage capacity of main memory (core)
        - storage capacity of immediate-access storage (ias)
        - storage capacity of magnetic tape
        - characters per printed line
    - Compatibility
        - program, tapes, cards
    - Reliability
        - error detection, error correction techniques, mean time to failure, etc., redundant components
    - Special features
        - memory lockout, parallel processing

2. Expansion potential
    - Slack time (amount of available free time on each piece of equipment)
        - central processor, magnetic tapes, immediate access stroage, card punch, printer, remote terminals, etc. (through all other equipment offered)
    - Maximum expansion (number of units that can be added to system)
        - magnetic tapes, immediate access storage, card punch, printer, etc. (through all other system equipment offered), extra core, disk drives
    - Compatible equipment
        - larger processors
        - higher performance units

Figure 4. Desirable Features - Equipment Characteristics and Expansion Potential [5]

4.3. Determining Cost

All cost items must be considered in the evaluation. Items such as the cost of supplies or personnel may prove to be nondifferentiating in a given selection, but they should be included for the sake of completeness.

Treating cost items as one-time costs or continuing costs is a matter of cataloging. The following rules must govern any proper treatment of cost items:

1. The costs must be spread proportionately over the expected life of the system.
2. The system costs should reflect the costs of any planned system expansion.

Costs should be called out for individual items (except when the entire system is to be purchased as a package). No cost items should be duplicated. That is, the system should not be charged twice for the same equipment or service. For example, if a card reader is used both online and offline, its full cost should not be shown twice. Program development costs should be handled similiarly.

Wherever possible, equipment characteristics should be measured in terms of system performance (time, capacity, etc.). This determines the system's cost and responsiveness and shows any room for expansion within that system's boundaries.

The following are some general rules with regard to cost-value technique applications.

1. The user's value figure for the desirable features should be independent of vendor pricing except that values should not exceed market price.

2. In some cases, the vendor may not be able to give cost figures for supplying service or equipment equal to some of the levels/specifications desired, due to lack of facilities. In other cases, it might be practical only for the vendor himself to provide the service. In such cases, the cost value of such a service must be determined individually, and may be considerably higher than the costs charged by any other vendor. The higher cost value should then be the base.

3. The value of the desirable features should reflect the user's judgement as to the items' relative operational utility.

4. All items being evaluated should be explicitly identified as mandatory or desirable with appropiate cost and/or value assessments.

5. Each item must be evaluated individually in the context of mandatory or desirable and the impact it contributes toward the overall system.

6. At the outset of the evaluation the expected useful life of the system must be stated. Subsequent evaluation of all mandatory and desirable features should then reflect this planned life and/or anticipated expansion in the cost/value accounting scheme.

The circumstances of the selection of a given computer system determine the specific items to which prospective evaluators will apply the cost-value technique. The methods described herein for determining cost and values are by no means the only methods that might be used. The key point is to apply the cost-value technique with the end user in mind.

5. An Example of Cost Value Selection

In the simulated DoD procurement (see Figure 1 for full system specifications), the cost-value technique was employed to stratify the processing demands into the two major categories of mandatory and desirable features.

5.1 Mandatory features

1. The proposed system must be capable of accomodating a daily scientific processing workload of 2.44 x 10**10 floating point operations on an average day and 3.08 x 10**11 on a worst case day. Associated with these floating point operations would be overhead instructions in the amount of 9.76 x 10**10 and 1.23 x 10**12 for the average and worst case days respectively. These figures represent a 95% confidence interval about the workload mean[2].

2. The proposed system must accomodate the daily data management workload of:
(a) maintaining approximately 40 megabytes of data base on line for query by the time-sharing terminals.
(b) maintenance of data bases by daily transaction postings. It was estimated that each record contained 150 bytes of data and that 100,000 instruction executions were required per record update. Also, backup copies of updated data bases were required on removable media.

3. The proposed system must be compatable with FORTRAN IV.

4. The proposed system must have a support/development system capable of time-sharing operation on 3 - 7 terminals.

5. Hardcopy output is required.

5.2 Desirable features

1. Terminal and/or hardcopy graphics capability is desired.

2. Mainframe, mass storage devices, and operator's console should fit within 150 square feet of floor space.

3. Data bases should be supported by a data base management system.

4. Tape interface for entry of the data base update transations is desired.

---

[2]Four studies were shown by Stone [6] that attempted to determine the average instruction mix of scientific programs. The study conducted by Arbuckle (69) yielded the highest percentage of the four studies for floating point calculations, that percentage being 17.1%. In order to provide additional computing power in the system being designed, an assumption was made that 20% of each scientific program would consist of floating point calculations. Given that assumption, the remaining 80% of each program would be overhead (loa, sto, etc). The 20% floating point portion of the programs was subdivided into the percentages of 70% add/subtract, 20% multiply, and 10% divide.

5. Purchase cost should be in the range $100,000 to $200,000.

6. Desire to have 100 hours of program test time on hardware made available by the vendor.

7. Desire delivery schedule such that installation could be completed within 180 days.

Resource constraints on the simulated procurement experiment precluded considering all relevant factors discussed in Figures 3 and 4. Only the most important factors are shown.

5.3 Evaluation Templates for Desirable Features

The presence or absence of these desired features in a vendor's proposal was incorporated into simple evaluation templates in the following manner.

1. Graphics Capability- If a proposal included the graphics feature, $19,500 would be deducted from the vendor's quote thus enhancing the competitiveness of the bid. This amount represents the cost of a graphics terminal with a microprocessor device which would be incurred if a separate contract. was required for the acquisition of this capability.

2. Space Available- If additional space is required for hardware installation, a negative value charge would be assessed and added to the quote. This penalty charge would consist of $5,000 for the removal of one structural wall and $80/ square foot for each additional square foot of floor space needed.

3. Data Base Management- The value of a record management software package was determined to be $25,000. This amount would be deducted from any vendor's quote if the proposal included that capability.

4. Tape Interface Capability- The daily data base update transactions would be received on tape. If the proposed system could not accept the tape input and data conversion to another medium was required, a cost of $50/tape would be incurred. For lack of this tape interface capability, a penalty of $5,000 would be assessed and added to the vendor's quote.

5. Equipment Delivery- The cost-value of early delivery was determined by the costs which would be incurred in leasing computational services if the desired installation deadline could not be met, or the savings that would result from early delivery. The following schedule applies:

Delivery Date		Value	
0-30	days	10,000	Positive
31-60	days	8,000	values
61-90	days	6,000	(to be sub-
91-120	days	4,000	tracted from
121-150	days	2,000	proposal)
151-180	days	-0-	
181-190	days	-5,000	Negative
191-210	days	-10,000	values
211-240	days	-15,000	(to be added
241+	days	-20,000	to proposal)

6. Vendor Support during Program Test Time- 100 hours of machine time was desired for program compatability testing and debugging. The value assigned to this hardware time was set at $50/hour for predelivery time and $25/ hour for post-delivery time. A maximum ceiling of $5000 was established. Proposals offering this machine time would have quotations reduced by the appropriate amount.

5.4 Costs

Costs would be computed on an additive basis over the estimated life of the system, considering both one-time and continuing costs. The system with the least cost would be considered the most desirable. The cost items that would constitute factors in this determination were:

- equipment cost based on 5 year life
- maintenance costs over 5 year life.
- vendor support of the system.
- site preparation costs.
- equipment installation charges.
- equipment transportation charges.
- any other differentiating cost items.

With the mandatory and desirable features now identified, segregated, valued, and documented, vendors were contacted and presented requests for proposals (RFP's). Briefings were held with each vendor in order to thoroughly describe the system requirements, all aspects of the intended environment, and the method by which the individual proposals would be competitively evaluated under the cost-value technique.

Two vendors formally replied to the RFP, providing a total of three systems that met all the mandatory features and many of the desirable features. Figure 5 is a tabular summary of the bid proposals submitted by these two companies and represents an itemized listing of the cost elements in the vendors' proposals. Figure 2 itemizes the value credits granted as a result of the vendor providing desirable features in addition to

Vendor A	system 1	system 2
package:	88,900	101,000
cpu, memory, disk drive, disk controller, tape drive, tape controller, operator's console, operating system & software, battery backup system, cabinets & assembly hardware		
add-on disk drive	18,000	19,000
multiplexor	2,310	2,310
crt terminals (6)	11,400	11,400
terminal cables		
25 feet (3)	180	180
100 feet (3)	360	360
line printer	11,800	11,800
array processor		
cpu	11,900	11,900
memory	9,700	9,700
software	2,500	2,500
communications and software interface	3,850	3,850
subtotal	160,900	174,000
less discounts	13,295	14,605
total hardware cost	$147,605	$159,395
training	2,000	2,000
maintenance (approx. 10% hardware purchase cost/year for 5 years)	73,800	79,700

Vendor B	system 3
cpu	64,000
peripheral interface	4,120
disk drive	33,000
tape drive	10,550
line printer	4,950
asynchronous I/O controller	4,020
crt terminals (6)	12,060
operating system & support software	800
array processor	
cpu	11,900
memory	9,700
software	800
communications and software interface	3,850
battery backup system	980
graphics terminal	9,900
graphics software	1,700
cabinets & assembly hardware	1,450
subtotal	173,780
less discount	14,583
total hardware cost	$159,197
training	3,040
maintenance (approx. 10% of hardware cost/yr) (79,600 less 20% for 5 yr contract)	63,680

Figure 5. Vendor Bid Proposals

meeting the mandatory features. The net total value amount was then applied to the total cost quotations from the vendors. Figure 2 depicts the adjusted vendor bids, or derived cost as described in Section 3, after applying the cost-value model.

In sum, the vendor bids were validated and evaluated for mandatory and desirable features. The mandatory features were validated by estimating the time required (instead of benchmarking) for scientific and data base workloads. The desirable features were evaluated for accuracy and completeness. Applying the cost-value technique resulted in Vendor B having the lowest cost-value derived cost, and, as a result, was chosen the winner.

6. Merits of the Cost-value Technique

While implementing the cost-value technique in the competitive environment of the simulation, certain elements of the process highlighted themselves as having considerable merit. The most noteworthy was the manner in which this technique forces one to refine system requirements more carefully than in other selection techniques. Most other procurement methods leave areas of ambiguity in the total system requirements and concentrate only on areas of interest to the user. For example system cost, processing capability, etc., are often covered in unequal levels of detail. As a result of incomplete requirements definition, the vendors then have the liberty of proposing a much wider variety of systems, emphasizing more strongly those aspects in which their systems excel. The marketing assumption of "what is required" may or may not yield the system that is really needed. The cost-value technique requires intimate awareness of acceptable system performance criteria independent of the marketing environment of the computer industry. By prior determination of the mandatory and desirable features developed from the user's inputs, the specifications of the system are designed such that the best proposed system is the most strongly rewarded. This preparation makes the buyer a better "shopper" and, therefore, produces a system much better tailored to the needs of the users.

Also in this process, life cycle costs are explored. In examining these costs, lease/purchase decisions are facilitated and system growth margins per dollar cost can be compared. Managerial accounting models (internal rate of return, net present value, etc.) may be utilized to ensure the system will be efficient and effective for the organization.

The selecting/purchasing agent is forced to choose a system very close to what the user actually wants. All too often a buyer is presented with only minimum system specifications. If the buyer then procures a system that merely meets minimum specifications, the user's true needs may not be met.

The cost-value technique has a further beneficial impact upon vendor's responses. Once the vendors understand the contents of the RFP, they know exactly which systems are capable of meeting the mandatory items. They then tailor the potential systems to maximize their cost- value ratings. The "openness" of the selection method is appealing. Most of the sales personnel of the companies are not familiar with the cost-value technique, so some education is often required.

Clearly the authors found many favorable attributes of the cost- value technique as a system selection methodology. However, there are some limitations to the technique that should be discussed. Some are unique to this method while others are common to most methods commonly employed today.

7. Limitations of the Cost Value Technique

The first difficulty encountered was the determination of the mandatory and desirable features and the dollar-values that should be assigned to the desirable category.

Joslin [4] gives good examples of the boundary limits of the dollar-values for the desirable features, but it is the buyer who must ultimately assign a dollar-value to each feature. The more desirable a feature, the higher the value, but the actual value assignment is up to the buyer. If this subjectivity leads one away from the true market value of a commodity or service, then the financial creditability of the model may fall prey to a disgruntled vendor if his system is not selected.

Another problem noted in using the cost-value technique is that of determining a vendor's reputation for quality and service. This factor could be significant during the life of the system, especially with respect to hardware service changes and software updates and requires considerable scrutiny. The vendor's reputation was not considered as a mandatory or desirable feature in the experiment described above and was only superficially viewed. The final determination was subjective as to the vendor's reputation for service and quality. Even though the cost-value technique was not applied in evaluating the vendor's reputation for service and quality in the experiment, the following approaches might be pursued. Check user responses to <u>Datapro Reports on Minicomputers</u> [7] vendor/equipment surveys. Interview knowledgeable people currently using the equipment to get first hand information concerning the hardware and the vendor's industry wide reputation. If there are several installations of the proposed machine, a maintenance/reliability data base already exists. Hence, vendors who are typically reluctant to divulge this data, may do so if enough pressure is applied. The vendor reputation problem is more difficult for small peripheral companies. Usually there is a price break for plug compatible devices. However, their service and reliability factor is much harder to evaluate. There may also be a hardware interface problem. The small vendor's reputation may be determined only by talking to actual users and getting firm reliability data. For example, in the simulated experiment selecting an arithmetic processor required evaluation of smaller vendors.

The next area of concern that was encountered has been briefly mentioned. Most of the vendor sales personnel contacted did not understand this selection technique. The level of understanding ranged from "never heard of it" to "never actually used it". therefore, if one is to pursue this technique in today's market it is encumbent upon the buyer to properly educate (or at least inform) the vendors with repsect to this method. Even then, the selecting/purchasing agent should anticipate receiving some proposals which may be developed in part on misintreptations. Time permitting, the vendor should be contacted and allowed to correct the proposal following further explanation.

The technique does not fully explain how one goes about designing an integrated system combining the equipment of one vendor with that of another. This problem is not unique to this acquisition methodology, but is universal with all selection methods. This problem is projected to be ever increasing, particularly in view of the exponentially expanding peripheral hardware market. Specifically, during the selection simulation, problems were encountered in the selection of the various graphic units, array and floating point processors, and terminals. "Off-the-shelf" package systems were proposed that met the stated requirements, even though some of the component subsystems were barely capable of meeting specifications, while others in the package were an "over-kill". Although it would have been desirable to purchase each individual element of the system from the firm who specifically met the requirements, it was found that the price breaks afforded by purchasing the package systems from a single vendor made distributed purchases cost prohibitive. This shortcoming of the cost-value model is intensified as the system under consideration increases in size. There are many variables to be considered in trying to formulate a guideline to apply in this case: different cost-values, how maintenance contracts may be effected, how splitting up the package may effect the "package price", whether the units are compatible, the costs of interface hardware and software in terms of dollars and performance degradation, and others. These questions must be examined in detail when attempting to "mix a system".

8. Conclusions

The authors found in the academic application of the cost-value technique a tool which is felt to be superior to other methodologies for system selection. Although certain shortcomings of the model were noted, these deficiencies did not outweigh nor necessarily counter the advantages of the cost-value technique. The technique has been presented to create an awareness of the model and to demonsrtate its applicability to the current environment. Through the highlighting of the merits and deficiencies of this technique, it is anticipated that the buyer may better predict possible problem areas and formulate plausible alternatives should he desire to employ this model. The ultimate goal, of course, is the selection of optimal systems.

The authors would like to acknowledge the encouragement and support of Dr. Lyle A. Cox.

References

[1] Datapro 70, The EDP Buyer's Guide, Datapro Research Corp., vol. 3, McGraw-Hill Co., Chicago, July 1978.

[2] McQuaker, R. J., Computer Choice, North-Holland Publishing Co., Amsterdam, 1978.

[3] U. S. President's Reorganization Project, National Security Team Report, Federal Data Processing Reorganization Study, October, 1978.

[4] Joslin, E. O., Computer Selection, Technology Press Inc., Fairfax Station, Va., 1977.

[5] Bayraktar, A. N., Computer Selection and Evaluation, master's thesis, U. S. Naval Postgraduate School, Monterey, Ca., 1978.

[6] Stone, H. S., Introduction to Computer Architecture, Science Research Associates, Inc., Chicago, 1975.

[7] Datapro Reports on Minicomputers, Datapro Research Corp., McGraw- Hill Co., Chicago, 1979.

COMPUTER SELECTION: TO MEASURE OR NOT TO MEASURE

S. A. Mamrak

The Ohio State University
Columbus, Ohio 43210

Measurement phases of a computer selection process are costly and time-consuming. A decision must be made prior to measurement whether the data obtained will be worth more to the decision maker than the cost of conducting the measurement experiment. This paper illustrates how statistical decision theory can be applied to answer that cost/value question.

Key Words: Computer measurement; computer procurement; computer selection; decision theory.

1.0 Introduction

The selection of a computer system or service is a complicated process with many different evaluation phases. Mandatory performance criteria must be assessed, followed by desirable performance criteria. Further the criteria may be measurable (such as a certain turnaround time for processing a benchmark workload) or nonmeasurable (such as possession of a certain compiler).

Nonmeasurable criteria, either mandatory or desirable, are relatively inexpensive to evaluate. Typically they require an answer to a question like "Does the computer system have a text editor?" or "Can the computer system be delivered before January 1, 1980?" Measurable criteria, on the other hand, are typically expensive to evaluate, requiring sophisticated measurement equipment and the time and talents of skilled personnel.

Because of the high cost of obtaining measurement data, it is important to determine under what circumstances measurement phases of a selection process should in fact be executed. The question is one of cost versus value: will the data resulting from a measurement experiment be worth more to the decision maker than the cost of obtaining the data. This is a fundamental question addressed by statistical decision theory. The purpose of this paper is to illustrate how decision theory can be applied to the measurement question in a computer selection process.

2.0 Background

A previously developed model of the computer selection process is presented in Figure 1 [AME79]. A sequence is illustrated in which classes of selection criteria should be applied in the process of choosing the best alternative computer system.

Initially, criteria must be defined which state what is meant by best. These criteria can be categorized in two ways. They are either measurable or nonmeasurable and they are either mandatory or desirable. Thus, the selection criteria can be classified as Mandatory Nonmeasurable (MN), Mandatory Measurable (MM), Desirable Nonmeasurable (DN), and Desirable Measureable (DM). Examples of these criteria are presented in Table 1.

The sequence of applying the selection criteria is composed of three phases. The application of MN criteria in Phase I is managed easily, since each alternative either does or does not have the required characteristics. Phase II, which further reduces the number of alternatives, involves the application of MM criteria.

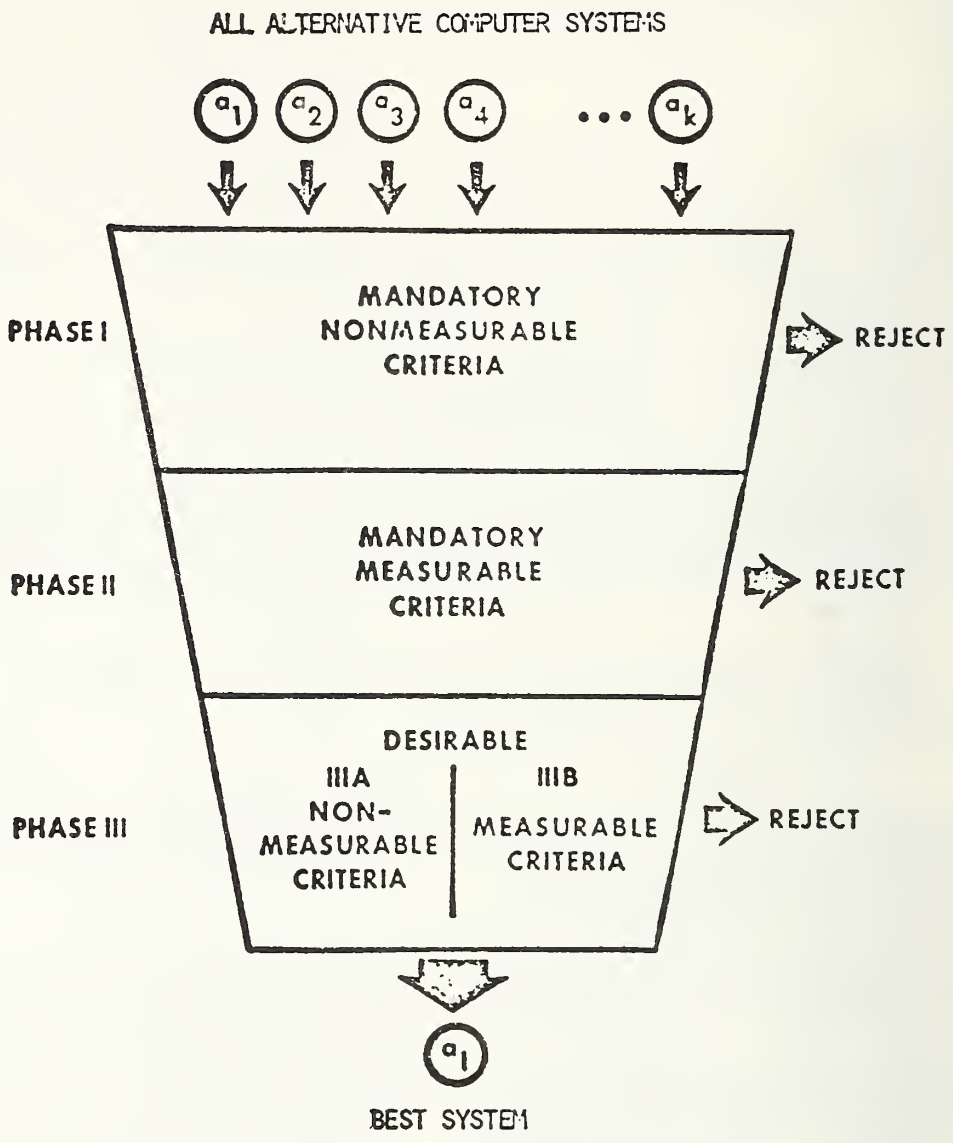

ALL ALTERNATIVE COMPUTER SYSTEMS
a_1
a_2
a_3
a_4
a_k
PHASE I
MANDATORY NONMEASURABLE CRITERIA
REJECT
PHASE II
MANDATORY MEASURABLE CRITERIA
REJECT
PHASE III
DESIRABLE
IIIA NON-MEASURABLE CRITERIA
IIIB MEASURABLE CRITERIA
REJECT
a_1
BEST SYSTEM

FIGURE 1. Model of the Computer Selection Process

Experiments are conducted on the alternatives which survived Phase I and the performance of each is documented. For each MM criterion, measurements are gathered from every system and a decision is made as to whether or not the criterion is satisfied. Failure to satisfy a single MM criterion results in an alternative's elimination.

Finally, determination of the best alternative is made in Phase III. This stage is separated into two parts, Phase IIIA for the application of DN criteria and Phase IIIB for the application of DM criteria. For DM criteria, data are collected from each alternative being evaluated and compared. On the basis of relative performance with respect to the DN and DM criteria, an

Table 1. Examples of Performance Criteria

Type	Example
Mandatory Nonmeasurable	1. The system must be fully delivered and operational no later than September 1, 1979.
	2. Timesharing service must include FORTRAN, Basic, Lisp, SNOBOL and editing facilities.
Mandatory Measurable	1. The mean-time-to-failure for a specific one month period must be greater than 4 hours.
	2. 95% of all trivial command response times must be less than 1 second.
Desirable Nonmeasurable	1. It is desirable that the system include Pascal and COBOL facilities.
	2. It is desired that the system provide a text editing capability.
Desirable Measurable	1. It is desired that the system provide a mean turnaround time for the benchmark run of 5 minutes or less.
	2. It is desired that 95% of all trivial command response times be 0.5 seconds or less.

alternative is selected as the best. Inherent in the application of Phase III selection criteria is the notion of "preference." Alternative systems must be evaluated with respect to various desirable criteria and assigned values over a preference range. These values must then be summarized to form a single composite preference measure for each system.

Basically, there have been two lines of approach for deriving global preference measures for competing computer systems, excluding an ad hoc technique of "eye-balling" the data in some intuitive way (see TIM73 for a complete discussion). The first of these is a "preference scoring" approach, also called "scoring systems" or "weighted factor methods," and the second is a "cost-value" technique. The primary difference between the two approaches is the time in the decision-making process at which cost is considered. In the first approach a global performance indicator is obtained for each alternative system and then performance/cost pairs are used to make the final selection. In a cost-value approach, cost is the sole basic preference measure applied to each criterion under consideration.

The preference scoring approach is in general held to be too formal to model the process it represents. It requires too many decisions of too exact a nature. Given a set of performance criteria for selection, an analyst must first weigh the subjective importance of each performance criterion relative to the others. At the same time, a range of possible values must be assigned to each criterion, from minimally acceptable to most desirable. Then, each system under study must be evaluated with respect to each criterion, and based on the evaluation, assigned a suitable value within the prescribed range. For non-measurement data, this assignment is purely subjective. Finally, global value measures are derived, to be combined with cost to form value/cost preference pairs for each system. The selection of the best system is then made by examining the value/cost pairs.

The cost-value approach, credited mainly to Joslin [JOS77], is free from most of the disadvantages of the preference scoring approach and is more widely accepted. Basically, a dollar value is placed on each measurable and non-measurable desirable criterion. Any system meeting a performance criterion is credited with its corresponding dollar value, relative to the degree to which the criterion is met. The total credits for a system are subtracted from the total cost and the system with the lowest price is selected as the best. An example of a final cost-value accounting table is presented in Figure 2. Total costs and total values of each contending system were calculated and cost minus value figures (last line in the Figure) were obtained. According to the Joslin method, the computer system proposed by Vender Y_A would be selected.

The nature of the measurement question in computer selection can be clarified by reference to Figure 2. The only measurable criteria evaluated in that study was run time for the benchmark workload. Values were assigned as shown in the Figure, based on measurement results. The question is: was it necessary to evaluate workload run time, a rather expensive and lengthy process, or could the correct selection have been made without this information?

It should be noted that only desirable criteria are assigned values in the Joslin scheme. Meeting mandatory requirements enables a system to be entered into the cost-value competition. So, the question to measure or not applies only to desirable, measurable criteria, since all mandatory measurable criteria must be evaluated.

3.0 Statistical Decision Theory

The decision problem addressed by statistical decision theory is of the same form as the measurement question in computer selection. The decision problem (see RAI61 for a more extensive discussion) is concerned with the logical analysis of choice among courses of action when

1) the consequences of any course of action will depend upon the "state of the world"

COST-VALUE ACCOUNTING TABLE

Items	Vendor X	Vendor Y_A	Vendor Y_B	Vendor Z
Cost				
Site preparation	$ —	$ —	$ —	$ 2,000
Transportation and installation	—	—	—	—
Vendor support	—	—	—	—
Operating personnel and conversion	310,000	268,000	268,000	241,000
Equipment and maintenance	359,900	493,000	652,700	753,100
Total cost	669,900	761,000	920,700	996,100
Value				
Workload	$ 27,750	$ 95,800	$ 95,800	$128,500
Equipment expansion	—	—	75,800	125,500
Sort/merge	80,000	100,000	100,000	100,000
Automatic debug	—	—	—	60,000
PERT	—	10,000	10,000	20,000
On-site maintenance	—	40,000	40,000	40,000
Program test time	10,000	25,000	25,000	25,000
Program support	40,000	40,000	40,000	40,000
Equipment delivery	50,000	10,000	10,000	−30,000
Space available	− 7,000	—	—	—
Total value of desirable features	200,750	320,800	395,800	508,500
Cost of basic requirements	469,150	440,200	524,900	487,600

FIGURE 2. Cost-value Accounting Table [JOS77, p. 210].

2) the true state is as yet unknown

3) it is possible at a cost to obtain additional information about the state

Analogously, the computer selection problem is concerned with choosing among alternative computer systems where

1) the consequence of choosing an alternative system will depend upon the characteristics of the contending systems

2) the true characteristics are as yet unknown

3) it is possible at a cost to perform measurement experiments to obtain additional information about system characteristics

Further assumptions about the decision problem are that the choice has been confined to a well-defined set of contenders and that the decision maker will incorporate basic preferences concerning consequences and basic judgments concerning the state of the world in the decision making process. A more precise statement and analysis of the decision problem is presented in the remainder of this section and an illustrative sample application in a computer selection is presented in section 4.0.

3.1 The Basic Data

It is assumed that the decision maker can specify the following data defining the decision problem.

1. Space of terminal choices (or acts): C = {c}. The decision maker wishes to make a single choice from some domain C of potential choices.

2. State space: S = {s}. The decision maker believes that the consequence of making a choice c depends on some "state of the world" (or computer system) whose behavior cannot be predicted with certainty. Each potential system will be labelled by an s with domain S.

3. Family of experiments: E = {e}. To obtain further information on the importance to be attached to each s in S, the decision maker may select a single experiment e from the family E of potential experiments.

4. Sample space: Z = {z}. Each potential outcome of a potential experiment e will be labelled by a z with domain Z.

5. Utility Evaluation: u (e,z,c,s). The decision maker assignes a utility u(e,z,c,s) to performing a particular e, observing a particular z, making a particular choice c and then finding that a particular s obtains. The evaluation u takes account of the costs of experimentation as well as the consequences of the final choice.

6. Probability Assessment: $P_{s,z/e}$ on S X Z. For every e in E the decision maker directly or indirectly assigns a joint probability measure $P_{s,z/e}$ to the Cartesian product space S X Z, which will be called the possibility space. The joint measure determines four other probability measures:

a) The marginal measure P_s on the state space S. This is a measure the decision maker would assign to s prior to knowing the outcome of experiment e. In a computer selection problem, this would be an assignment of a probability to each s in S that it was in fact the best computer system.

b) The conditional measure $P_{z/e,s}$ on the sample space Z for given e and s. In a computer selection decision this is an assignment of a probability of an outcome z, given that experiment e was performed on system s.

c) The marginal measure $P_{z/e}$ on the sample space Z for a given e but unspecified s. In a computer selection decision this is an assignement of a probability of an outcome z, given that an experiment e was performed.

d) The conditional measure $P_{s/z}$ on the state space S for a given e and z; the condition e is suppressed because the relevant aspects of e will be expressed as a part of z. This is a measure the decision maker would assign posterior to knowing the outcome z of experiment e. In a computer selection decision, this is an assignment of a probability of computer system s being the best computer system, given outcome z.

3.2 Assessment of Probability Measures

For any given e, there are several methods for assigning the complete set of measures just defined. In a computer selection decision it is most likely that a decision maker will be able to assign marginal measures P_s ("a" above) based on past experience, judgment and information about already collected nonmeasurable criteria. In addition, assuming the measurement experiments will be conducted using valid statistical techniques, [AME79,MAM77,MAM79], the decision maker will also be able to assign conditional measures $P_{z/e,s}$ ("b" above), the probability of a given outcome, given an experiment e on s. From these assignments the marginal measures on Z ("c" above) and the conditional measures on S ("d" above) can be computed, as will be illustrated below.

The assignment of marginal measures, P_s, is one which requires basic judgements on the part of the decision maker concerning which system is in fact the best, given all the evidence accumulated up to this point. The evidence may be objective, such as the current ranking of the systems based on evaluation of all nonmeasurable criteria. It may be subjective, such as a feeling that one system is indeed better than all the others, and it may be some combination of subjective and objective pieces of data. At this point in the decision, the decision maker is called upon to make explicit the nature of the evidence being used to assign the P_s measures.

3.3 The Decision Tree

Given E, Z, C, S, u and $P_{s,z/e}$, the general decision problem is: how should the decision maker choose an e and then, having observed z, make a choice c, in such a way as to maximize the expected utility. This problem can be represented by a decision tree as shown in Figure 3. In that figure, the decision maker selects e_i in E and "chance" selects z_j in Z, according to the measure $P_{z/e}$. Then in step 3, the decision maker selects c_k in C and chance selects s_ℓ in S according to the measure $P_{s/z}$. The result is that the decision maker receives u(e,z,c,s) as a payoff.

In computer selection, when the decision is whether to measure or not, the decision tree is more appropriately analyzed using "backwards induction" rather than the forward analysis described above. Instead of starting by asking which experiment e the decision maker should choose at step 1, the decision maker starts by asking which choice c would be made in step 3 if a particular experiment e already has been performed with an observed outcome z. The difficulty with this approach is that, since the system s which will be chosen is still unknown, the utilities of the various possible choices are uncertain. This difficulty can be resolved by treating the utility of any c for a given (e,z) as a random variable $u(e,z,c,\tilde{s})$ and applying the operator $E_{s/z}$ which takes the expected value of $u(e,z,c,\tilde{s})$ with respect to the conditional measure $P_{s/z}$. Thus at step 3, looking forward, the decision maker must compute

$$u^*(e,z,c) = E_{s/z}u(e,z,c,\tilde{s}).$$

Moving backward to step 2, since the decision maker wants to maximize the expected utility, then faced with a given (e,z) at step 2, the choice c should be made for which u*(e,z,c) is the greatest. Since the decision maker is free to make any choice c, the utility of entering step 3 with history (e,z) and the choice c still to make is

$$u^*(e,z) = \max_c u^*(e,z,c).$$

Thus, u*(e,z) needs to be computed for all possible (e,z), and then the initial choice of an experiment can proceed.

At this point, step 1, the utilities of each possible experiment e are uncertain because the outcomes z are unknown. This difficulty is resolved in exactly the same way it was resolved when a choice c had to be made in step 3 without certain knowledge of s. Thus, the utility of any z for a given e can be treated as a random variable $u^*(e,\tilde{z})$ and the operator $E_{z/e}$ can be applied. So

$$u^*(e) = E_{e/z}\, u^*(e,\tilde{z})$$

where $E_{e/z}$ is applied with respect to the marginal measure $P_{z/e}$.

The decision maker wishes to choose the e for which u*(e) is the greatest and therefore, the utility of being at step 1 with the choice e to make is

$$u^* = \max_e u^*(e) =$$

$$\max_e E_{z/e} \max_c E_{s/z}u(e,\tilde{z},c,\tilde{s}).$$

The various steps and sample calculations are illustrated in the example application below.

4.0 Computer Selection Example

The cost-value data presented in Figure 2 will be used as the basis for a sample application of decision theory to the question of whether or not to perform a measurement experiment in a computer selection process. The table of Figure 2 will be modified however, as shown in Figure 4, to present cost-value results for the evaluation of nonmeasurable criteria only. Essentially, this change required eliminating the "workload" value entry line from the table in Figure 2, since that is the only measurable criterion shown. If a computer selection were done based on the nonmeasurable data, the system of Vendor X would be chosen. Assume a value table has

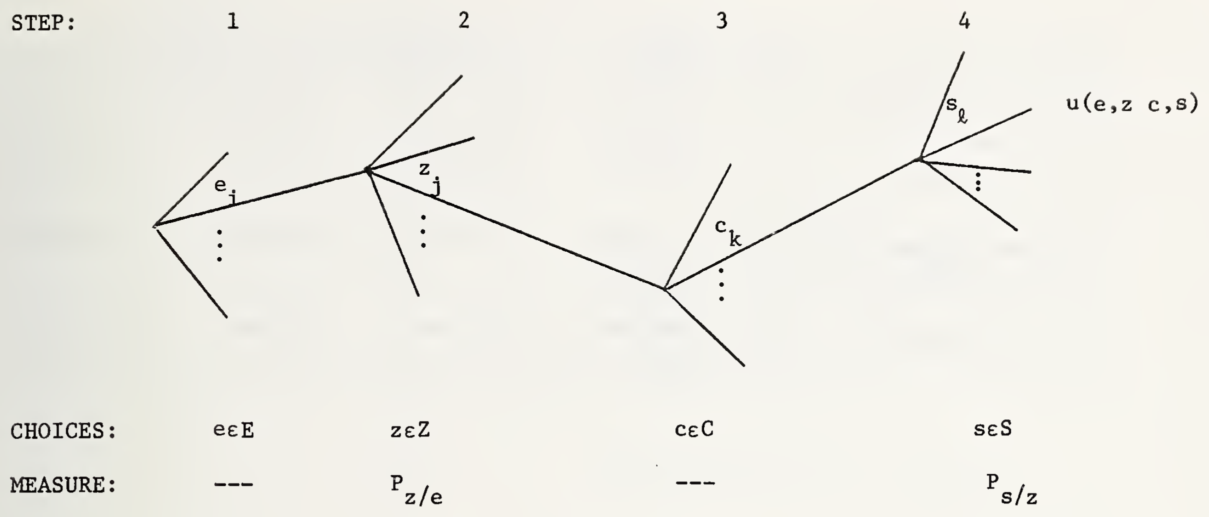


FIGURE 3. A Decision Tree

been appropriately determined for workload performance as is presented in Figure 5. Further, it is known that the cost of conducting the experiment is \$10,000. The decision to be made at this point is whether to conduct a measurement experiment (run and measure the benchmark workload) or not.

The maximum possible value to be gained by any system from conducting the experiment is \$60,000 (see Figure 5). A gain of a \$60,000 value to the system of either Vendor Y_B or Vendor Z will still not cause it to be selected, so these two systems can be eliminated from the competition at this point and the workload obviously need not be run on them. However, the "Cost of Basic Requirements" figures for Vendors X and Y_A are within \$60,000 of one another, so that it is possible that an experiment might reverse their ranking. The decision theory developed in section 3.0 can now be applied.

Table 2 presents the possible choices in this situation. Tables 3-6 present the various required probability measures. The values in Table 3 are obtained from the probability statements associated with the particular statistical procedure used to collect and analyze the experimental data. For example, one entry in Table 3 indicates that if the experiment is performed and if s_1 is in fact the right choice, then it is 85% certain that z_1 will be the outcome of the experiment. The values in Table 4 are estimates of the probability that a given system is in fact the best one (has true lowest cost). In Table 4 the probabilities show that the decision maker believes the two systems are equally likely to be the best. This could be construed as the "fairest" assignment of probabilities, or could reflect the judgement of the decision maker that in fact the two contending systems are closely matched overall.

The values in Tables 5 and 6 are derived from the values in Tables 3 and 4. In Table 5, the joint measure on S X Z is derived by computing for each s and z, $P_{z/e,s} \cdot P_s$. The marginal measure on Z is then obtained by summing the joint measures for each z. Table 6 contains conditional measures on S. Here $P_{s/z} = (P_{z/e,s} \cdot P_s)/P_{z/e}$.

The decision tree for this data is presented in Figure 6. The probability assignments to the various branches were taken from Tables 3-6. The utilities were computed using knowledge of the cost difference between the systems of Vendor Y_A and Vendor

ITEM	VENDOR X	VENDOR Y_A	VENDOR Y_B	VENDOR Z
Total Cost	$669,900	$761,000	$920,700	$996,100
Total Value of Desirable Features (Nonmeasurable)	$173,000	$225,000	$300,000	$380,000
Cost of Basic Requirements	$496,900	$536,000	$620,700	$616,100

FIGURE 4. Cost-Value Table Based on Nonmeasurable Criteria

Hours Required to Process	Value
50	$ 0
40	30,000
30	60,000

FIGURE 5. Value Table for Workload Run

TABLE 2. Possible Choices

SPACE	ELEMENTS	INTERPRETATION
C	c_1	Choose system of vendor X
	c_2	Choose system of vendor Y_a
S	s_1	System of vendor X has the true lowest cost
	s_2	System of vendor Y_A has the true lowest cost
E	e_0	Do not experiment
	e_1	Experiment (run the workload)
Z	z_0	Outcome of e_0 (a dummy)
	z_1	Outcome of e_1 more favorable to s_1
	z_2	Outcome of e_1 more favorable to s_2

TABLE 3. Conditional Measures on Z ($P_{z/e,s}$)

	e_0		e_1	
	s_1	s_2	s_1	s_2
z_0	1.0	1.0	.0	.0
z_1	.0	.0	.85	.15
z_2	.0	.0	.15	.85

TABLE 4. Marginal Measures on S (P_s)

S	P_s
s_1	.5
s_2	.5

TABLE 5. Marginal Measures on Z ($P_{z/e}$)

	Joint Measure on S X Z		Marginal Measure on Z
	s_1	s_2	($P_{z/e}$)
z_0	.0	.0	.0
z_1	.425	.075	.5
z_2	.075	.425	.5

$$P_{z/s} = P_{z/e,s} \cdot P_s$$

TABLE 6. Conditional Measures on S ($P_{s/z}$)

	s_1	s_2
z_0	.5	.5
z_1	.85	.15
z_2	.15	.85

$$P_{s/z} = \frac{P_{z/e,s} \cdot P_s}{P_{z/e}}$$

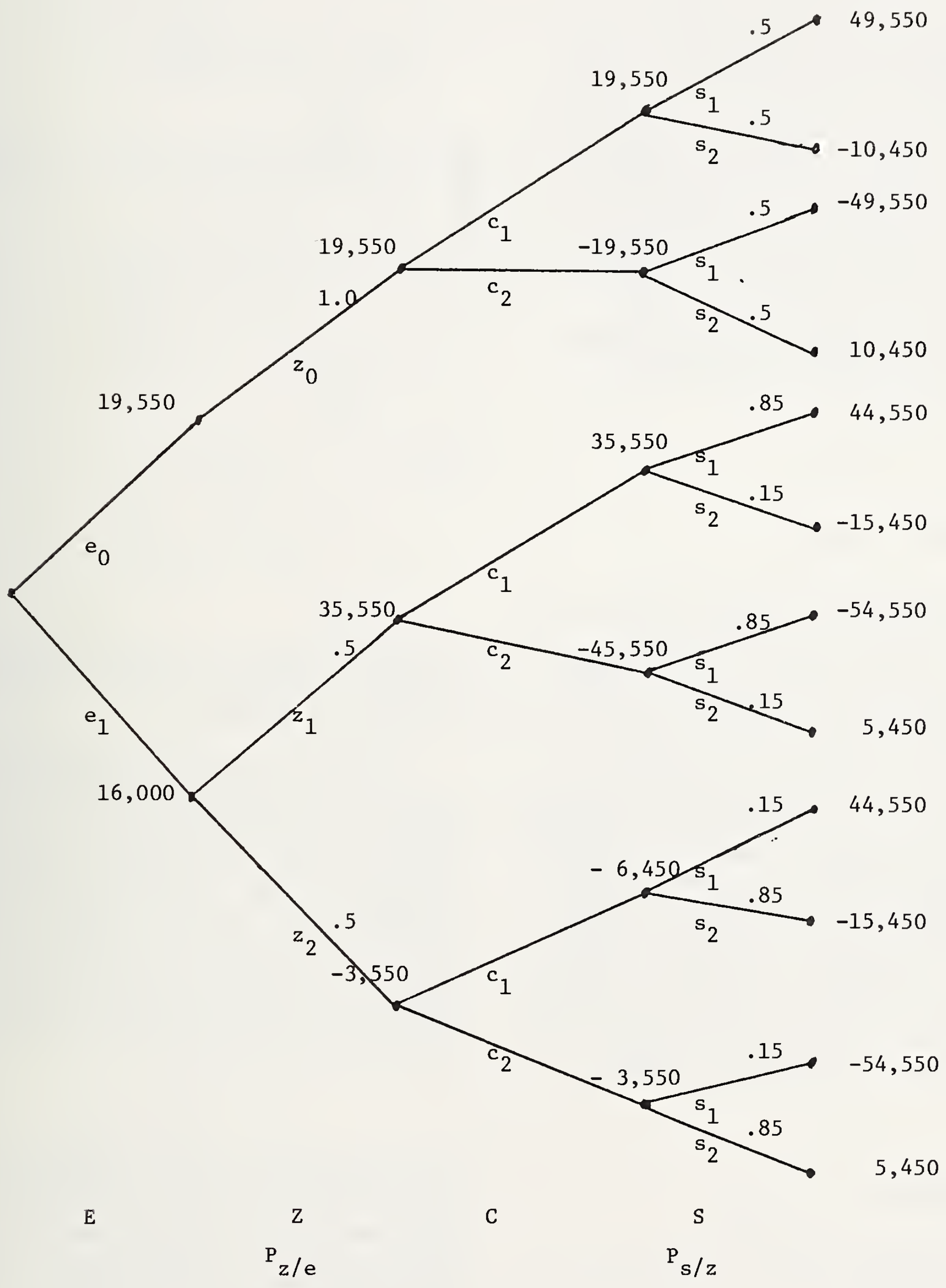


FIGURE 6. Decision Tree for Computer Selection (Example 1)

X from Figure 4 ($536,000 - $496,900 = $39,100), the range of possible values assigned to each system as a result of a measurement experiment ($0 - $60,000), and the cost of the experiment ($10,000). For example $u(e_0, z_0, c_1, s_1)$ is the utility of not performing the experiment, choosing the system of vendor X, and having that system in fact be the correct choice. The utility was calculated by averaging the maximum gain in this case ($39,100 + $60,000 = $99,100, since there already exists a $39,100 difference between the two systems and the best possible outcome of the experiment if in fact the system of vendor X is best, is that it will be assigned a value of $60,000 while the system of vendor Y will be assigned a value of $0) with the minimum gain in this case ($1, since that's the least difference recognizable between the two systems). Thus

$$u(e_0, z_0, c_1, s_1) =$$

$$\frac{1 + 99,100}{2} = 49,550.$$

A second example, to clarify the utility calculations, is the calculation for $u(e_1, z_1, c_1, s_2)$. This is the utility associated with performing the experiment and choosing the system of vendor X as a result of the experiment, when in fact the system of vendor Y_A was the best choice. This utility was computed by averaging the maximum loss in this case ($20,900 + $10,000 = $30,900, since the worst that could happen if in fact the system of vendor Y_A is best is that it should have been credited with a value of $60,000 and the system of vendor X with $0, which leaves a difference between the systems of $496,900 -$476,000 = $20,900, plus the cost of having conducted the experiment which is $10,000) with the minimum loss ($1, since this is again the smallest recognizable difference between the systems). Thus,

$$u(e_1, z_1, c_1, s_2) =$$

$$-(\frac{30,900 + 1}{2}) = - 15,450.$$

As an example of the calculations for the utilities $u^*(e,z,c)$:

$$u^*(e_0, z_0, c_2) =$$

$$u(e_0, z_0, c_2, s_1)P_{s_1/z_0} +$$

$$u(e_0, z_0, c_2, s_2)P_{s_2/z_0} =$$

$$-49,550(.5) + 10,450(.5) = -19,550$$

Having computed all the $u^*(e,z,c)$ the $u^*(e,z)$ can be computed, for example, as

$$u^*(e_1, z_2) = \max\{u^*(e_1, z_2, c_1),$$

$$u^*(e_1, z_2, c_2)\}$$

$$= \max \{-6,450,-3,550\}$$

$$= -3,550.$$

The calculation of the $u^*(e)$ is illustrated by

$$u^*(e_1) = u^*(e_1, z_1)P_{z_1/e_1} +$$

$$u^*(e_1, z_2)P_{z_2/e_1}$$

$$= 35,550(.5) + -3,550(.5)$$

$$= 16,000.$$

Finally, u^* is computed as

$$u^* = \max\{u^*(e_0), u^*(e_1)\}$$

$$= \max \{19,550, 16,000\}$$

$$= 19,550.$$

The optimal course of action is thus to use e_0. That is, don't experiment and choose the system of vendor X.

The results derived from the analysis of the decision tree depend critically upon the initial probabilities P_s and the utilities assigned by the decision maker. If in this case, for instance, the decision maker judged that it was more likely that the system of vendor Y_A was in fact the best, the original P_s values might have been assigned as $P_{s1} = .25$, $P_{s2} = .75$. As can be seen from the data in Tables 7-10 and the associated decision tree in Figure 7, the decision not to experiment would have been reversed.

5.0 Conclusions

The decision maker in a computer selection process needs assistance in deciding to measure or not, since computer measurement is an expensive and time-consuming activity. Statistical decision theory provides a formal analysis tool for making this decision. It allows the decision maker to incorporate all relevant information obtained up to the decision point, including information about system rankings based on nonmeasurable performance criteria. The cost of the experiment and the expected gain from performing it can also be explicitly incorporated into the

TABLE 7. Conditional Measures on Z ($P_{z/e,s}$)

	e_0		e_1	
	s_1	s_2	s_1	s_2
z_0	1.0	1.0	.0	.0
z_1	.0	.0	.85	.15
z_2	.0	.0	.15	.85

TABLE 8. Marginal Measures on S (P_s)

S	P_s
s_1	.25
s_2	.75

TABLE 9. Marginal Measures on Z ($P_{z/e}$)

	Joint Measure on S X Z		Marginal Measure on Z
	s_1	s_2	($P_{z/e}$)
z_0	.0	.0	.0
z_1	.21	.11	.32
z_2	.04	.64	.68

$$P_{z/s} = P_{z/e,s} \cdot P_s$$

TABLE 10. Conditional Measures on S ($P_{s/z}$)

	s_1	s_2
z_0	.25	.75
z_1	.66	.34
z_2	.06	.94

$$P_{s/z} = \frac{P_{z/e,s} \cdot P_s}{P_{z/e}}$$

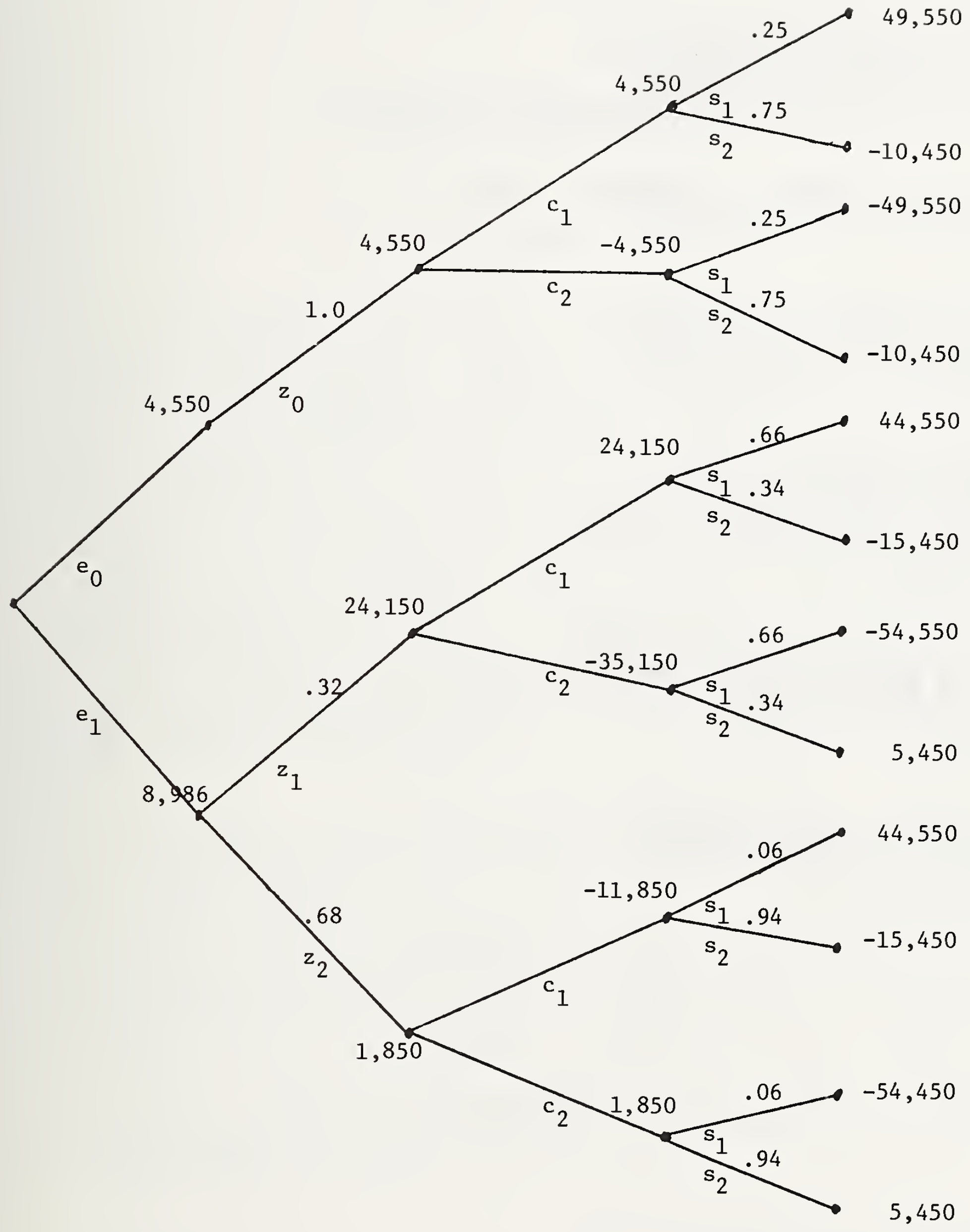


FIGURE 7. Decision Tree for Çomputer Selection (Example 2)

process. Thus, decision theory provides a means for determining if in fact the cost of obtaining the data is less than its value in the decision making process.

6.0 References

AME79 Amer, P. D., Experimental Design for Computer Comparison and Selection, Ph.D. Dissertation, Department of Computer and Information Science, The Ohio State University, March 1979.

JOS77 Joslin, E. O., Computer Selection, The Technology Press Incorporated, Fairfax Station, Virginia, 1977.

MAM77 Mamrak, S. A. and P. A. DeRuyter, "Statistical Methods for Comparison of Computer Services," Computer, Vol. 10, No. 11, November 1977, pp. 32-39.

MAM79 Mamrak, S. A. and P. D. Amer, A Methodology for the Selection of Interactive Computer Services, NBS Special Publication 500-44, National Bureau of Standards, Washington, D.C., January 1979.

RAI61 Raiffa, H. and R. Schlaifer, Applied Statistical Decision Theory, Graduate School of Business Administration, Harvard University, Boston, Massachusetts, 1961.

TIM73 Timmreck, E. M., "Computer Selection Methodology," Computing Surveys, Vol. 5, No. 4, December 1973, pp. 200-222.

BENCHMARKING INTERACTIVE SYSTEMS: MODELING THE APPLICATION

Christopher R. Spooner

The MITRE Corporation, McLean, VA

A benchmarking methodology has been developed that applies to competitive procurements for dedicated interactive applications, where the hardware and at least some of the operating system are to be off-the-shelf, but where a suitable implementation of the application does not yet exist. The methodology was developed by The MITRE Corporation under the sponsorship of the Federal Aviation Administration (Contract DOT-FA79WA-4184). It consists of preparing a model of the proposed application in the form of a set of modules of synthetic code, and requiring each of the competing vendors to run the model on his proposed system. The vendor provides two items: a package that interfaces his operating system to the buyer-provided model, and a separate benchmark driver (hardware plus software). The driver reads artificial messages off a buyer-provided scenario tape and delivers them in real-time. Having time-stamped the corresponding responses, it writes them to an output tape for subsequent analysis. The model is written in a very high level language, from which it is translated automatically by software into High Order Language for delivery to vendors.

This paper describes the modeling of the application. The model is not intended to measure the vendor's design or programming skills, but it must drive the hardware in approximately the same manner as would his eventual implementation. It must do this for all competing configurations, even though none of them are known when the benchmark is being developed. After briefly summarizing the overall benchmark approach, the paper shows how the application is modeled in terms of abstract components whose implementation can be vendor-dependent, with a top-down approach being taken to the functional requirements for the application. Relevent considerations are discussed in the text of the paper.

Key words: Benchmarking; distributed processing; interactive systems; kernels; modeling; computer performance measurement; real-time; simulation; synthetic programs.

1. Introduction

It has been recognized for some time that benchmarking interactive systems is considerably more difficult than benchmarking batch systems. A benchmark has been developed for the following situation. Competitive bids are being invited for a dedicated interactive system. Both the hardware, which is to be a cluster of loosely coupled mini-computers, and at least the essential nucleus of the operating system, are to be off-the-shelf. The application will be written by the successful bidder after the contract is awarded, but the hardware and the nucleus of the operating system are to be benchmarked before contract award. Reference [1] presents an overview of the techniques that have been developed for benchmarking under these circumstances.

After summarizing the overview, this paper develops one aspect of the methodology, namely the modeling of the proposed application.

2. Overview of the Benchmarking Approach

The procurement to which the new methodology is being applied is for the automation of the Flight Service Stations of the Federal Aviation Administration. Vendors compete to supply off-the-shelf hardware and to develop the application software. A key element in the automation is the set of 20 Flight Service Data Processing Systems (FSDPSs), each of which provides an interactive service to over a hundred terminals in its area, for weather briefings and for the filing of flight plans. It was expected that some 4 to 16 mini-computers (depending on size) would be required for each of the 20 configurations, and that on the order of half a million bytes of software would be needed. The proposed off-the-shelf FSDPS configurations will be benchmarked, together with at least a nucleus of off-the-shelf operating system. The application will be written subsequently in FORTRAN or COBOL (referred to collectively as High Order Language, or "HOL"), at vendor choice. There are several partial prototypes from which one can gain some understanding of the subsequent application.

To establish performance, there are two quite different approaches that we can consider, namely modeling and measurement. In our case the hardware and the operating system cannot be modeled because the buyer does not know in advance what will be bid. However, being off-the-shelf, they can be measured, provided that they can be driven as the intended application will drive them. The application software, on the other hand, cannot be measured because it does not yet exist in suitable form; but by studying the specifications and the partial prototypes the buyer is able to model it. The solution is therefore to model the application, but in the form of a set of programs that can be run in real-time on the system being benchmarked, so that the performance of the system can be measured. The programs are "synthetic" in that their purpose is to impose the modeled resource requirements rather than to perform the application functions. A combination like this of modeling and measurement is not, by itself, new; the challenge here is how to implement it for the case where yet-to-be-designed application software spans a yet-to-be-known distributed configuration of yet-to-be-identified hardware.

Figure 1 shows an overview of the benchmark. It will be seen that the following components interact at benchmark run time. The buyer-provided model of the application, which has been delivered in the vendor's choice of HOL, runs under the hardware and operating system (or nucleus thereof) that are provided off-the-shelf by the vendor. Between them lies a vendor-developed package known as the Vendor Interface Package (VIP), which allows the model to present a vendor-independent interface. The VIP acts as a "sub-executive" or "application executive" to schedule the individual actions of the model, thus enabling the vendor to control the running of the model, so that it can reflect the way his eventual application might best run.

The model is driven in real-time by synthetic messages from a Benchmark Driver, which is a separate processor outside the configuration being benchmarked; and it returns synthetic responses to this driver. The driver in turn obtains the messages from a buyer-provided scenario tape and it delivers each one at the time specified on that tape. It time-stamps the responses and writes them to another tape for later analysis.

Producing the software for this kind of benchmark can present the buyer with a formidable problem. Reference [2] describes the solution to this problem, which was to write all the buyer-provided benchmark software in the same very high order language, and to use an automated code-production system to translate the source into standard HOLs.

3. The Structure of the Model

Clearly, the validity of this benchmarking methodology depends on how well the model replicates the strains on the underlying system that the eventual implementation will impose. The model is written so that the vendor can control the way that it runs on his system, and much of the flexibility for this control lies in the way the model is constructed, so it would be useful first to describe the structure.

3.1 Overview of the Structure

Figure 2 shows a much simplified model as a set of pieces of code, each of which represents a necessary area of the application. Clearly, any eventual software design must include code for each of these areas. Moreover, each transaction that enters the system will need to visit particular areas; and some, at least, of these areas will need to be visited in a specific order. The transactions therefore can be modeled as following specific major routes through these pieces of code. However, the model does not have to stipulate where the code should reside (in which processor, etc.), nor does it specify the relative rates of progress for transactions that are functionally independent of each other. This latitude immediately gives some of the flexibility for representing different design approaches.

Each of the pieces of code models its real-life counterpart by requesting resources (CPU, code-space, I/O, and data areas). With some procurements the sequence of requests could be driven by stochastic algorithms within each piece of code. However, we are considering here the type of application

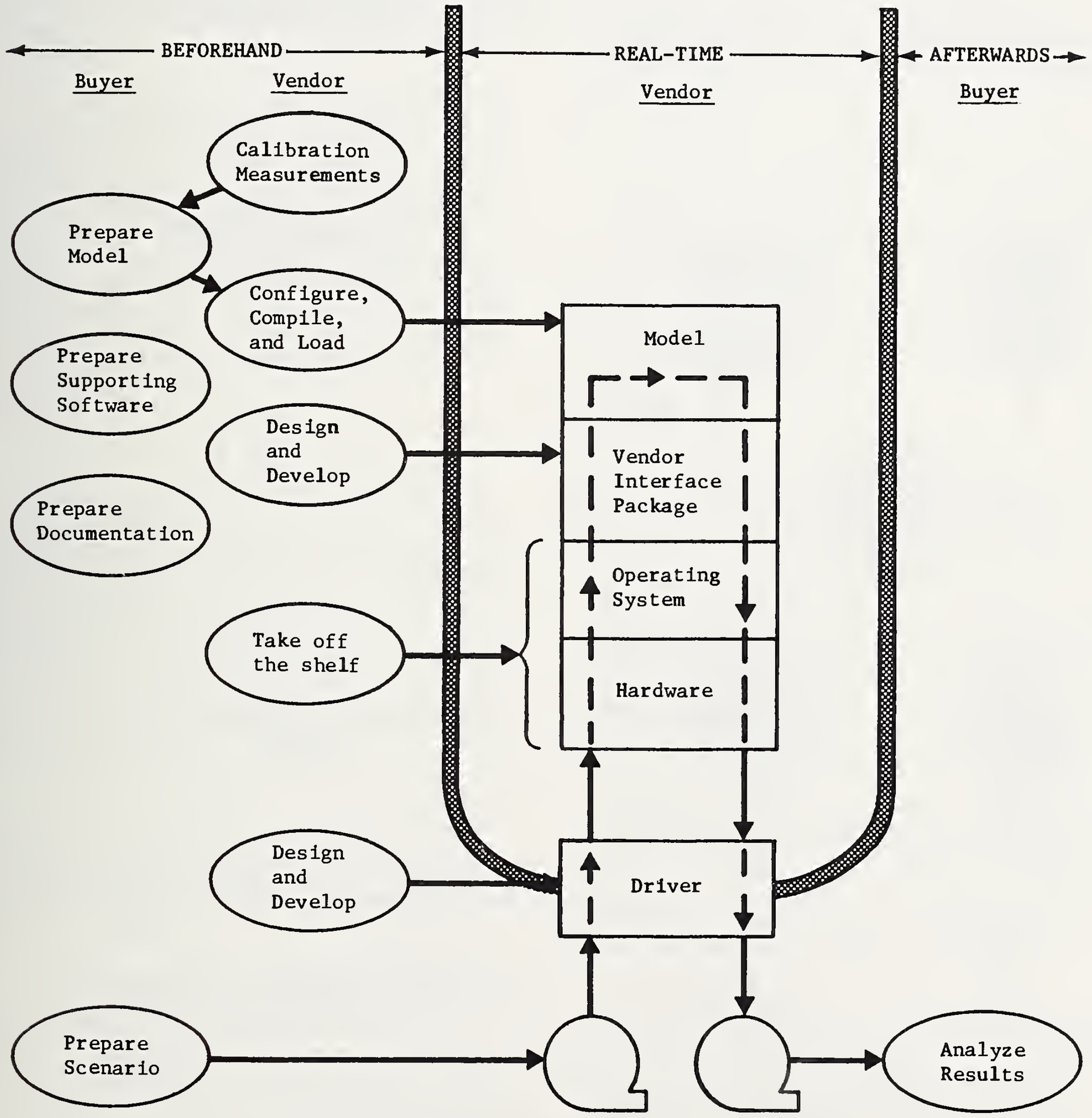


Figure 1. Overview of Benchmark

where the actions of the individual pieces are closely dependent on each other, and where the measurement of interest, namely response time, corresponds to a sequence of actions that may span the entire application. In such a situation, it is simpler to have an input-driven model; hence from a top-level viewpoint the model is seen to be driven by the contents of the synthetic input messages. Each message contains coded data that directs its path through the model and determines the resource requests that are to be generated along the route.

At a finer level of detail one can regard the model and the VIP as a partnership, each driving the other. The VIP is driven by requests from the model, referred to as VIP Service Requests (VSRs). Some of these VSRs request resources, while others indicate which parts of the model are currently eligible to execute. The VIP in turn decides (or lets the operating system decide) when the resource requests shall be honored, and which of the eligible parts of the model shall run next. After servicing each VSR, the VIP initiates the execution of the selected part of the model. The model is entirely dependent on the VIP for all external services; this gives the VIP the opportunity to control the running of the model and thereby tailor it to suit the architecture of the configuration.

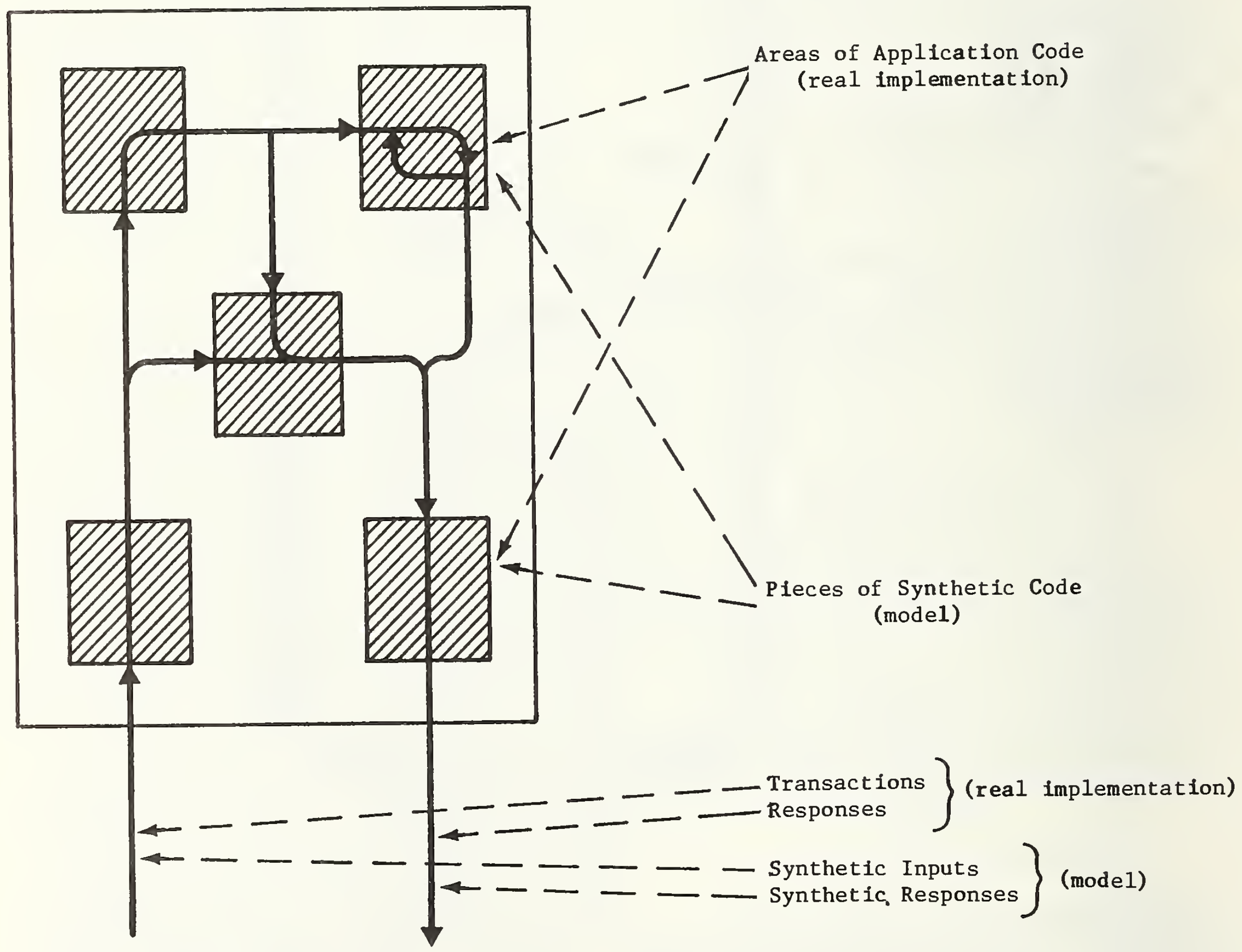


Figure 2. The Model of the Application

It is important that we select the right kind of component with which to build the model. Suppose, for example, that the model were to be expressed explicitly in terms of individual disk transfers. Not only would such a model arbitrarily eliminate vendors proposing other storage media but, more subtly, it would prejudge how the medium is to be used. Clearly we have to find "abstract" components that express what is essential for the application without committing the model to any particular hardware configuration or software design philosophy; and the components must be such that each vendor can implement them for the benchmark in a way that is appropriate to his system.

Four types of abstract components were adopted. The model has to contain code and data areas; and it will be seen below that two types of data area are needed. One more type of component is needed, to express the potential parallelism within the model. Since the implementation of these four components, in common practice, varies from system to system, with little agreement on exact terminology, we have to describe them in abstract terms and leave the reader to relate them to any implementation he is familiar with.

3.2 Code

The functions of the application are decomposed into areas which are modeled as pieces of code referred to as "cells". Each cell is designed to consume a calibrated amount of CPU-time, and to occupy a calibrated amount of memory space. Part of this modeled time and space is taken up controlling the running of the model, the rest being made up by synthetic code. The FAA model uses some 110 cells during the period in which response times are being measured.

In the default case, every transfer of control from one cell to another goes via the VIP. The "GoToCell" VSR provides the VIP with the opportunity both to distribute the cells across processors and to schedule their execution.

While the cell is the unit of code for modeling, the unit of code that is delivered to the vendor for compilation is the "module". Normally, there would be one cell to each module; but, if the vendor decides that excessive use of the "GoToCell" VSR would impose unrealistic overheads, he can request that cells be combined (in any combination of his own choosing) into multi-cell modules. The vendor is free to compile the modules as reentrant, serially reusable, or neither; as resident in Main Store, or brought into memory either deliberately or by demand paging; and to duplicate any module in any number of processors. Apart from some simple and obvious restrictions after an interrupt, the VIP is permitted, whenever it returns control to the model, to select any copy (in any processor) of any module that contains the required cell.

3.3 Natural Flows

Whereas the code of the model is divided into cells, the flow of execution is divided into potentially parallel "Natural Flows" of action. This latter division is the corner-stone of the modeling approach, and it reflects the importance placed on order of execution, both in the understanding and in the implementation of real-time systems. A Natural Flow is defined as a sequence of actions that is driven by data from one or more specific sources, where for any given set of data the sequence is predetermined. The sources of data are individual input ports, the system timer, and other Natural Flows. For example, the sequence for a given terminal is, at the grossest level, the endless loop "read, interpret, do it, reply, read, . . .".

Just as individual programs run asynchronously, and potentially in parallel, in a multi-programming environment, so do the Natural Flows in the model. Natural Flows are not entirely asynchronous, however, for at specific points they deliberately coordinate with each other; this represents the functional coordination constraints that are implicit in the functions to be performed. Data-base discipline provides an example of such a functional coordination constraint, which is modeled by not allowing one Natural Flow to read a specific record while another Natural Flow is writing to it. Other than at these functional coordination points, which are clearly identified in the model, the Natural Flows of the model are independent of each other. In other words, they can be entirely asynchronous one to another except where it is specified otherwise.

In an actual implementation, the Natural Flows will also constrain each other in another way, referred to as implementation constraints: for example, when they compete for the same physical resource. However, these implementation constraints will vary from vendor to vendor, so they have no place in the benchmark model. The model is expressed in terms of Natural

Flows, together with specific functional coordination constraints, because this is a simple and convenient way to express the essential minimum constraints implicit in the specifications. The VIP and the operating system will impose the implementation constraints.

Clearly an actual implementation will not necessarily be structured around the Natural Flows identified in the model, but because the Natural Flows represent the minimum necessary synchronization, they will be present in any implementation, however disguised. In practice, the scheduling of the implementation will be in terms of functions (the Natural Flows will be interspersed frequently with each other), or short-lived tasks (each can map to a phase in the career of a Natural Flow), or permanent, free-running tasks (each can map exactly to a Natural Flow). Thus the Natural Flow is conceptually what many, though by no means all, mean by tasks or processes; but a new term helps to avoid confusion when the mapping is not one-to-one.

In the FAA model there are some 150 permanent Natural Flows. Most of them are dedicated one-to-one to terminals or other logical input sources, while the remaining few represent service operations that need not be performed in-line with the requesting Natural Flow (for example, a service Natural Flow oversees the spooled output to a device used by many Natural Flows). As will be seen, the vendor can also define temporary Natural Flows in places where further parallelism might benefit him (for example, to stack up a queue of reads from disk).

3.4 Permanent Memory

Two properties of a data area are of interest here: whether it must survive system failures, and when it is to be accessed. Permanent Memory models data areas that must be recoverable after a system failure but need not be accessible to the individual instructions from the CPU, whereas Working Space models those that must be directly accessible to the CPU but need not survive failures. Deliberate copying between the two is modeled. However, if Permanent Memory on a particular system is accessible to the CPU, and it can be shown that this does not impair the recoverability of the material, then the modeled copies can be omitted on that system.

Modeling for recovery after system failure was regarded as important in the case of the FAA model, hence the emphasis on copying to Permanent Memory. Recovery itself is not being benchmarked, but the model imposes the run-time strains caused by the laying of adequate recovery trails.

The use of Permanent Memory is divided into "files", which are subdivided into "records". This represents a logical division of the data, the record being the unit of data that the model requests. These logical divisions need not correspond to the organization of data that will be used subsequently, in the actual implementation, nor to the way that the material is physically arranged during the benchmark run.

The data base in the FAA model consists of some 45 files. Data base requirements for this application are relatively simple, so a logical division into files and records is both natural, and unlikely to bias the model in favor of any particular style of implementation. This will not be true in all procurements; in some, the expression of the data base in modeling terms would need considerable care.

3.5 Working Space

Working Space models the working areas that are not local to a cell. For example, in the case of FORTRAN it models COMMON, together with any internal arrays that hold data that is passed to and from other cells. It can be implemented as permanently or dynamically allocated space either in Main Store or in Backing Store (disk, etc.); in the latter case it must be brought into Main Store when it is being used.

Each piece of Working Space is "attached" to one or more Natural Flows; it is never regarded as associated with a cell. Thus, when several Natural Flows request sole access to the same Working Space, it is the Natural Flows that are blocked, not the cells from which the requests were issued. Typically, however, a Working Space remains associated with the Natural Flow that requested its creation in the first place, and is never attached to any other Natural Flow. For example, each user-terminal has a Natural Flow dedicated to it, and a Working Space attached to this Natural Flow models the administrative data that pertains to the status of that terminal. This Working Space is visible to any cell (or instance of execution of that cell, in the reentrant case) while but only while that Natural Flow is using the cell (or instance of execution). Thus the continuity of a

Natural Flow (many of which can use the same cell) is defined by the presence of the Working Spaces that are currently attached to it. In this sense the Natural Flow is a flow of data being operated upon by successive cells of code.

Though the model dictates when Working Spaces are to be allocated, and what size they are to be, the VIP has considerable freedom as to how they are implemented. In particular, the model informs the VIP in advance whether it will alter the data, or only read it, or neither for the time being. Moreover, when Working Spaces are to be visible to the model, the VIP is free to decide where in the model's address space they shall appear. However, one has to be careful not to let one's concern for generality lead to the VIP itself imposing an unrealistic load on the system. Therefore an important consideration when planning the details is to allow the VIP to remain simple. For example, the properties of excessive and of dynamically changing size, which might be awkward to implement on some systems, are confined to one Working Space per Natural Flow, leaving the majority of Working Spaces relatively simple to allocate efficiently.

3.6 The Interface with the VIP

Since the VIP plays the role of operating system to the model, one would expect the VSRs to have the appearance of procedure calls. However, it was found that reversing the normal call-return pattern gives the VIP more flexibility to influence the course of execution. Thus the model appears to the VIP as a set of procedures: the VIP calls the model, not the other way around. A VSR is a return from a procedure call, while the completion of that VSR is the occasion of a fresh procedure call to the model. The return links for procedure calls between cells of the model are handled by the model. The result of these arrangements is to make it simple for the VIP to switch Natural Flows whenever it is called by a VSR. It does not have to disentangle itself from stacks of return-links.

Basically the task of the VIP is to administer the status of each Natural Flow. For each, separately, it has to implement the continuing loop "call module, service VSR, call module, . . .". Except at functional coordination points, these loops are independent of each other, and there is great variety in the way they can be overlapped in real-time. How the 150 or so loops are superimposed represents an important part of making the model behave in the kind of way that one would expect from the vendor's eventual implementation.

Since each vendor designs his own VIP, the structure of each VIP is vendor-dependent, thus providing part of the flexibility for mapping the model to different kinds of operating-system environments. In particular, a vendor can implement his VIP as several cooperating copies, or instances: one per CPU, for example, or one per multi-CPU processor.

4. The Approach to Modeling

Having introduced suitable building-blocks for modeling, we now consider how to use them to model a particular application.

4.1 The Objective

The methodology developed here consists of a set of techniques, which should not be confused with the policy that will determine their use. The same tools can be used, albeit in slightly different ways, to serve a variety of purposes. If the techniques described here are being used to provide a design monitoring tool for use with a single vendor, for example, then clearly one will try to model that vendor's software design as closely as is practicable. On the other hand, if the purpose is to compare competing vendors, none of whom have yet contracted to design the software, then one's objective is different.

This paper addresses the case where the purpose of the benchmark is to measure the capacity of the hardware and the operating system, and not the vendor's skill at designing and programming. In other words, we assume here that for benchmarking purposes these skills need not be considered as dominant factors. Our object in modeling the application is to impose approximately the same kind of load on the system as would a design that is appropriate for the architecture of that system. We do not necessarily have to model in detail a design that does not yet exist; we only have to postulate an appropriate and competent design.

4.2 The General Approach

The general approach to modeling is to consider the functional requirements of the application in a top-down manner. In this way, one can identify the major flows of control (the Natural Flows), and the major areas for which code will be required. Similarly one can identify the major data areas.

Following through on the top-down approach, in the case of code, the major areas are successively decomposed into smaller areas. At each level of decomposition, the areas of function can be represented as equivalent units of model-code; and the decomposition continues until we arrive at a set of code-units that can run on each vendor's system, and which each vendor can configure to represent the way he intends to drive his hardware. These code-units are the cells of the model.

Figure 3 is adapted from the source listing of the FAA model, and shows one of the cells that model a weather briefing. This cell, which is used by the Natural Flows dedicated to user terminals, explicitly models the use of code-space and CPU-usage, and then returns to the cell that called it. Implied at this level of presentation are references to the currently attached Working Spaces, and a VSR "GoToCell" at the end. The code-production system referred to earlier automatically translates cells in this very high level source into HOL units (procedures, subroutines, etc., as appropriate) for delivery to vendors.

```
cell WindComputations

     CodeLength 200 CodeUnits
     UseCPU 5000 CPUUnits
     EndOfCell
```

Figure 3. A Simple Cell (adapted from FAA model)

Some further modeling is sometimes appropriate, within the cell, for in many cases one can identify a necessary order of internal operations. Figure 4 shows an excerpt from one of the cells that model the analysis, or "route conversion," of a proposed flight plan and it illustrates the operations within that cell. VSRs are identified by comments in the left-hand margin of the form "V10]" and references (usually procedure calls) to other cells of the model by comments of the form "p10]." Notice how the behavior of the model is directed by parameters carried by the transactions in the input scenario.

```
            CodeLength 50 CodeUnits

            unless ConvertingRouteElementsInParallel,
V2]                     AcquireWorkingSpace  WorkArea#2, WorkAreaForRouteConversion words

            with ThisElement, from 1, to ScenarioDirected#RouteElements
               if ConvertingRouteElementsInParallel,
V20]p31]                     SpawnTemporarily  StartingAt  RouteElementsInParallel,
                                          WithCopyOf WorkArea#1, PlusDataWord ThisElement

               else
P32]              ConvertElement  ThisElement
                  UseCPU  500 CPUUnits
                  end else
               end with

            if ConvertingRouteElementsInParallel
               loop  ScenarioDirected#RouteElements  times
V17]              AwaitProd  receiving RecordArea
                  UseCPU  500 CPUUnits
V5]               ReleaseWorkingSpace RecordArea
                  end loop
               end if
```

Figure 4. Modeling within a Cell (adapted from FAA model)

It will be noticed that the model takes the form of a skeleton design that is fleshed in by synthetic code instead of by the detailed logic of the implementation. The approach to designing the model is indeed somewhat similar to that for designing the real implementation. However, there is a difference in emphasis. The objective is not a single design, as such, but an object that encompasses all reasonable ways that the design could be done. Not only must one pause at each point to consider the variety of possible design approaches, but one must consider the question in reference to a wide range of machine architectures. If the buyer is willing to contract to vendors who offer to advance the state of the art, it would indeed be difficult to consider all possible approaches, for some of them may not be invented yet. Fortunately, most procurements are sized on the assumption that only well-established design approaches will be used.

The top-down approach implies ignoring detail and concentrating on the essentials of the broad picture. The functional specifications will no doubt dwell heavily on details such as those of the man-machine interface. These are irrelevent to sizing, for one is looking for the broad movement of data, and the major requirements for code and CPU. As an example in the FAA application, a number of message types are specified as being input by users. For the corresponding model, a number of input types are used, each of which represents the transactions that lead to a particular pattern of internal loads. The classification of transactions into benchmark input types is therefore resource-oriented, as opposed to the function-oriented classification into message types in the specifications; and the correspondence between specified message types and benchmark input types is not necessarily one-to-one.

4.3 Vendor Options

Whenever one descends a level of detail, and divides a single unit into several pieces, one is making design decisions. Sooner or later, one will make decisions that are biased towards or against one kind of architecture, or one kind of design approach. At such points, one has to consider whether the various possibilities are allowable in a sizing exercise; if two or more possibilities are indeed allowable, then a "vendor option" is inserted into the model at that point. The model is programmed to behave in any of several ways, depending on the setting of a switch. Each vendor elects how the switch will be set in his version, and the code-production system mentioned earlier takes care of translating his setting into the HOL that he receives.

In the FAA application, a simple example of an option is provided by incoming data from radar sites, which can be processed by hardware or by software. A vendor option allows for the simulation of a small or a large amount of CPU-use, and the vendor who proposes the extra hardware may have the switch set to reflect low CPU-use. Other options allow for spawning (creating) the temporary Natural Flows mentioned earlier, at points where extra parallelism might better reflect the power of some architectures. Figure 4 includes an example of this: if the option is taken, then a Natural Flow obeying the code in this cell will spawn a set of temporary Natural Flows. Having done so, the "parent" Natural Flow will wait in this cell for each offspring to return the results of its calculations. The offspring does this with a coordination signal that carries with it a Working Space called "RecordArea."

4.4 Miscellaneous Considerations

In some areas of the model, a decision may have to be made as to which side of the VIP-model interface a particular feature should lie. Should it be modeled, or should the VIP or the operating system be expected to provide it? In the FAA case, the question arose with data base management, with the spooling of output to other computers, and with the scheduling of events where the schedule must survive system failures. The answer depends on two factors: whether it is reasonable to expect all vendors to have the feature available off-the-shelf, and how reliable the off-the-shelf feature is likely to be. If the buyer is unwilling to trust the reliability of what the vendors may offer at benchmark time, then it is better to model that feature, for though the benchmark is not a test of functionality per se, it does seek to establish that there is enough hardware to support proper implementations for all functions.

The VIP has to schedule the model intelligently, as would the "application executive" in the real implementation. Intelligent scheduling requires a cooperative interface between application and executive; so a consideration when designing the model is to arrive at a VIP-model interface that is helpful to the VIP in this respect.

Various practical considerations make it an advantage to keep the logic in the model simple; hence as much as possible of the decision making should be placed in the program that runs beforehand, off-line, producing the scenario load-tape. The parameters in the load tape can thus reflect previously-run statistical programs and direct the model in simple terms such as "read 5 records from file 7". Only where the modeled input message is too short to contain the instructions must the model make its own decisions.

5. Quantifying the Model

So far we have considered the structure of the model. The components in this structure now have to be sized. The code, the data areas, and the CPU usage in the model each represent specific functions, so each needs to be quantified. The quantification of code-length and CPU-usage raises interesting questions that we can briefly review here. It is intended that these questions will be discussed further in a later paper.

Since the real code of the eventual application is not yet available, we represent it in the first instance by a Sample Program composed of HOL code that is typical of the application code that will have to be written later. In the FAA case it was decided that one program (that is, one for each allowable HOL) could reasonably typify the whole of the application, in respect of both code-length and CPU-usage. In other applications one might consider using a blend of Sample Programs, where one Sample Program, for example, represents typical computation, and another the character-manipulation, and so on.

5.1 Sizing the Model

This step of the new benchmarking methodology, namely the sizing of the model, corresponds closely in many ways to traditional methods of sizing new computers. Traditionally, the estimated requirements for code space and CPU, etc., are the end-product of sizing. The new methodology also uses these estimates, but it takes them as input data and places them in the system context where they belong. With earlier generations of computers, one assumed programming of average competence, and specified to the vendor the raw power that one wished to have available for the proposed applications. With the new methodology we do the same, but at the level of the individual cell. We ignore the vendor's claims about his programmers, and specify the raw power that we wish to have effectively available for the function that each individual cell represents. The system itself is then used to extrapolate from cell power to system power. Cell sizing involves estimating and here, as before, the proportion of guesswork to scientific derivation depends on the supporting evidence that can be brought to bear.

It is important to be clear what each sizing quantity represents. Code-length sizing must represent several elements: actual code, constant data, and those variables that are not modeled as Working Spaces. It should also distinguish between data locations that are written to and those that are not. The CPU-usage must cover actual instruction-obeying, and waits (such as the demand-loading of cache memories) where the system cannot make use of the idle time. However, it must not include the time for any I/O, since the effect of these waits is taken care of in the overall benchmarking approach. For example, on a paged machine we wish to reproduce, per cell, address spaces that correspond to those for a competent real implementation, and with the same use of the CPU. How these address spaces are maneuvered, and how the CPU is allocated to them, is a function of the underlying system, and this applies equally to model and to real implementation. Reference [3] contains an interesting discussion of some of the considerations that apply to this step of the benchmark.

The sizing of the model has to be presented in vendor-independent terms. It would be tempting to use the Sample Program as the unit of measurement (both for code-length and for CPU-usage). A cell which is sized as 2 units of space and 3 units of CPU-usage, for example, would then be delivered to vendors containing 2 copies of the Sample Program, and logic to initiate 3 executions of a Sample Program. Unfortunately, we cannot do this. Any Sample Program which makes a serious attempt to be typical is most unlikely to be small, so one would be calibrating cells in fractions of a unit. Another difficulty will appear shortly.

We therefore have to take a more involved approach. We start by arbitrarily choosing a reference machine, and we define vendor-independent units in terms of it (for example 100 microseconds, and 100 bytes, on that machine). To the vendor these are abstract units: see Figures 3 and 4 (where the numbers are for illustration only). To convert time and space in abstract units, to time and space on a particular vendor's system, we run the Sample Program on that system. By comparing time and space, there, with time and space on the reference machine, we obtain a multiplying factor for each that is valid for typical application code. Thus we can derive the actual time and space that should be set aside for each cell on the vendor's machine. Notice that this procedure takes into account the efficiency of the code produced by his compiler.

5.2 Reproducing the Sizing

We now have to arrange that the HOL model that is delivered to each vendor will compile into code that uses this required time and space. The delivered model contains two types of code: benchmark machinery (the logic of the benchmark) and synthetic padding, which between them must use the correct time and space. It is the presence of the benchmark machinery (which is unlikely to consist of typical application code) that rules out using the Sample Package for the synthetic padding. We need a common basis for comparing application code with machinery code; and actual time and space on the vendor's system provides the most satisfactory comparison. Time and space for the benchmark machinery are therefore measured on each vendor's system and subtracted from the target sizes for the cells, to give the time and space that must be made up by the padding.

The synthetic padding consists of HOL "kernels" (that is, small units of code designed for the express purpose of reproducing CPU-usage or code-length, see [4]); and the time and space that these will use, when compiled, is also measured for each competing system. The kernels must have the following properties. They must not be optimized out by the vendor's compiler; in those cells for which it has been possible to postulate a pattern of address-referencing, they must make references according to that pattern; they must enable one to reproduce whatever proportion of code-space, fixed data, and variable data that has been determined for each cell; and either the effect of cache memory must be the same when they are measured and when they are used, or the effect when they are used must be determinable. Once the kernels have been measured on the vendor's hardware, one can calculate how many of each must be inserted into each cell and how many times each should be exercised. The automated code-production system takes care of the calculations and the insertion of kernels.

6. Summary

The new benchmarking methodology measures the performance of hardware and operating system by running on it a buyer-provided model of the intended application. The model is input-driven, and consists of synthetic programs that the vendor can distribute and schedule as he wishes. To do this he is allowed to provide a package that interfaces the model to his operating system. The two principal problem areas concern the modeling of the application (the subject of this paper) and the production of versions to send to vendors (which was solved by using an automated code-production system).

The application has to be modeled in a manner that is flexible enough to represent design approaches for a wide range of architectures that will not be identified until later. The solution is to identify abstract components that can be implemented by the vendor in vendor-dependent ways; and to design the model in terms of these components, with a top-down approach to the functions of the application. Wherever alternative design approaches appear valid, and the choice of approach can significantly affect the pattern of resource requirements, a vendor option is built into the model. Having used these techniques to model one particular application, one's subjective reaction is that they are powerful and could be used for a variety of applications. It would be helpful to gain more experience with them so that the breadth of their applicability can be explored.

References

[1] Spooner, C. R., Benchmarking Interactive Systems, proc. Summer Computer Simulation Conference, Toronto, July 1979, pp 791-798.

[2] Spooner, C. R., Benchmarking Interactive Systems: Producing the Software, to be published in proc. Conference on Simulation, Measurement and Modeling of Computer Systems, Boulder, August 1979.

[3] Williams, J. N., Construction and Use of a General Purpose Synthetic Program for an Interactive Benchmark on Demand Paged Systems, proc, 1977 Annual Conference, ACM, pp 459-465.

[4] Buchholtz, W., A Synthetic Job for Measuring System Performance, IBM Systems Journal, Vol 8, No 4, pp 309-318, 1969.

CPEUG 79

Installation Management

INSTALLATION MANAGEMENT

Carol B. Wilson

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

The principal goal of any ADP installation should be to serve its organization by meeting the computing needs of its user population at the least cost. Installation management encompasses all of the functions which an ADP installation manager performs in optimizing his installation resources to the user workload to meet this goal. A computer performance management program aids the manager in this task by providing quantitative data for many of the installation management functions in an orderly and timely manner. Computer performance evaluation supplies the tools and techniques needed to capture the data and to optimize the installation resources.

In this session, we have papers concerned with several areas in which computer performance management support installation management. The first and last presentations are concerned with an overall view of CPM with respect to the needs of the manager. The second paper deals with an organization's efforts to plan, provide, and ensure specific user service levels. The third paper discusses how measures for control, planning, and accountability were designed into a heterogenous distributed processing environment.

Methodology for Establishing Computer Performance Management presents a top-down approach to designing a CPM program to meet the needs of ADP managers in a heterogenous environment. Technical considerations are delayed until objectives are defined which lead to the specific needs and capabilities for the CPM program being developed. A phased implementation approach is then used where available tools are installed first and then the gaps between their capabilities and the needs of the CPM program are filled by locally developed means.

Naval Laboratories - User Quality of Service Standards discusses the activities of the Naval Laboratory Computing Committee, a group of Naval R and D laboratories, to define measures of service for availability, turnaround time for various sizes of jobs, and interactive processing that could be used by all of their facilities. The sizes of jobs for turnaround time are calibrated by a synthetic benchmark and are related to the cost of processing. The service levels for the interactive measures are validated by a standard scenario.

Network Control and Computer Systems Accountability describes work done to design a terminal interface unit for each node of a multi-vendor, "after the fact" network. The objectives of the design included making provisions for controlling the growth of the network, for ensuring the security of processing, for accountability of use (including small, special-purpose computers with no accounting systems of their own), for identifying cross-system usage patterns, and for providing performance data to automatically generate utilization and planning reports.

An Informational Model for the ADP Manager is an oral presentation concerning the work underway at the National Bureau of Standards to determine the informational requirements of the ADP manager which could be supported by a CPM program. The talk discusses installation management models which are being used to develop a list of measures. A workshop session on this same topic will be held later in the conference.

METHODOLOGY FOR ESTABLISHING A COMPUTER PERFORMANCE MANAGEMENT SYSTEM: A CASE STUDY

Geoffrey H. Goodman

International Computing Company
Dayton, Ohio 45431

The first steps in establishing a computer performance management system are critical to success. Lack of attention to organizational and management needs are as likely to lead to failure as lack of attention to technical details. This paper presents a methodological framework which is intended to assist those who are establishing a CPM capability. Techniques are suggested to enable the individual to assess the needs of his or her own organization, and to design a performance management system to meet those needs.

Key words: Performance Management System (PMS); needs; capabilities; methodology.

1. Introduction

The methodology described below was developed as part of a contract to design a computer performance management system (PMS) for a Federal government organization. The specifications of this PMS are uniquely suited to the computer environment for which it was designed. However, the methodology which was used could be applied in nearly any organization which was contemplating establishing a PMS.

The mission of our client organization is to perform engineering analysis of military hardware, using data from several external organizations. Different directorates are responsible for different sorts of analysis, and the types of data processing support provided to them varies accordingly. Some organizational elements are responsible for scientific and engineering analysis which relies heavily upon on-line computational services, while others concerned with reduction of raw data and production of finished reports are heavy users of batch processing service. Until recently, each engineering directorate owned and operated its own computers. In all, twenty-four computers of various manufacturers are in operation, ranging from UNIVAC 1110 and IBM 360-65 mainframes to special purpose microcomputers. In 1978, a Directorate of Data Services was formed and was given responsibility for all data processing resources in the organization. As the new Directorate of Data Services began the task of consolidating its resources, the need for some form of centralized performance monitoring system was recognized. We received the task of designing such a system and developing a plan for its implementation.

2. Clarifying Basic Objectives

The requirement which we received called for the design of a "performance measurement" system. While this document expressed a requirement to measure and report certain performance characteristics, the basic objectives of the system were not stated. Our first step was to discuss the requirement with our primary point of contact, the Systems Engineering Division, to determine what the objectives were.

Computer performance evaluation was not a new concept to the organization, we were told. Since as early as 1974 performance measurements had been carried out in order to identify and correct performance deficiencies of individual computers. The individual CPE efforts associated with the larger mainframes were generally perceived as having been successful, and no serious problems in the efficiency of individual computers were identified to us. However, because of the previous decentralization of responsibility for computer systems, there was no management information system which provided a clear grasp of the data processing system as a whole. Top level ADP management was uncomfortable with the dearth of historical data and the lack of appropriate measurements to support decisions concerning allocation and acquisition of ADP resources. Once this need had been properly articulated, it was apparent that a new management information capability would have to be designed, while the existing CPE capability would still be needed to support the requirement for tuning individual systems. The objective of our task was now understood to be the design of a performance management system (PMS) which would satisfy the needs of ADP managers.

3. Establishing Responsibility for the Performance Management System

Establishing organizational responsibility for a performance management system is one of the critical decisions which must be made. In this case, responsibility for CPM had already been placed, and was therefore beyond the scope of our analysis. Responsibility had been placed with the Quality Assurance Branch of the Systems Engineering Division. This branch is concerned with establishing standards and procedures for ADP systems, and for providing management information concerning the use of ADP resources. The Quality Assurance Branch therefore was responsible for working with us and monitoring our efforts. Although we had no control over the decision, we feel that the decision to place responsibility for the PMS in the hands of the Quality Assurance Branch is a proper one. It is essential that the organizational element responsible for the PMS be more attuned to the needs of management than to strictly technical factors of systems tuning. Nonetheless, the decision to place responsibility for the PMS with the Quality Assurance Branch creates the potential for coordination problems. The Systems Software Branch had had, and maintains, the responsibility for CPE efforts required for system tuning. The Computer Operations Branch actually runs the software which is used for performance measurement. It is clear that creating and maintaining the proper communications and spirit of cooperation between these three organization elements is critical to the establishment of the PMS.

4. Gaining Management Support for a Top-Down Design Approach

Because of the technical complexity of designing a CPM system for such a heterogeneous computer environment and the necessity of showing some progress in a relatively short time, there was some temptation to immediately begin to tackle technical aspects of the problem. Since CPE tools were already available, one possible course of action was to begin to patch those together to somehow produce the management information which was needed. We soon realized that premature involvement in technical details would endanger the success of our entire effort.

We felt that the only effective approach to the design problem would be to begin with a detailed assessment of the information requirements for such a system. In order to do this, we briefed the Technical Director and Director of the Directorate of Data Services on our recommended approach. In the briefing, we characterized this approach as a "top-down" design, which would begin with definition of requirements and work toward an implementable design which would fulfill them. The alternative "bottom-up" approach, we cautioned, could result in a design which was technically workable but which failed to address the (still undefined) management information requirements of the organization. We then presented our plans to survey the personnel who would be using the CPM information, in order to define the objectives of the PMS in detail. Gaining management support for this approach proved beneficial in two ways: it relieved pressure to begin work on technical details before requirements were assessed; and it provided us full access to managers within the Directorate of Data Services who were able to provide insights on their requirements for the PMS.

5. Separating Needs From Capabilities

Our plan to survey personnel who would use the PMS became our primary means of determining requirements, or needs. We decided that during the early part of our analysis, needs and capabilities should be separated. If discussion of needs and capabilities was allowed to intermingle, the result would be a statement of needs which we knew could be satisfied. Valid requirements which could not immediately be matched with a capability would perhaps not be stated by the people we surveyed. By separating analysis of needs and capabilities we also lessened the risk that we would ignore stated needs because we had a preconceived idea of a capability to be provided. Another advantage to this approach is that the task could be broken into two independent efforts. The person or people assessing the requirements for the PMS need not necessarily be the same people to evaluate the available capabilities. Although we did not employ this kind of division of effort, it was useful to be able to jump from one portion of the task to the other when progress on the first portion was unavoidably slowed down.

6. Designing and Conducting a Survey to Determine CPM Needs

The challenge of surveying managers on their CPM needs is to get them to provide specific ideas and opinions on the measurements to be used. We decided that the best method of doing this was to create a "straw man" set of CPM measures, and encourage additional ideas and constructive criticism from the people we surveyed. We began by collecting a list of different performance measures from various sources.[1] Our final list contained 101 different measures. Similar types of measures were aggregated into categories, without any preconceived scheme of what categories ought to emerge from the process. The categories thus created were Timeliness, Accuracy, Cost, Reliability, Utilization and Productivity.[2] Representative measures which we felt would be useful were selected for each category. In all, 31 measures were selected for use in the CPM survey. The survey was divided into six parts, one for each category of measure.

[1] Two useful sources of CPM measures were General Services Administration, Management Guidance for Developing and Installing an ADP Performance Management Program (2) and Sardo and Gotthardt, Meaningful Data Processing Performance Measures for Management (3)

[2] Our efforts to categorize performance measures were influenced by the work of Phillip Howard and Barry Stevens. Therefore, the use of their categories (timeliness, accuracy, cost, and reliability), while not preconceived, was not coincidental. The categories of utilization and productivity were created to categorize other measures which we encountered.

Each section was introduced by a short statement of the purpose of the measurement category (see Figure 1 for an example).

RELIABILITY

Purpose: To determine the degree to which processing time is lost as a result of ADP equipment failure. Some measures identify the specific hardware responsible for the failure.

Figure 1. Introduction to a Performance Management Category

Following this, the individual measures were presented (see Figure 2). Under the name

RELIABILITY MEASURES

MACHINE/DEVICE RELIABILITY INDEX BY DEVICE/COMPONENT BY MACHINE	$100\% \times \frac{\text{Scheduled Time-Unsked Downtime}}{\text{Scheduled Time}}$
ON-LINE RELIABILITY INDEX BY MACHINE	$100\% \times \frac{\text{Scheduled On-Line Time-Unsked Downtime}}{\text{Scheduled On-Line Time}}$
DOWN TIME(UNSKED) BY DEVICE/COMPONENT BY MACHINE	Σ Unscheduled Down-Time During Period
SITE FAILURES BY MACHINE	Nr. of failures due to environmental problems, (e.g., power, air conditioning) during the period
SITE FAILURE DOWN-TIME BY MACHINE	Σ Time down due to environmental problems during the period
BOTTOM QUARTILE ADPE BY DEVICE/COMPONENT	Identification of ADPE which has been in the bottom 25% in reliability (based on downtime) for more than one consecutive period
MEAN TIME TO FAILURE BY MACHINE	$\Sigma \frac{\text{Time Operable Before Failure}}{\text{Nr. of Failures During Period}}$

Figure 2. Examples of Measures Presented in the CPM Survey

of each measure, some idea of how it is to be applied is provided. For example, data may be gathered at the device/component level, and aggregated at the machine (individual computer system) level. Opposite the name of each measure is a brief description of the derivation of each measure. Since these measures were chosen to stimulate discussion, only simple mathematical expressions were used. Where managers were interested in more sophisticated measures, they indicated their needs during the discussion. Ample time was provided to discuss the CPM measures. Interviews ranged from 45 minutes to 1½ hours in length.

To provide some structure to the survey we asked each person interviewed to provide his/her general impression of each of the measures proposed, in one of the following forms:

1. Interviewee does not feel that this measure should be used (because it is misleading, inaccurate, etc.).

2. Measure may be useful for others, but does not meet this interviewee's needs.

3. Interviewee presently uses this measure or its functional equivalent.

4. Interviewee would like to use this measure, but the capability is unavailable to him.

This technique was used to attempt to discover the specific information needs of the ADP managers we interviewed. We did not attempt to identify a single set of measurements which would be equally useful to managers having very different responsibilities (e.g., software development, computer operations and requirements analysis). Instead we looked for patterns of needs which would suggest the structure of a system to provide different groups of management reports to different users.

7. Documenting CPM Needs

In our experience, the process of documenting the performance management information needs of the organization has proved to be one of the most important steps in designing a PMS. The "needs" end of the two-pronged needs/capability design approach culminates in a "CPM Needs Document." The document which we produced contained three major sections: the assumptions we had made, the information needs identified, and the design requirements for a system to fulfill these needs. After any omissions or errors have been corrected, this document becomes the "design target" for the PMS.

7.1 Design Assumptions and Constraints

The first section of our CPM Needs Document identified the assumptions and constraints which we would apply in designing the PMS. The first assumption was that the objective of the system was to provide management information to support allocation of and planning for computer systems, rather than technical information needed for system tuning. We also made the assumption, on the grounds of cost and development time, that commercially-available software would be used wherever possible in preference to custom designed software. We identified the constraint that our initial CPM effort would measure only currently operational systems (i.e. measurement of systems under development was to be deferred). Finally we made the very important assumption that there was a commitment to "do the do-able." Given a spectrum of measurement techniques which range from rather simple to extremely difficult and costly, we assumed that the organization would maintain a commitment to implementing those techniques which were feasible to them.

7.2 Findings of the CPM Survey

The second section of the CPM Needs Document summarized the findings of the CPM survey. The first six parts of this section each summarized the needs expressed by the interviewees in each of the six categories of measurements. An assessment of the value of different sorts of measures, including especially those suggested by the interviewees, was provided for each category. In the seventh part of this section, different patterns of needs were analyzed and described. Our findings were summarized in the matrix shown in Figure 3. Top ADP management expressed an interest in measures of utilization which could be used for capacity planning purposes. High interest was also expressed in the most visible category of user service measures -- timeliness. Less interest than we had anticipated was expressed by ADP management in measures of cost. We felt that this was because ADP management did not wish to embark on a cost charge-back scheme at this time, and preferred that we did not pursue this objective to the detriment of other performance measures.

Other parts of the ADP organization

having similar CPM needs were grouped together. The Computer Operations Branch and Operating Systems Software Branch expressed similar interest in categories of measures which reflected how effectively they were accomplishing the task of keeping the system in operation, and at a high rate of throughput -- the categories of timeliness, accuracy, reliability and utilization.

	TIMELINESS	ACCURACY	COST	RELIABILITY	UTILIZATION	PRODUCTIVITY
ADP Management	X				X	
Systems Engineering	X	X	X		X	
Quality Assurance	X	X			X	
Requirements			X	X	X	
Computer Operations	X	X	X	X	X	
Systems Software	X	X		X	X	
Operations Support	X	X			X	
Applications Software	X	X			X	

X = significant utility perceived

Figure 3. Perceived Usefulness of Performance Measurement Categories

The two branches of the Systems Engineering Division, the Requirements Branch and Quality Assurance Branch had fairly similar requirements. But although both branches were quite interested in "utilization" we found that they were looking at it from different perspectives. The Quality Assurance Branch tended to look at utilization from the standpoint of resources consumed by a particular application -- a resource management perspective. The Requirements Branch was more concerned about utilization from a hardware component or machine standpoint -- a capacity planning perspective.

The Applications Software and Operations Support Branches expressed interests in performance data which reflected their roles in measures of timeliness, accuracy, and utilization (from the standpoint of resources consumed by a specific application). The data system users outside the Directorate of Data Services were not included in the CPM survey at this time. If they had been, we would expect their needs to be much like those expressed by the Applications Software and Operations Support Branches.

We were surprised to find out that none of the interviewees expressed a strong interest in measures of productivity, since measurements in this category provide the types of data typically used by managers to guage the efficiency of their organizations. Further discussion in this area indicated that most managers already had information which they considered necessary to track the productivity of their organizations, and they were wary of establishing new measures in the context of a computer performance management system.

7.3 Design Requirements

In the third section of the CPM Needs Document we stated the requirements of a PMS to fulfill the needs which had been identified. These requirements were stated in a general manner, but with sufficient specificity to provide objectives for design of the PMS. We identified six requirements to be met by the PMS.

1. Objective-directed Performance Standards.
2. Management by Control Limits
3. Historical and Trend Analysis
4. Traceability of Resources
5. Cross-system Comparability
6. Flexible Report Generation.

The first two design requirements were general enough in nature to accomodate nearly any kind of performance measurement capability. The last four requirements, which related more directly to needs stated during the CPM survey, began to suggest the use of certain types of CPM tools in preference to others.

7.3.1 Objective-directed Performance Standards

By objective-directed performance standards we meant standards of service to be provided to the user of the ADP system. Not all the categories we proposed in the CPM survey are equally applicable to measuring service provided to the user. The categories of Utilization and Productivity are more process-directed. Our view is that measures of utilization and productivity are useful for capacity planning or cost analysis, but that objective-directed (or service-oriented) measures are more useful as performance standards.[3]

7.3.2 Management by Control Limits

Our second requirement was for management by control limits. This means that for every performance standard, a minimum acceptable performance is established as the preliminary control limit. When the preliminary control limit is breached, it provides management with some early warning to allow action to be taken before the critical control limit is violated.(1) Most of the people interviewed felt that establishing control limits was a good idea, but some expressed doubt about whether we would succeed in establishing them through negotiated agreement between the user organization and the ADP organization (the way it should ideally be accomplished).

7.3.3 Historical and Trend Analysis

Most of the interviewees expressed a lot of interest in our third requirement, historical and trend analysis. The organization had suffered in the past for lack of adequate information on the performance history of its systems. Adequate answers to increasingly frequent questions from higher echelons of the Federal government were sometimes difficult to come by. In addition to the need for historical data to answer questions from management, such data is needed in order to carry out the objective of management by control limits. Without a historical data base it is difficult to assess performance trends relative to established control limits.

7.3.4 Traceability of Resources

Most of the interviewees in the CPM survey expressed an interest in being able to trace computer resources consumed back to a specific organizational element, to a functional area (which might be part of an organizational element or an aggregate of elements), or to a particular computer application. Our statement of a requirement for "traceability of resources" meant that the consumption of resources could be looked at by different managers from the different perspectives just described.

7.3.5 Cross-System Comparability

In the extremely heterogeneous computer environment in this organization, the requirement to compare dissimilar systems was one of the most important goals. Ideally, we wanted to be able to produce performance statistics which presented a valid comparison, and to provide the comparison of different systems in a single succinct report.

7.3.6 Flexible Report Generation

The final requirement was for the flexible report generation. We learned through the CPM survey that there was a wide variety of CPM information needs within the organization. Different groups of reports would have to be provided to different managers. We also knew that the CPM system would have to respond to the needs of ad hoc studies which are frequently conducted. A flexible means of report generation would also provide additional choice in the types of measurements that we could use. We felt that this was particularly important in a heterogeneous system environment. "Canned" sets of performance measurements in off-the-shelf CPM tools might not provide the means of valid comparison of different systems. A flexible report generator would provide the means to easily develop custom reports, based upon measurement algorithms which we developed.

8. Evaluation of Available CPM Tools

Concurrently with the analysis just described, we conducted a preliminary evaluation of commercially-available CPM tools which might be relevant to our needs. The usually heterogeneous systems environment significantly reduced the number of options open to us. At minimum we needed CPM tools to measure the performance of IBM 360, UNIVAC 1110, DECSYSTEM 10, and PDP-11/70 systems. After this preliminary evaluation of available tools, only a handful of possibilities were available.[4] Had

[3]See Thomas Bell, <u>Computer Performance Management Through Control Limits</u> for a discussion of objective-directed and process-directed controls, (1).

[4]<u>The EDP Performance Management Handbook, Volume 2: Tools and Techniques</u>(5) was used to provide an initial inventory of commercially available performance measurement tools.

we been concerned with measuring the performance of a homogeneous environment for which many CPM tools exist (e.g., IBM 370), we would have had to develop the evaluation methodology for this stage much more fully.

9. Designing an Initial Operational Capability Based on Available Tools

At this point, for the first time in the design process, we formally brought together the pieces which had been intentionally separated in the analysis phase. The capabilities of relevant CPM tools were compared with the requirement stated in the CPM Needs Document. We began by considering CPM tools in three basic classes: software monitors, hardware monitors and job accounting systems.

9.1 Software Monitors

Software monitors were determined to be least useful for our purposes. The overhead required to run a software monitor continuously, as would be required to create a historical data base, would be significant. An even more severe problem would be coping with the massive amount of data which would be produced. But the basic problem was the fact that software monitors would not provide succinct answers to the questions being asked by upper level management, which were typically concerned with the resources allocated and consumed by different organizations and applications, and the available capacities of various systems to absorb new applications. This was an area in which the organization had had experience. Software monitors were available on several systems and had been used successfully in the past for trouble-shooting and system tuning. However, they had not proved useful for providing the required information to management.

9.2 Hardware Monitors

Hardware monitors had the attractive characteristic of being usable on the wide variety of computers in this systems environment. They were unique in their ability to provide fully comparable measurements for different machines, which was one of our selection criteria. Unfortunately, they were deficient in meeting other objectives. In order to provide the sort of historical data we wanted, hardware monitors would have to be hooked up continuously to a number of computers. This would be very expensive relative to other options. A more serious limitation was the inability of hardware monitors to provide information on the resources consumed by individual organizational elements and applications. This was one of our primary objectives. Combination hardware and software monitoring tools, which might have provided an attractive capability, were not available for the mix of systems we needed to measure.

9.3 Job Accounting Systems

Job accounting systems had the capability to provide the required management information relating resource utilization to organizational elements and applications. The relatively low acquisition and operating costs and low system overhead were also attractive. However, the objective of comparability of measurements was not met.

Job accounting systems use the measurements output to system log records, however, they may have been calculated. For management applications, this deficiency is partially offset by the fact that some measures can be made comparable by expressing them as dollar values. Based on the results of our CPM Survey, a job accounting system met our needs most directly.

At this point, our choice had narrowed down to two possibilities. Both were job accounting systems which could be used on more than one of our major mainframe types. Based on a comparison of these two tools with the requirements which we had identified, we recommended the procurement of one of these. The data base associated with this system provided a good basis for historical and trend analysis. The design of this data base made it possible to integrate information from other computers' system logs or from manually generated data. This characteristic made it possible to extend the capabilities of this system to monitor the performance of computers for which CPM tools were not readily available.

10. Define System Enhancements to Fill the Gap Between Initial Operational Capability and Desired Capability

It came as no surprise to find that a number of the needs stated in the CPM Needs Document could not be fulfilled by the initial operational capability which we had conceptualized. To define this gap with some precision, our client asked us to provide an inventory of the specific measures and performance data which are desired (based on the CPM Needs Document). This document can be used as a baseline against which to evaluate present capabilities and projected enhancements. We have already identified areas of shortfalls which will require additional work. For example, the measures of timeliness which are available are inadequate, and certain measures of the accuracy and utilization categories will require some custom design work. At this stage, we found that it was useful to have stated (and gotten confirmation of) some of the assumptions stated in the CPM Needs Document. The most important is probably the commitment to "do the do-able." As unfulfilled needs are recognized, there may be a temptation for some people to waiver in their commitment to take those first important steps. Another temptation may be to suggest a massive software design effort to "do it right" from scratch. Individuals succumbing to this temptation need to be reminded of the assumption that the use of available software should be maximized, and why that assumption was made. Aside from the problem of the cost of custom designing such software (which is likely to be exorbitant), a decision to go the custom software route puts off management's access to CPM information for a significant period, perhaps years. Using the documentation of CPM requirements as a guide, it is possible to begin a CPM program with the best available off-the-shelf technology and then systematically improve it until it approaches full satisfaction of the organization's needs.

11. Conclusions

During the initial design phases of the performance management system discussed here, we took certain actions which we recommend as part of the methodology for an organization comtemplating establishing a performance management system. In summary, these actions are:

1. Clarify the basic objectives of the design effort. Understand the basic differences between resource management and system tuning objectives and determine which is to be pursued.

2. Gain top-level management support for a top-down approach to designing the PMS. This support will facilitate getting the information needed to design an effective PMS.

3. Establish organizational responsibility for the PMS. Define responibilities for developing, operating, and analyzing output from the system.

4. Perform separate analysis of needs and capabilities. To intermingle these prematurely will result in stating only those needs which can easily be satisified.

5. Conduct a survey to determine the CPM information needs of managers within the organization. Use individual interviews to insure maximum exchange of information.

6. Write a CPM Needs Document to express the objectives of the PMS to be designed. Assumptions and conclusions should be clearly identified. Have the CPM Needs Document formally accepted as an official statement of requirements, even if it takes more than one try to do it.

7. Design an initial capability based on the commercially available CPM tools which best meet your stated needs. Avoid the temptation to try to custom design your own system until you have had some experience working with an operational PMS.

8. Design system enhancements to fill the gap between the initial operational capability and desired capability. Gaining experience with the initial operational system should provide significant insights which will become design considerations for system enhancements.

References

(1) Bell, Thomas E., "Computer Performance Management", in Management and Evaluation of Computer Performance, Computer Measurement Group, Inc., 1978.

(2) General Services Administration, Management Guidance for Developing and Installing an ADP Performance Management Program, 1978.

(3) Gotthardt, David W. and Sardo, Victor M., "Meaningful Data Processing Performance Measures for Management", in Management and Evaluation of Computer Performance, Computer Measurement Group, Inc., 1978.

(4) Howard, Phillip C., and Stevens, Barry A., The EDP Performance Management Handbook, Volume 1: Audit and Control, Applied Computer Research, 1978.

(5) Howard, Phillip C., The EDP Performance Management Handbook, Volume 2: Tools and Techniques, Applied Computer Research, 1978.

(6) National Bureau of Standards, Guideline on Computer Performance Management: an Introduction, 1977.

NAVAL LABORATORIES' QUALITY OF SERVICE STANDARDS

Joseph S. Dodds

Engineering and Computer Sciences Directorate
U.S. Naval Ocean Systems Center
San Diego, CA 92152

This paper summarizes the objectives, history and status of user quality of service standards that are being developed by the Naval Laboratories for their general purpose computer centers. The problem includes social, economic and technical challenges. Standards must be relevant to the users, non-competitive with other evaluation processes, economically applied and consistent among the different computer architectures installed at each Laboratory. Each standard currently being used will be presented along with the rationale for the standard.

Key words: Availability standards; batch processing; calibration programs; computer standards; interactive processing; quality of service; response time; turnaround time.

1. Background

General purpose (GP) computer centers at each Laboratory operate much like a service bureau, providing batch and interactive service to users from many different disciplines and charging each user for the services he receives. The GP manager is faced with the task of balancing capacity to meet the needs of the community, and of acquiring compilers, data base management systems, engineering aids and a host of other software packages needed to support his diverse customer base. Each Lab has established one or more "user groups" to work with the GP manager in analyzing requirements and evaluating the performance of the GP center. These intra-Lab groups have existed since the early sixties and have proved most valuable for the relationship between local computer service suppliers and their users.

In the late sixties it also became apparent that many issues that appeared to be local issues were common to all Naval Labs, so about ten years ago the Naval Laboratories formed an interlaboratory computer committee, the NLCC (Naval Laboratories Computer Committee). The interests of the NLCC have ranged from review and recommendations on proposed Federal controls of computers to development of an interlaboratory computer network. One of the high interest items on the agenda of the NLCC has been the development of computer quality of service standards, standards that address the types and levels of GP computer service the Labs should be supplying its scientific, engineering and management community.

In the early 1970's a draft set of service standards was distributed for review and comment. The standards were written from a user's point of view, attempting to articulate standards that would be meaningful to users, set rational expectation levels, and be simple and direct enough for a user to determine whether or not he had been served within limits of the standards. This user orientation needs to be clearly understood. Unlike the highly internalized concerns usually associated with performance monitoring, e.g., concerns with the physical management and fine tuning of a system's I/O, memory, CPU, disk access, the NLCC Quality of Service Standards attempt to address the end result as seen by a engineer, programmer or scientist who is using the computer as a tool.

The 1970 draft set of standards included standards for availability, batch turnaround time, interactive response time and keypunch service times. The draft also recommended a series of good management practices for control of a systems hardware/software configurations, analysis of user community needs and establishment of computer systems advisory groups made up of users and suppliers of computer services. The concepts were well received by NLCC. Members concurred in the recommended management practices. They agreed that the times given for batch turnaround and for interactive response stated rational and desirable goals, and they further agreed to put the draft standards into practice at their Laboratories.

Practice rapidly revealed one serious flaw - measurement of performance varied. Laboratories with IBM equipment measured performance one way. Those with UNIVAC equipment another, and CDC a third and so on and so on. Although the standards aimed for uniformity across Laboratories, the heterogeneous array of computer systems appeared to work against the objective of commonality and uniformity.

To remedy this flaw the NLCC reconstituted its Quality of Service Standards Subcommittee in 1977. The subcommittee was tasked to recommend actions, new standards or techniques that would bring uniformity and commonality to the measurment and interpretation of User Quality of Service Standards. The balance of this paper will detail the results of the committee work up to May of 1979 and describe tasks that are still underway to evaluate and validate the committee's recommendations.

To keep the reader from misconstruing the scope or intent of the standards, remember that these standards are experimental and being tested by the Naval Labs for the Naval Labs.

2. Batch Turnaround Standards

Before stating the batch turnaround standard we need to define the terms job, turnaround time, job size and priority. The following discussion also provides some insight on why the definitions were selected.

In this day of comprehensive and flexible operating systems, multi and parallel processing, and interfaces between streams of programs that may diverge on funding, user population or submittal sequences, the meaning of "a job" can differ greatly. Since all comparisons of performance to standards are based on the concept of a job, agreement on a single meaning is essential. A job is, therefore, used in the standards to mean one task or group of tasks that are identified as a single unit of work against which resource utilization is collected and uniquely identified.

The general principle that standards must be acceptable to the community to which they apply affected the definition of turnaround time. Standards that are perceived as excessively burdensome, cost more to apply than they are worth, or in other ways fail to gain concensus may as well remain unstated. A NLCC turnaround time standard based on portal-to-portal, input counter to output bin would have exemplified this principle. One or two Naval Labs do collect counter to bin time data; most do not and were opposed to collecting counter to bin data. The machine measured turnaround time, therefore, was adopted as a workable and acceptable definition.

Operating of remote sites serviced by a Naval Laboratory Computer Center is usually the responsibility of a separate chain of command. The central site is without authority to direct when the remote site will accept a completed job. Hence, the standards include the caveat that the turnaround period terminates when remote batch terminals are informed that files are ready for transmission.

The standard's definition of turnaround time is: The lapsed time between reading the job's last system directive and either:

1. Outputting the last line of data, or

2. In cases of remote output sites, informing the site that all output data is ready.

Turnaround time is also dependent on the amount of work that must be done to transform inputs into outputs. The amount of work defines job size, a term that is also descriptive of a user's general perception of the computer resources applied to a job. Each Laboratory was certainly familiar with such a concept. Operating systems incorporate some job size descriptors that are useful for scheduling job streams. The question is how to establish job sizes in such a way that jobs of a given size run on say a CDC 6600 will turn out to be classified the same way when executed on a UNIVAC 1100?

The key to this problem in the turnaround standard lies in a calibration program. The calibration program is an arbitrary program, (See Figure 1), reads an arbitrary number of records, R, performs the computations an arbitrary number of times, N, and prints an arbitrary number of lines L. Reference [1] provided the original structure for the Calibration Program and Exhibit A is a listing of the program as currently written for the NLCC standards.

When run as a Calibration Program two combinations of values are inserted for R, N and L. Resource utilization data and the cost data are then collected for each combination.

The results of the run with the lower set of values (R = 11,700, N = 827,000, L = 10) establishes the lower boundary (A_L) of a medium size job the combination of higher values (R = 25,070, N = 1,722,000, L = 10) establishes the upper boundary (A_U) of a medium size job in terms of that specific computer center. A_L and A_U are expressed as Work Units (WU). One WU equals one dollar of charges at normal priority.

Results of the calibration program are used to establish job size. Small jobs are those with WU's less than A_L, Medium jobs are those that have WU's ranging from A_L to A_U, Large jobs have WU's greater than A_U. A user can refer to the tail sheet of his output, compute his time in the system, immediately read the cost (WU) and determine whether or not that job had been processed within Standard turnaround time.

The Calibration Program concept assumes:

a. That the Naval Laboratory computer centers operate much as a commercial service, that is, costs are recovered by charging users on an "as used basis". Cost and cost recovery are targeted to produce a zero balance. Production during a given period of x number of WU's charged at the base priority rate should defray all costs for that period.

b. Classification of job size is, therefore, independent of computer capacity.

c. Three job sizes are sufficiently discriminating for user verification of the quality of his service.

d. Work Units equate to resources used.

A word needs to be said about the repeatability of turnaround time for a particular job. It is well known that job run times vary as system workloads vary. There is little variation, however, in accounting of resources used. A job, therefore, that is close to the A_L or A_U boundary may when run one time be processed within the stipulated time, and when run again be outside, either above or below the stipulated time. A "90% of all jobs" phrase was introduced to compensate for the phenomenon of run time variations and also to allow for those jobs which utilize the system resources in an unusual manner, causing them to severely bias the cost/turnaround time algorithm.

Another factor, in addition to job size, that impacts turnaround time is the user requested priority for his job. While the computer center manager is concerned with the cost and productiveness of his computer system, the project manager is concerned with the productiveness of his people and the totality of his costs. The scaled priority response times and costs gives the project manager an opportunity to balance his production needs and financial resources. He can use the NLCC standard as an input for making his trade-offs and as a means of evaluating the service he receives.

Satisfying a request for high priority handling of a job can degrade the WU production potential. The higher the priority of the job, the more jobs that are temporarily set to one side, the greater the impact on production. Hence, higher rates are usually charged for high priorities, compensating for the impact on overall system throughput. The opposite is true for jobs running at low priority.

The experimental batch turnaround standard can now be stated completely.

2.1 Definitions

<u>A Job</u> is a single task or group of tasks that are identified as a single unit of work against which resource utilization is collected and uniquely identified.

<u>Turnaround Time</u> is the lapsed time between reading the job's last system directive and either:

1. Outputting the last line of data, or
2. In cases of remote output sites, informing the site that all output data is ready.

<u>The Calibration Program</u> is a set of program and program data used to establish the lower and upper limits

of computer resources used on medium size jobs. Figure 1 provides a listing of the calibration program.

The Normal Priority is selected by management of a computer site as the priority that will be used for determining the normal and adjusted turnaround time standards described in paragraph (c).

Work Units (WU) are the computer resources for which utilization data is recorded during the initiation, processing and termination of a job and from which user billings are generated. One WU equals one dollar of charges at normal priority.

WU_j is the sum of the Work Units applied to job $_j$.

2.2 Measuring Job Size

The lower and upper boundaries for a Medium size job are determined at each site by the Calibration Program; A_L is Lower Boundary of Medium Job; A_U is the Upper Boundary. Job size categories will be:

Small, where	WU	$<$	A_L		
Medium, where	A_L	$\leq$	WU_j	$\leq$	A_U
Large, where	WU_j	$>$	A_U		

2.3 Turnaround Standards

Ninety percent (90%) of all jobs within the three job size categories shall have the following turnaround times under Normal priority during all periods of scheduled availability:

Small - 45 minutes or less

Medium - 120 minutes or less

Large - 360 minutes or less

When a priority is offered at a special rate, the following adjustments shall be made to turnaround times.

1. For step increases in rates, turnaround times will be $.75^P$ of the Normal priority computed to the nearest tenth of an hour, expressed in minutes. (p = one tenth of the percentage change in rate relative to the normal rate).

2. For step decreases in rates, turnaround time will be 1 + p of the Normal priority.

3. Examples of medium job size:

Priority	Rate Change	Minutes Turnaround
Highest	+ 20%	66
Higher	+ 10%	90
Normal	0	120
Lower	- 10%	240
Lowest	- 20%	360

3. Interactive Response Time Standards

The batch processing standards are stated in a way which permits a user to directly determine whether or not his job was done within the stipulated time. The same is not true for interactive standards. The determination as to whether or not the response of a system is meeting the interactive standards will not be directly available to the user. His feedback on performance will be in the form of reports and statistics prepared from data collected at one terminal location that includes a specially designed Interactive Measurement Tool (IMT) that executes scenarios and records the computer response for each scenario action.

The IMT will run in a half duplex mode, timing responses to the nearest tenth of a second, and simulating user think time delays and typing speeds. Table 1 contains the experimental standard scenario plus the expected response times for each event in the scenario. The solicit response means that the computer is ready for the next input. In practice, the representation of that event may vary from computer system to computer system.

These NLCC standards are fairly straightforward. The definitions of terms and contents are based largely on earlier studies performed by the National Bureau of Standards [2] [3].

3.1 Interactive Standards

Response Time for interactive or conversational users is defined as the time interval between the pressing of the last key of the user's input message to the display of the first character that responds directly to the user's request.

```
C  NLCC STANDARDS COMMITTEE CALIBRATION PROGRAMS VERSION JUNE 1978.      SYN 0001
C  ANSI FORTRAN VERSION  OCTOBER 1977.  NORA M. TAYLOR,  DTNSRDC         SYN 0004
C  VERSION 1.1 7AR. 1978  PREVENT OVER OPTIMIZATION OF KERNEL            SYN11  1
C  SYNTHETIC PROGRAM FOR JOB STREAM SIMULATION AND TIMING COMPARISONS.   SYN 0005
C  MODIFIED FROM PL/I PROGRAM, P. 56, PROCEEDINGS OF FALL JOINT          SYN 0006
C  COMPUTER CONFERENCE, 1971.                                            SYN 0007
C  INPUT PARAMETERS VARY THE COMPUTE TIME AND I/O TIME.                  SYN 0008
C  R = NO. OF RECORDS TO BE PROCESSED IN EACH FILE.  AT LEAST 1 RECORD   SYN 0009
C    WILL BE DONE.                                                       SYN 0010
C  N = TOTAL NO.  OF TIMES TO EXECUTE COMPUTE KERNEL.                    SYN 0011
C  L = TOTAL NO. OF LINES TO PRINT                                       SYN 0012
C  LPR = NO. OF LINES TO PRINT FOR EACH RECORD PROCESSED (UP TO L).      SYN 0013
      INTEGER C,CHECK,OUT1,OUT2,OUT3,OUT4,P,R                            SYN 0014
      DIMENSION MDATA(5)                                                 SYN 0015
      DATA MDATA(2),MDATA(3),MDATA(4),MDATA(5)/1,1,1,1/                  SYN 0016
C  LOGICAL TAPE UNITS                                                    SYN 0017
      OUT1=8                                                             SYN 0018
      OUT2=2                                                             SYN 0019
      OUT3=3                                                             SYN 0020
      OUT4=4                                                             SYN 0021
      C = 5                                                              SYN 0022
      P = 6                                                              SYN 0023
C  INITIALIZATION                                                        SYN 0024
      CHECK=0                                                            SYN 0025
      L=1                                                                SYN 0026
      LINES=0                                                            SYN 0027
      LPR=1                                                              SYN 0028
      NREPS=0                                                            SYN 0029
C   TO CONTROL PRINTING BY INPUT PARAMETERS L AND LPR, CHANGE            SYN 0030
C  THE READ CARD BELOW TO READ(C,61) R,N,L,LPR.  FORMAT IS OK.           SYN 0031
      READ(C,61) R, N                                                    SYN 0032
61    FORMAT(8I10)                                                       SYN 0033
      IF(R.LE.0) R=1                                                     SYN 0034
      IF(N.LT.0) N=0                                                     SYN 0035
      WRITE(P,60) R, N                                                   SYN 0036
60    FORMAT(10H1SYN 1.1--,I10,10H RECORDS, ,I10,17H COMPUTE KERNELS.)   SYN11  2
C  CREATE A FILE                                                         SYN 0038
      DO 80 J = 1, R                                                     SYN 0039
         MKEY=J                                                          SYN 0040
         MDATA(1)=J                                                      SYN 0041
         MCHECK=CHECK+J                                                  SYN 0042
         MSUM = J - 1                                                    SYN11  3
80    WRITE(OUT4,61) MKEY,MSUM,MCHECK,MDATA                              SYN 0044
      ENDFILE OUT4                                                       SYN 0045
      REWIND OUT4                                                        SYN 0046
C  THE OVERALL NUMBER OF COMPUTE KERNELS IS LIMITED TO N BY THE NREPS    SYN 0047
C  COUNTER AND IS DISTRIBUTED OVER THE  I/O RECORDS AS FOLLOWS--         SYN 0048
C  FOR N.GE.R, THE KERNEL IS DONE N/R + 1 TIMES PER RECORD,              SYN 0049
C  FOR N.LT.R, 2 TIMES PER RECORD,                                       SYN 0050
C  FOR N = 0, NONE.  N IS PREVENTED FROM BEING NEGATIVE.                 SYN 0051
      LIM = (N-R)/R +3                                                   SYN 0052
C              MAIN LOOP                                                 SYN 0053
      DO 5 L1 = 1, R                                                     SYN 0054
         READ(OUT4,61) MKEY,MSUM,MCHECK,MDATA                            SYN 0055
         IF(NREPS.GE.N) GO TO 82                                         SYN 0056
C              CARD SYN 57 REMOVED IN SYN 1.1                            SYN11  7
C  COMPUTE KERNEL                                                        SYN 0058
            L2 = 1                                                       SYN 0059
2     L2 = L2+1
C  CALL A USER FUNCTION TO PREVENT UNWANTED OPTIMIZATION.                SYN11  8
            RN = F(FLOAT(L2))                                            SYN11  4
            MDATA(1)=SQRT(RN)                                            SYN 006
            MDATA(2)=EXP(RN/1.E15)                                       SYN 0061
            MDATA(4)=L1+L2                                               SYN11  5
            MDATA(5)=L2                                                  SYN 0064
            MSUM=MSUM+1                                                  SYN 0065
            MCHECK=LIM /L2                                               SYN11  6
            NREPS=NREPS+1                                                SYN 0067
            IF(LIM.GT.L2) GO TO 2                                        SYN 0068
82       WRITE(OUT1,61) MKEY,MSUM,MCHECK,MDATA                           SYN 0069
         WRITE(OUT2,61) MKEY,MSUM,MCHECK,MDATA                           SYN 0070
         WRITE(OUT3,61) MKEY,MSUM,MCHECK,MDATA                           SYN 0071
         IF(LINES.GE.L) GO TO 5                                          SYN 0072
C  LINE PRINTER LOOP.  DONE ONLY ONCE IN STANDARD VERSION.               SYN 0073
         DO 4 K = 1, LPR                                                 SYN 0074
            WRITE(P,61) MCHECK,MDATA(5),K,LINES,NREPS                    SYN 0075
4        LINES = LINES + 1                                               SYN 0076
5     CONTINUE                                                           SYN 0077
      WRITE(P,61) MKEY,MSUM,MCHECK,MDATA                                 SYN 0078
      REWIND OUT1                                                        SYN 0079
      REWIND OUT2                                                        SYN 0080
      REWIND OUT3                                                        SYN 0081
      REWIND OUT4                                                        SYN 0082
       STOP 77                                                           SYN 0083
       END                                                               SYN 0084
@FOR,IS F
      FUNCTION F(X)                                                      SYN11F 1
C    DUMMY FUNCTION TO PREVENT UNWANTED OPTIMIZATION.                    SYN11F
      F = X                                                              SYN11F 2
      RETURN                                                             SYN11F 3
      END                                                                SYN11F 4
```

Figure 1. Calibration Programs

Table 1. Standard Scenario #1

(Total Standard Lapsed Time 20 Minutes Thru Event 17)

EVENT	GENERAL OPERATOR ACTION	SPECIFIC OPERATOR ACTION	COMPUTER SYSTEM		
			ACTION	RESPONSE MEASURE EVENT	STANDARD RESPONSE TIME (SECONDS)
1	SIGN on Sequence (ID,Password, etc.)		System Dependent *	Solicit	30
2	ATTACH PERMANENT FILE		Search library, attach file DIMENSIØN II (10000) DØ 110 N = 1,5 WRITE (6,10) 10 FØRMAT ('DATA-PLEASE') READ (5,20) M 20 FØRMAT (A1) DØ 100 J = 1,10000 100 II (J) = 1,10000 110 CØNTINUE STØP END	Solicit	6
3	CREATE TEMPORARY FILE SPACE		Find and resolve space	Solicit	3
4	COPY		Write permanent file into temporary file	Solicit	3
5	CALL EDITOR		Load and execute the editor	Solicit	6
6	FIND LINE	Find 'II (J)'	Locate specified character string	First character of string	2
7	CHANGE LINE	Change to '--100 - J (II) = J'		First character of string	2

* This activity is subject to a multitude of procedural variations within the NLCC Community.
The Standard encompassed only essential procedures such as User ID, password and accounting information.

Table 1. (Continued)

EVENT	GENERAL OPERATOR ACTION	SPECIFIC OPERATOR ACTION	COMPUTER SYSTEM		
			ACTION	RESPONSE MEASURE EVENT	STANDARD RESPONSE TIME (SECONDS)
8	DELETE LINE	Delete		Solicit	2
9	INSERT LINE	Insert '--100 - II(J)=J'		Solicit	2
10	END EDIT		Save temporary file	Solicit	2
11	COMPILE		Compile (Suppress Listing)	Solicit	15
12	LINK, EDIT, GO		Build object code and execute	First character of program message	15
13	[DELAY 30 SECONDS]				
14	DATA INPUT		Execute	First character of program message	5
	(Repeat Event 14 for four (4) more times)				
15	TERMINATE PROGRAM			Solicit	5
	(Repeat Events 4 thru 15 four (4) more times)				
16	CATALOG TEMPORARY FILE			Solicit	6
17	DELETE TEMPORARY FILE			Solicit	3
18	(Transfer scenario event/time data from terminal to host)				
19	SIGN OFF			First character of accounting data	

A standard Scenario is a predefined sequence of events describing user initiated commands, system reponses and the standard response times for each event and for the total scenario. (Table 1 provides a description of the Standard Scenario).

Ninety percent (90%) of the actual interactive response times for each event in a standard scenario shall equal or better the standard response time. Ninety percent (90%) of the active executions of a scenario shall be completed within the standard elapsed time.

Comparison of a system to the above standards shall be determined by collecting data on a full scenario during every hour of prime shift over five working days. That is: ten samples per day, 50 samples in all.

4. Availability

There are two standards for availability. One for availability during periods when most of the users are at work and are asking for service, the Prime Hours, and the second for all other scheduled service times, the Non-Prime Hours. The differentiation was made to emphasize the need for superior system performance during the heavy demand periods. Heavy periods not only represent times when most users are trying to get their work done but also periods that seem most commonly afflicted by unscheduled interruptions.

The scope of the down time period covers the user's perception of availability not the perception of hardware maintenance personnel or system maintenance programmers. Partial degradation of a system (e.g. loss of a printer or tape drive or some memory) that does not completely close down the system is not included in the computation of availability. The impact of partial degradation will be reflected in the ability of the GP center to meet the batch and interactive standards.

4.1 Availability Standards

Prime hours are the 8-1/2 hour period designated by each Laboratory as its standard first shift plus one and one-half hours at the end of the first shift. Total prime hours equal ten (10).

Non-prime hours are all the hours scheduled for support of general users less the ten (10) prime hours.

Down time is the lapsed time that a computer system is down during either Prime or Non-prime hours. Being down means that no useful work can be performed for general users. Down time ends when system hardware and software recover is complete.

Availability equals one minus down time during a scheduled period divided by the hours in that period.

$$A = 1 - \frac{T_D}{H}$$

Prime hours availability shall be 98%.

Non-prime hours availability shall be 90%.

5. Conclusion

Recall that the NLCC Quality of Service Standards are still experimental and being tried only by participating Naval Laboratory Centers. Still in progress are tasks for:

a. Delivery of system reports that will convey to users how well a given computer center has performed in comparison with the standards.

b. Validation that the standards do indeed describe servicing attributes that users class as important and that the measures for each attribute are reasonable.

While there is much left to be done perhaps even to the extent of throwing out some ideas and introducing new ones, the NLCC Quality of Service Standards are filling a previous void by providing a baseline for the continued evaluation and development of user oriented standards.

References

[1] Wood, D. C., Forman, E. H., Throughput Measurement Using a Synthetic Job Stream, AFIPS Conference Proceedings, Volume 39, 1971 Fall Joint Computer Conference, pp. 51-55.

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

[3] Watkins, S. W., Abrams, M. D., Survey of Remote Terminal Emulators, NBS Special Publication 500-4, April 1977.

CPEUG 79

Workload Analysis and Capacity Planning

COMPUTER SYSTEM MIGRATION PLANNING THROUGH BENCHMARK PERFORMANCE EVALUATION

Arabinda Mukherjee
AT&T
Piscataway, NJ

Aridaman K. Jain
Bell Laboratories
Holmdel, NJ

and

Bruce A. Ketchledge
AT&T
Piscataway, NJ

ABSTRACT

This paper presents the development of a program which provides guidelines for migration from an IBM 168-3 to an IBM 3033. This program, based on the Bell System 3033 benchmark performance data, consists of analytical and empirical models. The benchmark consisted of several real and synthetic job streams, which were run on the 168-3 and the 3033 under an MVS Operating System. The migration guidelines are in terms of (i) key 3033 system performance measures, (ii) gross configuration tuning information, and (iii) execution times for batch job steps. Furthermore, a component of this program can be used as a capacity planning aid for an existing 3033 system.

1. Introduction

In March 1977, IBM announced the 3033 processor, a member of the System/370 series at the high end of the performance scale. An integral component of the 3033 processor is the System/370 Extended Facility. This facility, when supported by the MVS/SE (System Extensions) program product, improves the performance of MVS. The 3033 processor was initially advertised as being capable of an instruction execution rate of 1.6 to 1.8 times that of a 168-3 processor. In May 1977, representatives of several Bell System entities agreed to conduct a joint 3033 benchmark to assess its potential in the Bell System. The on-site portion of the benchmark began on February 20, 1978, and ended on April 8, 1978.

The objectives of the benchmark study were to:

a. develop guidelines for migration from an IBM 168-3 to an IBM 3033, including the development of internal[1] and external[2] performance ratios, and

b. obtain planning data for selected applications.

This paper presents the development of a program for migration planning from a

[1]The ratio of 168-3 total CPU active time to 3033 total CPU active time for a given job stream.

[2]The ratio of 168-3 elasped time to 3033 elapsed time for a given job stream.

168-3 to a 3033 for batch environment. This program can also be used as a capacity planning aid for an existing 3033 system. Section 2 of this paper contains a discussion f the benchmark test plan, including the xperimental design. A detailed description f the migration planning program is preented in Section 3. The data analysis and evelopment of empirical as well as analytial models, which are components of the igration planning program, are presented in ection 4. Section 5 contains a summary.

2. Benchmark Test Plan

Five applications (A-D,F), which consist of one or more job streams, were used to ccomplish the benchmark objectives. Workload type, program type, language and number of job streams for each of these applications are given in Table 1. These applications were used to carry out a set of general experiments described in Section 2.1. Additionally, Application F was used to conduct a series of special experiments described in Section 2.2.

2.1 General Experiments

The sequence of experiments for each application was as follows:

a. Perform multiple runs for a given job stream on the 168-3 processor to establish a base of comparison (the 168-3 base experiment),

b. run the same job stream on the 3033/SE processor by using the same hardware configuration as was used on the 168-3 (the 3033/SE untuned experiment),

c. run the same job stream again on the 3033/SE processor as in (b) but after elimination of obvious system bottlenecks[3] (the 3033/SE tuned experiment), and

d. repeat the "3033/SE tuned experiment" on an identical configuration as in (c) with the 3033 processor without the SE feature (the 3033 experiment) for Applications A-D.

[3]Elimination of bottlenecks consisted of one or more of the following:

(i) an increase in the number of initiators,

(ii) an increase in the number of I/O devices, and

(iii) data set placement changes.

In the case of Application F, there are 15 job streams (defined in Table 5). In order to conduct the experiment under extreme conditions, four of these job streams, which consist of the largest percentage of an individual activity, were used in the above sequence of experiments.

The performance measurement data, which are used for the migration planning program, are summarized in Table 2 for Applications A-D and in Table 3 for Application F. These performance data are used to build the internal performance ratio models and the difference performance ratio model which are described in Section 4.

2.2 Application F Experiments

The objective of Application F experiments was to describe job step execution time as a function of the state of the system in terms of: (i) level of CPU activity (CPU utilization), (ii) level of I/O activity (start I/O per second), and (iii) level of paging activity (total PAGING/second), under certain conditions[4].

Four tasks, which can be characterized as CPU bound (TASK 1), I/O bound (TASK 2), PAGE bound (TASK 3), and balanced (TASK 4) were used to create the 15 job streams. TASK 4 requires the least amount of resources and does moderate amount of CPU, I/O, and PAGING activity. Tasks 1 and 2 were created from Task 4 by increasing CPU and I/O activity, respectively. Task 3 was created from Task 4 by increasing both CPU and Paging activity. The number of Direct Access Storage Device (DASD) files, total number of executions of Execute Channel Program (EXCP), CPU seconds, and EXCP/CPU second for each of these tasks are given in Table 4. The composition of the 15 job streams is given in Table 5. These compositions were developed to cover as wide a range of system state variables (i.e., CPU utilization, SIO rate and paging rate) as possible.

Each job stream consisted of 60 job steps. The execution sequence of the tasks was generated at random based on their

[4]These conditions are:

(i) all job steps have the same dispatching priority,

(ii) all I/O is confined to 3330 (Mod. I and II) and/or 3350 type devices, and

(iii) I/O load is equally distributed among the DSAD devices.

composition in the job stream. A summary of the performance data for the 15 job streams in Application F is given in Table 6. These performance data were used to build the job step execution time models, which are described in Section 4.

3. Migration Planning Program

Detailed analysis (see Section 4) of the benchmark performance data has led to the development of a migration planning program called the 3033/PLAN. An additional feature of the program is that it can be used as a capacity planning aid for an existing 3033 for pure batch environments.

Migration planning guidelines from the 168-3 to the 3033 are given in terms of the following predicted 3033 performance measures:

(i) Problem Program, Supervisor State, and Total CPU utilization,

(ii) SIO rate/second,

(iii) internal and external performance ratios compared to the 168-3 CPU, and

(iv) execution times of batch job steps in the 3033 environment.

Additionally, the program develops tuning guidelines for the 3033 environment in terms of the number of initiators and the number of DASD devices to make effective use of the faster processor.

For an existing 3033, the program can be used as a capacity planning aid in terms of the execution times of batch job steps for projected 3033 CPU utilization, SIO, and PAGING rates. Optionally, prior to its use as a capacity planning aid, the program can be used to tune the 3033 system.

3.1 Scope

The scope of the program is defined by the following:

(i) The program is limited to single processor configurations only.

(ii) The program is valid for MVS/batch environment only.

(iii) As a capacity planning aid, the program is limited to prediction of execution times of batch job steps having equal priority and all I/O confined to 3330 and/or 3350 type devices.

(iv) The 3033 total paging rate, which is required in the estimation of the execution times of batch job steps, is derived by multiplying the 168-3 total paging rate by 3.5[5]. An accurate estimate of the total 3033 paging rate is beyond the scope of this program.

(v) For the 3033, unrealistic execution times of batch job steps may result when the SIO rate/second is higher than 160 and/or the paging rate/second is higher than 40.

(vi) The I/O configuration is limited to 3330 (Mod. I and II) and 3350 type devices. During the optimization process, only 3330 type devices are added to eliminate device bottlenecks.

3.2 Assumptions

The migration planning program is based on the following assumptions:

(i) the computer system is in a steady state condition,

(ii) the I/O load is equally distributed among the disk drives, and

(iii) the working set size (or the average real memory owned) is approximately equal to 300K bytes for all job steps.

3.3 Detailed Description

3033/PLAN makes use of both empirical and analytical models. 3033/PLAN, written in FORTRAN, is self-prompting for ease of use and is driven by a simple set of input parameters which are easily available from standard measurement tools and accounting packages. This program produces simple output and as mentioned earlier, it can be used in two different ways:

(i) as a migration planning aid through the use of 168-3 system performance and job step data, and

(ii) as a capacity planning aid for an existing 3033 through the use of projected 3033 system performance and job step data.

The empirical and analytical models, which

[5]This is the average ratio of the 3033 paging rate to the 168-3 paging rate observed in this study.

are used in 3033/PLAN, are described in Section 4.

3.3.1 Migration Planning Aid

Figure 1 shows a gross level system flow for the 3033 Migration Planning Aid program in terms of input, process, and output. A flow chart of both migration planning and capacity planning algithms is shown in Figure 2. The individual steps of the migration planning algorithm are presented below in more detail than in Figure 2:

(a) Read 168-3 system performance and job step data.

(b) Calculate the seconds during a one hour interval for (i) 168-3 total CPU, (ii) 168-3 Problem Program (PP) CPU, and (iii) 168-3 Supervisor State (SS) CPU from their respective percent utilizations.

(c) Estimate superivsor state ratio[6] and problem program ratio[7] by using the SRATIO and PRATIO models (Section 4) respectively.

(d) Calculate 3033/SE PP CPU seconds by dividing 168-3 PP CPU seconds by PP ratio estimated in Step (c).

(e) Calculate 3033/SE SS CPU seconds by dividing 168-3 SS CPU seconds by SS ratio estimated in Step (c).

(f) Calculate total 3033/SE CPU seconds by adding 3033/SE SS CPU seconds and 3033/SE PP CPU seconds.

(g) Calculate internal performance ratio by dividing total 168-3 CPU seconds by total 3033/SE CPU seconds.

(h) Use the analytical model ANMOD (Section 4) to estimate 3033/SE CPU utilization from the number of initiators and the number of DASD devices in the 168-3 configuration, the SIO per 168-3 CPU second, the ratio of SS CPU seconds to PP CPU seconds for 3033, and the PP ratio. This CPU utilization represents the case when only the 168-3 CPU is replaced with a 3033/SE CPU (i.e., no other change in the 168-3).

(i) Calculate 3033/SE job stream elapsed time from 3033/SE CPU utilization and total CPU seconds. Calculate PP and SS CPU utilization from 3033/SE PP and SS CPU seconds and job stream elapsed times.

(j) Calculate external performance ratio by dividing the 168-3 job stream elapsed time of one hour by 3033/SE job stream elapsed time. Also, calculate SIO/second for the 3033/SE from SIO per 168-3 CPU second, 3033/SE CPU utilization and internal performance ratio. For 3033/SE system, go to step (m).

(k) Estimate the difference in the internal performance ratios between 3033/SE and 3033/NON-SE from 168-3 SS percent and 168-3 SIO rate per second by using the empirical model SEMOD (Section 4). Calculate 3033/NON-SE internal performance ratio by subtracting the above difference from the 3033/SE internal performance ratio.

(l) Estimate 3033/NON-SE performance by using Steps (h), (i) and (j).

(m) Estimate execution times of job steps by using the empirical model APPLAN168 (Section 4).

(n) Estimate average device utilization by using the analytical model DEVUT (Section 4).

(o) Go to Step r if 3033 (SE or NON-SE) CPU utilization is at least 85 percent and average device utilization is at most 40 percent.

(p) If device utilization is greater than 40 percent, add two 3330 devices. Recalculate CPU and device utilizations and go to Step (o).

(q) Add two initiators. Recalculate CPU device utilizations and go to Step (o).

(r) If initiators or devices are added to the original configuration, recalculate execution times of job steps by using APPLAN168 from the 3033 tuned system performance data and 168-3 job step data.

[6]Supervisor state ratio = 168-3 SS CPU seconds/3033 (SE) SS CPU seconds.

[7]Problem program ratio = 168-3 PP CPU seconds/3033 (SE) PP CPU seconds.

(s) Stop.

3.3.2 Capacity Planning Aid

The program input can be either current or projected 3033 (SE or NON-SE) system performance and job step data. The program can be used to either tune the system first (if the system is not tuned properly) and then predict execution times of job steps in the tuned environment or predict execution times of job steps directly without any tuning.

Figure 3 shows a gross level system flow in terms of input, process and output. A detailed flow chart of the algorithm is incorporated in Figure 2. The individual steps in the algorithm are presented below:

(a) Read 3033 (SE or NON-SE) system performance data and job step data.

(b) Use[8] Steps (n), (o), (p), (q), and (r) of Section 3.3.1 to develop the tuning guidelines and the job step execution times if the tuning option is elected. If the tuning option is not chosen, then go to Step (c) below.

(c) Use the system performance and job step data from the above steps to estimate execution times of batch job steps by calling the APPLAN3033 model.

(d) Stop.

4. Data Analysis and Model Development

The migration planning program discussed in Section 3 consists of empirical and analytical models developed from benchmark performance data and classical queueing theory. This section presents the data analysis, which led to the development of the empirical models, and a brief description of each empirical and analytical model.

4.1 Internal Performance Ratio Models

The internal performance ratio is the ratio of 168-3 total CPU time to 3033 total CPU time for a given job stream. The primary objective of the internal performance ratio models was to map 168-3 total CPU time to 3033 (SE or NON-SE) total CPU time based on 168-3 system performance measures. Two models which accomplish this objective, the supervisor state ratio model (SRATIO) and the Problem Program ratio model (PRATIO), are discussed in this section.

[8]In Step (r) of Section 3.3.1, use APPLAN3033 instead of APPLAN168.

The data needed for the fitting of internal performance ratio models are given in Tables 2 and 3. These tables show that the ratio of internal performance of a 3033/SE processor to that of a 168-3 processor ranged from 1.65 to 2.29 in the problem program state and from 2.33 to 3.02 in the supervisor state. It may be recalled that the overall ratio was initially advertised to be between 1.6 and 1.8.

4.1.1 SRATIO

Let	SRATIO_i	=	the ratio of supervisory state time in 168-3 to supervisory state time in 3033/SE for job stream i,
and	SIO_i	=	168-3 start I/O rate per second for job stream i.

Several regression models were explored to describe SRATIO as a function of SIO. The following linear model explains about 86 percent of the variation in SRATIO:

$$\text{SRATIO}_i = 3.227 - 0.01888(\text{SIO}_i) + 0.00010(\text{SIO}_i)^2 \quad (1)$$

The adequacy of the above model was examined through the techniques of analysis of variance and residual plots. Analysis of variance showed that the fitted regression coefficients in Model (1) are statistically significant. An examination of some residual plots did not reveal any unusual patterns. The statistical significance of the fitted regression coefficients and the lack of unusual patterns in the residual plots support the adequacy of Model (1). In the subsequent sections an empirical model is referred to as adequate, when similar examination through the techniques of analysis of variance and residual plots yields similar results.

4.1.2 PRATIO

Let	PRATIO_i	=	the ratio of problem program CPU seconds in 168-3 to problem program CPU seconds in 3033/SE for job stream i,
and	MIPS_i	=	168-3 Millions of Instructions Per Second for job stream i.

PRATIO was found to be highly correlated with MIPS. The following linear regression model explains about 90 percent of the variability in PRATIO:

$$\text{PRATIO}_i = 2.682 - 0.228\ (\text{MIPS}_i)\ . \quad (2)$$

As in the case of Model (1), Model (2) was found to be adequate.

4.2 CPU Utilization Model

CPU utilization (3033/SE and 3033/NON-SE) is required for the computation of the external performance ratio and the tuning of the 3033 system. Here, a closed queueing network approach is used to derive an expression for CPU utilization. This expression is referred to as the ANMOD model in Section 3.

The characterization of the computer system as a network of queues is given below:

a. The CPU is represented as an M/M/1 queue [1][9] whose service rate equals the SIO rate per CPU second.

b. Each I/O device is represented as single M/M/1 queue whose average service time was set to 40 milliseconds (for 3330 type devices) or 33 milliseconds (for 3350 type devices). I/O activity is assumed to be evenly distributed over the devices.

c. Each program is represented as a transaction. The number of transactions was set equal to the number of initiators. The transactions (programs) alternately visit the CPU and I/O servers. Within a single program, CPU and I/O activity do not overlap.

The steady state solution of the queueing network is obtained by the use of a formula for the delay at an infinite calling population server with a suitable modification [2]. The resulting expression for 3033 CPU utilization is given below:

$$\rho_C = \frac{b}{2a} - \frac{\sqrt{b^2-4ac}}{2a} \quad (3)$$

where

ρ_C = CPU utilization on the 3033,

$a = \left(\frac{I-1}{I}\right)[1+\delta+(I\delta+I-\delta)/D]$,

$b = 1 + \frac{I-1}{D} + \lambda(I\delta+I-\delta)$,

$c = \lambda I$,

$\lambda = \frac{\mu_D}{\mu_C}$,

I = Number of initiators,

μ_C = SIO rate per CPU second,

δ = Supervisor overhead

$= \frac{\text{Supervisor State CPU seconds}}{\text{Problem Program CPU seconds}}$,

μ_D = Assumed device service rate

$$= \begin{cases} 25 & \text{for} \quad 3330 \text{ type device} \\ 30 & \text{for} \quad 3350 \text{ type device,} \end{cases}$$

D = Number of I/O devices.

[9]Numbers in brackets indicate the references at the end of this paper.

4.3 Difference Performance Ratio Model

The difference performance ratio, denoted by SEMOD, is the difference in the internal performance ratios of the 3033/SE and the 3033/NON-SE. SEMOD is modeled as a function of two variables: (i) % CPU time in supervisor state on the 168-3 and (ii) start I/O issued per second on the 168-3.

The CPU time in the problem program state on a 3033 processor is not affected by the presence or absence of the SE feature. However, the CPU time in the supervisor state on a 3033 processor is reduced by the installation of the SE feature.

W. W. Everett of Bell Laboratories [3] has fitted the following model to describe the difference in the internal performance ratio between 3033/SE and 3033/NON-SE:

$$\text{SEMOD}_i = -0.0472 + 0.00131\ (\text{SIO}_i) + 0.00092\ (\text{SS}_i), \quad (4)$$

where SS_i is % CPU time in supervisor state on the 168-3 for job stream i, SIO is start I/O issued per second on the 168-3 for job stream i, and SEMOD is defined above. Model (4) was found to be an adequate fit.

4.4 Job Step Execution Time Models

Two models, one for migration planning based on 168-3 job step data and the other one for capacity planning based on 3033 job step data, are discussed in this section. Both of these models are developed from Application F data.

4.4.1 APPLAN168 for Migration Planning

Several regression models were fitted to describe execution time per 168-3 CPU second as functions of 168-3 job step data and estimated 3033 system performance measures. In these models, a job step is characterized by (i) EXCP per 168-3 CPU second and (ii) a qualitative measure of page reference characterisitics. 3033 system performance data consisted of (i) % CPU utilization, (ii) SIO per second, and (iii) total PAGING rate. The following model was found to be adequate:

$$\begin{aligned}\mathrm{APPLAN168}_{ij} = &-36.76 + 0.4169(\mathrm{CPUTIZ}_j)\\ &+ 0.2413(\mathrm{SIO}_j) - 0.0218(\mathrm{PAGRAT}_j)\\ &+ 0.1481(\mathrm{EXCP}_i) + 1.2012(\mathrm{PAGING}_i)\\ &- 0.0424(\mathrm{EXCP}_i)(\mathrm{PAGING}_i)\\ &+ 0.0002442(\mathrm{SIO}_i)(\mathrm{EXCP}_i)\\ &- 0.00277(\mathrm{CPUTIZ}_j)(\mathrm{SIO}_j), \qquad (5)\end{aligned}$$

where i = task number, $i = 1,2,3,4$;

j = Application F job stream number, $j = 1,2,\ldots,15$;

$\mathrm{APPLAN168}_{ij}$ = execution timer per 168-3 CPU second for task i in job stream j;

CPUTIZ_j = % CPU utilization for 3033 for job stream j;

SIO_j = SIO per second for 3033 for job stream j;

PAGRAT_j = total paging rate for 3033 for job stream j;

EXCP_i = number of EXCP per 168-3 CPU second for task i;

PAGING_i = a qualitative measure of paging derived from the average real memory owned for task i in 168-3.

The above model explains about 97 percent of the variability in APPLAN168 and it was found to be adequate. The predicted value of APPLAN168 is multiplied by the job step CPU time to estimate the job step execution time.

4.4.2 Model for Capacity Planning Aid

Section 4.4.1 discussed a model to estimate the execution time per 168-3 CPU second. This section discusses a model, which provides an estimate of the job step execution time per 3033 CPU second. The model described here can be used as an aid for capacity planning. Several regression models were fitted and the following model was found to be adequate:

$$\begin{aligned}\mathrm{APPLAN3033}_{ij} = &-77.24 + 0.8896(\mathrm{CPUTIZ}_j)\\ &+ 0.5081(\mathrm{SIO}_j) - 0.0463(\mathrm{PAGRAT}_j)\\ &+ 0.1350(\mathrm{EXCP}_i) + 2.7097(\mathrm{PAGING}_i)\\ &- 0.01414(\mathrm{EXCP}_i)(\mathrm{PAGING}_i)\\ &+ 0.0002864(\mathrm{SIO}_j)(\mathrm{EXCP}_i)\\ &- 0.0059(\mathrm{CPUTIZ}_j)(\mathrm{SIO}_j), \qquad (6)\end{aligned}$$

where i = task number, $i = 1,2,3,4$;

j = Application F job stream number, $j = 1,2,\ldots,15$;

$\mathrm{APPLAN3033}_{ij}$ = execution time per 3033 CPU second for task i in job stream j;

CPUTIZ_j, SIO_j, and PAGRAT_j are the same as in Model (5);

EXCP_i = Number of EXCP per 3033 CPU second for task i;

PAGING_i = a qualitative measure of paging derived from the average real memory owned for task i in 3033.

Model (6) explains about 96 percent of the variability in APPLAN3033.

4.5 Device Utilization Model

As discussed in Section 3, determination of the optimum number of initiators and DASD devices to make effective use of the 3033 processor is based on (i) the estimated 3033 CPU utilization and (ii) the estimated average device utilization. This section gives the development of an analytical model, called DEVUT, for the estimation of the average device utilization.

Let ρ_D denote the device utilization on the 3033. Under the characterization of the computer system described in Section 4.2, ρ_D is derived as follows:

ρ_D = (average service time at a device)

  • (average SIO rate per second for each device).

Since the average service time at a device is equal to $1/\mu_D$ and the average SIO rate per second for each device is equal to $\mu_C \cdot \rho_C / D$, where μ_D, μ_C, ρ_C and D are defined in Section 4.2,

$$\rho_D = \left(\frac{1}{\mu_D}\right)\left(\mu_C \cdot \rho_C \cdot \frac{1}{D}\right) = \frac{\rho_C}{\lambda D}. \qquad (7)$$

5. Summary

3033/PLAN, which consists of both analytical and empirical models, has been developed from benchmark performance data. The analytical models are used to derive CPU utilization and device utilization as functions of 168-3 or 3033 system performance measures.

Empirical models have been fitted to estimate internal performance ratios and job step execution times. The internal performance ratio models can be used to predict 3033 performance measures from 168-3 performance measures. Two job step execution models, one for migration planning based on 168-3 job step data and the other for capacity planning based on 3033 job step data, have been developed.

The migration planning program is useful for planning the migration from a 168-3 processor to a 3033 processor without any benchmarking. This program can also be used as a capacity planning aid for an existing 3033 processor through the use of projected 3033 system performance and job step data.

References

[1] E. G. Coffman, Jr. and P. J. Denning, Operating Systems Theory, 1973, Prentice-Hall, Inc.

[2] B. A. Ketchledge, Unpublished work.

[3] W. W. Everett, Unpublished work.

Table 1. Characteristics of the Benchmark Applications

Application	Workload Type	Program Type	Language	No. of Job Streams
A	Batch IMS	Real	PL/I	1
B	Batch	Synthetic	Fortran	1
C	Batch	Real	Cobol	1
D	Batch	Real	Cobol	1
F	Batch	Synthetic	Cobol	15

Table 2. Benchmark Performance Data (Applications A-D)

	APPLICATION A			APPLICATION B			APPLICATION C			APPLICATION D		
Character-istic	168-3 Base Exp. (1)	3033/SE Tuned Exp. (2)	Ratio (1)/(2)	168-3 Base Exp. (3)	3033/SE Tuned Exp. (4)	Ratio (3)/(4)	168-3 Base Exp. (5)	3033/SE Tuned Exp. (6)	Ratio (5)/(6)	168-3 Base Exp. (7)	3033/SE Tuned Exp. (8)	Ratio (7)/(8)
Stream Elasped Time	3041	1540	1.97	1402	860	1.63	2282	1605	1.42	1427	834	1.71
Total CPU Seconds	2956	1454	2.03	1361	769	1.76	1970	891	2.21	1418	694	2.04
CPU Utili-zation (%)	97.2	94.4		97.0	89.3		86.3	55.5		99.3	83.2	
Problem Program (PP) CPU seconds	2667	1344	1.98	1087	658	1.65	1291	615	2.09	851	471	1.80
PP CPU Utili-zation (%)	87.8	87.2		77.5	76.5		56.6	38.3		59.6	56.4	
Supervisor State (SS) CPU Seconds	288	110	2.62	273	110	2.48	679	276	2.46	567	223	2.55
SS CPU Utili-zation (%)	9.4	7.1		19.5	12.7		29.7	17.2		39.7	26.7	
SIO Rate/Sec.	32.1	63.6		65.8	102.4		128.0	179.0		132.8	217.0	
Total Paging per second	0.97	3.27		3.98	20.36		1.40	1.89		15.93	12.43	
Total No. of DASD Devices	10	10		12	12		12	18		12	16	
Total No. Of Initiators	4	4		8	16		8	8		10	18	
Millions of Instructions Per Second	2.96			4.68			2.45			2.25		

Table 3. Benchmark Performance Data (Application F)

	JOB STREAM 6			JOB STREAM 7			JOB STREAM 8			JOB STREAM 9		
Characteristic	168-3 Base Exp. (1)	3033/SE Tuned Exp. (2)	Ratio (1)/(2)	168-3 Base Exp. (3)	3033/SE Tuned Exp. (4)	Ratio (3)/(4)	168-3 Base Exp. (5)	3033/SE Tuned Exp. (6)	Ratio (5)/(6)	168-3 Base Exp. (7)	3033/SE Tuned Exp. (8)	Ratio (7)/(8)
Stream Elapsed Time	2895	1243	2.32	1178	739	1.59	1757	796	2.20	980	498	1.96
Total CPU Seconds	2887	1232	2.34	1081	520	2.07	1741	765	2.27	960	431	2.22
CPU Utilization (%)	99.6	99.3		92.0	69.3		98.9	96.1		97.9	86.2	
Problem Program (PP) CPU Seconds	2663	1158	2.29	811	407	1.99	1541	681	2.26	803	368	2.18
PP CPU Utilization (%)	91.9	93.2		69.0	54.3		87.7	85.6		81.9	73.9	
Supervisor State (SS) CPU Seconds	224	74	3.02	270	112	2.40	200	84	2.33	157	63	2.49
SS CPU Utilization (%)	7.7	5.9		23.0	15.0		11.3	10.6		16.0	12.6	
SIO Rate/Sec.	17.6	42.8		88.0	160.0		31.0	83.0		50.4	106.7	
Total Paging Per Second	0.61	4.41		1.32	9.33		2.10	35.30		12.40	13.50	
Total No. Of DASD Devices	12	12		12	12		12	12		12	12	
Total No. Of Initiators	8	12		8	12		8	12		8	12	
Millions Of Instructions Per Second	2.06			2.5			2.12			2.28		

Table 4. Task Characteristics for Application F

Task	No. of DASD Files	Total No. of EXCP	CPU Seconds		EXCP/CPU Second	
			3033/SE	168-3	3033/SE	168-3
TASK1 (CPU)	2	600	21.68	50.87	27.67	11.79
TASK2 (I/O)	3	1800	6.31	12.37	285.20	145.50
TASK3 (PAGE)	2	600	11.75	26.81	51.06	22.37
TASK4 (BALANCED)	2	600	5.25	11.25	114.28	53.30

Table 5. Job Stream Composition for Application F

Job Stream	% of TASK			
	1	2	3	4
1	25	25	25	25
2	55	15	15	15
3	15	55	15	15
4	15	15	55	15
5	15	15	15	55
6	85	5	5	5
7	5	85	5	5
8	5	5	85	5
9	5	5	5	85
10	45	45	5	5
11	45	5	45	5
12	45	5	5	45
13	5	45	45	5
14	5	45	5	45
15	5	5	45	45

Table 6. Summary of Application F Job Stream Data

Job Stream	%CPU Utilization	SIO/SEC	Pages/ Sec.	Mean[a] & Std. Dev. of Exec. Time/10 TASK1	TASK2	TASK3	TASK4
1	98.44	96.68	13.41	19.86	14.12	16.00	8.46
				2.07	1.88	2.10	1.02
2	99.36	59.60	6.42	23.73	13.54	19.25	10.28
				1.58	0.89	1.80	1.01
3	95.41	130.77	11.46	14.41	14.59	12.14	7.60
				1.08	2.41	1.37	1.32
4	99.04	88.99	18.88	19.42	13.13	16.27	8.62
				1.87	1.23	1.35	1.05
5	97.74	101.56	11.35	18.93	13.17	15.13	8.37
				1.51	1.06	1.39	1.18
6	100.00	42.83	4.41	26.15	14.65	22.30	12.55
				1.70	1.20	2.25	0.21
7	72.97	159.77	9.33	13.20	14.85	11.10	9.60
				0.56	1.36	1.21	2.80
8	98.12	83.67	35.31	19.96	13.25	15.75	8.95
				0.85	0.49	1.50	0.35
9	95.04	106.70	12.46	16.90	12.80	15.46	7.60
				0.00	3.11	0.99	0.72
10	98.61	90.88	6.60	21.71	14.49	16.83	10.10
				2.50	1.23	2.05	0.00
11	99.59	57.93	11.98	22.87	13.10	18.70	10.50
				1.43	1.99	2.10	0.30
12	99.51	62.73	6.21	22.74	14.50	20.03	10.03
				1.71	0.14	1.36	1.13
13	97.31	129.21	18.67	16.63	13.60	12.95	8.43
				1.76	1.56	1.73	0.57
14	86.91	141.96	10.44	13.70	14.20	11.26	7.27
				1.69	1.99	2.26	1.29
15	98.97	96.74	21.20	18.96	12.95	14.84	8.35
				1.11	0.63	1.04	0.65

a. For each job stream the mean and standard deviation are given on the first and second lines, respectively.

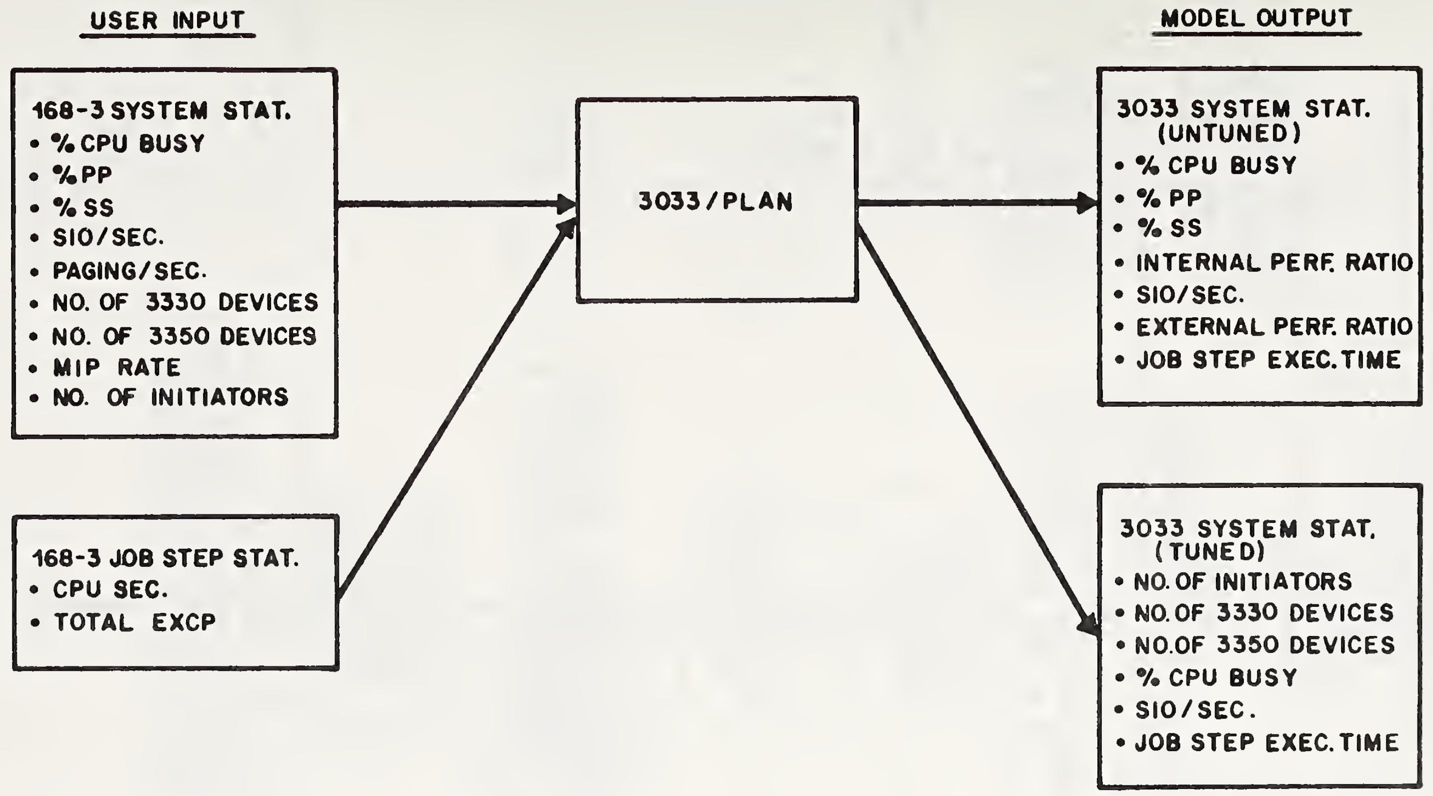


Figure 1. 3033 Migration Planning Aid System Flow

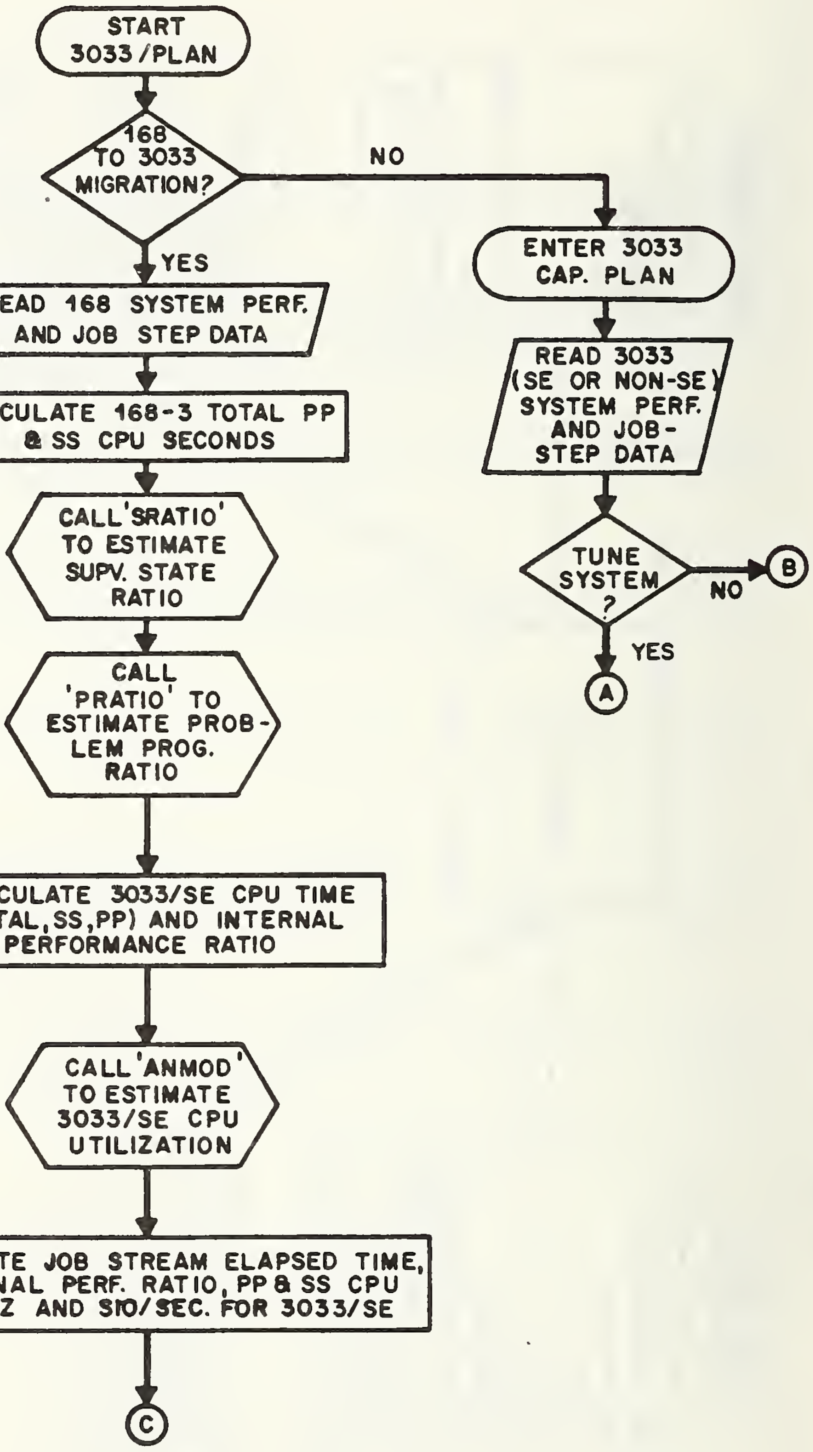

START 3033/PLAN
168 TO 3033 MIGRATION?
NO
YES
READ 168 SYSTEM PERF. AND JOB STEP DATA
CALCULATE 168-3 TOTAL PP & SS CPU SECONDS
CALL 'SRATIO' TO ESTIMATE SUPV. STATE RATIO
CALL 'PRATIO' TO ESTIMATE PROBLEM PROG. RATIO
CALCULATE 3033/SE CPU TIME (TOTAL, SS, PP) AND INTERNAL PERFORMANCE RATIO
CALL 'ANMOD' TO ESTIMATE 3033/SE CPU UTILIZATION
CALCULATE JOB STREAM ELAPSED TIME, EXTERNAL PERF. RATIO, PP & SS CPU UTILIZ AND SIO/SEC. FOR 3033/SE
C
ENTER 3033 CAP. PLAN
READ 3033 (SE OR NON-SE) SYSTEM PERF. AND JOB-STEP DATA
TUNE SYSTEM ?
NO
B
YES
A

Figure 2. 3033/Plan System Flow Chart

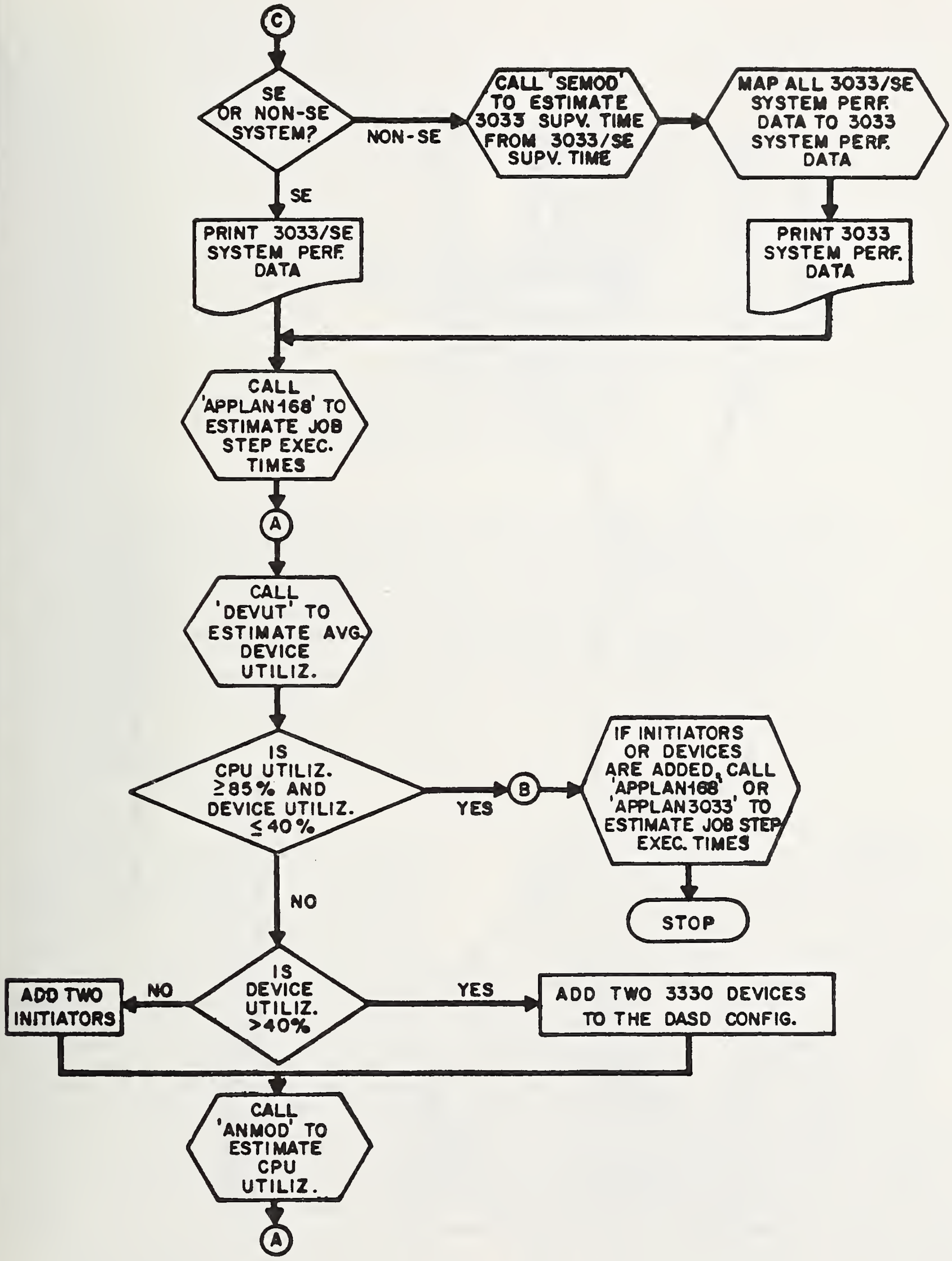
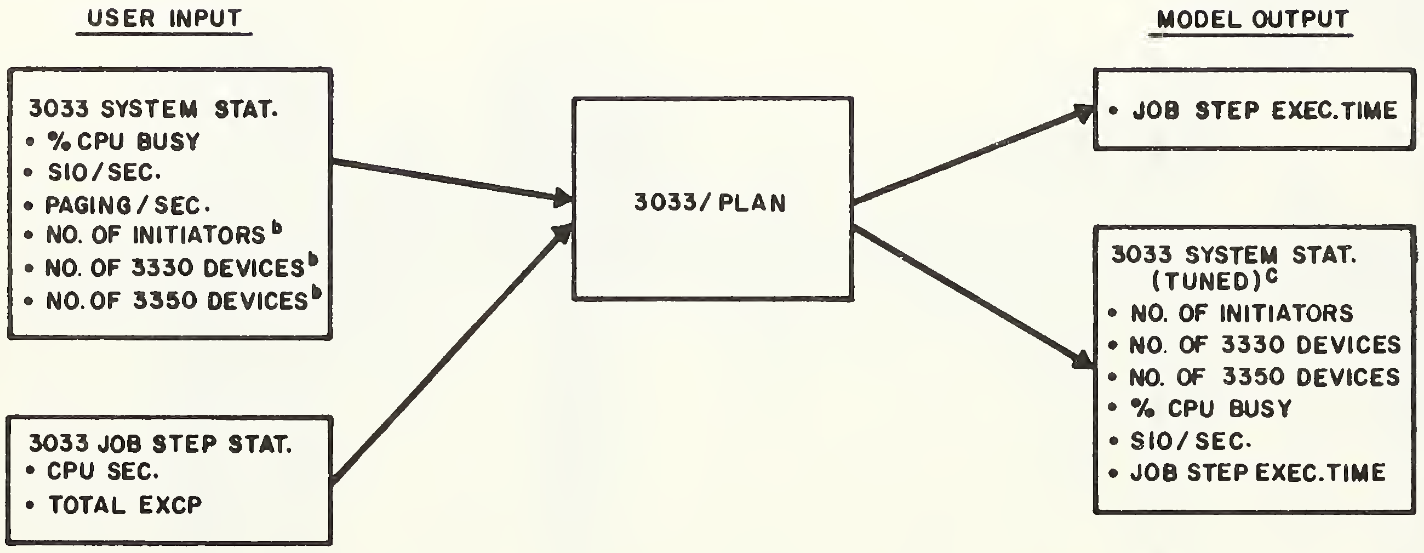

C
SE OR NON-SE SYSTEM?
NON-SE
CALL 'SEMOD' TO ESTIMATE 3033 SUPV. TIME FROM 3033/SE SUPV. TIME
MAP ALL 3033/SE SYSTEM PERF. DATA TO 3033 SYSTEM PERF. DATA
SE
PRINT 3033/SE SYSTEM PERF. DATA
PRINT 3033 SYSTEM PERF. DATA
CALL 'APPLAN168' TO ESTIMATE JOB STEP EXEC. TIMES
A
CALL 'DEVUT' TO ESTIMATE AVG. DEVICE UTILIZ.
IS CPU UTILIZ. ≥85% AND DEVICE UTILIZ. ≤40%
YES
B
IF INITIATORS OR DEVICES ARE ADDED, CALL 'APPLAN168' OR 'APPLAN3033' TO ESTIMATE JOB STEP EXEC. TIMES
STOP
NO
IS DEVICE UTILIZ. >40%
NO
ADD TWO INITIATORS
YES
ADD TWO 3330 DEVICES TO THE DASD CONFIG.
CALL 'ANMOD' TO ESTIMATE CPU UTILIZ.
A

Figure 2 (Cont'd). 3033/Plan System Flow Chart



b. REQUIRED IF TUNING OPTION IS ELECTED
c. AVAILABLE UNDER TUNING OPTION

Figure 3. 3033 Capacity Planning Aid System Flow

AN OPTIMAL SAMPLE SIZE ALLOCATION SCHEME FOR BENCHMARK DESIGN

Satish K. Tripathi
Karen D. Gordon
Ashok K. Agrawala

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

A major problem in benchmark design is the selection of the jobs to compose the benchmark. In this paper, stratified sampling is applied to the problem. The strata in the job population are identified by clustering the jobs on the basis of job features. The clusters are then viewed as strata. Within a stratum, jobs are selected by simple random sampling. The question of how many jobs to select from each stratum is addressed in this paper. An extension of Neyman's result on the optimal allocation of sample points among strata to the multidimensional case is used. The technique is then applied to a real workload, and the results are examined.

Key words: benchmarking; clustering; performance evaluation; stratified sampling.

1. Introduction

Benchmarking is a commonly used technique for evaluating the performance of computer systems [Benw 75]. In this approach, a set of jobs (or job steps) which can run on the system to be evaluated is collected or generated. The set of jobs, which is referred to as a "benchmark," is submitted to the computer system as a test workload, and the performance of the system is measured. If the same benchmark can be run on several computer systems, their performance can be compared, making this technique well suited for computer acquisition problems. However, in order to generalize the results of a benchmark study to a system handling its natural workload, the benchmark has to be "representative" of the natural workload handled (or to be handled) by the system.

The natural workload of a computer system is the composite of all processing requests submitted by its user community. Current large-scale computer systems typically accept a wide variety of requests from a large number of users. As a result, their natural workloads are very complex. Each processing request, or workstep (which is defined to be the smallest unit of work being modeled), can be characterized by the quantitative demands it makes on certain system resources: hardware (e.g., CPU, memory, I/O devices, etc.) or software (e.g., compilers, editors, file utilities, etc.). The variability of the worksteps can be expressed by defining a multivariate probability distribution of resource usage for the workstep population [Ferr 72, AMB 76, SK 74, Arti 76]. This multivariate distribution then characterizes the natural workload of the computer system. Clearly there is no reason to believe that such a distri-

This research was supported in part by the NASA, Goddard Space Flight Center, under Grants NASA#5-24092 and NASA#5-24407, and by the General Research Board of the University of Maryland.

bution will have any nice parametric form or that it will be a highly clumped distribution. Studies [SK 74, AMB 76] have shown that such a distribution tends to be a multimodal distribution.

These multimodal distributions can be expressed as mixtures of several unimodal distributions. In order to use the mixture distribution model, however, we need to be able to decompose the original multimodal distribution into its component unimodal distributions. For the nonparametric situation of computer system workloads such a decomposition can be achieved using clustering techniques [Ande 73, Hart 75, AMB 76]. Clustering techniques accept the observed population sample as input and proceed to divide the sample into groups of "similar" observations. Each group or cluster should have a unimodal distribution, a component of the original multimodal distribution.

The aim of a computer system evaluation study is to estimate certain system performance measures for the system executing its natural workload. When using the benchmarking methodology, a set of worksteps is executed on the actual system. The basic steps in designing a benchmark are:

1. selecting a set of worksteps that is "representative" of the real workload, and
2. determining how the actual running of the worksteps should be carried out.

At first glance it might seem that the second step is trivial; i.e., that one should input all the worksteps at once. This might have been a reasonable solution for a uniprogrammed batch environment. For today's systems, however, a careful design of the sequencing of the worksteps is essential. It might also be desirable to run some of the worksteps more than once in order to reproduce the job mix of the natural workload. While the importance of the second step to the overall evaluation process cannot be emphasized enough, in this paper we address ourselves to the problem of selecting the worksteps to be used in the test workload.

In selecting a set of worksteps which is representative of the system's natural workload one is faced with the problem of obtaining a suitable sample from the workstep population (or distribution, in the case of a synthetic workload). The techniques of statistical sampling can be used for this purpose [Coch 77]. It should, however, be noted that most of the sampling techniques reported in the literature are for univariate distributions.

A straightforward approach to sampling is that of simple random sampling. It can clearly be used for sampling from a multivariate distribution. To obtain desired levels of confidence, however, a rather large sample is required. The confidence in the sample can be improved by going to the stratified sampling technique [Coch 77].

In order to use stratified sampling, the population is divided into a set of identified strata. A number of sample points are then selected from each stratum to form a global sample. Determining the sample size for each stratum in a stratified population is an important problem in using stratified sampling. Methods based upon minimizing the cost of the sampling process [Coch 77], minimizing the variance of the population mean [Neym 34], and satisfying a set of constraints on the cost function and the variances of the sample means of various strata have been proposed [Chat 72].

There has been very little work towards extending these results for stratified sampling from a multivariate distribution. Some heuristics have been proposed for specific applications, but no formal theory has been developed [Coch 77]. The problem of multidimensional stratified sampling and optimal allocation of sample points among strata has been formulated and solved in [AGT 79]. In this paper we present a scheme that uses the multidimensional stratified sampling technique in selecting the worksteps for a benchmark. Multidimensional stratified sampling is briefly described in Section 2. In Section 3 we show how multidimensional stratified sampling can be used in the design of benchmarks by considering the workload handled by the UNIVAC 1100/40 system at the University of Maryland.

2. Multidimensional Stratified Sampling

Let us first consider a univariate stratified population with L strata. Let $\overline{x1}, \overline{x2}, \ldots, \overline{xL}$ be the sample means for the L strata based upon samples of size n1, n2, ..., nL respectively. Assume that the total population size is N with N1, N2, ..., NL being the sizes of the corresponding strata. Let $n = \sum_i ni$.

Define

$$\overline{x}st = \frac{\sum_i Ni\,\overline{xi}}{N}$$

Suppose that one is interested in estimating E[X], the population mean. $\overline{x}$st is an unbiased estimate for E[X]. If the objective is to estimate E[X] as accurately as possible, $\overline{x}st$ should have the smallest possible variance. For a given sample size n, Neyman obtained the values of the ni's such that the variance of $\overline{x}st$ is minimized [Coch 77]:

$$(A) \quad ni = n \frac{NiSi}{\sum_i NiSi}$$

where Si is the standard deviation for the ith stratum. Equation (A) is known as Neyman's optimal allocation.

Following the same notation, let

$$<Xi> = \begin{bmatrix} Xi1 \\ Xi2 \\ \cdot \\ \cdot \\ \cdot \\ Xik \end{bmatrix} \quad \text{and} \quad <\overline{x}i> = \begin{bmatrix} \overline{x}i1 \\ \overline{x}i2 \\ \cdot \\ \cdot \\ \cdot \\ \overline{x}ik \end{bmatrix}.$$

where Xij is a random variable representing the value of the jth variable in the ith stratum, and $\overline{x}ij$ is the sample mean of the jth variable in the ith stratum. We assume that there are k variables.

Let Mi denote the variance-covariance matrix for the ith stratum.

That is,

$$Mi = \begin{bmatrix} v[Xi1] & cov[Xi1,Xi2] & \cdots & cov[Xi1,Xik] \\ \vdots & & & \vdots \\ cov[X_{ik},X_{i1}] & cov[X_{ik},X_{i2}] & \cdots & v[Xik] \end{bmatrix}$$

In practical situations one wants to estimate various means ($\overline{x} \cdot j = \frac{\sum_i Ni\ \overline{x}ij}{N}$) with (possibly) varying degrees of accuracy. In other words, one might be interested in getting an estimate for the mean of $<\alpha^T> <X>$, a linear combination of x.j's:

$$\alpha_1 X\cdot 1 + \alpha_2 X\cdot 2 + \ldots + \alpha_k X\cdot k$$

For a given $<\alpha>$ (weight vector) we want to obtain the optimal allocation of sample sizes for each stratum. It has been proved in [AGT]79 that if $Y = <\alpha^T><X>$ and $\overline{y}st = \sum_i (Ni<\alpha^T><\overline{x}i>)/N$, then $V[\overline{y}st]$ is minimized for a fixed total sample size of n if

$$ni = n \frac{Pi\sqrt{<\alpha^T> Mi <\alpha>}}{\sum_i Pi\sqrt{<\alpha^T> Mi <\alpha>}}$$

where Pi = Ni/N.

The above result implies that more observations are needed from a stratum where the variables are very highly correlated than from a stratum where the correlation is low.

Also, fewer observations are needed from a stratum where the variables have very high negative correlations than from a stratum where the correlations have very low values.

3. An Example: Design of a Benchmark

As noted in Section 1, the workload of a computer system can be described by an appropriate multivariate distribution. In order to study the applicability of the multidimensional stratified sampling technique to the problem of benchmark design, we examine the workload handled by the Univac 1100/40 computer system at the Computer Science Center of the University of Maryland. This system is one of two systems used by the faculty and students for educational and research computing. The system is accessible through batch and interactive facilities and usually carries a heavy load.

The workloads of several different days (spec., November 7-10, 1977) are analyzed using the data available from the system accounting logs. A job is treated as a workstep. Each workstep's resource requirements are quantified in terms of the following features:

1. number of programs executed
2. number of SUP's (Standard Units of Processing*)
3. mean number of core blocks
4. CPU time
5. ER and CC charges (Executive Requests and Control Card)
6. number of words transferred to/from disk
7. number of words transferred to/from drum

In general, when each workstep is described as a 7-dimensional vector, the workload is characterized by a multivariate distribution in those 7 dimensions. The distribution is typically multimodal and can be expressed as a mixture distribution in which each component is a unimodal distribution [AMB 76]. For the nonparametric case of a computer system work-

*For a detailed description of these quantities see [MAF 76].

load, such a mixture distribution can be decomposed into its components using clustering techniques [ANDE 73, Hart 75]. We use the technique of [AMB 76] in this paper.

First, one of the workloads is selected as a base workload. Second, the feature values of all the jobs in the base workload are scaled in the following way: for each feature, its minimum value is scaled to 0 and its 98th percentile value to 10. For SUPS, CPU, ERCC, DISK, and DRUM, the values input to the scaling process are actually logarithms of the true values. The purpose of logging and scaling the feature values of jobs is to make the clustering of jobs more meaningful [AMB 76]. Third, the jobs in the base workload are clustered according to their scaled feature values. Fourth, the feature values of the jobs in the other workloads are scaled according to the same scaling factors used for the base workload. Finally, the jobs in the other workloads are clustered according to the clusters found for the base workload. In particular, a job is placed into a cluster if it is within a certain distance of the mean of the cluster in the base workload and if no other cluster mean is closer.

Table 1 presents a summary of the clustering results for November 9, 1977. The results for the other days are similar. The table gives the number of jobs in each cluster. It also gives the means and variances of the feature values of the jobs by cluster. It should be noted that the values in the table are scaled values.

An interesting application of clustering is to use the clusters as strata for multidimensional stratified sampling. Since each cluster has samples which are "similar," it contains points with small variances and hence makes a good candidate for a stratum. The optimal sample size for samples from each stratum remains to be chosen.

Tables 2a and 2b compare the proportional and optimal allocations for the workloads of four days in November 1977. The total sample size is 100. For the optimal allocation cases, equal weighting of features is assumed: ie., $<\alpha^T> = (1,1,1,1,1,1,1)$. The tables show that the two strategies yield significantly different allocations. However, the allocations for the different days are very similar. For the four workloads, the variance of the estimate of the mean of $<\alpha^T> <X>$ is 8-16% smaller under optimal allocation than under proportional allocation. The improvement of stratified simple random sampling with optimal allocation over simple random sampling is 26-42%.

Table 3 gives the optimal allocations associated with several different weight vectors for the November 9 workload. As expected, different weight vectors yield different allocations.

Figure 1 gives a graphical comparison of proportional allocation and two optimal allocations, one with $<\alpha^T> = (1,1,1,1,1,1,1)$ and the other with $<\alpha^T> = (0,0,0,1,0,0,0)$ (i.e., all weight given to CPU time). Consider clusters 1 and 2, which represent 29.4% and 37.4% of the jobs in the workload. For optimal allocation with $<\alpha^T> = (1,1,1,1,1,1,1)$, the number of points allocated to cluster 1 is larger than for proportional allocation, and the number allocated to cluster 2 is smaller. An examination of the variance-covariance matrices for clusters 1 and 2, given in Tables 4a and 4b, reveals why this is so. All entries in the matrix for cluster 1, except some associated with CPU time (the entries marked with an asterisk), are higher than the corresponding entries in the matrix for cluster 2.

For optimal allocation with $<\alpha^T> = (0,0,0,1,0,0,0)$, the clusters with a relatively small CPU variance are allocated fewer points than for proportional allocation, while those with a larger variance are allocated more points. The CPU variances for the clusters of November 9 are as follows: .09, 1.55, 1.29, 4.77, .95, 5.86, and 7.71. As a result, clusters 1 and 5 are allocated fewer points, and clusters 2,4,6,7 are allocated more points. Cluster 3 is allocated approximately the same number of points.

4. Concluding Remarks

In this paper we have presented a scheme for selecting the worksteps to comprise a benchmark. The scheme is based on multidimensional stratification and thus takes into account the fact that each workstep has to be characterized as a multidimensional vector. Recognizing that the benchmark experiment is to be conducted to estimate the values of some system performance measures, the relative importance of the components of the vector used to describe a workstep is taken into account by a weight vector which is used in deriving the optimal allocation of sample points to different strata in the workstep population. The sampling within each group can be done using simple random sampling.

Table 1. Clustering Results for November 9, 1977

Cluster	# jobs	Feature values [$\frac{\text{mean}}{\text{variance}}$]						
		# progs	SUPS	CBS	CPU	ERCC	Drum	Disk
1	976	.45	3.05	2.45	.20	3.05	2.29	2.26
		.12	3.17	5.04	.09	3.31	2.35	6.03
2	1242	.83	5.26	4.89	1.96	4.41	7.83	3.21
		.06	.38	.88	1.55	.39	.21	2.52
3	168	4.15	7.92	3.02	2.63	8.02	7.33	7.63
		2.86	.29	1.02	1.29	.44	3.12	1.47
4	454	.30	4.08	8.93	1.87	3.06	3.38	5.22
		.03	1.77	3.28	4.77	1.10	4.97	2.15
5	295	1.84	6.31	2.86	1.25	6.38	5.34	6.31
		.59	.36	1.15	.85	.45	2.62	1.59
6	74	1.83	8.34	4.79	8.17	6.70	7.03	6.95
		.96	.90	4.13	5.86	1.03	3.39	3.86
7	102	10.35	9.92	3.87	6.11	9.77	9.58	9.31
		33.38	.62	2.00	7.71	.89	2.54	2.05

Table 2a. Proportional Allocation of 100 Sample Points

Date	Cluster1	Cluster2	Cluster3	Cluster4	Cluster5	Cluster6	Cluster7
11/7/77	30.3	38.1	5.4	10.4	10.8	1.3	3.2
11/8/77	29.2	37.2	4.8	11.6	11.5	2.0	3.4
11/9/77	29.4	37.4	5.1	13.7	8.9	2.2	3.1
11/10/77	30.6	38.2	6.1	10.5	8.0	2.2	4.2

Table 2b. Optimal Allocation of 100 Sample Points

Date	Cluster1	Cluster2	Cluster3	Cluster4	Cluster5	Cluster6	Cluster7
11/7/77	48.2	23.9	4.4	12.2	5.8	1.3	4.2
11/8/77	41.9	26.1	3.4	15.7	5.9	1.7	5.3
11/9/77	42.4	27.2	3.8	14.1	5.5	2.3	4.7
11/10/77	41.3	28.1	4.6	12.8	5.0	2.5	5.6

Table 3. Optimal Allocations of 100 Sample Points for November 9, 1977
Alphas=(# progs, SUP5, CB5, CPU, ERCC, Orum, Oisk)

Alpha	Cluster1	Cluster2	Cluster3	Cluster4	Cluster5	Cluster6	Cluster7
1111111	42.4	27.2	3.8	14.1	5.5	2.3	4.7
0111111	43.2	27.9	3.2	14.8	5.4	2.3	3.2
0112211	40.6	29.0	3.3	15.7	5.4	2.4	3.7
1000000	17.8	15.9	15.1	3.9	12.0	3.9	31.4
0100000	49.3	21.8	2.6	17.1	5.0	2.0	2.3
0010000	44.2	23.5	3.4	16.6	6.4	3.0	2.9
0001000	7.8	41.2	5.1	26.4	7.2	4.8	7.5
0000100	50.6	22.1	3.2	13.6	5.6	2.2	2.7
0000010	36.1	13.6	7.2	24.4	11.5	3.3	3.9
0000001	40.6	33.4	3.5	11.3	6.3	2.5	2.5

Table 4a. Variance-Covariance Matrix for Cluster 1 on November 9, 1977

	# progs	SUP5	C8S	CPU	ERCC	Orum	Oisk
# progs	.12						
SUP5	.47	3.17					
C85	.13	.72	5.04				
CPU	.02*	.14*	.35	.09*			
ERCC	.49	3.14	.24	.07*	3.31		
Orum	.34	1.48	2.37	.17	1.39	2.35	
Oisk	.47	3.21	.98	.29*	2.75	1.23	6.03

Table 4b. Variance-Covariance Matrix for Cluster 2 on November 9, 1977

	# progs	5UP5	C8S	CPU	ERCC	Orum	Oisk
# progs	.06						
5UP5	.05	.38					
CB5	.08	.05	.88				
CPU	.05*	.66*	-.08	1.55*			
ERCC	.05	.28	.03	.27*	.39		
Orum	.03	.12	.22	.14	.05	.21	
Oisk	.10	.57	.62	.57*	.58	.31	2.52

* Entries marked with an asterisk are the ones whose values are higher for cluster 2 then for cluster 1.

Clearly no implication about the importance of a group can be made from the size of a sample taken from that group. For example, if a group represents 90% of the population and has worksteps with extremely small variances, a sample size of one might be sufficient. That one workstep may, however, have to be replicated several times to achieve an appropriate benchmark.

The design of the actual benchmark experiment, we believe, plays a very crucial role in the success of a study. Much more work is necessary in that direction.

We gratefully acknowledge the assistance supplied by the staff of the University of Maryland Computer Science Center, especially Ira Gold, in obtaining the data used for this report. Helpful comments were also received from Jeff Mohr of the University of Maryland and Tony Chu of the University of Toronto.

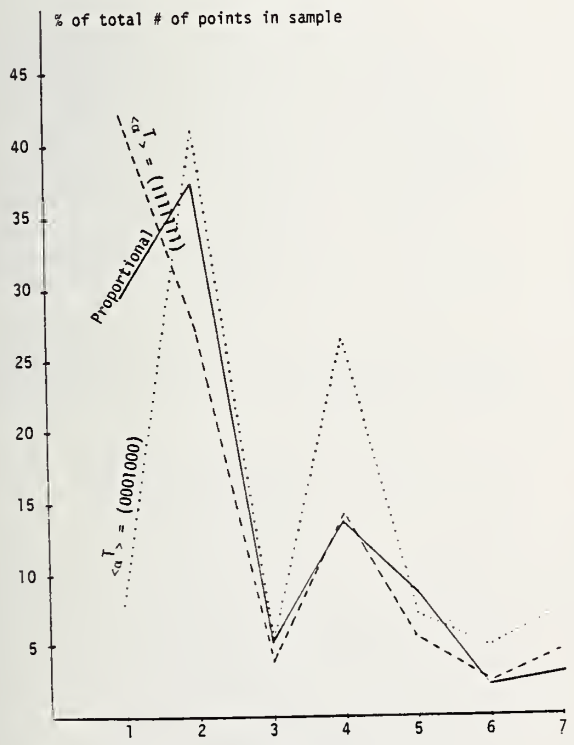


Figure 1. Allocations for November 9, 1977

References

[AGT 79] Agrawala, A.K., Gordon, K.D., and Tripathi, S.K., "Multidimensional Stratified Sampling in Benchmark Design," University of Maryland, Department of Computer Science, Technical Report TR-750, March 1979.

[AMB 76] Agrawala, A.K., Mohr, J.M., and Bryant, R.M., "An Approach to the Workload Characterization Problem," Computer, Vol. 9, No. 6, June 1976, pp. 18-32.

[Ande 73] Anderberg, M.R., Clustering Analysis for Applications, Academic Press, New York, 1973.

[Arti 76] Artis, H.P., "A Technique for Determining the Capacity of a Computer System," CPEUG Proc. of the 12th Meeting, November 1976, pp. 150-162.

[Benw 75] Benwell, N. (ed.), Benchmarking: Computer Evaluation and Measurement, Hemisphere Publishing Corporation, Washington, D.C., 1975.

[Chat 72] Chatterjee, S., "A Study of Optimum Allocation in Multivariate Stratified Surveys," Skand. Akt., Vol. 55, 1972, pp. 73-80.

[Coch 77] Cochran, W.G., Sampling Techniques, 3rd ed., John Wiley & Sons, New York, 1977.

[Ferr 72] Ferrari, D., "Workload Characterization and Selection in Computer Performance Measurement," Computer, July/August 1972, pp. 18-24.

[Hart 75] Hartigan, J., Clustering Algorithms, Wiley, 1975.

[MAF 76] Mohr, J.M., Agrawala, A.K., and Flanagan, J.F., "The EXEC-8 Log System, Part I - Description," University of Maryland, Department of Computer Science, Technical Report TR-434, January 1976.

[Neym 34] Neyman, J., "On the Two Different Aspects of the Representative Method: The Method of Stratified Sampling and the Method of Purposive Selection," Journal Royal Stat Soc., Vol. 97, 1934, pp. 558-606.

[SK 74] Screenivasan, K. And Kleinman, A.J., "On the Construction of a Representative Synthetic Workload," CACM, Vol. 17, No. 3, March 1974, pp. 127-133.

COMPUTER WORKLOAD FORECASTING

James E. Mc Neece

Office of Information Systems Research & Development
Federal Bureau of Investigation
Washington, DC 20535

Experience has shown that the successful initiation of computer workload forecasting is directly dependent upon the quality of the data furnished by users to the requesting organization. Inaccurate data almost inevitably causes delays in the documentation cycle. The purpose of this paper is to serve as a guide for performing the analysis required to forecast workload requirements. Application of the methodology suggested herein should significantly reduce the risk of inaccurate or misleading projections.

1. Introduction

The approach described has been developed on the foundation of experience in performing the analysis required to support major procurements of data processing resources. This paper is intended as a guide; only the framework of the total analysis required is defined. A good measure of creativity on the part of the team assembled to perform the analyses is required in each instance.

Insofar as possible, this guide is organized in a "how to" fashion, describing the following steps of the analysis:

a. Defining Data Requirements
b. Developing Workload Projections
c. Sizing Equipment
d. Performing Sensitivity Analysis.

Under the initial section, the types of data elements required to support the analysis will be defined and their sources will be listed. Responsibility for maintaining and organizing the necessary data will be defined and fixed organizationally. The role of the user in the forecasting effort will be clarified.

In order to project accurately, the quantitative resource requirements of user production applications and applications under development must be estimated and projected over the life cycle of the system(s). Additionally, the performance characteristics which are required for these applications must be identified. (For example, response time requirements must be specified for interactive systems.) Finally, all functional capabilities required of the system must be substantiated. (For example, user requirements for a data base management system or a transaction processor must be defined in detail).

2. Defining Data Requirements

The data required to develop user workload forecasts must be supplied in part by the computer centers and in part by the user. Because this data will be used to develop statistically based workload forecasts, historical monthly data for at least two years is desirable.

These information requirements are specified in detail below. As will be apparent upon review of these data elements, not all the information requirements listed will be obtainable from computer resource accounting and billing data. Those that may not be available from these sources are indicated and must be relatable to the billing data. An example of such an information requirement that is extremely important would be the one for transactions or quantifiable events on a monthly basis.

Information which is not available through the billing and accounting data source must be maintained by the user. In the case of transactions or quantifiable events data related to "major" application systems, the user will be asked to meet with Center personnel on an individual basis to agree on the nature of the quantifiable events to be recorded on a monthly basis for each major system. Thus, when the time comes to produce workload forecast support documentation, the Center and the user will have at hand the data required to validate user information processing requirements.

All of the data elements listed below should be available on a monthly basis for existing workload.

2.1 Information Requirements to be Generated on a User Basis

The data elements falling in this category describe computer resource utilization of the user. The requirements of the Center itself must also be isolated and identified. As mentioned above, this data will be retained by the Center, though user guidance may be necessary to separate production accounts from development and maintenance accounts.

A. Character Storage Data, DASD

1. Space allocated by device type (measured in tracks, cylinders, blocks, sectors, characters, etc. -- as appropriate to the Center).

2. Space used by device type.

B. Program Development and Maintenance Resource Requirements

1. Aggregated CPU time required.
   a. Development Work
   b. Maintenance Work

2. Number of man-months of programmer/analyst time required.
   a. Development Work
   b. Maintenance Work

3. Number of Compile Jobs Run.
   a. Development Work
   b. Maintenance Work

C. Character Storage Data, Tape Media.

1. Number of tape reels assigned

2. Number of tape reel mounts performed

D. Resource Requirements for Production Work.

1. CPU time required

2. Core Requirements (measured in Kilobyte minutes)

3. EXCP's (or other relevant measures of I/O activity) by device type

4. Total Monthly Connect Time

5. Second Generation Computer Workload (wall clock hours)

E. Telecommunications Workload.

1. Number of lines read

2. Number of lines transmitted

3. Average number of characters read per read activity

4. Average number of characters transmitted per write activity

2.2 Information Requirements to be Generated on a "Major Application)

For purposes of this data collection requirement, any single application system which has consumed or is expected to consume 10 percent of the computer resources billed to the user on an annual basis will be defined as a "major" application. This definition breaks down when a user workload is fragmented in such a fashion that the "major" applications account for less then 50 percent of total annual resource requirements. Therefore, where this condition occurs, the user should choose its largest applications for reporting purposes such that aggregate requirements of these applications equal or exceed 50 percent of the total.

As in the case of information to be maintained on a user basis, these time series data will be retained by the Centers on a monthly basis. Users will be required to identify "major application" accounts and to police utilization of those accounts.

A. "Transactions" or "Quantifiable Units" Workload.

B. Processing Resources Required (for each, list by batch and interactive mode).

1. Billable CPU time required

2. Core requirements

3. EXCP's (or other relevant measures of I/O activity) by device type

4. Total monthly connect time (interactive/transactions processing type activities)

C. Telecommunications Workload.

1. Number of lines read

2. Number of lines transmitted

3. Average number of characters read per read activity

4. Average number of characters transmitted per write activity

D. Data File Requirements (list by each separate file which supports the major application).

1. File Media

2. Average number of records contained in the file

3. Average record length in characters

4. Data Set Organization

5. Number of data set accesses

6. Identification of all files which require personal and corporate data.

2.3 Information Requirements to be Generated on a "Planned Application"

All of the data listed above must be maintained on a monthly basis to describe the resource requirements of existing systems. At the time of documenting the workload forecast, users will additionally be required to forecast resource requirements of planned applications. The required data elements for each major new application to be developed and implemented during the system life will be as follows:

1. Projected monthly transactions workload

2. Projected monthly CPU time required:
Interactive/Batch

3. Projected character storage requirements, by file

a. File media
b. Average record length in characters
c. Number of records to be contained in the file
d. Data set organization
e. Data set accesses by month

4. Projected telecommunications Workload

a. Monthly connect time (for interactive jobs)
b. Number of lines read monthly
c. Number of lines transmitted monthly
d. Average number of characters read per read activity
e. Average number of characters transmitted per write activity

2.4 Information Requirements to be Generated on a "Center" Basis

Finally, a monthly time series of center performance indicators must be maintained by the Center in order that the capacity of the existing computer system may be estimated and users' expectations on system performance characteristics established. These data requirements are as follows:

1. Average turnaround time for batch jobs.

2. Average guaranteed turnaround time for batch jobs.

3. Average response time for interactive applications (where applicable).

4. Measure of CPU active time by shift (derived by an established sampling technique).

3. Developing Workload Forecasts

In order to predict quantitiative user workloads, three projection techniques are applied and the results are cross-checked:

a. Trend regressions are developed for all resource utilization time series.

b. Transactions-based forecasts of user resource requirements are developed using linear regression techniques.

c. User forecasts of new applications' resource requirements are aggregated through time over the complete period of the system life.

Each of these techniques is described below:

3.1 Trend Regression for Resource Utilization Time Series

The first forecasting technique involves a gross projection on Center resource requirements. Figure 1 represents the framework of this analysis. Time is the independent variable; computer resource utilization is the dependent variable. Such an analysis is performed for each computer resource identified in the time series data collected (e.g., billable CPU time, DASD character storage requirements, monthly connect time requirements).

3.2 Transactions-Based Forecasts of User Resource Requirements

In order to accomplish the "Transaction based" forecasts of user resource requirements a three step analysis is performed. The forecasts should be developed for each "major" application described.

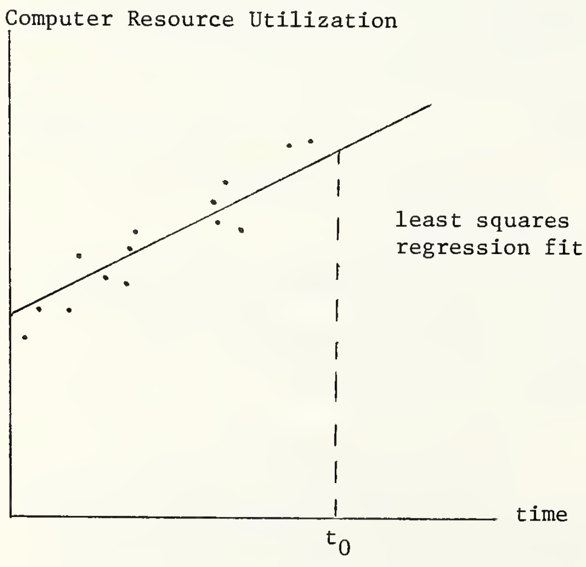


Figure 1. Trend Regression Technique

1. In the first step a regression analysis is performed using time as the independent variable and transactions volume as the dependent variable.

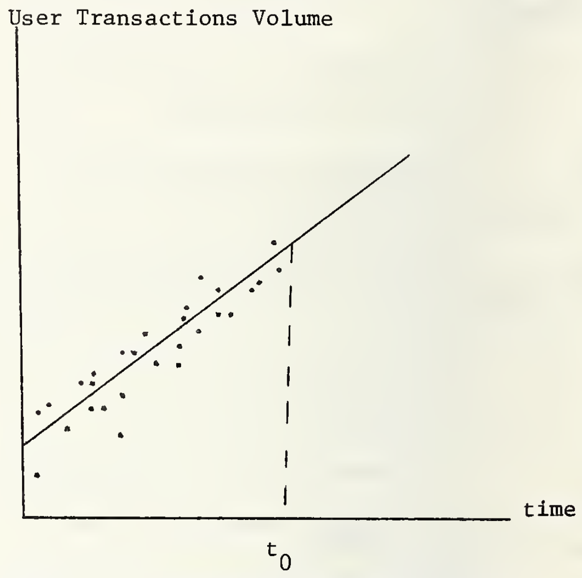


Figure 2 Transactions Based Technique

Figure 2 represents this analytical step; a least squares regression technique is applied. If necessary or desirable, the long run secular trend can be isolated from seasonal fluctuations by using a moving average time series.

2. The second step of the analysis requires a set of linear regression fits. In this case, however, the independent variable is user transaction volume; the dependent variable is computer resource utilization. A separate regression should be performed for each of the following dependent variables. Figure 3 represents one such regression, with transaction volume represented as the independent variable and computer resources required as the dependent variable.

a. Data Set Accesses

b. Aggregated Character Storage Requirements (measured as "tracks used" by the computer centers and as as "characters stored" by the user)

c. Aggregated telecommunications workload. Separate analyses could be performed for remote batch, interactive/conversational, etc., workloads if aggregated totals appear to justify such consideration

d. Monthly Connect Time

e. Core requirements, measured in kilobyte minutes

f. Tape reel mounts

Figure 3 represents one of these regression results, with data set accesses shown to be related directly to the monthly transactions volume.

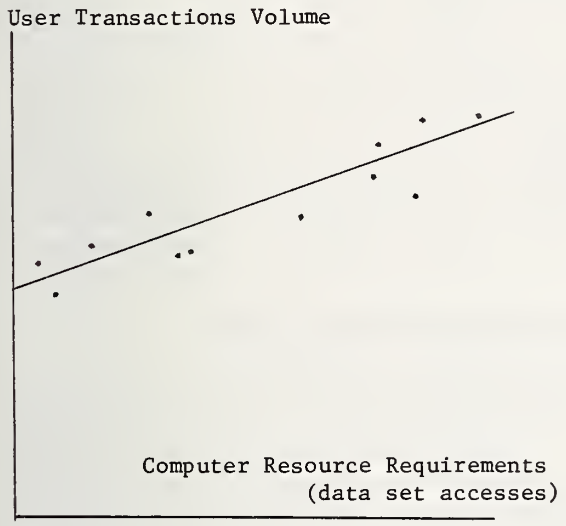


Figure 3 Transactions Based Techniques

3. The final step of the analysis is to translate the projected growth rate of transactions, derived in the step one trend analysis, into a projection of resource requirements over the system life cycle. Figure 4 demonstrates this process graphically. As can be seen, the historical transactions data is used to forecast the growth of user workload. The projected transaction volumes by year are then related to a forecasted computer resource requirement using the regression result derived in step 2 of the analysis.

Aggregating related resource requirements for all "major" applications results in a projection for each measure of production workload. To this sum must be added the workload generated by all minor and ad hoc production applications as well as the projected applications development and maintenance workload. The minor and ad hoc production workload may be estimated using the Trend Regression technique described above. (Figure 1). Development and maintenance workload may be estimated as follows:

1. A linear regression is performed which relates computer resources utilization for application system development and maintenance work (dependent variable) to the number of man-months devoted by user personnel to programmer/analyst efforts (independent variable). Experience has shown this relationship to be direct, as indicated in Figure 5.

2. As a check on this projection technique, a two-step regression methodology is applied. First, the time series of "man-months of programmer/analyst effort" is regressed against the time series of monthly total compile jobs run. This latter time series is then regressed against the time series of computer resource utilization. The analysis is depicted in Figure 6:

Having projected production and developmental computer resource requirements of the users, the analysis proceeds by aggregating these projected requirements, adding in the overhead resource requirements identified for the computer center itself.

3.3 Aggregated User Forecasts of Applications Resource Requirements

The third technique which is applied to estimate workload growth over the system life period is non-statistical. Users are asked to describe all new production applications which are expected to be developed and implemented during the period. Resource requirements estimates are then provided for each application, both for its development and for running it in production mode. Next the users are asked to describe the stability of the existing base workload. In other words, any existing applications which are expected to be expanded or phased down or out during the system life are identified and their resource requirements specified. Having a schedule of both increments and decrements to the base through time, the analyst can then develop the user's forecasted growth path. This forecast would then be compared with those derived in the first two techniques. If serious discrepances exist, these must be resolved within the group performing the analysis.

4. Equipment sizing

In this step of the analysis, a number of alternative computer configurations capable of processing the projected workload must be specified. Generally, the

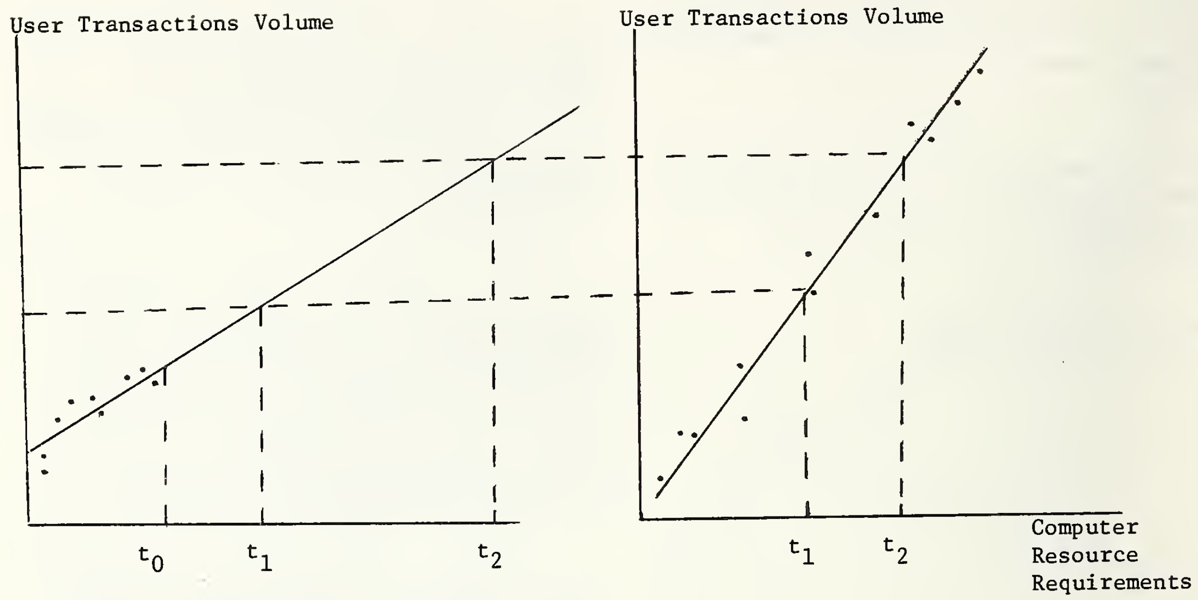


Figure 4 Derivation of Computer Resource Requirement Projections

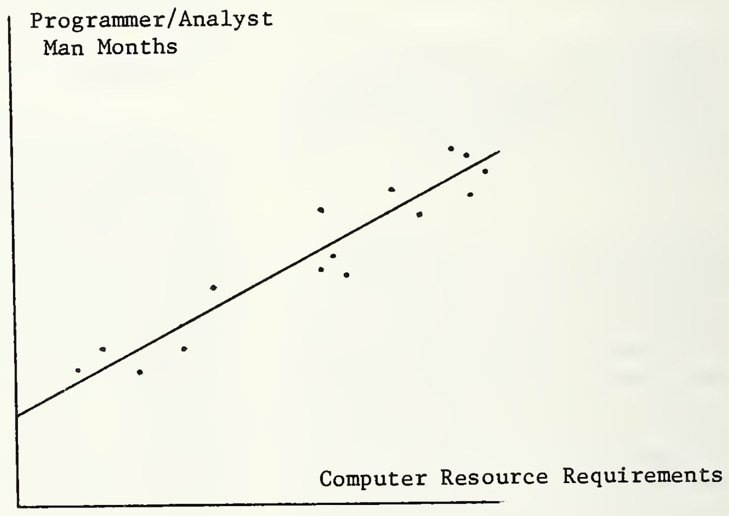


Figure 5 Estimated Relationship Between Programmer Man Months and Computer Resource Requirements

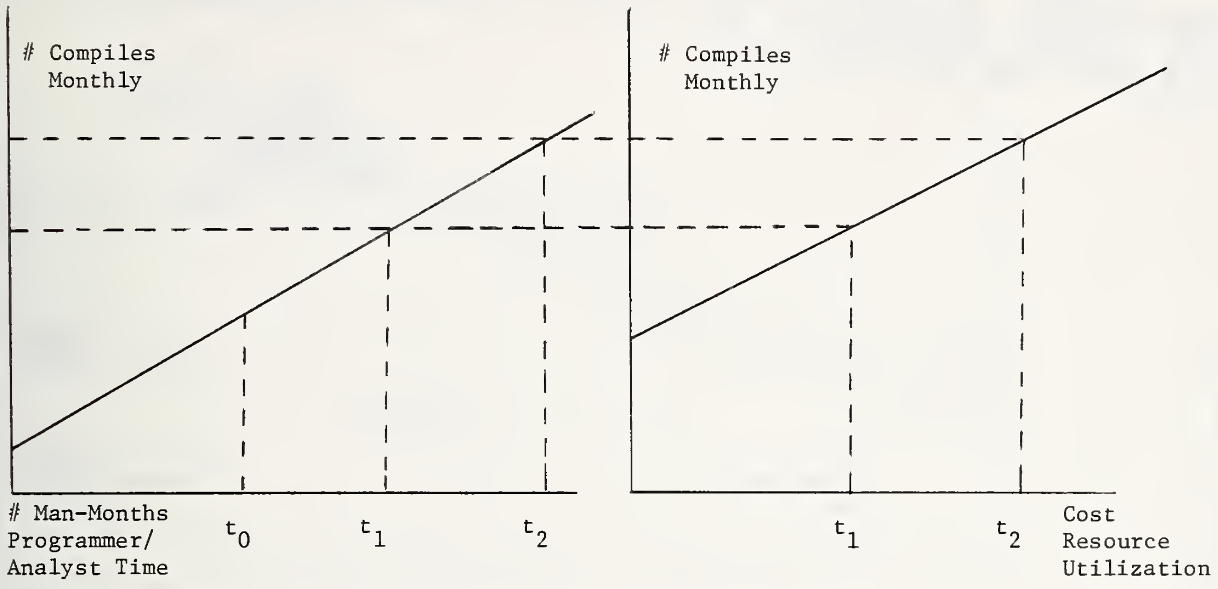


Figure 6 Derivation of Development Time Workload Forecast

sizing analysis will place emphasis on CPU-memory requirements and DASD character storage requirements since the system components which perform those functions generally account for 80 percent or more of equipment costs.

Sizing DASD configurations to required on-line character storage volumes is a straightforward process. Variables which may be considered in developing actual equipment include recording density, data transfer rate, and physical size specifications.

Sizing processor-memory configurations may be somewhat more difficult. The approach which is generally most fruitful for this purpose requires some benchmarking to compare the relative processing power of existing equipment with the capabilities of some mainframes which are judged to be in the approximate size range necessary to process the workload.

This operation may be constrained by inserting the requirement that at no time during the system life will an excess demand situation be tolerated. If such a constraint is not imposed, the cost analysis must quantify the impact of capacity saturation on users as a cost item.

Figure 7 indicates only one system capacity projection (a case in which one augmentation of equipment is expected to occur early in the systems life). The analysis should actually develop several capacity alternatives, which are then compared in a costing model.

For the one system capacity projection depicted in Figure 7, the costs of excess capacity and excess demand are represented. Excess capacity costs are implicit in the equipment costs and will not be treated explicity in an economic analysis. Excess demand costs, on the other hand, must be treated explicity. These costs are estimated in the economic analysis by attempting to forecast the cost of procuring necessary computer resources in a commercial service bureau environment. If excess demand is planned, this estimation method will probably yield quite accurate results. However, if workload exceeds the forecasted volumes and unplanned excess demand occurs, this estimation method will probably seriously underestimate excess demand costs.

5. Sensitivity Analysis

In the sensitivity analysis section of the documentation, all critical assumptions made in the body of the analyses must be "tested." The objective is to determine the limits within which quantitative alterations can be made.

The assumption which must be examined in this section is that the workload forecasts are accurate.

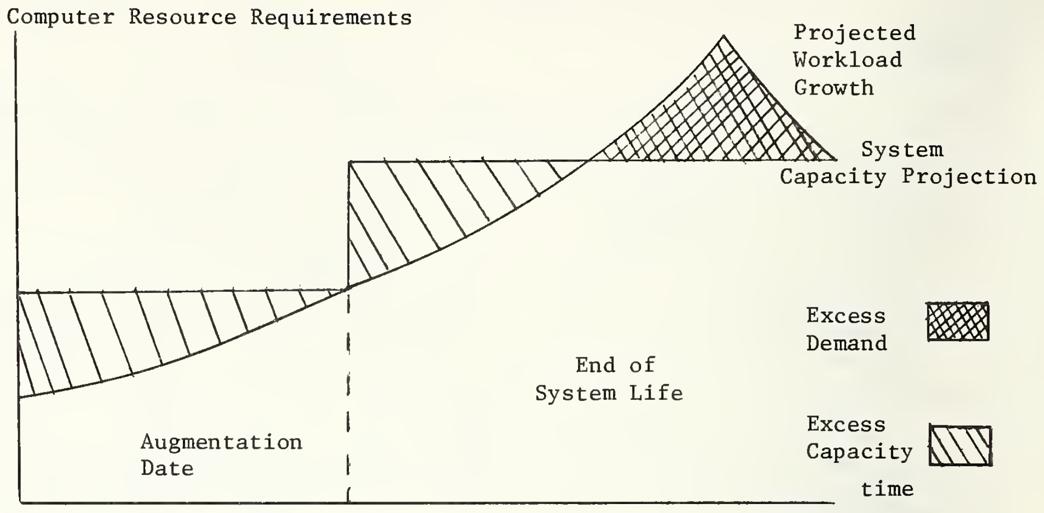


Figure 7 Fitting Capacity to Projected Workload Requirements

5.1 Sensitivity to Variance in Future Workload

This assumption may be tested by observing the impact of increasing or decreasing the projected workload volumes by given proportions. Additionally, the possibility that each workload measure was initially overestimated by as much as 50 percent should be considered. If the ordering of alternatives is not changed at either of these limits of the range, the conclusion may be drawn that the recommendation of the analysis is not sensitive to changed workload volume expectations.

5.2 Sensitivity to Variance in Actual Resource Requirements

The analyst must consider the possibility that the estimate of resources required to process the predicted workloads is either high or low. Again, any resource which comprises ten percent or more of the total system cost should be examined. (Generally, this will require focus on the area of personnel, computer equipment and perhaps telecommunications services.)

In testing the sensitivity of the analystical results to over or under estimation of resource requirements, the approach should parallel that taken in testing resource cost sensitivity. The magnitude of the percentage change in resource requirements needed to change the ordering of altertives should be reported to management.

5.3 Sensitivity to Variance in Ad hoc Assumptions

Finally, the sensitivity analysis should investigate the possible impact of changing any ad hoc assumptions made in the analysis which could effect cost.

Formatting the sensitivity analyses is of critical importance to supporting management evaluation of the forecast. In the case of each assumption tested, the original assumption should be stated clearly. Following this, the degree to which the quantitative aspect of the assumption must be changed before the ordering of alternatives is also changed must be reported.

6. Conclusion

This paper has summarized the methodology required to produce computer workload forecasts. Clearly, the circumstances surrounding each individual effort are unique and as such require alternations to the basic approach. However, if the data collection program described in the first section of the paper has been successfully maintained, the analyst should have minimal difficulties in adapting to the unique aspects of his particular problem.

CPE Experiences

A SIMULATION MODEL OF JES OUTPUT PROCESSING

H. Pat Artis

Bell Laboratories
Piscataway, New Jersey 08854

The design and implementation of a discrete simulation model of the IBM Job Entry Subsystem (JES2 and JES3) output processing is presented. This model was developed for sizing printers at remote stations and at the central site.

1. Introduction

The sizing of printers for IBM Job Entry Subsystem (JES2 and JES3) remote work stations is currently[1] a problem of particular interest. The proliferation of remote work stations has increased the complexity of the sizing problem and mandated the need for a sizing model. This paper presents a model that was developed to size printers for remote work stations. The paper discusses:

- model selection,

- data collection,

- model design,

- sample problem.

Each of these topics will be discussed in detail in the following sections.

1. A technique for identifying and measuring the effects of poorly sized remote stations has been presented in a paper previously published by the author.[1]

2. Model Selection

The first step in the modeling process was the selection of a queuing or a simulation model of the problem. This selection was influenced by the nature of a remote work station. The following list of attributes will provide the reader a general understanding of the environment.

- A remote work station may have one or more printers (servers). In the event that there are multiple servers, they need not be all of the same speed.

- The load applied to a remote station is often transient in behavior. Moreover, the servers may be totally saturated during periods of peak load.

- At all but trivial load levels, it has been the author's experience that the arrivals demonstrate exponential behavior. However, the service times are generally distributed for all load levels.

- A multi class (limited delay dependent priority) queuing discipline is used by the operating system to schedule work for the servers. The queuing discipline is also non pre-emptive.

The well known [2] difficulties associated with M/G/m models, transient and saturated behavior, and multi class priority schemes led the author to select a trace driven simulation model. Although the aforementioned problems can be solved [3], the simulation model was most attractive because of the ease of implementation and limited number of events to be simulated.

3. Data Collection

The data for the modeling effort was collected from the Systems Management Facility (SMF) [4] log file normally produced by the MVS operating system. In particular, the JES Job Purge (Type 26) and the JES Output Writer (Type 6) record types were used as sources of data for the model. The JES Job Purge record contains the following items used by the data collection program that generates input for the simulation model:

- JES Job Number,
- Job Name,
- Account Code,
- Job Class,
- Execution End Time and Date, T_1.

Since a job may spool multiple output files, there may be more than one JES Output Writer record for each job. The record for the i th output file for a job contains:

- JES Job Number,
- JES Output Device,
- Number of Logical Records Printed,
- Output Class,
- Output Forms,
- Write Start Time and Date, $T_{(2,i)}$,
- Write End Time and Date, $T_{(3,i)}$,

The Type 26 and all of the Type 6 SMF records for a job may be assembled using the JES job number as a key.

The data collection program produces a report with the following statistics for each printer collected during the study period specified by the user. The statistics for each printer are:

- average lines per minute (LPM),
- a list of the output classes served,
- a list of the form types used,
- the total lines printed,
- number of jobs printed,
- utilization during the study period.

An output file is produced containing one record for each print file that arrived for output processing (i.e., the job that created the print file ended execution) during the study interval. Each of these records contain the following items:

- Output Device,
- Job Name,
- Job Class,
- Output Class,
- Account Code,
- Output Form,
- Arrival Time, T_1,
- Actual Output Queue Time, $T_{(2,i)}-T_i$.

The Job Name and Account Code entries are provided to allow the user to track jobs of particular interest or to select portions of the current print load to use in sizing a new remote.

4. Model Design

The discrete simulation model was implemented in PL/1. A flowchart of the model is shown in Figure 1. The principal modules in the model are:

LOAD ARRIVALS	The module reads the trace file produced by the data collection program. An exit is provided to the user to allow subsets of the print load to be selected as a load for a simulated remote,
INITIALIZATION	This module loads the user input control data that specifies the number of printers at the remote and their respective speeds. Information on the output classes served and form types available for each printer. It also sets the clock and statistics collection arrays to their initial values,

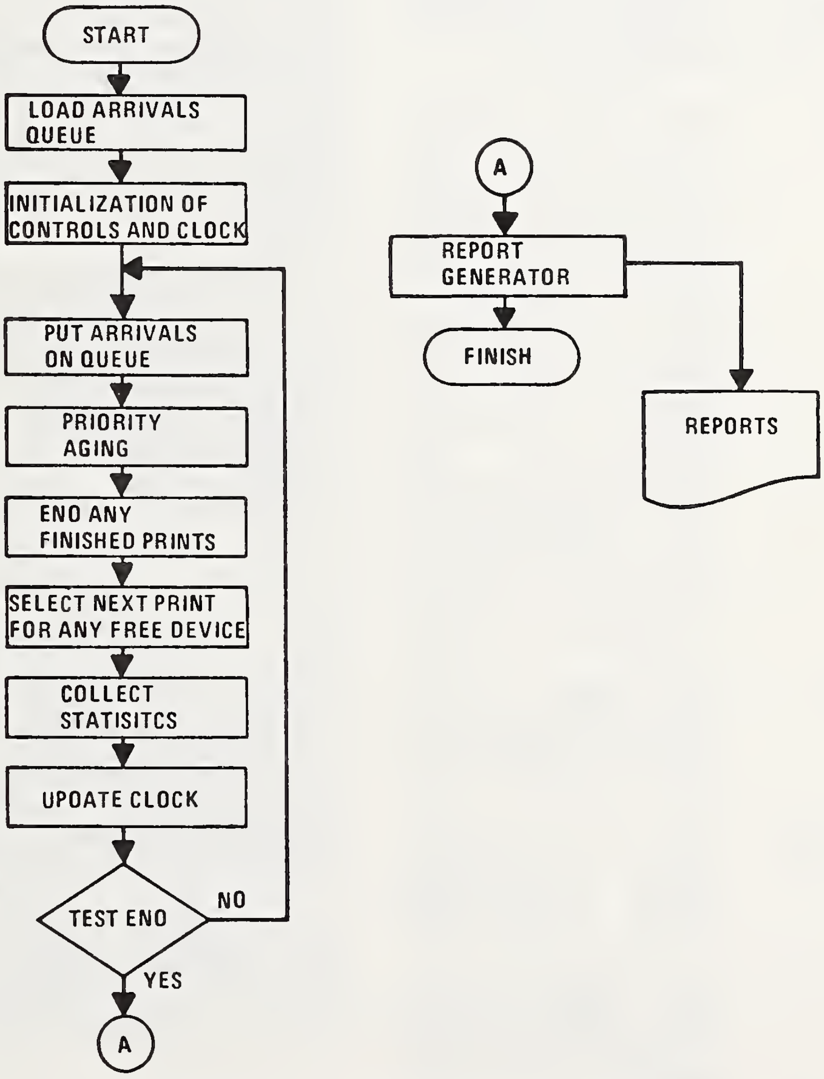


Figure 1. Simulation Flow Chart

ARRIVALS	This module simulates arrivals by taking all "printouts" that arrive at the current clock value and placing them on queues that are maintained by output class, form type and priority. A user exit is provided to allow the user to select the initial priority for the printout,
PRIORITY AGING	This module provides a user exit to inspect the contents of the output queues and age the priorities of the waiting printouts,
SELECTION	This module attempts to schedule a printout on each free printer. It selects the highest priority printout in the current class-forms queue being served by the printer. When a printout is selected, an end event for the printer is scheduled based on the printer's speed and the number of lines in the printout. An exit is provided to allow the user to implement form changing strategies,
STATISTICS	This module collects statistics on printer utilization, queue lengths and wait to print times,
UPDATE CLOCK	This module updates the simulation clock. A one second clock step is used for this simulation.
REPORTS	This module summarizes and reports on the statistics collected by the program.

Typical run times for the simulation are from five to thirty CPU seconds on a 370/168 for an eight hour study period. To date, the longest running time has been fifty seconds for a heavily loaded remote with six printers.

5. Sample Problem

The model discussed in the previous section was used to evaluate the potential upgrade of a small remote work station with a 300 LPM printer serving a group of programmers. This study was also used to validate the model.

The data collection program provided the statistics shown in the following table about the printer's actual use during a four hour period. This study period was carefully selected such that the printer was idle with a queue length of zero at both the beginning and of the interval. This selection was made to avoid end effects in the validation of the model.

Actual Printer Statistics	
Avg Utilization (%)	78.9
Avg Wait Time (sec)	268
Standard Deviation (sec)	405
Maximum Wait (sec)	3125
Lines Printed	46,742
Files Printed	94
Device Effective LPM	246

As shown in the previous table, the printer's effective print rate was 246 LPM although it was rated at 300 LPM. The differences between a printers rated and effective LPM values is caused by variations in the number of characters per print line, use of non-preferred print characters and vertical spacing of the printer. It has been the author's experience that a printer's effective print rate rarely exceeds 90 percent of its rated value.

To validate the model, a single 246 LPM printer (server) was specified for the simulation model. The results of the simulation are shown in the following table.

Simulation Statistics (246 LPM)	
Avg Utilization (%)	79.5
Avg Wait Time (sec)	265
Standard Deviation (sec)	408
Maximum Wait (sec)	3168
Lines Printed	46,742
Files Printed	94

As can be seen, the results of the simulation and the actual measured data compare favorably. The model was then used to evaluate the upgrade of printer to a device rated at 1100 LPM. The effective rate of the device, 900 LPM, was estimated by multiplying the device's rated speed by the ratio of the effective and rated speeds of the 300 LPM printer. The results of the simulation are shown in the following table.

Simulated Statistics (900 LPM)	
Avg Utilization (%)	21.6
Avg Wait Time (sec)	14
Standard Deviation (sec)	32
Maximum Wait (sec)	358
Lines Printed	46,742
Files Printed	94

As shown in the table, the utilization of the device decreased by the ratio of the old and new printers effective rates. However, the high speed provided a significant reduction in the average and maximum queuing delay since the faster server was not subject to long periods of saturation.

6. Comments

The simulation model presented in this paper provides a useful tool for sizing printers for remote work stations. One issue not addressed by this paper is the capacity of the host processor and transmission line to support higher speed output devices. The problems of buffer size and line capacity are best approached by queuing models. The reader may wish to consult some of these models already available in the literature [2,5].

REFERENCES

[1] Artis, H.P. "Workflow, A Technique for Analyzing JES Systems", published in the AFIPS Conference Proceeding, vol. 48, 1979.

[2] Kleinrock, L. "Queuing Systems, Vol. II", John Wiley & Sons, 1976.

[3] Roode, J. D., "The Operational Analysis Approach to Computer System Modeling", Symposium on Computer Resource Performance Management, Pretoria, South Africa, April 1979.

[4] "Systems Management Facility", IBM, July 1977, EC28-0706.

[5] Allen, A.O., Probability, Statistics and Queuing Theory, Academic Press, 1978.

DESIGN FOR PERFORMANCE

Michael J. Kirrene

AVCO Financial Services
620 Newport Drive
Newport Beach, CA 92660
and
Mitchell G. Spiegel**

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

Expensive on-line systems are difficult to justify to top management -- unless your competitors or contemporaries are using them to advantage.

In 1971, AVCO Financial Services (AFS), a subsidary of the AVCO Corporation, set out to construct an on-line system for its consumer credit operation. Funding was approved in 1972 and the design project formally began in January 1973. The first pilot branch office was converted in November 1975, and the conversion of the last branch occurred in May 1977. This is a history of how major strategic performance decisions were successfully made for AVCO's on-line system.[1] AVCO's process of strategic performance decision-making, "design for performance," encompasses present ideas about capacity planning and performance evaluation.

AFS employed a variation of the conventional approach to the system design methodology -- a process of prototyping the system and its interface with the organization in every design phase to focus on the performance issues.

Key words: On-line system design, prototyping, performance management, management control, capacity planning, long-range planning, financial applications, performance evaluation, audit, measurement, modeling, remote terminal emulation, system testing.

1. Introduction

Technological advances cause continual changes to today's complex information systems. Performance management of information systems has evolved from an operational discipline to a tactical management tool -- capacity planning. But the emphasis of capacity planning performance tools is directed at multiprogramming of primarily production batch systems. On-line applications are a secondary objective. The view of performance is inward, concentrating

The views and conclusions contained in this paper/presentation are the authors, and should not be interpreted as representing the official opinions or policies of the Federal Computer Performance Evaluation and Simulation Center, the US Air Force, or of any other person(s) or agency associated with the Federal Government.

**Mr Spiegel acted as a consultant to AVCO for a previous employer.

on efficiency and the management and allocation of scarce computer resources among users. Performance, or its lack, is discovered after implementation. However, the on-line user has a different perspective: 'If it doesn't perform, it doesn't work!' Performance issues permeate the system life cycle in an on-line environment.

Performance characteristics of on-line systems are indigenous to and are decided in the design phases. Once strategic performance decisions are made, it is usually difficult, if not impossible, to obtain improvement. A system methodology is required that will support strategic performance decision-making. Conventional system design consists of a linear sequence of phases that usually stand alone. To work effectively, it must be possible for the design team (with or without the user) to establish firm system requirements and specifications (e.g., consider the procurement of a system, using a benchmark). Seldom are the results of a benchmark demonstration used to advantage in prior or subsequent phases of the conventional system design process. If a performance problem is discovered, the design team must loop back to an earlier design phase and repeat all steps to bring performance back to specification. Excessive loopiness caused by frequent changes in requirements or by design inadequacies is disasterous to the on-line system design process.

Good on-line system designs benefit from prototyping to make strategic performance decisions. Prototyping is a compromise strategy between the pure linear approach and the loopy linear approach.[2] Phases are interdependent, each generating sufficient depth of performance information to accommodate inevitable changes in specification without complete redesign. The user is part of the design team. Users have prototypes of system features built for evaluation and also participate in design decisions. Results of vendor demonstrations are used to develop prediction standards[3] for subsequent phases (e.g., test, acceptance, and operation).

The AVCO case study demonstrates the many facets of on-line system performance and the prototype system methodology used to obtain that performance. A brief overview of consumer finance orients the reader to the application. Each major step taken by AVCO in the design phase is described with emphasis on performance implications, strategic performance design, and design methodology. Observations taken from the operational system are contrasted with design expectations.

2. Case Study Background

2.1 The Nature of the Business

Consumer finance is one of the oldest retail trades in the United States, having begun in the middle of the nineteenth century. Today there are over 18,000 licensed consumer finance offices regulated and supervised by state law. AVCO Financial Services (AFS) is a division of the AVCO Corporation and is the third largest consumer finance company in the United States. AFS operates over 1,100 consumer finance offices nationwide.

Finance office procedures follow a daily pattern. Mail payments are posted in the morning to account ledgers. Across-the-counter collections build to a mid-day peak, while delinquent accounts are followed up by letter and phone. Customers in need of a loan or other services arrive sporadically through the entire day. At closing time, branch office statistics and summaries are produced for individual branch managers, supervisors, and auditors.

Application for credit is initiated when a customer arrives at a branch office. Following a successful credit check, an appropriate type of note is consummated. The whole transaction requires about 45 minutes. Once a loan is consummated, most types of transactions take place by mail and phone. Successful loan-making is measured by the <u>accuracy</u> of and <u>speed</u> in consummating a loan contract, and maintaining up-to-date status on the customers' account. Home office control of the branch office operation is accomplished by periodic checks and grading by supervisors and review of accounts by internal auditors.

2.2 Development of an Information Service

AFS's major business objective was to change the orientation of branch operations from accounting to financial consulting and marketing of various financial services by implementing an information system. Branch profit margins were tight under the manual operation. Growth could be achieved only by streamlining clerical operations and enhancing the ability of branches to handle additional lines of business and volume increases without adding personnel.

The information system was to provide better control over assets and revenues and improved customer service (accuracy and speed of consummating a loan and maintenance of up-to-date records). All information system management functions were physically co-located with the headquarters manual organization operations.

3. Concept of Operation

3.1 Design Strategy

AFS management concluded that early 1970's on-line system technology required a highly centralized systems approach. Such future directions as distributed processing and data bases could be integrated into the operation when they became proven technologies. Automation of the most important business services took place first. After the management and staff gained experience with the initial service capability, more sophisicated and complex service automation was attempted.

Because personnel salaries were rapidly overtaking equipment costs, the system was developed in a high level language -- COBOL was selected. It was assumed that this would reduce both development and ongoing maintenance costs.

Software needs determined host system requirements. Because of the long-range plan for a complex automation service and the state of the art of control system software, OS/VS2 was selected as the operating system and CICS was selected as the monitor. It was predicted that future changes in computer technology would improve hardware cost effectiveness. Therefore, it was decided to avoid modifications to system programs. Any special requirements would be satisifed via vendor defined user exits.

3.2 System Architecture

The automated system is called the Branch Operating System (BOS). Simply put, the Branch Operating System (BOS) is a communications network composed of a terminal at each branch office connected to a central computer at AFS Headquarters. At Headquarters, the system maintains up-to-date files from which reports are prepared for management's use in supervising branch operations. The system automates routine business transactions performed by the branch offices.

The system supports over 1,000 on-line terminals and requires an IBM S/370 Model 3033 with 6 million bytes of memory. The daily transaction rate varies between 100,000 and 170,000 transactions per day. BOS consists of approximately 600 programs and 600,000 source statements. Over 90% of the programs are written in COBOL. The data base has over two billion bytes stored on IBM 3330's. A COMTEN 476 communications processor controls the network, and performs message switching.

Cost justification of BOS was based upon cost avoidance, primarily through reduced personnel costs. No attempt was made to quantify the intangible benefits of faster, more detailed, and more reliable information. The break-even point to recover the estimated development cost of $18.3 million was initially projected to occur about three years after conversion. The ultimate cost of the system was $24.4 million.

To ensure orderly development, the system was planned in four design phases. The first design phase took ten months and cost approximately $1.5 million. It included the system design and specifications of the software needed to operate the system. Additional phases were undertaken only after the first design design phase was successfully completed. A feasibility study preceeded the four design phases.

4. Assembling the Design Team

4.1 Management Participation

The project was given maximum visibility by having the project director report to the Vice President of the Financial Services Division. A technical administrator reported directly to the project director, to be responsible for project planning, technical design monitoring and system integration activities. A position of techncial documentation supervisor was created to centralize the definition of documentation requirements and implement, establish, and maintain the BOS document library. Once the system became operational, an information systems division was formed, reporting directly to the President of the Financial Services Division. The system division consists of over 250 staff members.

4.2 User Participation

Success in meeting user needs depended heavily upon the accuracy of the information and the thoroughness of the cooperation provided by users who were called upon to assist in the system development. Dedicated user participation was achieved in most cases by transferring people from their existing duties to the project payroll and physically locating them within the project team.

The nature of user responsibilities varied from one user group to another, as well as from one phase of the project to another. In general, however, the users participating in the project were required to (1) define their requirements clearly and in detail; (2) furnish information to project personnel on request; (3) modify and develop branch and home office staff and procedures as required to support BOS development and operation; (4) assist in the development of BOS manuals, forms, and documents; (5) review and physically sign-off on all system specifications at various points throughout the project's life; (6) notify BOS project personnel of all changes in laws, regulations, policies, or Company plans that add, delete, or modify BOS requirements; (7) assist in developing branch conversion and installation plans; and (8) physically execute acceptance tests.

Several problems emerged from this approach. Several of the user representatives turned out to have been marginal performers in their home organizations. This resulted in some painful culling. The technical degree of complexity of the system became more extensive than normal because of the lessened ability to control technical vs. functional trade-offs. The user representatives were new to data processing and found difficulty in understanding the language and documentation of the technical staff, making specification sign-off a dubious exercise in some cases. Finally, there was a tendency by some line organizations to disown aspects of the system because "their" people were not directly involved in the decision and trade-off processes.

Another element of user participation instituted was a weekly meeting during the project design stages to review the progress of the project with the company division heads. These meetings were invaluable to securing those major decisions that typically bog down a major project during its early stages.

Because of extensive user participation, the final system is a much better functional system than typical and is truly user oriented.

4.3 Consultant and Vendor Participation

Consultants and vendor technical personnel were employed as technical support personnel during the design phases since in-house skills in that area were in short supply. Management consultants interviewed user and management groups to obtain an unbiased statement of requirements. Another consultant developed models that produced predictions of quantitative standards for system performance.

The hardware vendor's technical personnel provided system architectural alternatives for evaluation by AFS project personnel, supplied results of benchmarks for similar finance industry requirements, and demonstrated the performance of selected alternatives according to specifications supplied by AFS.

A software vendor either designed those parts of the system software which were either unavailable off-the-shelf or modified those parts of the software that were considered to be performance liabilities. In addition, the software vendor designed and coded all of the application software to interface with custom code.

4.4 Systems Development Methodology

Because of the involvement of several outside hardware vendors, several consultants, a contract programming company, and the magnitude of the project, the procedures for controlling the project were formal and structured. The project was phased; work plans for all activities were detailed; an automated budget tracking system was installed; performance and product quality standards were established prior to development; and organization components such as quality control, performance control and project control were created. During the course of the project, clear visibility of the current status was maintained at all times. This visibility, however, did not prevent overruns.

In spite of stringent controls, the time required for the design phases overran by 50%, and the program design and coding phase overran by 100%. Pilot testing on the other hand was accomplished in 60% of the originally estimated time, and conversion was completed in 13 months in contrast to an original schedule of 15 months. The accuracy of schedule estimates tend to become the measure of productivity rather than such objective criteria as user acceptance.

Highly proceduralized methodologies for developing systems can be expensive, time-consuming and irritating to users and vendors alike. The key is good judgement in the use of formality and structure. A project like BOS requires a high degree of control, while retaining flexibility to make changes.

The project design team followed a "prototype" strategy rather than a traditional "linear" strategy to minimize conflict and encourage communication among groups. The "linear" strategy requires each successive activity to follow logically from its predecessor. When detailed analysis reveals problems, a loop back to an earlier phase is required. Decisions must be made at each phase. The tendency is for specifications to be frozen without test at successively lower levels of detail. Changes in design are discouraged at all levels once such decisions are made.

The "prototype" strategy followed the same sequence. However, an initial, highly simplified version of the system was designed, implemented, tested, and brought into operation. Users, management, and designers reviewed the prototype. Prototype cycles were repeated in greater detail until the actual system was in full operation. Prototypes were constructed from benchmarks, models, hardware and software components, and documentation drafts.

Management felt that this approach would enable it to better handle frequent changes in user requirements or design inadequacies. The trade-off involved was accepting a higher initial project cost to develop and cycle through prototype versions of the system rather than attempt to define and freeze requirements and design early-on, thereby risking expensive loop-back periods to accomodate changes.

5. Feasibility Studies

5.1 Strategic Performance Issues

One of the first tasks of the design team was to define the performance attributes and measures in detail. Definitions of the performance attributes were essential to the plan for producing predictions according to the indicated measure. Performance predictions were subsequently used to make strategic decisions and establish performance standards for every successive phase of the project from concept of operation to full operating capability.

A recap of AFS's major business objectives, derived system performance attributes, and measures established for those attributes is given in Table 5-1.

5.2 Make or Buy Decision

AFS management established a task force to evaluate for purchase the most appropriate on-line system then installed at a competing finance company. Management intended AFS to become operational with an on-line system in the shortest possible time--even if it required complete acceptance by AFS of the operating procedures of the selling company. The task force recommended that AFS build its own on-line system. The task force had evaluated five existing competitive systems and concluded that none would meet AFS' needs.

5.3 Requirements Specification

A management consultant was tasked to perform a study of Headquarters and branch office operations. Business level processes, performance, and workload requirements were defined. (See Section 3, Concept of Operation.) A study of manual office operations was conducted for a full one-month period (January 1972) at 85 branch offices in 45 states. Generally, two offices (one large and one small office) were selected from each state. The concept of operation was discussed with branch office personnel, division representatives, and Headquarters management. A functional process description of the BOS operation was developed. Major features were to (1) automatically value-rate loan and sales applications; (2) prepare contract documents, checks, receipts, insurance claim forms, etc., on the terminal; (3) monitor the receipt and disbursement of all branch funds; (4) prepare customer statements, and management summaries and transmit them at night to the applicable unattended branch terminal; (5) provide inter-terminal communication capability between the branch and any other branch office, area executive office, or Headquarters; (6) permit branch personnel to interrogate any customer account or dealer record in the

Table 5.1 Recap of Performance Issues

BUSINESS OBJECTIVE	SYSTEM PERFORMANCE ATTRIBUTE	MEASURE
DESIGN AND DEVELOP SERVICES	FLEXIBILITY	RELATIVE COST TO DEVELOP AND MODIFY SYSTEM ARCHITECTURE
COST CONTROL	SYSTEM AND STAFF COSTS	RELATIVE LIFE CYCLE COST PER TRANSACTION
RELIABLE CUSTOMER SERVICE	AVAILABILITY	TIME TO FAILURE AND TIME TO REPAIR OF SYSTEM
ACCURACY	AUDITABILITY AND INTERNAL CONTROLS	VERIFICATION OF ACCURACY AND VALIDITY OF DATA RECOVERY FROM INCORRECT OR UNAUTHORIZED OPERATIONS
IMPROVED STAFF PRODUCTIVITY AND SERVICE	RESPONSIVENESS AND EASE OF USE	TIME REQUIRED TO COMPLETE A SERVICE, TRAIN AN OPERATOR
MANAGEMENT CONTROL OF SERVICE DISTRIBUTION	CONTROLLABILITY	PREDICTABLE PERFORMANCE STANDARDS

system; (7) generate, on a daily, weekly, and/or monthly basis, exception reports to support collection, solicitation, and regional management activities; (8) support branch personnel training activities by providing a training mode at the terminal; (9) provide a comprehensive auditability and control package for all in-house software and vendor packages; and (10) produce documentation in accordance with standards developed by users and those responsible for audit and control.[4]

Summary statistical data were obtained for branch office operations. The average number of transactions per account per month was calculated. An analysis of daily transaction activity indicated that the single significant peak for systems design consideration was the Monday morning condition occuring at the beginning of a month. Thus, the period studied included this condition.

Assumptions about the occurrence of transactions during the eight-hour period in each time zone were made based on the typical business day (See Section 2, Case Study Background.) A profile of "peak Monday" traffic volume by hour was developed from these assumptions.

The project team employed the first of a series of prototypes to derive system performance requirements. Acceptable system delay times were developed by examining the operator and manager activity workloads for major transaction types at a prototype of the BOS terminal. Extreme and average elapsed arrival time distributions of major transaction types were also constructed from scenarios acted out by users at the prototype terminal. Subsequent projects in industry and Government have automated the system prototype.[5,6]

Operators and user management specified the following performance measures: (1) a 15-second delay for 90% of the transactions and a maximum delay of 60 seconds (based on the number of people in an office, the office workload, and goals for the amount of work to be accomplished during average and peak business periods); (2) the system configuration was required to sustain a combined throughput of 19 BOS transactions per second and 0.4 administrative messages per second offered via the communications network in a peak period; (3) a central site (host and front end) availability of 95% up time per month, network (high-speed line) availability of 99% and local net (low-speed line) and terminal availability of

95%; (4) a count of the number of attempts per month to concurrently access the same data and Headquarters control over account number (key) assignment to minimize collisions (concurrent access); (5) detection of failure in system (host or network), amount of time inoperative, type of error and recovery to correct state of data, and detection of failure in a process with deletion of results and all intermediate states prior to update.

6. Acquisition and Design

6.1 Vendor Evaluation and Selection

Specifications were furnished to interested vendors in April 1972, and three proposals were received. IBM proposed the PARS/Financial System. "PARS/Financial" is a derviative of the Airline Industry Package for financial institutions. PARS/Financial is based on ACP (Airline Control Program) software. ACP has proven itself as a large on-line communications system, capable of handling high-volume message and/or transaction throughput with rapid response. At the time of request, ACP Model 5 was just being phased into production availability. Two IBM Systems 360/Model 165's were required to support all AFS performance requirements on the ACP Model 6, which was slated for production within the necessary time frames. AFS management rejected the IBM proposal for several reasons--OS software subsystems could not co-habitate with ACP; the 3705 communications controller could not be attached; terminals could be locked-out on multi-dropped lines; lack of special terminal features; and high configuration costs relative to the two other proposals.

IBM also proposed a second hardware/software configuration. The features of the second configuration -- OS/370, CICS, TSO, Programmable front-end 3705's, better line control under BTAM, new financial terminals -- and lower cost (by using two 370/155's in place of the 370/165's) made this proposal more attractive. AFS requested IBM to provide a benchmark demonstration of a prototype version of the system. AFS furnished a team of six people for one month to describe the functional, performance, and workload requirements to IBM personnel.

In May 1972, a prototype demonstration was constructed. The equipment configuration is shown in Figure 6.2. The 360/50 using a forerunner of the teleprocessing network simulator (TPNS) acted as a driver system (RTE-Remote Terminal Emulation) to the system under test (370/155).[7,8] Transmission of transactions over actual lines to and from simulated concentrators was initiated by exercizing the normal host BTAM polling function. Transactions were described by type, frequency of occurrence, data access requirements and message content. In the host, synthetic applications were developed by linking CICS macros to other programs that accessed files and executed an average number and mixture of instructions equivalent to estimates of application activity. Seven files were formatted to replicate a minature version of the data base. Files contained 10% of the actual record volumes estimated for the real system.

Measures of work accomplished, system behavior, and utilization of resources were obtained from the benchmark demonstration. Each unique message rate was measured for a period of six minutes following a six-minute period to allow the system to stablize. The length of the interval was empirically determined by IBM benchmark team members. Typical measurement periods require 10-to-15 minute intervals for startup and statistics gathering. The interval length was reduced because the workload was homogenous (only one mode--transaction processing) and a small number of transaction types accounted for 95% of all activity (payments alone were nearly 70% of the total).

The results indicated that further alternatives be explored because the IBM System 370/155-II CPU resource was heavily loaded, primarily with network and data base I/O control activity. It was too early in the system life to have little or no resource safety factor. Furthermore, features desired for audit, control and error handling were limited with this approach. AVCO requested proposals for "intelligent" front-end computers capable of handling numerous functions and off-loading the host and a special purpose data base management system to reduce file I/O activity. An independent software house proposed a tandem front-end that would require only a single host system. The proposal appeared to have desirable performance features similar to ACP (efficient host performance) coupled with the advantage of transaction inventory, control and auditability in the front-end and superior data base access, data recovery, and process recovery in the host. The hardware availability of the system was about the same as that of IBM's configuration, while system cost and flexibility

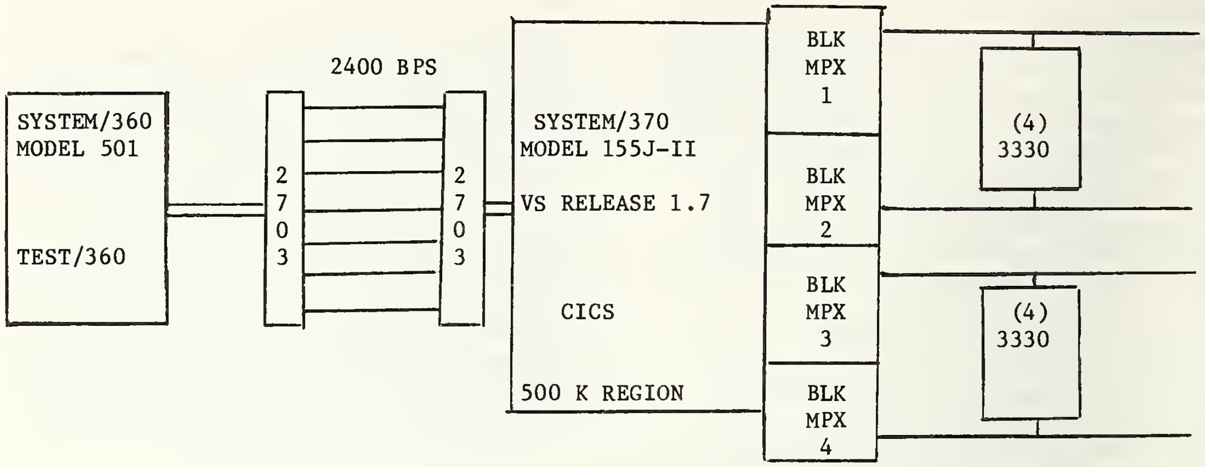


Figure 6.2 IBM Benchmark Configuration

were improved. The principal disadvantage was the risk of designing and implementing special software in both front-end and host systems. The proposal was accepted in July 1972.

6.2 Detailed Design

In September 1972, board approval was sought and granted for Phase I (the Detail Requirements and System Design phase) of a four phased system development project. Phase I was estimated to take 10 months and cost $1.5 million. Board approval for funding to undertake later phases was to be requested only upon successful completion of Phase I. Total costs were estimated at $16.7 million.

In January 1973, the Branch Operating System (BOS) Project was formally begun. The four project phases were originally defined and scheduled as follows:

Phase I -	<u>January 1973 - October 1973</u> Detail User Requirements and System Design.
Phase II -	<u>November 1973 - September 1974</u> Program Design, coding and all levels of testing (i.e., unit, subsystem, system and pilot testing activities).
Phase III -	<u>October 1974 - April 1975</u> Live Pilot (i.e., parallel operation in five selected branches in the state of California).
Phase IV -	<u>May 1975 - August 1976</u> Conversion of branches to the BOS System

The second prototype cycle sought to minimize the risks inherent with the independent contractors' approach in Phase I, the detailed user requirements and system design phase. A consultant was retained for the purpose of developing models of end-to-end performance. A peak-hour model was constructed using a package simulator. The data obtained from the requirements study were used to develop descriptions of the applications process. Network characteristics were obtained from a discrete FORTRAN line simulator model of the communications network. A simple central server queuing model of the front-end processor was also developed.

The results of the package simulator were calibrated to the benchmark data collected with the RTE. A "performance budget" was created, allocating a maximum value for each component of end-to-end response time. The "performance budget" allowed trade-offs to be made between process, workload, and response time components. Analytical models for each on-line hour of the peak-day were developed. Results indicated that the software house proposal would meet performance requirements with a slim reserve of CPU cycles on an IBM Ssytem 370 Model 158 host. The front-end resources appeared sufficient to do the job.

By mid-1973 BOS system design began. Model data were provided to the design teams. Application programming received reports enumerating estimates of application core, CPU time, and working set size requirements and I/O operations performed. Data base designers were provided estimates of access patterns and access times for various transaction types, based on account number key mapping schemes, load factors, file block size, empty space management strategies, and overheads for data and process recovery schemes. Operating system programmers were given estimates of CPU time and space requirements for special software. Throughput and turnaround time reports were studied by management. The off-line functions (unattended terminal) at each branch were examined to see if work could be completed before the next business day. Prototype models of various software and configuration options were used by most of the staff.

The third cycle of the prototype methodology began with the selection of equipment for installation. Management opted for an early prototype system to study design concepts in detail and permit BOS programs to be compiled and tested in a VS environment. An IBM System 370 Model 145 was acquired. The system was instrumented with hardware and software monitors. Measurements of software path lengths and I/O counts were made, and compared with model data.[9] Models were updated and new forecasts made.

Another study conducted with the prototype 370/145 system examined the long-range feasibility of using the in-house system for time-sharing (program development) resources. The amount of special software required a sizeable number of maintenance programmers. Results of measuring TSO software and its interaction with the on-line and batch products systems indicated the need for more processing power than has been scheduled to support in-house programming.[10] Because a multi-processor version of the IBM System 370/Model 158 was announced, it was decided to stay with the system on order and obtain a second processor if and when necessary. In February 1974, the IBM System 370/Model 158 was installed to service increasing BOS development requirements. This system operated under OS/VS2.

Phase I system design was formally concluded in November 1974. Two AFS business processes; namely, Thrift and Flooring were removed from the project's scope in order to alleviate schedule delays. Front-end computer measurement data indicated that the front-end system would not be able to achieve anything approaching its forecast response time budget.[11] AFS obtained the services of an independent consultant to study the problems observed in the front-end software and hardware. The consultant concluded that the front-end would not satisfy BOS long-term requirements. Performance requirements and specifications previously established by prototyping and modeling were advertised in a request for proposal for off-the-shelf front-end systems. After the bid solicitation, AFS decided to replace the front-end with a COMTEN 476 front-end system which would handle up to 1,500 terminals. A larger COMTEN configuration was also defined which would service up to 3,000 terminals to satisfy the long-range growth of the network.

In a presentation to the AFS Board of Directors in April 1975, it was reported that BOS development had been delayed thirteen months beyond the original schedule, that a second 370/158 processor was now required, that a replacement front-end communications system was needed and had been identified, and that additional funding was required to complete BOS development.

6.3 Test and Implementation

AFS assumed full responsibility for BOS testing and implementation. Testing requirements had been grossly underestimated by the application software vendor and a comprehensive test plan was non-existent. AFS was consequently required to develop a test plan and to re-test much of the system. At this point, AFS defined the following plan incorporating a fourth prototype cycle to ensure that the BOS system would be ready for mass conversion of the branches:

<u>Regression Subsystem Testing</u> - A re-test of all BOS Subsystems. This activity was scheduled to be complete by August 1975.

<u>System/Pilot Testing</u> - The entire BOS System would be tested using a controlled test script of historical data taken from selected branch offices. AFS expected to complete this activity in early September 1975.

<u>Shakedown Test</u> - A series of tests using all facilities of the BOS System to prove that the system functions accurately according to state laws where AFS operates.

<u>Parallel Pilot Test</u> - Operation under BOS in selected branches is parallel with existing systems. AFS expected to complete this test by the end of December 1975.

<u>Live Pilot Test</u> - Starting with two California branches and growing to

six, BOS would process data in a completely live mode. Branch personnel would operate the terminals and use the results for normal business activities. Any network problems would be confined to one geographic area. If no abnormal conditions arose during this activity, mass conversion would commence in May 1976.

Mass Conversion - Branches would convert to BOS in groups ranging from 25 to 100 branches per month. This process was scheduled to start in May 1976 and be completed in June 1977.

As AFS progressed into mass conversion, it was recognized that the on-line system would require more CPU resources than originally anticipated. A performance task force was established during mid-1976. This task force was chartered with identifying performance enchancements that could be implemented in the system so that AFS could complete mass conversion on the installed IBM System 370 Model 158 MP. In order to gain the greatest benefits, they focused on the system software, such as CICS and data base input/output subsystem. Through their efforts, mass conversion was completed on schedule, without additional CPU upgrades. To emphasize the improvement obtained from this performance activity, measurement statistics are presented in Table 6.1.

The statistics below demonstrated the significant improvement in transaction throughout rate and a reduction in transaction path length. The data indicate the dramatic growth in path length from conventional CICS to a highly modified CICS with a special recovery data design (to meet audit and accuracy requirements), and the effect of moving to MVS and to a later version of MVS with improvement in path length.

At the present time, the Branch Operating System is maintaining 1.1 million loan and sales finance account receivable records. During the first month of operation after mass conversion was completed (June 1977), the business was conducted in 956 branches with a terminal base of over 1,000. The average number of transactions per day during June 1977 was 115,000. The front-end communications system offered message switching capabilities to these branch offices with an average of 11,000 messages per day observed during June 1977. Highlights of activity and performance figures supplied to management for October 3, 1978 are shown in Table 6.2.

During the later stages of mass conversion, performance did emerge as a major problem and the system plus all other AFS processing ultimately approached the CPU capacity of the IBM System 370 Model 158 MP. Sizable tuning efforts were required to avoid additional upgrades. The ultimate solution became the implementation of an unsupported IBM access method called Start I/O (SIO) within the data base management system. At the present time, the BOS system has no performance problems.

Approximately seven people are required to perform non-discretionary maintenance of the system. An additional twenty people are making functional changes to the system. Compared to other systems of this size, these appear to be acceptable numbers.

7. Feedback from the Operational System

7.1 Study Successes and Failures

Calendar year 1978 marked the attainment of the terminal and transaction volume estimates made five years earlier. The system is meeting its performance objectives as originally specified. Management feels

BENCHMARK TEST	TRANSACTIONS PER SECOND	NUMBER OF CPU's	CPU %	TRANSACTION PATH LENGTH
May 1972	8.0	(1)	65	52,000
Nov 1976	5.9	(2)	126	393,000
Jun 1977	9.1	(2)	137	111,200

Table 6.1 Host Measurement Statistics

Table 6.2 AFS BOS Daily Operations Report DATE: 10/03/78

START-UP: 06:00	SHUTDOWN: 20:00		SCHED. OPERATING TIME: 14:00	TERMINAL PDP. 1,195
OUTAGES	NO.	TIME(MIN)	PEAK PERIOD 9:00-11:00 P.S.T. CENTRAL SITE AVAILABILITY: 96.42%	Month to Date: 97.97%
Host	-	-	Total Branches On-line: 1,086	Host Response Time: (SEC)[1] AVG 0.63 MAX 41.8
FCS	1	0:30	Total Terminals On-line: 1,143	TRANSACTIONS: 131,156
Both	-	-	Surging Overload: 110/1242[2]	MSGS SWITCHED: 14,778
Total	1	0:30	[2]Occurrances/Seconds of Host Polling Lost	OVERNIGHT RPTS: 5,954

[1]Add 15 seconds for maximum worst case network delay time.

that the effort was worthwhile to consider performance a strategic issue and a general and continuing concern. The strategic design criteria were sound and their validity is improving with time. The only recent hardware configuration change was a replacement of the '158 MP' by a single IBM 3033 processor as a cost saving measure, and to provide access to the latest version of the system control program (MVS/SE).

In retrospect, some improvements could be made to the prototype strategy. The requirements study was an expensive, time-consuming effort. With such a homogeneous user population, a much smaller sample could have sufficed. Recent advances in sampling techniques serve to reduce the number of interview hours and concomitant costs of workload characterization.

Of all the models employed, the least successful was that of the front-end computer. The front-end was modeled as an I/O bound system. Efforts were concentrated on accessing problems and queue delays in the I/O subsystem. The models failed to take into account such processing activities as polling, buffer management, traffic accounting, journaling, and serial use of resources. The path lengths to perform transaction activities and control I/O, created a processor bound front-end system with the same observable surging characteristics prevalent in the host RTE demonstration.

At the time of the implementation phase, the vendor's RTE (TP Driver) was still experimental. Problems discovered only after exposure to an initial group of users could have been located by stress-testing the system with the RTE.[12]

How did it all turn out? People costs did, in fact, outrun hardware costs. Thirty one percent of total project costs were people oriented versus 20% for computer hardware. It further turned out that COBOL was much less of a performance problem than anticipated. Less than 15% of all cycles are consumed by COBOL-generated code.

AFS found it necessary to develop its own data base management system, however, because available packages were either overly sophisticated or could not provide adequate performance. While this was counter to the basic design strategy, it allowed complete control over one of the most performance sensitive aspects of the system.

7.2 Conclusions and Recommendations

We are convinced that prototyping enhances a system methodology to "design for performance." Based on the success of this project, we conclude: Ex Opere Operatur (The act validates itself.)

By acknowledging that system performance is a problem from the outset, and that most performance issues are settled during the strategic decision-making periods, a company or agency can remain in control of the situation. Capacity planning after the fact, without knowing what the strategic performance factors are, will not be as effective, and will likely increase the risk of loops back to earlier phases, or failure to meet performance objectives. Instead, management must obtain performance data from prototypes and models to make the best decisions under risk about on-line systems. The source of good performance data for on-line systems are prototypes and predictive models, whose estimators can be used to manage a "performance budget." Measures of on-line systems on an ad hoc or post hoc basis reveals little of the problem. Resources must be allocated far in advance of on-going operations. Once in place, these resources and their users are not easily recast.

[1] Kirrene, M. J., Design of Operations-Oriented Decision Support Systems, Workshop At Ninth Annual SMIS Conference on MIS Productivity, Los Angeles, September, 1977.

[2] Bally, L., Brittan, J., and Wagner, K. H., A Prototype Approach to Information System Design and Development, Journal of Information and Management, Volume 1, 1977, pp. 21-26.

[3] Giles, H., Successful Network Management Hinges on Control, Data Communications, August 1978, pp. 33-41.

[4] Statement on Auditing Standards: Number 3 (SAS-3), American Institute of Certified Public Accountants, 1974, pp. 1-12.

[5] Nielsen, N. R., Brandin, D. H., and Placko, M. A., Design and Simulation of the Man-Machine Interface Using Microprocessor Systems, SRI International, October 1977 (Unpublished).

[6] Spiegel, M. G., Marshall, T., and Hodak, M., Prototype Integrated Financial Management System, Teleprocessing Specifications for Navy Regional Data Automation Command (NARDAC), Contract N00600-72R-112, January 1974.

[7] Remote Terminal Emulation Specifications for Federal ADP System Procurements, Automated Data and Telecommunciations Service, General Services Administration, October 1978.

[8] Use of Remote Terminal Emulation in Federal ADP System Procurements, Automated Data and Telecommunications Service, General Services Administration, March 1979.

[9] Chambers, J. F., Findings on Hardware Monitor Measurements of OS/VS2 Release 1.6 Nucleus, SHARE Computer Measurement and Evaluation Newsletter, Number 28, November 1974, pp. 27-51.

[10] Chambers, J. F., The FCB HASP Marco Can Reduce System Performance, SHARE Computer Measurement and Evaluation Newsletter, Number 30, March 1975, pp. 78-79.

[11] Chambers, J. F., Communication Line Monitoring with a Hardware Monitor, SHARE Computer Measurement and Evaluation Newsletter, Number 35, January 1976, pp. 24-28.

[12] McQuillan, J. M., and Falk, G., A Comprehensive Approach to Network Testing, Data Communications, August 1977, pp. 63-66.

Quantitative Methods In CPE

QUANTITATIVE METHODS IN COMPUTER PERFORMANCE EVALUATION

Aridaman K. Jain
Bell Laboratories
Holmdel, NJ 07733

1. Introduction

Computer Performance Evaluation (CPE) is concerned with measurement, analysis and evaluation of computer systems. Most of the CPE studies in the literature concentrate on the collection of data, and they place inadequate emphasis on the analysis of data and interpretation of results. CPE studies are conducted by using one or more of the following approaches: (i) analytical, (ii) discrete simulation, and (iii) empirical. The empirical CPE studies in the literature generally lack the proper use of quantitative techniques. These studies can be improved by the use of statistical techniques, such as design of experiments and data analysis.

This session attempts to encourage CPE analysts to use quantitative methods for modeling computer system performance. The topics covered in this session include application of clustering techniques [1] and development of performance measures [2].

Statistical steps in a CPE study and some useful statistical background for CPE analysts have been described by the author in a recent article in Performance Evaluation Review [3]. The next two sections of this paper, based on [3], present a brief review of past empirical studies and suggestions for future work, respectively.

2. Review of Past Empirical Studies

A review of past empirical CPE studies is useful for an examination of the strengths and weaknesses of these studies. The review is divided into three parts: (i) studies deficient in the use of statistical techniques, (ii) studies which made good use of statistical techniques, and (iii) studies both proficient and deficient in the use of statistical techniques. One CPE study is reviewed in each of the three parts.

2.1 Example of a Study Deficient in the Use of Statistical Techniques

A majority of the empirical CPE studies emphasize the collection of data, with very little emphasis on their analysis and interpretation. One such study is reviewed below.

"Honeywell Time Sharing Experiments on a CPU Bound System" [4] reported on the effects of (i) increased workload and (ii) certain parameter changes. The results consisted of average response times without any considerations of inherent random variability. By comparing the average response times, it is claimed that the lengthening of the time slice (given by TSS to its subsystems) improves the response.

Had a proper statistical analysis been done by fitting a regression model, and the question asked "whether the fitted regression coefficient associated with the length of time slice might be merely an estimate of zero," the answer would probably have been "yes". Why? Even though the raw data are not available for such regression modeling, an examination of the following averages (read off a figure from the reviewed paper) supports the above inference regarding the regression coefficient associated with the length of time slice:

Experiment No.	Time Slice (Milliseconds)	Average Response Time (Seconds)
15	15	20.8
12	25	20.2
14	25	19.2
20	35	18.4

Experiments 12 and 14 are identical. The average response times fcr these two experiments differ only because of random variability. This random variability can account for the observed differences between the 25 milliseconds time slice experiments and the other two experiments with time slices 15 and 35 milliseconds, respectively. This result indicates that the length of time slice has no effect on response time. Thus, it is not sufficient to consider just the averages without their associated variabilities.

2.2. Example of a Study Which Made Good Use of Statistical Techniques

There are several studies in the literature which have made good use of statistical techniques. One such study is reviewed below.

Tsao and Margolin [5] present the statistical methodology applied in their analysis of a multifactor paging experiment. They statistically designed an experiment to study the effect of four factors (main memory size, problem program, deck arrangement and replacement algorithm) upon the paging process. All 81 possible combinations (3 levels of each factor) of the four factors were observed.

A preliminary step in the analysis of data was the computation and plotting of summary statistics. For the response variable, number of page swaps (PS), the range of 81 responses was about eight times the average. This prompted the rescaling to logarithms. Regression models were fitted and the technique of analysis of variance was employed to decompose the variability of the response into components attributable to various factors. This decomposition permitted an examination of the relative importance of the effects of the various factors in the modeling of the response.

The paper by Tsao and Margolin [5] is quite exhaustive in coverage of the statistical methodology for their multifactor paging experiments. It is recommended for your study.

2.3 Example of a Study Both Proficient and Deficient in the Use of Statistical Techniques

Many CPE studies belong to this category. They are proficient in the use of some statistical techniques, but deficient in the use of others. One such study is described here. "A Model of Virtual Storage Personal Computing (VSPC) Based on QNET4" [6] was concerned with the development of a model to aid the process of locating and relieving potential bottlenecks. The authors select a subset of 23 commands to represent transaction types. No information is given on the criteria used to determine their representativeness. The 23 commands are grouped into 6 classes based on path length of command (average number of instructions executed). No discussion of system service routine path lengths is given. A potential user has no idea of what the effect on potential bottlenecks will be if system software performance parameters are altered.

The authors state that an empircal study of the region of convergence bias showed that this region corresponds well within known points of stability of the VSPC system as measured under VS1. It would have been useful to provide data to enable a comparison of the change in the effect predicted with an estimate of the inherent variability. The value of a CPE study is enhanced by providing quantitative data so that readers can make their own judgments about the study.

Data collection and analysis in this paper are lucid. The use of randomization to avoid bias and sampling to reduce time and cost are clearly explained. The solution methodology is presented carefully, in a well-structured fashion. The use of a graphic illustration of the flow of control among software elements is a particularly effective technique that allows the reader to "see" the behavior of the major software modules.

The validation of the results is limited to a discussion of how the model was calibrated. Confidence exists between endpoints of the calibrated model, but no attempt is made to verify the predictability and appropriateness of assumptions and constraints by performing experiments outside the calibrated range.

3. Suggestions for Future Work

In order to help make CPE studies more useful, three areas of future work are suggested: (i) quantification of management's objectives, (ii) workload representation, and (iii) forecasting of load. In the first area, it is necessary to do further work on the translation of management's objectives into an objective function of measurable variables. This objective function can then be used to evaluate the performance of a computer system.

In the second area, we need to answer questions such as "what is a representative workload for a given computer system" and "how is workload characterization affected by assumptions." In the third area, it would be useful to develop models for forecasting future load which are valid for the following factors: (a) growth in loads, (b) change in the nature of load, and (c) change in configuration.

The final suggestion is that more use should be made of the techniques of design of experiments and statistical data analysis for the planning and analysis phases of CPE studies. Review of CPE studies revealed that the conclusions of some of these studies would/could have been different, if statistical techniques had been used properly.

References

[1] Hartrum, T. C. and Thompson, J. W., "The Application of Clustering Techniques to Computer Performance Modeling," Proceedings of the 15th CPEUG Meeting, 1979.

[2] Dzelzgalvis, I., "Performance Comparison Measures for Computer Systems," Proceedings of the 15th CPEUG Meeting, 1979.

[3] Jain, A. K., "A Guideline to Statistical Approaches in Computer Performance Evaluation Studies," Performance Evaluation Review, Vol. 7, No. 1-2, 1978.

[4] Proceedings of the 12th CPEUG Meeting, 1976.

[5] Tsao, R. F. and Margolin, B. H., "A Multifactor Paging Experiment: II. Statistical Methodology", Proceedings of a Conference on Statistical Computer Performance Evaluation, edited by W. Freiberger, 1971.

[6] Downs, R. X. and Tweeden, H. E., "A Model of Virtual Storage Personal Computing (VSPC) Based on QNET4," Proceedings of the 1977 Sigmetrics, 1977.

THE APPLICATION OF CLUSTERING TECHNIQUES TO COMPUTER PERFORMANCE MODELING

Thomas C. Hartrum
Jimmy W. Thompson

Electrical Engineering Department
Air Force Institute of Technology
Wright-Patterson Air Force Base, Ohio 45433

The performance of a given computer system is to a large extent dependent upon its workload. A fairly standard approach to workload modeling is to define the workload by the amount of computer resources each job consumes. If each resource is considered to be one element in an ordered set, then a job's workload requirement can be represented as an n-dimensional vector $\bar{X} = (x_1, x_2, \ldots, x_n)$ where, for example, x_1 = CPU time, x_2 = central memory used, and so forth. By applying vector distance measurements the workload can be partitioned into clusters of "similar" jobs based on their nearness in n-space. This approach has been applied to describe workloads more accurately than by the aggregate resource usage of all jobs taken together. This paper investigates the possibility of extending the clustering technique to modeling and predicting computer performance. If one hypothesizes that performance is a function of resource consumption, then one should be able to determine a predictable range of performance for each cluster. This paper presents the results of the application of this technique to the workload characterization of a Cyber 74 computer and the subsequent investigation of the relationship between a job's turnaround time and its workload cluster.

Key words: Cluster analysis; computer modeling; computer performance; empirical models; modeling; performance modeling; workload characterization.

1. Introduction

An important aspect of computer performance evaluation is the development of models for predicting performance of given jobs or to measure the change in performance due to a modification of the system. One approach is the use of empirical modeling, where a relationship is determined between output performance variables and input workload parameters, based on observed data. One of the more popular empirical techniques, regression analysis, has limited use due to the nonlinearity of the relationships and the interaction and lack of independence among the workload parameters. In an effort to solve these problems, this paper explores the possibility of an empirical model based on a clustering analysis of the workload. Cluster analysis is a technique which attempts to find natural groupings of jobs based on similar resource requirements. Usually cluster analysis is used to model a system's workload for the development of benchmarks and simulation inputs. This paper considers modeling the relation between performance and workload on a cluster basis, resulting in a number of possibly disjoint empirical performance models, one for each cluster.

Some of the concepts developed in this paper were applied to an analysis of the workload and performance of a Cyber 74 computer. The results are discussed in this paper. The workload was successfully modeled using cluster analysis, and an improvement in turnaround time due to a policy modification was verified using a cluster - by - cluster approach. Some problems were encountered in the development of a cluster based performance model, and possible reasons for these problems are discussed.

2. Concepts of Clustering as Applied to Modeling

2.1 Clustering Techniques

Clustering is a technique usually applied to the area of workload characterization [2][1], largely as an attempt to more accurately model the interdependencies of the workload parameters. Workload is traditionally defined in terms of a set of job parameters representing resource demands [3] [4] such as CPU time, I/O time, and central memory. In many studies, these parameters are then treated as independent random variables, so that the workload is characterized as a set of probability distribution functions, one for each parameter. In order to account for interdependencies between parameters (such as most jobs with large memory requirements also require large processing times), a more accurate workload representation is needed. These interdependencies can be modeled by defining a job vector $\bar{X}$ (where the $^-$ denotes a vector) which is an ordered set of the job parameters. The workload is then characterized as a vector distribution instead of independent parameter distributions. Thus $\bar{X} = (x_1, x_2, \ldots, x_n)$, where, for example, x_1 might be CPU time, x_2 might be core memory required, and so forth. The proper description of the workload then becomes a joint probability density function over the set of job parameters. Such a probabilistic description may be extremely difficult to manage. Cluster analysis attempts to simplify this description in the following way. If the workload can be broken up into m classes or clusters of jobs, where a given cluster contains jobs that have similar resource requests, the joint p.d.f. $p(\bar{X})$ can be given by,

$$p(\bar{X}) = \sum_{i=1}^{m} p(\bar{X}|C_i)p(C_i) \qquad (1)$$

where $p(\bar{X}|C_i)$ is the conditional p.d.f. of the jobs in class C_i, which hopefully can be represented by a simple, unimodal function, and $p(C_i)$ is the p.d.f. of the classes C_i. A simplifying assumption at this point would be to reduce the conditional p.d.f. to a point estimate if the jobs within each cluster were nearly enough alike.

The effect of this approach is to break the large, complex distribution into a number of simpler distributions. There are three tasks needed to characterize a particular workload in this manner.

1. The m clusters C_i must be determined.

2. $p(C_i)$, the distribution of the clusters, must be determined.

3. The m conditional distributions $p(\bar{X}|C_i)$ must be determined.

One approach to the implementation of tasks 1 and 2 is to represent each job of a measured workload as a point in n-space, and then to apply a clustering algorithm (well established in the field of pattern recognition) to find the clusters based on the distance between the job points in n space, [1], [2], [5]. If the cluster centers are already defined, then a new job can easily be assigned to one of these clusters by finding the minimum of the distances between that job ($\bar{X}$) and each of the defined cluster centers ($\bar{C}_i$). The basic concept is clearly illustrated for a two-dimensional case in figure 1. The vectors themselves are

$$\bar{X} = (x_1, x_2) \quad \text{- the job vector} \qquad (2)$$

$$\bar{C}_i = (C_{i1}, C_{i2}) \quad \text{- the cluster center vector for cluster } C_i \qquad (3)$$

$$\text{hence } ||\bar{X}-\bar{C}_i|| = \sqrt{(x_1-C_{i1})^2 + (x_2-C_{i2})^2} \qquad (4)$$

Thus the distance measure being used is the standard Euclidean distance. The more sophisticated clustering algorithms which are usually applied utilize a weighted distance, where the weighting is usually based on the variance of each parameter

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

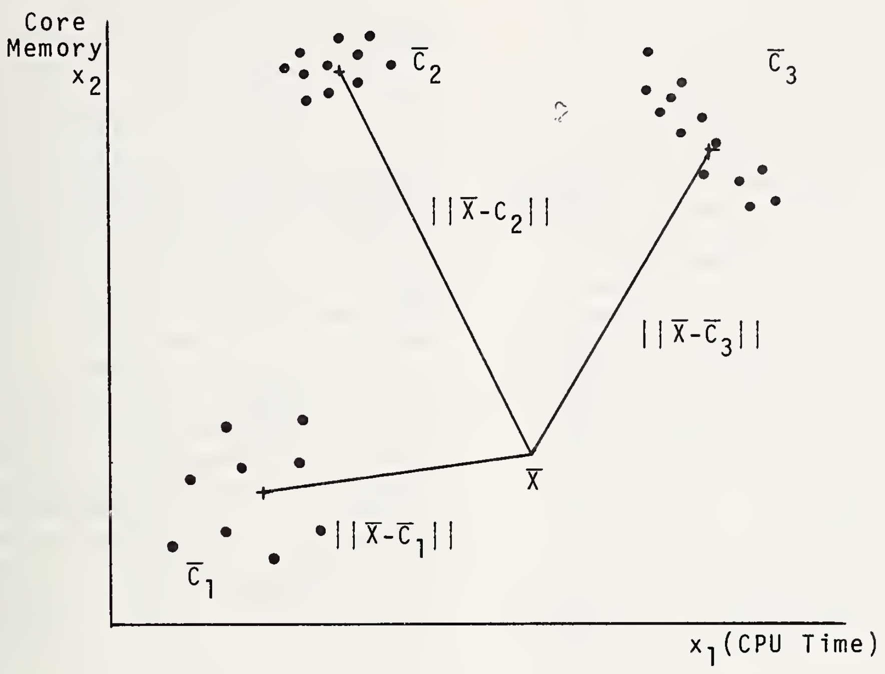


Figure 1. Euclidean Distance in 2-Dimensions

along its axis, or on the variances within each cluster. Since the principal axes in this n-space are based on parameters with uncompatible units (e.g. words of memory vs CPU time), each parameter has to be mapped into an appropriate dimensionless quantity. Such a mapping may involve such techniques as simple scaling, or logarithmic transformation [2], [5]. In characterizing an unknown workload, however, the cluster centers are not usually known a priori. An iterative technique is used, starting with a set of "seed" cluster centers, and on each pass, the following steps are performed:

1. Determine, for each job, the nearest cluster center.

2. If the job falls within a predefined minimum distance from that cluster, it is assigned to the cluster. If the job falls outside this minimum distance, it is established as a new cluster center.

3. At the end of the pass, all clusters with less than some minimum number of jobs assigned are dropped from the set of cluster centers.

This process continues until the set of clusters doesn't change from one pass to the next, at which point the m clusters C_i are defined. By counting the number of jobs assigned to each cluster, the distribution $p(C_i)$ can then be determined. Finally, the conditional distributions $p(\bar{X}|C_i)$ are determined by examining the jobs within each cluster C_i. This brief discussion of clustering provides the necessary background for the remainder of this paper. The interested reader is directed to references [1], [2] and [5] for a more rigorous discussion of this technique.

2.2 Clustering as a Performance Modeling Technique

Clustering as defined in Section 2.1 is represented as a way of characterizing, or modeling, a system's workload. This may be an end in itself if the intent of a study is to analyze the workload for its own sake. More often, though, the modeling of the workload is only a preliminary step in the study in order to provide a workload model to drive a model of the system. Thus if analytical distributions can be found for the conditional distributions $p(\bar{X}|C_i)$ and for the cluster distribution $p(C_i)$, the resulting workload model can be used in conjunction with analytical models of the system. If only empirical distributions can be defined, they can be used to generate synthetic workloads to drive a Monte Carlo simulation, or they can guide the selection of a benchmark workload to run on the actual system [2], [6]. These approaches all require a separate system model based on some knowledge of the system. Another approach to modeling system performance, sometimes referred to as empirical performance modeling [3], is based on finding a model relationship between performance and workload from empirical data alone. This "black box" approach assumes no knowledge of the actual functioning of the system, but seeks only to relate the dependent variables (performance) to the independent variables (workload) based on observed occurrences of these variables. Clearly such models have little value in analyzing causes of system performance or in tuning the system. However, they do satisfy the requirements for a predictive model. Thus, once the model has been developed, it can be used to predict the performance of a given workload without considering the "why" or "how" of the performance.

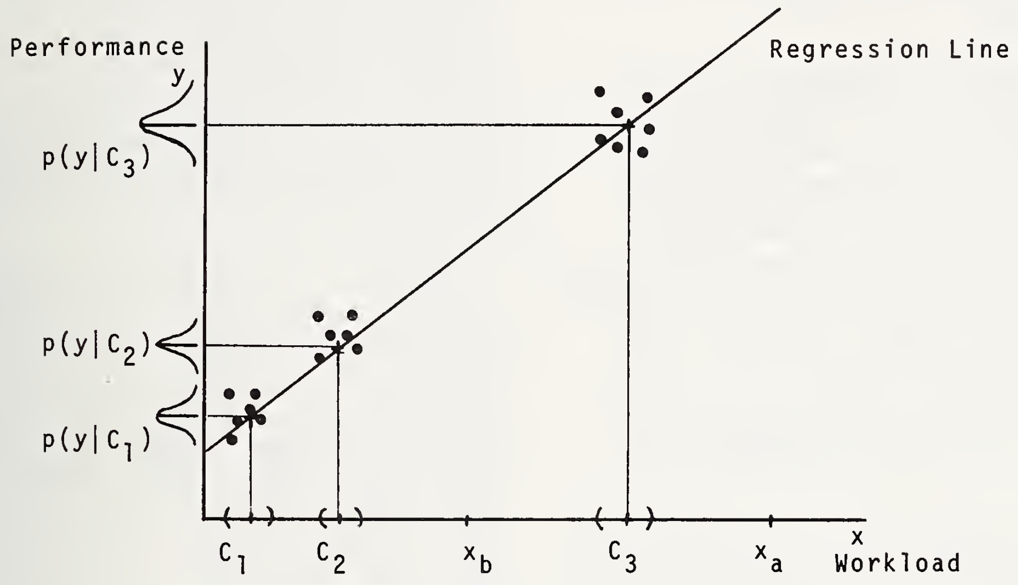
Multilinear regression is one of the more popular empirical modeling techniques. The model is of the form

$$y_j = a_j + \sum_{i=1}^{n} b_{ij}x_i \tag{5}$$

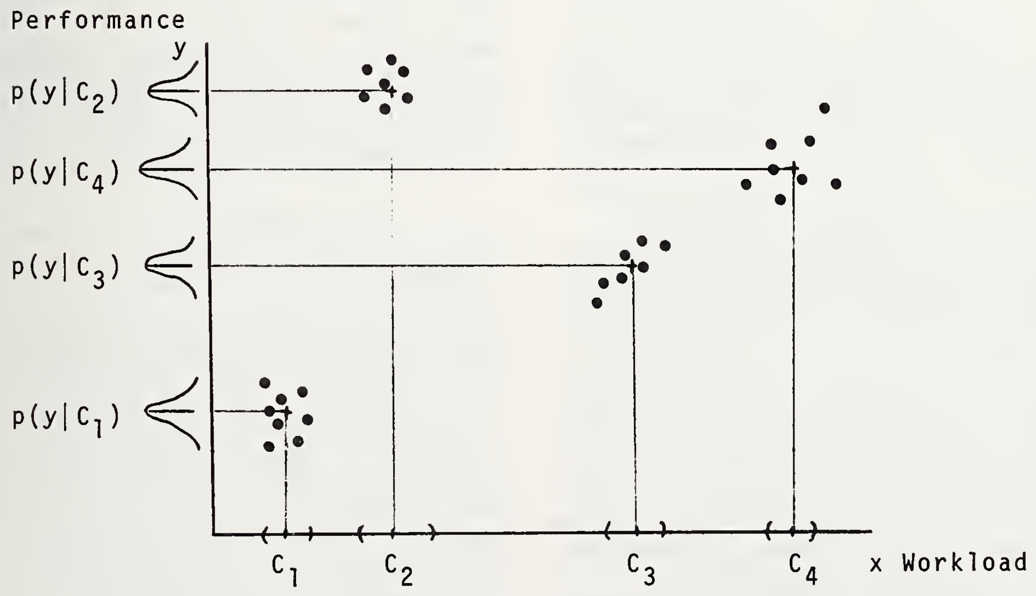
where y_j represents a given performance measure (e.g. turnaround time), the x_i are the workload parameters as defined in Section 2.1, and a_j and the b_{ij}'s are the model parameters derived from the empirical data. One use of such a model is to predict the performance y'_j for a particular workload $\{x'_i\}$. Note that since we are dealing with a statistical model, a point estimate of y'_j may not be sufficient. Rather a confidence interval (or prediction interval) can be calculated about the predicted value y'_j. Another use of this model would be to verify the effect of a system change. By comparing the system's performance after a modification with that predicted for the unmodified system with the same workload, a statistical hypothesis test can determine whether the modification had a significant effect on performance.

One of the major drawbacks to multilinear regression is the assumption that the dependent variable is linearly related to the independent variables. Although a linear approximation is a useful tool in many cases, it may introduce unacceptable errors in some models. Although many multilinear regression programs are available, nonlinear techniques are much more difficult to come by. A common approach is to try to find some transformation of the data which makes it linear, and to apply linear techniques to the transformed variables. An example would be transforming $y = a\,x^b$ to $y_1 = \ln(y) = \ln(a) + b\ln(x) = a_1 + bx_1$. Such transforms are often difficult to find in the case of multivariable data. Another problem with regression is the assumption of no interaction between the independent variables (not to be confused with actual independence of the independent variables). This assumption indicates, for example, that the effect of CPU time on performance is the same regardless of memory requirements. That is, the regression coefficient b_{kj} is independent of the values of the variables x_i. If the model is to be used primarily for predictive purposes, this problem can be compensated for by including in the model a term containing the product of the interacting variables. Finally, the independent variables are assumed to be truly independent. As indicated in Section 2.1 this is often not the case for multiple workload parameters. While a non-zero correlation between some of the independent variables may have negligible impact on the predictive capability of the model, variables which actually contribute to performance can be dropped from the final equation.

These problems with multilinear regression will occur over the entire range of the independent variables. Once the jobs have been sorted into clusters, however, within the range of a given cluster the data may be well behaved, that is, it may meet all of the multilinear regression assumptions. This effect is illustrated in Figure 2 for the violation of linearity in the case of a single independent variable. Figure 2a portrays the case where a linear relationship does exist and illustrates the concept of performance modeling by clusters. The independent variable x (workload) is partitioned into clusters C_1, C_2, C_3, and the remaining range of x into which few jobs fall. Associated with each cluster C_i is a distribution of performance, $p(y|C_i)$. In this case, clearly the linear regression model could have been used to predict the



a. Linear Relationship, where x_a and x_b represent workloads whose performance is to be predicted using the model.



b. Nonlinear Relationship

Figure 2. Use of Clustering to Eliminate Nonlinearity

performance distribution $p(y|x')$ for a given x', where in regression models this distribution is assumed normal and can be estimated from the data.

In figure 2b, however, it is not possible to accurately model the data linearly (although one can find a least-squared-error straight line for any given data), and finding an appropriate nonlinear function does not appear to be easily accomplished. Note, moreover, that it is completely unnecessary to define a nonlinear function over the range of x because the relationship of y to x is irrelevant over those intervals of x in which no jobs are expected to be found. Since the intervals of x between cluster boundaries contain no jobs of interest, then performance can be modeled by finding the distribution of performance $p(y|C_i)$ for each cluster C_i. If the range of x for each cluster is much smaller than the total range of x, then the effect of the nonlinearity disappears, as shown in figure 2b. Similarly, one can argue that if the range of a given cluster is small, the effects of interaction and dependencies between independent variables can be ignored.

Clearly it may be the case that the range of a cluster may not be small compared to the total range of the independent variable, and that the function may not be well behaved within the cluster. However, by removing the effect of the other clusters and modeling only $p(y|C_i)$ for that cluster, one may still gain in modeling ability over attempting to fit a nonlinear model over the whole range. This is illustrated by cluster C_2 in figure 3. Its range is relatively large, and clearly the data in that cluster is nonlinear. However, elininating clusters C_1 and C_3 allows a reasonable nonlinear model to be fit to C_2 alone (e.g. perhaps an exponential or quadratic function) whereas finding a function to fit all the data together appears to be a bit more difficult.

Figure 3 illustrates another possible situation. Notice that although the workload nicely clustered into distinct clusters C_1 and C_3, the performance distributions associated with these two clusters fall right on top of each other. This corresponds to the case in linear regression where the data fits a straight line nicely, but the resulting line is horizontal. The latter example indicates that y is functionally independent of x. The cluster case similarly indicates that performance is independent of whether jobs fall into cluster C_1 or C_3. Stated differently, jobs in clusters C_1 and C_3 would be predicted to have the same performance. Thus for the purposes of performance prediction modeling, clusters C_1 and C_3 could be combined into a single cluster. In linear regression, this situation can be detected by performing a statistical test of the hypothesis that the regression coefficient $b_{ii} = 0$. In the clustering case one can perform a similar statistical test on whether the performance distributions $p(y|C_1)$ and $p(y|C_3)$ are the same.

One other problem arises in using the clustering approach to performance modeling. The same problem occurs in regression modeling, but can be more easily assumed away there. An empirical model is by definition based on observed data. The model is, in effect, defined only over the range of this observed data. The problem occurs when one wants to predict the effect of a value of the independent variable which falls outside this range. Usually one assumes that the model is still valid outside the range over which it was defined. If, for example, one wishes to predict the performance of a job with workload parameter x_a using the model of figure 2a, one must assume the model can be extrapolated that far. This may be a risky assumption since the true curve might suddenly turn upward past C_3, but the assumption is usually made for lack of any other approach. Less risky is the assumption of interpolation, as illustrated by point x_b in figure 2a. This is often not a problem since the observed data is well distributed over its entire range. Thus in either case one can predict outside the range of the model by assuming the model stays valid and by recognizing the attendent risk.

In the cluster model, however, the problem arises that the model is completely undefined between clusters. Thus one cannot predict the performance of a job which does not fall into one of the defined clusters without in effect defining a hypothetical model between clusters. For example, in the one dimensional case illustrated in figure 2b, one could connect the centroids of the clusters with straight lines. However, this is very presumptuous to say the least since, had cluster C_3 not been in the original data set, this approach would have predicted it to lie on a line connecting clusters C_2 and C_4. A more serious problem than that of predicting the performance of a single "nonstandard" job, is the problem of predicting the performance of a major change in workload which would create new clusters.

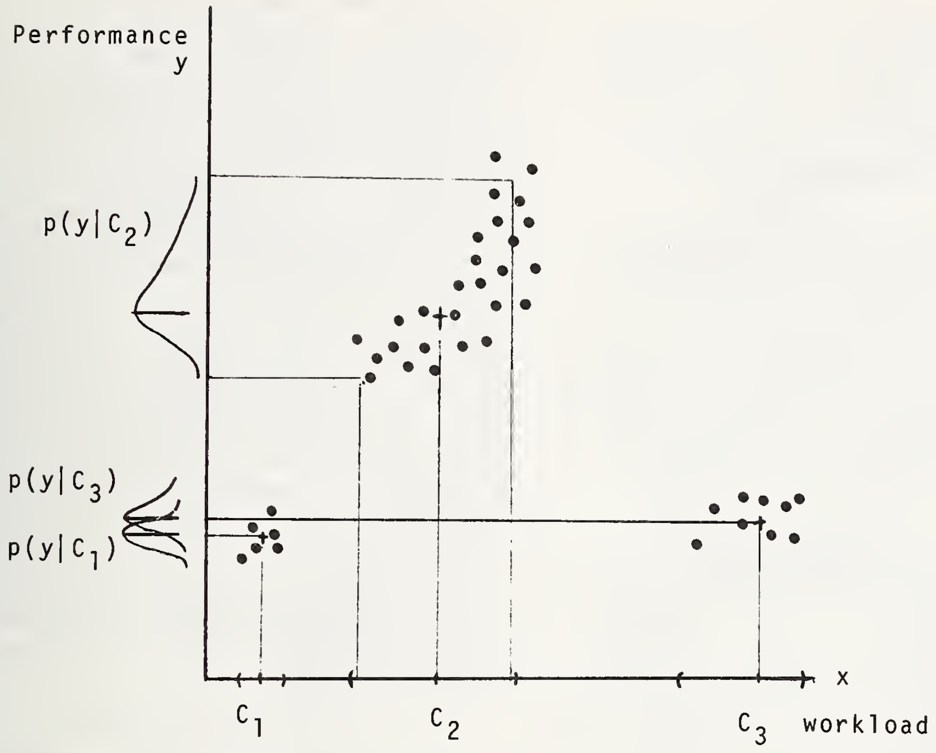


Figure 3. Complex Cluster Relationships

Although these problems place some limits on the applicability of this approach, the cluster approach does provide a predictive model which can overcome many of the limitations of multilinear regression.

3. A Case Study

3.1 Description of a Modeled System

A study was made of the workload and turnaround time for batch jobs on a Control Data Corporation CDC Cyber 74 Computer at the Aerospace Systems Division (ASD) Computer center, a component of the Air Force Systems Command, located at Wright-Patterson AFB, Ohio. When the study began, the Cyber was processing a combination of batch and intercom jobs. During the study a policy change was made, and all but a few intercom jobs were moved to a companion computer (a CDC 6600) and all of its batch jobs were moved to the Cyber in an attempt to improve batch turnaround time. The objectives of the study were (1) to determine whether clustering techniques could be applied to the diverse ASD workload; (2) to apply the concept of performance modeling on a cluster basis; and (3) to verify the improvement in turnaround time resulting from the policy change.

The Cyber 74 was designed to overlap as many processes as possible. The system provides for multiprogramming by using small independent computers, peripheral processing units (PPUs), to accomplish input-output while the main CPU executes the computation instructions. There are 20 identical peripheral control processors associated with the Cyber at ASD. Each is independent of the other and has its own 4K of memory for programs constructed from a 64 instruction repertoire. All peripheral processing units have access to 131K 60-bit words of central storage, and to the peripheral channels. With this capability, the PPUs act as system control processors as well as I/O processors. This permits the central processor to continue high speed computations while the PPUs do the slower I/O and supervisory operations. Not only do the PPUs perform all I/O required by central storage programs, they also contain the major portion of the operating system, called the monitor. PPU number 0 is designated as the unit to hold the monitor, which is discussed further in the next paragraph. A second PPU is assigned to control the operators console.

The Network Operating System/Batch Environment (NOS/BE) controls the operation of the Cyber 74. NOS/BE supports batch, interactive and graphics processing. An NOS/BE program JANUS, which is located in PPU 0, monitors the total operation of the system, including directing the CPU and other PPU actions. To provide a multiprogramming capability, JANUS stores up to 15 jobs in central memory. These jobs reside in 15 variable length areas of memory called control points; the size of a control point is called its memory field length. Additionally, JANUS is responsible for calling the scheduler program into execution to initiate job execution for new or swapped-out jobs. The job flow of the Cyber NOS/BE operating system is diagrammed in figure 4. A job, which is a sequence of task and program steps, is read in and stored on disk. Those jobs inititated by control cards are stored in the input queue. Jobs spawned via intercom terminals are entered directly into the central memory (CM) queue. Also stored in the central memory queue are those jobs that have been swapped out (removed from a control point due to an expiration of their time slice). These jobs are all in competition for an active control point when it becomes available.

The decision of whether the next job will come from the input queue or the central memory (CM) queue is dependent upon the circumstances that required the calling of the scheduler (reference figure 5). If a job has used its base quantum and a higher priority job is in the CM queue, a swap-out and swap-in will be done. Secondly, if a job has terminated, thereby giving up its control point memory, the scheduler will check the input queue for a job to initiate. The highest priority job in the input queue will then be brought into memory. The priority of jobs in the input queue is dependent upon three factors. Jobs with the smallest I-O and CPU time and central memory requested will be given a priority higher than normal batch. The aging rate priority is added on an hourly basis. Within the input queue are batch jobs for all user organizations. Therefore, the ASD computer center has imposed a software Organizational Scheduler between the input queue and the NOS/BE scheduler.

Each user organization of the Cyber is authorized a percentage share of the Cyber resources. This resource use is calculated in computer resource units (CRUs) for each organization's batch and interactive jobs. CRUs are calculated for each job as a percentage of CPU plus I-0 time plus central memory used. To be selected for job activation, a job in the input queue must have the highest priority and the CRUs used by its organization must be within the CRUs authorized for this hour. The Organizational Scheduler exhibits control only over the input queue (batch jobs), even though CRU's used are a measure of batch and interactive jobs. An organization using intercom extensively can therefore penalize batch users. These controls, such as priority job classes and aging, are used in analyzing performance. Additional controls placed on the Cyber system dictate that during the 0800-1600 hours time frame, all interactive jobs, and only those batch jobs requesting less than 170K octal words of memory, 400 seconds of CPU time and 1,000 seconds of I/O time, will be processed.

3.2 Data Collection

The data collection for workload and performance measures was made available by the Boeing Computer Service program, CLARA. The CLARA program accepts data from the NOS/BE operating system Dayfile tapes, which contain all event messages generated by the operating system during normal processing. A new Dayfile tape is started every 24 hours (0001-2400). Each Dayfile message begins with the computer clock time of day followed by messages of the following three types

- Identification data, such as job name, receipt data, job origin (batch, interactive), and account number.
- Initiation and termination times, such as the time a job entered the queue, and the times that a job entered and left a control point.
- Computer resources used, such as CPU, PPU, and I-0 time and central memory, tape drives, and disk sectors used.

The CLARA program is then used to process each day's Dayfile tape to reduce its large volume down to a more manageable size. These data are stored on magnetic tapes, titled Permanent Data Tapes (PDT), by CLARA.

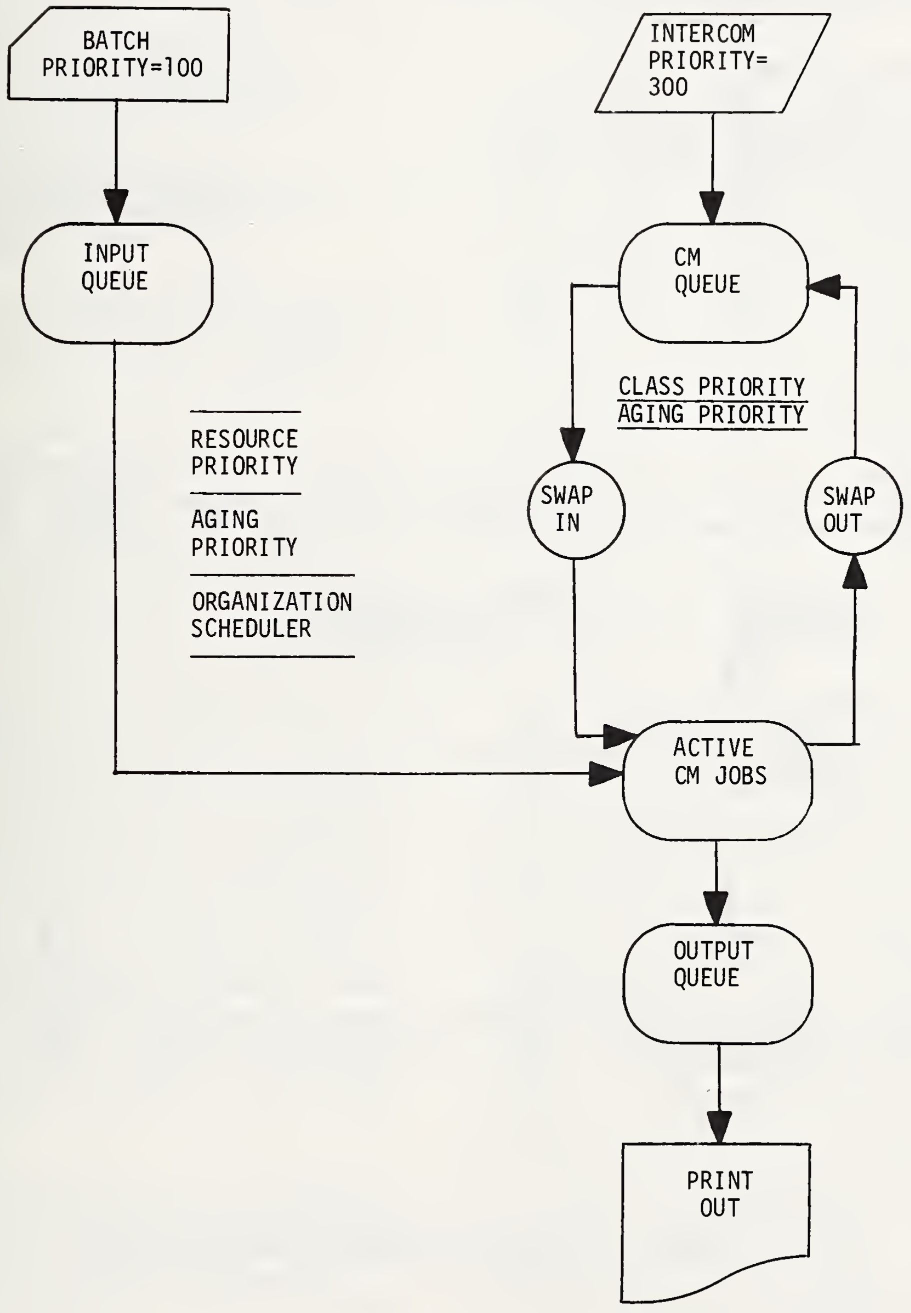


Figure 4. Cyber Job Flow (Reference [1])

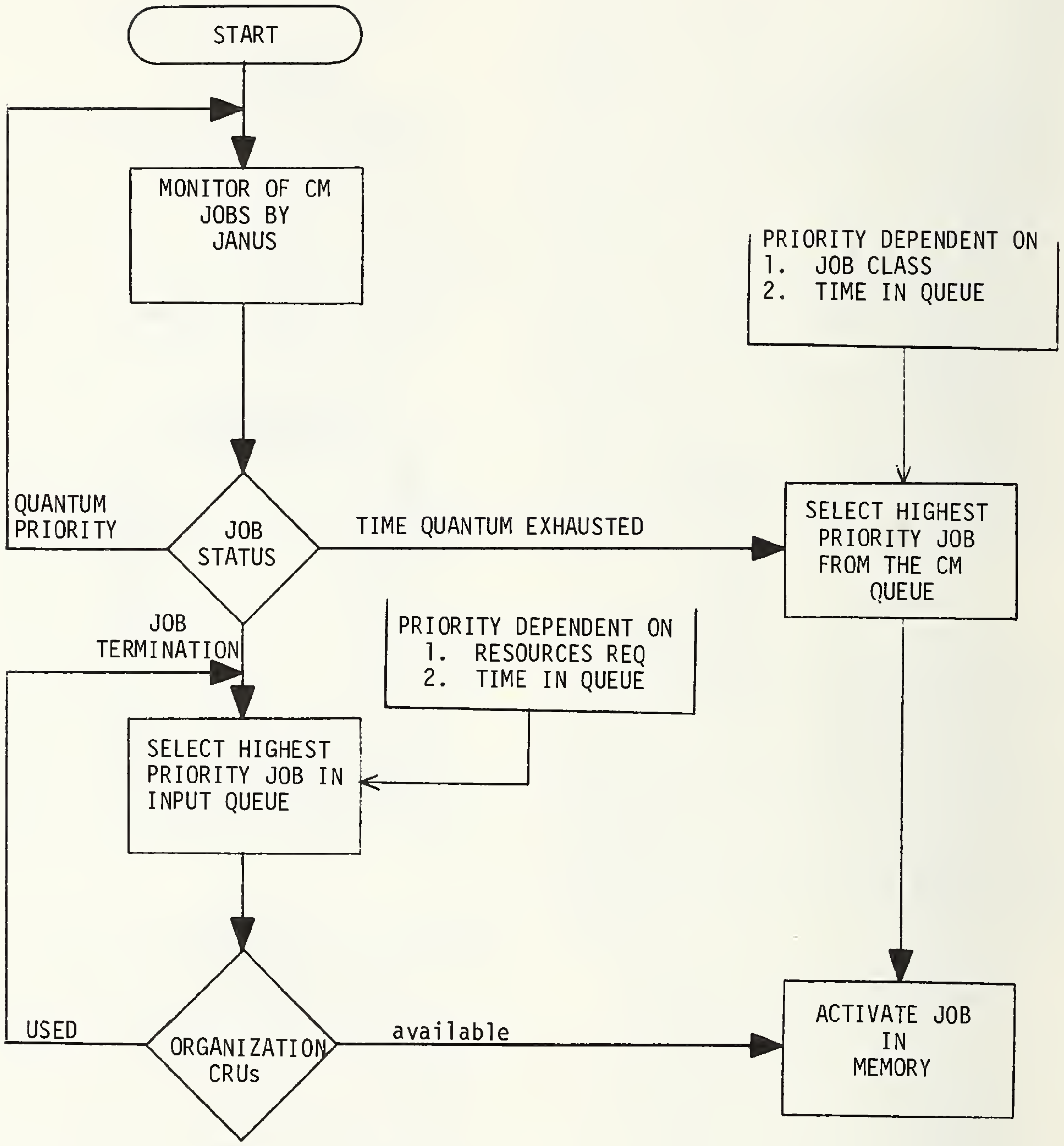

START
MONITOR OF CM JOBS BY JANUS
PRIORITY DEPENDENT ON 1. JOB CLASS 2. TIME IN QUEUE
QUANTUM PRIORITY
JOB STATUS
TIME QUANTUM EXHAUSTED
SELECT HIGHEST PRIORITY JOB FROM THE CM QUEUE
JOB TERMINATION
PRIORITY DEPENDENT ON 1. RESOURCES REQ 2. TIME IN QUEUE
SELECT HIGHEST PRIORITY JOB IN INPUT QUEUE
USED
ORGANIZATION CRUs
available
ACTIVATE JOB IN MEMORY

Figure 5. Job Scheduling Logic (Reference [1])

For each day, the following files are produced on the PDT:

- The Summary file which is a general description of the day's activities for the system as a whole.
- The Tape-Reel file for magnetic tape activities.
- The Queue file for the times a job entered and departed the job queues.
- The Execution file which contains the job identification and computer resource usage fields.

The files of interest in this study were the Queue and Execution files. In attempting to access information for each of these for a reduced data file, several problems were encountered.

The initial problem with the CLARA PDT was discovered during an attempt to read the Queue and Execution files. Attempts to access the files revealed that the PDT Execution file was missing for each day due to an error in the job control cards used in executing the CLARA program. This data source was therefore invalidated.

Compounding this problem was a previous decision by the computer center to quit producing PDT tapes based on the assumption that the historical PDT tapes were valid and would be sufficient for workload analysis. Because the historical PDTs were invalid, the computer center agreed to resume production of PDTs until sufficient data was gathered. Due to the time delay caused by this problem only two weeks' data were available for modeling. Furthermore, many of the parameter values that were supposed to be stored in the CLARA files were missing or invalid. Table 1 defines the parameters that were finally extracted from the CLARA tapes.

In order to define a minimum set of job parameters from those listed, an attempt was made to eliminate non-independent variables based on correlation and known relationships. Finally, four parameters were chosen to characterize jobs: CPU, I-O, CM, and Disk.

3.3 Results

The workload was processed by a clustering algorithm in the space of these four parameters. An existing algorithm called WISH, a version of Wishart's variant on the K-means method [7], was chosen based on availability and minimal memory requirements. Once clusters were defined, each job was tagged with its cluster identification for further processing. This allowed turnaround time, input queue time, and control point time to be analyzed as a function of cluster number.

The first matter of interest was whether natural clusters could even be found in the ASD workload, consisting of jobs from a dozen different research organizations. Table 2 shows the clusters that were found for the week preceeding the change to mostly batch and table 3 shows clusters for the week following the change. As can be seen in both cases, eight or nine clusters were defined. These tables define the clusters based on an entire week's workload. In both cases, when the data analysis was done on a day by day basis, the percentage of jobs falling into each cluster was nearly the same. That is, the $p(C_i)$ distributions were stationary on a day to day basis over the week's data. It can also be seen that many of the clusters changed between the two intervals. This is to be expected, since the policy change was a change in the workload submitted to the Cyber and not a change to the system itself.

This fact that the workload itself changed becomes important in trying to ascertain the effect on performance of the change in policy. In using the cluster approach to verify the effect of a system change, one would compare the before and after performance on a cluster-by-cluster basis, thereby determining not only the overall improvement in performance, but also the relative performance change for each class of jobs. To do this, however, requires that the workload clusters themselves be the same before and after the change. In examining tables 2 and 3, it appears that some of the clusters did remain, some disappeared, and some new ones entered. Thus the pairs (A1, B1), (A4, B3), (A5, B4), and (A7, B5) each represent the same cluster before and after the change. These pairs were determined by comparing the coordinates of the cluster centers, and requiring all four coordinates to sufficiently close in both clusters. In this case, "sufficiently close" was determined by graphical techniques rather than statistical tests. Considering only those clusters, the improvement in mean batch turnaround time was significant at the 0.001 level. The changes in mean turnaround time were 35% for A4, 39% for A5, 42% for A7, and 49% for A1. The changes for the other clusters cannot be determined, since

Table 1. Reduced Job Parameters

NAME	DESCRIPTION
Job Name	Uniquely identifies each job.
Receipt Date	The date the job entered the input queue.
Job Origin	Identifies whether the job was initiated as batch, remote batch or intercom.
Organization Identifier	Identifies the organization which submitted the job.
In-Queue	Time in seconds the job entered the input queue.
In-CP	Time in seconds the job left the input queue and entered a control point.
Out-CP	Time in seconds the job left a control point and entered the output queue.
CPU	Central Processing time in seconds used by a job.
I-O	Input-Output time in seconds used by a job.
CM	Maximum central memory in kilo-words that a job occupied, same as CMR for batch jobs.
CMR	Maximum central memory requested by a batch job.
CRUs	The number of computer resource units used by a job.
PPUs	The PPU time in seconds used by a job, includes I-O channel plus I-O overhead time.
Tape-Drives	Maximum number of tape drives used at one time by a program.
Disk	Maximum number of disk sectors used at one time by a program.

Table 2. Cluster Parameters Before Change

Cluster	# Jobs	% of Total	CPU (Sec)	I-O (Sec)	CM (K words)	DISK (Sectors)	% Batch in Cluster	Mean Turnaround Time (min)
A1	1358	29.0	2	5	3	40	21	51
A2	485	10.4	11	15	5	200	8	34
A3	184	3.9	27	31	6	300	7	87
A4	1078	23.1	3	9	15	300	94	46
A5	264	5.7	8	28	17	550	72	72
A6	340	7.3	16	25	25	1000	97	81
A7	311	6.7	76	20	11	450	29	76
A8	330	7.1	14	125	17	1000	49	63
A9	319	6.8	82	170	16	2500	34	124

Table 3. Cluster Parameters After Change

Cluster	# Jobs	% of Total	CPU (Sec)	I-O (Sec)	CM (K words)	DISK (Sectors)	% Batch in Cluster	Mean Turnaround Time (min)
B1	753	16.7	0.5	5	2	90	85	26
B2	1361	30.2	3	9	12	300	99	33
B3	702	15.6	6	14	20	400	100	30
B4	459	10.2	12	30	19	900	99	44
B5	292	6.5	84	25	16	900	89	44
B6	310	6.9	19	80	19	3500	97	58
B7	238	5.3	62	48	35	2000	100	48
B8	386	8.6	55	238	20	3000	98	58

those clusters themselves changed. This is equivalent to trying to use the model to predict in the "gaps" between clusters, as shown in figure 2b. The cluster model is only defined in the neighborhood of the clusters. The improvement in batch turnaround was also tested over the whole workload in the aggregate. Here the improvement was 30% and significant at the 0.001 level also.

The final item of concern is the ability to use an empirical clustering model, as defined in Section 2.2, to predict performance on a cluster-by-cluster basis. Preliminary results indicate that this technique cannot be evaluated in general on the Cyber data. As can be seen from the final column of tables 2 & 3, the mean turnaround times are very close for many of the clusters. Using a test on means, a few of the mean turnaround times are significantly different from each other but most are not. This indicates that turnaround time is largely independent of cluster for the data under analysis. This at first seems intuitively wrong, since the clusters are based on resource usage and one would expect (since the Cyber input scheduler favors jobs with small I-O, CPU, and memory) that turnaround time should also be a function of resource usage.

However, the ASD Computer center has added the Organizational Scheduler, as described in Section 3.1, which attempts to give priority to those organizations which have used less than their allocated percentage of resources. Thus one could expect turnaround time to also be dependent on the user organization and its total accumulated usage at the time the job entered the system. This leads to the suspicion that the organizational status has more effect on turnaround time than resource usage. Further studies are underway to verify this hypothesis. Hopefully this effect can be eliminated by including an organizational parameter in the clustering algorithm, or by modeling control point time, after the organizational scheduler has been passed. In addition, data from other computer systems will also be analyzed by this technique.

4. Summary

In this paper the concept was advanced of using workload clusters not only to define a system's workload, but also to develop an empirical model of the system's performance. The result is effectively a set of models, one for each cluster. One could then predict performance based on the cluster to which a job belongs. Since the clusters define a limited subset of the entire job space, and the system is modeled only in the domain of the clusters, the performance of a previously undefined cluster cannot be predicted since the system is not modeled in that workload region. If several clusters predict the same performance, then the performance is independent of membership in those clusters. Modeling on a cluster basis can be an aid in overcoming the difficulties of applying regression analysis to data which is nonlinear or contains interaction between variables.

The approach was applied in an attempt to empirically model the batch turnaround time of a Cyber 74 computer. The workload was successfully partitioned into clusters. The improvement in turnaround time due to a system change was verified using a cluster-by-cluster analysis for those clusters that existed both before and after the change. Finally, an attempt was made to demonstrate the use of clusters to model turnaround time. It was found that most clusters tended to result in similar ranges of turnaround time. This lack of variation in turnaround times is probably due to the locally added Organizational Scheduler. Current efforts are underway to account for this scheduler behavior and to demonstrate the applicability of the cluster modeling approach on data from this and other systems.

References

[1] Thompson, Jimmy W., Computer Performance Evaluation of Individual Workloads on the ASD CDC Cyber Systems, Masters Thesis, Air Force Institute of Technology, Wright-Patterson Air Force Base, Ohio: December 1978.

[2] Agrawala, A. K., Mohr, J. M., and Bryant, R. M., "An Approach to the Workload Characterization Problem," Computer, Vol 9, No. 6, June, 1976, pp 18-31.

[3] Svobodova, Liba, Computer Performance Measurement and Evaluation Methods: Analysis and Applications, American Elsevier, NY, 1976.

[4] Ferrari, Domenico, Computer Systems Performance Evaluation, Prentice-Hall, Englewood Cliffs, NJ, 1978.

[5] Agrawala, A. K., and Mohr, J. M., "Some Results on the Clustering Approach to Workload Modelling," Proceedings of the 13th CPEUG, New Orleans, LA, Sep, 1977, pp 23-38.

[6] Frazier, Larry R., "A Simulation Model for Capacity Analysis," H6000 Computer Performance Evaluation Users Group Meeting Minutes, Maxwell AFB, AL, 6-8 March, 1978, pp. 236-244.

[7] Anderberg, M. R., Cluster Analysis for Applications, Academic Press, New York, 1973, pp 169-170.

PERFORMANCE COMPARISON MEASURES FOR COMPUTER SYSTEMS

Ints Dzelzgalvis

International Business Machines Corporation
P.O.Box 390
Poughkeepsie, New York 12602

The need for performance ratings of computer systems is as fundamental as the measures of horsepower, calories, watts, etc. Yet the computer system performance measures that are of sufficient quality to quantify and evaluate reasonably the differences of systems of differing architectures, design, and operating systems are hard to find. That does not mean that comparisons are not made. In fact, comparisons are made quite frequently; however, the quality and value of the results is at best questionable. It will be shown that the classical performance measures of cycle time and MIPS never were accurate measures and with today's added complexity and clever system design tradeoffs, these measures can be downright distortions and/or inversions of the facts. The measures of thruput and response time/ turn around time are the real candidates for computer system performance measurement; however, their proper evaluation can be a tedious and costly undertaking. In addition, due to the complexity of the evaluation process, these measures are prone to error and biases which can quickly destroy their quality as proper measures to be used in a comparison. This paper focuses on the evaluation process of the response time and thruput measures, identifies the potential pitfalls, suggests some useful approaches to their proper evaluation, and identifies key problems yet to be resolved.

1. Introduction

The need for performance ratings of computer systems is great. The customer needs them to select the system that will satisfy his requirements at the minimum cost. The vendors, planners and designers need them to establish design points. The computer system manager needs them to evaluate the efficiency with which his system is running today and determine the potential that remains to be had.

Any discussion of performance must also include a discussion on price. Price is generally determined by the configuration which is a byproduct of the performance analysis. Pricing of a configuration is generally a fairly straightforward exercise and, thus, will not be discussed in great detail.

Performance rating implies a measure. Traditionally, the performance measures used by the industry have been either cycle times or instruction-execution rates (in millions of instructions per second MIPS) of the main processor in the system, the instructions here being machine level instructions. These measures today are not only misleading, but downright irrelevant as the measures of the performance power of today's data processing systems. One of the key reasons for this is that the user's units of work today are transactions, jobs, sessions and not instructions. In fact, the user has little knowledge of the number of instructions that his work requires because often the application code that processes his transaction was written by someone else, usually, in a higher level language. In addition, the software services provided to the user by the operating system require a substantial number of instructions themselves and in many real time systems tend to dominate the overall instruction count (path length).

Figure 1 provides an example of comparing several systems on the basis of the internal processor measures, KIPS, (thousands of instructions per second) and a comparison using the user workload measure thruput. The reader will quickly note that the internal comparison leads one to believe that the processor from the System A family is a substantially worse price/performer relative to the other systems. On the other hand, when viewed from the thruput point of view, System A processor is a substantially better buy than any of the others. The numbers in the chart reflect real system families that were studied in depth by the author. The inferences and conclusions stated in this paper are the results of the study. Because thruput is very workload dependent, the comparisons based in Figure 1 are valid only for the environment which is represented by the workload used in the study. It is for this reason

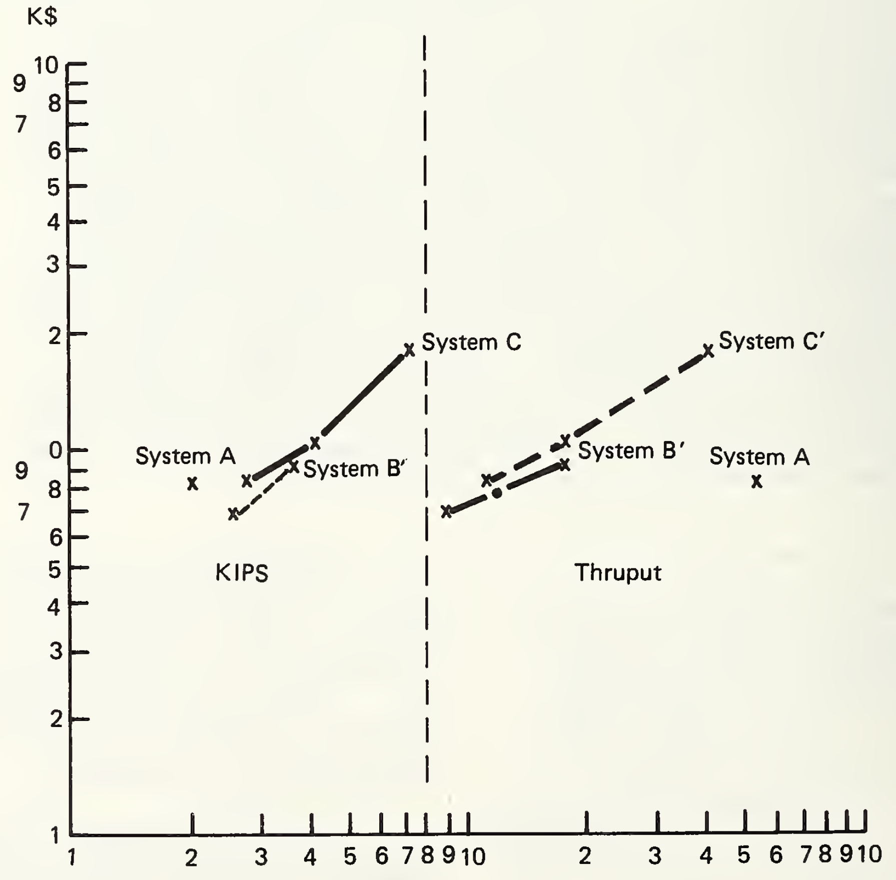


Figure 1. Dollars Vs. Performance

that anonymity will be maintained of the actual systems that were studied.

The paper describes a method for developing the necessary figures of merit that will allow one to perform reasonable cross-system comparisons. The method is depicted as a process whose major elements are identified in Figure 2. This process was tested out in a study of three systems with radically different architectures and design. The results of this study will be used to illustrate the key elements in the process.

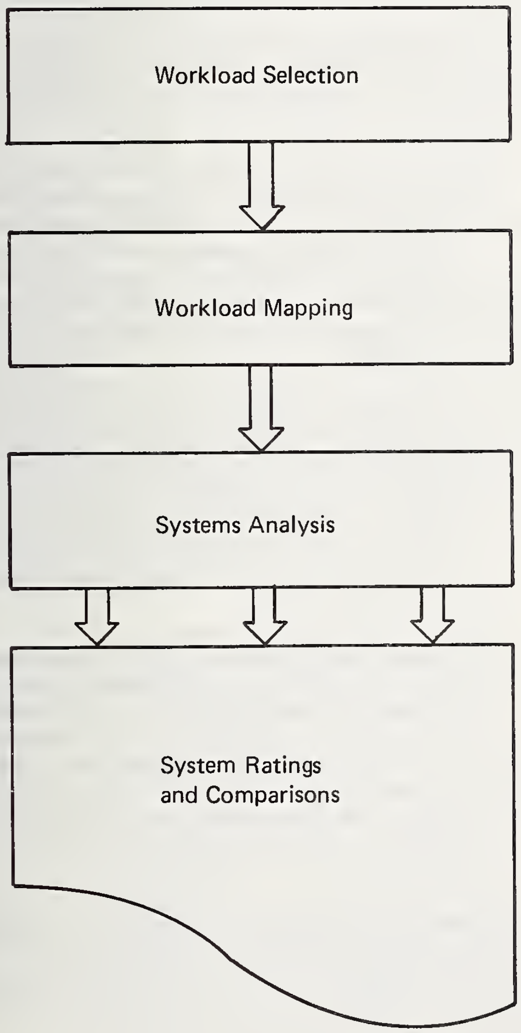


Figure 2. The Performance Evaluation Process

2. Units of Work

The precise definition of the unit of work depends on the work type. The three most common work types include: batch processing where the unit of work is a job; interactive processing where the work unit is a transaction which in turn might be made up of a number of messages; and session-oriented processing, such as program development on an interactive time sharing system (for example, TSO).

2.1 Batch, Turnaround Time and Thruput

The number of jobs that the system can process over a given period of time is referred to as thruput. Turnaround time is time that elapses from the job's entry into the system to the time that the job is completed and the output is available to the user.

2.2 Interactive Environments, Response Time and Thruput

Response time is generally defined as the time from entering the last character of the input message to the receipt of the first character of the output message. Note that a transaction can consist of several messages. Generally the transaction response time is defined as the overall average message response time. The trivial message response times are excluded. Trivial messages are messages that involve little or no work. Examples are notifications of system or transaction status. Thruput, of course, is measured in the number of transactions that the system can handle over a period of time at a prescribed level of response time.

2.3 Session Environments, Response Time and Thruput

The response time is defined similarly to the interactive environment. However, thruput is defined in terms of the number of users that the system can support and still provide a given response time.

The definitions have purposefully been left with some flexibility which, as will be seen later, is needed to match the work unit requirements. The exclusion of terminal delays, implementation time, etc., has been done for practical reasons. Factors, such as terminal characteristics, implementation tradeoffs, availability, etc., are treated as separate variables from systems performance.

3. Workload

The process for evaluating the performance measures must begin with a proper selection of a workload. This step is often one of the more difficult ones to resolve, because little if any attention is given to defining a rationale for choosing an appropriate set of workloads for system rating prior to the study itself. It will be seen that the workload selection is crucial in the ultimate positioning of a given system against other systems. Thus, as the study proceeds and the positioning is revealed to the participants, changes in workload become the only means for changing the positioning of a given system; hence, the pressure for change by advocates of the various systems involved.

The term "workload" as used in this paper includes the jobs, transactions, sessions, etc. that a given system is required to execute at a given time. The workload differs from a classical benchmark in that it should utilize during its measurement period all of the resources of the system in roughly the same ratio as one would expect in the real environment. This is in contrast to a typical benchmark which consists principally of computer instructions for exercising the system's processor and a small data base to exercise its I/O handling capability. A workload also exercises the system's capacity and, thus, requires a full configuration. It allows the analysts to study reasonable price tradeoffs as well as performance.

4. Workload Definition

If the workload is to be used to study two or more systems of differing architectures and design, its form of definition should be systems-independent or generic. The definition should include:

- Scenarios of user work units
- Frequencies of work occurrence
- Data base definitions
- Geographical dispersion requirements
- Capacity requirements

There have been several attempts at formalizing and standardizing the semantics for such a workload; however, none has received a wide acceptance.

It cannot be emphasized strongly enough that selection of the proper workload is absolutely essential. The often questionable credibility of the whole systems analysis profession, because of some of its monumental errors in projecting a reasonable systems performance capability, can be traced to the selection of the wrong workload. We must recognize that this tendency for erroneous workloads is helped along by three forces:

- The lack of an analyst's appreciation for the influence of workload on the final results. The analyst, because of his training, generally pays much more attention to the modeling and measurement aspects of the analysis.

- Because of deadlines, lack of resources, etc., the analyst as well as the analysis requestor, is generally looking for shortcuts. This often results in the use of a benchmark and at times even a kernel, under the guise of a workload.

- The selection process is often helped along by a vendor who naturally would like to insure that the workload will frequently exercise the functions that his system does well and minimize the frequency of the functions that it does not do so well.

How often have we seen a system being selected on the basis of its ability to execute optimized floating point operations, while the customer's work that is targeted will only be running COBOL!

For a given customer, workload selection is usually a fairly straightforward, if tedious, process because most of the work is either running or well defined. Workload selection for a planner/designer on the other hand is a more difficult task because the workload must represent many customers. Because study of each customer is impractical, statistical methods need to be employed in selecting a practical sample and then projecting the results for the whole. The selection is further complicated by the usual planning objectives. These are that the workload be representative of system usage in the marketplace, that the workload stress key system resources and major functions, that the workload relate to history, and that it be complete and well defined.

Establishment of a criterion for representativeness requires current and planned usage studies of the customer base for each of the systems being studied. The study will also have to collect data on application characteristics, transaction loads, data base characteristics, etc. The resultant statistical data then should serve as a base for establishing the workload environments, wherein all of the systems of interest are competing. A common problem in workload definition is that the ultimate choice is too loosely defined and incomplete. This allows the various users of the workload to make assumptions that substantially affect the system performance which, unless carefully managed, can distort the final results.

In the case of our study, the workload finally selected includes the five applications described in Figure 3A. Furthermore, the LATS Company for which the workload is named is a small company all housed in one building. Hence, there is no need for a communications network. The interactive terminals are all locally attached. Furthermore, the workload in LATS is such that each application has terminals dedicated to it, that is, a given terminal will interact with only one application (Figure 3B).

The LATS Company intends to use its computer system to take orders interactively for its products, prepare shipping instructions in real time, but perform its invoicing instructions in batch mode. The invoicing job is performed in two steps. First, the invoices are assembled from the master files and previously generated order files. The finished invoices are then assembled in an invoice data set. Finished invoice data sets are then passed on to the spooling job which prints them out on a 300-lpm printer. To insure that the invoices will be available when needed, it has been established that the system must be able to create a minimum of 280 invoices per hour. The peak hour transaction rates for the LATS application are summarized in Figure 3C. Figure 3D illustrates the application message flow. Note that all applications except for order entry consist of a single message pair. The handling of each invoice in the batch job is analogous to the message concept in the interactive applications. Figure 3E depicts the data referencing flows between the files.

Applications	Operations Mode
Order Entry Shipping Key Data Entry	Interactive
Invoicing Spooling	Batch

Figure 3A. LATS Workload

5. Workload Mapping

The next step in the evaluation process is the mapping of the workload onto each of the target systems. This requires an interface which, for our purposes, will consist of a programming language, data base language, communication access method, operating system, etc. In addition, we need a good understanding of each of the systems so that the mapping can take full advantage of the system features. For proper comparison one must make sure that the interface levels are "equal" in function. It would not be very meaningful to compare System A with IMS-type of data support vs System B with a BDAM level of support. Insurance of the interface consistency is not a trivial exercise. To illustrate the kind of problems one faces here, let us return to our sample. The LATS Company has identified three systems that it feels are worthy candidates.

System A is well established and functionally mature. This system is based on an architecture that is existent and well understood. It provides COBOL as well as other rich system functions. However, the system is relatively expensive and, due to its age, the interfaces do not provide all of the new design features which enhance ease of use and system integrity. It is reasonable to assume that the system is very reliable.

System B is an up-to-date system which provides all of the function required, including COBOL. Incorporated into its interface are many integrity and usability features. The system, because of its advanced technology, is very attractive in price. The system is believed to be reliable. The COBOL offering has all the function needed for LATS and is a direct follow-on from a previously existent offering.

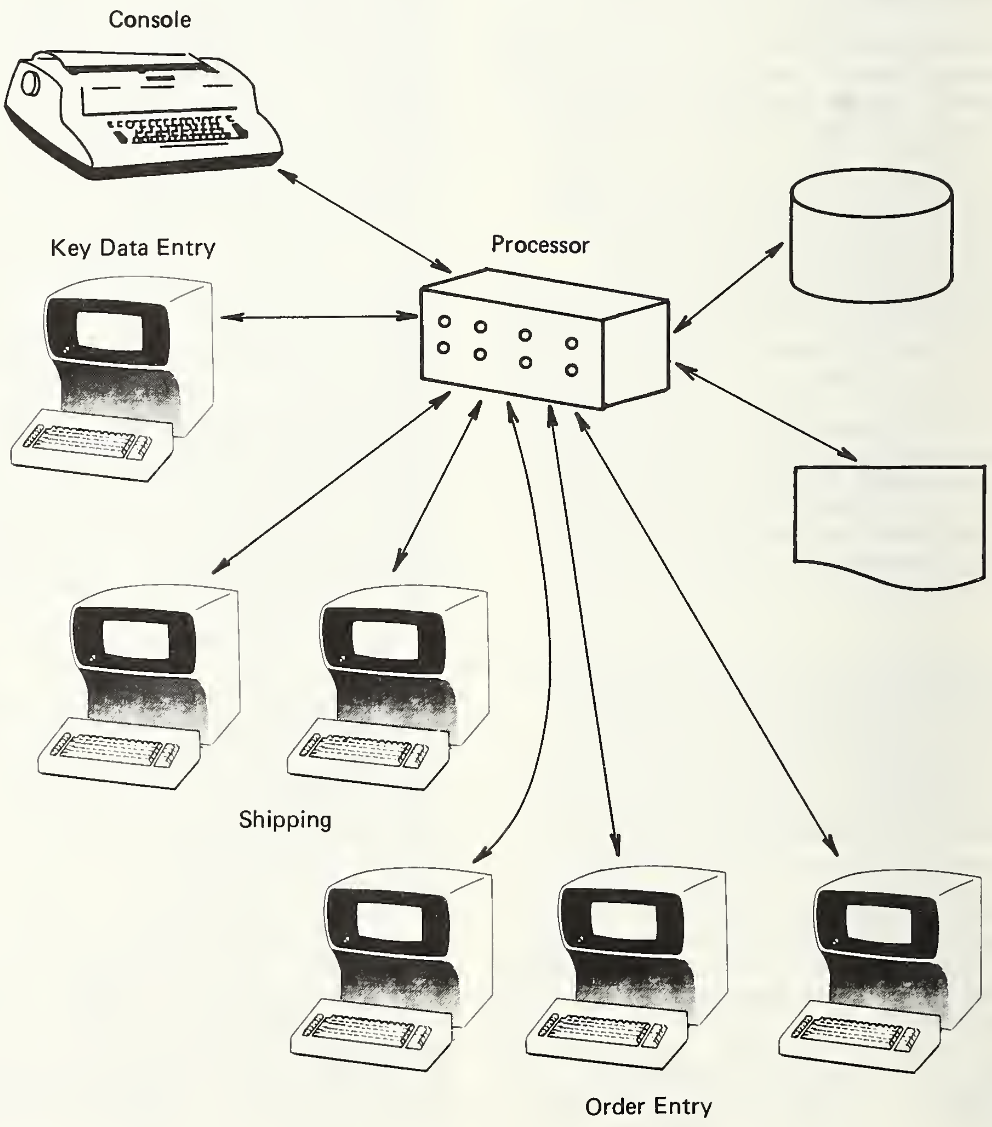


Figure 3B. LATS Workload Base Configuration

	Number of Terminals			
	6	12	18	24
Order Entry (Orders/Hr)	94.5	189	283	378
Shipments (Orders/Hr)	624.0	1,248	1,872	2,496
Key Data Entry (Entries/Hrs)	54.5	109	164	218
Invoicing (Invoices/Hr)	240.0	280	280	280
Spooling (Lines/Hr)	18,000.0	18,000	18,000	18,000

Figure 3C. LATS Rates Workload Peak Hour Transaction Rates

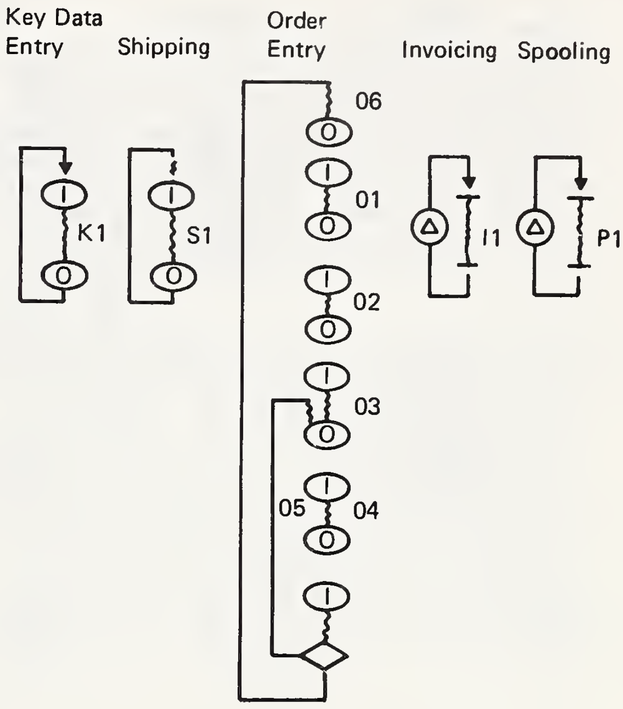


Figure 3D. LATS Workload Transactions

System C is a relatively low-priced system that has been well established in some environments for several years. The system offers a broad range of software support including transaction processors, COBOL, and data management support that would be useful for the LATS workload. However, the COBOL and data base package available for measurement were brand new offerings with potential exposures of coding and design defects.

The first problem in the interface selection is on System C. There actually are two choices of source language: COBOL, which is preferred and FORTRAN. However, COBOL is risky on System C because the functional limitations and potential defects in the first release could negate other advantages. The study alternative on this system is to implement the applications in FORTRAN which, as far as LATS is concerned, suffers in usability value. The required functions lacking in FORTRAN would be hand coded in Assembler. An experiment of this situation was performed. It confirmed that the use of the first release software available at the time would definitely have made System C unattractive, while the use of FORTRAN with some RYO function satisfied function and reliability requirements. Furthermore, a more detailed study revealed that the FORTRAN implementation closely approximated the intended COBOL implementation when all of the early release limitations in the Vendor's COBOL were removed. This result then indicated that LATS Company should use a FORTRAN implementation for the comparison, but, if Vendor C is selected, careful study would have to be made on the plans for use of its COBOL.

A second problem in the selection process occurs on System B memory management alternatives. This company's system provides two alternative forms of memory management in its transaction manager. On the one hand, the user could select a memory management option which utilizes the minimum memory by freeing most committed memory whenever the transaction is in a major wait state (for example, waiting for a message from the terminal). This, of course, requires additional processor instructions to STORE and RESTORE the status as well as to transfer and assemble the resources. With expensive memory, and lightly loaded processor, this alternative would be a good choice. The other memory choice ignores the memory savings potential and, thus, saves the processor an ad-

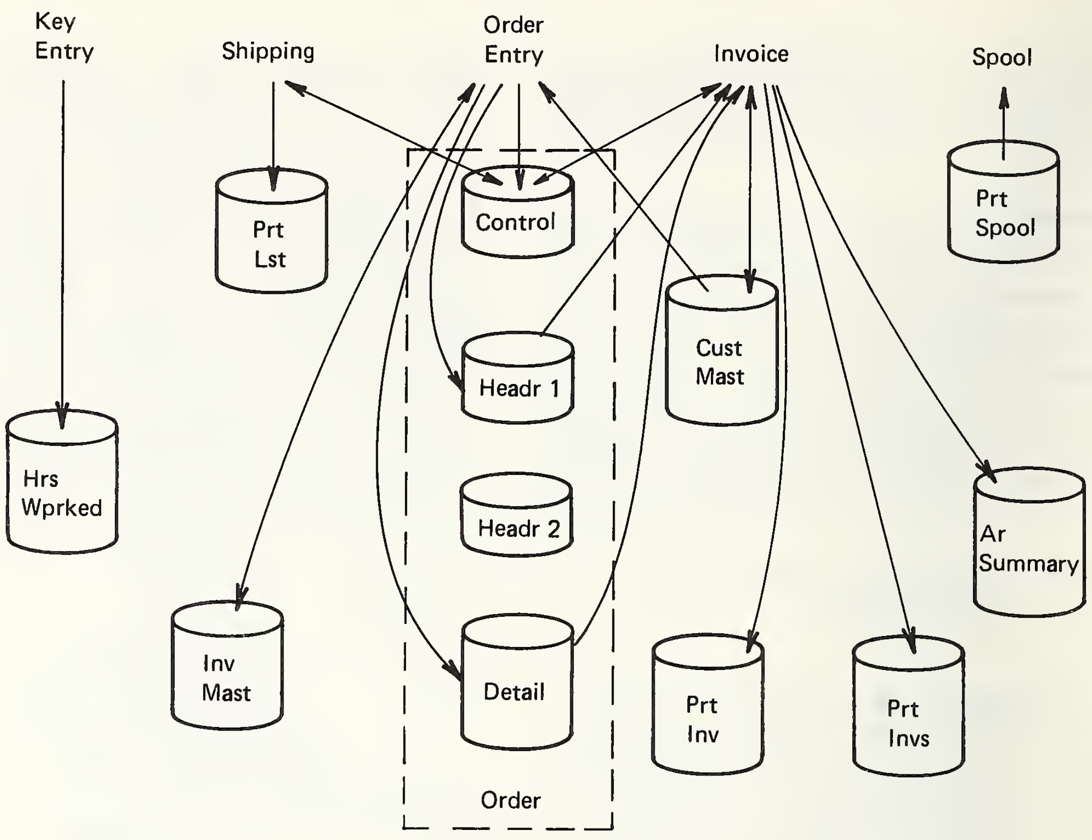


Figure 3E. Data Referencing Flows for LATS Workload

ditional processing load. Let us call this memory management scheme as "fast." The selection here should be preceded with an evaluation of both alternatives and a projection of future memory prices. The fact that the memory prices in the industry have fallen dramatically within the last year would indicate that "fast" might be the right choice.

6. Systems Analysis

After workload mapping, the analyst is ready to proceed to the evaluation phase. In the ideal but improbable case, the analyst could code and measure the workload on each system. More likely he will have to depend on modeling. The choice of modeling techniques and aids is great and generally each analyst already has a favorite. The author has yet to find a technology that seriously inhibited the accuracy of his results. Much more important than the identification of a specific modeling technology, are the depth of the model, skill and cleverness of the modeler, and the accuracy of the parameters and algorithms being employed. The latter fall into two categories, hardware characteristics and software characteristics. A sample list of each is provided in Figure 4. The hardware characteristics are generally available from vendor literature. Software characteristics are more difficult to determine; however, we have found most of what is needed in software support manuals or through guesses resulting from system design implications. The one exception is path length. Our solution to the path length was:

- To identify frequently used functions, contract for time at service bureaus which had the machines installed, and then measure the path length using simple programs that repeatedly called on these functions.

Hardware Characteristics	Software Characteristics
MIPS (million of instructions per sec)	Path Length By System Function Including
Cache Size	Data Access Path
	Message Access Path
I/O Structure	Transaction Mgr. Path, etc.
Service Times	
	Memory Management Algorithms
DASD Access Characteristics (seek time, rot. delay, xfer time etc.)	Other Resource Mgr. Algorithms
Etc.	

Figure 4. Key Model Input Requirements

- For the less frequently used functions we estimated the path length by sketching out a primitive program that would perform that function.

For a sample of the path length for the three systems studied, see Figure 5. The description of how the measurements were constructed and carried out is a subject for a whole paper and will not be dwelled upon here, other than to say that the measurements required about one man-month per system of which about 5-10 hours were actually spent on the system itself.

Modeling each of the systems required considerably more effort. Unfortunately modeling is still an art and, hence, very dependent upon the skill and ingenuity of the modeler. A good modeler can make a difficult job look easy, while a lesser skilled modeler makes a trivial job appear very complex. However, even the best are limited to the information that they are provided. To illustrate the problems of the latter, let us return

Receive Message from Terminal

		Language			
System	Function	Assembly	Fortran	Cobol	Basic
0	Per msg.	4,000	11,000	N/A	21,000
	Per char.	462	N/A	N/A	727
P-X (2)	Per msg.	1,700	4,500	N/A	N/A
V	Per char.	155	182	N/A	N/A
P-Y	Per msg.				4,891
V	Per char.				2,309
P-X (3)	Per msg.		6,680	4,000	
V	Per char.		291	277	
P-X (3)	Per msg.		6,380		
U	Per char.		218		
Q	Per msg.		19,300	17,400	
	Per char.		0	0	

Total RECEIVE MSG. pathlength = "Per msg." + No. of char. in msg. "Per char."

Legend:
System designation code includes system name (e.g. P), name of operating system (e.g. X), which release (e.g. (2)), and the terminal type (e.g. V).

Figure 5. Intrinsic Function Pathlengths (Native Instructions)

to one of the systems in our study. In reviewing the hardware and software descriptions, nothing unusual was noticed. When the model was being calibrated, a serious bottleneck was observed at the DASD controller by the measurements, but not the model. Detailed analysis revealed that the DASD controller, operating within the systems setup that was being measured, did not properly overlap SEEKS between the two devices that were attached to it. It turned out that the cause was the lack of some software support within the operating system that we were using. This example also points out the value of calibration.

7. System Comparisons

Having a model, we can now develop our desired results. Using the sample LATS workload and models of the three hypothetical systems, we can derive a service time vs thruput relationship for each of the systems.

7.1 System Parameters

Before interpreting the results, let us discuss some essential characteristics of the three systems.

System A. 32-bit architecture. Stand-alone channels, two levels of interrupts. The system utilizes a type of virtual memory space managed by software with hardware assists. The processor executes about 200 thousand instructions per second (KIPS) on the type of work studied.

System B. 16-bit architecture. Bus with native adapter type of I/O architecture. Several levels of interrupt. All operations performed via registers. The architecture defines a virtual memory space which is managed by software with some hardware assists. The processor is rated at 300-400 KIPS on the work studied.

System C. 16-bit architecture. Bus structure for I/O. Several levels of interrupt. The operating system provides for fixed partitions and swapout. The processor is rated at 600-800 native KIPS.

7.2 Configurations

The configurations required to perform the LATS workload were very similar. They consisted of:

- A processor
- 512 bytes of memory
- 2 Winchester type DASD spindles
- A 300 lpm printer
- Enough CRT terminals to support the required thruput
- DB/DC software (when available)
- COBOL/FORTRAN
- VSAM-like access method

The one difference is that System C terminals have only enough capability to allow data changes to occur on one line at a time, while the other systems have terminals that have a large enough buffer to allow full screen operations. The significance of this is that System C terminals will require more interactions with the processor, thus requiring additional path length per unit of work. Another more subtle difference between the systems involves the user's interfaces. System C software provided the least function for transaction management and consequently required additional programming on the part of the user in the application portion of his system. System B provided all the function required, but it was less modular than System A. This resulted in System B having to invoke some unneeded modules which generated additional path lengths. Another factor contributing to System B path lengths was the constraint of 512K bytes of memory which forced it to use a systems software provided memory management scheme, which by saving the need for additional memory, requires additional path length. To understand System B's full capabilities, two different sets of its implementations are shown. The RYO version uses the minimum system-provided functions and consequently avoids some of the overhead identified above. Of course, the disadvantage of this is that the LATS Company programmers would have to provide significantly more programming support than with the other implementations. The other System B implementation is designated as full DB/DC.

7.3 Results

Figure 6 gives a plot of the number of terminals on the system against average non-trivial message

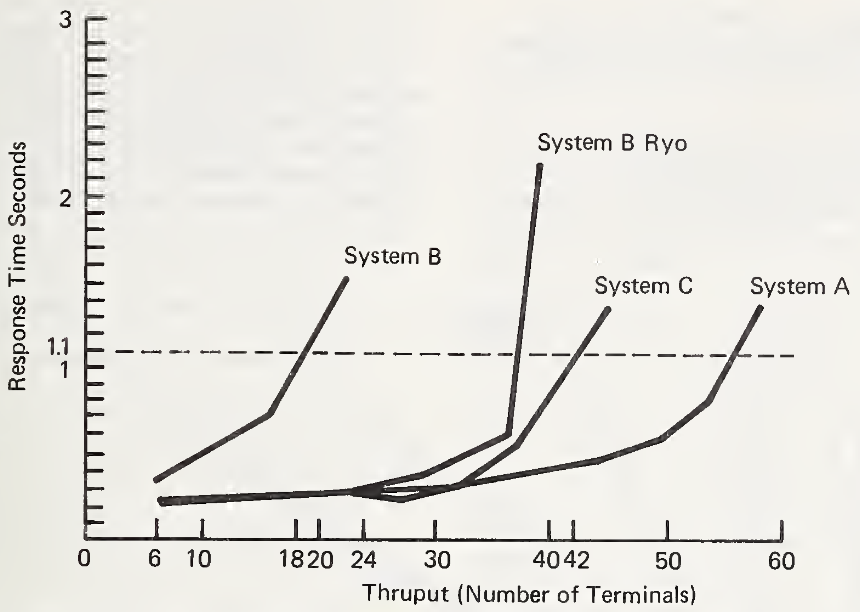


Figure 6. LATS Application Average Response Time

response time. As expected, the response time increases with increasing system load. Generally there is a threshold of response time beyond which the response time is unacceptable to the user. In the case of the LATS Company this threshold turns out to be 1.1 seconds on the average. Thus, a way to estimate a given system's performance capability is to look at the thruput capability of the system at the threshold of the user's response time. In our case this would be:

System A	- 54 Terminals
System B RYO	- 36 Terminals
System B	- 18 Terminals
System C	- 42 Terminals

Although the response time threshold for our workload was given, our approach of looking at the performance capability has caused at times some heated debates. There is a school of thought that believes that there is a universal constant that represents the response time threshold for all users in all circumstances. Those who believe in this then quickly engage in an argument on whether 1.1 is appropriate. Our response to the argument is that irrespective of what that constant might be, the 1.1 second response time on this workload appears to represent a time when the system is becoming saturated, and the response time is beginning to climb quite fast. Thus, as luck would have it, the response time threshold point and the system saturation points coincided for all of the systems under study. The saturation point equivalence will become more apparent when we look at processor utilizations. Another school of thought appears to want the performance ranking to be done on a response time scale at fixed thruput. Our problem with this philosophy is that we do not know how to measure the value of faster response times, especially when they are under the threshold, and the differences are within 1/10 seconds.

The most startling aspect of the thruput results in Figure 6 is that the thruput (number of terminals) capability of the systems has no correlation with the processor (KIPS) ratings that are within the systems. The 200 KIP processor in System A has more thruput capability (54 terminals) than either the 300-400 KIP processor in System B or even the 600-800 KIP processor in System C. (Note Figure 1.).

In Figure 7 we are plotting the processor utilizations against the number of terminals for each of the systems. Note that each of the systems, except for System C, at their previously developed thruput capability points, are about 70% utilized. Of all the resources at this point the processors, except that of System C, were the most utilized. System C has a bottleneck at the DASD devices.

The System C bottleneck is caused by its memory manager. The real memory requirements are illustrated in Figure 8. As can be seen from the chart, the 30 terminal case for the system exceeds the available memory and consequently swapping of partitions must commence. Swapping generates additional work within the CPU (note the change in the CPU utilization curve for System C) and ad-

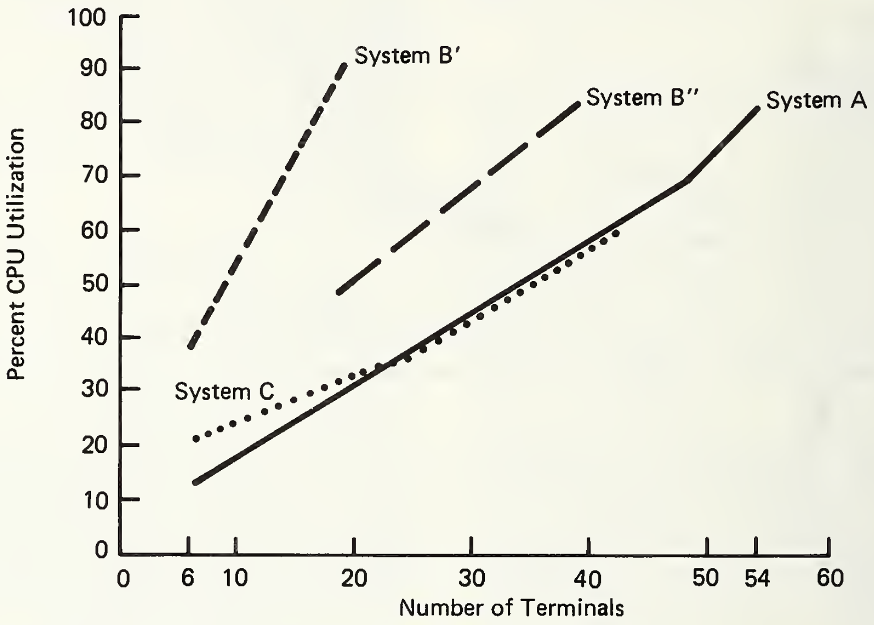


Figure 7. LATS CPU Utilization

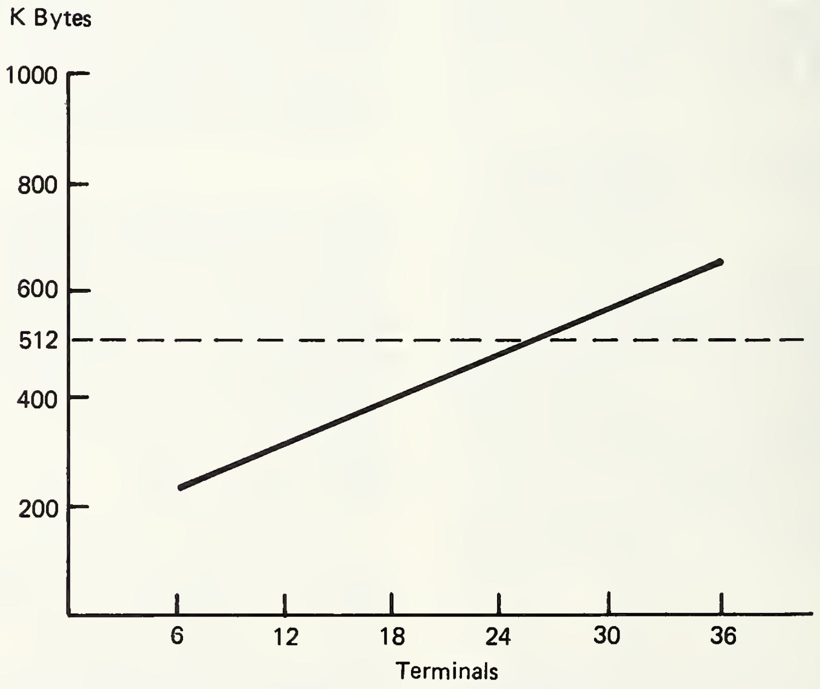


Figure 8. Memory Requirements for System C Based on LATS Workload

ditional load on the DASD subsystem, from which the data are swapped. This additional I/O activity is responsible for saturating the I/O subsystem on System C. This bottleneck can be alleviated by adding another I/O device. Had that been done, it is most probable that System C thruput capability would be greater than shown. System B avoided the I/O problem through use of its in-line memory management scheme. System A avoided the problem by providing a demand paging mechanism.

The above study has provided us with two important pieces of information: the performance capability of the system, as well as a knowledge of the configuration required to achieve the different levels of performance. By pricing out the latter, we have price or cost which we can use to plot performance against price (Figure 9). For a given user, (whose workload requirements are known) the chart can provide a quick reference of the best choice. For a vendor, the chart identifies its competitiveness (or lack thereof) over a range of performance capability. For a planner of future systems (assuming for the moment that the four sets of results represent four different system offerings), the chart can provide insight as to proper granularity of offering spacing.

As is generally the case, most successful studies generate new and additional questions. The more intriguing one in this study is the reason for almost a negative correlation between systems thruput and processor capability. The reason, of course, is that the faster processors require more instructions to do a similar function than the slower processor in System A. This shows up quite clearly in the path length comparisons of the LATS workload (Figure 10). It is interesting to note that none of the more obvious explanations, such as the 16 vs 32 bit architecture, differences in instruction sets, or the other architecture features, were the major contributors. Instead, the channel and memory management strategies and assists made the largest difference. The channel difference shows up most clearly in the spooling job (System A had one while the others did not). System A had dynamic paging with hardware assists to handle memory overflow problems. System B used the previously discussed scheme, which resulted in a very substantial path length increase, while System C uses a SWAP technique, which puts most of the burden on the I/O paths in the system. The one other significant factor which did not appear to affect System C performance too badly was its use of a less than full function CRT.

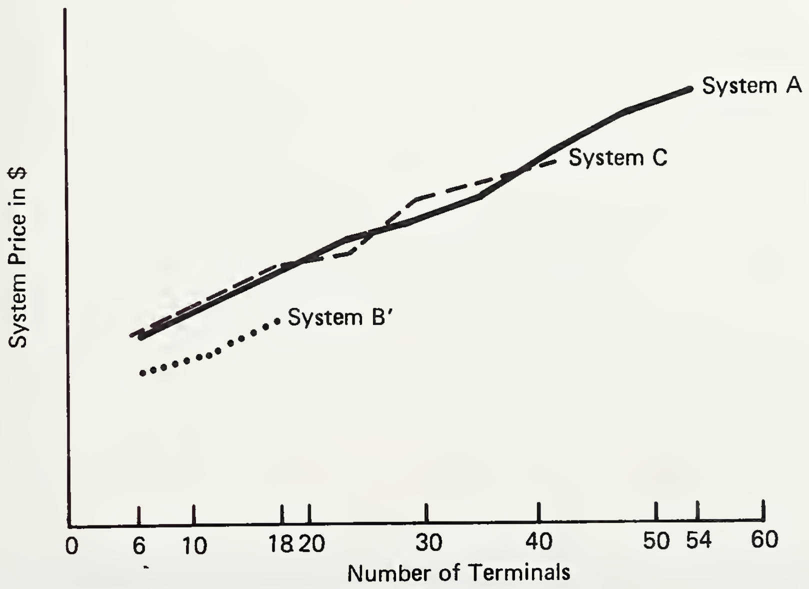


Figure 9. System Price Vs Number of Terminals

	System A	System B RYO	System B	System C
Order Entry (Per Order)	424.2	969.7	1807	554.4
Shipping (Per Order)	31.3	60.0	127.1	73.1
Key Deta Entry (Per Labor Card)	22.3	41.8	119.3	118.4
Invoicing (Per Order)	31.8	61.3	61.3	83.7
Spooling (Per Line)	.8	12.1	12.1	16.7

Legend:
Native Instructions (K)
Includes PIO

Figure 10. Pathlength Comparisons

The study also has suggested that the performance differences of a given system are very sensitive to the workload which is being used to evaluate the systems. Had we had a scientific workload that consisted of a lot of small computation-intensive programs which were not affected by some of the system factors discussed earlier, then we would have seen radically different sets of comparison results. Had we used a 1200 lpm printer, System A would have stood out even more. This sensitivity to workload implies that prudence should be used when comparing systems. It also illustrates that raw native MIPS cannot be used as a figure of merit for system power.

CPE Tools

EVENT DRIVEN CAPACITY PLANNING

Springer W. Cox

Digital Equipment Corporation
Maynard, Ma. 01754

Accurate performance prediction for capacity planning has historically been hindered by inadequate workload descriptions. Present techniques often rely on accounting information without considering its sufficiency as a basis for performance prediction. Further, the performance reports generated for capacity planning often force the analyst to take sizable intuitive steps in reaching his conclusions. As a result, planning decisions must be based on predictions of unknown accuracy and sensitivity to error.

A performance prediction system for capacity planning is under development. The important events of system/workload interaction are recorded and used to drive an efficient hardware/software simulation model. After each workload has been traced, the model can predict performance under a wide variety of loads, mixes, and configurations. For a small but diverse domain of demand paging environments, accurate response time predictions have been achieved.

Keywords: Capacity planning; hardware monitors; modeling; performance evaluation; performance prediction; simulation; validation; workload characterization; workload management.

1. Preface

The objective of this paper is to describe the development and validation of a performance prediction system intended for use in several capacity planning situations.

The structure of this paper reflects the structure of the development/validation project. First, I consider the need for a quantitative model of known accuracy. I proceed to a choice of methodology based to some extent on intuitive considerations. Next, I briefly discuss the evolution of dependencies from a project management point of view. Then I explore the structure of the model and its validation, in some detail. In the conclusion of the paper, I condense the results and relate them to the overall context.

The resulting modeling system, called the System Performance Predictor (SPP), has several distinguishing features. I believe the workload characterization represents some new applications of techniques. The level-of-detail falls between instruction traces and accounting data. The virtual memory demands are characterized by lists of distinct page identifications. These page references are revealed by artificially induced page faults. During simulations, pages are treated separately and are not grouped into "mean working sets" or some other aggregate. Even so, the processing requirements of the SPP are not excessive.

The design of the tracing program followed from the needs of the simulator, not the reverse. In order to attack the workload-dependent CPU power problem, I implemented a workload instruction sampler within the tracing program. The method used to quantify workload demands attempts to separate effects which are intrinsic to the workload from environment dependent effects. The latter are more easily handled by the simulator.

Perhaps the most important distinction is the scope of the validation. Many synthetic environments of gradually increasing complexity were simulated first. The final validations are to be performed on unfriendly (never seen before) workloads. The resulting accuracy estimates will be associated with routine use of the SPP. The overall accuracy is a measure of the sufficiency of our workload characterization as a basis for performance predictions in a demand paging environment.

The objectives of the project and expected use of the model led to the development of a workload trace program, a trace-reading interface, and a trace-driven simulator. These subgoals led to a large number of measurement objectives for which a hardware monitor was adapted. Finally, the output of the model was compared to validation runs on the real system driven by a remote terminal emulator (RTE).

2. Introduction

2.1 Motivation

There are many decisions a computer installation manager must make which affect the quality of service received by the users of the system. Often, workloads tend to change and performance tends to become degraded as users find new ways to make themselves more productive. In order to control these effects, the installation manager must make judgements about how the data processing budget should be spent. Also, the effects that changes in budget will have on the users of the system must be anticipated.

One step in this complex decison-making process is to predict changes in the performance of the data processing system that occur when certain options open to the installation manager are exercised. For example, if the paging activity of a virtual memory system seems high, is it better to buy more memory, buy a faster swapping device, use a slower processor (to save money), use a faster processor, or do nothing? An accurate performance prediction methodology has great value in such situations. The System Performance Predictor (SPP) was intended to provide a solution to these and other problems which must be faced by managers of DECSYSTEM-20 installations.

An important deficiency in present methods is that when a prediction is made, there is no quantification of the error. Thus, sizable resources can be gambled on predictions of unknown accuracy obscured by an unknown amount of statistical noise. Further, the sensitivity of these decisions to small errors in assumptions is generally not known. For these reasons, the accuracy of the System Performance Predictor is being vigorously tested. The environments where the accuracy of the SPP is to be measured attempt to span the environments and workloads used on real systems. The model is to accurately predict the following changes to the execution environment:

1. Change in direct access storage device (DASD) performance characteristics.
2. Change in number of DASD devices.
3. Change to new DASD attachment configurations.
4. Change in real main storage.
5. Change to different power CPU.
6. Change in terminal and batch load (number of terminals, think time distribution, etc.).
7. Change in workload mix.

Other capabilities may be added later.

2.2 Choice of Methodology

The decade of the 1970's has seen a great improvement in the diversity of tools available to computer performance analysts. Although many techniques seem to have their essential features rediscovered periodically, many genuine advances in modeling techniques have occurred recently. Unfortunately, truly objective studies of applicability and accuracy are difficult to find. The result is that performance analysts are faced with a bewildering array of performance prediction techniques and claims of accuracy which are occasionally quite misleading.

Many models for performance prediction have been validated only in a very undemanding sense. Strict validation means that both accuracy and statistical noise associated with the predictions have been evaluated for similar workloads and execution conditions. The current lack of standardization in modeling methodology has unfortunate effects. One of them is that validation reports must be examined in extreme detail if any results are to be applied elsewhere. It is not always clear which of the model's predictive powers have been tested by the validation.

Table 1 is a list of modeling techniques considered in the present project. It is roughly ordered from top to bottom in increasing accuracy, increasing input requirements, increasing development costs, increasing lateness of availability and increasing cost of operation. This list is not complete and reflects the prejudices of the author.

Table 1 Performance Models

1. Linear Constraint
2. Stochastic
   - A. Simple Queuing
   - B. General Markovian
   - I. Queuing Networks (Product Form)
   - II. Approximate Solutions
     - Iterative
     - Decomposition
     - Diffusion
     - Ad Hoc
3. Simulation
   - A. Queuing Network
   - B. Hierarchical/Hybrid Approximations
   - C. Detailed Simulation
     - Distribution Driven
     - Trace Driven
4. Prototype/Synthetic Workload
5. Empirical
   - A. Linear Statistical
   - B. System Profiles
   - C. Analysis of Response
   - D. Resource Consumption

Linear constraint models are easy to understand and have relatively simple input requirements. They form rather loose bounds on the performance of multiple resource systems and are most useful in feasibility studies. During empirical studies, linear constraints, including conservation of space-time products, are useful consistency checks. Since their assumptions are quite credible, they forcefully state that many conditions are impossible. In essence, the maximum utilization of each resource puts a constraint on the throughputs (and therefore responses)of each workload type. Operational analysis is a variation based on mathematical identities between carefully defined performance variables. Many of its relationships were presented in [12][1]. It did not offer new advantages as a predictive model because, although its mathematical identities are useful for checking internal consistency, additional assumptions, such as homogeneity [14], are necessary for predictive purposes. The accuracy of these predictions can be expected to fall in the class of queuing networks with product form solutions [2]. In practical applications, stochastic assumptions cannot be tested satisfactorily and homogeneity can usually be ruled out. Therefore, we must rely on empirical determinations of their applicability.

Simple queuing models with external arrivals should be well understood for the insights they provide. However, they should be applied to real modeling situations only with extreme care. For a small user population or high resource utilizations, they can be quite misleading [6] [32]. The accuracy of commonly used closed queuing networks and the diffusion approximation using product form must also be questioned. Although these models are appealing in many ways, when applied to real operating systems they commonly force one to gloss over important details that can affect performance. Under some modeling objectives (such as high level design) queuing networks have lent direction. However, recent work [23] demonstrates that in the general case, more than two moments of the service time distributions are necessary to specify the behavior of closed queuing networks. Further, important effects such as storage allocation must often be ignored. Both accuracy and precision of this class of models needs to be quantified when used in real evaluation situations. As this body of information grows, it may then become possible to study generalities, but the current literature is still shallow.

An analytic queuing network model may best be used in conjunction with a queuing network simulator. The analytic model is used to validate the simulator at many important points. The simulator checks for mistakes and tests the sensitivity of analytic results to deviations from the ideal conditions assumed. The analytic model generally yields results at less processing cost and allows families of designs to be studied parametrically. Also, there are several mathematical constructs available in analytic modeling which cannot be simulated without difficulty. One example is the processor-sharing queuing discipline. When approached in a simulation by decreasing the round-robin scheduling quantum, the simulation run time becomes excessive. On the other hand, the queuing network simulator can be used to study service time distributions with irrational Laplace transforms, storage allocation, blocking, and other effects which are mathematically intractable. Also, the internal statistics potentially available from simulations are far superior.

Ad hoc stochastic models have not yet made their mark on practical CPE applications. One possible reason is the scarcity of highly trained personnel. Another possible reason is that the accuracy properties of these models when applied to non-ideal situations, do not justify attention to much-abstracted detail.

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

Prototypes and detailed empirical studies [13] require that well-developed hardware and software be available and operational. Both are ruled out from the current project because our objective is to provide a vehicle that can predict system or network performance under unavailable loads, mixes, and configurations.

The requirements detailed in Section 2.1 and the present considerations led to the choice of a trace-driven simulator as the prediction vehicle. The sensitivity to differences in workload implies that trace measurements would have to be taken. These data could be reduced or could drive a simulator directly. A distribution-driven simulator would require more trace processing and would discard much measured detail. The requirement to accurately model demand paging discouraged me from using product form queuing networks. Intuitively, I felt that the use of mean working sets in an ad hoc analytic model would not satisfy the accuracy criterion. I ruled out SIMNET, an inhouse queuing network simulator, for the same reason. Therefore, the method of choice was a traçe-driven simulator. If such a model could satisfy the accuracy criterion, it could then be studied with the intent of short-cutting part of the simulation overhead with analytic and distribution-driven techniques.

Objections to simulation based on high processing cost are losing their validity as hardware costs continue to drop. As we shall see, the actual simulator overhead turned out to be quite reasonable, when a trace optimization procedure was applied. The operating characteristics of the SPP during the validation are summarized in the appendix (section 5.2).

2.3 Operational Design

Figure 1 displays an overview of the SPP system. The heart of the system is an event-driven simulator. Inputs to the simulator are illustrated as being divided into three distinct categories. First is the configuration parameters. These include such information as the number of disks and terminals attached to the computer. The second set of inputs is the actual characterizations of the user jobstreams, such as CPU demand and virtual memory requirements. This information is derived from traces of the jobstream. The raw trace data is abstracted, which results in a reduction of the resolution in the timing of some jobstream events. Some of the details of this abstraction are presented in the appendix. Finally, the third set of inputs is the system parameters. These quantify the timing of system functions and hardware characteristics. They primarily consist of time elements for various operating system overheads which have been measured previously. The simulator consumes resources for system overhead at rates dependent on these elements and on the state of the simulation.

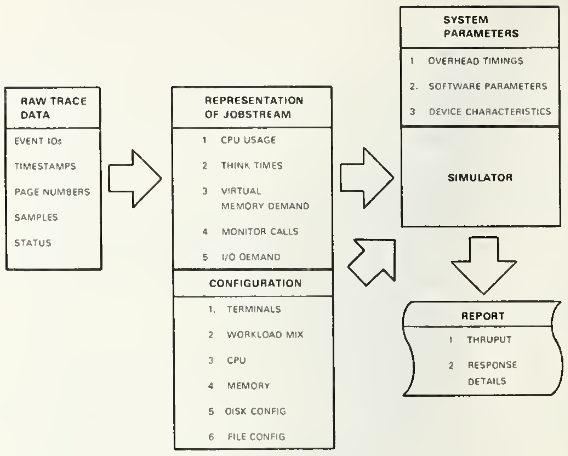


Figure 1 SPP Input/Output

To operate the SPP, the user supplies a representation of his workload and a description of how it is to be "executed" (e.g., number of terminals, etc.) to the simulator. The workload representation is automatically generated by a special trace program which has been run prior to the simulation. Optionally, the resulting trace can be condensed by an optimizing program. The simulator then uses one or more traces to estimate the new performance timings under a hypothetical environment. The validation of the simulator, as discussed below, will allow the user to estimate the accuracy of the performance predictions. Finally, the performance analyst can look for small variations in inputs and assumptions which cause wide swings in the performance predictions. If found, they may warrant more measurement expense, or less confidence in the predictions.

2.4 Simulator Design Consideration

The starting point for the design of the simulator was a consideration of resource demands. It was important to identify which demands determined the throughput and mean response time of the workload execution. In making this choice, I considered existing computer system models and the intuition of analysts familiar with the system. I chose CPU bursts, page references and modifications, monitor calls, I/O actvity, and terminal input to be elementary events. I anticipated that this level of detail might be simulated with a satisfactory simulation run time.

By far, the most important force acting on the design of the simulator was the logic of the operating system. Many of the high level scheduling decisions of the operating system have analogs in the simulator. Models of many software parameters available in the real system are included. And lastly, the performance characteristics of I/O devices, CPU and real memory have led to related parameters in the simulator. A first level of detail describing the implementation of the SPP is presented in the appendix.

2.5 Project Activities

The design choices made for the model and the goals of the project imposed many requirements on the more detailed activities of the project. The simulator itself was implemented in SIMULA running under TOPS-20. The design of the simulator required that a tracing program be developed which could deliver a description of any desired workload. It soon became clear that it would be convenient to develop an assembler language interface for reading the trace files by direct access. Later, additional function was required of this interface such as abstraction of the trace to a coarser level of detail and removal of undesirable distortions.

In addition to these development activities, several major measurement and evaluation activities were required. The most straightforward of these was an experiment to measure the validation statistics. These measurements were taken on a system driven by a remote terminal emulator. However, there were several other major measurement requirements. To deliver these data, we adapted a hardware monitor for attachment to the initial target CPU.

A validate-and-repair cycle was placed in the original project plan in anticipation of the initial failure of the simulator to satisfy the accuracy criterion. If and when these discrepancies appeared between the simulations and the RTE-driven real system measurements, both simulator and system would be instrumented, measured, and analyzed for the failure. This cycle had to be minimized because of time limitations. The last step was to establish the accuracy of the model and to measure meaningful confidence bounds about the predictions of the SPP.

3. Measurement and Validation

3.1 Model Parameterization

Many system parameters integrated into the simulator were related to frequently used functions of the monitor. We associated these parameters with well-defined system operations, so that the simulator could invoke them at the appropriate times. We sought small variances in the actual run times of these functions, so that implementing the simulator to charge a constant overhead for each of these functions would not cause significant error in its predictions. In those cases where this was not possible, the durations were divided into more elemental components which were selected by the state of the simulation. Short, but frequently executed operations must be included, if they have significant impacts on system operation. The CPU overheads selected for representation as simulator parameters were those associated with:

1. Scheduling processes
2. Page faults
3. Memory garbage collections
4. Periodic system operations
5. I/O initiation
6. I/O interrupts

Other parameters include hardware characteristics such as the mean disk seek time.

To measure the parameter values, we adapted a fairly sophisticated hardware monitor. This hardware unit was able to monitor the virtual addresses associated with instruction fetches. By programming the hardware monitor to determine the elapsed time between various instruction fetch events, we generated run-time histograms for individual routines in the operating system. Examination of these time distributions provided the mean run time and its variance for each routine of interest. The hardware monitor also could detect the execution of specific microinstructions and map time distributions of I/O events. We used it to measure data for parameterization of the simulator, performance enhancement, debugging the trace programs, and correction of artifact in the traces.

3.2 Trace Artifact Correction

I began the artifact removal analysis by qualitatively laying out CPU time and other events represented in the trace beside the desired simulated time. The objective was to use traces from the distorted trace environment as a source for timing values which drive the simulator. An additional goal was that the algorithm chosen to correct the trace should be able to withstand minor changes to the trace program. The requirement of robustness to trace program development implied that the artifact removal algorithm had to have self-correction aspects. Three available approaches toward this end are:

1. To timestamp entry and exit from the trace routines.
2. To use sampling techniques.
3. To do empirical regression studies of overhead on trace subroutine execution rates [13].

Load-dependent system overhead imposes an additional complication. Measurements commonly show that system overhead CPU consumption increases with load. A trace under a single environment could not be expected to quantify these effects. Another form of load dependency appears as time-driven system processing. The simulator can be driven by multiple traces, but system processing that occurs at a rate linear with time must be simulated exactly once. Therefore, we chose to remove such activity from the traces. A motivation for removing monitor traces is that the sampling is known to be synchronized with the system's scheduling activity. This could have been fixed by "randomly" generating exponentially distributed intersample times. Instead, samples of scheduler and interrupt processing were removed and the sampler was presumed to be uncorrelated to the remaining user program processing.

In view of these considerations, the artifact correction algorithm removes all samples except those representing user processing and user-initiated monitor call processing. An empirically determined factor was derived from time distributions measured by the hardware monitor. This constant factor quantifies the CPU time represented by each sample. Load-dependent effects, interrupt processing, and scheduling are therefore not represented in the derived traces. All of these last effects are initiated by the simulator based on the values of previously measured parameters.

3.3 Validation Strategy

Before we executed the formal validation plan, we entered into an initial calibration phase. By this time we had measured all the parameters of the model, and were ready to investigate the predictons of the SPP. Two points from the validation experiment were used to make fine-tuning adjustments. This calibration is described in the next section. We then executed the validation plan using new traces and seeds for the random number generations.

The System Performance Predictor was too complex for us to leap immediately into a complete validation. To minimize complicated effects, I chose to increase the complexity of the validations gradually. In order to begin with an environment of minimal activity, the simulation of CPU activity with no users active was used as the first validation point. Then we increased the complexity to vacuous transactions with minimal processor demands. Neither of these points, however, appear in the formal validation plan. We then proceeded to the case where a single resource is fully utilized by a known synthetic workload. This last environment is simpler than one can expect in a live situation, but offers great advantage for studying the ability of the model to capture the essential behavior of the system. It was then necessary to proceed to more complex situations after examining the behavior of the model in conditions of one-at-a-time resource saturation.

In a different dimension from the saturation of resources are the complexities introduced by increasing the level of multiprogramming. In addition to complex interprocess effects, increasing load tends to expose new system effects. Therefore, in accord with the strategy of starting the validation in simple environments, we examined single terminal runs first. These initial single user validations were restricted to a synthetic transaction (CPUJOB) composed of 5.1 seconds of user CPU time. I then proceeded to examine the ability of the SPP to predict the behavior of many interactive users of the CPUJOB transaction. This introduced the complexities of unpredicted transaction arrivals and process scheduling. In addition to modeling system effects, I had to generate a think time distribution (uniform from 0 to 20 seconds) to match that of the RTE. This introduced an extra source of variance over and above the use of the SPP with traced think time. The demand paging aspects of the SPP were tested using a more complex synthetic job (MEMJOB). It repeatedly touched 106 user pages every 100 msec of user CPU time for a total of 1.5 seconds. We felt that this job was somewhat more active than most real workloads with respect to its virtual memory demand. As I did with the CPU intensive workload (CPUJOB), I tested the ability of the SPP to predict the performance of a variety of loads running MEMJOB transactions interactively. To continue the progression to more realistic workloads, I included a real disk-to-tape utility program (DMPRJOB) in the validation plan.

I measured the effects of loading on the MEMJOB workload (across three different real memory sizes) by designing and executing a 3 × 4 full factorial experiment. The RTE-driven (REAL) runs had three replications and allowed the generated think time distributions to be randomized across all runs. The SPP runs (SIMULATED) were blocked in order to remove the variances caused by tracing and think time generation from each of the three blocks. These variances were randomized across blocks.

In both REAL runs and SIMULATED runs, measurements were not collected until the choice of the starting state appeared to have no effect. The RTE data were collected from 5 minutes until 25 minutes after the start of the run. The simulation data were collected from 2 until 7 simulated minutes after the beginning. Within each run a mean response time is derived from a least 25 individual responses of the same transaction type. Each mean response time is then considered to be an independent observation.

Before I began the formal validation test, I calibrated the SPP by making a few fine-tuning adjustments.

3.4 Calibration Procedure

I found that additional calibration adjustments had to be made over and above the measurement of parameters. I adjusted two unmeasured parameters at each of two validation environments in order to correct the simulations.

Both elapsed time and CPU time were corrected for the real disk-to-tape utility job (DMPRJOB). Elapsed time was corrected by adjusting a multiplicative factor for estimating overlapped I/O time. This factor is multiplied by a sample count derived from the trace and removed from the estimate of user I/O time. The CPU time of the utility program was then corrected by adding a CPU demand for each I/O request. These last two adjustments were then iterated until both correct elapsed time and correct CPU demand were simulated. This method was not straightforward, but I used it beause it did not affect SPP accuracy elsewhere.

Two additional calibration adjustments were made for the simulation of overcommitted memory. When the job which caused high page fault activity (MEMJOB) was traced and simulated, the simulated CPU demand was too high. As a first order correction, I subtracted that amount of CPU time per traced page fault which corrected the CPU demand. Presumably, the method of artifically inducing page faults in order to identify referenced pages (see the appendix) was causing extra CPU samples to appear in the trace.

The last calibration knob that I adjusted, prior to the formal validation, affected the number of dynamic simulated page frames. I found that I could bring the simulated response times closer into line with the RTE runs by removing 80 simulated page frames. Further investigation revealed that approximately 80 pages were actually needed by the real system for I/O buffers and system tables. Thus, a previously ignored system parameter was serendipitously estimated by adjusting a calibration variable. This discovery lent additional credibility to the SPP.

3.5 Validation Results

The results of the validation are presented in Table 2 and graphed in Figures 2 and 3.

Table 2 Validation Results

					MEAN RESPONSE ± 95% C.I.	
WORKLOAD	USERS	MAIN STORAGE	CPU	DISKS	REAL	SIMULATED
CPUJOB	1	512K	KS	1	5.1 ± .1	5.3 ± 0
	3	512K	KS	1	10.0 ± .4	13.5 ± 1.4
	6	512K	KS	1	22.1 ± 2.0	25.9 ± 2.9
	8	512K	KS	1	35.4 ± 1.3	35.9 ± 2.2
DMPRJOB	1	512K	KS	1	53 ± 4.3	51.3 ± .5
MEMJOB	1	512K	KS	1	1.5 ± .1	1.5 ± .1
	3	512K	KS	1	2.2 ± .2	3.6 ± .4
	6	512K	KS	1	3.3 ± .1	5.9 ± 1.4
	8	512K	KS	1	13.4 ± 1.0	12.9 ± 4.3
	1	384K	KS	1	1.5 ± .1	1.5 ± .1
	3	384K	KS	1	2.1 ± .2	3.6 ± .4
	6	384K	KS	1	12.3 ± 1.0	11.7 ± .9
	8	384K	KS	1	31.8 ± 5.7	20.0 ± .9
	1	256K	KS	1	1.5 ± 0	1.5 ± .1
	3	256K	KS	1	6.7 ± .9	3.9 ± .6
	6	256K	KS	1	31.6 ± 15.4	30.2 ± 17.8
	8	256K	KS	1	94.5 ± 20.7	104.1 ± 21.0

The results are presented as the mean response time (in seconds) plus or minus a 95% confidence interval (also in seconds) for both real executions and simulations. For the CPU intensive workload (CPUJOB) a factorial experiment measured the SPP's ability to predict load effects. For the high paging workload (MEMJOB), a factorial experiment measured the SPP's validity in predicting load effects, main storage effects, and their interactions. For simplicity, only the interval estimates are presented here. The data include a validation point running a real disk-to-tape utility job (DMPRJOB).

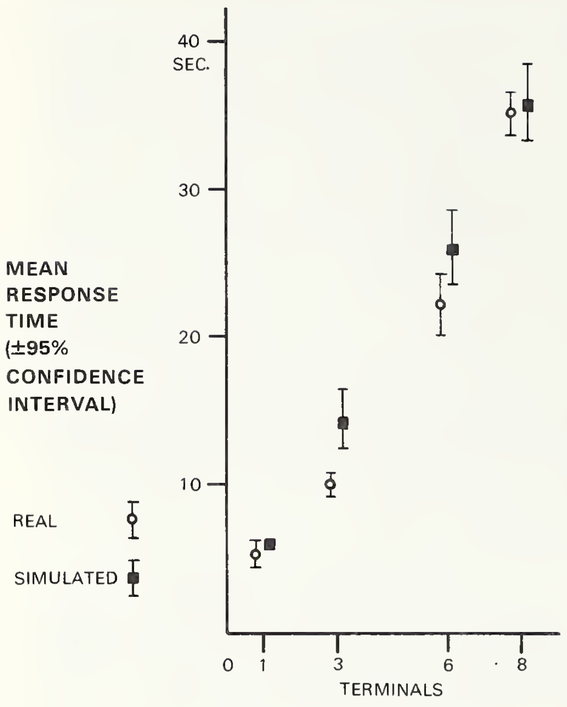


Figure 2 CPUJOB Validation

The results show that the predictions of the SPP were generally good.

The mean overall accuracy was that the SPP's predictions were high by 10%. As shown in Table 2, a few errors in uncalibrated environments were larger, but most predictions were not significantly different from REAL mean response times. In general, the SPP captured the quantitative effects of each of the dependent variables in that saturation conditions seem to be simulated adequately. Examination of Table 2 under midrange load conditions (3 and 6 terminals), however, reveals that the SPP predictably delivered slightly high response time estimates under conditions of sufficient main storage. This "midrange bulge" reflects the fact that, as of this writing, fine-tuning adjustments to simulations in the medium load environments have not been applied. I began to investigate possible corrections to this in the sensitivity analysis.

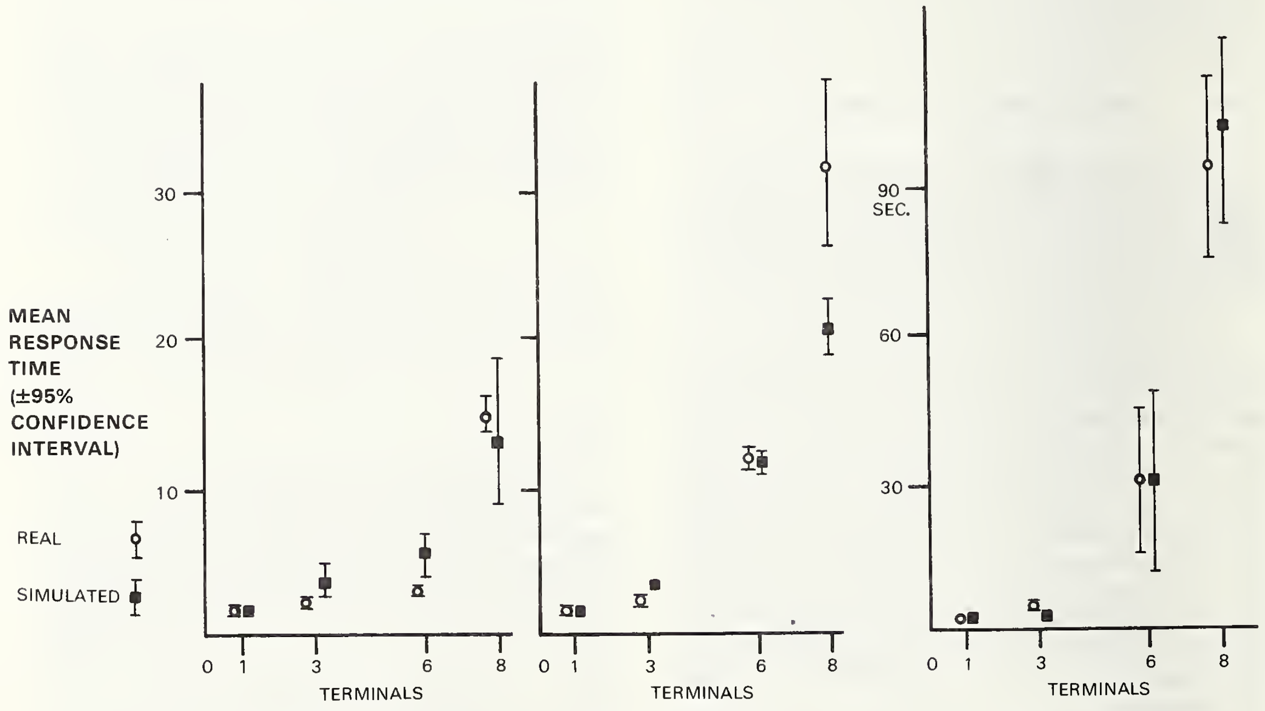


Figure 3A 512K Figure 3B 384K Figure 3C 256K

Figure 3 MEMJOB Validation

3.6 Sensitivity Analysis

In order to determine which adjustments of the SPP were critical, I designed and executed a screening test for sensitivity. Its objective was to identify parameter groups which have strong effects on the predictions of the SPP. Also, I hope to use the analysis to direct future measurement and fine tuning efforts.

I began by separating the parameters into three groups:

1. Trace factors
2. Page frame calibration
3. System overhead parameters

I chose an execution environment of six users of the MEMJOB workload in 512K words of memory. This is an environment wich expresses the "midrange bulge" described above, which is one of the SPP's current weaknesses.

The design of the experiment is a 2X2X2 factorial with three blocks. Each of the three parameter groups has two levels: unchanged from the validation experiment, and modified by changing all group members (in unison) 10% in the direction of increasing resource demand. The variance across traces and think time generations was blocked out. Within each block, I ran simulations using all combinations of parameter group settings.

Table 3 Sensitivity Test Data

1. TRACE FACTORS	2. PAGE FRAME CALIBRATION	3. OVERHEAD FACTORS	MEAN RESPONSE TIMES (SECONDS)		
0	0	0	6.4	5.4	5.8
0	0	1	5.8	5.5	5.1
0	1	0	6.4	5.4	5.8
0	1	1	5.8	5.5	5.1
1	0	0	7.8	7.3	6.2
1	0	1	7.2	6.8	5.9
1	1	0	7.8	7.3	6.2
1	1	1	7.2	6.8	5.9

0 = SAME AS VALIDATION RUN
1 = ALL GROUP MEMBERS CHANGED 10%

The results are tabulated in Table 3. An Analysis of Variance is presented in Table 4. The data indicate that both trace factors and system overhead parameters exhibited strong main effects

Table 4 Sensitivity Test ANOVA

	SUM OF SQUARES	DEGREES OF FREEDOM	MEAN SQUARE	F
1. TRACE FACTORS	8.640	1	8.640	78.209
2. PAGE FRAME CALIBRATION	0	1	0	0
3. SYSTEM OVERHEADS	1.127	1	1.127	10.198
4. INTERACTION (1–2)	0	1	0	0
5. INTERACTION (1–3)	0.007	1	0.007	.060
6. INTERACTION (2–3)	0	1	0	0
7. INTERACTION (1–2–3)	0	1	0	0
8. BLOCKS	4.413	2	2.207	19.975
9. ERROR	1.547	14	0.110	
10. TOTAL	15.733	23		

but that the 10% page frame calibration adjustment had no effect. This is consistent with the shape of the response time curves in Figure 3. The simulated effects of constricting main storage tended to take hold later but more suddenly than in the REAL runs. The simulator did not feel significant main storage effects with six users of 512K words.

The sensitivity test did not reveal any significant interactions among parameter groups. A strong effect, however, was felt across blocks of the experiment. Most of this variance arises from workload tracing. Had the experiment not been blocked, the effects of the system overhead parameters would have been obscured by this variance.

The direction for future fine-tuning efforts begins to take shape from the sensitivity test. The strongest effects were due to the trace factors—which describe the workload to the simulator and had effects at all load levels. The system overhead parameters had a smaller overall effect but are strongly implicated in the midrange bulge because, intuitively, they should be most strongly expressed when simulated workloads interact. From the sensitivity test we see the need to remove variance from tracing and from changes in group 1 parameters when studying the effects of individual system overhead parameters. Interactions, however, do not yet appear to be a problem under these conditions.

The large number of control variables in the SPP provides both opportunity and challenge. There is a great potential for fine-tuning the SPP in new environments. However, some of the cabibrations will interact with and/or invalidate previous fine-tuning efforts. If further validation measurements uncover inaccuracies in the SPP, it may become desirable to initiate a more systematic calibration procedure [17]. However, I am hopeful that few additional adjustments will be necessary.

4.0 Conclusions And Remarks

At points where I fine-tuned the SPP, the performance predictions of mean response time were not significantly different from benchmark measurements. When all mean response times used in the validation are included, the mean SPP prediction error was about 10%.

Several calibration adjustments were made prior to the formal validation. The last one was an adjustment to the amount of dynamic memory which turned out to have an analog in the real system. Before these calibrations, the SPP mean response time predictons had been accurate to within roughly 30% (80% worst-case).

In view of this experience I cannot yet claim general validity of the SPP. I hope to expand the domain of validity by testing with unfriendly (never seen before) workloads and environments. I hope to develop the op-code sampling technique (see appendix) in cross-CPU projections and to investigate the effect of mixed workloads and new DASD configurations.

The use of the SPP in capacity planning situations is still experimental. Considering its present state of validity, it appears to have the potential of satisfying the original project objectives. With additional development and validation, I am sure that the range of environments which enjoy accurate performance predictions can be enlarged. But for some applications which require less accuracy or only relative results, the SPP may be useful in its present state. This would have to be confirmed by experience.

Workload characterization is an important area in the general capacity management problem. The tracing function of the SPP can automatically generate and optimize libraries of workloads which may even be modified artifically. It therefore may provide a vehicle for future studies in various aspects of the workload characterization problem. Perhaps the most significant result of the present work is that the workload description used was sufficient to predict performance in a fairly diverse set of demand paging environments. Within this domain of validity, the mean response time varied by nearly two orders of magnitude.

Capacity management presents a need for a sufficient workload characterization feature to be thoroughly integrated into the design of the operating system. Since nearly all performance problems are confounded by the workload characterization problem, we cannot expect to systematize CPE methodologies until such fundamental descriptors are easily available.

5. Appendix

5.1 SPP Implementation Details

This section begins to explore a first level of detail of the implementation of the SPP. It is an attempt to present a description of those mechanisms which account for the behaviour of the SPP.

5.1.1 Workload Characterization

The initial workload characterization was chosen by considering successful models in existence and by estimating which aspects of the workloads must strongly influence the performance predictions. For example, performance is sensitive to CPU demands, paging activity and I/O delays. Therefore, the demands of workloads on these (and other) resources must be quantified by the workload characterization.

It is useful to understand the overall operation of the trace program in order to fully appreciate the workload characterization. The trace program inserts breakpoints into the system in order to record the important events as the workload interacts with the system. The program was designed to produce the following time-stamped records:

1. User CPU time.
2. Entry to and exit from monitor calls.
3. Page faults.
4. Lists of modified pages.
5. Satisfaction of terminal input wait.
   These records contain much additional detail.

The traces deliver several alternate sources of the CPU time demanded by the workload. A sample taken every eight milliseconds of real time also contains an elapsed timestamp and a process (virtual) CPU timestamp. The CPU demand could be calculated from either timestamp or by using sampling techniques. Several methods were tried during the validation runs in order to maximize accuracy of the simulator. The method chosen was to count samples of user or monitor call time and to apply an empirically determined CPU time factor to associate an average run time with each sample.

The objectives of the project required further characterization of the CPU demand. Measurements taken at DIGITAL had demonstrated the sensitivity of relative CPU speed (in this family of CPUs) to the instruction mix of the user workload. This variation could be as large as a factor of 3 or more and was clearly unacceptable when predicting the workload's performance under a new CPU. Therefore, op-code information is sampled by the trace routines to permit an accurate estimate of processor demands when executed on different CPUs. Unfortunately, the timer interrupt sampling routine does not yield an unbiased random sample over time. Since timer interrupts occur after instructions are completed, PC samples of long (important) instructions will point to the following instruction. To approximate a random sample more closely, the saved PC is backed up to the previous instruction, whose op-code is traced. This tends to sample those important instructions of long duration (floating point, decimal) but also incorrectly samples unexecuted jumped instructions. Similarly, those instructions emulated by software must also be detected and weighted. Although op-code samples are traced, presently they are not used by the simulator.

Paging behavior is highly dependent on the execution environment. The descriptor of virtual memory demand must allow the simulator to generate increasing CPU consumption and I/O activity per work unit as simulated real memory becomes overcommitted. This might be possible by measuring the resource consumption due to paging activity under the trace environment and then using the simulator to map these effects to the simulated environment. But this would put extremely heavy requirements on the simulator, because then it would have to select its actions in a manner dependent on the environment in which the workload was traced. The problem would be reduced by performing traces in a strictly controlled environment. But even here, one could expect two workloads yielding identical paging characteristics in a simple environment to behave very differently in a loaded environment. For these reasons a more intrinsic descriptor of virtual memory demand was sought.

The characterization chosen was to identify those pages actually referenced in small time windows. The period was chosen experimentally to trade-off predictive accuracy with trace overhead and artifact. At end of each interval, the trace program nullifies all the user virtual address translations by manipulating system tables. Then, each virtual page referenced in the following interval will reveal itself by a page fault, which is traced. In addition, modified pages are identified because they generate pageout activity during simulations. In this manner, artificially induced page fault events are used to drive simulated virtual memory references.

Monitor paging is part of the workload characterization. The detection of monitor call paging became a function of the simulator via a bitmap lookup. A list of monitor calls is built during the workload trace and associated with a block of CPU time. The simulator performs a simulated virtual memory reference for each monitor call whose bit is set in the bitmap. The virtual addresses to be referenced are kept in a table in the simulator. In this way, the indirect system virtual memory demand due to the execution of the workload is part of the workload characterization.

Several other events are used to describe the workload. User I/O activities other than paging are detected and quantified by the duration of the appropriate system call from start and end timestamps available in the trace. Finally, user think times are detected and quantified by terminal input timestamps.

5.1.2 Simulation Overview

The simulation separates the non-load-dependent factors that represented the individual jobstreams from the load-dependent operations associated with the conditions of execution. The former are recorded in the traces, the latter are applied actively by the simulator. In addition, the simulator must model the interactions of the workload and load-dependent system operations. This, in turn, results in the requirement for parameterization of the simulator to quantify these load-dependent operations.

5.1.2.1 Simulator Logic

The simulation model of the system was written in SIMULA-67. The following functions of the TOPS-20 operating system were incorporated into the model. The operating system is a demand paging system designed primarily for timesharing. A set of transactions called the balance set is maintained in real memory. Transactions circulate through four prioritized run queues, starting at the bottom of the highest priority queue. When a transaction on the three highest priority run queues has received a fixed amount of simulated CPU time the transaction is demoted to the bottom of the next lower priority run queue. The balance set is identified on the next entry into the scheduler following an event which potentially can cause a task switch. These include a page fault, a user specified I/O request, the expiration of a run queue CPU time quantum, or the completion of an I/O event. The scheduler is entered periodically and when a process blocks. The transactions on the run queues are selected for the balance set in order of deceasing priority as long as the combined "pages needed" count does not exceed the number of available page frames. Workload execution then proceeds by round robin scheduling of transactions in the balance set. Some scheduling control algorithms in the operating system were not modeled in the simulation.

The simulated paging activity is initiated from a list of referenced pages and a list of modified pages kept with each user CPU request. All pages required for a simulated CPU request must be in simulated real memory before the simulated CPU request is granted. There is a separate, globally sharable address space for the system's command processor. All referenced pages must be paged in if they do not already exist in simulated real memory. When a demanded page is not present, a page fault is simulated and a real frame is allocated by taking it from a free frame queue. The page frame is then moved to the in-use page frame queue. The process which generated the page fault is then blocked until the page-in is complete. If the free page frame queue has become depleted beyond some threshold count (a model parameter), a global garbage collection (GGC) is scheduled. This GGC routine steals all page frames belonging to transactions out of the balance set. GGC moves unchanged pages directly to the free page frame queue and moves changed pages to the page-out queue where they remain until each page out I/O is posted complete. At those times the page frames are moved to the free queue. Periodically, a local garbage collection routine is invoked for a single transaction in order to clear out old pages even though their parent process is still active. A parameter is set to specify the age of the oldest page frames to survive the purge. The age of each page frame to be examined by the local garbage collector is stamped at completion of the most recent page-in. When its page frame is stolen, a modified page is written to a slot on a paging device. The simulator pre-allocates these page slots contiguously in the center of each simulated disk. Each terminal is assigned a distinct disk cylinder for page I/O.

Disk I/O requests are chosen from a queue associated with each controller. The seek requests are chosen so that the simulated disk arm sweeps the disk from low address to high. There is a fairness count which limits the number of data transfers at a cylinder before the sweep continues. These algorithms are patterned after those in the operating system. As is true in the real I/O subsystem, data transfers are serialized by each controller. The simulated seek times are calculated as a slanted step function of cylinders traversed.

User I/O to non-DASD devices is detected from monitor call elapsed time derived from the trace. The portion of this time which is overlapped by effective user processing is estimated and removed from the total elapsed time. The remainder is passed to the simulator as a single-user I/O request with no contention. The simulator, in addition, consumes an amount of simulated CPU time (a parameter) for each I/O request.

There are many ways to control the simulated environment. The user of the simulator specifies how many terminals or batch streams each trace file will supply. This determines the simulated workload mix. Additional simulation run control variables set the durations of the run pre-measurement start-up period. The user of the simulator may also specify the memory size, the I/O configuration, changes to the operating system parameters, and CPU power information. The execution of the workload may be further controlled by using the think times measured in the traces, or by forcing them to be sampled from one of several distributions.

The simulator is quite measurable. One great advantage in studying simulations is that the measurement artifact can be negligible for software-oriented statistics. At present, there are several traces which deliver data describing the internal workings of SPP simulations. The state of real memory, disk I/O, disk I/O queues, and run queues can all be observed by turning on the appropriate trace(s). SIMULA provides additional variable tracing facilities. And finally, a detailed internal performance report is generated by the simulator along with the external performance predictions. These internal data are not intended to be exact analogs of system data. They are intended to be used for diagnostic purposes.

5.1.3 Trace Program

The tracing program is based on the use of a special monitor call. This system function allows the program to insert breakpoints into the monitor. These breakpoint routines construct trace records in a buffer which is written to a disk or tape file by code operating in a user address space.

One breakpoint gets control on entry to a periodic timer routine and is used to sample several data items. In addition to collecting information regarding process status, the timer breakpoint samples both the program counter and the op-code of the interrupted process. If the process has been interrupted while in a user address space, the breakpoint traces down the paging data structure and makes a real memory access to retrieve the op-code. The present version of the trace backs up one memory call before retrieving the instruction.

In order to dissociate the virtual memory requirements from system activity, paging statistics taken by the operating system are not used. The monitor maintains an entry for each frame of real memory in the Core Status Table (CST). Entries are invalidated periodically by the trace program. In addition to the CST, a hardware table of valid translations is invalidated so that future virtual memory references must execute full address translations. The invalid entries inserted into the CST cause the first reference to each virtual page to reveal itself by a page fault. This event is then recorded by another breakpoint routine. A third breakpoint corrects the table entry. When the CST is modified by the trace routine, a trace record is generated which identifies each modified page belonging to the traced job. The consequence of monitoring memory page references in this manner is that it is possible to ascertain which pages are referenced during intervals between hardware page table invalidations. This technique considerably reduces the trace overhead and volume of input to the simulator, while still providing enough information to evaluate the paging behavior of the system under various loads. It may introduce inaccuracies because closely spaced references to the same virtual page are not detected or recorded. However, closely spaced references to the same page should not strongly influence actual paging behavior.

Three other breakpoints provide timestamped traces on the microscopic level. A pair of breakpoints detects entry to and exit from any user-called monitor routine. The third breakpoint detects terminal input and thus demarcates the beginning of interactive transactions.

5.1.4 Trace Abstraction

Microscopic traces may be abstracted prior to or during a simulation. During this process, the CPU demands occuring within a page detection time interval are grouped together. If a think time has occured recently, its duration is inserted into the record. Then the IDs of pages referenced in the interval are associated with the CPU burst in two lists (user and command processor). Similarly, a list of monitor calls is built and attached to the abstracted trace record. Non-DASD I/O requests are detected from the durations of these monitor calls. The condensed trace record is completed by adding a list of IDs of modified pages. Peforming this abstraction step prior to the simulation greatly reduces simulation overhead. This "optimization" is quantified in the next section.

5.2 Operating Characteristics

The run times of the various components of the SPP are important because in the past many simulations have suffered intolerably high processing requirements. As you might expect, these characteristics of the SPP turned out to be very workload-dependent. The act of tracing generally caused the trace workload to take one to two times longer than normal in a single jobstream environment. When such a trace is used to drive the simulator directly, the CPU time used by the simulation was up to 10 times the simulated time (8 terminals, MEMJOB). However, if the trace is abstracted by the "optimizing" program before the simulation, this ratio dropped from 10:1 to 5:1. These were the worst-case numbers encountered in the validation.

The optimizing program had an even stronger effect on the CPU intensive workload. Some simulations of CPUJOB actually ran faster than real-time and were never worse than 2:1 (8 terminals) when the traces were optimized.

The cost of running the optimizing program was roughly 1-2 seconds elapsed time per 1 second of traced time. All the numbers given above refer to traces, optimizations, and simulations run on a fairly low cost machine, the DECSYSTEM-2020. Some additional improvement can be expected because only a few performance optimization techniques available have been applied to the SPP itself.

I would like to thank Joel Emer, also at DIGITAL, for his contribution to this project. Thanks also to Rollins Turner, Richard Glantz, Jim Hughes, Dan Murphy, Dennis Phillips, Terry Potter, Andy Vesper, and Alice Cox.

References and Bibliography

1. Bard Y., An analytic model of the VM/370 System, **IBM Journal of Research and Development.**, 22, 1978.
2. Baskett, F., Chandy, K., Muntz, R. and Palacios, F., "Open, Closed, and Mixed Networks with Different Classes of Customers, **Journal of the ACM**, 22, April 1975, pp 248-60.
3. Boyse, J.W. and Warn, A straight-forward Model for Computer Performance Prediction, **ACM Computing Surveys**, 7, 2, 1975.
4. Browne, J., Chandy, K.M., Brown, R.M., Keller, T.W., Towsley, D.F. and Dissly C.W., Hierarchical Techniques For The Development of Realistic Model of Complex Computer Systems, **Proc IEEE**, 63, 6, 1975, pp 966-975.
5. Brown, R.M., Browne, J.C. and Chandy, K.M., Memory Management and Response Time, **Communications of the ACM**, 20, 3, March 1977, pp 153-165.

6. Buzen J.P., Guidelines for the Use of Infinite Source Queuing Models in the Analysis of Computer System Performance, **Proc NCC**, 1974, pp 371-4.
7. Buzen, J.P., Fundamental Operational laws of Computer System Performance, **Acta Inf.**, 7, 2, 1976, pp 167-182.
8. Chandy, K.M., and Sauer, C.H., "Approximate Methods for Analyzing Queueing Network Models of Computer Systems". **ACM Computer Surveys**, 10, 3, September 1978, pp 281-317.
9. Chandy, K.M., Herzog, U., and Woo, L., "Parametric Analysis of Queueing Networks," **IBM Journal of Research and Development** 19, 1, January, 1975.
10. Chandy, K.M., Herzog, U., and Woo, L., "Approximate Analysis of General Queuing Network," **IBM Journal of Research and Development**, 19, 1, January, 1975.
11. Chiu, W.W., and Chow, W., "A Performance Model of MVS," **IBM Systems Journal**, 17, 4, 1978, pp 144-62.
12. Cox, S.W., "Interpretive Analysis of Computer System Performance," **Performance Evaluation Review (Proc Sigmetrics)**, 3, No. 4, 1974, pp 140-155.
13. Cox, S.W., "Experimental Methods in Computer Performance Evaluation," **Proc. Eurocomp Conf on Computer Performance Evaluation**, London, 1976, pp 255-275.
14. Denning, P., and Buzen, J., "The Operational Analysis of Queuing Networks Models," **ACM Computing Surveys**, 10, 3, 1978, pp.225-261.
15. Dunlavey, R., "Workload Management," **EDP Performance Review**, Vol 6, No. 5, 1978.
16. Ferrari, D., *Computer Systems Peformance Evaluation*, Prentiss-Hall, 1978.
17. Gomaa, H., "The Calibration and Validation of a Hybrid Simulation/Regression Model of a Batch Computer System," **Software Practice and Experience**, Vol. 8, 11-28, 1978.
18. Gomaa, H., "Regression Models for the Evaluation of Computer System Performance," **Proc. Eurcomp. Conf. on Computer Performance Evaluation**, London, 1976, pp. 69-99.
19. Kleinrock, L., *Queuing Systems I*, John Wiley, New York, 1975.
20. Kleinrock, L., *Queuing Systems II*, John Wiley, New York, 1976.
21. Kobayashi, H., "Application of the Diffusion Approximation to Queuing Networks, I Equilibrium Queue Distributions," **J.A.C.M.**, 21, 2, 1974, pp. 316-38.
22. Kraemer, W., "Performance Investigations with DOS/VS Based Operating System Model," **IBM Systems Journal**, 17, 4, 1978, pp. 409-443.
23. Lazowska, E.D., "Characterizing Service Time and Response Time Distribution in Queuing Network Models of Computer Systems," **U. of Toronto Technical Report CSRG-85**, October, 1977.
24. Lipsky, L., Church, J.D., "Applications of a Queuing Network Model for a Computer System," **ACM Computing Surveys**, 9, 3, 1977.
25. Mead, R.T., and Schwetman, H.D., "Job Scripts—A Workload Description Based on System Event Data," **Proc. NCC**, Vol. 47, pp. 457-464.
26. Mendenhall, W., *Introduction to Linear Models and the Design and Analysis of Experiments*, Duxbury Press, 1968.
27. Muntz, R., "Queuing Networks: A Critique of the State of the Art and Directions for the Future," **Computing Surveys**, 10, 3, 1978, pp. 353-9.
28. Reiser, M., "Interactive Modeling of Computer System," **IBM Systems Journal**, 15, 4,
29. Schatzoff, M., and Tillman, C., "Design of Experiments in Simulator Validation", **IBM Journal of Research and Development**, 19, No. 3, 1975, pp. 252-62.
30. Saur, C., "Simultaneous Resource Possession in Queuing Models of Computers," **Performance Evaluation Review**, Vol. 7, 1 & 2, pp. 41-52.
31. Teorey, T.J., "General Equations for Idealized CPU—I/O Overlap Comparisons," **Communciations of the ACM 21**, 6, June 1978, pp. 500-509.
32. Turner, R., "An Investigation of Several Mathematical Models of Queuing Systems," **Proc 1978 CPEUG Meeting**, pp. 103-12.

A FORTRAN Synthetic Program for Benchmarking

Patricia M. Fleming
Andrew C. Rucks

US Army Concepts Analysis Agency
Bethesda, Maryland 20014

Benchmarking is a generally accepted and essential element in the competitive procurement of computer systems. A benchmark workload consists of a set of application programs, synthetic programs, or a combination of these designed to be representative of expected system workload. A benchmark workload developed from application programs is unacceptable because it (1) is potentially biased in favor of the incumbent vendor, (2) may not be truly representative, and (3) may contain data that is subject to privacy and security restrictions. A benchmark workload constructed from synthetic programs does not suffer from these limitations. The basis for employing synthetic benchmarks is well established; however, previous synthetic programs have failed to provide a means to test system capacity through the execution of a reasonable variety of programming functions. The FORTRAN Synthetic overcomes this disadvantage of previous synthetics by providing a set of programming functions which test a wide range of system capabilities. The Synthetic is modular in structure to provide representativeness and control of instruction mixes; parameter driven to provide control of processing time; and provides a means of controlling memory usage. The structure of the FORTRAN Synthetic and the process of workload mapping is presented.

Key words: Benchmarking; synthetic program; workload mapping; performance evaluation.

1. Introduction

Benchmarking is an essential tool in the competitive computer procurement environment. It is essential in the sense that benchmarking is the only means for comparing the workload processing capacity of competing systems. While benchmarking is not a system selection and evaluation panacea, the results of a benchmark test are regarded by both the computer industry and procuring activities as requisite input to any selection decision.

A major problem in benchmarking is that of coalescing a set of programs that adequately and accurately portray the expected workload that the the system to be selected will process. Developing a microcosm of the subject system's workload is the objective of benchmark preparation. This objective is accomplished through the processes of workload analysis and workload mapping. Workload analysis is a two step process of defining the composition of future data processing work. In step one, historical workload data (e.g., CPU and I/O resource consumption, numbers of jobs, language processor usage, pages printed, etc.) are examined for trends. In the second phase, endogenous and exogenous variables (e.g., organizational objectives, customer requirements, technology, etc.) are studied to determine their impact on workload trends. With the future as firmly in hand as possible, the task of benchmark preparation (i.e., workload mapping) can commence. Ideally, a benchmark workload (program mix) is a linear transformation of the expected workload. This linear relationship is stated in the following equations.

$W = f(A)$, where W is a workload and A is the set of applications comprising this workload.

$W' = bf(A)$ where W' is the benchmark workload, b is the transformation factor, and A is defined above.

It should not be inferred from these equations that the development of a benchmark workload (W') is simplistic and objective. On the contrary, the development of a W' is complex and subjective. A W' may be composed of application programs, synthetic programs or a combination of these two types.[1] The last alternative is often selected because designers of a W' recognize the limitations of application benchmarks, but are uncomfortable with the process of mapping synthetic program parameters with workload analysis results.

2. Limitations of Application Benchmark Programs

The construction of a benchmark workload mix from application programs presents a conundrum. Application programs tend to be machine or system dependent in that they are most likely written in language dialects which are peculiar to currently installed systems (e.g., UNIVAC 1100 Series FORTRAN V, IBM FORTRAN IV (Level G), Control Data Corporation FORTRAN Extended Version 4, etc.). This dependence is natural because language dialects are constructed to take advantage of particular operating systems and architectures. Thus, a benchmark constructed from application programs would tend to be biased in favor of the incumbent vendor not only in terms of workload performance but also by causing nonincumbent vendors to incur disproportionate conversion costs. In addition to the language dialect problem, the variety of application programs in a system workload makes the task of selecting a set of application programs to form a benchmark workload a difficult one. Because of this program selection difficulty, it is doubtful that a truly representative set of application programs can be selected. A third limitation of application benchmarks is that the data required by applications in the benchmark may be subject to privacy and security restrictions. The "sanitizing" of these data is a time consuming activity which diverts resources from the much larger problem of achieving representativeness.

[1]An application program directly contributes to the processing of end work as opposed to computer systems programs, language processors, and other utility programs. A synthetic benchmark program is a parameterized, functional computer program designed to represent a particular class or function of application programs for benchmarking purposes only; and serves no other useful function [2].

3. Advantages of Synthetic Benchmark Programs

A synthetic benchmark program does not suffer from the several limitations of an application benchmark program. A good synthetic benchmark program possesses the following qualities which overcome the limitations discussed in the previous paragraph: (1) it consists solely of the constructs of a generally used standard high level language (e.g., ANSI FORTRAN or ANSI COBOL); (2) it exercises a typical set of programing language functions (e.g., integer arithmetic, decision making, direct access input/output, etc.); (3) it provides a set of user controllable parameters for synthesis of an application workload into a synthetic workload; and (4) it produces repeatable, predictable, and verifiable results.

The literature of benchmarking establishes the basis for employing synthetic programs and discusses several applications of synthetic benchmarks [3] [4] [5]. These applications are resource oriented since they are based on the synthetic program developed by Bucholz [1]. The Bucholz synthetic and its derivatives are designed to exercise the CPU and I/O resources but do not address the testing of a system's capacity to execute a reasonable variety of programing functions. Without doubt, the objective of benchmarking is to insure delivery of a system with adequate resource delivery capacity. However, the test of this ability should take place within an environment where the target system resource delivery capacity is demonstrated through the performance of a variety of programing functions. This paper proposes a FORTRAN Synthetic program for benchmarking which enables representation of almost any workload by providing the means to control three critical variables in a benchmark workload -- (1) CPU and I/O processing time, (2) main memory requirements and (3) instruction mix.

4. The FORTRAN Synthetic[2] and Its Operation

The FORTRAN Synthetic is modular in structure to provide representativeness and control of instruction mixes; parameter driven to provide control of CPU and I/O processing time; and provides control of memory space through local arrays contained in each module.

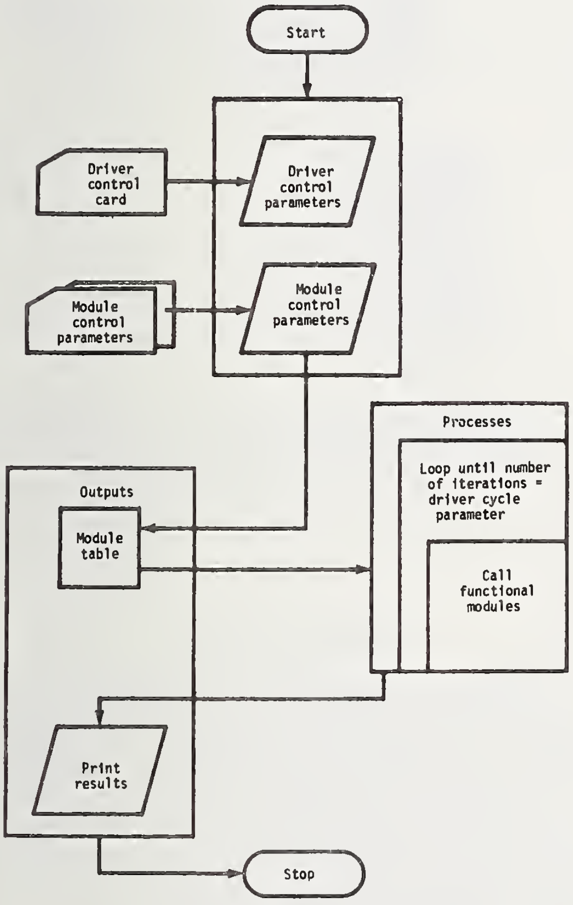

Start
Driver control card
Driver control parameters
Module control parameters
Module control parameters
Processes
Loop until number of iterations = driver cycle parameter
Call functional modules
Outputs
Module table
Print results
Stop

Figure 1. DRIVER Functions

[2]The concept of the FORTRAN Synthetic was developed by the authors. Program development was by Messrs. Donley Hoffman, James Phelan, and Howard de St. Germain of the US Army Training and Doctrine Command Systems Analysis Activity (TRASANA), White Sands Missile Range, New Mexico. Revisions to the initial coding and implementation of the Synthetic into benchmark workload mixes were cooperative efforts.

DRIVER, Figure 1, is the managing module or main program for the FORTRAN Synthetic. Each of the other modules of the Synthetic is independent and dedicated to performing a specific data processing function. DRIVER accepts and interprets control cards, transfers program control to the appropriate modules and initiates output operations. DRIVER and module control parameters are required inputs. DRIVER parameters specify the number of times the selected set of functional modules will be executed, the print option selected (see Table 2 for an explanation of the print options), and the seed value for the pseudo random number generator. Module control parameters identify the set of modules to be used in a particular application and specify the module major and minor loop control values which are passed from DRIVER to a module for functional control. The modularity and functional independence of the FORTRAN Synthetic are depicted in Figure 2. Table 1 presents a list of the modules of the FORTRAN Synthetic by name and numeric identifier (used as an input parameter) and briefly describes the programing function performed by each module. (The FORTRAN Synthetic currently contains 13 functional modules. The structure of the Synthetic does not preclude the creation and insertion of additional functional modules.)

Table 1. Functional Module Descriptions

Module name	Module identifier*	Description
FINPT**	1	Formatted input
FOUTPT	2	Formatted output
NFINPT	3	Nonformatted input
NFOTPT	4	Nonformatted output
MDIM	5	Multiple dimension array manipulation
DECIS	6	Decision making***
IARITH	7	Integer arithemetic
RARITH	8	Real arithmetic
MATH	9	Mathematics functions****
DAIO	10	Direct access I/O
CALLS	11	Subroutine calling
SORT	12	Sort
FOTPTU	13	Unprinted formatted output

*This identifier appears on the module control card.

**Requires dummy input in card form. If this module is used in a run, the input should be appended to the control card deck (see Figure 5).

***Logical IF, arithmetic IF, and GO TO.

****Sine, cosine, square root, and exponential.

4.1 Module Structure

Even though the functional modules are independent, they have a common structure. A flow diagram describing the general processing characteristics of the functional modules is presented in Figure 3, a listing of the source code for the integer arithmetic module IARITH is presented in Figure 4 and Table 2 describes each of its variables and parameters.

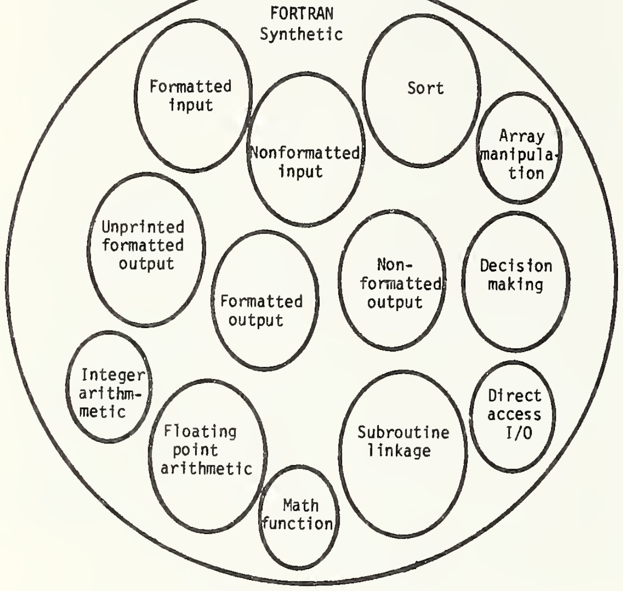


Figure 2. Moduler Structure of the FORTRAN Synthetic

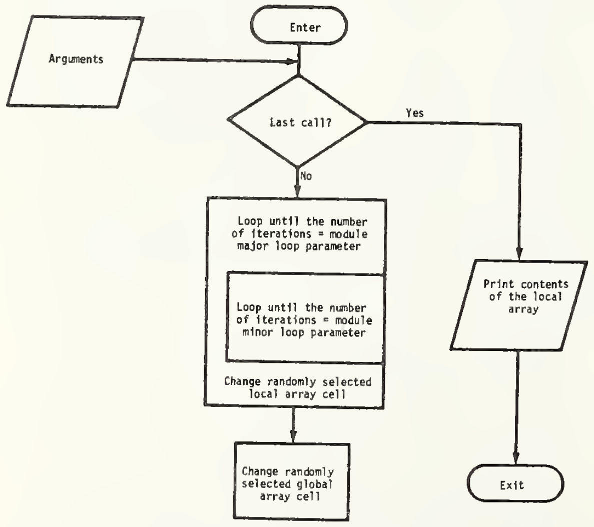

Arguments
Enter
Last call?
Yes
No
Loop until the number of iterations = module major loop parameter
Loop until the number of iterations = module minor loop parameter
Change randomly selected local array cell
Change randomly selected global array cell
Print contents of the local array
Exit

Figure 3. Generalized Functional Module

```
 1            SUBROUTINE IARITH)NREP,IPRFL6,LOOP)
 2            CONNON /BLOBAL/ NOLOB,IOLOB(1)
 3            DIMENSION LOCAL(1)
 4            DATA LOCAL /1*0/
 5            DATA IOIN /1/
 6            DATA INCR/1/
 T            DATA NOEF/1/
 8            DATA INT1,INT2/T7,-53/
 9      C
10      C      TNIS NOOULE PERFORNS FORTRAN INTEGER ARITNNETIC.
11      C      RESULTS ARITNNETIC OPERATIONS ARE RANDONLY STOREO IN TNE
12      C      'LOCAL' ARRAY ANO PRINTED AT THE ENO OF TNE RUN
13      C      WHEN CALLEO WITN PARAMETER 'NREP' SET TO ZERO.
14      C      TNE NUNBER OF REPETITIONS OF TNE NOOULE NINOR LOOP
15      C      IS SPECIFIEO BY THE PARAMETER 'LOOP'.  IF LOOP = 0
16      C      THE DEFAULT VALUE NOEF IS USEO.
17      C
18      C      RULES FOR SETTINO  KEY VARIABLES:
19      C            NOEF SNOULD BE SET IF DEFAULT VALUES ARE TO BE
20      C             USEO )I.E., LOOP = 0)
21      C            DINENSION OF IGLOB .EO. DINENSION OF IGLOB IN DRIVER
22      C            OINENSION OF LOCAL .EQ. DESIREO SIZE ANO EACH
23      C             CELL OF LOCAL INITIALIZEO TO ZERO )SEE LINE 4)
24      C            IOIN .EO. DINENSION OF LOCAL
25      C            INCR .LT. IOIM
26      C            INT1 .EQ. ANY POSITIVE INTEOER )SEE LINE 8)
2T      C            INT2 .EQ. ANY NEBATIVE INTEOER )SEE LINE 8)
28      C
29      C
30            IF (NREP.EQ.0) 8O TO 1000
31            IF(LOOP .EQ. 0) LOOP = NOEF
32            CALL IRAND)IOIM,INOX)
33            INO=INOX
34            IF )INO.OT.IOIN/2) INO=-INO
35            DO 200 I=1,NREP
36            DO 100 J=1,LOOP
3T            INT1=(INT1-INO)/)J+5)
38            INT2=(INT1+INT1)-((J-9)*I)
39        100 CONTINUE
40            LOCAL(INOX)=LOCAL)INOX)+INT2
41            INOX=INOX+INCR
42            IF )INOX.OT.IOIM) INOX=INOX-IOIM
43        200 CONTINUE
44            CALL IRAND)NGLOB,INOX)
45            IOLOB(INOX)=INT1
46            RETURN
47      C
48      C      PRINT RESULTS
49      C
50       1000 WRITE (6,11000)
51            CALL PRTINT(LOCAL,IOIN,IPRFLG)
52            RETURN
53      11000 FORNAT (26H1OUTPUT FRON NOOULE IARITH)
54            END
```

Figure 4. Integer Arithmetic Module

4.2 Module Control

In order to correctly use a module, critical variables must be set prior to compilation. These critical variables are identified by the symbol # in Table 2 and rules for setting these variables are described in Figure 4. (Each module of the Synthetic contains internal documentation describing the purpose of the module, the meaning of arguments passed to the module by DRIVER and rules for setting key local variables.) Module control parameters are transfered to the module as arguments in the call from the DRIVER. Three control parameters are required by each module: (1) the number of major loop iterations; (2) the number of minor loop iterations; and (3) the desired print option. The module minor loop control need not be specified, if the use of the default value (NDEF) is desired. In this case, NDEF should be set to tune the module to consume a specific quantity of resource (CPU and I/O time); then only the module major loop and the DRIVER loop parameters are required to tune the Synthetic for each application.

Table 2. Variable Usage in the Integer Arithmetic Module (IARITH)

Name	Type	Usage
I	Integer	Major loop control.
IOIM#	Integer	Upper bound for local array subscript. Must be set equal to the dimension of LOCAL.
IGLOB#	Integer	Common array shared by all modules.
INCR#	Integer	The interval for "stepping through" LOCAL.
INO	Integer	Pseudo random number used for arithmetic.
INOX	Integer	Pseudo random number initially returned by IRAND* and incremented by INCR. Used for LOCAL and IGLOB referencing.
INT1#	Integer	Result of arithmetic operation.
INT2#	Integer	Result of arithmetic operation.
IPRFLG	Integer	Calling argument to control the amount of printed output to be produced.**
J	Integer	Minor loop control.
LOCAL#	Integer	Array local to the module. Used for memory sizing.
LOOP	Integer	Calling argument specifying the number of iterations of the module minor loop.
NOEF	Integer	Default value for number of minor loop iterations. Used when the argument LOOP = 0.
NGLOB	Integer	Upper bound of global array IBLOB subscript.
NREP	Integer	Calling argument specifying the number of major loop iterations.

*IRAND is a pseudo random number generator provided as a support module for the FORTRAN Synthetic. This module produces a uniform distribution of positive integers. The inclusion of this module enhances transportability and reproducability of the Synthetic.

**Four print options are available: (1) no print; (2) print all arrays; (3) print those array values that have been changed during processing; and (4) print every nth changed array value.

4.3 Module Logic

All modules have similar logic. (In the following discussion all line number references refer to Figure 4.) If the value of the major loop parameter NREP is zero, (indicates final call to a module) the major and minor loop sections of the module are bypassed (line 30) and the module initiates a call (line 51) to the support module PRTINT to cause the appropriate printed output to be generated. The module major loop (lines 35 through 43) causes the value of a selected cell in the local array to be changed to the value resulting from a minor loop (lines 36 through 39) operation. For example, in IARITH, the value of INT2 is stored in the selected local array cell (line 40). The selection of the local array cell is random for the first iteration of the major loop and a function of the initially selected cell and the variable INCR for all subsequent iterations of the major loop. The module minor loop performs a language function, e.g., integer arithmetic. Upon completion of the prescribed number of minor and major loops, the module stores the result of a minor loop operation in a randomly selected cell of the common array IGLOB (line 45) and returns control to DRIVER (line 46).

4.4 Module Table

DRIVER directs the processing of all modules, based on the module control deck. Module control parameters are stored by DRIVER in a array which is referred to as the Module Table. As it is currently written, DRIVER will accept up to 100 entries in the Module Table. The Module Table permits considerable flexibility in the design of a particular application of the synthetic. For example, assume that it has been determined that in order to simulate a particular application program the following number and sequence of operations needs to be repeated 10 times: four nonformatted input, three integer arithmetic, four subroutine calls, a sort of 100 records, two integer arithmetic, and two formatted output. The Module Table would contain six entries--one for each function in the sequence. In order to simulate the workload described in the example, DRIVER causes the set of functions specified in the Module Table to be executed ten times, i.e., there would be ten iterations of the DRIVER loop (see Figure 1). After completing the appropriate number of DRIVER loop iterations, DRIVER initiates one additional call to each module identified in the Module Table to generate printed output, i.e., in this last call, the value of the argument NREP (the number of module major loop iterations) is zero.

A control card set, Figure 5, consists of one DRIVER control card, several module control cards, and, if required, input data cards. The latter are required only for the formatted input module (FINPT). Each module control card becomes an entry in the Module Table. Figure 6 presents the control card images required for the FORTRAN Synthetic to simulate the example application program described in the preceding paragraph.

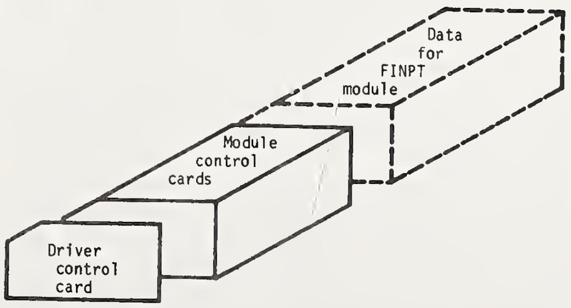
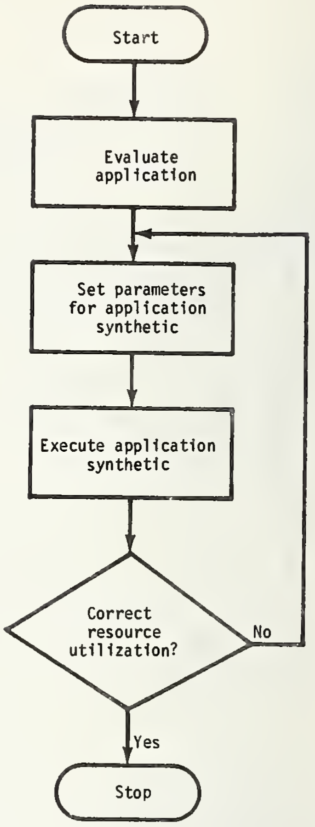


Figure 5. Control Card Deck for the FORTRAN Synthetic

```
             Card column*
Card
number   1-10    11-20    21-30

  1       10       1       1513
          |        |         |____ Pseudo random number seed.
          |        |______________ Print flag.
          |_______________________ Number of iterations of the
                                     DRIVER loop.

  2        3       4         1
          |        |         |____ Module minor loop iterations.
          |        |______________ Module major loop iterations.
          |_______________________ Module identifier for NFINPT
                                     (see Table 1).

  3        7       3         5

  4       11       4         0

  5       12     100       100

  6        7       2        10

  7        2       2         1

  8        0       0         0     Designates end of module
                                     control deck.

*Values should be right-justified.
```

Figure 6. Example Control Card Set



Start
Evaluate application
Set parameters for application synthetic
Execute application synthetic
Correct resource utilization?
No
Yes
Stop

Figure 7. Workload Mapping Process

5. Workload Mapping for the FORTRAN Synthetic

The process of mapping an application program to the synthetic is depicted in Figure 7. The first step in the mapping process is to determine the programing functions within an application program or set of application programs that are to be represented. This process generally involves some subjective evaluation by a programer familiar with the application, since few utility programs exist for the purpose of generating language function usage statistics. For a given application, the analyst must identify memory requirements and the frequency of use and sequence of execution of programing functions. Having identified the relevant characteristics of the application program, the analyst selects the appropriate functional modules, sets the size of local arrays to achieve the memory usage desired, sets local variables in each module, and compiles these modules into an object program. Initial settings of parameters are made, and the application synthetic is executed. The run time statistics of the synthetic are compared to those of the application to determine the degree of agreement. This process of setting parameters and executing the synthetic is repeated until the variance in run time statistics between the synthetic and the application program is acceptable.

Significant flexibility in "tuning" the synthetic is provided by giving the user parameterized control at every level of processing, i.e., DRIVER, module major and minor looping. The FORTRAN Synthetic provides the means whereby almost any workload mix (i.e., one consisting of applications exhibiting different mixes of language constructs) can be represented through concurrent processing of several versions of the Synthetic with different sets of parameter values and local array and variable settings for each version. For example, assume that an actual workload, W, consists of three classes of application programs A, B, and C. Therefore, $W = A + B + C$. The objective is to design a benchmark workload mix, W', such that $W' \cong W$. Through an iterative process of workload mapping, a set of three synthetic programs are configured such that $A' \cong A$, $B' \cong B$, and $C' \cong C$, where A', B', and C' are versions of the FORTRAN Synthetic. Thus, W is represented by W' in that $W' = A' + B' + C' \cong W$.

6. Conclusion

The FORTRAN Synthetic, consisting of approximately 1,150 lines of code (including comments), provides a flexible tool for constructing the components of a benchmark test.

Coupled with workload analysis, the Synthetic can be configured to represent the salient characteristics of a variety of workloads. The Synthetic offers the advantages of being machine independent, easy to use, highly flexible, and capable of producing experiments which are repeatable.

References

[1] Bucholz, W. A synthetic job for measuring system performance, IBM Sys J. 8, 4 (1969), 309-318.

[2] Conte, Dennis M. Findings of the Standard Benchmark Library Study Group. Institute for Computer Science and Technology, National Bureau of Standards, Washington, 1979, Appendix A.

[3] Ferrarri, Domenico, Workload characterization and selection in computer performance measurement, Computer (July/August 1972), 18-24.

[4] Sreenivason, K. and Klieman, A. J., On construction of a representative synthetic workload, Comm ACM 17, 3 (March 1974), 127-133.

[5] Wood, David C. and Forman, Ernest H., Thoughput measurement using a synthetic job strem, AFIPS Conference Proceedings 39 (1971), 51-55.

The NBS Network Measurement Instrument

Marshall D. Abrams

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

Dorothy C. Neiman

Commtex Inc.
2411 Crofton Lane
Crofton, MD 21114

The NBS Network Measurement Instrument (NMI) represents the third generation implementation of an approach to the measurement of interactive computer networks, teleprocessing systems, and network services which focuses on the service delivered to users rather than on the internal operating efficiency of the system. The information obtained aids users in the quantitative evaluation of such systems and services. The performance measures and measurement conditions are described. The applicability of the stimulus - acknowledgment - response model to interactive asynchronous, character synchronous (e.g., bi-sync), and bit synchronous (e.g., SDLC and ADCCP) communication is discussed. The NMI is presented in terms of its functions, statistical capabilities, architecture, and communications protocol state transitions.

Key words: Computer; computer communications; computer performance measurement; data measurement; measurement; performance measurement.

1.0 INTRODUCTION

The NBS Network Measurement Instrument (NMI) is the third generation system employed by NBS to measure the quality of service delivered through a computer network. The NMI has been developed under contract to NBS by Commtex Inc., and is implemented around a novel multiple microprocessor system architecture to achieve minimal costs and portability while carefully adhering to the measurement conditions and stimulus - acknowledgment - response model which characterized previous NBS measurement systems.

The NMI embodies a methodology for the measurement of computer performance in terms of the service provided to the interactive terminal user. The orientation, or philosophy, supported by the NMI is that performance measurement techniques which indicate internal measures of performance are essentially meaningless to a remote user (or user population). The user is more concerned with the visible amount of work performed and the cost of the interactive service. Examining performance from a user's viewpoint leads to external measures of performance.

2.0 PERFORMANCE MEASURES

Computer service requirements should be stated in system-independent functional terms. For interactive use, three key measures are response time, turnaround time, and throughput. While there are many definitions of response time, a preferred definition is the elapsed time from the last user keystroke until the first meaningful system character is displayed at the user's terminal [3]. Turnaround time is measured from beginning to end of a specified sequence of operations, usually called a "job." The beginning and end may be specified points in the interactive dialogue such as the first character of logon to last output following logout. In transaction processing, where there may be neither logon nor logoff, a job is equivalent to a series of transaction steps all of which are concerned with a single subject. Throughput is calculated by dividing the number of jobs completed in a period of time by the duration of that period. The validity of throughput measurements is increased by employing a long time period, such as a day or a week, thereby averaging differences among jobs.

Individual users are concerned with the service they receive, not the throughput of the entire system. In contrast to the evaluation of system efficiency, which is concerned with the time and cost to run a group of jobs, service evaluation is concerned with the time and cost to run each individual job. Actually, the times of concern are of different types. Efficiency is concerned with the CPU time charged to the job(s), while service evaluation is concerned with total elapsed time of a job. (In most multiprogramming systems, the elapsed time for a job is considerably greater than the CPU time charged to the job.) The evaluation of efficiency is thus based on the measurement of the internal functioning of a computer system taken as a whole, rather than of its external manifestations, taken individually.

The goals of improved efficiency and improved service may well be at odds with one another. An improvement in internal performance does not necessarily imply an improvement in service. Indeed, the opposite may actually be true. Frequently, it is only possible to improve service at the expense of efficiency.

For procurement purposes, a combination of response time, turnaround time, and throughput may best describe organizational requirements. Specification of requirements by a combination of performance measures makes it possible to include different levels of concern.

3.0 THE NBS NETWORK MEASUREMENT SYSTEM

In 1975 the NBS Institute for Computer Sciences and Technology developed a Network Measurement System (NMS) [2] for measuring the performance of computer networks in terms of the services they render. Although there were many additional factors [1], the quality of service was evaluated in terms of those parameters which were most straightforward to quantify, namely response time, throughput, and turnaround time, all of which are defined in context below. The NMS consists of a data acquisition device called the Network Measurement Machine (NMM) and a Data Analysis Package (DAP) for generating reports about the quality of network service delivered to interactive terminal users as well as a characterization of user demands and network utilization.

The Network Measurement Machine (NMM), shown in the photograph behind the NMI, was implemented on a PDP 11/20* and employs regular and special purpose hardware controlled by a specially written software system. The regular hardware included the processor, an operator's console, disk and magnetic tape storage, two programable clocks, and data communications interfaces. Special purpose hardware was employed to connect the NMM to the network that was to be measured. This hardware included an automatic calling unit and line selector (ACU/LS) for computer-controlled origination and answering of data calls, and a specially designed communications line interconnections device called a "data probe".

The special software system was a real-time interrupt-driven scheduler incorporating various drivers and handlers for the standard and special purpose peripherals attached to the NMM. A command interpreter serving the operator's console made it possible to view the status of the NMM and to view the status of selected ongoing communication.

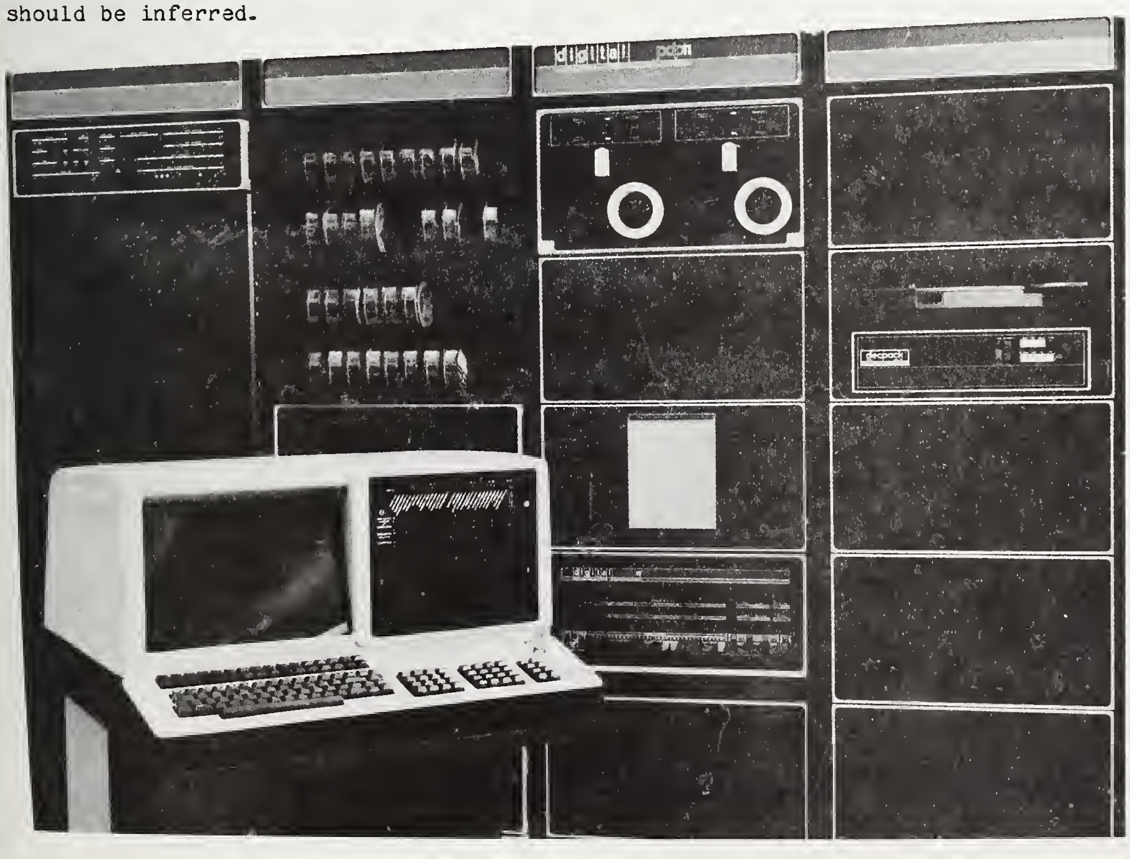
----

*Commercial products are named for identification purposes only. No evaluation should be inferred.

Data was not structured nor analyzed during acquisition. Rather, all characters were time-tagged and written onto magnetic tape for subsequent analysis. Conversations from up to eight terminals operating in asynchronous mode could simultaneously be acquired.

Once recorded, the data are processed by the DAP. The processing may be briefly described as follows: The multiple conversations on the tape are first separated into individual conversations. Each conversation is then processed to remove character echoes (if appropriate), and scanned to build a structure file which contains pointers to the user and network messages. Different stratifications of the data (e.g., grouping by software processor employed by the user) can be noted in the structure file. Conversations may then be analyzed individually or in aggregate, reports generated, and a file written for additional data processing by independent statistical packages.

The basic function of the NMM was to receive characters from both computer and user and to record these characters as well as information concerning the source of the



character, the time at which the character occurred, and the state of the communication line at the time the character existed. Aside from this basic function, the most important feature of the NMM was that it did not perturb the network being measured.

4.0 STIMULUS ACKNOWLEDGMENT RESPONSE MODEL

Complex phenomena are often best understood in terms of models. In order to investigate the service delivered by interactive systems, a set of computer programs has been written to implement several models. These programs accept as input the measured data from the NMM and calculate various items of derived data required by that model. Statistical analysis of this measured and derived data is also performed, as is report generation. An understanding of these models is fundamental to an understanding of our approach to network measurement and evaluation. The model employed for interactive computer utilization is called the "Stimulus-Acknowledgment-Response" model.

The "stimulus" is the input from the user; the "response" is the output from the computer caused by the stimulus. As will be further explained below, the "acknowledgment" is the output from the computer, or the network, which occurs after the stimulus but before the response. The acknowledgment may convey some information to the user (e.g., that the stimulus was received and is being processed) which is not germane to the content of the stimulus.

4.1 Asynchronous Communication

Character asynchronous computer communication often employs the convention that a line constitutes a unit of input and that a line is terminated by a carriage return (CR). Since the CR (only) returns the print position to the first column of the current line, the computer issues a line feed (LF) following receipt of a CR to move the print position to the following line. Other control characters may accompany the LF, but their presence will not be recognized by the user since they have no visible effect. Non-printing characters are included in the acknowledgment as an expedient.

At first one might think that these non-printing characters constitute the beginning of the response; upon careful consideration one concludes that they do not. The LF is important, however, in that it provides feedback which reassures the user that the computer is still functioning. Nevertheless, this LF does not constitute a meaningful response to the stimulus. In psychological terms, the LF provides partial closure, but the user is still waiting for complete closure to be provided by the response.

4.2 Synchronous Communication

The acknowledgment identified in asynchronous interactive computer utilization may also be present in synchronous interactive utilization. It is important to differentiate between this user-level acknowledgment and the communications protocol acknowledgment. The communications protocol acknowledgment is at much too low a level of detail to be of interest when evaluating service. In synchronous communication, as in asynchronous, the user-level acknowledgment must be recognized by semantic analysis. To account for this receipt of system characters which do not convey information to the user, a state was introduced called the "acknowledgment."

5.0 INTERACTIVE TERMINAL COMMUNICATIONS

An interactive communications environment consists of a source device (usually a terminal) connected to a destination device (usually a computer) by an electrical data path. The total data path is usually complex, incorporating such communications facilities as local hardwired digital transmission lines, modems for digital to analog signal conversion, and common carrier public or leased telephone lines of varying quality and speed. This data path supports binary, bit serial, and full or half duplex data transmission. The NMM measured only character asynchronous transmission. The NMI measures character asynchronous, character synchronous, and bit-oriented synchronous communication.

5.1 Asynchronous Bit-serial ASCII Communication

The NMI measures one channel of asynchronous bit-serial communication in ASCII which connects to the Data Terminal Equipment (DTE) through an RS-232-C interface. Communication conforms to FIPS 17-1 (ANSI X3.16-1976). Terminals supported include, but are not limited to, Teletype model 40, Xerox 1620, and Tektronix 4000-series. For such terminals, the NMI is able to automatically measure a newly connected terminal. Transmission rates of 110, 150, and 300 bps are detected through recognition of a predetermined character entered by the terminal operator. The NMI is capable of accepting a command to identify this character. The NMI provides a command to identify the communications rate when it exceeds 300 bps. The rates supported include, but are not limited to, 600, 1200, 1800, 2400, 4800, 7200, and 9600 bps.

5.2 Character-oriented Synchronous Communication

The NMI measures one channel of character-oriented synchronous communication employing IBM 3270 Information Display System protocol as specified in [4] and [5].

5.3 Bit-Oriented Synchronous Communication

The NMI measures bit-oriented synchronous communication employing the Advanced Data Communication Control Procedures (ADCCP) as specified in the most recent version of ANSI BSR X3.66 and the IBM Synchronous Data Link Control (SDLC) Procedures as specified in the most recent version of an IBM publication [4].

5.4 New Interface Standards

The NMI is able to measure three new standard interfaces. The first two are electrical interfaces and the third is a logical interface. The logical interface is RS-449, which specifies the transition from RS-232-C to RS-422 or RS-423). The electrical interfaces are RS-422 for balanced circuits and RS-423 for unbalanced circuits. Generally, equipment would utilize the logical interface (RS449) with either of the electrical interfaces (RS422 or RS423).

6.0 NMI FUNCTIONS

6.1 Processing Of Echo-plex Communication

When the DTE operates in echo-plex, the NMI is capable of identifying and eliminating from statistical analysis all characters which are echoed exactly as transmitted. The algorithm employed is derived from the one employed in the DAP [6]. The echo removal algorithm produces a half-duplex approximation of an echo-plex conversation.

Any number of user characters may be interposed between a user character and its echo. The problem is to identify when a character transmitted from the network is an echo and when it is the beginning of a network output sequence. The algorithm requires that the beginning of the user stimulus be identified. This is accomplished by defining any character from the user as terminating a network transmission. Likewise, any character from the network which is not an echo terminates the user transmission. The procedure involves placing user characters into a buffer and maintaining pointers to the end of the buffer as well as to the current user character. Each character from the network received is an echo candidate and has to be compared with the current user character. As long as a match exists, the pointer is advanced and the process repeated. When the end of the buffer is reached, the user transmission is terminated. If there is a nonmatch before the end of the buffer, the remaining user characters end the network transmission and begin the next user transmission. By this definition, endings and beginnings are strictly determined by time sequence.

The NMI provides a command which will cause it to accept lower and upper case alphabetic characters as equivalent for the purposes of echo removal.

The NMI provides a command to identify a known transformation which occurs in echo-plex operation. A known transformation is identified by the character transmitted by the terminal and the character string received by the terminal. For example, the character transmitted by the terminal could be ETX and the transformed echo could be the two-character sequence "^C" (the " is not part of the transformed echo). When a known transformation has been identified to the NMI through the use of this command, that known transformation is treated as if a single character exact echo had been detected.

6.2 Recognition Of Acknowledgment

The NMI separates the data transmission to the terminal from the network into two components for the purposes of the analysis described below. The first (in time sequence), the acknowledgment, may consist only of non-printing characters (also known as control characters). The second component, called the response, begins with the first meaningful printing character and extends to the end of the message group (also known as a transaction).

The NMI provides a command to identify a fixed header which occurs at the beginning of the data transmission to the terminal. This fixed header is a known string of characters which may be preceded and/or followed by non-printing control characters. When a fixed header has been identified to the NMI through the use of this command, the fixed header is included in the acknowledgment. For example, printing of the time of day and the cost of processing the stimulus would not be considered a meaningful response and would be treated as part of the acknowledgment. Thus the response begins with the first printing character following the end of the fixed header.

6.3 Measures, Statistics, And Reports

The NMI provides the following measures, statistics, and reports. Measures of time are demonstrably accurate to within 0.01 second.

6.3.1 Fundamental Measures

The NMI collects data sufficient to describe the following measures.

* Stimulus delay time: The elapsed time from the receipt of the last character from the network in one message group until the transmission of the first user character in the next message group.

* Stimulus transmit time: The elapsed time from the transmission of the first user character until the transmission of the last user character in a message group.

* Stimulus character count: The number of characters comprising the stimulus.

* Acknowledgment delay time: The elapsed time from the receipt of the last user character in the stimulus until the transmission of the first character from the network in the acknowledgment.

* Acknowledgment transmit time: The elapsed time from the transmission of the first character from the network until the transmission of the first meaningful character in a message group.

* Acknowledgment character count: The number of characters comprising the acknowledgment.

* Response delay time: The elapsed time from the receipt of the last character from the network in the acknowledgment until the transmission of the first meaningful character from the network in the response.

* Response transmit time: The elapsed time from the transmission of the first meaningful character from the network until the transmission of the last character in a message group.

* Response character count: The number of characters comprising the response.

6.3.2 Derived Measures

The NMI calculates the following measures by performing simple arithmetic operations on the preceding measures.

* Stimulus transmission rate: The stimulus character count divided by the stimulus transmit time.

* Acknowledgment transmission rate: The acknowledgment character count divided by the acknowledgment transmit time.

* Response transmission rate: The response character count divided by the response transmit time.

* Response time: The elapsed time from the transmission of the last character of the stimulus until the receipt of the first meaningful character of the response.

* Response completion time: The elapsed time from the transmission of the last character of the stimulus until the receipt of the last character of the response.

* Transaction time: Elapsed time from the transmission of the first character of the stimulus until the receipt of the last character of the response.

6.3.3 Statistical Treatment

The large number of observations which might be obtained by the NMI require statistical treatment before reports can be produced. The statistical treatment of the measures specified above is described in this section.

* Standard Sampling Interval: The NMI accepts a command from its operator to establish the upper and lower limits for each of the measures specified above. These limits constitute the specification of the standard sampling interval for each measure.

* Frequency Count Distribution: The NMI accepts a command to divide the sampling interval into a specified number of equal class intervals, which it employs to summarize the observations which occurred. A counter is associated with each class interval to record the number of observations of the measure which occurred during a measurement activity. Two additional counters are provided, one for all observations less than the specified minimum and one for all observations greater than the specified maximum. These equal class interval counts are directly utilized for the production of histograms.

* Percentile statistics: The NMI calculates 50% level (median), 90% level and 95% level statistics from the class interval counts. A given percentage statistic is defined as the number of observations which occurred at or below the stated percentage level.

  The NMI accepts a command to produce a report of the statistics and histograms calculated. The command specifies which report is required. For each measure the report contains an identification of the measure, the percentile statistics, and a histogram of the class interval frequency counts.

* Histogram content: The histogram presents combined class intervals in order to fit on the output page. The histogram is scaled to occupy at least 50 per cent of the area of the output page unless every occupied class interval is presented, in which case at least one dimension of the histogram occupies 70 per cent of the available space in that direction on the page. The histogram identifies the range of each class interval reported, the number of observations occurring in that class interval, the cumulative percentage of observations for each class interval, and a bar chart which displays the proportional number of observations in that class interval.

* Hard Copy: The report may be generated on hard copy. The NMI provides an RS-232-C port for connection of an external asynchronous terminal which may be employed to print the report. The terminal connection supports ASCII communication at 110 to 9600 bps and a page width of 80 columns.

* Soft Copy: The NMI provides a direct view display on which is presented all statistics, reports, and histograms specified herein.

7.0 NMI ARCHITECTURE

The NMI architecture is based on a three bus, three Z-80 microprocessor system organization to effect the total set of capabilities the instrument is intended to provide. As shown in Figure 1, the processors themselves are augmented by general and special purpose subsystems which further tailor the multi-processor configuration to the application. The most significant of these subsystems include the:

1. Dual Channel Serial Data Acquisition Subsystem -- which satisfies the unique requirement for a serial interface capable of monitoring both transmit and receive data paths of, optionally, either start/stop, synchronous or

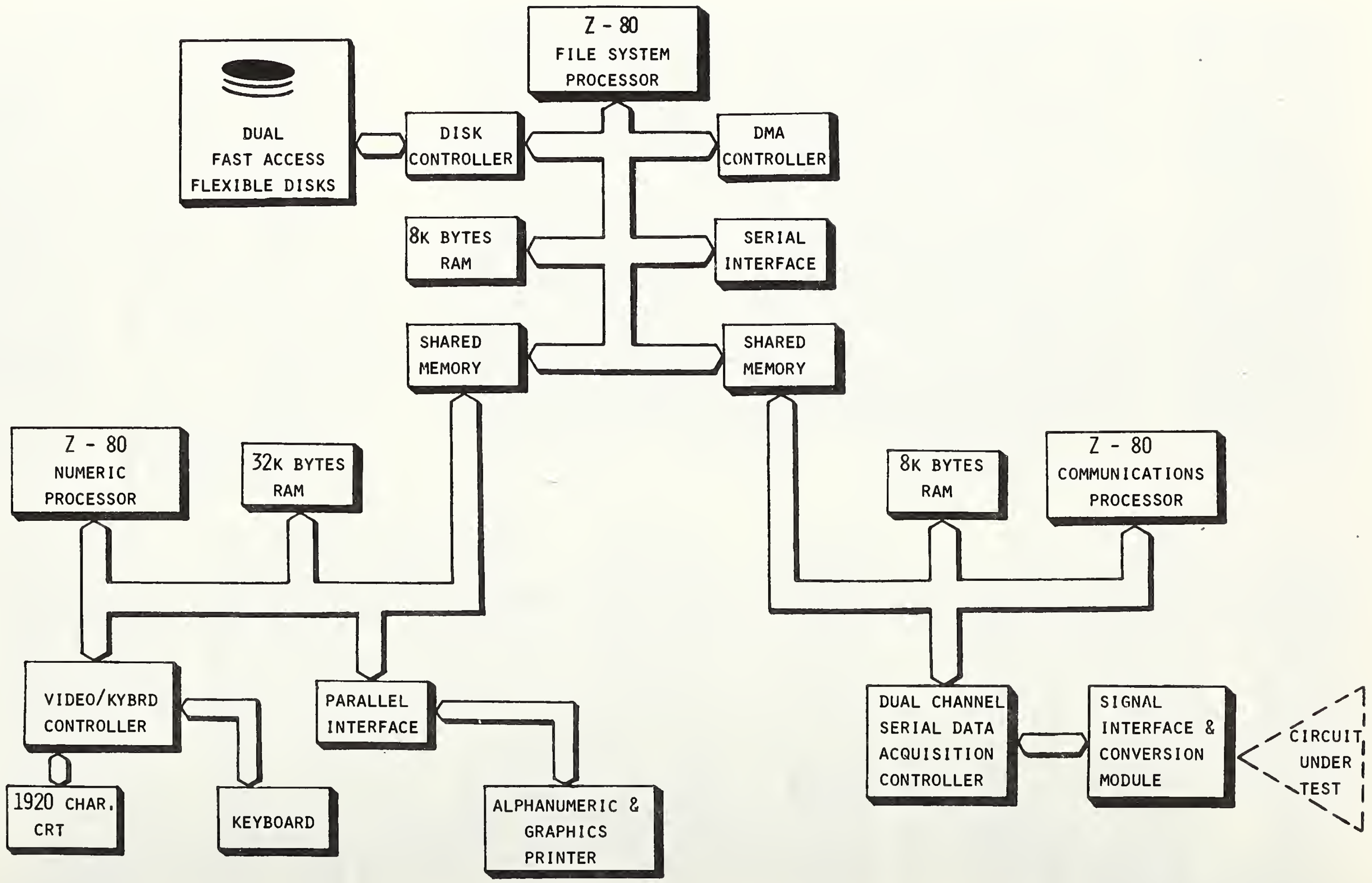


Fig. 1. Network Measurement Instrument -- System Architecture

bit oriented links (protocols) which may further require either RS-232-C, RS-422, RS-423 or current loop electrical interfaces.

This subsystem also provides the operator with the ability to "qualify" the instrument's execution in terms of any four "control" or "handshaking" signals contained in the electrical interface of the circuit under test.

2. Mass Storage Subsystem -- which, while taking advantage of the economics and portability of flexible disk technology, employs voice coil positioned drives. Access speeds are therefore several times faster than typical floppy disk systems. A random access seek can require as little as 33ms, as fast as many large disk systems.

   Fast bulk memory is therefore available to support the large storage requirements of a traffic analysis system as compared to a diagnostic tool.

The hardware formation, therefore, yields a closely coupled distributed processing system architecture which affords a partitioning of functions among the three processors described below.

7.1 The Communications Processor

The Communications Processor is responsible for directing the Serial Data Acquisition Controller to monitor data transfers on the circuit under test, to correlate those transfers to a real time clock, and generally to perform the measurement tasks which facilitate subsequent data reduction and statistical description of the traffic observed during the measurement period. This processor forwards measurement data to the numeric processors for additional analysis prior to storage by the File System Processor.

7.2 The Numeric Processor

The Numeric Processor, as the name implies, executes the data reduction programs which produce the measures, statistical summaries, and reports describing the nature of the data transmission on circuits monitored by the Communications Processor.

The Numeric Processor also handles operator communications from the system console and drives the hard copy printer onto which paper copies of measurement results are written.

7.3 The File System Processor

The File System Processor manages the files of measurement data as well as the measurement and data reduction programs. The measurement data is contained in three files which are produced at successive stages of data reduction. The Conversation File is a recording of the conversation being monitored and consists of time tagged, sequenced character count representations of user and network messages. The Measurement File is a sequential representation of the fundamental measures, described in paragraph 7.3.1, for each transaction observed during a conversation. The Summary File is a summary of the statistical treatment of a single measure over one or more conversations; it contains the counters associated with the equal class intervals composing the standard sampling interval.

The composition of measurement and data reduction programs responsible for measuring a conversation is a function of the communications mode: asynchronous, character-oriented synchronous, or bit-oriented synchronous. The File System Processor provides for the linkage to the necessary programs.

7.4 Data Measurement Logic

The data measurement logic is table driven, making it relatively easy to add the capability to measure other data communications protocols. These tables are derived from state diagrams, such as the one for asynchronous communication shown in Figure 2. The state diagrams for other communications protocols differ in detail only. It is, therefore, instructive to examine this state diagram:

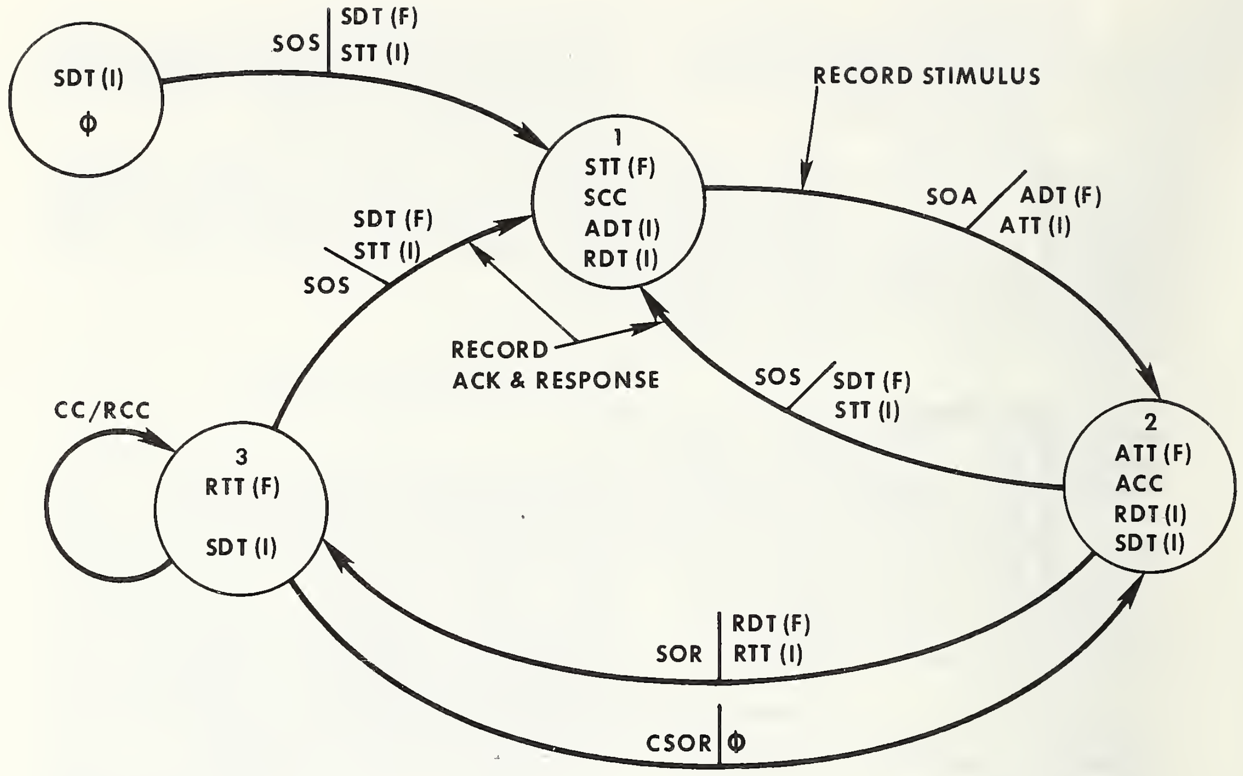


Fig. 2. Asynchronous SAR Model

1. The inputs represented in this diagram, which generate state transitions, are the beginning and endings of user and network messages composing a transaction: SOS, start of stimulus; SOA, start of acknowledgment; and SOR, start of response. An asynchronous communications transaction consists of stimulus, acknowledgment, and response messages only. The program logic which identifies these message boundaries is a combination of echo removal and fixed header recognition for asynchronous communications.

2. The non-transitional inputs are the message character counts and their associated times.

3. The outputs represented in this diagram are the variables associated with the fundamental measures: SDT (I) & SDT (F), stimulus delay time, initial and final; STT (I) & STT (F), stimulus transmit time, initial and final; SCC, stimulus character count; ADT (I) & ADT (F) acknowledgment delay time, initial and final; ATT (I) & ATT (F), acknowledgment transmit time, initial and final; ACC, acknowledgment character count; RDT (I) & RDT (F), response delay time, initial and final; RTT (I) & RTT (F), response transmit time, initial and final; and RCC, response character count. The variables updated by the transitional inputs are designated in the state transition diagram as follows: Transitional Input/Variable Updated. Tne variables updated by the non-transitional inputs are listed in the state bubble.

4. Stimulus measures are finalized by the beginning of network transmission; therefore, the transition from state 1 to state 2 causes the calculation and recording of the transaction stimulus measures. Likewise, the beginning of user transmission, a transition from state 2 or 3 to state 1, finalizes the acknowledgement and response measures.

7.5 Conclusions

The NMI implements the measures, data analysis, and data presentation recommendations from _Guidelines for the Measurement of Interactive Computer Service Response Time and Turnaround Time_ [3], in a portable instrument. As such, it makes possible the determination of computer service rendered which was previously very difficult to accomplish. The ability to measure asynchronous, character-oriented, and bit-oriented data communications is a very useful extension on the prior state-of-the-art.

8.0 GLOSSARY

acknowledgment
All the initial non-printing characters which are output by the network following a _stimulus_. A fixed heading on all output may be considered as part of the acknowledgment rather than as part of the _response._

echo-plex
The mode of terminal and network operation wherein the printer or display and the keyboard on the terminal are not connected so that every character transmitted by the terminal to the network must be transmitted by the network to the terminal in order to be visible to the operator.

first meaningful character
The first character in the output from the network which transfers information to the user. A printing character. When there is a fixed heading on all output which is defined as part of the acknowledgment, the first printing character following the fixed header.

message group
The _stimulus_ input from the user and the following _acknowledgment_ and _response_ output from the network constitute one message group. Also known as transaction.

network
The combination of data communications circuits and data switching components to which terminals and computers are connected. From the viewpoint of a terminal connected to the network, it is impossible to determine whether a character originates within a network processor or a remote host computer.

response
Output from the network. When there is an _acknowledgment_, the response begins with the _first meaningful character_.

stimulus
Input from the human to the network.

turnaround time
The time interval between the initiation of a job or function and the availability of the results.

9.0 BIBLIOGRAPHY

1. Abrams, M. D. and Treu, S., "A Methodology for Interactive Computer Service Measurement," _Communications of the ACM_, December 1977, pp. 936-944.
2. Abrams, M. D., Cotton, I. W., Watkins, S. W., Rosenthal, R., and Rippy, D. E., "The NBS Network Measurement System," _IEEE Transactions on Communications_, October 1977, pp. 1189 - 1198.
3. _Guidelines for the Measurement of Interactive Computer Service Response Time and Turnaround Time_, Federal Information Processing Standards Publication 57, August 1978.
4. _General Information -- Binary Synchronous Communication_, IBM Systems Development Division, GA27-3004-2, October 1970.
5. _IBM 3270 Information Display System Component Description_, IBM Corporation, GA27-2749-6, September 1977.
6. Watkins, S.W., and Abrams, M.D., _Interpretation of Data in the Network Measurement System_, NBS Technical Note 897, February 1976.

Tutorials and Case Studies

PERFORMANCE ANALYSIS OF A SATURATED SYSTEM

A Case Study

Nancy Lennon and Walter P. Bond, Jr.

Computer Sciences Corporation
Kennedy Space Center, FL 32952

This paper describes some recent experiences that the authors have had in attempting to gather decision-making data from an on-line system which was totally saturated, and for which minimal performance measurement tools were provided. The case study presented illustrates the limited use of data resulting from post facto measurement techniques. An attempt has been made to illustrate the type and quality of results which can be expected under these circumstances.

Key words: Inventory management; on-line; performance; response time; saturated system; transaction processor.

1. Introduction

The Shuttle Inventory Management System (SIMS) provides total on-line computer support for both the NASA Space Shuttle Program and the Kennedy Space Center base operation.

The initial design of SIMS assumed no hardware or software constraints; consequently SIMS was implemented on hardware which was unable to support the actual workload. Projections of workloads indicated that SIMS soon would experience an even greater backlog of transactions.

Response time was unacceptable; certain transactions required up to 30 minutes to complete. Queue lengths continued to increase throughout the operational day. Substantial down-time due to hardware failure further compounded the throughput problem.

The users of SIMS lacked confidence in the systems ability "to do what it said it would do" and often checked their work several times to make sure the transaction "took." This behavior factored greatly in the SIMS workload.

This combination of unreliable hardware, slow response time, inadequate throughput, and lack of user confidence led to an official post implementation system review.

2. Definitions

SIMS - The Shuttle Inventory Management System is a series of integrated on-line logistics support programs providing total supply support for Kennedy Space Center. SIMS was implemented on an H635 computer using the Transaction Processing Executive (TPE) and Integrated Data Store (IDS) software operating under GCOS.

TPE - The Honeywell Transaction Processing Executive.

TPAPS - The Transaction Processing Application Programs.

GCOS - General Comprehensive Operating Supervisor: The Honeywell 635 Operating System.

IDS -- Integrated Data Store: Honeywell database management system.

Response Time - The time between initiation of a transaction and completion of processing for that transaction.

Queue Time - The time a transaction spends in TPE before being passed to a TPAP.

3. Background

The functional description of the Shuttle Inventory Management System (SIMS) was released by NASA in October, 1974. At that time the decision was made to utilize the existing hardware configuration consisting of an H635 computer and associated peripheral equipment including a Honeywell DATANET 355 as the telecommunication front-end processor. The system software to be used included a version of GCOS operating system, with the Honeywell Transaction Processing Executive (TPE) as the only available software package to interface the applications control program with the DATANET 355. The application software development took place over a three year period and resulted in an operational on-line system in February, 1978. In September, 1978, a post implementation system review was conducted to determine the steps needed to improve SIMS performance.

3.1 SIMS

The Shuttle Inventory Management System is a supply support system comprised of three sub-systems:

1. Supply system - items, people, warehouses, stock, issues
2. Data system - records, reports, cataloging, purchase orders, stock, level
3. Computer complex - H635 and peripherals, automation of data system

3.2 Hardware Description

The SIMS Computer Complex to be investigated consisted of a dedicated Honeywell 635 computer (256K), with the following peripheral equipment:

16	IBM 2314 disk drives
2	180 disk drives
1	DATANET 355
73	Remote terminals

3.3 Software

The SIMS software occupied 250K words of memory (out of 256K) allocated as follows:

General Comprehensive Operating Supervisor	36K
Transaction Processing Executive	70K
Transaction Processor Application Programs	
Inventory	72K
Cataloging	41K
Output Procedure	22K
Recoveries	9K
Total	250K

The SIMS database and software required 80M words of on-line storage.

3.4 Workload Summary

The actual workload was averaging 5,000 transactions per day with a backlog of 2,000 to 3,000 per day. The projected workload for the next five years ranged from 9000 transactions per day to a maximum of 14,000 transactions per day.

3.5 Performance Considerations

Intuition established the framework for analyzing SIMS:

A. Response time appeared excessive (No comparison to similar systems was available)

B. System would not be able to process projected workload (Was workload realistic?)

C. No performance data were available (No requirement, lack of resources)

D. SIMS was assumed to be I/O bound (Response time problem infers too many terminals)

E. TPE was a known problem area (Inappropriate tool for high volume transaction processing)

F. Immediate improvement needed (With no impact on resources)

4. The Performance Analysis

The purpose of the SIMS study was to identify the problems and to suggest alternatives which could improve system response time and increase system throughput.

4.1 Quick Fixes

Quick fixes were suggested based on the following assumptions:

- o SIMS was I/O bound (too many terminals)
- o SIMS would not be able to process projected workload within required time frame (on-line load too heavy)

In order to address the I/O problem, the terminal operators were put on a schedule by which each terminal was on-line only during a specified portion of the day.

To solve the throughput problem, a batch collection facility was installed which routed lower priority transactions to a batch collection file for overnight processing.

Some improvement was recognized by these quick fixes.

4.2 Performance Data

The lack of system performance data prevented early identification of the SIMS problems. To install software probes and operating system measurement code would have degraded the already unacceptable response times even further. Additionally, the available core was limited due to TPE buffers.

The initial sources of data to measure SIMS were as follows:

END-OF-JOB statistics
Memory Dumps
Terminal Operator Logs
Stopwatch Studies
H635 Console Logs

In order to gain insight into the response time and queue time problems, a software patch was added to the transaction processing journal tape to record time into TPE, time into TPAP, time out of TPAP, and time out of TPE for each on-line transaction. These data were categorized by transaction, by TPAP, and by location.

Based on these limited data and an analysis of SIMS processor utilization, some tentative conclusions resulted:

- o SIMS system is not I/O bound (see Figure 1)
- o SIMS system is CPU bound
- o TPE is "hogging" CPU
- o Need more than two paths through the system (more TPAPS)

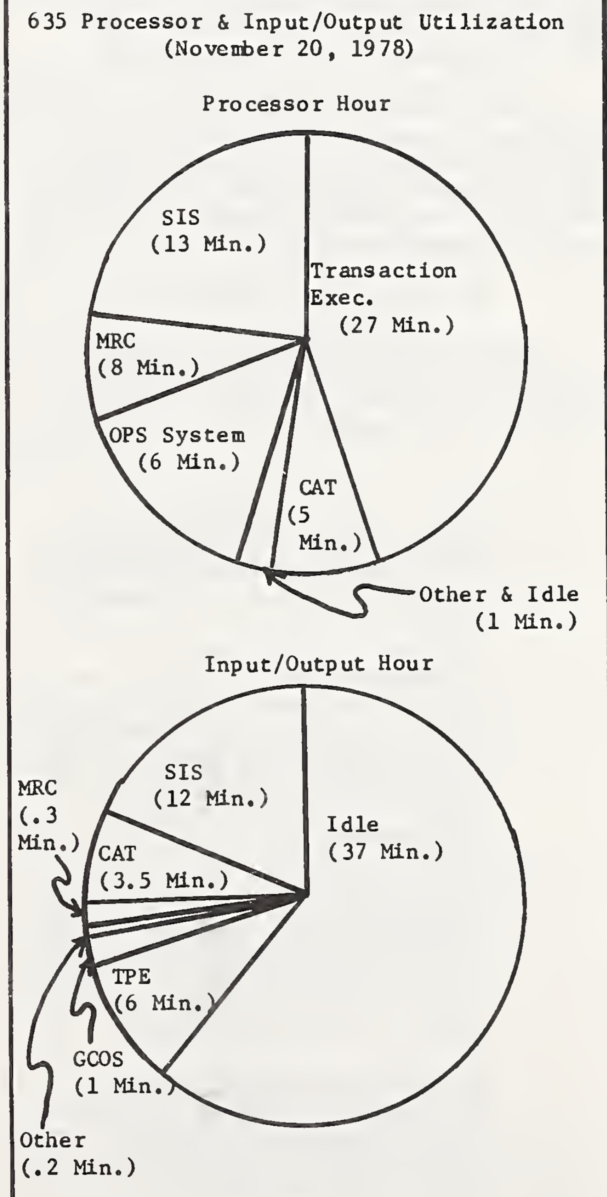


Figure 1

4.3 Interim Solutions

The CPU problem could be addressed in two ways:

- o Faster CPU (Replacement of H635)
- o New transaction processor (Replacement of TPE)

Due to the excessive down-time experienced by the H635, the decision was made to obtain a faster CPU. Subsequently a Honeywell 66/60 with a memory size of 512K was procured by NASA for the SIMS system. The SIMS system was converted with few enhancements to the H66/60. The response time improvement was approximately 40%.

Enhancements to the Honeywell File Management System permitted the two SIMS TPAPs to be segmented functionally into eight independent TPAPs. This change contributed to the total throughput improvement of approximately 60%.

4.4 Long Range Solutions

Software conversion to the H66/60 provided an opportunity to enhance SIMS without total redesign; however, several major system improvements remain possible.

A. TPE - TPE was designed for low volume systems; studies are underway to investigate conversion to another of Honeywell's transaction processors.

B. Fragmented File - A final step in the SIMS improvement plan is to examine the database. A number of redundancies were incorporated into the database design as "work-arounds" for initial hardware inefficiencies.

C. Overlay Activity - In order to compensate for lack of memory, heavy overlay activity was designed into the initial system. This inefficiency remains a source of possible system degradation.

D. Optimize Application Programs - The Honeywell Extended Instruction Set (EIS) was not available on the H635. SIMS TPAPs can be recompiled under EIS on the H66/60.

E. Performance Data - In order to provide an adequate base-line for future system decisions, daily performance data are to be gathered, analyzed and archived.

5. Conclusion

Post implementation system performance measurements have often been the rule. As illustrated by this case study, such limited measurements can be gathered and, further, can ultimately lead to correct decisions. However, techniques for the collection and analysis of such data in a saturated operational environment must necessarily be cursory; results will be marginal and may lead to erroneous conclusions. Provision for the systematic collection of performance data is an essential component of the initial system specification.

The authors wish to recognize the contributions of both the NASA SIMS Study Teams and of Roy Kilpatrick and Dale Hurtig of Computer Sciences Corporation. However, the authors assume total responsibility for any inaccuracies or omissions in this report.

TELEPROCESSING TRANSACTION THRUPUT PERFORMANCE

Bob Irwin

Programmer/Analyst: Computer Performance/Capacity Unit
The Hartford Insurance Group

Financial industries are dependent on Teleprocessing systems to execute their daily transactions. Unacceptable Teleprocessing performance can, and does, impede the expected flow of line of business transactions, thus degrading the anticipated level of work units for that line of business. This paper is an analysis of TCAM's tuning parameter: BUFFER DELAY VALUE. The analysis develops a mathematical characterization of TCAM's BUFFER DELAY VALUE which describes the race conditions inherent in TCAM logic. Such race conditions potentially degrade Teleprocessing transaction throughput. The analysis resulted in a TCAM modification to search a table of BUFFER DELAY VALUES, where each value corresponds to a number of stations currently active on the line. The instantaneous optimal DELAY VALUE generated by this table acts to optimize line utilization and prevents station input transaction line lockouts.

Key words: Input lockout; mathematical modeling; queuing models; race conditions; TCAM data flow; teleprocessing.

1. Introduction

The objective of this paper is the development of a mechanism to optimize TCAM half duplex line utilization to avoid conditions of line input lockout resulting from dominating line output processing.

Such a mechanism takes the form of a mathematical characterization of TCAM's global tuning parameter, BUFFER DELAY VALUE (BDV). This mechanism is currently being coded at The Hartford Insurance Group to promote the balancing and optimizing of telecommunication half duplex line utilization for particular lines of business.

Sections 1 and 2 describe a model of TCAM message flow, TCAM's EQUAL PRIORITY SCHEDULING ALOGRITHM and the interactions between the model and the algorithm which result in line input lockouts. Section 3 describes the mechanism developed to mitigate conditions of telecommunication line input lockout. Lastly, Section 4 describes the algorithm added to TCAM which dynamically chooses a BUFFER DELAY VALUE for a particular station on a teleprocessing line.

1.1 Definition of Symbols

MSG:	Message
STA,S:	Station
DQ:	Destination Queue
APGM:	Application Program
Q:	Queue
MH/LI:	Message Handler/Line In
MH/AO:	Message Handler/Application Out
MH/AI:	Message Handler/Application In
MH/LO:	Message Handler/Line Out
STCB:	Station Control Block
LCB:	Line Control Block
BDV:	Buffer Delay Value
L:	Line

2. TCAM Message Flow

The Hartford Insurance Group implements a large scale teleprocessing system to process daily insurance transactions spread over a national network of lines and branch offices. The teleprocessing system executes on an AMDAHL V6 computer using SVS and TCAM. The message processing program interfacing with TCAM is named TELLY and processes insurance transactions or message types such as:

- o NEW BUSINESS
- o INQUIRY
- o POLICY CHANGE
- o RENEWAL
- o CLAIM
- o CANCELLATION

The messages constitute the online transaction flow of the business. As such, improvements or degradation in the performance of these messages/transactions have a direct impact on the throughput potential of various lines of business within the corporation.

The following model (see Figure 1.) and narrative description is a generalized "walk through" of TCAM message flow.

2.1 NARRATIVE DESCRIPTION, refer to figure 1.

MH/LI

- o Message (MSG) stored in CPU buffer associated with line
- o MSG prefix is filled and other validation checks are performed
- o MSG is placed on destination queue (DQ) associated with application program (APGM) or accepting station (STA).

MH/AO

- o "Reads" MSG from an APGM DQ and stores it in buffer
- o MSG is placed on a READ-AHEAD-Q and is "read" into MSG WORK AREA for that APGM, PREFIX is removed at this time
- o APGM processes message. APGM may return a processed response MSG to a STA or another APGM.

MH/AI

- o Processed MSG "WRITTEN" to buffer, MSG PREFIX & HEADER created
- o MSG Is:
    - Place on DES-Q associated with a STA. or
    - Place on DES-Q associated with another APGM.

MH/LO

- o STA SELECTION
    - MH/LO searches station Control Blocks (STCB) of the Line Control Block (LCB) node until there is no work to be done, i.e., the STCB subchain is searched while:
        + Output is queued for STA
        + STA is eligible
            - STA not in buffer delay
            - STA not being held
- o MSG read from a STA's DQ into buffer
- o MSG prepared for output transmition
- o MSG transmitted
- o Only when there is no output for a line, are input MSG processed by autopolling that line.

MH/LO Diagram (see Figure 2.)

The MH/LO diagram is a specific instance of the TCAM message flow diagram. It isolates the conditions of line input/output utilization and ignores APGM queuing, MH/AO, MH/AI and APGM message processing.

The flow of messages to and within MH/LO is as follows:

A message is transmitted to the Terminal Control Unit (TCU) from a terminal (in TCAM terminology a terminal is a station), the MH/LI logic of TCAM interfaces with the TCU hardware and queues the input message. After input message processing the resultant output message is placed on an output STA DQ associated with same line. Then MH/LO logic of TCAM, scans each line for STAs on that line which have output queued and are eligible to print. Output processing continues until there is <u>no</u> work (queued output messages) for eligible stations on that line.

Under these conditions the line output utilization can be such that no input messages are acknowledged, resulting in a line input lockout or unacceptable service for input terminal processing.

2.2 Control of Line Utilization

TCAM's BUFFER DELAY VALUE (BDV)

The parameter available to "tune" half duplex line utilization is the BUFFER DELAY VALUE.

BDV is that period of time a station on some line is not eligible to receive output. In effect, it is a time switch which controls the input/output selection algorithms of TCAM.

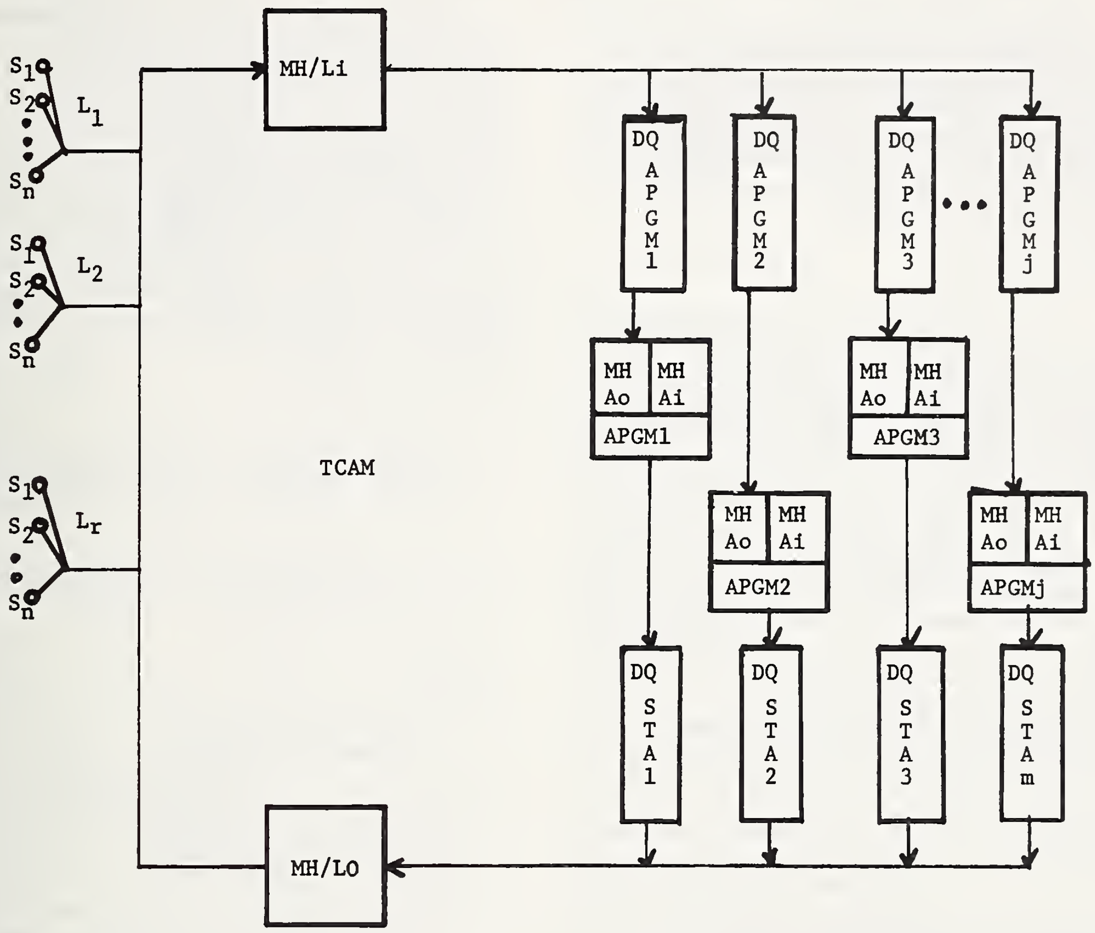


Figure 1. TCAM Message Flow

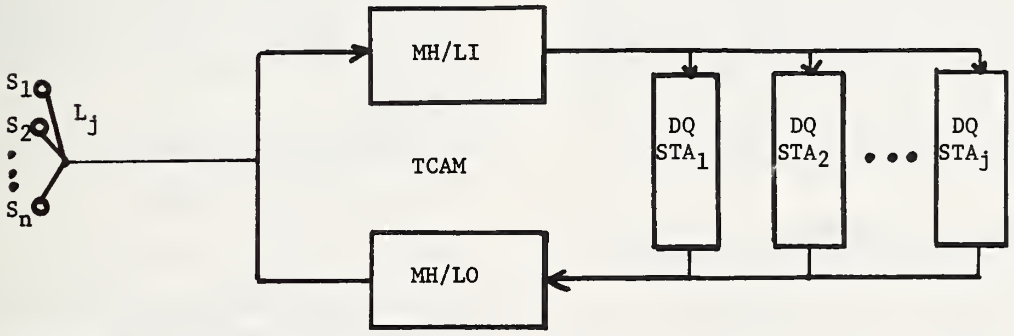


Figure 2. MH/LO Diagram

Therefore, BDV is a value which may be used to detain line output message processing in favor of line input processing. It can be viewed as that time required for a station to print a message and thus, the interval of time in which no attempt by TCAM is made to transmit the succeeding message to that station.

BDV Specification

BDV specification is not immediately obvious. The variables, print time, transmit time, CPU time (includes modem turnaround time) number of stations on a line and TCAM's Equal Priority Scheduling algorithm are all required elements factored into the decision of an appropriate BDV.

Default BDVs or mis-specified BDVs in the positive or negative direction can exacerbate existing conditions of line input lockouts or what the author designates as negative capacity (CPU capacity absorbtion with no correspondent effective work being performed). Such conditions have a degrading effect on the transaction flow of a business dependent on the effectiveness of its telecommunications system.

TCAM EQUAL PRIORITY SCHEDULING ALGORITHM AND BDV

Given an LCB and associated STCB Link List, the EQUAL PRIORITY SCHEDULING ALGORITHM will continuously search that list while output is queued for some station on the line and that station is eligible, see figure 3. A station is eligible if it's not observing its buffer delay or is not being held. Only after all output queued for that line is satisfied does TCAM honor input for that line. For the purposes of the following discussion, we will assume the station is never in a HOLD STATE.

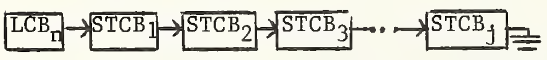


Figure 3. LCB Link List

Given the EQUAL PRIORITY SCHEDULING ALGORITHM and peak loading conditions line control block chain could loop on itself until all output queued for stations on that line is transmitted. Unless a judicious value for buffer delay is selected real time input requests from the field branch offices will be denied, moreover, invalid buffer delay values can result in negative capacity or underutilized printers.

The "EQUAL" in TCAM's EQUAL PRIORITY SCHEDULING ALGORITHM can be construed to mean: Accept input for a line and perform output for a line then go to the next line and do, until all lines are satisfied. Thus, lines are equally "flip-flopped" between input and output; however, the internals of the algorithm allows a disproportinate percentage of time expended in servicing the output component while input must wait. TCAM;s BDV can be implemented to bring the above condition back into an "equal" state.

The OBJECTIVE

Given the above problem description, we are now prepared to state our objective in controlling half duplex line Utilization:

The objective is to choose a BDV such that TCAM performs only one pass of an LCB chain for output.

The objective makes the following implication:

a. No spinning within an LCB chain occurs.

b. After one pass on an LCB chain do an autopoll of that line, (i.e., honor input for that line).

c. a. and b. above are performed as an ordered pair for each line in the telecommunications system.

The conditions under which half duplex input line lockout exists have been described; an objective which mitigates this condition has been found; and lastly, a mechanism is now required to meet our objectives.

3. A mechanism to control line utilization

Definition of Variables

Let

PBUF	= BYTE width of station hardware buffer
Po	= Station print speed in bytes per second
MSZE	= Maximum message size
Lo	= Line transmission speed in bytes per second.

And

Pm	= Message print time
Tm	= Message transmit time

3. A mechanism to control line utilization (continued)

Cm	=	Message CPU preparation time
N	=	Number of stations eligible for output and having output queued
k	=	The ratio of PM and (TM + CM)
D	=	BDV

Then

$$Pm = \frac{PBUF}{Po}$$

$$Tm = \frac{MSZE}{Lo}$$

$$k = \frac{Pm}{(Tm+Cm)}$$

For

Po	=	40 CPS (characters or bytes per second)
Lo	=	133 CPS
PBUF	=	MSZE = 500 characters or bytes
Cm	=	0.5 sec. (mean value derived from measurement)

We have

Pm	=	12.50 char/sec
Tm	=	3.76 char/sec
k	=	2.93

D, is to be derived.

Assumptions and Initial Conditions

a. Message size approaches or equals TC-241 hardware station buffer size of 500 characters

b. Station print speed is 40 CPS. This ignores a potential speed increase with the inclusion of TAB CONTROL CHARACTERS in the output message.

c. We are interested in peak loading conditions where output is queued and N is large.

d. Pm is 12.50 char/sec, Tm is 3.76 char/sec and k = 2.93

Development of the Buffer Delay Value

Let us make the initial observation that the period of time a buffered station is ineligible to receive the next output message is that period of time the station is busy printing the currently transmitted output message. Then

$$D = Pm = \frac{PBUF}{Po}$$

This observation is clear, the succeeding output message for a station should not be transmitted during the interval of time in which that station is busy printing the current message. If, in fact, the succeeding message is transmitted before its print time has expired then the station will reject the message (respond NAK) and TCAM "throws away" the candidate output message which is waiting transmission in an incore buffer.

Thus, the first approximation of BDV is PM. Given that BDV=PM does this value meet the constraints of the objective? For one station on a line the value clearly meets the objective (see Figure 4), this is also the case for two stations on a line. However, when $N \geq 3$ the message transmit time (Tm) plus the CPU message preparation time (Cm) to service $N \geq 3$ stations exceeds Pm and the first station on the chain now becomes eligible for output. TCAM's EQUAL Priority Queuing Algorithm senses the eligible station and upon completion of servicing all output requests for that line loops back on the same chain again for output service. Under these conditions, no input will be honored until all output is services or $N \leq 2$.

Then the limiting time value for the first approximation is, See figure 4.

$$L = \sum(Tm+Cm) \leq Pm.$$

Furthermore, it is now evident that BDV is not a constant value but must dynamically change as a function of N.

For lines have $N \leq 2$ and the above capacity characteristics we have the following special form of BDV

$$D = Pm$$

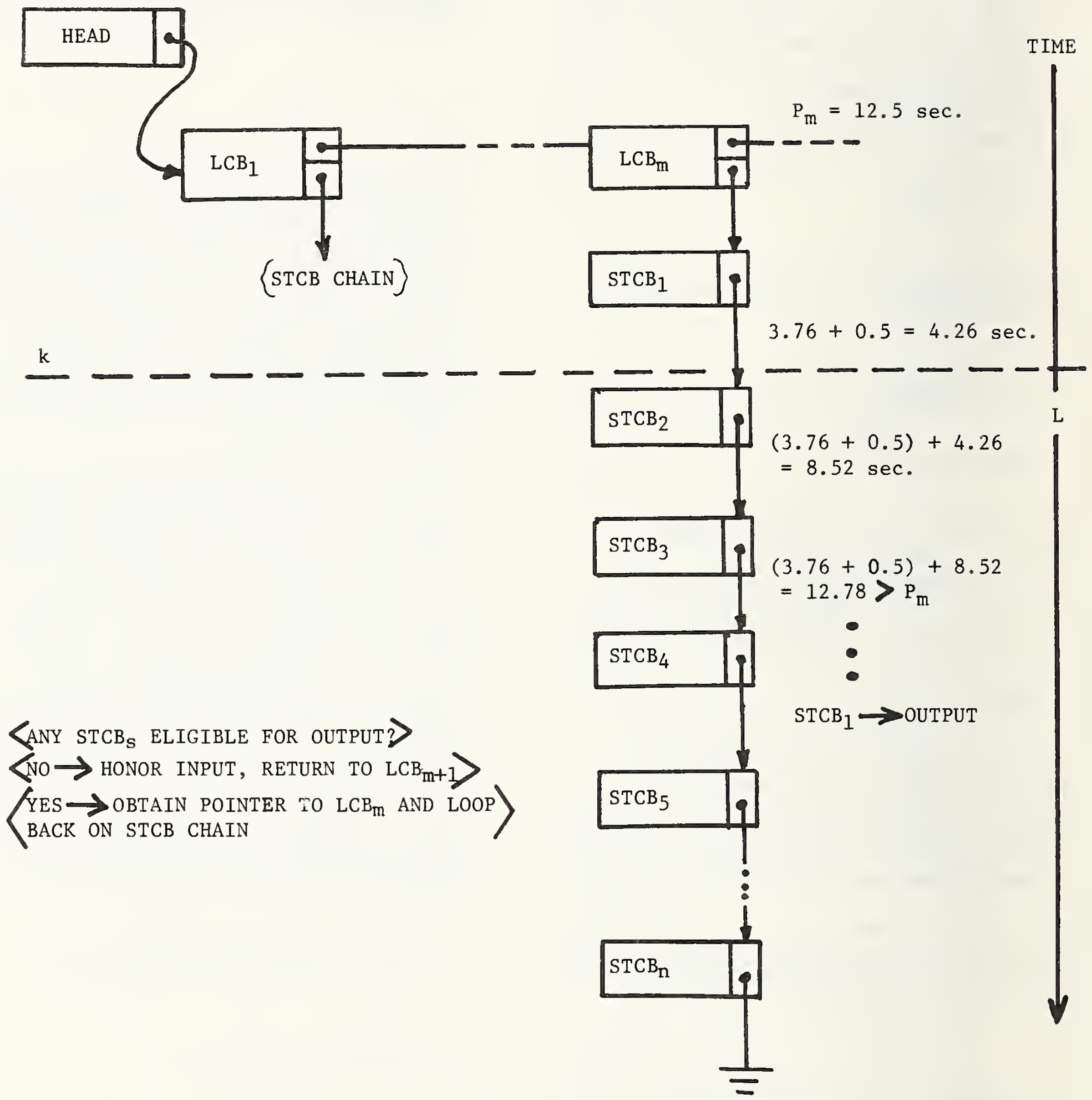

HEAD
TIME
P_m = 12.5 sec.
LCB_1
LCB_m
{STCB CHAIN}
$STCB_1$
3.76 + 0.5 = 4.26 sec.
k
L
$STCB_2$
(3.76 + 0.5) + 4.26 = 8.52 sec.
$STCB_3$
(3.76 + 0.5) + 8.52 = 12.78 > P_m
$STCB_4$
$STCB_1$ → OUTPUT
⟨ANY $STCB_s$ ELIGIBLE FOR OUTPUT?⟩
⟨NO → HONOR INPUT, RETURN TO LCB_{m+1}⟩
⟨YES → OBTAIN POINTER TO LCB_m AND LOOP BACK ON STCB CHAIN⟩
$STCB_5$
$STCB_n$

Figure 4. STCB Chain Processing

3. A mechanism to control line utilization (continued)

This special case equation may be generalized for varying capacity lines and stations and CPUs as follows

$$D = Pm, \text{ where } N \colon \frac{Pm}{(Tm+Cm)} = k$$

Notice, given the above line, station capacities the ratio of Pm and (Tm+Cm) is approximately 2.93, rounded down to integral stations is 2. This is the value we derived empirically and has an important function in deriving a dynamic BDV for instantaneous values of N.

Let $k = \frac{Pm}{(Tm+Cm)}$ where k is the boundary where looping occurs

And the number of integral station $N>k$ be

$(N-k)$

Then for each integral station increment$>k$ the time summed into the first approximation of BDV must be

$(N-k) * (Tm+Cm)$

Therefore, the instantaneous buffer delay value for some value of N is

$$D = Pm + (N-k) * (Tm+Cm)$$

$$D = Pm + N*Tm + N*Cm - k*Tm - k*Cm$$

$$D = Pm + N*(Tm+Cm) - k*(Tm+Cm)$$

Substituting $k = \frac{Pm}{(Tm+Cm)}$

$$D = Pm + N*(Tm+Cm) - \frac{Pm}{(\cancel{Tm+Cm})} * (\cancel{Tm+Cm})$$

$$D = \cancel{Pm} + N*(Tm+Cm) - \cancel{Pm}$$

$$D = N*(Tm+Cm).$$

Notice that in the simplification of the equation describing BDV that BDV is no way dependent on our intuitive first approximation; the speed of the print device, Pm. Print time vanishes from the equation and the somewhat unexpected results above falls out of the algebra.

4. TCAM DYNAMIC BUFFER DELAY ALGORITHM

A modification to add the DYNAMIC BUFFER DELAY ALGORITHM to TCAM is currently underway at The Hartford Insurance Group. Subsequent to the addition, a hardware line monitor measuring interpolling time will be used to measure the performance impacts of the addition.

This addition to TCAM will be an algorithm which dynamically assesses N for each line and assigns the BDV corresponding to N, to a station requesting output service. The algorithm is driven from the following table, see table 1. On this particular line, there is a maximum of $N \lesssim 17$ stations eligible for output service.

A statistical refinement could be coded into the algorithm whereby a mean message length could be obtained from sampling intervals. This would closely approximate Tm rather than assuming a worse case Tm where the message size approximates the maximum message length of 500 characters. The effects of this, however, requires further analysis.

Table 1. BUFF DELAY for $N \lesssim 17$

N	N·(Tm+Cm)
1	4.26
2	8.52
3	11.78
4	17.04
5	21.30
6	25.56
7	29.82
8	34.08
9	38.34
10	42.60
11	46.86
12	51.12
13	53.38
14	59.64
15	63.90
16	68.16
17	72.42

I am grateful to Charles Sarra and Terence Berinato of Aetna Life & Casualty for their patience and thorough descriptions, both verbal and handwritten of the inner workings of TCAM. In particular to Charles Sarra for his senior knowledge of Telecommunications Systems and Terence Berinato for his descriptions of the queuing delays inherent in TCAM/MCP message processing.

Thanks are also in order for Michael Hufman who built the logic for the dynamic BUFFER DELAY ALGORITHM, tested and imple-

mented it in The Hartford's teleprocessing environment. Mr. Hufman is currently performing line monitoring experiments with BUFFER DELAY turned on and off for comparative analysis.

Lastly, I wish to thank Tom Quick of The Hartford's Performance Unit and Ken Klein of The Hartford's Teleprocessing Unit for their critiquing and suggestions in the development of these ideas.

METHODOLOGY FOR PERFORMANCE EVALUATION AND CAPACITY PLANNING

Dr. Arnold O. Allen

IBM Systems Science Institute
3550 Wilshire Boulevard
Los Angeles, California 90010

In this tutorial we summarize a methodology taught at the Los Angeles IBM Systems Science Institute in a class called Performance Evaluation and Capacity Planning.

Key words: Analytic queueing theory models, capacity planning, performance evaluation.

1. Introduction

We are in agreement with John Spragins [1][1] that some of the literature in the performance evaluation area consists of durable nonsense in the sense of the three principles of durable nonsense, which are:

"First Principle. For every durable item of nonsense, there exists an irrelevant frame of reference in which the item is sensible."

"Second Principle. Rigorous arguments from inapplicable assumptions produce the world's most durable nonsense."

"Third Principle. The roots of most nonsense are found in the fact that people are more specialized than problems."

2. Overview

We have developed a practical methodology that can be applied in such a way as to avoid the perils of durable nonsense. The methodology has been successfully applied to real-world performance problems by people who are not technical giants. In this tutorial we will describe the methodology we call UMPV for (i) understand, (ii) model, (iii) predict, and (iv) validate. This is, of course, an iterative process. We will explain these basic steps and show how analytic queueing system models help us perform them. We will illustrate the process with several examples from Allen [2].

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

References

[1] Spragins, J., "Computer System Performance Modeling and Durable Nonsense," Proceedings of the Hawaiian International Conference on Systems Sciences, Honolulu, Hawaii, Jan. 1979.

[2] Allen, A. O., Probability, Statistics and Queueing Theory With Computer Science Applications, Academic Press, New York, 1978.

BENCHMARKING WITH REMOTE TERMINAL EMULATION

Thomas F. Wyrick
Directorate of System Evaluation
Federal Computer Performance Evaluation
and Simulation Center
Washington, DC 20330

and

Raymond E. Youstra
International Business Machines Corporation
Rockville, MD 20850

Remote terminal emulation is a benchmarking technique that uses an external, driver computer system to imitate the teleprocessing devices and device operators to be supported by, and to impose controlled workload demands on, the computer system being tested. Many operator and remote device characteristics and actions are represented precisely by the driver system. The driver system exchanges control and application data transmissions with the system being tested through operational data communication hardware and software. Remote terminal emulation also uses a monitor, external to the system being tested, to record on a log file certain aspects of the interaction between the driver system and the system being tested. Data reduction software produces various performance measures (e.g., turnaround time, response time) from the log file during and/or after the benchmark test. Practical experience has shown that remote terminal emulation can be used to satisfy many performance evaluation objectives, such as regression testing, stress load analysis, system integration, migration planning, acquisition evaluation, etc. Though originally developed for testing large host systems, this benchmarking technique has been used successfully by both vendors and users to test large and small hosts, front-end communications processors, packet switches, intelligent terminal systems, etc. Remote terminal emulation hardware and software are available from several sources.

This tutorial will introduce the concepts, terminology, and capabilities of remote terminal emulation, and will summarize the availability of remote terminal emulation facilities. The most widely used remote terminal emulator, IBM's Teleprocessing Network Simulator (TPNS), will be discussed in detail. The tutorial will describe the circumstances when remote terminal emulation is a cost-effective tool for managing existing teleprocessing systems. Recent Federal procurement regulations and guidance concerning when and how Government agencies should employ this benchmarking technique during ADP acquisitions will also be summarized. Copies of a Government handbook on emulation and an extensive bibliography will be distributed to attendees of the tutorial.

TUTORIAL OUTLINE

I. INTRODUCTION (Thomas Wyrick)

   A. Objectives of Tutorial
   B. Overview of Benchmarking
   C. Overview of Remote Terminal Emulation
   D. Availability of Emulation Facilities

II. TELEPROCESSING NETWORK SIMULATOR (Raymond Youstra)

   A. Architecture
   B. Capabilities
   C. Resource Requirements
   D. Script Language
   E. Installation/Execution
   F. Associated Tools

III. USE OF REMOTE TERMINAL EMULATION TO SATISFY NON-ACQUISITION OBJECTIVES (Raymond Youstra)

   A. Objectives
   B. Costs
   C. Benefits
   D. Example Use of TPNS

IV. USE OF REMOTE TERMINAL EMULATION DURING ADP ACQUISITIONS (Thomas Wyrick)

   A. Federal Procurement Regulations
   B. Remote Terminal Emulation Handbook
   C. Benchmarking Goals During Acquisition
   D. Procedural Guidance
   E. Standardized Specifications for Remote Terminal Emulation
   F. Example Use of Specifications During Acquisition

V. CONCLUSION (Thomas Wyrick)

PLANNING AND IMPLEMENTING REMOTE TELEPROCESSING SERVICES: MANAGEMENT PERSPECTIVES OF THE TSP

Robert L. DeMichiell, CAPT, USCG
Director, Computing Activities
United States Coast Guard Academy
New London, Connecticut 06320

and

Gerald L. Underwood, LCDR, USCG
Project Manager
USCG Research and Development Center
Avery Point, Groton, Connecticut 06340

This tutorial addresses a competitive negotiated procurement of remote timeshare services under the Teleprocessing Services Program (TSP). The guidelines provided by the Department of Transportation and the General Services Administration governing the U.S. Coast Guard have resulted in the implementation of timeshare services for the U.S. Coast Guard Academy. A final comprehensive report culminated a rather extensive effort.

It is the intention here to discuss many aspects of the procurement from the perspective of the personnel involved in the various phases of the process. The relevant issues will be examined and some reference literature on the general subject will complement the two main thrusts of the presenation: (1) to identify and clarify the procedures and options, and (2) to provide practical, locally-derived guidelines for future implementation of the program.

Although the guidelines are general in nature, they were derived from a detailed analysis of the events which recently were experienced by the authors. The completion of the negotiation for a five-year systems life procurement under TSP resulted in a savings in excess of half a million dollars.

Key Words: Teleprocessing services program; timesharing; competitive negotiated procurement; statement of work; remote teleprocessing; computer management.

1. INTRODUCTION

Most of the reference texts and the journal literature relating to computer system procurement focus on hardware and software acquisition. Other references deal with the analysis and design of user requirements. Some of the few guidelines available for procurement of commercial time-share services are provided by the federal government. However, these extensive procedures also provide several options for the user. These options are discussed from philosophical, procedural and political vantage points.

2. THE REQUIREMENTS

2.1 User Wants, Needs or Requirements

Sometimes an attempt to clarify this issue has either raised more questions than answers, or even more obscured the initial issues. In an age of justification, accountability, and quantification, some commentary is necessary to promote the sharing of solutions. The ongoing operational commitments must be met concurrently with time also devoted to the development and updating of new requirements. Different types of reqirements, in addition to a brief treatment of those developed for the specific environment here, are presented with regard to the implied time constraint.

2.2 The U.S. Coast Guard Academy - A Profile

In order to place in perspective the following more definitive presentation of procedures, people and problems, an overview of the Academy mission, faculty, students and total program is presented. Other settings probably contrast considerably, but the same questions of definitions of adequacy of computing and computer science probably prevail.

2.3 The Federal Computing Complex, Consultants, and Options

The federal government is a big user of computer resources, both inhouse and commercial services. Service institutions have increased significantly in recent years, and even suggest location of the equipment at the user site (facilities management). Therefore, updating, conversion techniques and decision-making become more complex. This dilemma is discussed and with relevance to the use of, and importance of (as appropriate) outside consulting services.

The general nature of the available options can best be described by some illustrative questions. Basic Agreement or Multiple Awards Schedule? SOW Requirements? Schedule? Definition of Responsiveness to RPF? Mandatory and Desirable Requirements "Mix"? Evaluation Criteria?

2.4 Protest Concerns and Commitments

The vendors are the experts in the procurement business. They complete several negotiations a year. How many have you been involved in lately? Fair treatment practices are discussed here, in addition to the confidentiality issues throughout the procurement action.

Someone has to develop the unique selection plan for your organization. A committee can't do it unless specific tasks are delineated and, hopefully, agree with the guidelines. This can be assured if the organization plans, staffs, and directs from the outset. A procurement team of individuals with quite diverse talents, expertise, and responsibilities are brought together by top-level management.

3. FROM RFP TO AWARD

General procedural comments on the statements of work, the TSP decision, the delegation of procurement authority, and a report of findings are summarized. This will cover the advertisement for bids and specific suggestions on work definition and vendor clarification and interpretations (the management of the negotiation sessions). Some mention is made of the administrative matters involved in establishing the capability demonstrations and benchmark validations.

4. CONCLUSIONS AND RECOMMENDATIONS

Some practical guidelines are summarized regarding all of the previous topics, and in addition, special attention is directed to two items: (1) writing styles and ambiguities, particularly in the definition of objectives and requirements, and (2) the final report to top management and the multitude of reviewing authorities. The presentation concludes with a response to a somewhat academic question (since TSP is a mandatory program) "Is it really worth it?" Our experience indicates that the answer is an unqualified" YES."

SELECTION AND EVALUATION OF INSTRUCTIONAL TIME-SHARING SERVICES (A TUTORIAL OUTLINE)

Richard T. Close
and
Raymond A. Kambeitz

U. S. Coast Guard Academy
New London, Connecticut 06320

The U. S. Coast Guard Academy has completed a multi-year procurement of instructional time-sharing services using the General Services Administration Teleprocessing Services Program (TSP). This fully competitive procurement included extensive technical and cost evaluations and adhered to Department of Transportation and Coast Guard directives for data processing contracts. The technical evaluation included an on-site operational capability demonstration with benchmarking. The final ranking of vendors used a point scoring system which included both the technial and cost analyses.

Key words: TSP; time-sharing; procurement; evaluation; technical analysis; benchmarking; cost analysis.

1. Background and Environment

1.1 Nature of Services

The U. S. Coast Guard Academy is an undergraduate institution which prepares young men and women for duties as officers in the Coast Guard. The curricula are technically-based and demand extensive computer support. In particular, software for graphics and engineering applications are crucial.

1.2 Procedures

The procurement was carried out using the Basic Agreement (BA) provisions of the Teleprocessing Services Program (TSP) of the General Services Administration. Additional regulations of the Coast Guard and the Department of Transportation also had to be considered.

2. Preliminary Assessment of User Needs

2.1 System Study

In-house and contractual studies had developed a reasonably complete picture of desired services. Hardware, software and systems support capabilities were identified.

2.2 User Profiles and Interviews

Current users had established a history of usage which was taken as a baseline for future activity. Potential users were interviewed to project utilization as services become available.

3. Development of the Solicitation Document (RFP)

3.1 Process

An iterative process was used to establish mandatory and desirable features. Rough drafts were circulated and comments returned. New drafts were made and recirculated. Novice users were aided by the GSA professional staff and the local user services group.

3.2 The Evaluation Plan

A point system was developed that provided a discrimination scale for both cost and technical evaluation. Since many technical features were included as mandatory items, the cost evaluation was given a higher overall weight in the evaluative process (a 75/25 ratio).

4. Operational Capability Demonstration and Benchmark

4.1 On-site Visit

An Operational Capability Demonstration (OCD) was specified in the solicitation document. A proposed agenda was submitted to each potential vendor. Members of a Technical Evaluation Team conducted the OCD.

4.2 Benchmark

A representative group of programs was developed and submitted to each vendor. These programs had to be run as a part of the OCD and also had to be demonstrated from the Academy.

5. Evaluation of Proposals and Award

5.1 Technical Evaluation

The technical evaluation included the OCD, benchmark and the assignment of technical points for desirable features. The benchmark was used to establish a relationship between the projected usage figures and the proposed services for each offeror.

5.2 Cost Evaluation

With the aid of the benchmark results, the Cost Evaluation Team calculated a systems-life cost. Present value discounts were used to establish an evaluated cost over the expected length of the contract. Cost points were awarded to potential vendors according to their relative evaluated costs.

5.3 Vendor Selection and Award

Cost points and technical points were weighted and combined to give a total point score for each potential vendor. The vendor with the most points was identified as the successful bidder and after various levels of approval had been obtained, a contract was awarded.

6. Conclusions and Recommendations

6.1 Results

The prospect of a long-term contract for instructional support should provide an element of continuity that was missing in the past. Users are more likely to commit time and effort to develop meaningful computer applications. The quality of service should also improve as the vendor is also more willing to devote resources for the longer term. In addition, from a financial stand point, the program was extremely successful. The competitive bid process, though long and involved, resulted in substantial savings.

6.2 Program and Future

The Teleprocessing Service Program is an effective vehicle for the procurement of time-sharing services. Although some of the detailed requirements were burdensome, there have been significant improvements in the program and with continued attention, TSP should become even better.

TUTORIAL ON BENCHMARK CONSTRUCTION

Helen Letmanyi

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

This tutorial will provide participants with a detailed overview of the benchmark construction process--the steps involved and the tools that can be employed to construct a benchmark. This tutorial is recommended for those who have an interest in constructing benchmarks for use in the competitive evaluation of vendor systems.

A brief review of the ADP system evaluation and selection process within the Federal Government will first be given to identify how the benchmark construction process fits into the total selection process. Next, the tutorial will discuss in a step-by-step fashion those tools and techniques which can be used to analyze existing workloads, project future workloads, and represent workload requirements via benchmarks. The importance of having definite objectives and goals prior to constructing a benchmark, as well as the need for benchmark validation and documentation will also be discussed. An outline of the topics to be covered in the tutorial follows.

TUTORIAL: BENCHMARK CONSTRUCTION

I. BRIEF SUMMARY OF BENCHMARK (B/M) CONSTRUCTION

THE B/M CONSTRUCTION PROCESS

STEP NO.	DESCRIPTION
1	Decide which workloads are to be represented.
2	Partition workload by distinct categories.
3	Quantify existing workload to be represented.
4	Quantify future workloads (for each workload category).
5	Determine which portions to represent.
6	Choose B/M programs.
7	Represent selected workload via B/M programs.
8	Fine tune B/M on present system.
9	Validate B/M and document LTD.

II. STEP 1. DECIDE WHICH WORKLOADS ARE TO BE REPRESENTED

A. Define objectives and system life.

B. What workload to represent: present, future, both.
Define time-frame to be represented (e.g., peak functions, peak resources, min-max peak, average workload, period of day).
Review scheduling practices, deadlines
Production vs. user-oriented.
Selected categories should reflect future objectives.
Decide on workload categories and sub-categories:

Batch:	Initiated at the central site
	Remote batch
	On-line initiated
On-line:	Interactive program development
	Text processing
	Transaction processing
	Data base query/update
	Data entry
	Interactive graphics
	Production jobs
Real-time:	Data collection
	Command and control
	Hybrid processing

Decide on performance criteria:
e.g., response time, turnaround time, throughput rate, elapsed time.

III. STEP 2. PARTITION WORKLOAD BY DISTINCT CATEGORIES

Importance of system independent categorization.
Workload levels:
  Level 1: Prime vs. non-prime
  Level 2: Processing mode (batch, on-line, real-time)
  Level 3: Functional grouping
           ADP operations
           Resource usage
Recommended approach.
Pros and cons.

IV. STEP 3. QUANTIFY EXISTING WORKLOAD TO BE REPRESENTED

A. Quantifying processing requirements.
   1. Determine sources of data.

WORKLOAD CATEGORIES

	LEVEL 3			LEVEL 2	
SOURCES OF DATA	By Resource Usage	By ADP Operations	By Functional Groupings	Batch	On-line
Accntg. Log	x	x	x	x	x
Software Monitor	x			x	x
Hardware Monitor	x			x	x
User Survey		x	x	x	x
Terminal Script		x	x		x
Hard Copy Output	x	x	x	x	x

   2. Determine what data is available vs. what data should be collected.
      Short term collection effort (cost, benefit).
   3. Analyze Data.
      a. Accounting Log Data:
           CPU usage
           Memory used
           I/O activity (I/O count)
           ADP activity (compile, execute, etc.)
           Unit device activity, volume
         Analysis Techniques:
           Clustering
           Joint probability density function
           Analytical methods
      b. Software Monitor Data:
           CPU usage
           Problem state
           Channel activity
           CPU/Channel overlap
         Analysis Techniques: Data reduction packages

c. Hardware Monitor Data:
    CPU usage
    Problem state
    Channel activity
    CPU/Channel overlap
  Analysis Techniques: Data reduction packages
d. User Survey: User satisfaction, user requirements
  Analysis Techniques: Workload Mapping (USDA)
e. Terminal Script
f. Hard copy output:
    CPU usage
    Memory used
    Elapsed time
    ADP operation (compile, execute, etc.)
    Unit device activity volume
    Functional grouping

4. Within each category, determine relative contribution of each sub-group.
5. Identify applications with special processing requirements:
    Large programs
    Time constraints, job dependencies
    Security restrictions

B. Quantify Service Requirements.
  1. Determine service measures; e.g., response time, turnaround time, throughput.
  2. Determine sources of data.
    a. Accounting Log:
        Connect time
        Elapsed time of the session
        Turnaround time
    b. Hardware Monitor: Response time
    c. Software Monitor:
        Response time
        On-line commands usage
        Number of concurrent users
    d. User Surveys: User defined service requirements
    e. Terminal Script: Response time/think time
    f. Hard Copy Output: Turnaround time, response time, elapsed time
  3. Determine what data is available vs. what data should be collected.
  4. Analyze Data.
    Recommended practices.

V. STEP 4. QUANTIFY FUTURE WORKLOADS (FOR EACH WORKLOAD CATEGORY)

Workload forecast (up to X years).
Alternative techniques should be used in future workload quantification for each workload category; e.g.,
    Trend (regression) analysis
    Extrapolation of clustering data
    User surveys
    New missions
Determine confidence interval on forecast

VI. STEP 5. DETERMINE WHICH PORTIONS TO REPRESENT

Having quantified the workloads in Steps 3 and 4, what to represent can be determined; e.g., time-frame, categories.
Determine augmentation point(s), or decide if vendor does.
Determine number of distinct benchmarks.

VII. STEP 6. CHOOSE B/M PROGRAMS

Represent portions identified in Step 5 and quantified in Steps 3 and 4.
Real programs:
  Pros and cons
Synthetic programs: Resource-oriented
                    Function-oriented
  Pros and Cons
Combination of real and synthetic programs
On-line scenario development
Data base generation
File structures
Important characteristics of B/M programs:
  Simplicity
  Representativeness
  Transportability
  System independence
  In higher-level standard language
Recommended practices

VIII. STEP 7. REPRESENT SELECTED WORKLOADS VIA B/M PROGRAMS

Formulate B/M mix(es).
The B/M mix must represent all important aspects of the real-workload:
  The number of jobs, the number and type of terminals
  The type and volume of ADP operations
  Job dependencies
  Variations in ADP operations
  Priorities
  Security requirements
Decide on live terminals vs. RTE.

IX. STEP 8. FINE TUNE B/M ON PRESENT SYSTEM

Exercise timed B/M test on present system.
Establish proper relationships among selected categories of B/M programs and on-line activities to the real-workload:
  Number of iterations of selected programs and on-line activities
  Volume of data
  Define B/M mixes (timing)
  Determine B/M duration

X. STEP 9. VALIDATE B/M DOCUMENT LTD

A. Validate B/M.
   Run B/M mix(es) on other systems to validate B/M:
   - Timing
   - Auditability
   - Portability
   - Repeatability
   - Modifiability

B. Be sure B/M is well documented and in standard higher-level language.
   - Brief description of each program.
   - Block-diagram, showing I/O file origination/destination.
   - File descriptions.
   - Program conversion constraints.

C. Document LTD.
   - Hardware configuration requirements.
   - Software configuration requirements.
   - Functional test.
   - Performance test (B/M mixes to run).
   - RTE/live terminals.
   - Validation data requirements.
   - Different configurations, optimizing techniques.
   - Credits.

USING ACCOUNTING LOG DATA IN PERFORMANCE REPORTING

James P. Bouhana*

Computer Sciences Department
Purdue University
West Lafayette, IN 47906

This tutorial summary outlines several topics pertinent to using a computer system's accounting log file as a basis for performance reporting. Some major topics are a log's organization and contents, the types of reports and displays which can be generated, and the problems encountered in using log data. It is concluded that although accounting logs have an understandably principal orientation toward accounting, they can be effectively used for performance reporting and for operations reporting as well.

Keywords: Accounting logs; performance evaluation; workload characterization.

1. Introduction

Although the accounting logs which exist for most multiprogrammable computer systems have been used in the past largly for user billing purposes, they are increasingly being used for performance analysis. Three factors contributing to this phenomenon are:

(a) Accounting logs are already there—they are maintained by a computer's operating system and require no software or hardware implementation effort by an installation,
(b) Accounting logs are always "on"—they are continously maintained and thus do not require any special action to activate them just before a performance problem develops,
(c) Accounting logs provide a central repository for performance data, which in addition to resource usages, may also include data from software samplers, event-driven trace routines and hardware monitors.

* This work was prepared in part during the summer of 1979 while the author was employed in the Systems Performance Analysis group of Digital Equipment Corporation in Maynard, Mass.

A significant additional attribute of accounting logs is their usefulness for operations management as well as for performance reporting and for billing. Such tripartite usage of a single data base is efficient, and there is no other readily-available data base having the same multiple-use characteristics. Although our prncipal concern is admittedly with performance reporting, we contend that any data added to the log to enhance performance reporting invariably also enhances an installation's capability for effective operations management and also for equitable billing policies.

2. Uses of Data

The set of performance analyses for which accounting log data are useful is varied. The fact that accounting logs contain data about individual programs permits specific-program analysis as well as system-wide analyses. Some examples of possible analyses are:

(a) User program optimization,
(b) Utility software optimization,

(c) Workload characterization,
(d) Operating system parameter adjustment,
(e) Configuration sizing and capacity planning.

Additional uses for log data lie in the realm of general performance "tools": methods which are not performance objectives by themselves but which are used in satisfying an objective. Some of these tools, which may also be used with data collected from other sources (such as a software monitor), are:

(f) Queueing network models,
(g) Time series analysis,
(h) Performance data bases,
(i) Kiviat graphs and other graphical displays of system-wide performance.

3. Log Organization

Basically logs are chronological records of significant events, but the manner in which the record is maintained varies. The two most common log structures are:

(a) Chronological by event occurrence time (e.g. program initiation/termination, file open/close, operator message, etc), with all events for different programs merged,
(b) Chronological by program termination time, with all events pertaining to a program ordered chronologically within a packet of log records associated with each completed program.

Some systems combine the two structures. For example, program initiation records may logged at the time of initiation, but all other records pertinent to a program may not be logged until program termination time.

4. Log Contents

Accounting logs vary widely in both the data which they record and the level of detail at which the recording takes place (e.g. the program level, the job level, the system level). Some categories of log data and examples of specific contents within each category follow:

(a) Program data: initiation and termination time, total service times for each resource used (e.g. CPU, tape, disk), memory used, I/O and paging counts, account and user identification,
(b) Job data: same items as for programs, but they may be specific to only the job itself (exclusive of resources used by its contained programs) or they may be totals for both the job and its contained programs (as in the case where separate process data are not recorded),
(c) File data: time of file open or close, device type, record and block size, access method, number of buffers, transaction counts, total device busy time,
(d) System data: system configuration, software parameters, checkpoint records, device errors, resources used for overhead activities,
(e) Other: log structure data (links and pointers), software sampler data, hardware monitor data.

5. Reports

Accounting log data can be used to generate tables of system-wide summary information as well as listings of all completed programs and their resorce usages, arranged in a variety of useful ways. The contents of the summary information and some ways of organizing reports are:

(a) Summary information, stating number of program completions, average multiprogramming level (MPL), average CPU utilization, average per-program CPU time, I/O operations, page faults, and memory size; number of system reboots,
(b) Reports sorted by program termination time, program initiation time, program name, user name, and account name (or number),
(c) Reports sorted by resources used by programs, such as CPU time, I/O operations, memory size, page faults, devices used, CPU rate, and elapsed time,
(d) A report sorted by the frequency with which programs are executed,
(e) Subset reports, listing only those programs contained within a restricted category such as a specific program name, user name, or account.

6. Plots

Plots visually display system-wide operating characterics. The two most common types of plots are time series and histograms, although scatter diagrams and other elaborations are also used. Anomolous behavior (indicated by "spikes") noted in plots can often be traced to a specific program or group of programs by cross-referencing with the reports. For problems noted in time series, the chronological reports are referenced; whereas problems noted in histograms necessitate referencing a report sorted by resource usage values. Some categories of plots follow:

(a) Resource usages: histograms of CPU time, I/O time, I/O counts, page faults, and memory usage per program, as well as elapsed time and CPU rate,
(b) System wide activity: time series of CPU and I/O channel utilization, as well as system-wide memory usage,
(c) Mix activity: time series of minimum and maximum MPL within each time interval, as well as number of program initiations and program completion rate,
(d) MPL-related plots: histograms and graphs of MPL frequency, MPL elapsed time; CPU, I/O, and memory utilization by MPL, program completion rate by MPL, and average program elapsed time by MPL.

7. Problems Areas

Since accounting logs historically were not designed with performance reporting in mind, they sometimes lack the precision and completeness that a performance analyst would want. Some areas which either inhibit or prevent accurate reporting are:

(a) Coarse system clock resolution for reporting CPU time usage,
(b) No separate accounting for time spent servicing interrupts,
(c) Non-visibility of system routines (e.g., program loading, file maintenance, spooling),
(d) Reporting peak rather than time-weighted average memory usage,
(e) Lack of genealogical information for spawned programs.

8. Conclusions

Even though accounting logs may not be the most perfect data base to use for performance reporting, they can and have been used successfully in performance contexts. As noted before, no other data base has the attributes of ready availability and of applicability to the three areas of performance, billing, and operations.

In the Executive Summary of the 1973 NBS/ACM Workshop on Computer Performance Evaluation, it was stated that "Improved accounting systems and associated analysis software are a major national CPE need." Since then, vendors and users alike have become increasingly aware of the performance-related users of acounting data, as evinced by the proliferation of accounting log analysis packages. The trend for the future seems to be greater accuracy and completeness of accounting log data combined with greater usage of such data for performance analyses.

An Informational Model for the ADP Manager

Wayne Douglas Bennett

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

The object of this discussion is to suggest a framework within which a meaningful set of computer performance measures can be defined. Computer performance management (CPM) may be viewed as a set of optimizations that can be derived from an analysis of the planning and control problems typically facing ADP Managers. The discussion focuses on these problems, generating some of the more important optimizations. Determination of the parameters required to perform these optimizations results in a preliminary list and categorization of desirable CPM measures. Emphasis is placed on long range planning and maximization of organizational benefit from the ADP resources.

The parameters of the long range optimization problem include resource cost, derived income (or cost reduction) from a proposed workload, and queue costs. Selection of an appropriate configuration, an appropriate portfolio, and reasonable performance thresholds are thus expressed in fiscal terms and yield a determination of net benefit to the organization. The four resulting categories of measures: workload, resources, performance, and budget provide the starting point for a proposed list of performance metrics to be discussed in a follow-up CPEUG workshop.

NBS-114A (REV. 9-78)

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET	1. PUBLICATION OR REPORT NO. NBS SP 500-52	2. Gov't. Accession No.	3. Recipient's Accession No.
4. TITLE AND SUBTITLE Proceedings of the Fifteenth Meeting of the Computer Performance Evaluation Users Group (CPEUG)			5. Publication Date October 1979
			6. Performing Organization Code
7. AUTHOR(S) James E. Weatherbee, Editor			8. Performing Organ. Report No.
9. PERFORMING ORGANIZATION NAME AND ADDRESS NATIONAL BUREAU OF STANDARDS DEPARTMENT OF COMMERCE WASHINGTON, DC 20234			10. Project/Task/Work Unit No.
			11. Contract/Grant No.
12. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS *(Street, City, State, ZIP)* Same as No. 9.			13. Type of Report & Period Covered Final
			14. Sponsoring Agency Code

15. SUPPLEMENTARY NOTES

Library of Congress Catalog Card Number: 79-600123

☐ Document describes a computer program; SF-185, FIPS Software Summary, is attached.

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 Fifteenth Meeting of the Computer Performance Evaluation Users Group (CPEUG 79) held October 15-18, 1979, in San Diego, California. With the theme "The Expanding Scope of CPE," CPEUG 79 focused on changes in CPE techniques that will occur in an era of increased use of distributed processing techniques. The program was divided into two parallel sessions with one session devoted to the presentation of technical papers on previously unpublished work and the other devoted to tutorials and case studies. The technical papers fall into one of two general categories, (1) application of CPE in installation management and (2) methods and tools of CPE as a technical discipline, and are presented in the Proceedings in their entirety. Summaries of several of the tutorials and case studies are also presented in 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)*

Computer performance evaluation; computer performance measurement; computer performance prediction; computer system acquisition; conference proceedings; CPEUG; CPE in auditing; installation management; hardware monitoring; on-line system evaluation; prediction methods; queuing models; simulation; software monitoring; workload definition.

18. AVAILABILITY [✓ Unlimited]	19. SECURITY CLASS (THIS REPORT) UNCLASSIFIED	21. NO. OF PRINTED PAGES 240
☐ For Official Distribution. Do Not Release to NTIS		
[✓] Order From Sup. of Doc., U.S. Government Printing Office, Washington, DC 20402, SD Stock No. SN003-003-02118-1	20. SECURITY CLASS (THIS PAGE) UNCLASSIFIED	22. Price $5.50
☐ Order From National Technical Information Service (NTIS), Springfield, VA. 22161		

USCOMM-DC

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)