TR-81-011

PERFORMANCE EVALUATION FOR SMALL BUSINESS COMPUTERS

by

William P. Wilson, Jr.

A Thesis submitted to the faculty of the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Master of Science in the Department of Computer Science.

Chapel Hill

1981

Approved by:

Peter Calingaert
Adviser

Connie U. Smith
Reader

Ervin M. Davison
Reader

WILLIAM P. WILSON, JR. PERFORMANCE EVALUATION FOR SMALL BUSINESS COMPUTERS.

(Under the direction of DR. PETER CALINGAERT)

ABSTRACT

This thesis explores the technical discipline of computer systems performance evaluation as it relates to the special needs of computers for small businesses. Topics include: a brief overview of the technical aspects of performance evaluation, especially as they relate to small business systems; a discussion of the nature of small business systems; and a description of a project conducted to develop benchmarks for the evaluation of the performance of small business systems.

TABLE OF CONTENTS

FIGURES

PREFACE

As a programmer on a project developing system software to support small business applications, I have dealt with the problem of trying to squeeze maximum functional utility from a small computer while maintaining adequate performance. The project initially failed to meet its performance goals. The failure was largely due to a lack of concern for performance issues in the early stages of development. The programmers on the project had no formal exposure to performance evaluation and did not consider the performance impact of many design decisions.

Because of the problems that I encountered in coaxing performance from my own software, I resolved to learn something about the discipline. This thesis serves as a focus for that effort. It is an attempt to fill a personal professional need, to complete a masters degree already agonizingly long in the finishing, and to present some information in a way that will aid other computer professionals.

I would like to thank my adviser Dr. Peter Calingaert for all of his assistance in preparing this thesis. I also appreciate the help of the other members of my committee,

Dr. Connie Smith, and Mr. Erwin Danziger who provided valuable input.

I would like to acknowledge the assistance of my employer, Data General Corporation, which provided material resources for this effort. Thanks go also to my co-workers who provided the benifit of their experience, advice on technical matters, and moral support.

Chapter 1

INTRODUCTION

Computer software, to be acceptable to its users, must measure up to certain standards, or design criteria. These criteria include functional adequacy, aesthetics and other human factors, cost, reliability, and performance. This document is concerned with performance. Specifically, it is concerned with the relationship between performance and one of the most rapidly growing segments of the computer marketplace, that area referred to as "small business systems" (SBS). Although the problems of performance that might be encountered in systems used to run small business applications are not unique, they are important. More than any other computer user, the small businessman is concerned with getting maximum value from his data processing equipment. His livelihood depends on it.

Throughout this document the term "system" or "computer system" will be used to refer to a combination of a computer (CPU and related hardware), system software (operating system, compilers, utilities, etc.), and applications software (financial programs, etc.). A small business system is a system based on a small computer and used primarilly for business applications.

The remainder of this document presents a broad, cursory overview of both performance evaluation and small business systems. In addition, it describes a project conducted to construct a representative model of an SBS workload and to examine the composition of that workload. The text focuses on practical aspects of performance evaluation; approaching the subject from a "how to do it" point of view.

The thesis is organized into four chapters, of which this is the first. Chapter two is a brief survey of performance evaluation. Topics covered include performance evaluation studies, workloads, measures of performance, and performance evaluation techniques such as simulation and measurement.

Chapter three explores the definition of small business systems. Topics include a classification of small business systems by hardware configuration and price, a description of systems software available with small business systems, and a discussion of the various areas of application for small business systems.

The fourth chapter describes a project undertaken at Data General Corporation to develop a set of benchmarks for small business systems. In addition, data gathered as part of the benchmark validation process are used to support some observations on the general nature of SBS workloads.

The thesis concludes in chapter five with a reiteration of the goals of the document and some comments on questions raised but not answered during the preparation of the thesis.

Chapter 2

AN OVERVIEW OF PERFORMANCE EVALUATION

This chapter presents a cursory overview of some major topics in performance evaluation. The objectives are to provide a context for further reading, and a basis for the discussion of performance evaluation relative to small business systems.

The information in this chapter is drawn from a survey of relevant literature. Specific attention is paid to low-cost techniques that could be applied to small interactive systems. Readers desiring more complete coverage should consult the textbooks by Svobodova (1976) and Ferrari (1978). For ongoing coverage of topics in performance evaluation the interested reader can consult almost any of the general computer science journals. Specialized coverage is provided in Performance Evaluation Review, a publication of the ACM Special Interest Group on Measurement and Evaluation (SIGMETRICS); in Simuletter, published by the ACM Special Interest Group on Simulation (SIGSIM); in Performance Evaluation published by North-Holland; and in the proceedings of various conferences sponsored by SIGMETRICS, SIGSIM, the Computer Performance

Evaluation Users Group (CPEUG), the Computer Measurement Group (CMG), and the National Bureau of Standards (NBS).

The text first presents the major types of evaluation studies, discussing the reasons for performing each type. The second section reviews basic techniques for evaluating performance with cost and accuracy as primary considerations. The major topics covered in this section are performance measures, methods of evaluation, and workloads. The third section discusses performance evaluation of small business systems, relating the evaluation of SBS's to that of large systems.

2.1 Performance Evaluation Studies

Performance evaluation is the subdiscipline of computer science that is devoted to the analysis and improvement of the performance of computer systems. Performance evaluation is generally done to provide information to solve a problem. There are three main classes of problems that utilize performance information in their solutions: comparison problems, where the performance of two or more computer systems are compared; analysis problems, where the performance of a single system is scrutinized; and performance projection problems for systems not available for analysis. These three classes of problems serve as foci for studies of computer systems performance.

Comparison studies of the performance of different computer systems are used in a variety of situations. The most noteworthy is the so-called "selection" problem, which involves contrasting the performance of different computer systems to determine which provides the best performance. This type of study is most often used in competitive purchase or lease situations, where a potential customer is comparing different vendors' systems. Note that performance evaluation should, in most instances, be a secondary criterion in selection, functional adequacy and cost being prime considerations. Dujmovic (1979) discusses the role of performance evaluation as part of a formal process of evaluation for selection. Gay (1980) describes a selection study where a benchmark was developed to compare the performance of several interactive systems.

Comparison is also used in performance improvement problems to track the actual changes in system performance that occur because of changes in system implementation or hardware configuration. A well established performance evaluation methodology can give a good description of performance changes over the development phase of a new product.

Analysis studies evaluate the performance of a single system. The goal of performance analysis is to find ways to improve the performance of the system. A system under analysis may be a new system being developed, or an

existing system that is being adjusted to perform optimally for its workload. Performance analysis can detect system bottlenecks and other performance problems before they become costly drains on system resources. By analyzing the performance characteristics of a system under development, problems in performance can be detected early and solved.

Projection studies predict the performance of systems, by using a knowledge of the structure of the system and its probable workload. Projection studies conducted as part of the design of a new system allow for early decisions with regard to marketability and may guide development away from non-productive paths, thus making better use of expensive development manpower [See Dowdy et al. (1979), Sangunetti (1979), Smith and Browne (1979)].

Another major use of projection is in capacity planning. Larger computer installations project the growth of their workloads, study the effects of that growth on performance and plan changes in the system hardware and software needed to maintain performance. Lo (1980) describes several experimental case studies in capacity planning.

2.2 A General Approach to Performance Evaluation

The approach to performance evaluation is essentially that of problem solving. The performance analyst first determines the objectives of a study (comparison for

selection, analysis for improvement, projection for development planning, etc.). He then determines what data are needed to carry out the objectives and designs experiments to produce that data. He must then determine the most economical method of conducting the experiments, construct and install appropriate tools, perform the experiments, and then interpret the results. This process is, of course, not nearly as clear-cut as it sounds. Information acquired at any step may make it necessary to repeat previous steps and gather more or different information.

2.2.1 Performance Measures

In section 2.1 the reasons for performance evaluation were discussed. These reasons lead directly to the goals of any performance evaluation study. It is important that the goals of a study be well defined before embarking, for it is very easy to become sidetracked on interesting but irrelevant issues, or to become enthralled with tool building. The first step in a performance evaluation study is a determination of goals. The second step of a performance evaluation study is the determination of the data that will describe the performance of the system. These data are called "measures" or "indices" of performance. There are three types of performance measures: responsiveness, productivity, and utilization.

Responsiveness is a measure of the quickness of the system in processing commands. Responsiveness is generally measured by "response time" in interactive situations and "turnaround time" in batch situations.

Response time is the time necessary to process a transaction. A transaction is loosely defined as a useful, measurable unit of work. The response time of an interactive transaction is the interval measured from the keystroke that dispatches the transaction for processing, to the point at which the system is ready to receive another command.

Absolute measures of response time may be useful in comparing the relative speed of two systems, but it is the quality of the response time that is of interest to computer users. Response-time quality is determined by several factors including response-time variability, and the users' perception of the complexity of the transaction. Schneiderman (1980) reviews several recent studies in this area.

The traditional measure of responsiveness in batch situations is turnaround time, the elapsed time from the submission of a job to the time at which the output is available to the user.

Productivity or throughput is the measure of system performance that indicates the amount of work that a

system is able to do in a given period of time. It is generally expressed in instructions per second, transactions per hour, or some other work:time ratio.

Utilization is the efficiency with which a system uses its resources. It is generally measured as the percentage of time that a resource is in use relative to the time that it is available for use. Utilization is used for bottleneck detection by comparing the utilizations of various key resources (such as CPU time and I/O channel time). If one resource is being used much more than another, that resource may be a bottleneck preventing cost-effective use of the entire system.

The foregoing measures are strongly interrelated. A study might examine the effect on throughput of workload changes that increase the utilization of a particular resource. Another commonly studied interaction is the relationship between throughput and response time.

As an example, consider a system that can process up to 200 transactions per hour while maintaining a response time under two seconds. If more terminals are added the system can process 400 transactions per hour, but the response time goes to ten seconds. The productivity of the system is higher, but at the expense of degraded response time. In this situation, one would question the desirability of more productivity.

2.2.2 Methods for Gathering Measures

Once the goals of a performance evaluation study have been determined and proper measures selected, the actual technique for producing the measures may be considered. There are three techniques for producing performance measures. They are distinguished by the way that they represent the system to be measured -- the "system model." The three techniques are analytic modeling, simulation, and measurement or experimentation. These techniques vary in their cost, applicability, ease of use, and accuracy. The choice of which one to use depends on the availability of the real system, the urgency of the situation, the resources available in both manpower and money, and the degree of accuracy required.

Analytic modeling uses a mathematical formulation as the model of the system to be evaluated. The workload is represented as parameters of that formulation and the measures are produced by solving either analytically or numerically. Analytic modeling is, in general, the cheapest form of performance evaluation. It is also the least accurate and requires mathematical skills not usually present in small business environments. It is frequently used in research and in first-approximation feasibility studies for advanced development.

There are two types of analytic models: deterministic and probabalistic. Deterministic models express the system

and its workload as a set of equations that may be solved numerically or analytically. The main problem with deterministic models is that of representing complex, dynamic systems in a formulation that is both accurate and mathematically tractable.

Probabilistic models relate the inputs and outputs of a system probabilistically. In one form of probabilistic model, the Markov model, the system and its workload are expressed as a state-transition system with probabilistic relationships between the states. The most commonly used form of Markov model is the queueing model in which the system is represented as a set of service centers connected by queues. Requests are serviced and moved through the system probabilistically. Such a system may be solved numerically to produce performance measures.

For a more complete discussion of analytic modeling see Svobodova (1976) chapter 3, and Ferrari (1978) chapter 4. Some recent practice oriented studies of the use of analytic modeling include Dowdy et al. (1979), and Kienzle and Sevcik (1979).

Simulation is another frequently used technique that can give performance information at a much lower cost than actual implementation and measurement. Simulation implements a model of the system that actually mimics the important activities of the system. Stimuli generated by the workload model drive the simulator, which generates

information about the system model's activity. This information can then be used to predict how the actual system would behave under similar circumstances. There are several commercially available software packages for simulation (e.g. GPSS). For more complete coverage see Svobodova (1976) chapter 5, and Ferrari (1978) chapter 3. Recent papers by Unger and Parker (1979) and Sanguinetti (1979) describe techniques for combining projective simulation and systems design.

Measurement or experimentation is a frequently used technique in performance evaluation. In measurement studies the system serves as its own model. Data-gathering instruments are placed in the system, a workload is run, and selected aspects of performance are measured. Measurement is often used in selection problems, where a standard workload (or benchmark) is run on several different systems in order to compare their performance. Another common use of measurement is in improvement studies where a system's performance is measured with the objective of finding improvements.

There is a considerable body of literature on measurement. General concepts and techniques are summarized in Svobodova (1976) chapter 6, and in Ferrari (1978) chapter 2. A good example of a generalized measurement system is presented by McDaniel (1977).

In any exercise in performance measurement, the measures to be gathered determine the tools. Tools may vary in complexity from simple timers built into a program to complex event-trace recording mechanisms.

All tools have certain common characteristics. They are all made up of three parts. Data are gathered by a "sensor" or "probe" part, which is placed in the system to be measured and detects activity of interest. The probe feeds its information to a "transformer" part, which converts it into usable form. The transformed data are then passed to an "indicator," which makes it available to the human user. Note that these three parts represent logical, not necessarily physical, divisions of function. In the actual implementation of a tool, the parts may be indistinguishable.

When considering tools, the overall objectives of the measurement study should be kept in mind. Tools vary in their cost, both to build and to use. Major differences may also be found in applicability, scope, resolution, and accuracy. Implementors of performance evaluation tools should pay particular attention to a characteristic called interference, also called measurement artifact, which is the tendency of a tool to affect the performance of the system that it is measuring. A tool causing interference can invalidate any measurements that it produces unless the interference is controlled statistically.

Tools are divided into two general classes based on whether they are implemented using hardware or software.

Hardware tools are electronic devices connected by wires to key points in the electronics of the system to be evaluated. They record pulses or other electronic manifestations of the behavior of the subject system. The transformer part of such an instrument consists of amplifiers or other electronics. The indicators may consist of pulse counters or graphic recorders. A common technique is to use a micro or mini computer as a transformer with a CRT or printer to display the results. The microprocessor can even accumulate the raw data for later analysis and can be programmed to do almost any type of pre-reduction desired.

Hardware tools generally produce little or no interference. With the use of microcomputers they are not much more expensive than software. One limitation of their application is the difficulty of detecting some of the more complicated higher-level system events with them. Some common uses of hardware measurement tools are to monitor I/O channel and CPU activity, and to collect memory address and instruction traces.

Software tools are implemented as routines placed in the software of a system to be measured. Software tools are easy to install by a programmer with some knowledge of the system to be monitored. They are extremely flexible and can

detect events on almost any level. It should be noted, however, that software tools, if not used carefully, can introduce a great deal of interference; thus experiments using them should be carefully controlled.

2.2.3 Modeling the Workload

In order to have meaning, measures of performance must be related to the workload under which they are observed. To this end, one of the first steps of any performance evaluation project is to compose a model of the workload on which the performance evaluation is to be based. The workload of a system can be defined as the "set of all inputs (programs, data, commands) the system receives from its environment" (Ferrari, 1978).

All workload models are based on abstractions of actual or potential workloads. The actual or potential workloads are referred to as "real workloads." The real workload for a given model could be an actual production workload, or the projected workload of a new system.

Most performance evaluation studies require some way to exercise the subject system model under a controlled or known workload. But real workloads are ornery beasts. In production environments the workload changes at the collective whim of the users, making it difficult to describe the workload at any given moment, much less control it. If an analytic or simulation model of the system is

being used, the workload often has to be expressed as a probability distribution or as a set of events.

For these and other reasons, workload models are constructed to meet the needs of particular performance studies. A workload model is a simplified version of a real workload.

When selecting a modeling technique, a number of factors must be balanced to produce an acceptable model. Workload models, like any other models, differ in their cost, which is always a prime consideration. Three other criteria are of special importance for modeling workloads. These are representativeness, reproducibility, and compactness.

If a workload model is to be representative, it must faithfully simulate the real workload in the aspects that it strives to model.

A reproducible workload model is one that produces the same effects each time it is used. Reproducibility can be difficult to achieve in multiprogramming workloads. This is because the interactions between the effects of different processes are important, and timing is difficult to control. This can be especially difficult in comparison problems where different system characteristics can change the timing of the model.

Compactness is a measure of the degree of detail in the model. A very compact model is usually less detailed, less representative, and cheaper to use than a less compact model.

The design of a workload model involves trade-offs between representativeness and the other characteristics. The more representative a model, the more complex and less compact it is.

A workload model is constructed like any other model. A description is formulated based on key characteristics abstracted from the real workload. This model is then implemented in the form of a computer program or mathematical formulation. The descriptive value of the model is tested by using it to predict the behavior of the real workload. Adjustments are made to the model and the process is repeated until the model functions properly. It is then used in place of the real workload in performance evaluation studies.

The process of modeling a workload can be pictured as follows.

Abstract -> Formulate -> Construct -> Calibrate -> Use

Note that, at any step, feedback can occur to any previous step. It could be determined during formulation or any later step that more information on the nature of the real workload is needed. This would require a return to the abstraction step to gather that information. During model

use, the discovery that the model construction is difficult to use could motivate reformulation or changes to the construction.

Abstraction is the step in which the characteristics of the real workload are determined. It can be a conceptual or experimental procedure and, in fact, is usually both. The experimenter comes to some conclusions about the nature of the workload either by observation of production workloads or consideration of the uses of the system.

Once information about the nature of the workload has been acquired it can be put back together in a model. This is the formulation step. One way of formulating the model is to describe the workload in terms of smaller models called job models, which describe portions of the overall workload. A job model represents a well defined set of resource demands, and is generally based on some identifiable component of the real workload. A full workload model is formulated by combining job models.

Once the workload model has been formulated, it must be implemented in some form that can be utilized by the system model. This is the construction step. The form of implementation will depend on the type of system model to be used.

The implementation of a workload model on a real system requires the translation of the model formulation to

a set of programs, which can then be executed. Such a model is known as an "executable model." A workload model that is to be used with an analytic or simulation model of the system will be implemented as a statistical description of the resource demands or in some other numerical form. Such a workload model is known as a "non-executable" model. Of the two types of models, the non-executable ones tend to be cheaper and easier to use because they are more compact than executable models.

Executable models are heavily used in selection problems where they are generally implemented as a set of programs and data taken from a production workload. When a production workload does not exist, job models can be taken from a library of job models with known characteristics. The job models can be tuned to produce the desired resource demands. Predefined job models for performance evaluation are also known as "kernels".

Executable and non-executable workload models are types of "synthetic" workloads. Synthetic workloads are used in performance studies where a great degree of workload control is desired because of reproducibility requirements or a requirement for representativeness in specific aspects of the workload. There is another type of workload, called a "natural" workload, which is a production workload running in its natural environment. Natural workloads are used for gathering information for tuning a system or for final

validation of design decisions made on the basis of evaluation using synthetic models. Natural workloads tend to be cumbersome to work with, non-compact, impossible to reproduce, but very representative.

Once the model is constructed it is calibrated by comparing its resource utilization characteristics with the real workload. The actual method of calibration will depend on how the model is implemented. If the model differs substantially from its real workload, calibration may involve re-evaluation of the formulation with new information from the real workload.

When the performance analyst is satisfied that the workload model is representative enough for the needs of the study, it may be put to use driving the system model, which produces the desired measures of performance.

More complete discussions of workload modeling may be found in Ferrari (1978) chapter 5, and Svobodova (1976) chapter 4. Some practice oriented articles include Nolan and Strauss (1974), who discusses workload modeling with an orientation towards selection problems; Oliveret al. (1974), and Sreenivisan and Kleinman (1974) who discuss experiences in the use of synthetic benchmarks; and Spooner (1979a,b,c), and Bashioum (1979) who describe a project concerned with the benchmarking of interactive systems.

2.3 Performance Evaluation and Small Business Systems

Performance evaluation of small business systems (SBS) has much in common with that of larger systems. Differences emerge mainly as matters of scale and economics. The end users of small business systems will not generally have the resources or desire to conduct performance evaluation studies themselves. This situation differs from the large systems environment where computation centers generally have systems and operations staffs. This enables them to spend time tuning their systems. Such institutions are able to prepare and execute substantial comparative studies prior to acquiring new equipment or software. The small systems user will depend more on published comparisons of system performance when selecting new systems, and will have to settle for whatever performance he gets with the systems he has. These constraints place the burden of performance on the system developer who must augment his knowledge of systems development with knowledge of the workload encountered in the SBS environment. Systems designed for small businesses must be adapted more to the small systems environment.

There is very little in the technical literature which focuses directly on the performance evaluation of SBS's. Some recent articles by Dyal and Dewald (1979), and Huff (1979) present the methodology and results of the performance evaluation of some specific systems. Lewis

(1978) compared three microprocessor based SBS's on the basis of suitability and performance. His article presents the findings of the comparisons and makes some general suggestions for improvements in such SBS's. Jalics (1978) compares the performance of minicomputer systems vs. large computer systems using simple COBOL benchmarks. He documents several areas that show drastic, if unsurprising, differences.

Chapter 3

SMALL BUSINESS SYSTEMS

This chapter presents information on the nature of small business systems (SBS's). Section 3.1 is a review of the market for, and vendors of SBS's, with projections for market growth. The second section describes the price and hardware configurations of SBS's. Section 3.3 discusses the common applications of SBS's. The chapter concludes with a description of system software typically available with these systems.

3.1 Market Growth and Dominant Vendors

According to a report by International Data Corp. (1980), manufacturers sold in 1979 an estimated total of 236,000 small computer systems for business use. By June 1980 the installed base included approximately 478,000 systems worth a total of $6.1 billion. The same study projects an installed base of 3,489,000 systems by 1984 with a value of $30.3 billion, an increase of 620% in the number of systems installed!

A number of manufacturers are rushing to cash in on this bounty. The list includes several traditional business

computer suppliers and minicomputer manufacturers who are gearing up to be competitive in this potentially lucrative market. Of this group of established manufacturers IBM leads the pack by several lengths, with other well known names from the computer industry such as NCR, Burroughs, and DEC vying for the part the giant can't consume. In addition to the established suppliers, there are several new entries to the game with Wang Laboratories setting a shining example for would-be entrepreneurs.

3.2 System Configurations

Small Business Systems are distinguished from large business systems by price and scope of application. Since base system price is determined largely by the hardware configuration, a small business system can be characterized by its hardware components.

There are two main divisions in the price structure of the SBS market: low-priced systems costing between $5,000 and $30,000 with software, and (relatively) high-priced systems costing between $20,000 and $200,000.

Low-priced systems are sold primarily by retail electronics vendors such as Tandy Radio Shack, various specialty computer stores, and office equipment suppliers. They may be purchased either off the shelf or by mail. Systems in this category cost between $5,000 and $20,000 when bought mail order or off the shelf. When a system is

bought from a supplier of custom business systems prices rise to the $10,000 to $30,000 range. These suppliers provide customized software and more personal service than the off-the-shelf and mail-order suppliers. Systems in this lower-priced market are generally configured around an eight-bit or sixteen-bit microprocessor. They carry from 4K to 64K bytes of random-access memory, and can be equipped with cassette tape or floppy disc holding up to two million bytes of on-line storage. These systems are one-terminal single-user systems, and are usually configured with a hard-copy printer.

Most of the dollar volume in the small business systems market comes from sales of high-priced systems. They are bought by businesses with gross incomes between $1,000,000 and $25,000,000 a year. These organizations buy computers from suppliers who specialize in providing computer systems for particular markets, such as manufacturers, construction contractors, wholesalers, doctors, and lawyers. These "high end" small business systems are generally built around hardware produced by major suppliers such as IBM, DEC, Hewlett-Packard, Data General, Wang, Burroughs, and Prime. These systems use 16-bit and 32-bit CPU's, carry 64K to 2M bytes of random-access memory, and have from 10 to 500 Megabytes of on-line disk storage. The systems support from 1 to 16 concurrent users of CRT terminals. They may be configured with more than one line printer and support a number of

other peripherals such as magnetic tape, punched cards, optical character recognition devices, letter quality printers, typesetters, and data communications devices.

3.3 Applications Software

SBS applications fall into three major categories: administrative and financial applications, office automation applications, and industry-specific applications. The primary applications are administrative and financial. Figure 3.1 provides a list of some of the most common administrative and financial applications.

General Ledger
Accounts Payable
Accounts Receivable
Payroll
Invoicing
Order Entry
Inventory Control

Figure 3.1
Administrative and Financial Applications

The second important group of applications is in the area known as office automation. The primary application here is word processing, which helps to automate typing and other document preparation functions. A word-processing system allows the use of other document management functions, such as electronic filing and electronic mail.

Figure 3.2 provides a more complete list of functions falling in the office automation classification.

Word Processing
Electronic Mail
Electronic Filing
Meeting Scheduling
On-line Appointments Book
Travel Planning
On-line Telephone Book

Figure 3.2
Office Automation Applications

One final area of application that is important is "industry-specific applications". Each industry has a set of applications that are amenable to automation and for which software is available. Some examples are as follows: insurance claims handling for doctors and dentists; client accounting for lawyers, accountants, and professional consultants; and special technical applications for engineers.

3.4 System Software

The system software for SBS's is usually made up of adaptations of general-purpose software with additions to enhance applications programming productivity. One major aid to productivity is the applications generator, which produces program skeletons based on descriptions of the input, output, and general function of an application. Such

a skeleton may often be used as is or may be enhanced by a programmer to produce a polished application. The primary languages provided with SBS's are BASIC, COBOL, and Assembly language. Most of the applications software currently available is written in BASIC, but COBOL is now offered by all of the major vendors and is seeing increasing use.

Operating systems for SBS's are similar to other operating systems for the same size machines. They provide such standard resource management facilities as schedulers, file systems, and printer spooling. Additional features designed specifically for business use involve file system enhancements to support keyed files (also called indexed or indexed sequential files). These are disk files that contain additional information (called indexes) that allows the file to be read sequentially in several different orders without re-sorting. The index information also allows fast lookup of single data records based on the value of a key.

Other features now being provided as system software for SBS's include forms management software for designing screen and printer layouts, programs for doing simple data entry and file updating, batch processing, and communications. (Almost all systems provide a communications facility that allows the SBS to emulate an IBM 2780 workstation.) Some larger SBS systems now provide such advanced features as database management with query facilities, and network communications.

3.5 Literature

There is a plethora of publications covering the SBS marketplace. General trade publications such as Datamation, Computerworld, and Electronic News contain new product announcements and articles of interest on various topics relating to SBS's. A publication from Data General Corp., The Insiders Guide to Small Business Systems, contains a discussion of SBS's oriented to a naive, prospective purchaser. Datapro Reports on Minicomputers contains detailed descriptions of most of the major SBS products on the market today and is updated on a continuous basis. All SBS suppliers will supply information on the capabilities of their own products.

Chapter 4

AN SBS WORKLOAD MODEL

This chapter describes the development of a set of benchmarks for small business systems. The development methodology is based on procedures for the development of workload models described in section 2.2.3. Topics covered are the development goals, a characterization of the real workload, the method used to construct the model, and the techniques used to verify the representativeness of the model.

4.1 Development Goals

The development of the benchmarks was motivated by a need to compare the performance of several existing and newly developed SBS's. Of particular interest were measures of performance as a function of the number of active terminals. The benchmark development was conducted as an adjunct to an ongoing SBS development effort. Factors including manpower costs, time constraints, and hardware resources affected the choice of approach, and led to five primary requirements for the benchmarks.

1) Tractability. The benchmarks had to lend themselves to frequent use. Long setup times and cumbersome operating procedures were not acceptable.

2) Reproducibility. The primary use of the benchmarks would be in comparison experiments; therefore, they had to produce the same loading effects each time they were run.

3) Portability. Because they were to be used with several different operating systems, the benchmarks had to be as system-independent as possible. This goal was achieved by implementing the benchmarks in standard COBOL.

4) Representativeness. One of the primary uses of the information to be produced by the benchmarks was to help justify the replacement of existing products. It was important that the benchmarks be demonstrably representative of the real workloads that they modeled. The results of a special experiment designed to assure this characteristic are presented later in this chapter.

5) Implementation cost. Because the benchmarks were a tool rather than a revenue-producing product, the implementation costs had to be as low as possible. Manpower costs were controlled by using an existing applications package as a base and modifying it to automate the measurement process.

4.2 Characterization of the Real Workload

The first step in constructing a workload model is to describe the real workload. The SBS workload was described by dividing the programs that constitute it into five classes determined by program function: data entry, data maintenance, query, report generation, and batched update.

Data entry functions accept information from the applications user via the interactive terminal. This information is checked for validity and consistency. The checking often involves reading files to see whether the information is consistent with other data. The new information is then used to update one or more detail and summary files.

Data maintenance functions are used to delete or change data already existing in a database. The applications user provides the program with selector information to specify the data to be changed. The selected information is then located and displayed to the operator. If the function is CHANGE, the user provides the new information, and the program checks its validity and updates the various detail and summary files. If the function is DELETE, the user is asked to confirm the deletion and the appropriate files are updated.

Query functions are used to find and display small quantities of information to the waiting user. Generally,

the query function displays information from the database with very little summarization or interpretation. Query functions do not generally update any files, but may do more file reading than do entry or update functions.

Report generation functions are similar to query functions in that they produce formatted displays of the information in the database. They are different in that they operate on larger volumes of data and may perform summarization and interpretation. They generally take a relatively long time to complete, and are often run as part of the evening batch stream.

Batched update functions are non-interactive and are run periodically to combine detail and log files into new summary and master files. They often produce reports as a byproduct.

Applications systems consist of several subsystems that apply the five primary functions to different sets of files. For example, the four most common applications are accounts payable (AP), accounts receivable (AR), payroll (PR), and general ledger (GL). These applications all involve data entry, data maintenance, query, reporting, and batch update functions. AP applies them to vendor and AP transaction files; AR to customer and AR transaction files; PR to employee and personnel record files; and GL to journal and ledger account files.

In designing the benchmarks, the assumption was made that all programs implementing a given function have similar performance characteristics, regardless of their applications. Adopting this assumption, the makeup of an SBS workload can be expressed as a quintuple <E,M,Q,R,U> where:

E is the number of terminals used for data entry functions;

M is the number of terminals used for data maintenance functions;

Q is the number of terminals used for query functions;

R is the number of terminals used for report generation functions; and

U is the number of terminals used for batched updates. (A batch stream is considered to be an active terminal.)

The sum of the elements of the quintuple is the total number of active terminals.

4.3 Constructing the Model

An existing applications package, written in COBOL, was used as a base from which to construct the automated

benchmark programs. The base programs were selected from a set of programs within the Accounts Receivable (AR) subsystem. Local applications programmers felt that the AR programs were reasonably representative and probably easiest to modify. The specific functions selected were invoice entry for data entry, invoice change/delete for data maintenance, customer invoice query for query, and midday invoice register print for report generation. No batch update program was selected because of time constraints and because such programs are generally run outside of normal operating hours.

The model was constructed by modifying the real COBOL programs used to implement these functions. The modifications allowed the programs to run non-interactively.

The following assumptions guided the conversion process.

1) The overhead induced by actually typing the data from the keyboard is not significant relative to other activities, such as data file manipulation and screen formatting. This assumption meant that the programs did not have to be driven externally.

2) The data file manipulations performed by the programs are a significant part of the workload. The sizes of the records and the number of keys associated with the

data generally dictate the amount of overhead associated with each I/O operation.

3) The number of operations that use files shared with other programs in the workload has a significant effect on the amount of concurrency control overhead (collisions on locked records) generated by the workload.

The function models were produced by removing from the real programs, ACCEPT statements that were used to read data from the terminal. The real programs were then modified so that they took input data from tables rather than from a user at a terminal. Computation, data types, screen formatting, and data file operations were left largely unchanged. Extra logic required to control the flow of the program was kept to a minimum.

The actual number of terminals to be assigned to each of the functions was based intuitively on consultation with application specialists who concurred that the proportions were reasonable. To control the workload and collect timing information a supervisory program was written that dispatches the function models, records their start and stop times, and produces reports from the measurement sessions.

4.4 Model Calibration and Validation

In this project the assumption was made that the only model calibration needed would be in the determination of

the values of the elements of the quintuple. This determination was based on intuition and on knowledge of real-world situations.

In model validation it is important that the model components -- in this case the programs implementing the various functions -- accurately represent the programs from which they were derived. A separate experiment was conducted to provide this information and, in addition, to collect certain abstract information about SBS workloads.

The validation experiment was conducted by collecting frequency distributions of operations for the real and model programs, and comparing to see whether the dynamic composition of the model was similar to that of the real application.

The data collection process was made easier by the fact that the COBOL compiler on one of the systems to be tested generates a pseudo-code that is executed by an interpreter. Recording instruments for the model validation experiments were added to the interpreter to gather the desired frequency distributions. (Actual performance measures were taken using an interpreter without the validation experiment instruments in order to reduce measurement artifact.)

In order to validate the comparisons of the real and model programs, the compiler-generated operations were

divided into five classes roughly corresponding to groups of COBOL source statements. The programs were compared on the percentage of the total number of operations falling into each class. The classes were as follows.

* Arithmetic operations. Generated from ADD, SUBTRACT, MULTIPLY, DIVIDE, and COMPUTE statements.

* Data movement operations. Generated from MOVE statements.

* Input / Output operations. Generated from READ, WRITE, REWRITE, START, DISPLAY, and ACCEPT statements. The COBOL compiler allowed DISPLAY and ACCEPT statements that could move whole screens full of data in one operation.

* Flow of control operations. Generated from PERFORM and GOTO statements. Note that each paragraph of a program has code at the end to return to an invoking PERFORM when appropriate.

* Conditional operations. Generated from IF statements.

Two other statistics were also gathered and compared. One was the number of operations needed to process a single interactive transaction. The other was the ratio of

references to the two available numeric data types. The two types are (in COBOL terms): USAGE IS COMPUTATIONAL, which is implemented as a two's complement binary integer; and USAGE IS DISPLAY, which is implemented as an ASCII character string (also known as zoned decimal). The collection of this latter statistic was motivated by an article by Jalics (1978) which indicates that operations on these two numeric data types have significantly different performance characteristics on small business systems.

Figures 4.1-4.7 show the results obtained from executing the real and model programs. The model and the real program had reasonably similar performance profiles for all four of the programs run.

	Real Program	Model
Data Entry	14.32	12.87
Query	19.06	17.24
Change/Delete	7.56	7.09
Report Generation	7.23	4.96
Average	12.04	10.54

Figure 4.1
Arithmetic Operations
Percent of Total

	Real Program	Model
Data Entry	19.11	18.86
Query	22.46	23.23
Change/Delete	26.49	27.12
Report Generation	14.40	11.85
Average	20.62	20.26

Figure 4.2
Data Movement Operations
Percent of Total

	Real Program	Model
Data Entry	1.76	1.59
Query	1.86	1.37
Change/Delete	2.87	2.51
Report Generation	1.18	1.00
Average	1.92	1.61

Figure 4.3
Input/Output Operations
Percent of Total

	Real Program	Model
Data Entry	11.62	11.39
Query	11.32	11.02
Change/Delete	11.79	10.96
Report Generation	15.01	15.25
Average	12.44	12.16

Figure 4.4
Flow of Control Operations
Percent of Total

	Real Program	Model
Data Entry	48.97	50.53
Query	42.55	44.50
Change/Delete	48.54	49.41
Report Generation	60.50	65.84
Average	50.14	52.57

Figure 4.5
Decision Operations
Percent of Total

	Real Program	Model
Data Entry	4163	3634
Query	1167	1045
Change/Delete	1811	1855
Report Generation	1364	1289

Figure 4.6
Operations per Transaction or Record

	Real Program	Model
Data Entry	5.21	3.08
Query	9.13	8.32
Change/Delete	2.88	2.42
Report Generation	1.69	1.21

Figure 4.7
Approximate Ratios of
References to DISPLAY vs. COMPUTATIONAL Type Data

4.5 Some Observations on SBS Workloads

The following observations were made in the course of building and testing the benchmarks. It is not known whether or not these are general characteristics of small business systems, but they were thought to be significant enough for inclusion in this discussion.

The data presented in Figure 4.7 indicate that the data types supported by a SBS could have a significant effect on the performance of a system. Numeric data represented as character strings (DISPLAY) experienced a significantly larger number of references than did data represented as binary integers (COMPUTATIONAL). The inherent slowness of DISPLAY arithmetic could make data types a significant factor in the workload.

The relatively large component of the workload represented by decision statements (Figure 4.5) is surprising. Even though visual inspection of the benchmark source code indicated that IF statements make up a sizable component of the program, the very large number of conditional operations seen at run time was not expected.

The very low I/O component (Fig 4.3) seen in the operation frequencies was not expected, although it is consistent with observations from visual inspection of the program source code. The pseudo-code I/O operations have a relatively high semantic content, and the dynamic

frequencies probably do not reflect their actual contribution to the workload.

One interesting effect observed while testing the benchmarks is that there is a "knee" in performance which is related to data file size. The knee occurs at different file sizes on different systems and is probably due to the organization of the index files and the way that they are buffered.

Conventional wisdom has it that I/O is a very large component of commercial workloads. Although I/O was only a minor aspect of this study, experience in production environments indicates a need for more investigation of this aspect of SBS workloads. Some interesting measures are: (1) channel throughput, differentiating between disk and screen I/O; (2) record size distributions for disk I/O; and (3) the number of keys per record. Other interesting topics for study are the effect of file sharing on performance, and the frequencies of collisions on shared files and records in real workloads.

Chapter 5

CONCLUSION

In the next ten years computers will become as important a tool of the small business as the typewriter is today. It will be the responsibility of the designers and builders of those systems to make cost effective, productive tools with a sensitivity to the special needs of the small user.

The writing of this thesis served as a focus for an investigation of the applicability of techniques for performance evaluation to the special needs of SBS's. Although most of the major topics of performance evaluation were mentioned, the focus of the investigative work was on characterizing the workloads of SBS's. In the course of this investigation several questions were raised that could not be dealt with within the time constraints of the project. Such questions as "What are acceptable response times for SBS's?", and "What are the throughput requirements?" are of special importance to the SBS developer. Much of the general work in performance evaluation is applicable to these and other SBS related questions. Where general research cannot provide the answers, specific investigation will be required to provide

developers with the information they need to produce cost effective information management tools for small business.

BIBLIOGRAPHY

Adams, J. C., Currie, W. S., and Gilmore, B. A. C., "The Structure and Uses of the Edinburgh Remote Terminal Emulator." Software -- Practice and Experience 8, 4, pp. 451-459, July-August 1978.

Bashioum, D. L., "Benchmarking Interactive Systems: Calibrating the Model." Performance Evaluation Review 9, 2, pp. 35-41, Summer 1980.

Data General Corp., The Insider's Guide to Small Business Computers. Data General Corp., Westboro, Mass., 1980.

Data General Corp., Interactive Cobol Programmer's Reference. Data General Corp., Westboro, Mass., undated.

Datapro Reports on Minicomputers, Datapro Research Co., Delran, New Jersey, updated monthly.

Dowdy, L. W., Agrawala, A. K., Gordon, K. D., and Tripathi, S. K., "Computer Performance Prediction via Analytical Modeling -- An Experiment." Proceedings Conference on Simulation, Measurement, and Modeling of Computer Systems, Boulder, pp. 13-18, 1979.

Dujmovic, J. J., "Criteria for Computer Performance Evaluation." Performance Evaluation Review 8, 3, pp. 259-267, Fall 1979.

Dyal, J. O., and DeWald, W., "Small Business Systems Performance Analysis." Performance Evaluation Review 8, 3, pp. 269-275, Fall 1979.

Ferrari, D., "Workload Characterization and Selection in Computer Performance Measurement." Computer 5, 4, pp. 18-24, July-August 1972.

_____, Computer System Performance Evaluation. Prentice-Hall, Englewood Cliffs, New Jersey, 1978.

Gay, A. R., "Benchmarking a Multi-Access System.", Software -- Practice and Experience 10, 1, pp. 45-55, January 1980.

Hellerman, H., and Conroy, T. F., Computer Systems Performance. McGraw-Hill, New York, 1975.

Huff, R. W., "System Characterization of a Retail Business System." Performance Evaluation Review 8, 3, pp. 277-284, Fall 1979.

International Data Corp., The Small Computer Marketplace. #2103, International Data Corp., Waltham, Mass., 1979.

Jalics, P. J., "Performance of COBOL programs on Mini vs. Large Computer Systems." SIGMINI Newsletter 4, 4, pp. 17-23, August 1978.

Kienzle, M. G., and Sevcik, K. C., "A Survey of Analytic Queueing Models of Computer Systems." Proceedings Conference on Simulation, Measurement, and Modeling of Computer Systems, Boulder, pp. 113-129, 1979.

Lewis, T. G., "Performance Evaluation of Three Microprocessor Based Systems in a Small Business Environment." SIGMINI Newsletter 4, 4, pp. 9-16, August 1978.

Lo, T. L., "Computer Capacity Planning Using Queueing Network Models." Performance Evaluation Review 9, 2, pp. 145-152, Summer 1980.

Lucas, H. C., Jr., "Performance Evaluation and Monitoring" Computer Surveys 3, 3, pp. 79-91, September 1971.

McDaniel, G., "METRIC: A Kernel Instrumentation System for Distributed Environments." In Proceedings Sixth Symposium on Operating Systems Principles, pp. 16-18, November 1977.

Nolan, L. E. and Strauss, J. C., "Workload Characterization for Timesharing System Selection." Software -- Practice and Experience 4, 1, pp. 25-39, January-March 1974.

Oliver, P., Baird, G., Cook, M., Johnson, A., and Hoyt, P., "An Experiment in the Use of Synthetic Programs for System Benchmarking." Proceedings AFIPS Conf. (NCC). 43, pp. 431-438, 1974.

Sanguinetti, J., "A Technique for Integrating Simulation and System Design." Proceedings Conference on Simulation, Measurement, and Modeling of Computer Systems, Boulder, pp. 163-172, 1979.

Shneiderman, Ben, Software Psychology. Winthrop, Cambridge, Mass., 1980.

Smith, C., and Browne, J. C., "Performance Specifications and Analysis of Software Designs." Proceedings Conference on Simulation, Measurement, and Modeling of Computer Systems, Boulder, pp. 173-182, 1979.

Spooner, C. R., "Benchmarking Interactive Systems." Proceedings Summer Computer Simulation Conference, pp. 791-798, Toronto, July 1979a.

_____, "Benchmarking Interactive Systems: Producing the Software." Performance Evaluation Review. 8, 3, pp. 249-257, Fall 1979b.

_____, "Benchmarking Interactive Systems: Modeling the Application." Proceedings CPEUG Conference, San Diego, Oct. 1979c.

Sreenivisan, K., and Kleinman, A. J., "On the Construction of a Representative Synthetic Workload." Communications of the ACM 17, 3, pp. 127-133, March 1974.

Svobodova, L., Computer Performance and Evaluation Methods: Analysis and Applications. Elsevier, New York, 1976.

Unger, B. W., and Parker, J. R., "An Operating System Implementation and Simulation Language (OASIS)." Proceedings Conference on Simulation, Measurement, and Modeling of Computer Systems, Boulder, pp. 151-161, 1979.