PERFORMANCE STUDY

OF GMIS

by

LUIS EDUARDO SCHMIDT SARMENTO

Engenheiro, Escola Politecnica da USP
Brazil
(1973)

SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE

DEGREE OF MASTER OF

SCIENCE

at the

MASSACHUSETTS INSTITUTE OF

TECHNOLOGY

June, 1976

Signature of Author .....
Alfred P. Sloan School of Management
May 7, 1976

Certified by .....
Thesis Supervisor

Accepted by .....
Chairman, Departmental Committee
on Graduate Students

PERFORMANCE STUDY OF GMIS

by

Luis Eduardo Schmidt Sarmento

Submitted to the Alfred P. Sloan School of Management on May 7, 1976 in partial fulfillment of the requirements for the degree of Master of Science.

Abstract

This paper presents and analyzes alternatives to improve the performance of GMIS (Generalized Management Information System).

System performance is measured by using tools developed by IBM to study the performance of VM/370 systems.

The data obtained in the measurement procedure is used in a GPSS program that simulates the following improvement alternatives under different loads on the computer and on GMIS itself:

a) optimize the programming of the data base virtual machine, so that each query is processed faster.

b) hardware upgrade, using a faster CPU

c) change GMIS architecture so that several queries can be processed simultaneously by having several data base VM's running in multi-programming and accessing a common segmented data base.

We conclude that alternative a is the one with the highest payoff. Alternative c is not efficient and hardware upgrade is not necessary given the present load conditions.

Thesis Supervisor: Peter Pin-Shan Chen
Title: Assistant Professor of Management Science

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

Introduction

1.1 Purpose of the Study

The purpose of this study is to measure GMIS performance under different loads on the computer, predict the performance degradation that the system may have by serving various users simultaneously, and analyze alternatives for performance improvement.

1.2 Description of GMIS

This section presents a brief description of GMIS (Generalized Management Information System). A more comprehensive description can be found in Gutentag [1] and examples of applications in Donnovan and Jacoby [2].

GMIS is a flexible data management system which permits rapid implementation of many applications systems, and currently is installed on the IBM System /370 Model 158 located at the IBM Cambridge Scientific Center. It uses the virtual machine concept extensively. A virtual machine may be defined as a replica of a real computer system simulated by a combination of software and appropriate hardware support. The virtual machine /370 system enables a single IBM /370 to appear functionally as though it were multiple independent system /370's.

A configuration of virtual machines used in GMIS is depicted in Figure 1.1 where each box denotes a separate virtual machine.

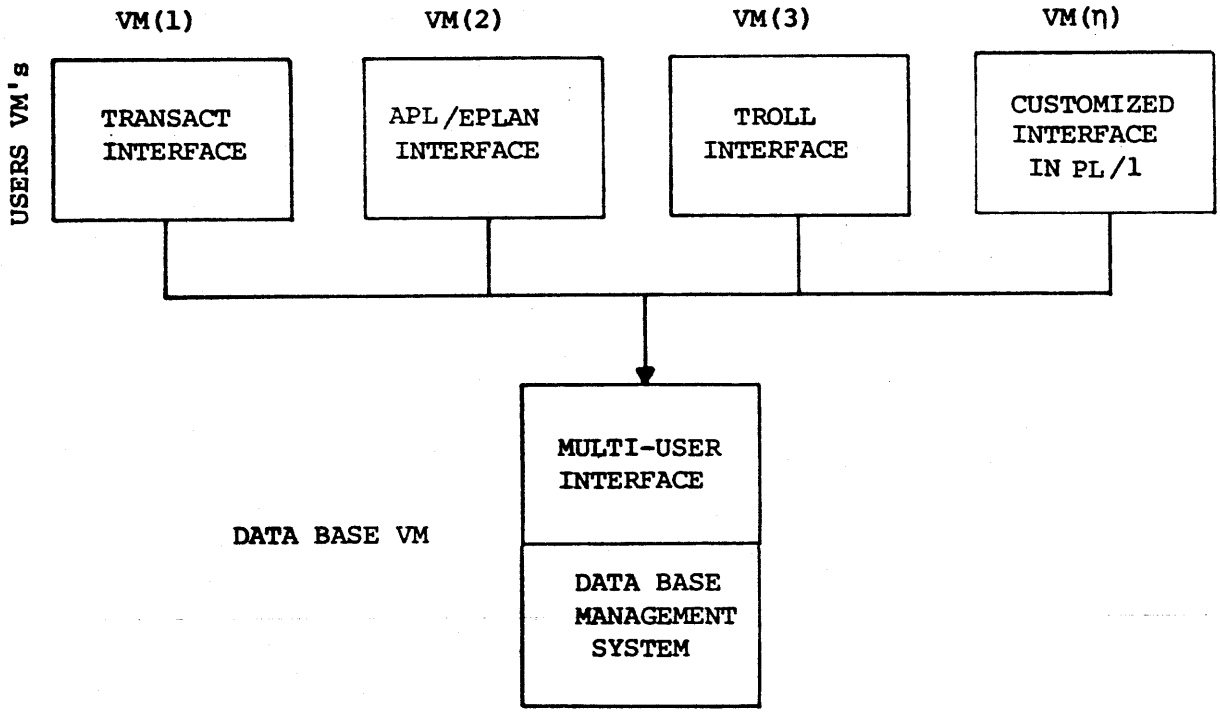


FIGURE 1.1

Each GMIS user resides in his own virtual machine with a copy of the user interface he requires. Each user transaction to the data base is written into a transaction file, and that user's request for processing is sent through a communication mechanism to the data base machine. The multi-user interface

processes each request in a FIFO order, by reading the selected user's transaction file, writing the results to a reply file that belongs to the user, and signaling to the user machine that the request has been processed. Each user interface reads the reply file as if the reply had been passed directly from the data base management system.

1.3 Contents of the Study

We start our Chapter 2 by analyzing the performance and load of the System /370 where GMIS is currently installed. The idea is to verify whether or not we can put the blame on an overcrowded computer for GMIS bad performance, since many other applications share the computer with GMIS. This analysis also provides the basis to judge whether a hardware upgrade is advisable as a means to improve GMIS performance.

In Chapter 3 we present a Queueing Model of GMIS. The model is used in a GPSS simulation program to determine the degradation in performance with each additional user, as well as the impact on performance of the following improvement alternatives:

a) optimize the programming of the data base machine so that each query is processed faster.

b) hardware upgrade, using a faster CPU (use the model 168 instead of the 158 being used).

c) change the software architecture so that several queries can be processed simultaneously by having several data

base machines running in multiprogramming and accessing a common segmented data base.

CHAPTER 2

Analysis of Performance and Load of the System /370 Where GMIS is Installed

This chapter presents a performance analysis of the system 370 model 158 with 2 megabytes, where GMIS is presently implemented. The framework we adopt in this analysis is the one proposed by Bard in a series of articles [3,4,5]. The measurement of the relevant performance data was made over a period of one month by using the Virtual Machine Performance Tool (VMPT). VMPT is an IBM aid that is described briefly in the next chapter. Further details on VMPT can be found in the user's guide [6].

2.1 Description of VM 370

VM /370 is, for the purpose of performance analysis, simply a virtual-memory time sharing system (for a more comprehensive description see [7,8,9]). Each user of the system may enter tasks, usually from a remote terminal. The system shares its resources among these tasks. The flow of user tasks through the system is depicted in Figure 2.1.

A user is in the dormant state until he has completed entering a task. Until proven otherwise, the task is assumed to be interactive, i.e., to require fast response while making only slight demands on system resources. While receiving

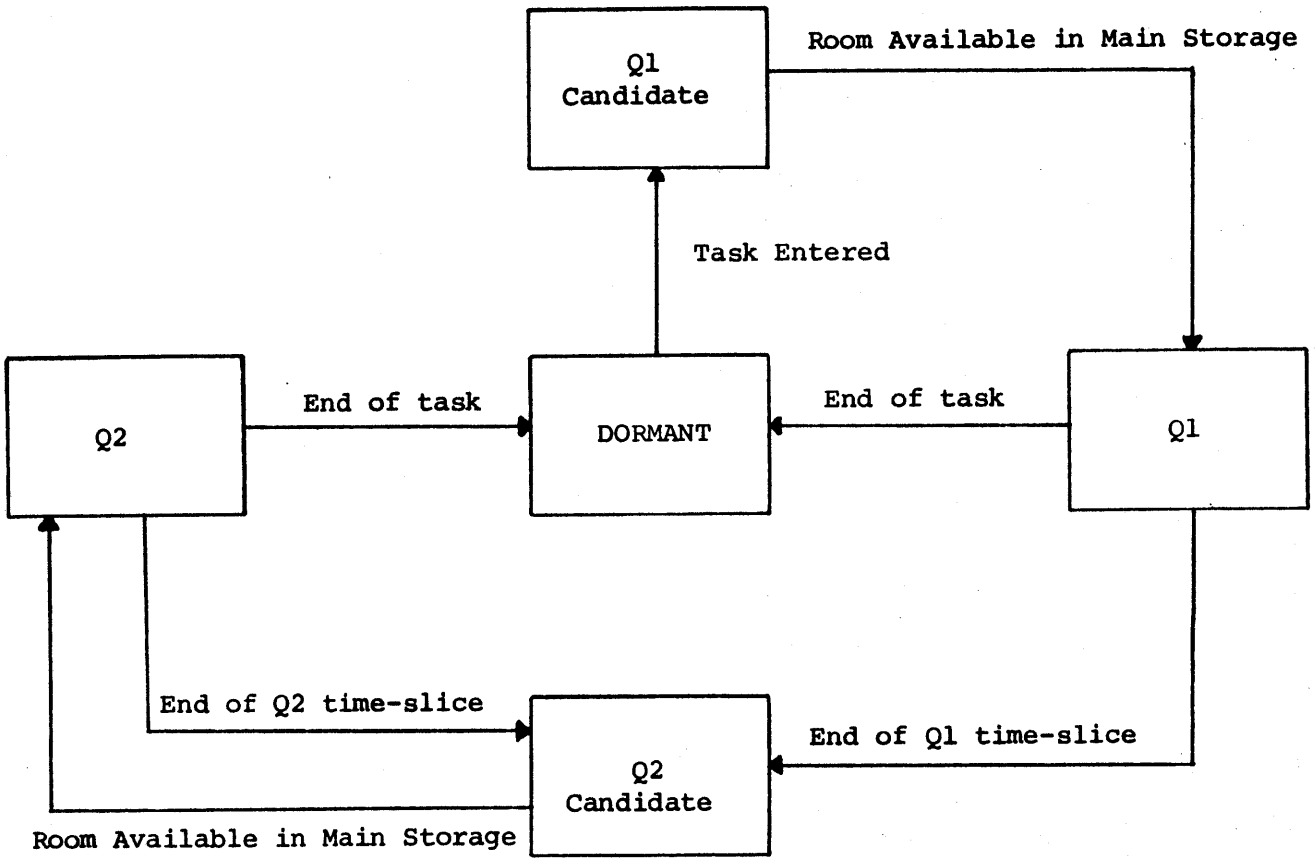

Q1
Candidate
Room Available in Main Storage
Task Entered
Q2
End of task
DORMANT
End of task
Q1
End of Q2 time-slice
End of Q1 time-slice
Q2
Candidate
Room Available in Main Storage

FIGURE 2.1

service, such tasks are said to be in Q1, but before being admitted to this state they are called Q1 candidates. If a Q1 task does not terminate before consuming a certain amount of CPU time (roughly 400 msec), it loses its interactive status. It now becomes a Q2 candidate, and is eligible to be admitted into Q2, which is the set of noninteractive tasks being currently serviced. There is also a limit (about 5 seconds) on the amount of CPU time that a task may receive during one stay in Q2. A task requiring more CPU time may cycle several times between the Q2 candidate and Q2 states.

The only tasks which may actually receive CPU time at any moment are those in Q1 and Q2. These are called in-Q tasks, and their number is the multiprogramming level (MPL). The Q1 and Q2 candidates are tasks which are ready to run, but are not allowed to do so at the moment because the system does not wish to overcommit its resources. Admissions from Q candidate to in-Q status is in order of task priority.

In-Q users' main storage requirements are met dynamically through a demand paging mechanism. The system maintains an estimate of each user's storage requirements; this estimate is called the user's projected working set. Admission is based principally on the availability of main storage space to accommodate the user's projected working set.

2.2 System Saturation and Bottlenecks

As the MPL goes up, the system is increasingly able to

overlap the use of its various components, consequently improving their utilization. Soon, however, one or more components approach 100 percent utilization so that no further increase is possible. The system then is saturated and one or more components are the bottlenecks responsible for the saturation. These components are the ones whose capacity must be increased first before overall system performance can be improved.

The main function of the system scheduler is to maintain the optimal MPL for the given system configuration and user work load. We will analyze in this section the relation between performance and the load placed on the system. We will adopt as a measure of system load the number of active users on the system. Active users are those logged on the system which used some CPU time during an observation period of 60 seconds.

The main hardware components of a VM /370 System are the CPU, main storage, the paging subsystem, and the I/O subsystem. We will analyze below the utilization of each of those components as a function of system load. But before doing that, let's see how the load on the system varies over a typical day on the IBM Cambridge Scientific Center installation. Appendix A shows how the number of active users and logged on users very over an average day. From the beginning of a day up to 7:00 a.m., the load is leveled at about 4 active users. At 8:00 a.m. the load starts going up until 10:00 a.m. reaching

a peak of 16 active users. It goes down as lunch time approaches and goes up again after lunch to reach a peak of 19 active users by 3:00 p.m. Then it declines sharply up to 6:00 p.m. and then smoothly until the end of the day.

CPU utilization analysis can be done based on two plots: one, presented in Appendix B, showing CP and problem states percentages as a function of active users; the other, presented in Appendix C, showing CPU wait percentage and its components. The VM /370 system always runs users' CPU instructions in the problem state and its own (CP) instructions in the supervisor state. The amount of time that CPU spends in problem state is therefore a good measure of "useful" work done, or through put attained by the system. The system breaks up total wait time into three components: Idle wait, when no high speed I/O are outstanding; Page wait, when outstanding I/O requests are primarily for paging and I/O wait, when outstanding I/O requests are not primarily for paging.

In Appendix B we can see that CPU problem state percentage get to its peak at about 22 active users. Total CPU utilization seems to level off thereafter with a small drop in problem state percentage being compensated by an increase in CP state percentage. By looking at Appendix C we can confirm this impression, verifying that total wait percentage levels off beyond 22 active users. This means that the system is saturated at a load of 22 active users, and since there still is a significant amount of wait (about 17%) at the saturation

point we can conclude that CPU is not bottlenecking the system. This wait state is mostly I/O wait and it may be due to poor overlapping of CPU and I/O activities, caused by main storage being insufficient to accommodate an adequate MPL, or because the I/O subsystem is not fast enough for this CPU.

Appendix D shows Q1, Q2 and Q candidates as a function of active users. Main storage is (or at least the scheduler thinks that it is) saturated when the Q candidates list is never empty. We can see from Appendix D that this happens at about 24 active users. However, the main memory is not bottlenecking the system since paging at this load is moderate (see page wait in Appendix C) and CPU is at its peak utilization. Main memory would have to be increased only if a more powerful CPU were installed.

Appendix E shows the percentage of pageble core utilization by users in Q and the percentage of pageble core demanded by all users in Q and Q candidates. We can see that core utilization is low even at the main memory saturation point. This might mean that the scheduler is keeping a lower MPL than main memory would accommodate (without increasing much the paging activity) because the Q-candidate tasks are of an I/O bound nature and an increase in the MPL would overcommit an already saturated I/O subsystem. If that is the case, the I/O subsystem is bottlenecking the system and should be expanded and/or better balanced. However, if the Q-candidate tasks are not of an I/O bound nature, the scheduler is

unnecessarily constraining the system, since an increase in MPL could be accommodated by main memory and would lead to a better overlapping of CPU and I/O activities, increasing CPU utilization and overall system performance.

In conclusion to the analysis above we may say that the IBM Cambridge Scientific Center computing facility is well balanced and dimensioned to handle its peak load near saturation. However, we think that an improvement in the I/O subsystem, if feasible, would better balance the system components and lead to better performance and response times, especially for I/O bound applications (as GMIS for instance). The upgrade of the other components of the system or the improvement of the I/O subsystem beyond the point in which it is no longer bottlenecking the system will be necessary only if the load increases in the future.

CHAPTER 3

Model of GMIS

This chapter presents a description and analysis of a queueing model of GMIS.

The model is used in a GPSS simulation program to determine the degradation in performance with each additional user, as well as the impact on performance of the following improvement alternatives:

a) optimize the programming of the data base machine so that each query is processed faster.

b) hardware upgrade, using a faster CPU (use the model 168 instead of the 158 being used).

c) change the software architecture so that several queries can be processed simultaneously by having several data base machines, running in multiprogramming and accessing a common segmented data base.

3.1 Model Description

We start with a queueing model that was presented in a working paper by Donovan and Jacoby [2]. Then we extend it so that some simplifying assumptions are dropped and alternative c above is included.

The model shown graphically in Figure 3.1 assumes that all modeling VM have virtual speed constant and equal.

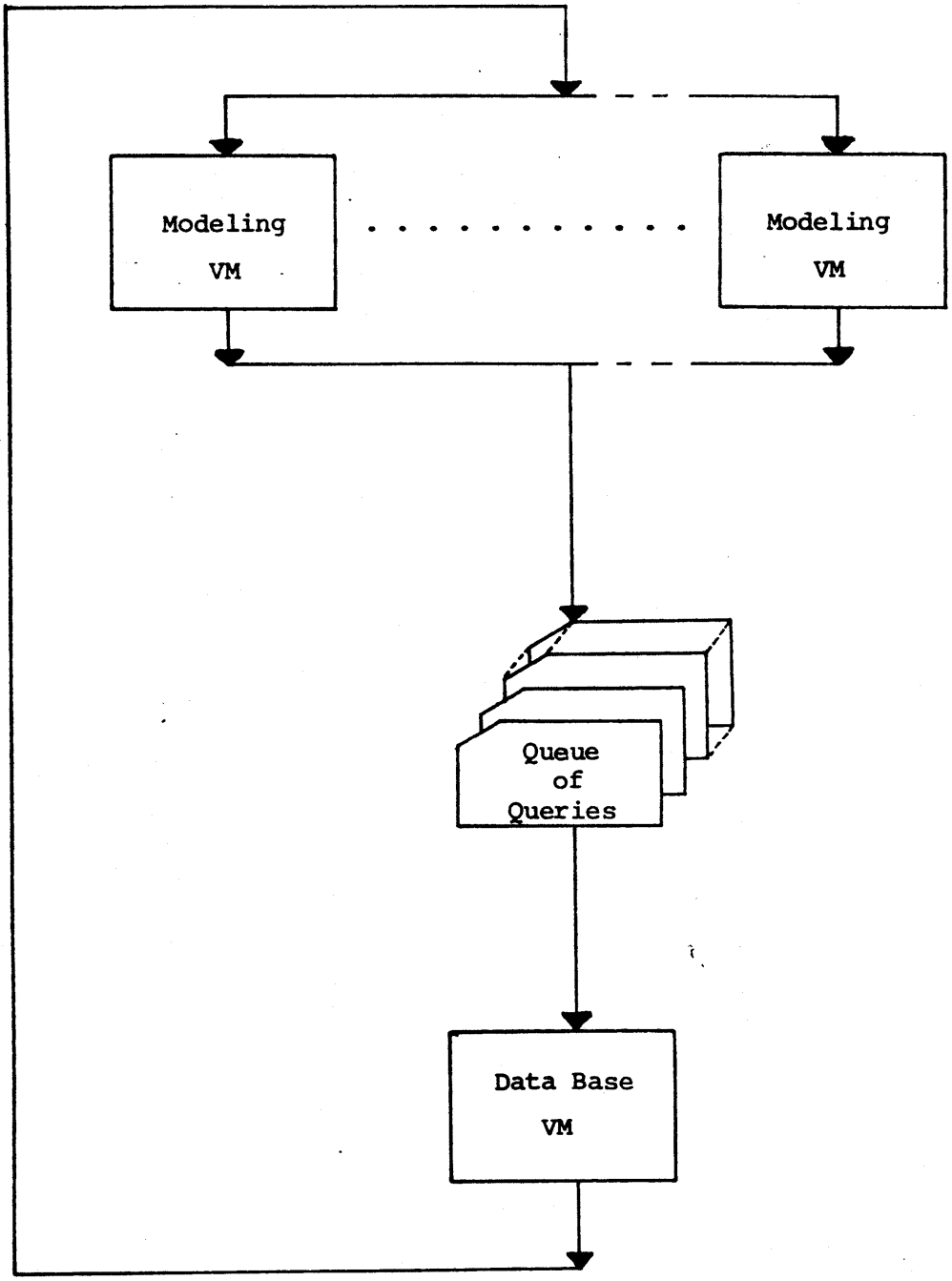


FIGURE 3.1

However, when some VM's are blocked, waiting for a query, the others are allocated a larger share of CPU processing power, and become faster in real time. It is assumed that each unblocked VM receives the same amount of CPU processing power, that all modeling machines are running in batch without interaction with the user between queries, and that all users logged on the computer are using GMIS (the last three assumptions will be relaxed later).

Let the following variables be:

M = number of modeling machines logged on

j = number of blocked modeling VM (waiting for data)

R_j = query rate of each modeling VM, when there are j blocked machines

L_j = rate at which a new query enters in queue, when there are j blocked machines

L = query rate of each modeling VM when running alone (virtual CPU time = real CPU time)

uj = rate at which the data base VM "serves" the incoming queries, when there are j blocked machines

u = service rate of data base VM, when running alone

Then we may write the relations:

$$R_j = \frac{L}{M-J+1} \qquad J = 1,2\ldots(M-1)$$

$$R_j = \frac{L}{M} \qquad J = 0$$

$$L_j = (M-j)\ R_j$$

$$u_j = \frac{L}{M-J+1} \qquad J = 1,2\ldots M$$

From the 3 first relations we can derive:

$$L_j = L \qquad J = 0$$

$$L_j = \frac{M-J}{M-J+1}\ L \qquad J = 1,\ldots M$$

By using Little's formula [10] we can get the expected time that a query stays in queue and is served.

$$E\ [t] = \frac{\sum_{j=0}^{M} j\ P_j}{\sum_{J=0}^{M} L_j P_j}$$

where P_j is the steady state probability that j machines are waiting for data. P_j can be determined by using a birth and death markov process [11,12] as illustrated in Figure 3.2.

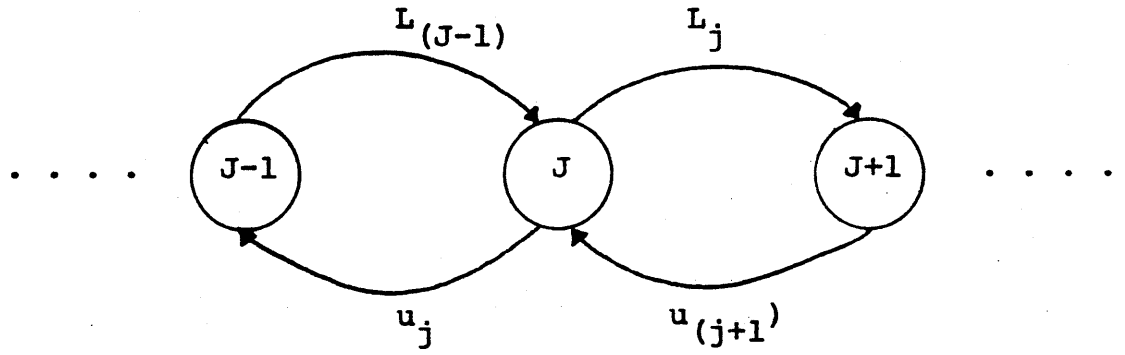


FIGURE 3.2

$$P_j = \frac{L_{(j-1)}}{u_j} \quad P_{(j-1)}$$

$$\sum_{J=0}^{M} P_j = 1$$

The improvement alternatives a and b presented earlier can be analyzed by using different values for L and u. Alternative c requires that we transform the model into a multiple server queueing model with state dependent parameters [12]. That is done by redefining the relations for L_j and u_j as follows:

$$\begin{cases} L_j = \dfrac{(M-J)\ L}{M} & J = 0,\ldots,N \\ L_j = \dfrac{(M-J)\ L}{(M-J)+N} & J = (N+1),\ldots,M \end{cases}$$

$$\begin{cases} u_j = \dfrac{J\ u}{M} & J = 0,\ldots N \\ u_j = \dfrac{Nu}{(M-J)+N} & J = (N+1),\ldots,M \end{cases}$$

where N is the number of "servers" (data base machines) in the system.

The above extension to the model does not include the overhead of coordinating several data base machines accessing a segmented data base in a multilock scheme. This simplification is more reasonable if we assume that only a small proportion of the transaction to be processed will eventually modify the contents of the data base.

The coordination mechanism could be built as follows:

All data base machines would share the same virtual card reader and read the virtual cards punched by the modeling machines in a way analogous to the communication mechanism currently implemented in GMIS. When a machine is processing a transaction that modifies a given segment of the data base,

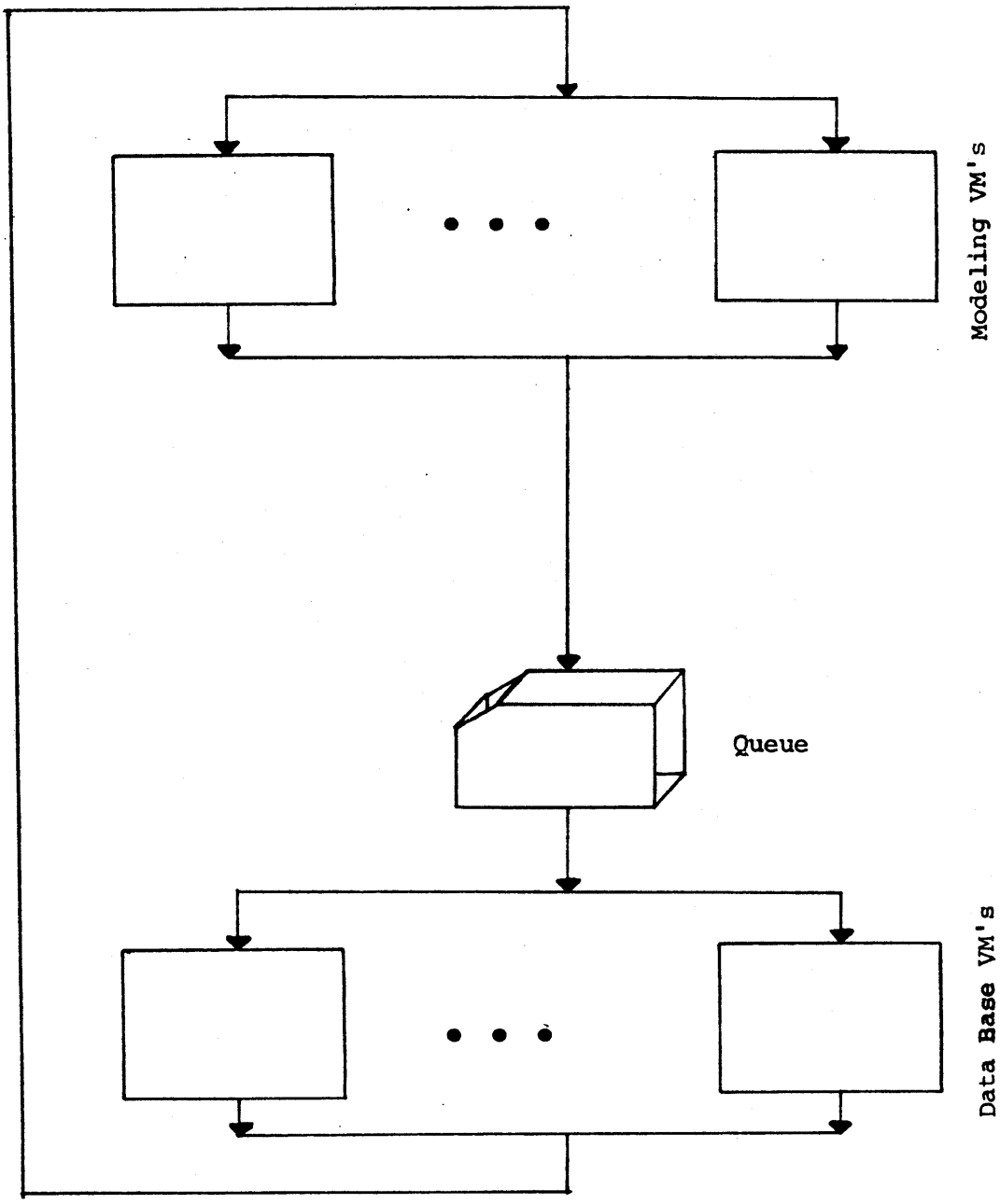


FIGURE 3.3

it would have to lock that segment during the update procedure. This locking could be done by means of a "key" located in the beginning of each segment or in a shared memory area. This key would be tested by the data base machines before accessing a segment.

We proceed now to modify the model to be interactive as shown in Figure 3.4.

The boxes representing service time of the VM's refer to the elapsed time from when the machine became a Q candidate at the start of the transaction until it is dropped from Q at the end of the transaction. Thus, network delays, printing time, keypunching time, etc., are included in think time.

The mathematical analysis of the scheme in Figure 3.4 is not simple, so we decided to use a simulation program in the analysis.

3.2 Parameters Evaluation

The parameters think time, service time of modeling VM and service time of data base VM, were measured by using the following tools:

VM/MONITOR [8,13] is a data collection tool designed for sampling and recording a wide range of data. The collection of data is divided into functional classes. The different data collection functions can be performed separately or concurrently. Key words in the monitor command enable the collection of data, identify the various data collection classes

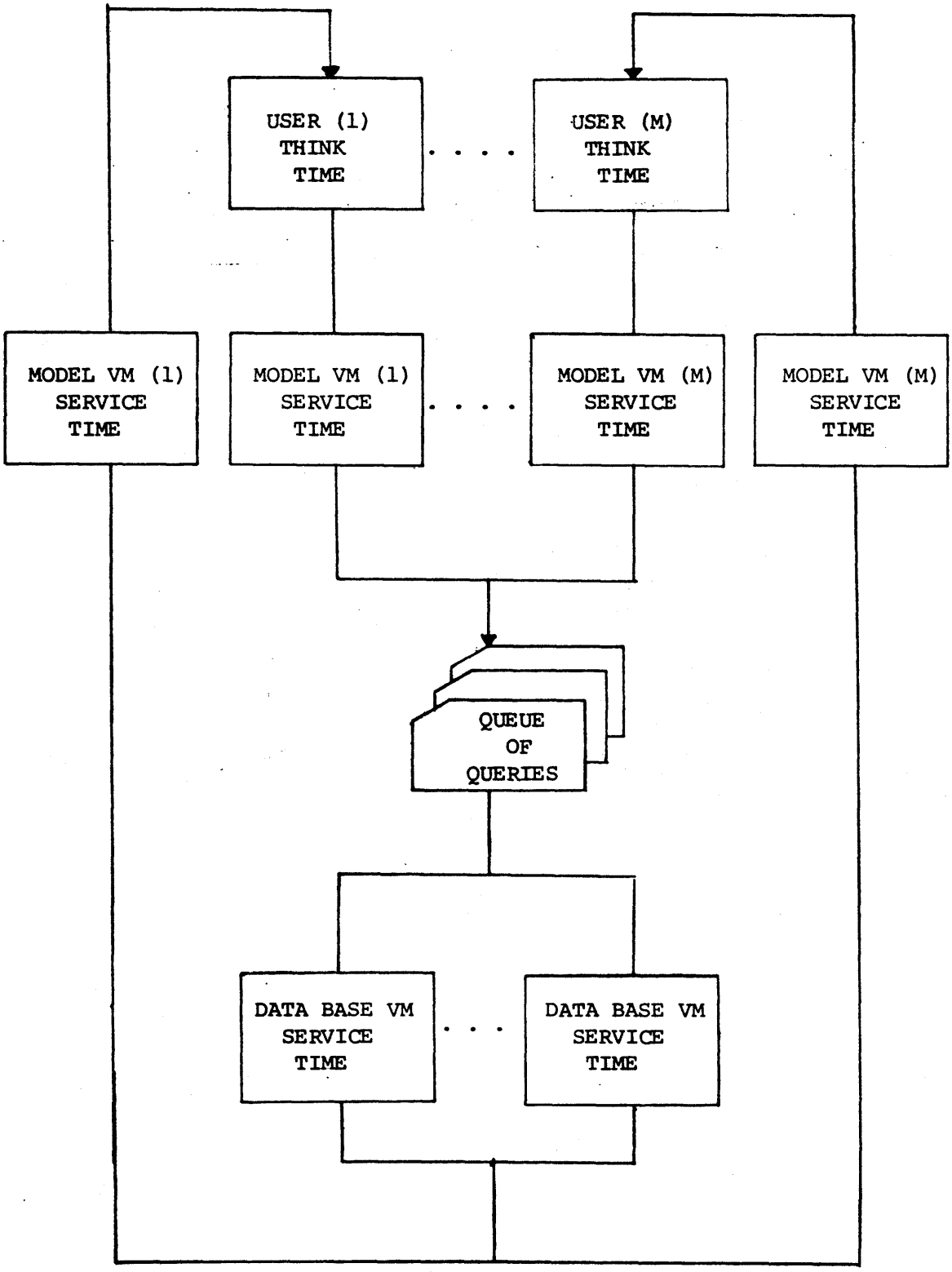

USER (1)
THINK
TIME
USER (M)
THINK
TIME
MODEL VM (1)
SERVICE
TIME
MODEL VM (1)
SERVICE
TIME
MODEL VM (M)
SERVICE
TIME
MODEL VM (M)
SERVICE
TIME
QUEUE
OF
QUERIES
DATA BASE VM
SERVICE
TIME
DATA BASE VM
SERVICE
TIME

FIGURE 3.4

and control the recording of collected data on tape for later examination and reduction. MONITOR CALL instructions with appropriate classes and codes are embedded throughout the main body of VM /370 code (CP). When a MONITOR CALL instruction executes, a program interruption occurs if the particular class of call is enabled. Monitor output consists of event data and sampled data. Event data is obtained via MONITOR CALL instructions. Sampled data is collected following timer interruptions.

VM /370 Predictor [14]. This is an IBM internal use program. It was designed to support marketing recommendations resulting in proposals for IBM equipment. It consists of two distinct parts:

1) The VM predictor model is an analytic model designed to predict VM /370 performance for specified system configurations and workloads. This model was not used in this study.

2) The VM predictor data reduction package is a set of programs to analyze data obtained by running VM/Monitor. This package produces a report describing the workload characteristics for each user logged on the system during the monitored period (see sample in Appendix F). It also produces a report summarizing systems performance during the monitoring section (see sample in Appendix G), for comparison to the VM predictor model performance predictions.

VM Performance Tool [6]. VMPT is an IBM aid designed for software analysis of VM /370 systems. The VMPT package is made

up of two components: a performance measurement package (VMP) and a performance data analysis package (VMA).

The VM /370 system maintains various sets of internal counters for system activity, for each I/O device and for each user logged into the system. The general functions of VMP are to read these VM /370 counters at specified intervals through a disconnected virtual machine, to save the values of these counters in a special disk file and to perform various types of analysis on the data read.

The VMA component of VMPT is designed to provide an extension to the analytic facilities in VMP and is used to obtain an overall characterization of system performance. The general functions of VMA are to compute simple statistics such as means and standard deviations and produce histograms, scatter and trend plots for selected variables. The primary source of input to VMA is found in the disk file created by VMP and VM /370 observations. (Sample VMA reports can be found in Appendixes A, B, C, D, E and H.)

3.2.1 User Think Time

Appendix H shows a VMA activity report that contains mean and standard deviation of think time for all interactive jobs (GMIS and non-GMIS users) over a period of one month. We decided to adopt these values in the simulation, since VMA does not provide reports broken down by users and the benchmark programs we monitored presented think times consistent with

those values. These benchmarks, as many other GMIS applications, are not actually interactive. They simply print on the terminal the answers for queries previously recorded on a disk file.

Based on the data on the VMA report we assumed the think time to be an Erlang distribution of second order with mean equal to 5.94 seconds. This value is less than one would observe in a real interactive job. But by adopting this distribution on the simulation we can produce a range of think times from near zero for those pre-programmed queries, up to 30 or 40 seconds when the user has to input some data through the terminal.

3.2.2 VM Service Times

The service times of the data base and modeling VM's were measured for two benchmark programs in several one hour monitoring sections. The monitor tapes obtained were analyzed by using the VM Predictor Data Reduction Package.

Both programs used as benchmarks are actual GMIS applications. Benchmark-1 runs on a APL environment and is a set of uniform complex queries of the type

```
SELECT TOT (SUMS) FROM PFF - CONSUMPTION WHERE MONTH = 1 AND
STATE = 'CT' AND COUNTY = 'FAIRFIELD' AND FUELTYPE IN
('MIDDLE DISTILLATE', 'DISTILLATE', 'NUMBER 2 OIL');
```

The answer for each query is printed on the terminal and saved in a file. At the end of the run several plots are out-

put in the printer.

Benchmark-2 runs on TRANSACT and is a set of non-uniform queries. Appendix I presents a complete list of these queries. This program instead of passing one query at a time to the data base VM, as benchmark-1 does, passes the complete set of queries in the beginning of the run. This way there is no communication between the modeling VM and data base VM for each query. They communicate only at the beginning and at the end of the job.

We decided to base our estimate of service time of data base VM on benchmark-2, because it contains a wider range of query types and to base our estimate of service time of modeling VM on benchmark-1 because the communication pattern between the two VM's for this program is similar to the scheme presented in Figure 3.4.

Figures 3.5, 3.6 and 3.7 show the scatter plots of average service time per query versus average number of users in-Q during the monitoring section for the data base and modeling machines. The number of users in Q measures the degree of multiprogramming during the monitoring section; that is, the average number of virtual machines (GMIS and non-GMIS users) that shared system resources during the monitored period.

The least square lines for service time of data base VM and modeling VM are respectively:

DB.VM service time = 5.44 x USERS - 1.46 seconds

M.VM service time = 1.13 x USERS - .94 seconds

FIGURE 3.5

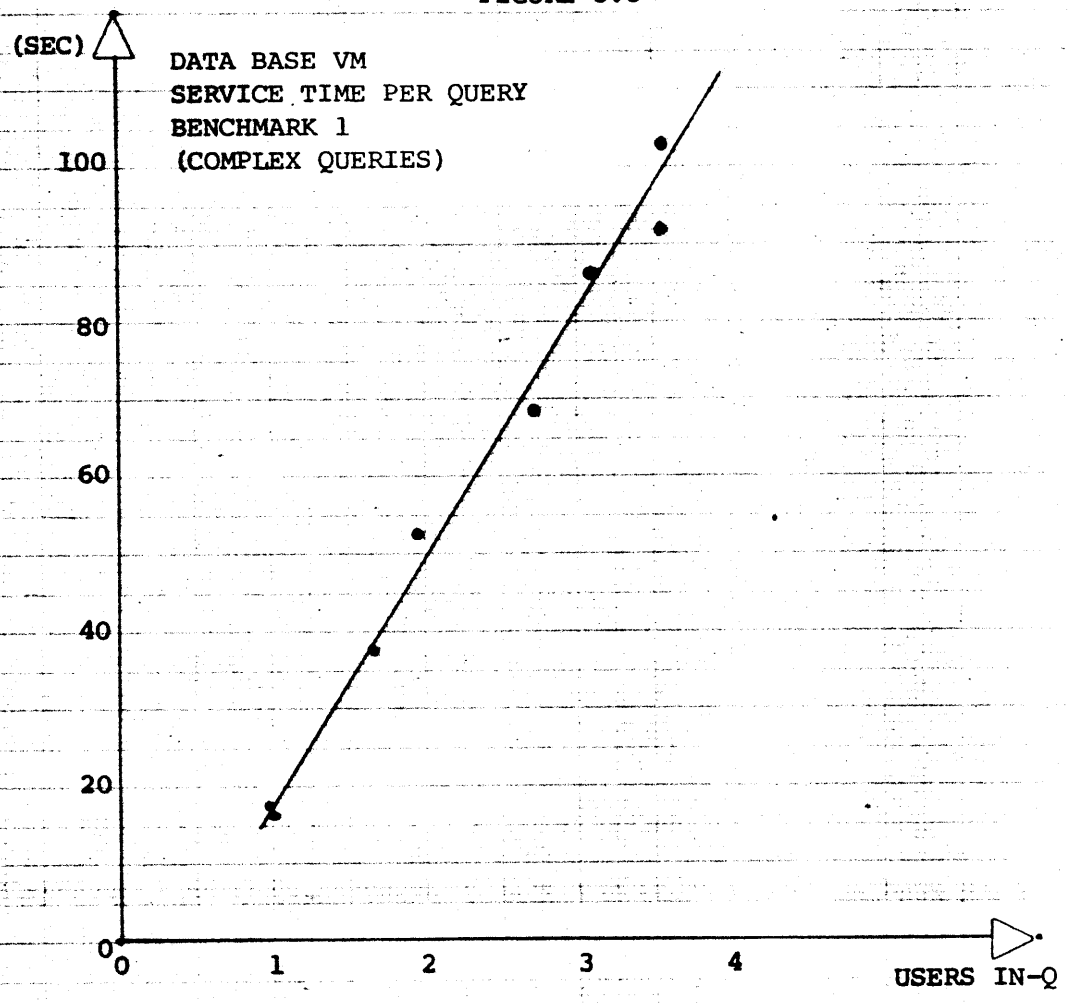


FIGURE 3.6

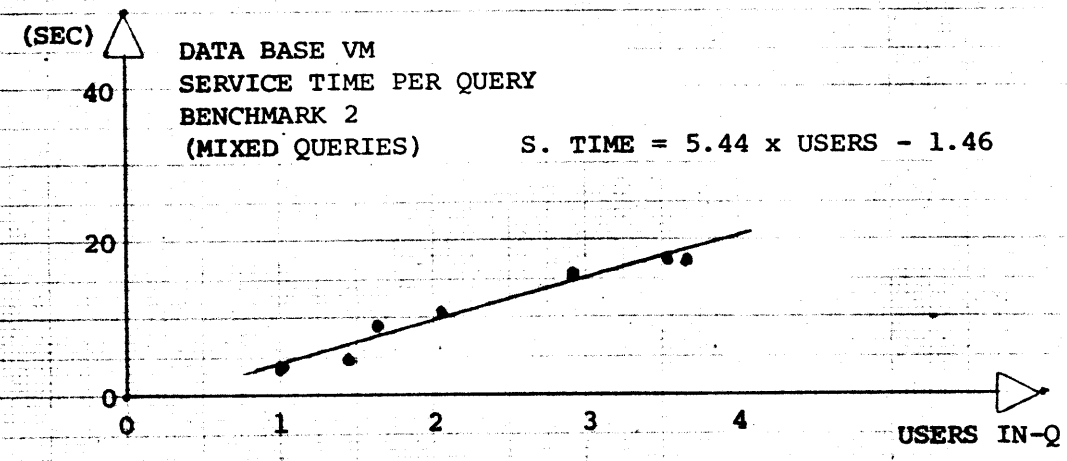
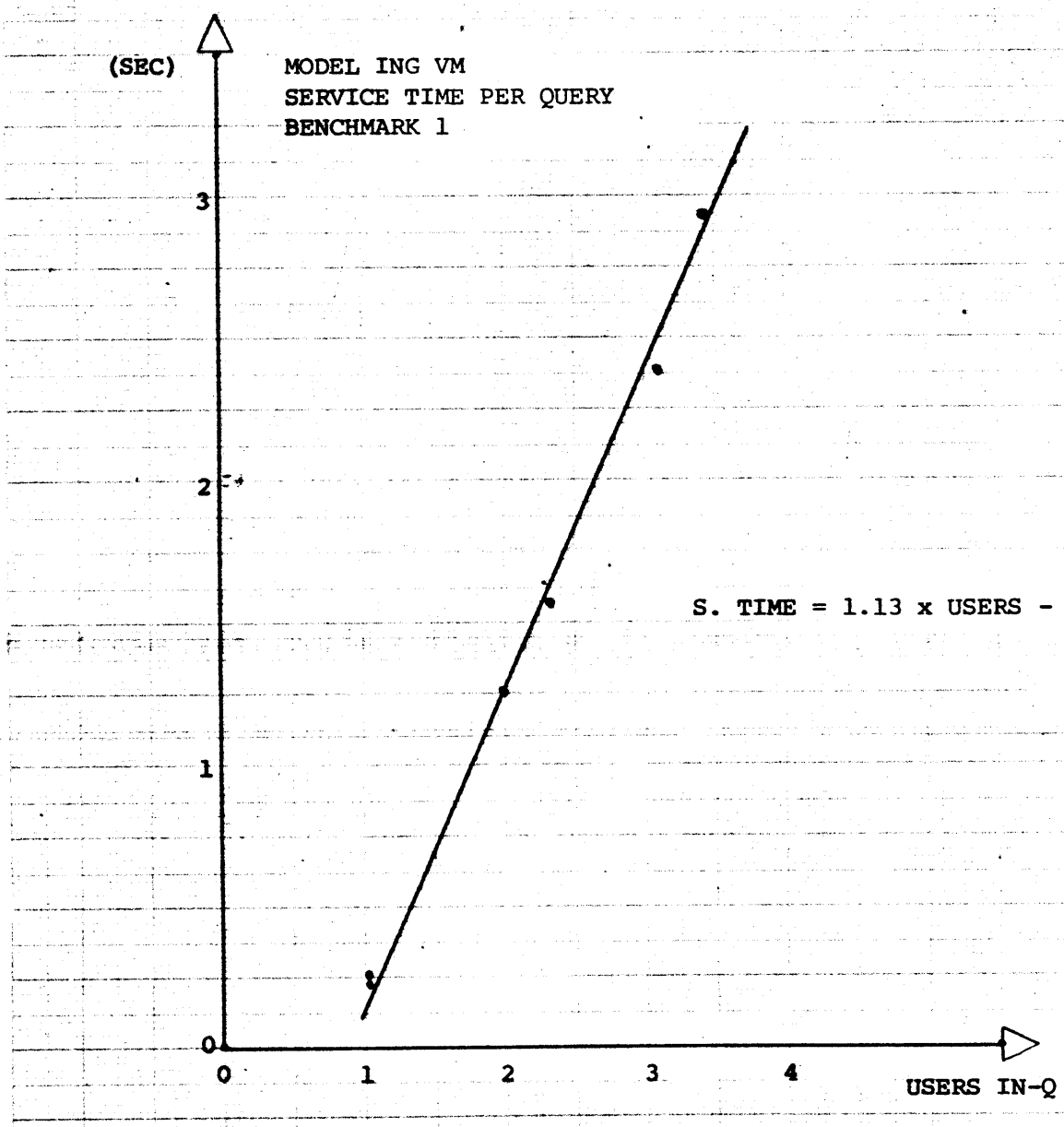


FIGURE 3.7


(SEC)
MODEL ING VM
SERVICE TIME PER QUERY
BENCHMARK 1
3
2
1
0
0
1
2
3
4
USERS IN-Q
S. TIME = 1.13 x USERS - .94

We assumed the service times to be exponentially distributed with a mean given by the above equations.

We may drop now the assumption previously made about all VM's receiving the same amount of CPU processing power. The modeling VM's requiring less system resources will tend to run in Q1 (high priority) and the data base machine will tend to run in Q2 (low priority, bigger time slice). These differences in priority are already accounted for in the above equations, since they result from direct measurement of the service times.

3.3 Simulation

We used GPSS [15,16,17] to simulate the model in Figure 3.4. Appendix J presents the coding of the program.

The program logs one user per hour into the system up to 10 users. It prints the statistics at the end of each simulated hour (time unit = 1/100 of a second). Appendix K presents a sample of these statistics.

We run several times the simulation of 1 to 10 active GMIS users, simulating various ways of improving GMIS performance. The results are presented and analyzed in the next section.

The holding times in each entity of the GPSS program are determined as follows:

Think time - a sample is drawn each time from an Erlang distribution of second order with mean equals to 5.94 seconds.

Modeling VM's running time per query - a sample is drawn

from an exponential distribution with mean equals to the result obtained by plugging the number of unblocked VM's into the equation presented in the previous section.

Data base VM running time per query is determined in the same way as the modeling VM's. But since the holding time in the data base VM is usually much longer than in the modeling VM's, the program reajusts several times the running speed of the data base VM during each query, to make it sensitive to the state changes of the other VM's.

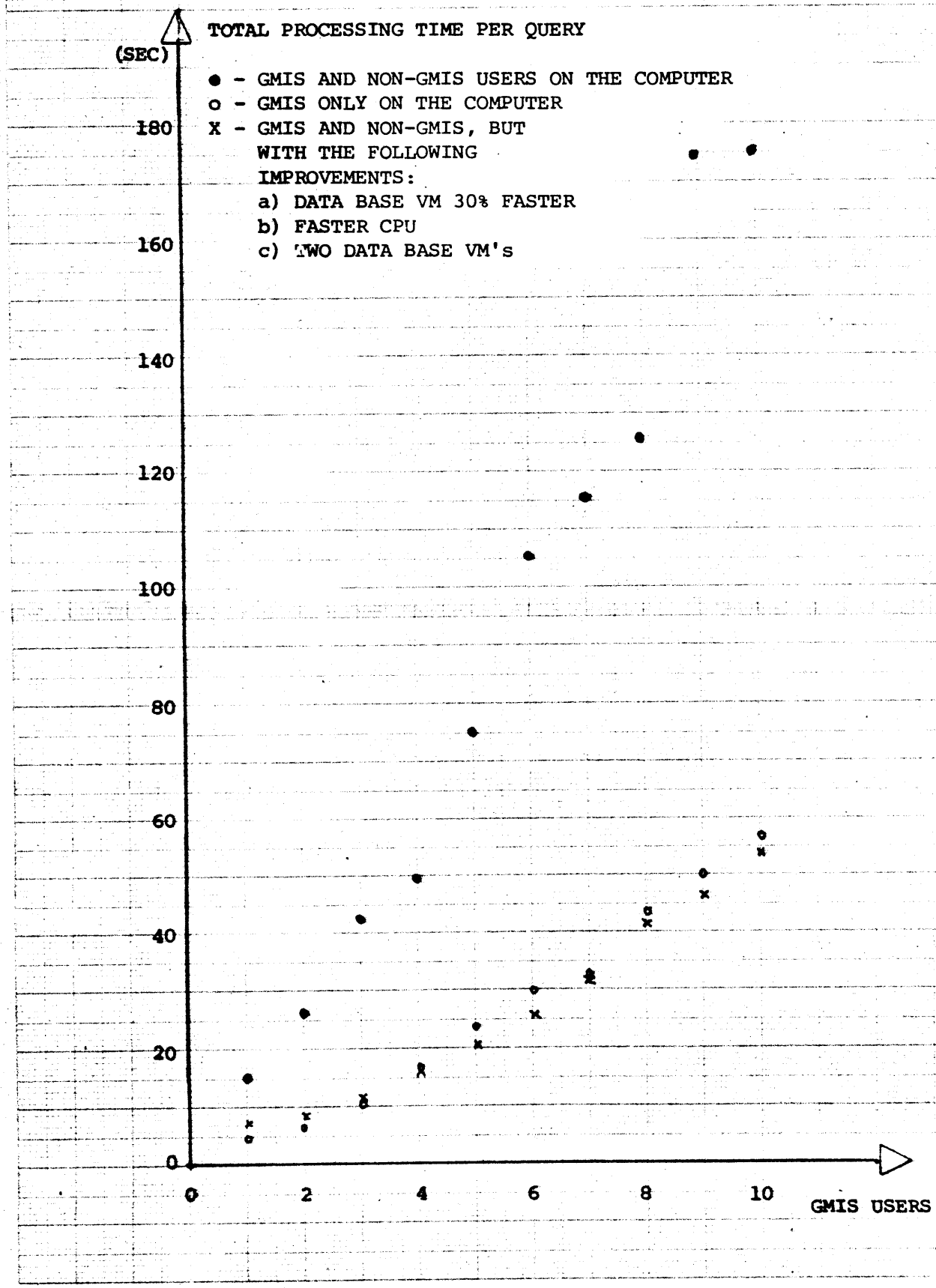
3.4 Analysis of the Simulation Results

3.4.1 The effect of having GMIS sharing the computer with other applications

Figure 3.8 presents the results of the first three simulation runs. It shows for each run the total processing time per query versus the number of GMIS users being simulated. Total processing time per query is the data base VM running time plus the time the query was in the queue, waiting to be processed.

In the first run, GMIS users shared the computer with non-GMIS users. There was an average of 2 non-GMIS users in-Q (running all the time), what corresponds to a load of about 8 to 10 active users on the computer. This, plus the load correspondent to GMIS users bring the number of active users on the computer up to 18 to 20. This is the peak load of a typical day on the IBM Cambridge Scientific Center installation (see VMA report in Appendix A). This load is typical only in

FIGURE 3.8


TOTAL PROCESSING TIME PER QUERY
(SEC)
● - GMIS AND NON-GMIS USERS ON THE COMPUTER
o - GMIS ONLY ON THE COMPUTER
X - GMIS AND NON-GMIS, BUT WITH THE FOLLOWING IMPROVEMENTS:
a) DATA BASE VM 30% FASTER
b) FASTER CPU
c) TWO DATA BASE VM's
180
160
140
120
100
80
60
40
20
0
0
2
4
6
8
10
GMIS USERS

its total because it is very unusual to find more than one GMIS user simultaneously on the system.

The second plot on Figure 3.8 shows the results when GMIS users only were simulated.

The third plot shows the results where again there was 8 to 10 non-GMIS users on the computer but with the following improvements being simulated simultaneously:

a) the programming of the data base VM was optimized to run 30% faster than before.

b) the CPU model 158, with basic machine cycle equals 115 manoseconds, was substituted by a model 168, with a basic machine cycle of 80 manoseconds [18]. Note: as pointed out in the previous chapter, CPU is not the only resource that may constrain performance. Thus what we mean by changing the CPU for a faster model is an upgrade in all system components (main memory, I/O and paging subsystems) so that the new system can outperform the old one in the proportion of their basic machine cycle times.

c) GMIS architecture was changed to have two data base VM's running in parallel.

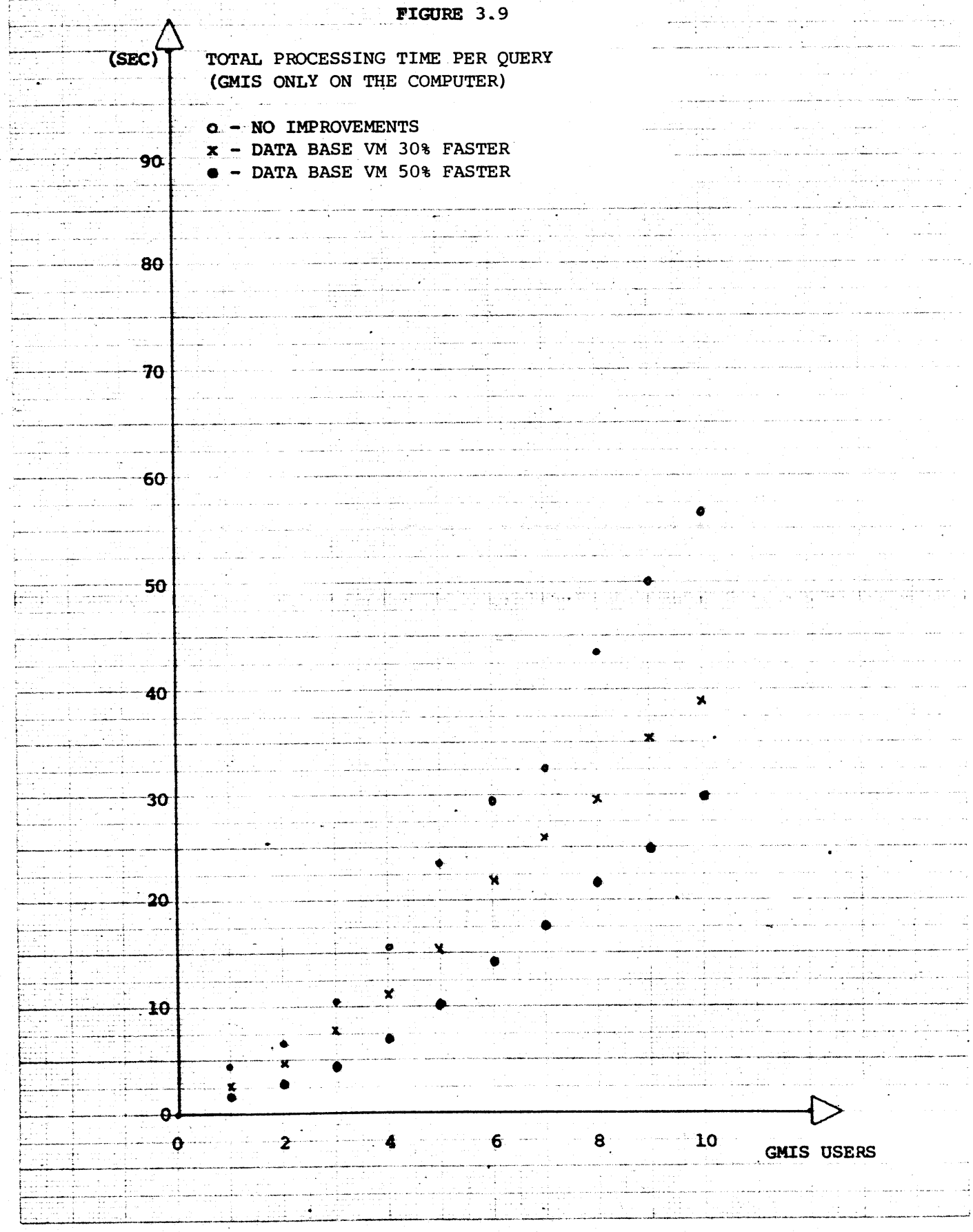
By analyzing the simulation results (Figure 3.8) for the runs described above, we can see that GMIS performance is very sensitive to the number of simultaneous GMIS users. In fact, it takes slightly more than 10 times longer to process a query when there are 10 GMIS users than when there is only one. GMIS performance is also very sensitive to the number of

non-GMIS users sharing system resources with GMIS users. It takes about three times as much to process a query when there is 8 to 10 active non-GMIS users on the computer than it takes when GMIS is running alone on the system. The combination of improvements we simulated with 8 to 10 active non-GMIS users had the net effect of making GMIS run as fast as it did when we simulated GMIS running alone with no improvements. This was a mere coincidence, but it does show that a combination of improvements can drastically reduce processing time per query.

3.4.2 The effect of optimizing the data base VM programming

Figure 3.9 presents the results obtained in two simulation runs where we simulated different degrees of optimization in the programming of the data base VM (data base VM 30% and 50% faster than before) with GMIS users only on the computer. Figure 3.9 also shows the plot for GMIS with no improvements so we can compare with the other plots. We can see in these plots that a certain percentage improvement on the data base VM coding causes total processing time per query to drop by about the same percentage. This makes sense since the data base VM is the bottleneck on the stream of queries and any improvement on the bottleneck should directly affect the overall performance, up to the point where this element is no longer constraining the system.

FIGURE 3.9


(SEC)
TOTAL PROCESSING TIME PER QUERY
(GMIS ONLY ON THE COMPUTER)
o - NO IMPROVEMENTS
x - DATA BASE VM 30% FASTER
● - DATA BASE VM 50% FASTER
90
80
70
60
50
40
30
20
10
0
0
2
4
6
8
10
GMIS USERS

3.4.3 The effect of having more than one data base VM

Figures 3.10 and 3.11 present the results for the simulation of 10 GMIS users being served by 1 to 10 data base VM's running in parallel. We can see that in the case where GMIS was simulated alone in the computer, there was no significant gain by having more than one data base machine. This result is reasonable because with 10 GMIS users on the system, the data base machine will tend to be running all the time with the modeling VM's blocked while one of them is being served. If we introduce, for instance, a second data base machine into the system, both will tend to be running all the time, but now with half of the speed because system resources are being shared by the two of them. In other words, we doubled the number of servers but we also doubled the time each server takes to process a transaction, so the expected waiting time in the system does not change. The same reasoning can be applied for more than two servers.

The results for the case where the 10 GMIS users were sharing the computer with 8 to 10 non-GMIS users (two non-GMIS users in-Q all the time) were different. When there is only one data base VM, it is running with one third of the speed it would run if GMIS was alone in the computer. This is because there is in average two non-GMIS users permanently sharing the computer resources with the data base VM. So, when we add a second data base VM, GMIS will be receiving half of the computer resources instead of only a third. In this case we

FIGURE 3.10

TOTAL PROCESSING TIME PER QUERY

(GMIS AND NON-GMIS USERS)

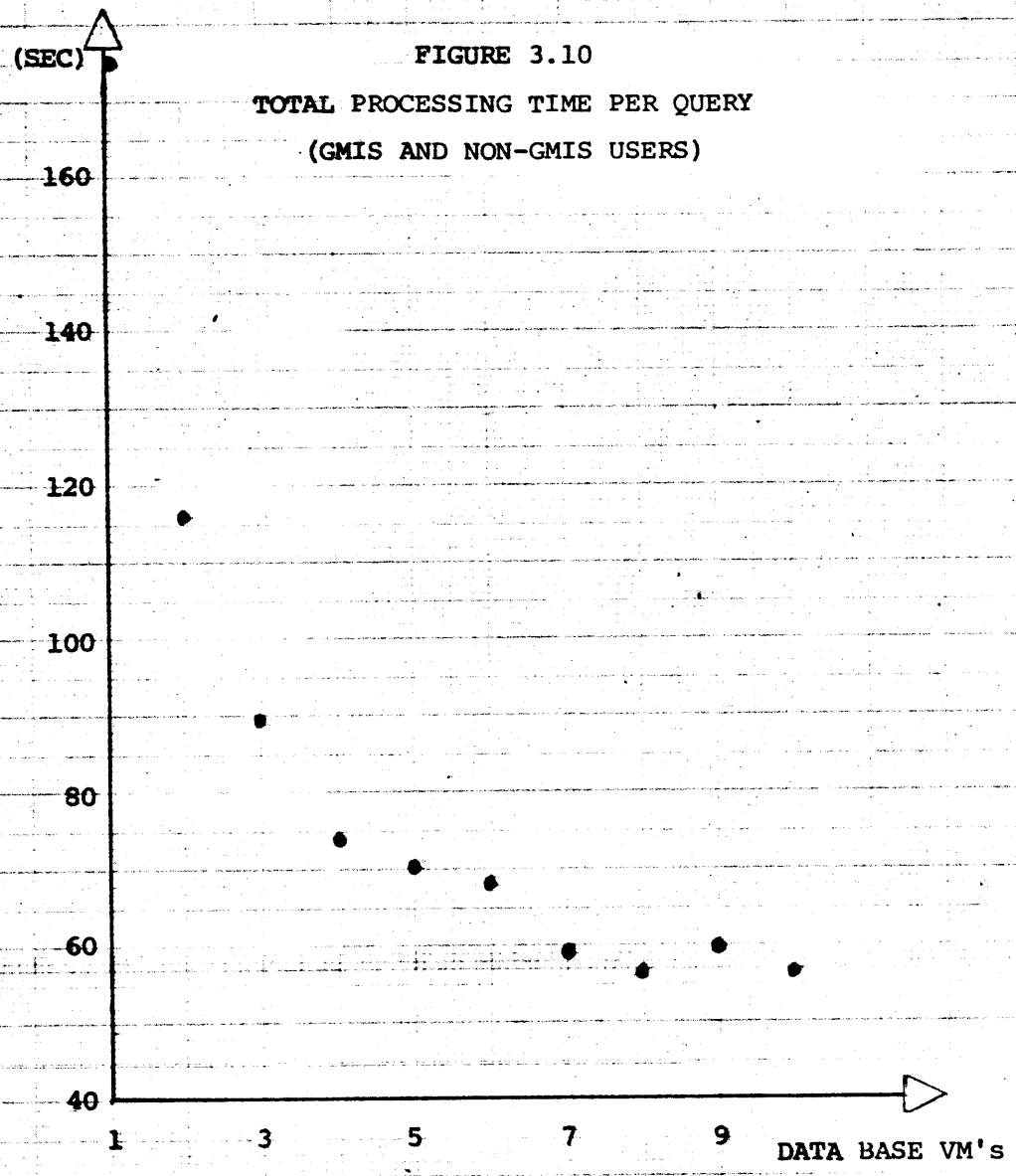
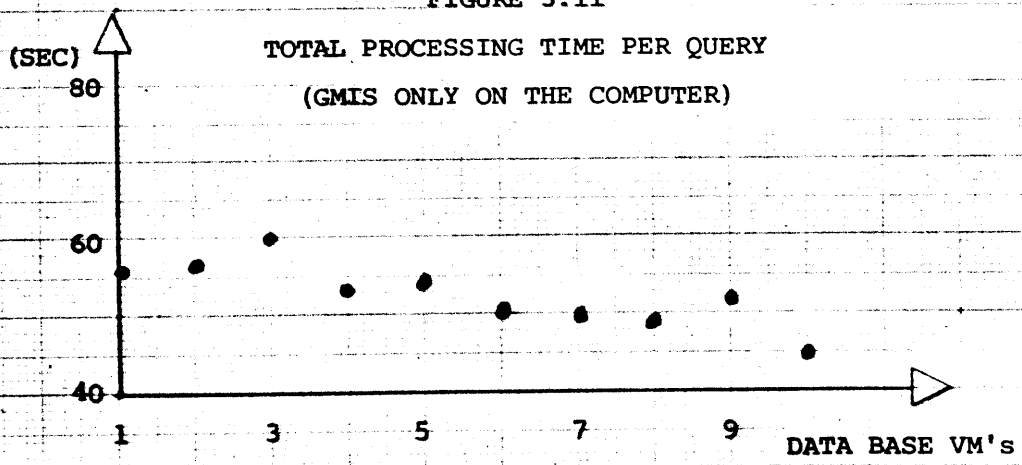


FIGURE 3.11

TOTAL PROCESSING TIME PER QUERY

(GMIS ONLY ON THE COMPUTER)



doubled the number of servers without increasing the service time in the same proportion. Thus the expected waiting time in the system is decreased. When we increased the number of data base VM's from 1 to 10 in this simulation run we had decreasing marginal gains in performance. This can be understood by the same line of reasoning, since when we have 1, 2, 3 ... 10 data base VM's, they receive respectively 1/3, 2/4, 3/5 .... 10/12 of the system resources.

In conclusion, there is no gain in performance by having several data base VM's when GMIS is running alone in the computer. Actually there would be a loss in performance due to the overhead of co-ordinating the data base VM's (this overhead is not included in the model). There is a gain in performance by having several data base VM's, when GMIS is not alone in the computer. But this gain in response time for GMIS users is obtained at the expense of all the other users. A similar effect may be obtained in a much simpler and more efficient way; that is, increasing the priority of GMIS users in relation to other users.

3.4.4 The effect of using a faster CPU

In another run we simulated GMIS running (only GMIS in the computer) in an upgraded installation that has a faster CPU, model 168. For comparison, we present in Figure 3.12 the results of this simulation together with the results obtained by simulating a data base VM 30% faster. The new

NUMBER OF GMIS USERS ON COMP.	FASTER CPU (168)		DB-VM 30% FASTER	
	*NO. OF QUERIES	**TOTAL PROC. TIME (SEC.)	*NO. OF QUERIES	**TOTAL PROC. TIME (SEC.)
1	414	2.72	391	2.70
2	663	4.52	624	4.66
3	764	7.21	713	7.76
4	817	10.50	753	11.18
5	808	15.06	771	15.21
6	798	19.67	717	21.99
7	817	23.51	730	25.87
8	773	29.92	754	29.62
9	816	32.57	750	35.14
10	852	34.86	758	38.34

FIGURE 3.12

*Number of queries that were completed during one simulated hour.

**Total processing time per query in seconds, includes DB-VM processing time plus time in the queue waiting to be processed.

CPU is roughly 30% faster than the old one.

The results of these two different ways of improving performance were very close, despite the fact that when we use a faster CPU we increase both the speed of the modeling VM's and the speed of the data base VM. The data base VM is the bottleneck element in the system, and the increase in its speed accounted for most of the overall gain in performance while the increase in the speed of the modeling VM's had little impact.

In Chapter 2, we concluded that the CPU was well dimensioned given the present load conditions. But an upgraded computer using a faster CPU model may be necessary in the future, in case the load on the system increases due to the implementations of new GMIS applications, for instance.

3.5 Model Validation

A complete validation of the simulation results is very hard to do. The first difficulty is that all simulation runs represent hypothetical situation in the future; that is, heavier usage of GMIS, code optimization of GMIS, faster CPU and change in GMIS architecture. So validation of the type predicted-versus-real will eventually be possible only in the future.

The second difficulty is that we had only one account on the computer available to us during this work. This made it impossible to measure GMIS performance with more than one VM

running the benchmarks simultaneously, to check against the simulation preduction.

What we did then to make sure the simulation results are coherent with the assumptions we made was to calculate the number of queries that should have been made by one GMIS user in every simulation run and compare it with the actual number of queries simulated for one user. The number of queries that should have been simulated can be calculated as follows:

$$\text{Number of Queries} = \frac{\text{Simulated time}}{\text{Expected time per query}} =$$

$$\frac{3600}{E[\text{think time}]+2E[\text{model VM serv. time}]+E[\text{D.Base VM serv. time}]}$$

The expected number of queries for each run is shown in Figure 3.13 together with the actual number of queries taken from the simulation statics for each run.

The discrepancies shown in Figure 3.13 are well within the accuracy of the model and we can say that the simulation results are coherent with the assumptions made about the parameters in the model.

Simulation	Expected # of Queries	Actual # of Queries
1) GMIS and non-GMIS users, no improvements	140	134
2) GMIS only, no improvements	349	340
3) GMIS only DB-VM 30% faster	395	391
4) GMIS only DB-VM 50% faster	433	448
5) GMIS and non-GMIS users, faster CPU, DM-VM 30% faster	217	215
6) GMIS only, faster CPU	401	414

FIGURE 3.13

CHAPTER 4

Conclusions and Recommendations for Further Research

This paper has presented the analysis and evaluation of several ways of improving GMIS performance.

We showed that changing the system architecture so that several queries are processed simultaneously by several data base VM's decreases GMIS response time, but increases response time for non-GMIS users. A priority scheme can achieve a similar result in a much simpler and more efficient way.

We showed further that the use of a faster CPU model as a means to improve GMIS performance is only advisable if the load on the computer increases in the future, and in this case other components of the system will probably have to be upgraded too. The system is presently dimensioned to work efficiently even when the load is at its peak during the day.

The optimization of the data base VM programming seems to be the alternative with the highest payoff since this is the bottleneck element in GMIS.

We recommend that further research is done to identify and optimize the most often used routines in the data base VM. Further research on the SEQUEL structure may also lead to better performance. For instance, SEQUEL does not have a JOIN command. It handles joins through nested queries, which is very

inefficient, if many rows of the tables involved have to be accessed.

REFERENCES

1. Gutentag, L.M.: "GMIS: Generalized Management Information System--An Implementation Description," Master's thesis submitted to the Alfred P. Sloan School of Management, May 1975.

2. Donnovan, J.J. and H.D. Jacoby: "GMIS: An Experimental System for Data Management and Analysis," MIT-Sloan School of Management Working Paper No. MIT-EL-75-011 WP, September 1975.

3. Bard, Y.: "Performance Analysis of Virtual Memory Time-Sharing Systems," IBM Systems Journal 14, No. 4, 1975.

4. Bard, Y.: "Experimental Evaluation of System Performance," IBM Systems Journal 12, No. 3, 1973.

5. Bard, Y.: "Performance Criteria and Measurement for Time Sharing Systems," IBM Systems Journal 10, No. 3, 1971.

6. IBM: "Virtual Machine Performance Tool (VMPT)," Order No. 7720-2693-1, April 1974.

7. IBM: "IBM Virtual Machine Facility /370: Introduction," Order No. GC20-1800-5, October 1975.

8. IBM: "IBM Virtual Machine Facility /370: System Programmer's Guide," Order No. GC20-1807-3, January 1975.

9. IBM: "IBM Virtual Machine Facility /370: Control Program (CP) Program Logic," Order No. SY20=0880, 1972.

10. Little, J.D.C.: "A Proof of the Queuing Formula $L = \lambda\omega$," Operations Research 9, 1961, pp. 383-387.

11. Hillier, F.S. and G.J. Liberman: "Introduction to Operations Research," Holden-Day, San Francisco, 1967.

12. Saaty, T.L.: "Elements of Queueing Theory, with Applications," McGraw-Hill, New York, N. Y., 1961.

13. Callaway, P.H.: "Performance Measurement Tools for VM /370," IBM Systems Journal 2, 1975.

14. IBM: "VM /370 Predictor," Cambridge Scientific Center, Preliminary Copy, January 1976.

15. Schriber, T.J.: "Simulation Using GPSS," John Wiley & Sons, 1974.

16. IBM: "General Purpose Simulation System/360 User's Manual," Order Number G420-0326-4, January 1970.

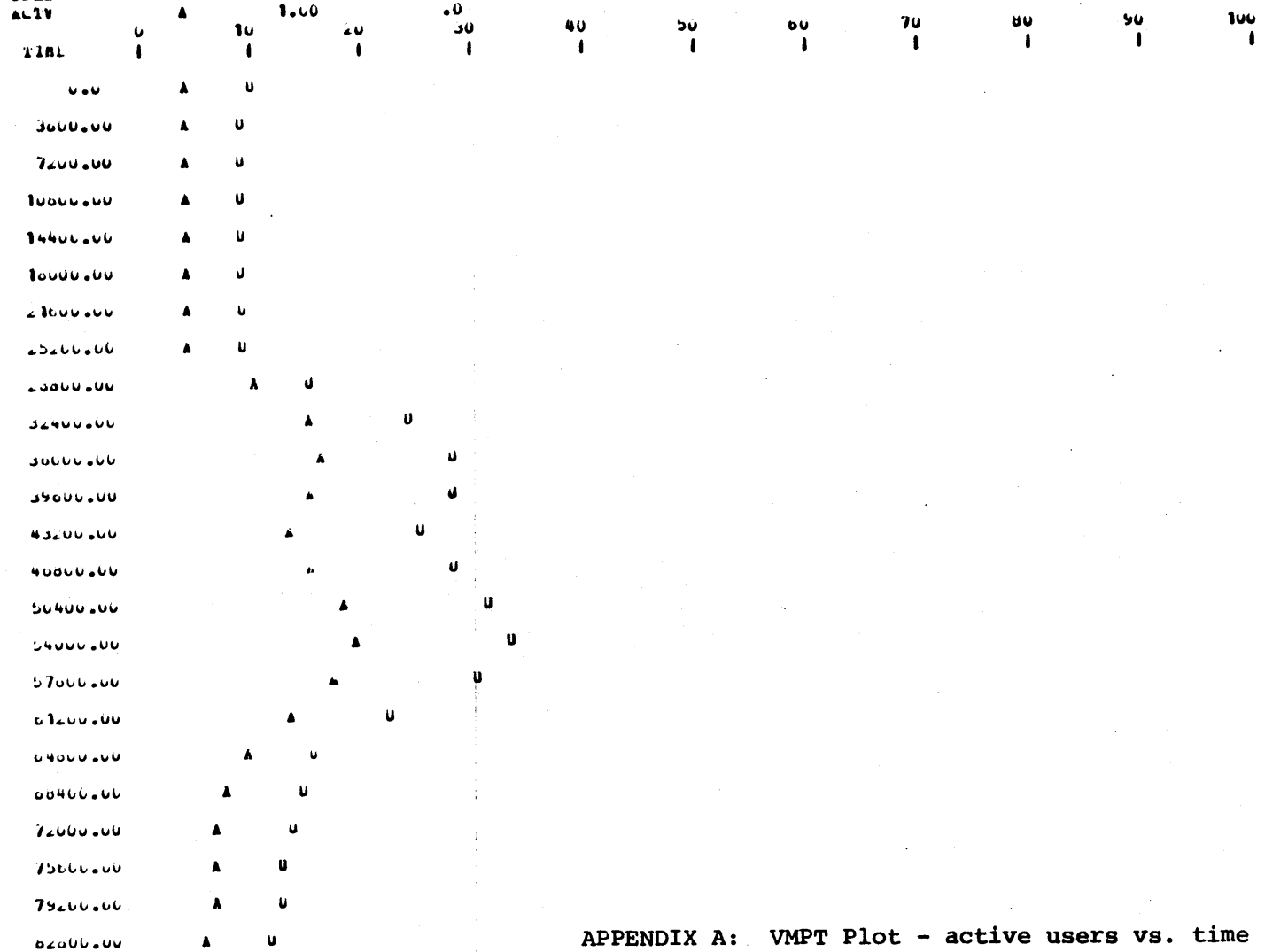
17. Gordon, G.: "System Simulation," Prentice-Hall, 1969.

18. IBM: "IBM System /370 - System Summary," Order Number GA22-7001-4, February 1975.

VMCSC5 VM ACTIVITY BY HOUR OF DAY

TO OBTAIN TRUE VALUE OF A VARIABLE, DIVIDE PLOTTED VALUE BY SCALE FACTOR AND SUBTRACT ORIGIN SHIFT

VARIABLE	SYMBOL	SCALE FACTOR	ORIGIN SHIFT
USER	U	1.00	.0
ACTV	A	1.00	.0

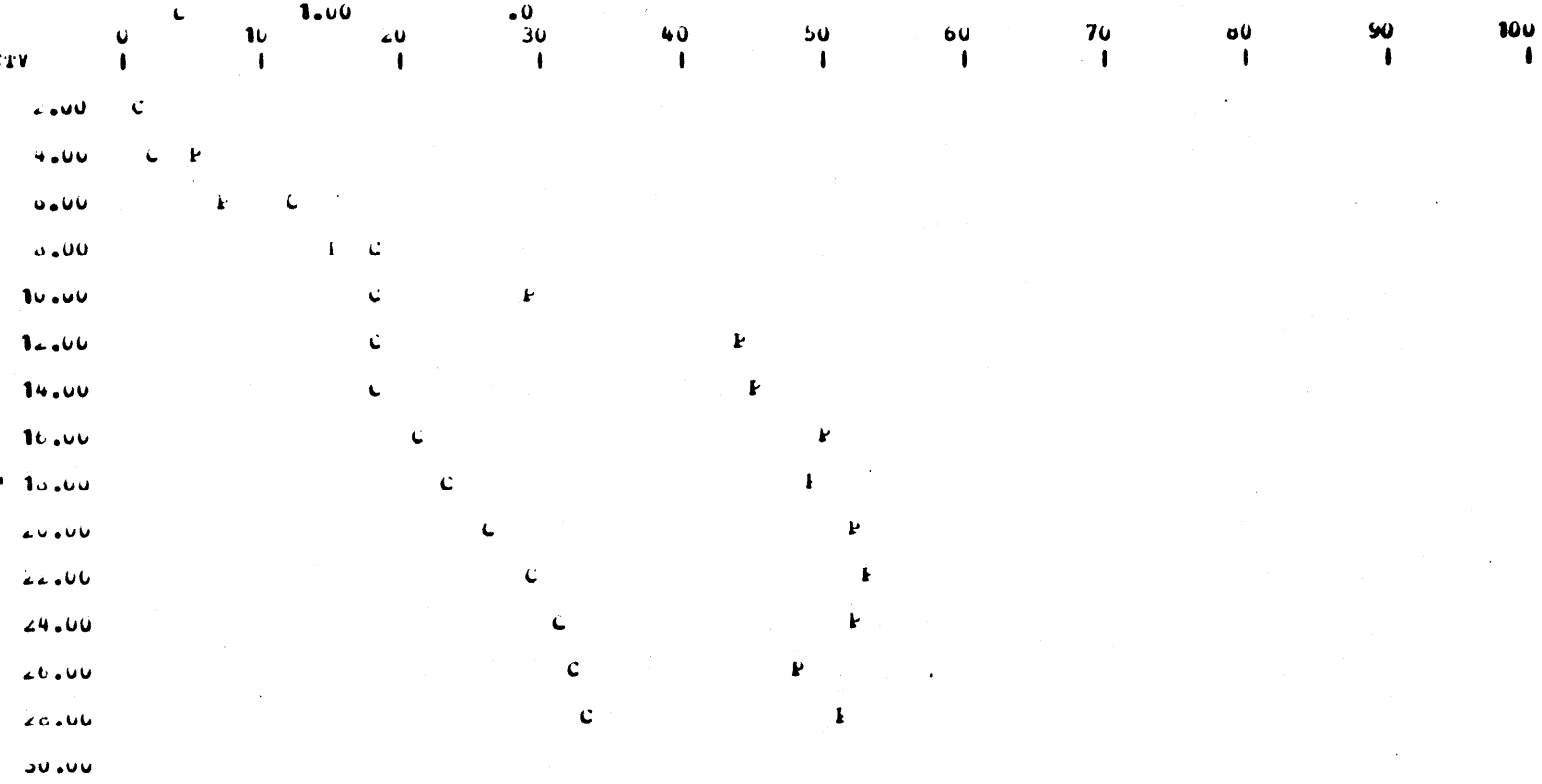


APPENDIX A: VMPT Plot - active users vs. time of the day

VMCSC5 VM ACTIVITY BY ACTIVE USERS

TO OBTAIN TRUE VALUE OF A VARIABLE, DIVIDE PLOTTED VALUE BY SCALE FACTOR AND SUBTRACT ORIGIN SHIFT

VARIABLE	SYMBOL	SCALE FACTOR	ORIGIN SHIFT
PROB	P	1.00	.0
CP	C	1.00	.0

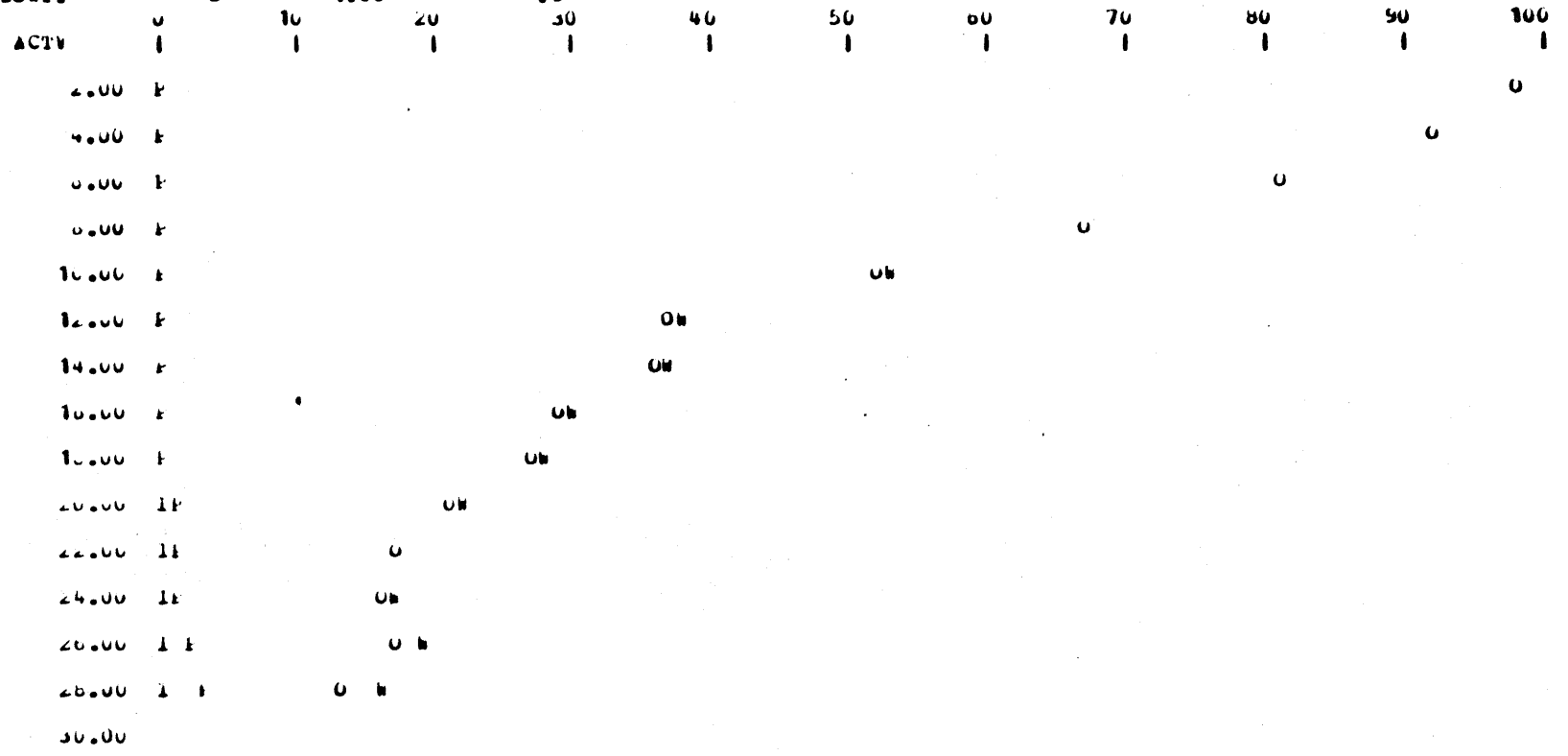


APPENDIX B
VMPT Plot - percent CPU utilization vs. active users

VMCSC5 VM ACTIVITY BY ACTIVE USERS

TO OBTAIN TRUE VALUE OF A VARIABLE, DIVIDE PLOTTED VALUE BY SCALE FACTOR AND SUBTRACT ORIGIN SHIFT

VARIABLE	SYMBOL	SCALE FACTOR	ORIGIN SHIFT
WAIT	W	1.00	.0
IDLE	I	1.00	.0
EMUL.	E	1.00	.0
TOTL.	T	1.00	.0



APPENDIX C
VMPT Plot - percent CPU wait vs. active users

VMCSC5 VM ACTIVITY BY ACTIVE USERS

TO OBTAIN TRUE VALUE OF A VARIABLE, DIVIDE PLOTTED VALUE BY SCALE FACTOR AND SUBTRACT ORIGIN SHIFT

VARIABLE	SYMBOL	SCALE FACTOR	ORIGIN SHIFT
Q1	1	1.00	.0
Q2	2	1.00	.0
CAND1	A	1.00	.0
CAND2	B	1.00	.0

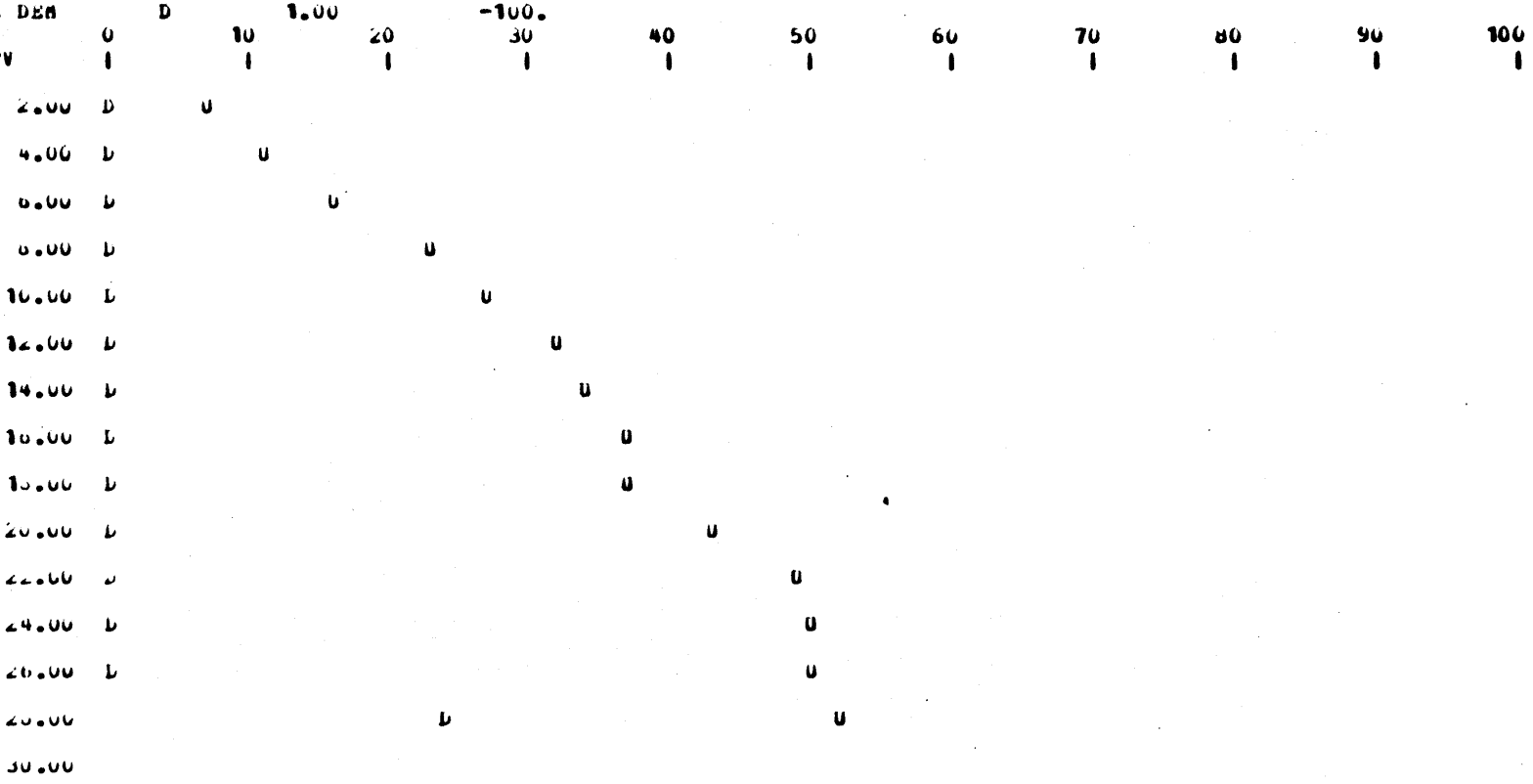
```
          0         10        20        30        40        50        60        70        80        90       100
ACTV      |         |         |         |         |         |         |         |         |         |         |
  2.00  b1
  4.00  b1
  6.00  b2
  8.00  b2
 10.00  b12
 12.00  b12
 14.00  b12
 16.00  b12
 18.00  b12
 20.00  b1 2
 22.00  b1 2
 24.00  Ab1 2
 26.00  b 12
 28.00   b1 2
 30.00
```

APPENDIX D
VMPT Plot - users IN-Q vs. active users

VMCSC5 VM ACTIVITY BY ACTIVE USERS

TO OBTAIN TRUE VALUE OF A VARIABLE, DIVIDE PLOTTED VALUE BY SCALE FACTOR AND SUBTRACT ORIGIN SHIFT

VARIABLE	SYMBOL	SCALE FACTOR	ORIGIN SHIFT
CORE UT.	U	1.00	.0
CORE DEM	D	1.00	-100.



APPENDIX E
VMPT Plot - pageble core utilization vs. active users

IBM PROPERTY

NEEMIS

*AV. PAGING SLOTS 187.0

*SOURCE CPU 0158 02A0

AVERAGE USERS IN CLASS 1.0

TIME OF FIRST REF. 0. LAST REF. 3300. TIME SPAN 3300. SECONDS

TIME IN Q (SEC): COUNTED 2448.195 LOST 4.139

*RATIO OF TRIVIAL TO NONTRIVIAL TRANSACTIONS 0.0

	TRIVIAL TRANSACTIONS	NONTRIVIAL TRANSACTIONS	OTHER	TOTAL
NO. OF TRANSACTIONS	0	41		
TOTAL VIRTUAL TIME	0.0	665.406	0.0	665.406
TOTAL OVHD TIME	0.0	999.227	0.0	999.227
ACTIVE USERS	0.0	0.956	0.0	0.956

	TRIVIAL TRANSACTION (AV. PER TRANS.)	NONTRIVIAL TRANSACTION (AV. PER TRANS.)	OTHER (AV. PER SEC. VIRTUAL TIME)	TOTAL (AV. PER SEC. VIRTUAL TIME)
*PROB (VIRTUAL) TIME	0.0	16.22940		
*CP OVHD TIME	0.0	24.37137	0.0	1.50168
*VIO	0.0	449.585	0.0	27.702
*VIRT. LINES PRINTED	0.0	0.0	0.0	0.0
*VIRT. CARDS READ	0.0	0.0	0.0	0.0
*VIRT. CARDS PUNCHED	0.0	0.976	0.0	0.060
*WORKING SET	0.0	132.348	0.0	132.348
*PAGING INDEX	0.0	24.583	0.0	1.515
*PAGE READS	0.0	23.244	0.0	1.432
*THINK TIME	0.0	17.193		
RESPONSE TIME	0.0	59.713	0.0	3.679
TIME IN ELIGIBLE LIST	0.0	0.001	0.0	0.000
TIME IN Q	0.0	59.712	0.0	3.679

STARRED (*) ITEMS ARE REQUIRED FOR INPUT TO PREDICTOR MODEL

APPENDIX F: VM/370 Predictor Report - user workload

```
***IBM PROPERTY***

                    SARGENTO MARCH 8  8PM

****     ANALYSIS OF SYSTEM PERFORMANCE    ****

ELAPSED TIME  3326.052    SECONDS

CPU UTILIZATION, PERCENT OF ELAPSED TIME:
 TOTAL CPU       PROB          CP       TOTAL WAIT       IDLE       PAGE         IO
     80.25      47.74       32.52            19.75        0.0       0.11      19.64

 PG RD     PG WRT
  /SEC       /SEC
   0.9        0.1

AVERAGE ACTIVE USERS     6.1    PER   60. SEC PERIOD

AVERAGE LOGGED USERS    12.9

AVERAGE: PGABLE PAGES       PERC. CORE UT.     CORE SAT. FACTOR       RESERVED PAGES         SHARED PAGES
                425.8                 35.0                  1.9                  0.0                 46.5

AV.     Q1         Q2       Q1 C       Q2 C
     0.20       2.12       0.00       0.00

AV PAGE SLOTS ALLOCATED TO DRUM    722.    DISK      38.

PERCENT PAGE ALLOCATION: DRUM    95.0    DISK     5.0

PERCENT PAGE READS (ESTIMATED): DRUM    99.9    DISK     0.1

IO DEVICE ACTIVITY
DEV  TYP     SER IO/SEC
100 0402 CPDRM1   0.23    101 0402          0.0    102 0402          0.0    103 0402          0.0    104 0402          0.0
105 0402          0.0     106 0402          0.0    107 0402          0.0    230 0440 VMDSK1   0.54   231 0440 19ACMS   0.0
232 0440 CPCMS4   0.0     233 0440 MEYER1   0.0    234 0440 CPCMS6   0.0    235 0440 CMS370   0.0    236 0440 CPCMS7   0.0
237 0440 CPCMS1   0.0     350 0410 VUSR04   0.46   351 0410 VUSR05   1.32   352 0410 VUSR01   0.99   353 0410 VUSR07   0.03
354 0410          0.0     355 0410          0.0    450 0410 VUSR03   2.18   451 0410 VUSR02   3.15   452 0410 VUSR06   0.00
453 0410 VMDSK2   0.37    454 0410          0.0    455 0410          0.0    580 0810          0.0    581 0810          0.0
582 0410          0.0     583 0810          0.0    584 0810          0.06   585 0810          0.01

OVERHEAD ADJUSTMENT FACTOR  1.058

TOTAL ACTIVE USERS    9
```

APPENDIX G
VM/370 Predictor Report - system performance

VMCSC5 VM ACTIVITY BY ACTIVE USERS

STATISTICS FOR VARIABLE THINK

ACTV RANGE G.E.	L.T.	OBS.	TIME (HRS)	AVERAGE	ST. DEV.	MINIMUM	MAXIMUM
2.000	4.000	244	66.89	3.7465	3.9564	.08606	50.418
4.000	6.000	633	163.50	7.2330	8.4188	.44995	117.17
6.000	8.000	172	33.37	8.5594	5.1069	1.8525	25.294
8.000	10.000	151	27.46	7.0515	4.9199	1.9773	40.387
10.000	12.000	205	19.32	6.4093	3.3790	1.1106	22.096
12.000	14.000	403	24.83	6.0739	2.4479	1.9042	23.500
14.000	16.000	563	20.10	5.7120	1.7618	1.8210	14.746
16.000	18.000	726	20.83	5.5779	1.7279	1.8830	17.054
18.000	20.000	520	10.84	5.4571	1.3729	2.5011	10.343
20.000	22.000	357	11.39	5.4214	1.5160	2.4955	10.993
22.000	24.000	200	5.59	5.1968	1.0439	3.0169	8.2647
24.000	26.000	90	3.26	5.1761	1.2015	2.4394	10.745
26.000	28.000	63	2.30	5.5001	1.1078	2.9144	7.0302
28.000	30.000	17	0.72	5.7430	1.4965	3.7091	8.9000
30.000	32.000	2	0.14	6.4884	1.4512	5.4423	7.4945

APPENDIX H
VMPT Report - think time vs. active users

FILE: CONTROL TOTAL A CAMBRIDGE SCIENTIFIC CENTER

```
SELECT * FROM  INTEGRTY;
SELECT * FROM  DOMCAT;
SELECT * FROM  CATALOG;
SELECT TABLENAM FROM INDEX;
SELECT TOT(YEAR1962) FROM TABLE1;
SELECT TOT(YEAR1967) FROM TABLE1;
SELECT TOT(YEAR1973) FROM TABLE1;
SELECT TOT(YEAR1962) FROM TABLE2;
SELECT TOT(YEAR1967) FROM TABLE2;
SELECT TOT(YEAR1973) FROM TABLE2;
SELECT TOT(YEAR1962) FROM TABLE3;
SELECT TOT(YEAR1967) FROM TABLE3;
SELECT TOT(YEAR1973) FROM TABLE3;
SELECT TOT(YEAR1962) FROM TABLE4;
SELECT TOT(YEAR1967) FROM TABLE4;
SELECT TOT(YEAR1973) FROM TABLE4;
SELECT TOT(YEAR1962) FROM TABLE5;
SELECT TOT(YEAR1967) FROM TABLE5;
SELECT TOT(YEAR1973) FROM TABLE5;
SELECT TOT(YEAR1962) FROM TABLE7;
SELECT TOT(YEAR1967) FROM TABLE7;
SELECT TOT(YEAR1973) FROM TABLE7;
SELECT TOT(YEAR1962) FROM TABLE8;
SELECT TOT(YEAR1967) FROM TABLE8;
SELECT TOT(YEAR1973) FROM TABLE8;
SELECT TOT(YEAR1962) FROM TABLE9;
SELECT TOT(YEAR1967) FROM TABLE9;
SELECT TOT(YEAR1973) FROM TABLE9;
SELECT TOT(YEAR1962) FROM TABLE10;
SELECT TOT(YEAR1967) FROM TABLE10;
SELECT TOT(YEAR1973) FROM TABLE10;
SELECT TOT(YEAR1962) FROM TABLE11;
SELECT TOT(YEAR1967) FROM TABLE11;
SELECT TOT(YEAR1973) FROM TABLE11;
SELECT TOT(YEAR1962) FROM TABLE13A;
SELECT TOT(YEAR1967) FROM TABLE13A;
SELECT TOT(YEAR1973) FROM TABLE13A;
SELECT * FROM INDUSTRY;
SELECT * FROM SOURCE;
SELECT * FROM CLASS;
SELECT TOT(YEAR1962) FROM TABLE14;
SELECT TOT(YEAR1971) FROM TABLE14;
SELECT TOT(YEAR1962) FROM TABLE15A;
SELECT TOT(YEAR1966) FROM TABLE15A;
SELECT TOT(YEAR1971) FROM TABLE15B;
SELECT TOT(YEAR1966) FROM TABLE16;
SELECT TOT(YEAR1971) FROM TABLE16;
SELECT TOT(YEAR1962) FROM TABLE17;
SELECT TOT(YEAR1971) FROM TABLE17;
SELECT TOT(YEAR1962) FROM TABLE18;
SELECT TOT(YEAR1967) FROM TABLE18;
SELECT TOT(YEAR1973) FROM TABLE18;
SELECT TOT(YEAR1962) FROM TABLE20;
SELECT TOT(YEAR1971) FROM TABLE20;
SELECT * FROM PRODSOR;
```

APPENDIX I: List of Queries of Benchmark 2

FILE: CONTROL TOTAL A CAMBRIDGE SCIENTIFIC CENTER

```
SELECT TOT(AMOUNT) FROM IMP74 WHERE DATE = 7401
AND FUELTYPE = 'RESIDUAL FUEL OIL';
SELECT TOT(AMOUNT) FROM IMP74 WHERE DATE = 7406
AND STATE = 'MA' AND FUELTYPE = 'RESIDUAL FUEL OIL';
SELECT TOT(AMOUNT) FROM IMP73 WHERE DATE = 7304 AND
FUELTYPE = 'RESIDUAL FUEL OIL' AND STATE = 'RI';
SELECT TOT(AMOUNT) FROM IMP73 WHERE DATE = 7309
AND FUELTYPE = 'RESIDUAL FUEL OIL' AND STATE IN
('CT','MA','ME','NH','RI','VT');
SELECT DEALER,DEALLOC FROM IMP72 WHERE DATE = 7201
AND FUELTYPE = 'RESIDUAL FUEL OIL' AND STATE = 'MA';
SELECT TOT(AMOUNT) FROM IMP72 WHERE DATE = 7206 AND STATE = 'MA'
AND FUELTYPE = 'RESIDUAL FUEL OIL';
SELECT COST FROM IMP71;
SELECT TOT(AMOUNT) FROM IMP71 WHERE DATE = 7107 AND FUELTYPE =
'RESIDUAL FUEL OIL' AND STATE = 'ME';
SELECT TOT(COST) FROM BOSTON WHERE MONTH = 7401;
SELECT TOT(COST) FROM BOSTON WHERE MONTH = 7412;
SELECT AVG(VOLUME) FROM CARSALES WHERE DATE = 7409;
SELECT MAX(MPG) FROM CARSALES;
SELECT MODEL,YEAR,MPGCITY,MPGHWY FROM MILEAGE WHERE
MODEL = 'CHEVROLET' AND YEAR = 1974;
SELECT MPGAVG FROM MILEAGE WHERE MODEL = 'CHEVROLET'
AND YEAR = 1974;
SELECT * FROM CARDATA;
SELECT TOT(TOTAL1972) FROM DEGDAY_STATION WHERE
STATE = 'CONNECTICUT';
SELECT TOT(G1970) FROM HOUSEUNITS WHERE STATE = 'MA';
SELECT TOT(O1970) FROM HOUSEUNITS WHERE STATE = 'MA';
SELECT TOT(L1970) FROM HOUSEUNITS WHERE STATE = 'MA';
SELECT TOT(FLTUNITS1975) FROM HOMECONSTRUCTION WHERE STATE = 'MA';
SELECT TOT(EIGHTROOM) FROM ROOMSINHOUSE WHERE STATE = 'MA';
SELECT TOT(EIGHTROOM) FROM ROOMSINHOUSE WHERE STATE = 'CT';
SELECT TOT(UNITS2) FROM UNITSINSTRUCTURE WHERE STATE = 'MA';
SELECT TOT(UNITS2) FROM UNITSINSTRUCTURE WHERE STATE = 'CT';
SELECT TOT(NINEORMOREPEOPLE) FROM PERSONSINHOUSE WHERE
STATE = 'MA';
SELECT TOT(FUELOIL) FROM FUELBURNEDBYHOME;
SELECT TOT(ELECTRICITY) FROM FUELBURNEDBYHOME;
SELECT TOT(UTILITYGAS) FROM FUELBURNEDBYHOME;
SELECT TOT(TWENTY5KANDABOVE) FROM INCOME;
SELECT TOT(UNHEATED) FROM HEATINGEQUIPMENT;
SELECT TOT(BUILT39_EARLIER) FROM STRUCTUREBUILT;
SELECT MAX(STORIES3_HIGHER) FROM STORIES;
SELECT TOT(CONCRETESLAB) FROM STRUCTURETYPE;
SELECT TOT(TOTAL1973) FROM DEGDAYCOUNTY WHERE STATE = 'MA';
SELECT TOT(TOTAL1972) FROM DEGDAYCOUNTY WHERE STATE = 'MA';
SELECT TOT(MARCH75) FROM DEGDAYAVSTATION ;
SELECT TOT(FEBRUARY75) FROM DEGDAYAVCOUNTY ;
SELECT TOT(DECEMBER74) FROM DEGDAYDEVSTATION;
SELECT TOT(JANUARY75) FROM DEGDAYDEVCOUNTY
WHERE STATE = 'NH';
```

APPENDIX I (continued)

```
BLOCK                                                                                                CARD
NUMBER   *LOC   OPERATION  A,B,C,D,E,F,G              COMMENTS                                       NUMBER
         *                                                                                             1
         *      FUNCTIONS   VARIABLES  TABLES                                                        2
         *                                                                                            3
          XPDIS FUNCTION    RN1,C24              EXPONENTIAL DISTRIBUTION MEAN = 1                   4
         0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/.6,.915/.7,1.2/.75,1.38                          5
         .8,1.6/.84,1.83/.88,2.12/.9,2.3/.92,2.52/.94,2.81/.95,2.99/.96,3.2                          6
         .97,3.5/.98,3.9/.99,4.6/.995,5.3/.998,6.2/.999,7/.9998,8                                    7
         *                                                                                           8
          SMD   FVARIABLE   113*V$ACTIV-94     SERVICE TIME MODEL VM                                 9
          STVM  FVARIABLE   (136*V$ACTIV-36)*P2/1000 SERVICE TIME D. BASE VM                         10
          THINK FVARIABLE   297*FN$XPDIS+297*FN$XPDIS       THINK TIME                               11
          ACTIV VARIABLE    S$MVM+S$TVM+2                                                            12
          XP    FVARIABLE   1000*FN$XPDIS                                                            13
         *                                                                                           14
          RTIME TABLE       MP1,1500,1500,22                                                         15
         *                                                                                           16
          TVM   STORAGE     1               START WITH ONE SEQUEL MACHINE                            17
         *                                                                                           18
         *      MODEL SEGMENT 1                                                                      19
1                GENERATE    360000,,1,10,1,2,F  LOGIN 10 USERS 1 PER HOUR                         20
2         BACK   ENTER       USER            USER STARTS THINKING                                   21
3                ADVANCE     V$THINK                                                                22
4                LEAVE       USER                                                                   23
5                ENTER       MVM             MODELING VM STARTS RUNNING                             24
6                ADVANCE     V$SMD,FN$XPDIS                                                         25
7                LEAVE       MVM                                                                    26
8                ASSIGN      2,V$XP                                                                 27
9                MARK        1                                                                      28
10               QUEUE       RDR             QUERY SENT TO TVM READER                               29
11               ENTER       TVM             TVM STATS RUNNING                                      30
12               DEPART      RDR                                                                    31
13               ADVANCE     V$STVM                                                                 32
14               ADVANCE     V$STVM                                                                 33
15               ADVANCE     V$STVM                                                                 34
16               ADVANCE     V$STVM                                                                 35
17               LEAVE       TVM                                                                    36
18               TABULATE    RTIME                                                                  37
19               ENTER       MVM             MODEL STARTS 2ND RUN                                   38
20               ADVANCE     V$SMD,FN$XPDIS                                                         39
21               LEAVE       MVM                                                                    40
22               TRANSFER    ,BACK           START NEXT QUERY                                       41
         *                                                                                           42
         *      MODEL SEGMENT 2                                                                      43
23               GENERATE    360000,,,,2     RUN FOR ONE HOUR & PRINT STATISTICS                    44
24               TERMINATE   1                                                                      45
         *                                                                                           46
         *                                                                                           47
         *      SIMULATE 1 TO 10 USERS  - GMIS ONLY ON THE COMPUTER                                  48
          ACTIV VARIABLE    S$MVM+S$TVM                                                              49
                START       1                                                                        50
                RESET                                                                                51
                START       1                                                                        52
                RESET                                                                                53
                START       1                                                                        54
                RESET                                                                                55
```

APPENDIX J: GPSS Code for the Model

```
        START     1                                                          56
        RESET                                                                57
        START     1                                                          58
        RESET                                                                59
        START     1                                                          60
        RESET                                                                61
        START     1                                                          62
        RESET                                                                63
        START     1                                                          64
        RESET                                                                65
        START     1                                                          66
        RESET                                                                67
        START     1                                                          68
*                                                                            69
*       SIMULATE 10 USERS WITH  1 TO 10  DATA BASE MACHINES                  70
 TVM    STORAGE   2                                                         71
        RESET                                                                72
        START     1                                                          73
 TVM    STORAGE   3                                                         74
        RESET                                                                75
        START     1                                                          76
 TVM    STORAGE   4                                                         77
        RESET                                                                78
        START     1                                                          79
 TVM    STORAGE   5                                                         80
        RESET                                                                81
        START     1                                                          82
 TVM    STORAGE   6                                                         83
        RESET                                                                84
        START     1                                                          85
 TVM    STORAGE   7                                                         86
        RESET                                                                87
        START     1                                                          88
 TVM    STORAGE   8                                                         89
        RESET                                                                90
        START     1                                                          91
 TVM    STORAGE   9                                                         92
        RESET                                                                93
        START     1                                                          94
 TVM    STORAGE   10                                                        95
        RESET                                                                96
        START     1                                                          97
*                                                                            98
*       SIMULATE 1 TO 10 USERS USING SEQUEL 30 % FASTER                      99
        CLEAR                                                               100
 STVM   FVARIABLE 70/100*(136*V$ACTIV-36)*P2/1000                           101
 TVM    STORAGE   1                                                        102
        START     1                                                         103
        RESET                                                               104
        START     1                                                         105
        RESET                                                               106
        START     1                                                         107
        RESET                                                               108
        START     1                                                         109
        RESET                                                               110
        START     1                                                         111
        RESET                                                               112
```

APPENDIX J (continued)

```
        START     1                                                      113
        RESET                                                            114
        START     1                                                      115
        RESET                                                            116
        START     1                                                      117
        RESET                                                            118
        START     1                                                      119
        RESET                                                            120
        START     1                                                      121
*                                                                        122
*       SIMULATE 1 TO 10 USERS USING SEQUEL 50 % FASTER                  123
        CLEAR                                                            124
 STVM   FVARIABLE  50/100*(136*VSACTIV-36)*P2/1000                       125
        START     1                                                      126
        RESET                                                            127
        START     1                                                      128
        RESET                                                            129
        START     1                                                      130
        RESET                                                            131
        START     1                                                      132
        RESET                                                            133
        START     1                                                      134
        RESET                                                            135
        START     1                                                      136
        RESET                                                            137
        START     1                                                      138
        RESET                                                            139
        START     1                                                      140
        RESET                                                            141
        START     1                                                      142
        RESET                                                            143
        START     1                                                      144
*                                                                        145
*       SIMULATE 1 TO 10 USERS USING FASTER CPU MODEL 168                146
        CLEAR                                                            147
 SMD    FVARIABLE  70/100*(113*VSACTIV-94)                               148
 STVM   FVARIABLE  70/100*(136*VSACTIV-36)*P2/1000                       149
        START     1                                                      150
        RESET                                                            151
        START     1                                                      152
        RESET                                                            153
        START     1                                                      154
        RESET                                                            155
        START     1                                                      156
        RESET                                                            157
        START     1                                                      158
        RESET                                                            159
        START     1                                                      160
        RESET                                                            161
        START     1                                                      162
        RESET                                                            163
        START     1                                                      164
        RESET                                                            165
        START     1                                                      166
        RESET                                                            167
        START     1                                                      168
*                                                                        169
```

APPENDIX J (continued)

```
*       SIMULATE 1 TO 10 USERS, SEQUEL 30 % FASTER, CPU=168, W/ 2 TVM'S   170
        CLEAR                                                             171
ACTIV   VARIABLE   S$MVM+S$TVM+2                                          172
SMD     FVARIABLE  70/100*(113*V$ACTIV-94)                                173
STVM    FVARIABLE  70/100*70/100*(136*V$ACTIV-36)*P2/1000                 174
TVM     STORAGE    2                                                      175
        START      1                                                      176
        RESET                                                             177
        START      1                                                      178
        RESET                                                             179
        START      1                                                      180
        RESET                                                             181
        START      1                                                      182
        RESET                                                             183
        START      1                                                      184
        RESET                                                             185
        START      1                                                      186
        RESET                                                             187
        START      1                                                      188
        RESET                                                             189
        START      1                                                      190
        RESET                                                             191
        START      1                                                      192
        RESET                                                             193
        START      1                                                      194
        END                                                               195
```

RELATIVE CLOCK 360000 ABSOLUTE CLOCK 360000

BLOCK COUNTS

BLOCK	CURRENT	TOTAL	BLOCK	CURRENT	TOTAL	BLOCK	CURRENT	TOTAL	BLOCK	CURRENT	TOTAL	BLOCK	CURRENT	TOTAL
1	0	1	11	0	339	21	0	339						
2	0	340	12	0	339	22	0	339						
3	1	340	13	0	339	23	0	1						
4	0	339	14	0	339	24	0	1						
5	0	339	15	0	339									
6	0	339	16	0	339									
7	0	339	17	0	339									
8	0	339	18	0	339									
9	0	339	19	0	339									
10	0	339	20	0	339									

APPENDIX K
Sample GPSS Output Statistics

STORAGE	CAPACITY	AVERAGE CONTENTS	AVERAGE UTILIZATION	ENTRIES	AVERAGE TIME/TRAN	CURRENT CONTENTS	MAXIMUM CONTENTS
MVM	2147483647	.035	.000	678	18.948		1
TVM	1	.389	.389	339	413.333		1
USER	2147483647	.575	.000	340	608.917	1	1

APPENDIX K (continued)

QUEUE	MAXIMUM CONTENTS	AVERAGE CONTENTS	TOTAL ENTRIES	ZERO ENTRIES	PERCENT ZEROS	AVERAGE TIME/TRANS	$AVERAGE TIME/TRANS	TABLE NUMBER	CURRENT CONTENTS
ROR	1	.000	339	339	100.0	.000	.000		

$AVERAGE TIME/TRANS = AVERAGE TIME/TRANS EXCLUDING ZERO ENTRIES

APPENDIX K (continued)

TABLE RTIME

ENTRIES IN TABLE	MEAN ARGUMENT	STANDARD DEVIATION	SUM OF ARGUMENTS	
339	413.333	434.000	140120.000	NON-WEIGHTED

UPPER LIMIT	OBSERVED FREQUENCY	PER CENT OF TOTAL	CUMULATIVE PERCENTAGE	CUMULATIVE REMAINDER	MULTIPLE OF MEAN	DEVIATION FROM MEAN
1500	327	96.46	96.4	3.5	3.629	2.503
3000	11	3.24	99.7	.2	7.258	5.960
4500	1	.29	100.0	.0	10.887	9.416

REMAINING FREQUENCIES ARE ALL ZERO

APPENDIX K (continued)