A SIMULATION STUDY OF ALTERNATIVES TO

UPGRADING LARGE COMPUTER SYSTEMS

by

DANIEL E. KREIMER

B.S. University of Missouri, Columbia, 1970

A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

Kansas State University
Manhattan, Kansas

1977

Approved by:

Major Professor

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to thank Drs. Fred Maryanski, Virgil Wallentine and Myron Calhoun for serving on my committee and offering their comments and corrections. A special thanks is given to Dr. Maryanski for his help and guidance in preparing this report.

DAN KREIMER

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

In this chapter some problems associated with the growth of computer systems will be reviewed. These problems are created by the increase in demand for a computer's resources. This increase requires that a system evolve from one configuration to another; that is, the system must be continuously expanded or upgraded to handle a larger workload. Two questions will be considered:

1) What is meant by throughput in a computer system and how is it measured?

2) Can this measure of throughput be used to determine the most cost-effective configuration of a computer system at any moment in its growth cycle?

1.2 BACKGROUND

This project evolved from a study performed for the Naval Education and Training Information Systems Activity (NETISA) by this author and Dr. Fred Maryanski. The purpose of the study was to determine a suitable replacement for a computer system which has reached the limits of its processing capabilities. The system is shown in Figure 1.1.

The hardware consists of an IBM 360/65 with 2.5 MB of real memory. On-line disk storage is equivalent to 36 IBM-3330-type drives. Twelve of these drives are devoted to an on-line data base system. Several tape drives, card readers and printers are available for batch input/output operations. Approximately 150 remote interactive terminals and 10 remote batch terminals are attached through an IBM-3705-compatible communications controller. It is planned that in the future over 500 terminals will be supported with an on-line data base of possibly 30 disk units.

The operating system currently in use is OS/MVT. One fixed partition of 500 KB is used to support a telecommunications monitor, ENVIRON/1 (1), which is marketed by CINCOM INC. This monitor (described in Appendix I) provides on-line data base inquiry/update capability for the remote terminals. The TOTAL data base system (2) is used for data management and BTAM is used for terminal access.

Approximately 1.4 MB of memory is devoted to batch processing. An average of 4 batch jobs are run concurrently and the job mix is varied between compute-bound and I/O-bound jobs. Jobs which access the data base have their own separate copies of TOTAL so careful scheduling must be used to avoid conflicts between the batch jobs and the on-line programs.

THIS BOOK CONTAINS NUMEROUS PAGES WITH DIAGRAMS THAT ARE CROOKED COMPARED TO THE REST OF THE INFORMATION ON THE PAGE.

THIS IS AS RECEIVED FROM CUSTOMER.

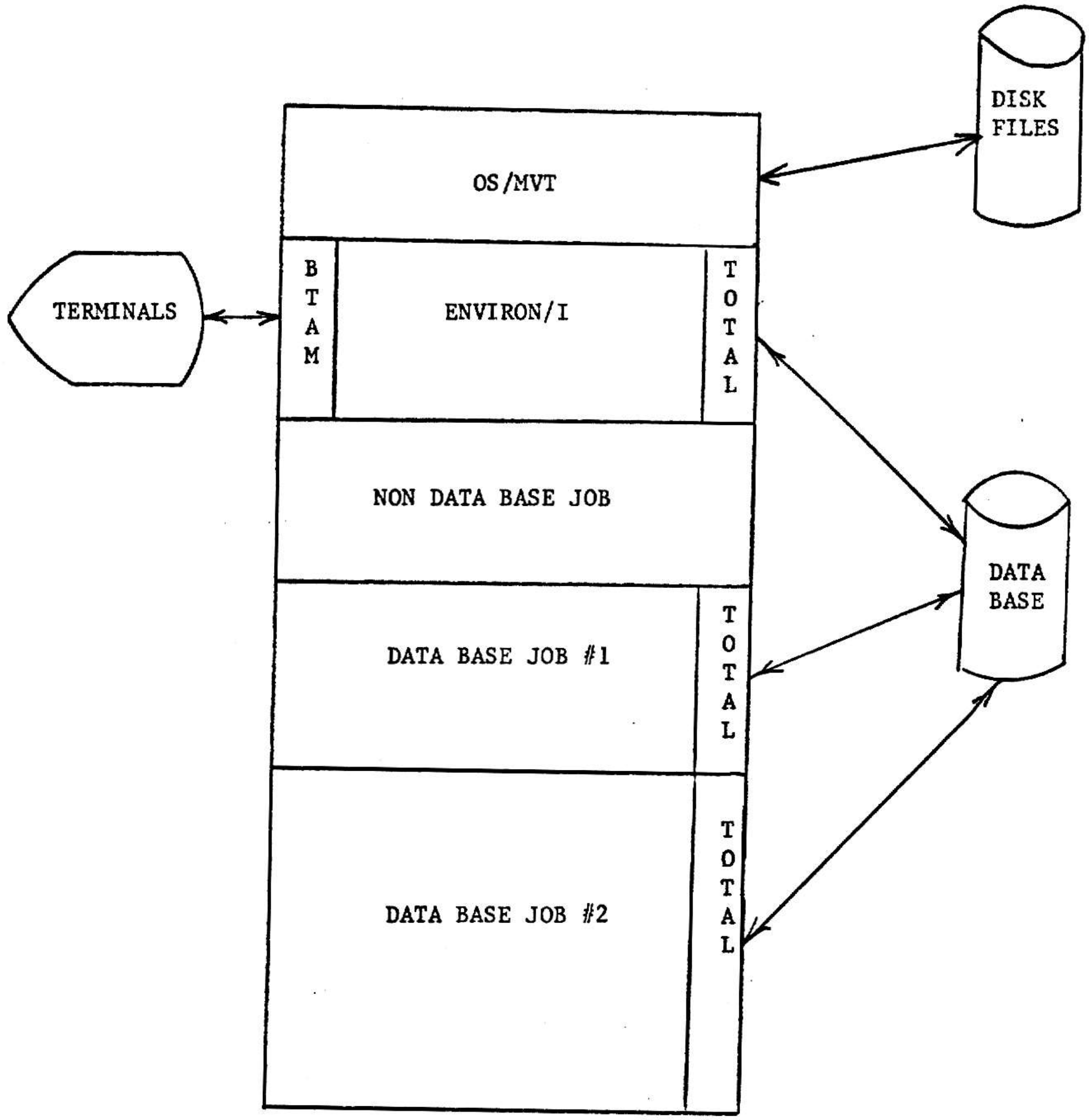


Figure 1.1. NETISA IBM 360/65

1.3 INCREASED DEMAND FOR SERVICES

The NETISA computing center was faced with increasing demands for services. In addition to the increase in the number of terminal users and in the size of the data base, more batch and real-time applications were planned. This increased demand had resulted in slower terminal response time and slower user turnaround on jobs of all types. It was evident that a system with greater throughput was needed.

In making the decision to upgrade or expand, there are few guidelines available. Since the demand on each particular system is different, there are no formulas or rules of thumb to draw upon. The decisions to expand must address the size and speed of memory, number and speed of processors, disks and drums, number and type of data channels. Indeed, the evaluations upon which to base the decisions may become a major part of the cost of expansion.

1.4 APPROACHES TO EVALUATION

Two approaches have been taken in evaluating the capability of a given computer system, analytic and simulation. In some systems the analytic method proves useful. Scherr (3) designed a simple model of the Project MAC system at MIT and Smith (4) constructed a model of a paged time sharing system. However, these models usually lack flexibility and cannot be easily changed to reflect different system configurations.

Simulation models, however, can be quite flexible as

THIS BOOK CONTAINS NUMEROUS PAGE NUMBERS THAT ARE ILLEGIBLE

THIS IS AS RECEIVED FROM THE CUSTOMER

shown by Scherr (3), O'connor (5) and Nielsen (6). These models are also dynamic and are better suited to investigating a constantly changing system. While there are many pitfalls in trying to simulate a computer system (6), simulation seems to be the best tool available at this time to evaluate system configurations and decisions can be based on these evaluations at a reasonable cost.

1.5 ADDITIONAL PROCESSING CAPABILITY

Faced with a backlog of batch and real-time applications, NETISA decided to upgrade to a larger system and let the major computer vendors rely on their experience to choose the correct hardware and software. A multi-processor configuration was planned. This upgrade would solve the immediate and long range problems but at an enormous price. A more cost-effective solution might be to extend the life of the existing system by off-loading some of the functions to additional processors. A simulation study could be used to evaluate these alternatives. The following possibilities exist:

1) Put the telecommunications monitor on a front-end processor. The monitor uses considerable CPU time for message processing, page swapping and communications overhead. In addition, the monitor requires 500 KB of main memory which could be used for new applications.

2) Move the data base system to a back-end processor.

This idea is not new (7); in fact, several data base systems including TOTAL have been put on minicomputers. The amount of additional throughput gained would depend on how heavily the DBMS was being utilized and also how efficient it was in processing requests. Studies have shown (8) that up to 40% of the CPU time in a large system may be devoted to DBMS processing. This solution seems very attractive when additional CPU time is needed.

3) Both the data base system and the monitor could be put on a front-end processor. It would appear that this solution would help the main processor (more memory, less CPU demand) but possibly not help the monitor.

4) Put the monitor on a front-end processor and the data base system on a back-end processor. This may be an effective but expensive solution. The use of minicomputers could make this solution more cost-effective, however.

1.6 OBJECTIVES

In this study a simulation model of the basic NETISA system and the four variations described above will be created. Each system will provide batch processing, telecommunications services and data base management. The parameters on the basic system will be adjusted so that the system resources are saturated and the throughput of each model will be measured to determine the advantages and disadvantages of that particular configuration. This will demonstrate how simulation is an effective tool in evaluating possible system expansion and upgrades.

CHAPTER 2

THE MODEL SYSTEMS

2.1 OVERVIEW

This chapter presents the five models used in this study. The first model represents the basic system which will serve as a reference. The resources (CPU and memory) of this reference system are fully utilized.

The remaining four models are variations of the basic model. Each variation provides additional processing capability by transferring some of the load to an additional processor. The gain in capability will depend on the job mix and terminal loading of the reference model.

2.2 MODEL REQUIREMENTS

If these models are to serve as as a tool in an evaluation they must meet certain requirements:

1) The configuration should be easily adjustable by modification of a few parameters. It should take only minor modification to change configurations.

2) The models should be responsive to changes in parameters. It may be necessary to make simulation runs with various job mixes and memory sizes or number of

terminals.

3) The level of detail should be such as to include the important variables yet still allow the model to run in a reasonable amount of time. This will require that activities an order of magnitude larger or smaller than this level be analyzed in a separate study or studies and the results be input to the main simulation model as parameter values.

4) The various algorithms and components of the model should be modular so they can be easily changed.

5) The models should give meaningful statistics on queue sizes and resource utilization as well as overall system throughput. This gives additional information on how the resources will be utilized in a particular configuration.

2.3 GENERAL ASSUMPTIONS

In all the models the CPU's and operating systems are assumed to be equivalent and comparable to an IBM 360/65 with OS/MVT. This assumption is made to simplify the model and may not reflect an actual system. A minicomputer which is more cost-effective would probably be used for the front-end or back-end processors. However, since the processing power of some mini's are approaching that of large machines, especially in a single application, this assumption is justified.

The amount of main memory allocated to the operating systems is assumed to be constant. This is not strictly true since a change from one configuration to another will usually cause a change in the size of the resident operating system. However, the change should not be large enough to affect the model.

The CPU's are connected by a channel-to-channel adaptor so the transmission delay between machines is negligible. This seems to be a reasonable assumption since the DBMS requests and responses are short and sent at very high speed. A small amount of delay is included in the model for queueing .

2.4 BASIC SYSTEM (I)

The basic system is shown in Figure 2.1. It is similar to the system described in Chapter 1. A fixed partition of 500 KB is allocated to the communications monitor. This monitor is modeled after Cincom's ENVIRON/1 and is typical of many teleprocessing systems which are currently in use. The amount of load that this monitor places on the CPU is highly dependent on the number of terminals on line and the particular applications which are being performed. The parameters for the model are taken from the NETISA study and appear in Chapter 3.

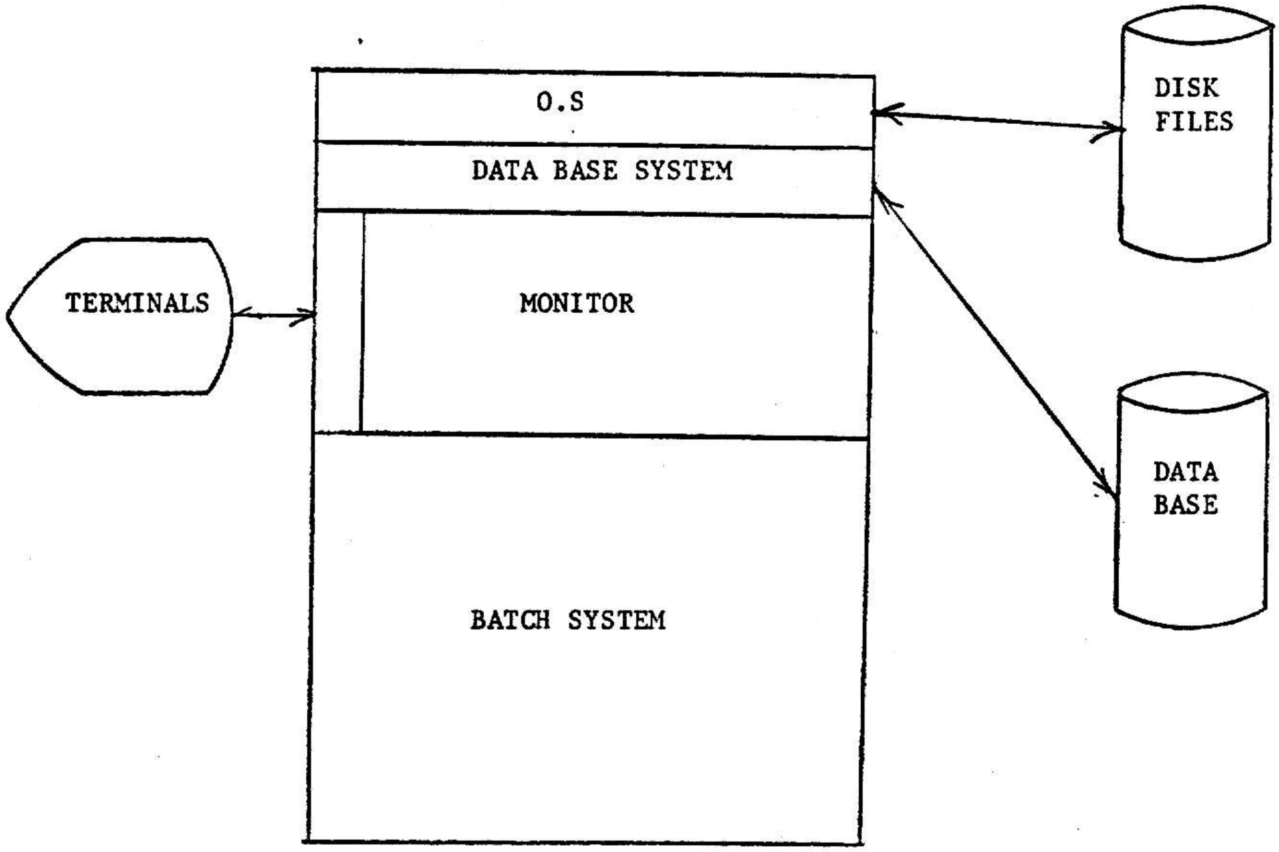


Figure 2.1 Basic System (Model I)

The data base management system model is a memory resident, re-entrant system which services both the monitor and the batch jobs. The amount of memory required for this system is assumed to be negligible compared to the batch and monitor requirements. There are two channels available for disk I/O and 12 disk drives for the data base. The channels may be shared among all the disk units.

The batch system is allocated 1.5 MB of memory. Within this area up to 15 jobs may be run concurrently and all jobs have equal priority when requesting system resources. The job mix is assumed to include both compute-bound and I/O-bound jobs.

2.5 FRONT-END MONITOR (II)

Figure 2.2 shows the first variation of the basic system. This system uses an additional CPU as a front-end to handle the communications monitor. All page swapping, checkpoint recording and communications overhead will be handled by this CPU. Data base requests must be sent to the main CPU for processing.

The advantages of this configuration would appear to be a gain of 500 KB of memory for the main CPU and additional CPU time for monitor message processing and overhead.

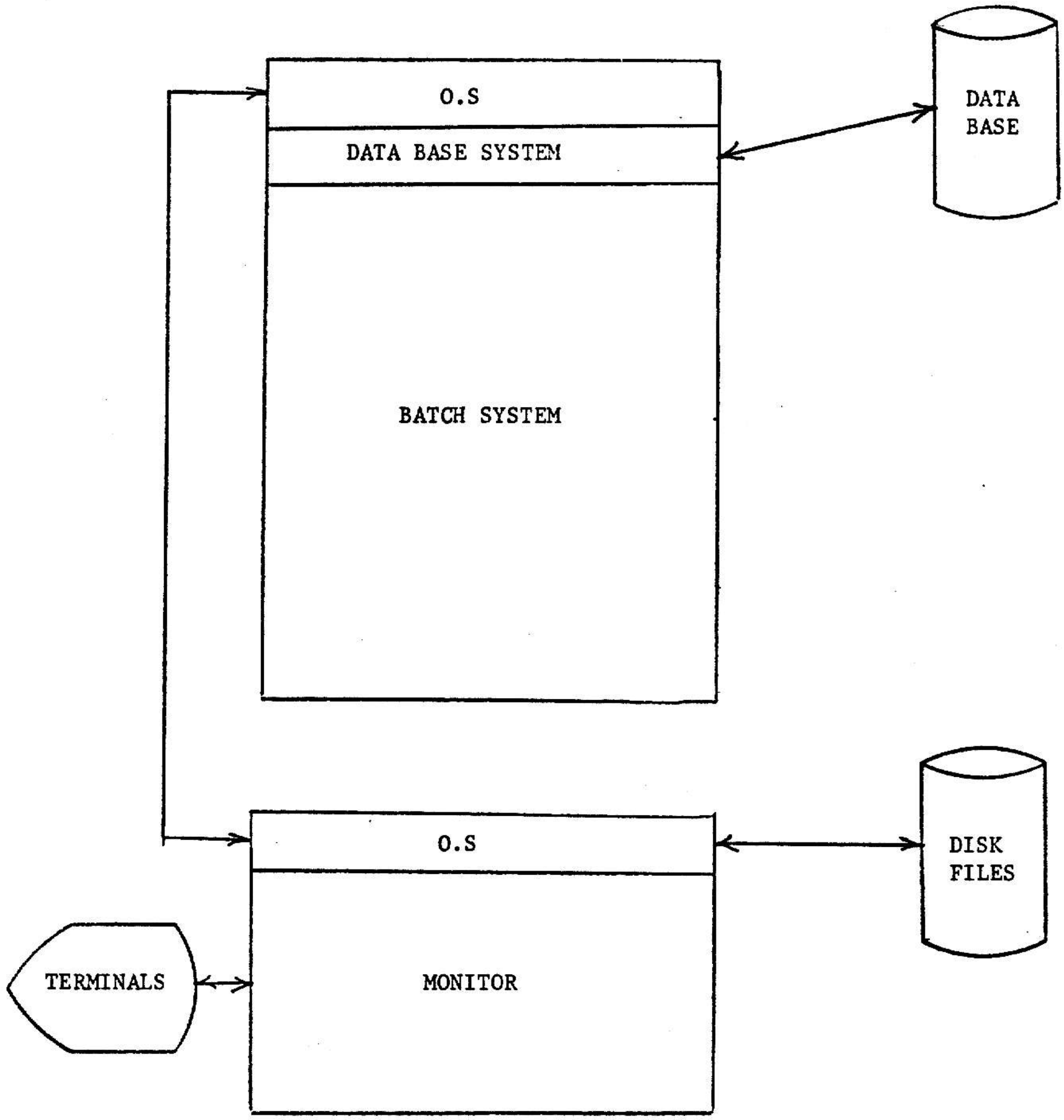


Figure 2.2. Front-end Monitor (Model II)

2.6 BACK-END DBMS (III)

Figure 2.3 shows a back-end data base system. This type of system can take considerable load off the main CPU. No additional memory is gained in this model but more CPU time for processing DBMS and disk I/O requests should be available. DBMS requests from both the monitor and batch system must be sent to this back-end processor. The gain in throughput should depend on how much of the basic system CPU time is used by the DBMS.

2.7 FRONT-END MONITOR/DBMS (IV)

The model shown in Figure 2.4 puts both the monitor and DBMS on an additional CPU with only the batch system on the main CPU. This configuration will provide 500 KB of additional memory to batch but the gain in CPU availability is not immediately clear. There will be additional processing power available but whether it can be utilized will depend on how the CPU load is distributed among the monitor, DBMS and batch systems. The batch system must send all DBMS requests to the additional CPU for processing.

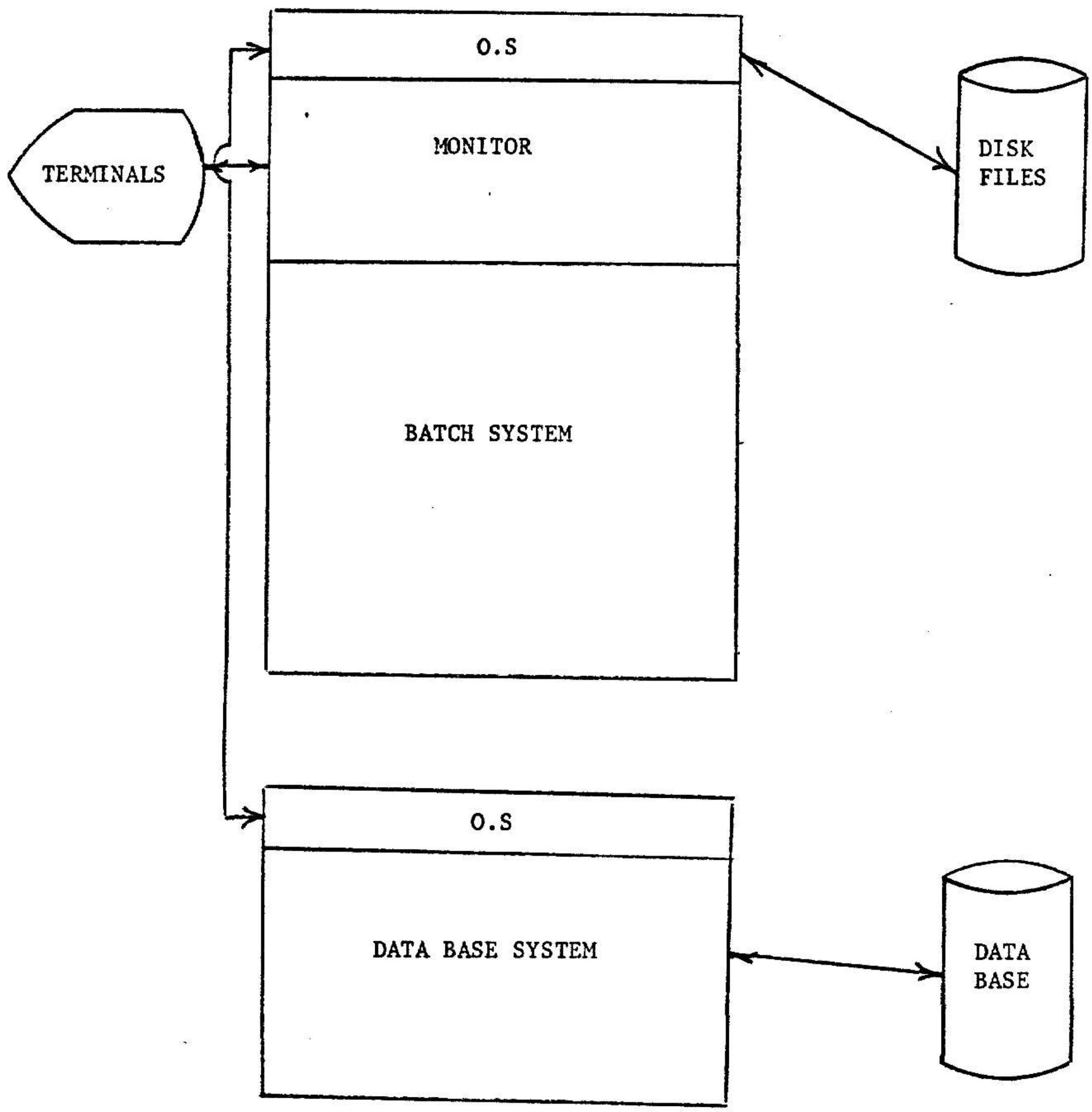


Figure 2.3. Back-end DBMS (Model III)

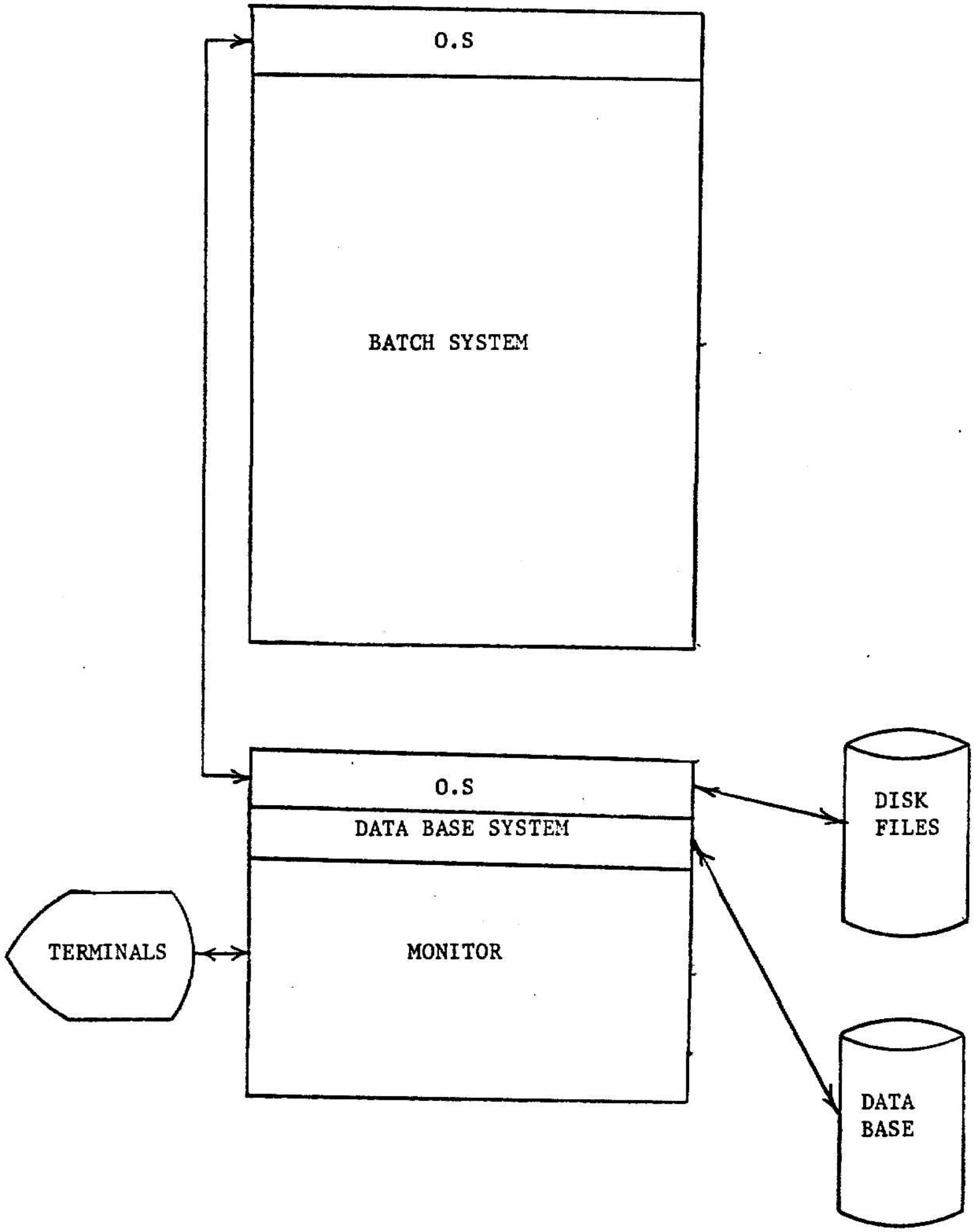


Figure 2.4. Front-end Monitor/DBMS (Model IV)

2.8 FRONT-END MONITOR, BACK-END DBMS (V)

Figure 2.5 shows a model which includes two additional CPU's. The second CPU is used as a front-end for the monitor and the third CPU is a back-end for the DBMS. There will be an additional 500 KB of memory available for batch and each major system will have its own CPU. As in configuration IV, the gain in throughput for the total system will depend on the demand for resources of each major system.

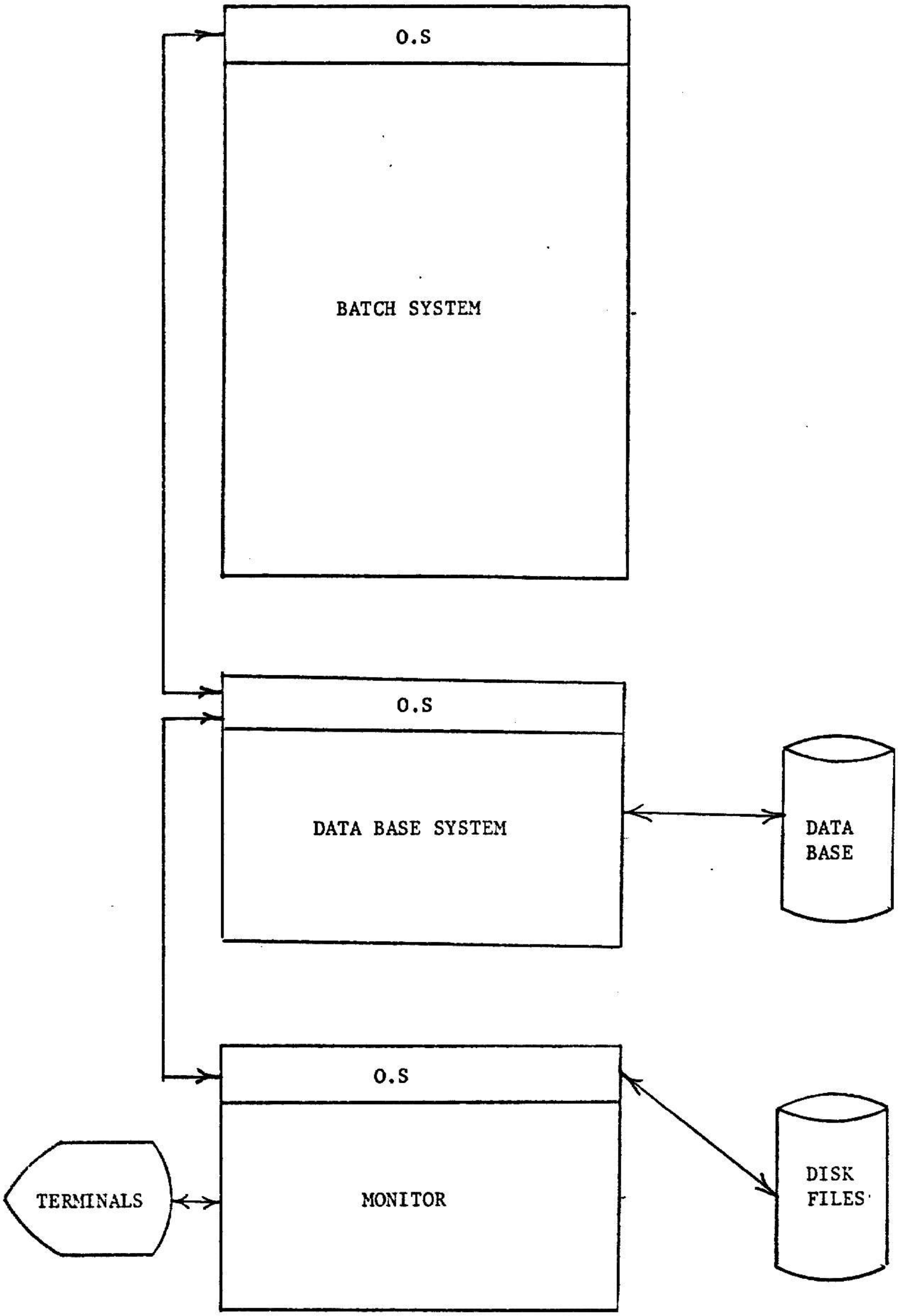


Figure 2.5. Front-end Monitor,
Back-end DBMS (Model V)

CHAPTER 3

IMPLEMENTATION

3.1 OVERVIEW

The implementation of the models will now be presented. The general assumptions and parameters used will be discussed and one model will be described in detail. Measurements of throughput in the models will be related to the real systems.

3.2 MODEL DESCRIPTION

Appendices II and IV contain flowcharts and listings of the five models. The systems were modeled in GPSS V and run on an IBM 370/158. The programs require approximately 2 minutes CPU time for 1 minute simulation time. This figure is highly dependent on program parameters. A simulation time of 1 minute was used for all runs. This time is long enough to get good random distributions and a reasonable measurement of throughput.

Each model has three sections; the monitor, the batch system and the data base system. A separate CPU scheduler is used for each CPU in the particular system. The unit of simulation time is 1 msec. This was chosen as a compromise so that both the fast CPU and the slow terminal could be studied in the same model. The primary considerations are

CPU, memory, and data base utilization and batch and monitor throughput. A 1 msec time unit is sufficient to get a good measurement of each of these items.

All five programs are similar. The change from one configuration to another is an addtional CPU and/or additional memory and a delay in sending data base requests to/from a back-end or front-end processor. Therefore only Model II, the front-end monitor, is chosen for detailed explanation below.

3.3 COMMUNICATIONS MONITOR

The monitor could be modeled in detail but a paging system is very complicated and difficult to model and probably should be evaluated in a separate study. We have used parameters obtained from the NETISA study to make a simplified model of this monitor. However, it should be accurate enough to give good results. The monitor can be regarded as a batch system which executes one job (message) at a time. The jobs have a very small CPU requirement (compared to real batch jobs) and a high I/O requirement because of paging and data base requests. All communications overhead is assumed to be handled by a programmable controller and not reflected in this model.

As shown in Appendix II, terminal messages (transactions) are generated at the rate of one message per terminal every 40 seconds. There are 100 simulated terminals. The messages are assigned the following parameters and queued at the input to the monitor:

1) Number of context blocks needed to swap in. These blocks hold the non-re-entrant data for each terminal. An average of 10 blocks is used.

2) Total CPU processing time needed for this message. An average of 10 msec or 400 msec is used in separate runs.

3) Total page swaps needed during this message processing. An average of 1 swap is used but this is highly dependent on whether the terminal users are accessing different parts of the application program. A Cincom study (1) shows that in data base update/retrieval with 10% of the application program in memory, the correct page will be resident 90% of the time. A NETISA study (9) seems to confirm this low swapping rate.

4) Total data base I/O's needed to process this message. An average of 5 is used.

5) An average output message length of 200 characters is used with a line speed of 4800 Baud.

All the above parameters were obtained from the NETISA study.

When a message gets control of the monitor it must swap context blocks and do page swaps. A separate swapping disk is assumed for these operations. The message must then hold the CPU for some compute time and perform the assigned number of data base I/O 's. In configuration II, the monitor must send the DBMS request to the main processor and receive a reply back. This requires 2 msec transmission time. The message transaction has priority over batch when requesting a facility such as the CPU or DBMS. This is done to insure fast terminal response time.

When the message has been processed, a simulated reply is sent to the terminal, the message transit time is tabulated and the transaction is terminated. The number of messages processed in a given time serves as a measure of monitor throughput.

The monitor can be viewed as a program which requires the following system resources:

1) 500 KB of main memory;

2) Some small amount of CPU time (<=500 msec) per message;

3) A large amount of disk I/O (data base and swapping).

3.4 BATCH SYSTEM

The batch model is not written to simulate a real batch job stream. It is intended to maximize the usage of system resources and provide a measure of the potential throughput of the various configurations. Parameters are adjusted on the various runs to correspond to different job mixes (i.e. compute-bound or I/O-bound jobs). If parameters from a real system were available, they could also be used.

The model generates 20 batch jobs (transactions) which are placed in a job queue awaiting memory allocation. The average job size is such that approximately half of the jobs are on the queue and half in memory at any time. This results in high utilization of available memory and CPU resources. Each job is assigned the following parameters:

1) Total CPU time needed. An average of 250 msec per job is used. This figure is very small compared to a true batch environment but a large number of short jobs can utilize the system resources to the same degree as longer jobs. Since many short jobs will complete in the 1 minute simulation time, the number of jobs completed will give a good measure of batch throughput.

2) Total memory needed. An average partition size of 300 KB is used. This number was experimentally determined by varying the job size (i.e. degree of multiprogramming) until the maximum throughput was obtained. This partition size was found to be optimum for both I/O-bound and compute-bound jobs.

A real batch job stream would be more unpredictable than this and again real parameters could be used if available. However, since we are trying to maximize the use of the system resources, this method will provide very high memory utilization. Good comparisons can then be made between configurations with different amounts of memory.

3) Time between DBMS requests. This parameter was set

to give 50% or 85% I/O wait time for different runs. These figures are typical of actual batch installations (10) with 50% being essentially a compute-bound job and 85% an I/O-bound job. I/O wait time is defined as follows:

$$W(\%) = \frac{\text{total I/O wait time}}{\text{total I/O plus total CPU time}} \times 100$$

4) CPU time slice. CPU time was allocated to the jobs in units of 100 msec. This figure is typical of many operating systems and it was found to give maximum throughput for the models.

After memory is allocated to a batch job, the job must compete with the other jobs for access to the CPU and DBMS. When a job has finished, it re-joins the job queue for re-execution. The number of jobs processed in a given time serves as a measure of batch throughput.

3.5 DBMS ROUTINE

A common DBMS routine is used for all jobs in the system. The routine is re-entrant and provides simulated access to 12 disk units. These units have an average seek time of 30 msec and transfer time of 2 msec. Two channels which can be multiplexed to the disk units are also simulated. The disk units are chosen at random while the channels are chosen according to the smallest channel queue.

Each DBMS request requires 1 disk access and 10 msec of compute time. These figures are chosen to provide a certain load on the system so the advantages of the different configurations can be seen clearly. Actual values for these parameters would depend on the data, file structures, applications and buffering techniques. However, since we are modeling a somewhat hypothetical system, we can feel free to arbitrarily set parameters as long as they are realistic.

3.6 CPU SCHEDULER

A separate CPU scheduler is used for each simulated CPU. This scheduler gives jobs the CPU on a first-come-first-served basis. The monitor has priority over batch jobs for this resource. An average of 1 msec is used for overhead each time a new job is initiated.

CHAPTER 4

RESULTS AND ANALYSIS

4.1 OVERVIEW

This chapter presents the method used for establishing model parameters and the results from the various simulation runs. An analysis of the data is made to explain why one configuration is most desireable under certain conditions.

4.2 TEST METHOD

Model I is used as the reference model. Assuming that the resources of this system are fully utilized, which System, II through V, would give the most additional throughput? The following steps were followed in making the simulation runs:

1) Assume some fixed CPU overhead for all CPU's. The remainder of the CPU time is available for useful work.

2) Maximize throughput on Model I by varying parameters for job size and time slice. Use the same values for the other models (i.e. assume same job size mix).

3) Make simulation runs using the following parameters:
(1) 50% I/O wait, 10 msec CPU time per message.

(2) 50% I/O wait, 400 msec CPU time per message.

(3) 85% I/O wait, 10 msec CPU time per message.

(4) 85% I/O wait, 400 msec CPU time per message.

4) Measure relative performance by the number of messages and jobs completed per unit time (1 minute).

4.3 SIMULATION RESULTS

A tabulation of the simulation results is given in Appendix III. A summary of each run is given below. The throughput is normalized using Model I as a reference.

(1)	MESSAGE THROUGHPUT	JOB THROUGHPUT
I	1.00	1.00
II	1.23	1.02
III	1.04	1.50
IV	1.83	1.31
V	1.49	1.51

Simulation run (1) used compute-bound batch jobs requiring 68.9% of available CPU time and with the data base system requiring 26.2%. The monitor CPU usage was very low at 1%. System II showed little increase in batch throughput since the monitor CPU demand was small. Even the additional 500 KB of memory made no difference since any additional jobs in memory were waiting for the CPU. Monitor throughput increased slightly because of the additional CPU dedicated to message processing.

System III provided 50% gain in batch throughput which shows the effect of removing the DBMS demands from the main processor. Systems IV and V both had significant gains in throughput for batch and monitor. In System IV the front-end CPU utilization was only 45% so the monitor throughput was very high. System V showed the effect of the delay in sending the monitor DBMS requests to another machine. The monitor throughput decreased significantly.

(2)	MESSAGE THROUGHPUT	JOB THROUGHPUT
I	1.00	1.00
II	1.09	1.27
III	.997	1.26
IV	1.30	1.38
V	1.27	1.77

Simulation run (2) used compute-bound jobs requiring 51% of available CPU time and the data base system 20.1%. The monitor usage was increased to 25.8%. System II again showed only a small increase in monitor throughput. The reason for this is that the monitor has highest priority for resources so an additional CPU is not too advantageous. The batch system did respond with some gain however. Systems III and IV showed the importance of putting the batch and DBMS on separate processors. Monitor throughput on System III was affected by the delay of sending DBMS requests to the back-end processor. Increased throughput for batch was obtained on all models especially Model V due to constant availability of the CPU.

(3)	MESSAGE THROUGHPUT	JOB THROUGHPUT
I	1.00	1.00
II	.940	.973
III	.940	1.10
IV	.819	1.29
V	1.03	1.43

Simulation run (3) used I/O-bound batch jobs requiring only 26.4% of available CPU time and the data base system requiring 62.3%. The monitor CPU usage was again low at about 1%. None of the systems showed a significant gain in monitor throughput. The reason for this is that the monitor was I/O-bound by the data base system. In fact, in Models I through IV both the monitor and batch are both I/O-bound by the DBMS. Data base utilization of the CPU indicates that 60 to 90% of available time is used for processing DBMS requests. Only System V with 3 processors gives some relief under these conditions and still the DBMS processor is utilized 97.5% of the time. More processing power for the DBMS is clearly needed. Possibly a multiple processor back-end DBMS could be used. (11)

(4)	MESSAGE THROUGHPUT	JOB THROUGHPUT
I	1.00	1.00
II	1.20	1.32
III	1.07	1.16
IV	.944	1.28
V	1.00	1.84

Simulation run (4) used I/O-bound batch jobs requiring 18.4% of available CPU time and the data base system requiring 42.4%. The monitor was set to 33.3% CPU usage. The results are similar to run (3). Both the monitor and batch systems were I/O-bound by the DBMS. The DBMS again required between 60 to 90% of CPU time.

System II showed some increase in batch throughput probably due to extra memory and thus more jobs in main memory. System V also had a large batch throughput due to increased memory and increased processor power. None of the systems had significant gains in monitor throughput since the monitor was usually waiting on the DBMS.

4.4 COST EFFECTIVENESS

The models presented will vary in cost depending on several factors. It was assumed that when the monitor was moved to another processor that the 500 KB additional memory would be available to the host. Thus the front-end would need at least 500 KB to run the monitor. It was also assumed that the memory used by the DBMS (TOTAL) was negligible.

This may not be the case with other data base systems which use 200 KB or more.

The cost of these various configurations must be considered relative to the cost of up-grading the original system. It must be noted that if one of these proposed systems were implemented, the main system could still be up-graded at a later date.

System III would be the least expensive of the four systems presented. Systems II and IV would be more expensive because of the additional memory and System V with two CPU's, two interconnection channels and additional memory would be most expensive.

CHAPTER 5

CONCLUSION

5.1 SUMMARY

The simulation results show that the addition of a front-end or back-end processor may increase throughput of the communications monitor, the batch system or both systems. A particular environment must be considered in order to determine the best configuration. Given below is a summary of the results for the four environments used in this study:

1) Compute-bound batch, I/O-bound monitor---More CPU time is needed for batch and removing the DBMS to a back-end is the best solution. If increased monitor throughput is needed, the monitor should reside on the same machine as the back-end DBMS. Configurations III,IV or V could be used. System III is most cost-effective.

2) Compute-bound batch, compute-bound monitor--- More batch throughput could be attained with configurations II through IV. Increased monitor throughput is attained with IV or V. Systems II or IV are most cost-effective.

3) I/O-bound batch, I/O-bound monitor--- Configuration

IV or V would give some additional throughput for batch. None of the proposed configurations helps the monitor. System IV is most cost-effective.

4) I/O-bound batch, compute-bound monitor--- Configuration II would help both batch and monitor with IV and V increasing batch throughput. Systems II or IV are most cost-effective.

5.2 Extentions

In this study a large and complex computer system has been simulated. Because of time and efficiency restraints, some assumptions and simplifications in the models were made. Although the results seem reasonable, one may wonder if these results have meaning in the real world. How could the accuracy of the results be evaluated and possibly improved? There are four areas where more study could improve accuracy:

1) The cost-effectiveness of all conversions should be studied in detail. All planning, hardware, software and maintenance costs should be considered for each configuration. Estimates of time needed to implement and useful lifetime of each system should also be included.

2) A more careful study should be directed toward bottlenecks in the system. For example, the monitor

paging disk utilization was 60-70% in the models. The use of a faster disk or drum might give the same results as an added processor with less cost.

3) An actual system should be studied in detail. Measurements should be made in both hardware and software to determine resource utilization, queue lengths and other parameters which could be used in the models. Subsystems such as ENVIRON/1 should be studied separately and input to the model as a set of demands on system resources.

4) One or more of the configurations should be implemented and additional measurements taken. The measured results should be compared with the simulation results to test accuracy of the models. Only by actual experimentation such as this can the validity of the assumptions be tested.

REFERENCES

1. DATAPRO INC. Reports on Data Communications, ENVIRON/1. Report No. C15-132-101.

2. CINCOM SYSTEMS INC. TOTAL Reference Manual. Cincinnati, Ohio. 1976.

3. Scherr, A.L. An analysis of time shared computer systems. MAC-TR-18, MIT Project MAC, Cambridge, Mass. 1965.

4. Smith, J.L. An analysis of time sharing computer systems using Markov models. Proc. AFIPS 1966 Spring Joint Computer Conf. Vol. 28 pp 87-95.

5. O'Connor, T.J. Analysis of a computer time sharing system-a simulation study. Unpubl. doctoral dissertation, Stanford Univ. Stanford, Calif. 1965.

6. Nielsen, N.R. The analysis of a general purpose computer time sharing systems. Doc. 40-10-1, Stanford Computation Center, Stanford, Calif. 1966.

7. Canaday, R.H., et. al. 'A Backend Computer for Data Base Management.', Communications of the ACM, Vol. 17,10, 1974, pp 575-582.

8. Maryanski, F.J., Fisher, P.S. and Wallentine, V.E. Evaluation of Conversion to a Backend Data Base Management

System, Proc. ACM Annual Conf. Oct. 1976, pp 292-297.

9. Pilant, W. Private communication, Naval Education and Training Information Systems Activity, NAS, Pensacola, Fla. 1976.

10. Madnick, S.E. and Donovan, J.J. Operating Systems, McGraw Hill Co. 1974.

11. Maryanski, F.J. Performance of Multi-Processor Backend Data Base Systems, Proc. of Conf. on Information Sciences and Systems, John Hopkins Univ. Baltimore, Md. Apr. 1977, pp 437-441.

APPENDIX I

ENVIRON/1 DESCRIPTION

A1.1 GENERAL

ENVIRON/1 is a communications monitor marketed by CINCOM INC. It is written in IBM 360 ASSEMBLER and requires a minimum of 200 KB of storage. The program uses BTAM to access remote terminal or printer devices. The programmer writes device independent application programs which permit terminal users to access on-line files. An interface to the TOTAL data base system (also a CINCOM product) is provided.

A1.2 PROGRAM DESCRIPTION

Figure A1.1 shows a block diagram of the ENVIRON/1 system. The control portion of the program contains device handlers, a task scheduler, interface to TOTAL, program loader and checkpoint recording software.

The application programs are coded in a language called TEBOL which is a dialect of COBOL with additional capabilities for telecommunications. A special compiler and link loader are available for compiling and merging TEBOL modules into one large re-entrant application program. The program is divided into 512 byte pages and stored in a program data set as

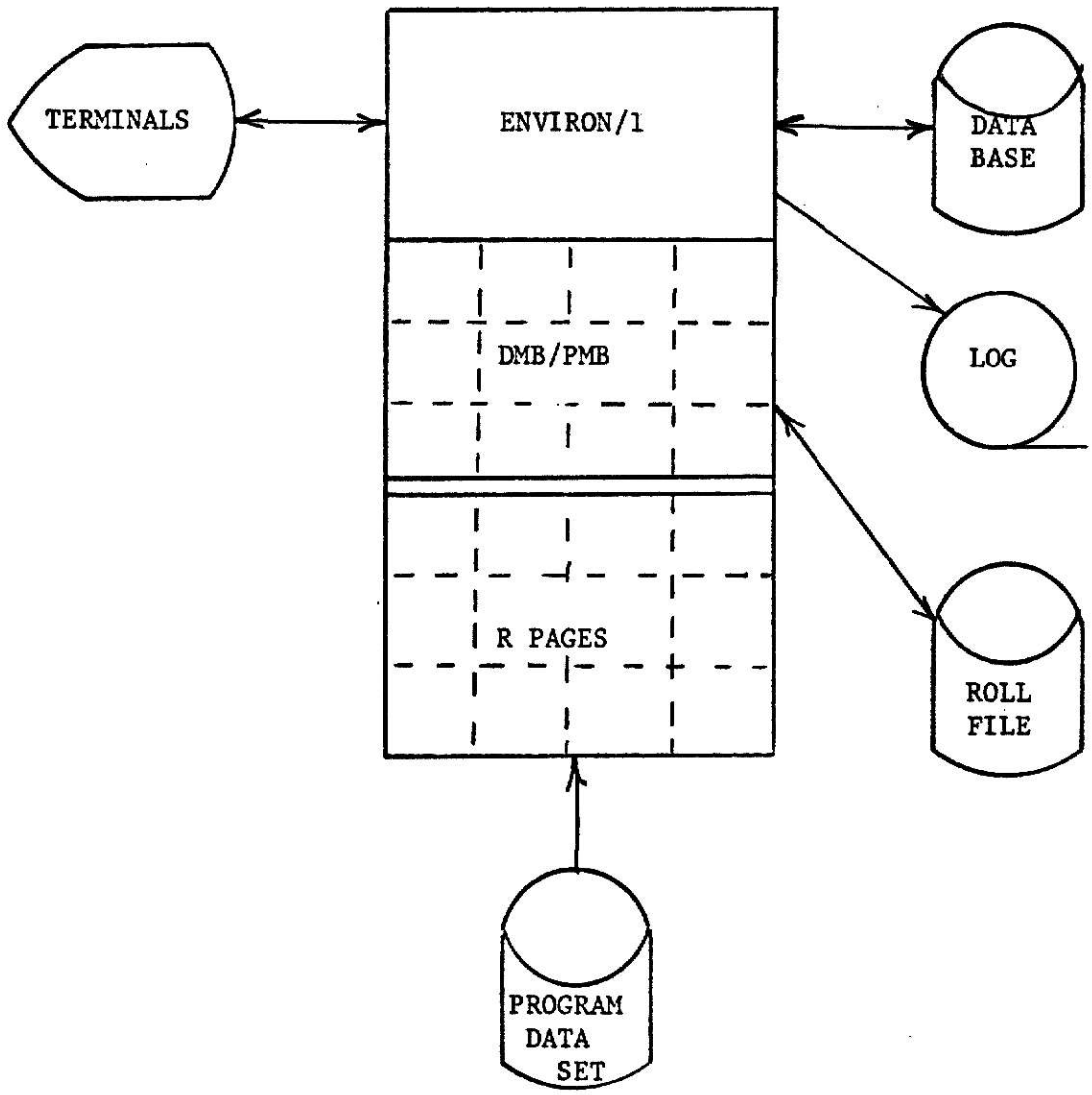


Figure A1.1. Communications Monitor

shown in Figure A1.1 The program data set is much too large to fit in core and is paged by the control program into the R-PAGE area. All on-line terminals use the same re-entrant program and thus the programmer has a single terminal concept.

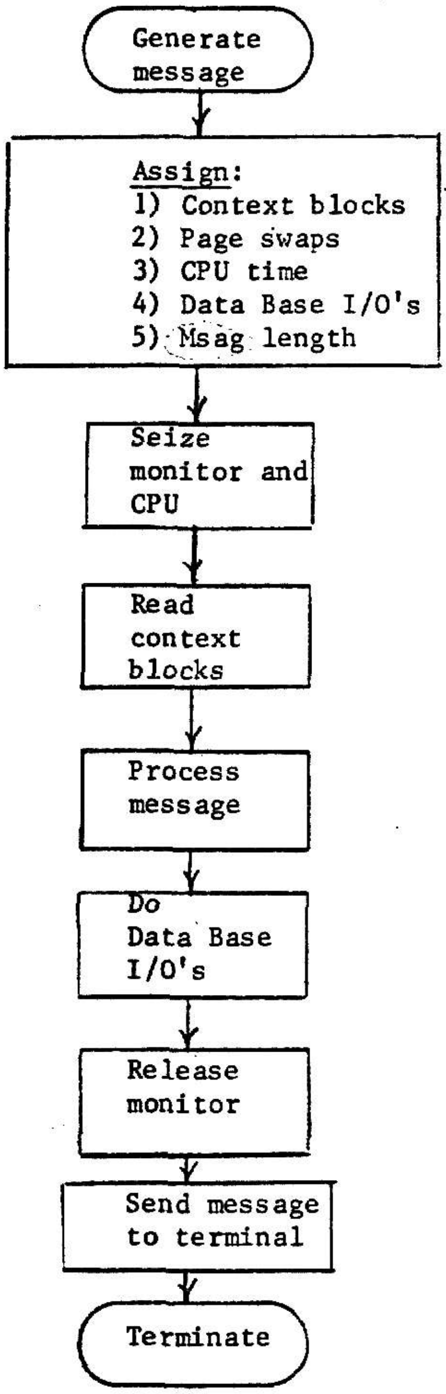
The non re-entrant data for each terminal is stored in 512 byte blocks in a file called the ROLL FILE. These blocks are also demand paged into an area called the DMB/PMB area. The number of these blocks assigned to a particular terminal is varied dynamically depending on the application being accessed.

A1.3 PROGRAM FEATURES

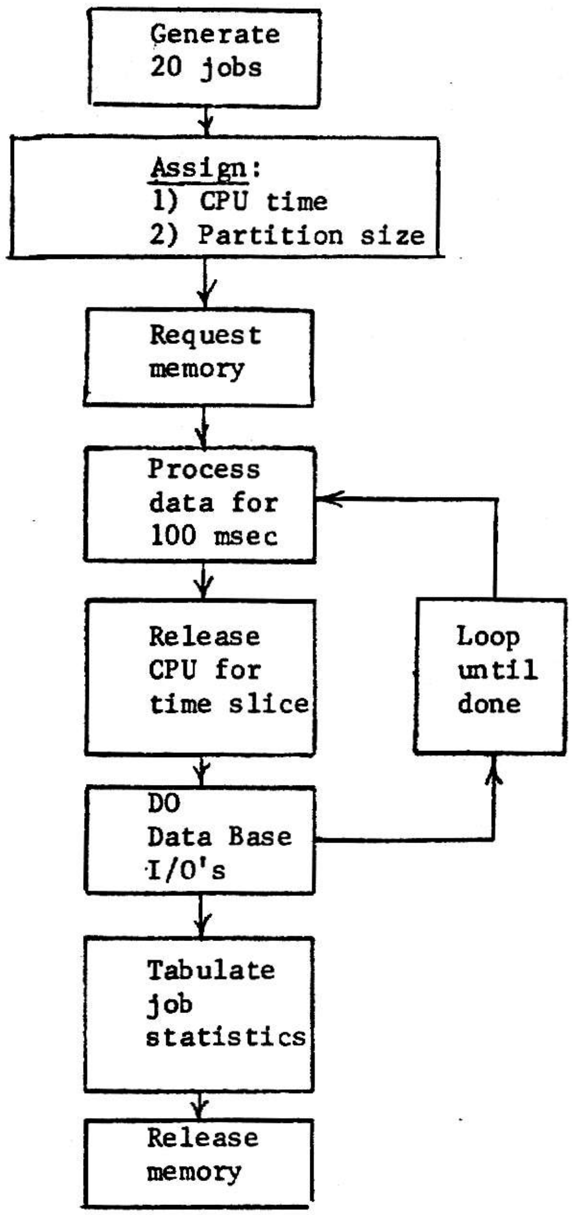
The monitor handles all communications functions. All transactions are logged on tape and periodic checkpoints are taken to prevent loss of data. Utilities are provided for rollback and recovery and maintenance of the program data set. A screen mapping subsystem is available to facilitate formatting of CRT displays. A supervisor mode exists for checking line activity and individual terminal status, taking terminals on or off line and broadcasting messages to selected terminals.

The application programs may be written in TEBOL or ASSEMBLER. An interface method to core resident FORTRAN or PL/1 programs is also provided. The programs have access to three types of disk files in addition to the data base files.

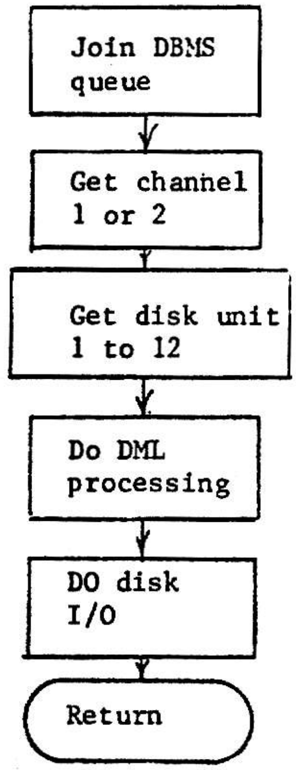
APPENDIX II



COMMUNICATIONS MONITOR



BATCH SYSTEM



DATA BASE MANAGEMENT SYSTEM

APPENDIX III

SIMULATION RESULTS

ABBREVIATION	MEANING
CPUM	CPU utilization by monitor.
CPUB	CPU utilization by batch system.
CPUD	CPU utilization by DBMS.
CPU1 CPU2 CPU3	Total CPU utilization.
OVH1 OVH2 OVH3	Scheduler task switch time.
CPQ1 CPQ2 CPQ3	Average CPU task queue contents.
MONQ	Average monitor queue contents.
JOBQ	Average batch job queue contents.
MEMU	Average memory utilization for batch jobs.
MSAG	Total number of messages processed.
MTIM	Average transit time for messages.
TRAN	Average time message held monitor.
JOBS	Number of batch jobs processed.
JTIM	Average processing time for batch jobs.

	CPUM	CPUB	CPUD	CPU1	CPU2	CPU3	OVH1	OVH2	OVH3	CPQ1	CPQ2	CPQ3	MONQ	JOBQ
I	.007	.689	.262	.999			.040			4.55			45.8	14.1
II	.008	.695	.271	.999	.017		.031	.008		4.13	0.00		41.7	12.0
III	.007	.942	.329	.979	.350		.028	.021		3.70	0.00		51.6	14.0
IV	.014	.958	.401	.980	.449		.021	.034		3.96	.067		21.2	13.2
V	.010	.968	.390	.991	.021	.412	.023	.011	.022	4.92	0.00	0.00	47.3	12.3

SIMULATION RUN (1)

	MEMU	MSAG	MTIM	TRAN	JOBS	JTIM
I	.887	57.0	17.9	1.01	175	32.2
II	.910	70.0	17.2	.844	179	28.2
III	.890	59.0	19.8	.999	263	29.9
IV	.930	104	9.45	.570	230	30.4
V	.916	85.0	18.4	.696	265	30.7

SIMULATION RUN (1) cont.

	CPUM	CPUB	CPUD	CPU1	CPU2	CPU3	OVH1	OVH2	OVH3	CPQ1	CPQ2	CPQ3	MONQ	JOBQ
I	.258	.510	.201	.999			.029			5.60			56.3	13.3
II	.313	.706	.260	.999	.318		.032	.005		4.69	0.00		47.1	11.9
III	.264	.699	.258	.985	.274		.020	.015		4.60	0.00		55.9	13.6
IV	.336	.726	.282	.743	.642		.016	.024		2.70	.356		52.6	13.1
V	.322	.974	.344	.996	.330	.366	.022	.007	.022	5.70	0.00	0.00	57.0	11.6

SIMULATION RUN (2)

	MEMU	MSAG	MTIM	TRAN	JOBS	JTIM
I	.908	44.0	21.9	1.33	139	31.9
II	.903	48.0	19.3	1.22	177	28.9
III	.900	43.0	23.7	1.36	175	30.4
IV	.881	57.0	20.1	1.01	192	31.2
V	.931	56.0	17.0	1.01	246	30.6

SIMULATION RUN (2) cont.

	CPUM	CPUB	CPUD	CPU1	CPU2	CPU3	OVH1	OVH2	OVH3	CPQ1	CPQ2	CPQ3	MONQ	JOBQ
I	.010	.264	.623	.981			.082			3.80			31.4	13.4
II	.009	.271	.651	.998	.019		.074	.009		4.10	0.00		43.9	11.6
III	.009	.353	.813	.420	.862		.057	.048		.240	0.00		34.5	13.4
IV	.006	.387	.889	.437	.961		.050	.065		.260	.150		35.0	11.3
V	.012	.421	.919	.472	.022	.975	.051	.010	.056	.350	0.00	0.00	39.2	10.0

SIMULATION RUN (3)

	MEMU	MSAG	MTIM	TRAN	JOBS	JTIM
I	.925	83.0	11.8	.709	73.0	31.8
II	.940	78.0	17.9	.758	71.0	30.8
III	.912	78.0	14.3	.748	80.0	25.3
IV	.916	68.0	15.7	.855	94.0	29.9
V	.945	85.0	15.9	.696	104.	28.0

SIMULATION RUN (3) cont.

	CPUM	CPUB	CPUD	CPU1	CPU2	CPU3	OVH1	OVH2	OVH3	CPQ1	CPQ2	CPQ3	MONQ	JOBQ
I	.333	.184	.424	.999			.057			7.10			49.4	10.8
II	.320	.278	.648	.999	.327		.073	.006		5.30	0.00		55.6	9.56
III	.298	.246	.555	.586	.590		.040	.035		1.78	0.00		41.4	14.2
IV	.298	.287	.632	.323	.976		.036	.045		.209	.397		41.5	10.8
V	.294	.410	.875	.463	.301	.929	.053	.007	.053	.280	0.00	0.00	44.0	12.2

SIMULATION RUN (4)

	MEMU	MSAG	MTIM	TRAN	JOBS	JTIM
I	.934	54.0	18.3	1.06[illegible]	50.0	35.6
II	.932	65.0	18.9	.908	66.0	31.1
III	.932	58.0	17.2	1.01	58.0	29.0
IV	.912	51.0	16.7	1.15	64.0	27.7
V	.917	54.0	15.4	1.09	92.0	30.9

SIMULATION RUN (4) cont.

APPENDIX V

PROGRAM LISTINGS

ILLEGIBLE DOCUMENT

THE FOLLOWING DOCUMENT(S) IS OF POOR LEGIBILITY IN THE ORIGINAL

THIS IS THE BEST COPY AVAILABLE

```
****************************************
*BATCH,MONITOR,+DBMS ON CPU1           *
****************************************
*
*
***************************************
*VARIABLES FOR COMM MONITOR           *
***************************************
*                                     *
 TERMS VARIABLE    100                 NUMBER OF TERMINALS
 AVGRE VARIABLE    40                  AVG OPERATOR RESPONSE TIME(SECONDS)
 MESTI FVARIABLE   (V$AVGRE/V$TERMS)*1000
*                                      TIME BETWEEN MESSAGES
 AVGTX VARIABLE    10                  AVG NUMBER OF CONTEXT BLOCKS PER TERMINAL
 CTXBF VARIABLE    300                 NUMBER OF CONTEXT BLOCKS  IN CORE
 CPCT  FVARIABLE   (V$CTXBF/(V$AVGTX*V$TERMS))*100
*                                      PERCENT OF CONTEXT BLOCKS IN CORE
 CTEXT FVARIABLE   (V$AVGTX*FN$EXPO)*FN$CTXF
*                                      NUMBER OF CONTEXT BLOCKS TO BRING IN
 CPUTI VARIABLE    10                  AVG CPU TIME NEEDED
 SWAPS VARIABLE    1                   AVG PAGE SWAPS PER TRANSACTION
 DBIOS VARIABLE    5                   AVG DATA BASE I/O 'S PER TRANSACTION
 AVGML VARIABLE    200                 AVG OUTPUT MESSAGE LENGTH
 OMSGL VARIABLE    V$AVGML*FN$EXPO
*                                      OUTPUT MESSAGE LENGTH
 AVGSP VARIABLE    4800               AVG LINE SPEED(BITS/SEC)
 SPEED FVARIABLE   (1/(V$AVGSP/8))*1000
*                                      LINE SPEED
 OTIME FVARIABLE   P5*V$SPEED          TIME FOR OUTPUT MESSAGE
****************************************
*VARIABLES FOR BACKGROUND JOBS         *
****************************************
 BJOBS VARIABLE    20                  NUMBER OF BACKGROUND JOBS
 BCPUT VARIABLE    250                 AVG CPU TIME NEEDED
 BPART VARIABLE    300                 AVG PARTITION SIZE NEEDED
 BDBTI VARIABLE    5
 BTSLC VARIABLE    100                 TIME SLICE
***************************************
*OTHER VARIABLES                      *
***************************************
 DMSF  VARIABLE    10                  DBMS OVERHEAD FACTOR
 OVHD  VARIABLE    1                   TASK SWITCH TIME
 DSKXF VARIABLE    2                   DISK TRANSFER TIME
 DSKAC VARIABLE    30                  DISK ACCESS TIME
 DISKS VARIABLE    12                  TOTAL DATA BASE DISK DRIVES
 DISKN VARIABLE    (RN3@V$DISKS+1)+20
*                                      DISK DRIVE
*
****************************************
*STORAGES                              *
****************************************
*
 BATCH STORAGE     1536                MEMORY AVAILABLE TO BATCH
*                                     *
***************************************
*FUNCTIONS                            *
***************************************
*                                     *
 EXPO  FUNCTION    RN2,C24            EXPONENTIAL DISTRIBUTION
0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/
```

```
.6..915/.7,1.2/.75,1.38/.8,1.6/.84,1.83/.88,2.12/
.9,2.3/.92,2.52/.94,2.81/.95,2.99/.96,3.2/.97,3.5/
.98,3.9/.99,4.6/.995,5.3/.998,6.2/.999,7/.9997,8

 CTXF   FUNCTION    V$CPCT,C11          CONTEXT PAGING FACTOR
0,1/10,.9/20,.8/30,.7/40,.6/50,.5/
60,.4/70,.3/80,.2/90,.1/100,0
*                                  *
************************************
*MODEL FOR MONITOR                 *
************************************
*                                  *
       SIMULATE
       GENERATE    V$MESTI,FN$EXPD    GENERATE INCOMING MESSAGES
       PRIORITY    10                  MAKE HIGHER THAN BATCH
       ASSIGN      1,V$CTEXT          NUMBER OF CONTEXT BLOCKS NEEDED TO BRING
*                                      IN
       ASSIGN       2,V$CPUTI,1
*                                      TOTAL CPU TIME NEEDED
       ASSIGN      3,V$SWAPS,1         TOTAL PAGE SWAPS NEEDED
       ASSIGN      4,V$DBIOS,1         TOTAL DBIO'S NEEDED
       ASSIGN      5,V$OMSGL           AVG OUTPUT MESSAGE LENGTH(CHAR)
       QUEUE       MONO                JOIN MONITOR QUEUE
       SEIZE       MON                REQUEST MONITOR
       DEPART      MONO                LEAVE MONITOR QUEUE
       TRANSFER    SBR,SKED1,11        GET CPU
       SEIZE       CPUM                COUNT MONITOR CPU TIME
*
*READ IN CONTEXT BLOCKS IF NEEDED
*
       TEST G      P1,K0,CKSW          ANY BLOCKS?
 CTXLP RELEASE     CPU1                YES
       RELEASE     CPUM
       SEIZE       20
       ADVANCE     V$DSKAC             PAGE OUT OLD CONTEXT BLOCK
       ADVANCE     V$DSKXF
       ADVANCE     V$DSKAC             PAGE IN NEW CONTEXT BLOCK
       ADVANCE     V$DSKXF
       RELEASE     20
       TRANSFER    SBR,SKED1,11        GET CPU AGAIN
       SEIZE       CPUM                COUNT MONITOR CPU TIME
       LOOP        1,CTXLP             LOOP IF MORE BLOCKS
*                                  *
************************************
*MAIN LOOP FOR COMM MONITOR        *
************************************
*                                  *
       ADVANCE     P2                  PROCESS MESSAGE FOR SOME TIME
 CKSW  TEST G      P3,K0,CKDB          IF P3 GT 0 WE NEED PAGE SWAP
*
*DO PROGRAM PAGE I/O'S
*
 CKSW1 RELEASE     CPU1
       RELEASE     CPUM
       SEIZE       20
       ADVANCE     V$DSKAC             DO PAGE SWAP I/O
       ADVANCE     V$DSKXF
       RELEASE     20
       TRANSFER    SBR,SKED1,11        GET CPU AGAIN
       SEIZE       CPUM                COUNT MONITOR CPU TIME
```

```
         LOOP       3,CKSW1              LOOP IF MORE SWAPS
*
*DO DATA BASE I/O'S
*
 CKDB   TEST G     P4,K0,MEND            IF P4 GT 0 WE NEED DBIO
 CKDB1  RELEASE    CPU1
        RELEASE    CPUM
        TRANSFER   SBR,DBMS,11           DO DATA BASE I/O
        TRANSFER   SBR,SKED1,11          SCHEDULE CPU
        SEIZE      CPUM                  COUNT MONITOR CPU TIME
        LOOP       4,CKDB1               LOOP IF MORE DBIO'S
 MEND   RELEASE    CPU1                  DONE WITH CPU
        RELEASE    CPUM
        RELEASE    MON                  DONE WITH MONITOR
        ADVANCE    V$OTIME              TRANSMIT MESSAGE TO TERMINAL
        TABULATE   XTIME                TABULATE TRANSIT TIMES
        TERMINATE  0                     REMOVE MESSAGE
 XTIME  TABLE      M1,0,1000,100
*
*************************************
*MODEL FOR BATCH JOBS               *
*************************************
*
        GENERATE   ,,,V$BJOBS,,,F        GENERATE BACKGROUND JOBS
 BINIT  ASSIGN     1,V$BCPUT,1           TOTAL CPU TIME NEEDED
        ASSIGN     2,V$BPART,1           TOTAL MEMORY NEEDED
        QUEUE      JOBQ
        ENTER      BATCH,P2              REQUEST MEMORY
        DEPART     JOBQ
        TRANSFER   SBR,SKED1,11          SCHEDULE CPU
        SEIZE      CPUB                  COUNT BATCH CPU TIME
*
*************************************
*MAIN LOOP FOR BATCH JOBS           *
*************************************
*
 BSET   ASSIGN     3,V$BDBTI,1           CPU TIME BEFORE DATA BASE I/O
 BLOOP  TEST G     P3,V$BTSLC,BDBIO
        ADVANCE    V$BTSLC               PROCESS BATCH FOR 1 TIME SLICE
*
*RELEASE CPU FOR TIME SLICE
*
        RELEASE    CPU1
        RELEASE    CPUB
        TRANSFER   SBR,SKED1,11          SCHEDULE CPU
        SEIZE      CPUB                  COUNT BATCH CPU TIME
        ASSIGN     1-,V$BTSLC
        ASSIGN     3-,V$BTSLC            DECREMENT COUNTERS
        TEST LE    P1,K0,BDONE           IF P1 EQ 0 WE'RE DONE
        TRANSFER   ,BLOOP
*
*DO A DATA BASE I/O
*
 BDBIO  ADVANCE    P3                    PROCESS BATCH FOR SOME TIME
        ASSIGN     1-,P3
        RELEASE    CPU1
        RELEASE    CPUB
        TRANSFER   SBR,DBMS,11           DO DATA BASE I/O
        TRANSFER   SBR,SKED1,11          SCHEDULE CPU
        SEIZE      CPUB                  COUNT BATCH CPU TIME
```

```
CKEN   TEST LE     P1,K0,BSET          IF P1 EQ 0 WE ARE DONE
*
*JOB COMPLETE, START ANOTHER
*
 BDONE RELEASE     CPU1
       RELEASE     CPUB
       LEAVE       BATCH,P2            RELEASE MEMORY
       TABULATE    BTIME               TABULATE BATCH JOB TIME
       TRANSFER    ,BINIT
 BTIME TABLE       M1,0,10000,100
*
*****************
*DBMS ROUTINE   *
*****************
*
 DBMS  QUEUE       DMSQ
       SEIZE       DMS                 GET DBMS ROUTINE
       DEPART      DMSQ
       TEST LE     Q18,Q19,GET1        PICK CHANNEL WITH SMALLEST QUEUE
       ASSIGN      10,K18              CHANNEL 1
       TRANSFER    ,GET2
 GET1  ASSIGN      10,K19              CHANNEL 2
 GET2  ASSIGN      9,V$DISKN           ASSIGN DISK NUMBER
       ASSIGN      12,P11              SAVE RETURN ADDR.
       TRANSFER    SBR,SKED1,11        GET CPU
       SEIZE       CPUD                COUNT DMS OVERHEAD
       ADVANCE     V$DMSE,FN$EXPD      DO OVERHEAD
       RELEASE     CPUD
       RELEASE     CPU1
       RELEASE     DMS
       QUEUE       P9                  ENTER DISK QUEUE
       SEIZE       P9                  GET DISK
       DEPART      P9                  LEAVE DISK QUEUE
       QUEUE       P10                 ENTER CHANNEL QUEUE
       SEIZE       P10                 GET CHANNEL
       DEPART      P10                 LEAVE CHANNEL QUEUE
       RELEASE     P10                 RELEASE CHANNEL
       ADVANCE     V$DSKAC             DO DISK ACCESS
       QUEUE       P10                 ENTER CHANNEL QUEUE
       SEIZE       P10                 GET CHANNEL
       DEPART      P10                 LEAVE CHANNEL QUEUE
       ADVANCE     V$DSKXF             TRANSFER DATA
       RELEASE     P9                  RELEASE DISK
       RELEASE     P10                 RELEASE CHANNEL
       TRANSFER    P,12,1              RETURN TO CALLER
*
****************
*CPU SCHEDULERS*
****************
*
 SKED1 QUEUE       CPQ1                ENTER CPU QUEUE
       SEIZE       CPU1                GET CPU
       DEPART      CPQ1                LEAVE CPU QUEUE
       SEIZE       OVH1                COUNT CPU OVERHEAD
       ADVANCE     V$OVHD,FN$EXPD      DO OVERHEAD
       RELEASE     OVH1
       TRANSFER    P,11,1              GIVE CPU TO CALLER
*
**********
*TIMING
```

```
**********
*
       GENERATE   60000
       TERMINATE  1
*                                    *
**************************************
*                                    *
       START      1
       END
/*
```

```
*****************************************
*BATCH+DBMS ON CPU1,MONITOR ON CPU2     *
*****************************************
*
*
***************************************
*VARIABLES FOR COMM MONITOR           *
***************************************
*                                     *
 TERMS VARIABLE    100                 NUMBER OF TERMINALS
 AVGRE VARIABLE    40                  AVG OPERATOR RESPONSE TIME(SECONDS)
 MESTI FVARIABLE   (V$AVGRE/V$TERMS)*1000
*                                       TIME BETWEEN MESSAGES
 AVGTX VARIABLE    10                   AVG NUMBER OF CONTEXT BLOCKS PER TERMINAL
 CTXBF VARIABLE    300                  NUMBER OF CONTEXT BLOCKS  IN CORE
 CPCT  FVARIABLE   (V$CTXBF/(V$AVGTX*V$TERMS))*100
*                                       PERCENT OF CONTEXT BLOCKS IN CORE
 CTEXT FVARIABLE   (V$AVGTX*FN$EXPO)*FN$CTXF
*                                       NUMBER OF CONTEXT BLOCKS TO BRING IN
 CPUTI VARIABLE    10                   AVG CPU TIME NEEDED
 SWAPS VARIABLE    1                    AVG PAGE SWAPS PER TRANSACTION
 DBIOS VARIABLE    5                    AVG DATA BASE I/O 'S PER TRANSACTION
 AVGML VARIABLE    200                  AVG OUTPUT MESSAGE LENGTH
 OMSGL VARIABLE    V$AVGML*FN$EXPO
*                                       OUTPUT MESSAGE LENGTH
 AVGSP VARIABLE    4800                AVG LINE SPEED(BITS/SEC)
 SPEED FVARIABLE   (1/(V$AVGSP/8))*1000
*                                       LINE SPEED
 OTIME FVARIABLE   P5*V$SPEED           TIME FOR OUTPUT MESSAGE
****************************************
*VARIABLES FOR BACKGROUND JOBS         *
****************************************
 BJOBS VARIABLE    20                   NUMBER OF BACKGROUND JOBS
 BCPUT VARIABLE    250                  AVG CPU TIME NEEDED
 BPART VARIABLE    300                  AVG PARTITION SIZE NEEDED
 BDBTI VARIABLE    5                    AVG CPU TIME BETWEEN DATA BASE I/O'S
 BTSLC VARIABLE    100                  TIME SLICE
***************************************
*OTHER VARIABLES                      *
***************************************
 DMSF   VARIABLE   10                   DBMS OVERHEAD FACTOR
 OVHD   VARIABLE   1                    TASK SWITCH TIME
 DSKXF VARIABLE    2                    DISK TRANSFER TIME
 DSKAC VARIABLE    30                   DISK ACCESS TIME
 DISKS VARIABLE    12                   TOTAL DATA BASE DISK DRIVES
 DISKN VARIABLE    (RN3@V$DISKS+1)+20
*                                       DISK DRIVE
 COMM   VARIABLE   1                    COMM TIME BETWEEN CPU'S
*
****************************************
*STORAGES                              *
****************************************
*
 BATCH STORAGE     2048                 MEMORY AVAILABLE TO BATCH
*                                       *
***************************************
*FUNCTIONS                            *
***************************************
*                                     *
 EXPO  FUNCTION    RN2,C24             EXPONENTIAL DISTRIBUTION
```

```
0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/
.6,.915/.7,1.2/.75,1.38/.8,1.6/.84,1.83/.88,2.12/
.9,2.3/.92,2.52/.94,2.81/.95,2.99/.96,3.2/.97,3.5/
.98,3.9/.99,4.6/.995,5.3/.998,6.2/.999,7/.9997,8

 CTXF   FUNCTION   V$CPCT,C11           CONTEXT PAGING FACTOR
0,1/10,.9/20,.8/30,.7/40,.6/50,.5/
60,.4/70,.3/80,.2/90,.1/100,0
*                                    *
**************************************
*MODEL FOR MONITOR                   *
**************************************
*                                    *
        SIMULATE
        GENERATE    V$MESTI,FN$EXPD     GENERATE INCOMING MESSAGES
        PRIORITY    10                   MAKE HIGHER THAN BATCH
        ASSIGN      1,V$CTEXT           NUMBER OF CONTEXT BLOCKS NEEDED TO BRING
*                                        IN
        ASSIGN       2,V$CPUTI,1
*                                        TOTAL CPU TIME NEEDED
        ASSIGN      3,V$SWAPS,1          TOTAL PAGE SWAPS NEEDED
        ASSIGN      4,V$DBIOS,1          TOTAL DBIO'S NEEDED
        ASSIGN      5,V$OMSGL            AVG OUTPUT MESSAGE LENGTH(CHAR)
        QUEUE       MONO                 JOIN MONITOR QUEUE
        SEIZE       MON                 REQUEST MONITOR
        DEPART      MONO                 LEAVE MONITOR QUEUE
        TRANSFER    SBR,SKED2,11         GET CPU
        SEIZE       CPUM                 COUNT MONITOR CPU TIME
*
*READ IN CONTEXT BLOCKS IF NEEDED
*
        TEST G      P1,K0,CKSW           ANY BLOCKS?
 CTXLP  RELEASE     CPU2                 YES
        RELEASE     CPUM
        SEIZE       20
        ADVANCE     V$DSKAC              PAGE OUT OLD CONTEXT BLOCK
        ADVANCE     V$DSKXF
        ADVANCE     V$DSKAC              PAGE IN NEW CONTEXT BLOCK
        ADVANCE     V$DSKXF
        RELEASE     20
        TRANSFER    SBR,SKED2,11         GET CPU AGAIN
        SEIZE       CPUM                 COUNT MONITOR CPU TIME
        LOOP        1,CTXLP              LOOP IF MORE BLOCKS
*                                    *
**************************************
*MAIN LOOP FOR COMM MONITOR           *
**************************************
*                                    *
        ADVANCE     P2                   PROCESS MESSAGE FOR SOME TIME
 CKSW   TEST G      P3,K0,CKDB           IF P3 GT 0 WE NEED PAGE SWAP
*
*DO PROGRAM PAGE I/O'S
*
 CKSW1  RELEASE     CPU2
        RELEASE     CPUM
        SEIZE       20
        ADVANCE     V$DSKAC              DO PAGE SWAP I/O
        ADVANCE     V$DSKXF
        RELEASE     20
        TRANSFER    SBR,SKED2,11         GET CPU AGAIN
```

```
         SEIZE       CPUM                          COUNT MONITOR CPU TIME
         LOOP        3,CKSW1                       LOOP IF MORE SWAPS
*
*DO DATA BASE I/O'S
*
CKDB     TEST G      P4,K0,MEND                    IF P4 GT 0 WE NEED DBIO
CKDB1    RELEASE     CPU2
         RELEASE     CPUM
         ADVANCE     V$COMM                        SEND DML TO HOST
         TRANSFER    SBR,DBMS,11                   DO DATA BASE I/O
         ADVANCE     V$COMM                        GET DATA FROM HOST
         TRANSFER    SBR,SKED2,11                  SCHEDULE CPU
         SEIZE       CPUM                          COUNT MONITOR CPU TIME
         LOOP        4,CKDB1                       LOOP IF MORE DBIO'S
MEND     RELEASE     CPU2                          DONE WITH CPU
         RELEASE     CPUM
         RELEASE     MON                           DONE WITH MONITOR
         ADVANCE     V$OTIME                       TRANSMIT MESSAGE TO TERMINAL
         TABULATE    XTIME                         TABULATE TRANSIT TIMES
         TERMINATE   0                             REMOVE MESSAGE
XTIME    TABLE       M1,0,1000,100
*
*************************************
*MODEL FOR BATCH JOBS               *
*************************************
*
         GENERATE    ,,,V$BJOBS,,,F                GENERATE BACKGROUND JOBS
BINIT    ASSIGN      1,V$BCPUT,1                   TOTAL CPU TIME NEEDED
         ASSIGN      2,V$BPART,1                   TOTAL MEMORY NEEDED
         QUEUE       JOBQ
         ENTER       BATCH,P2                      REQUEST MEMORY
         DEPART      JOBQ
         TRANSFER    SBR,SKED1,11                  SCHEDULE CPU
         SEIZE       CPUB                          COUNT BATCH CPU TIME
*
*************************************
*MAIN LOOP FOR BATCH JOBS           *
*************************************
*
BSET     ASSIGN      3,V$BDBTI,1                   CPU TIME BEFORE DATA BASE I/O
BLOOP    TEST G      P3,V$BTSLC,BDBIO
         ADVANCE     V$BTSLC                       PROCESS BATCH FOR 1 TIME SLICE
*
*RELEASE CPU FOR TIME SLICE
*
         RELEASE     CPU1
         RELEASE     CPUB
         TRANSFER    SBR,SKED1,11                  SCHEDULE CPU
         SEIZE       CPUB                          COUNT BATCH CPU TIME
         ASSIGN      1-,V$BTSLC
         ASSIGN      3-,V$BTSLC                    DECREMENT COUNTERS
         TEST LE     P1,K0,BDONE                   IF P1 EQ 0 WE'RE DONE
         TRANSFER    ,BLOOP
*
*DO A DATA BASE I/O
*
BDBIO    ADVANCE     P3                            PROCESS BATCH FOR SOME TIME
         ASSIGN      1-,P3
         RELEASE     CPU1
         RELEASE     CPUB
```

```
       TRANSFER   SBR,DBMS,11          DO DATA BASE I/O
       TRANSFER   SBR,SKED1,11         SCHEDULE CPU
       SEIZE      CPUB                 COUNT BATCH CPU TIME
 CKEN  TEST LE    P1,K0,BSET           IF P1 EQ 0 WE ARE DONE
*
*JOB COMPLETE, START ANOTHER
*
 BDONE RELEASE    CPU1
       RELEASE    CPUB
       LEAVE      BATCH,P2             RELEASE MEMORY
       TABULATE   BTIME                TABULATE BATCH JOB TIME
       TRANSFER   ,BINIT
 BTIME TABLE      M1,0,10000,100
*
*****************
*DBMS ROUTINE   *
*****************
*
 DBMS  QUEUE      DMSQ
       SEIZE      DMS                  GET DBMS ROUTINE
       DEPART     DMSQ
       TEST LE    Q18,Q19,GET1         PICK CHANNEL WITH SMALLEST QUEUE
       ASSIGN     10,K18               CHANNEL 1
       TRANSFER   ,GET2
 GET1  ASSIGN     10,K19               CHANNEL 2
 GET2  ASSIGN     9,V$DISKN            ASSIGN DISK NUMBER
       ASSIGN     12,P11               SAVE RETURN ADDR.
       TRANSFER   SBR,SKED1,11         GET CPU
       SEIZE      CPUD                 COUNT DMS OVERHEAD
       ADVANCE    V$DMSE,FN$EXPO       DO OVERHEAD
       RELEASE    CPUD
       RELEASE    CPU1
       RELEASE    DMS
       QUEUE      P9                   ENTER DISK QUEUE
       SEIZE      P9                   GET DISK
       DEPART     P9                   LEAVE DISK QUEUE
       QUEUE      P10                  ENTER CHANNEL QUEUE
       SEIZE      P10                  GET CHANNEL
       DEPART     P10                  LEAVE CHANNEL QUEUE
       RELEASE    P10                  RELEASE CHANNEL
       ADVANCE    V$DSKAC              DO DISK ACCESS
       QUEUE      P10                  ENTER CHANNEL QUEUE
       SEIZE      P10                  GET CHANNEL
       DEPART     P10                  LEAVE CHANNEL QUEUE
       ADVANCE    V$DSKXF              TRANSFER DATA
       RELEASE    P9                   RELEASE DISK
       RELEASE    P10                  RELEASE CHANNEL
       TRANSFER   P,12,1               RETURN TO CALLER
*
****************
*CPU SCHEDULERS*
****************
*
 SKED1 QUEUE      CPQ1                 ENTER CPU QUEUE
       SEIZE      CPU1                 GET CPU
       DEPART     CPQ1                 LEAVE CPU QUEUE
       SEIZE      OVH1                 COUNT CPU OVERHEAD
       ADVANCE    V$OVHD,FN$EXPO       DO OVERHEAD
       RELEASE    OVH1
       TRANSFER   P,11,1               GIVE CPU TO CALLER
```

```
*
 SKED2 QUEUE       CPQ2                   ENTER CPU QUEUE
       SEIZE       CPU2                   GET CPU
       DEPART      CPQ2                   LEAVE CPU QUEUE
       SEIZE       OVH2                   COUNT CPU OVERHEAD
       ADVANCE     V$OVHD,FN$EXPO         DO OVERHEAD
       RELEASE     OVH2
       TRANSFER    P,11,1                 GIVE CPU TO CALLER
*
**********
*TIMING
**********
*
       GENERATE    60000
       TERMINATE   1
*                                    *
**************************************
*                                    *
       START       1
       END
/*
```

```
**************************************
*BATCH+MONITOR ON CPU1.DBMS ON CPU2  *
**************************************
*
*
**************************************
*VARIABLES FOR COMM MONITOR          *
**************************************
*                                    *
 TERMS VARIABLE    100                NUMBER OF TERMINALS
 AVGRE VARIABLE    40                 AVG OPERATOR RESPONSE TIME(SECONDS)
 MESTI FVARIABLE   (V$AVGRE/V$TERMS)*1000
*                                     TIME BETWEEN MESSAGES
 AVGTX VARIABLE    10                 AVG NUMBER OF CONTEXT BLOCKS PER TERMINAL
 CTXBF VARIABLE    300                NUMBER OF CONTEXT BLOCKS  IN CORE
 CPCT  FVARIABLE   (V$CTXBF/(V$AVGTX*V$TERMS))*100
*                                     PERCENT OF CONTEXT BLOCKS IN CORE
 CTEXT FVARIABLE   (V$AVGTX*FN$EXPO)*FN$CTXF
*                                     NUMBER OF CONTEXT BLOCKS TO BRING IN
 CPUTI VARIABLE    10                 AVG CPU TIME NEEDED
 SWAPS VARIABLE    1                  AVG PAGE SWAPS PER TRANSACTION
 DBIOS VARIABLE    5                  AVG DATA BASE I/O 'S PER TRANSACTION
 AVGML VARIABLE    200                AVG OUTPUT MESSAGE LENGTH
 OMSGL VARIABLE    V$AVGML*FN$EXPO
*                                     OUTPUT MESSAGE LENGTH
 AVGSP VARIABLE    4800              AVG LINE SPEED(BITS/SEC)
 SPEED FVARIABLE   (1/(V$AVGSP/8))*1000
*                                     LINE SPEED
 OTIME FVARIABLE   P5*V$SPEED         TIME FOR OUTPUT MESSAGE
**************************************
*VARIABLES FOR BACKGROUND JOBS       *
**************************************
 BJOBS VARIABLE    20                 NUMBER OF BACKGROUND JOBS
 BCPUT VARIABLE    250                AVG CPU TIME NEEDED
 BPART VARIABLE    300                AVG PARTITION SIZE NEEDED
 BDBTI VARIABLE    5                  AVG CPU TIME BETWEEN DATA BASE I/O'S
 BTSLC VARIABLE    100                TIME SLICE
**************************************
*OTHER VARIABLES                     *
**************************************
 DMSF  VARIABLE    10                 DBMS OVERHEAD FACTOR
 OVHD  VARIABLE    1                  TASK SWITCH TIME
 DSKXF VARIABLE    2                  DISK TRANSFER TIME
 DSKAC VARIABLE    30                 DISK ACCESS TIME
 DISKS VARIABLE    12                 TOTAL DATA BASE DISK DRIVES
 DISKN VARIABLE    (RN3@V$DISKS+1)+20
*                                     DISK DRIVE
 COMM  VARIABLE    1                  COMM TIME BETWEEN CPU'S
*
**************************************
*STORAGES                            *
**************************************
*
 BATCH STORAGE     1536               MEMORY AVAILABLE TO BATCH
*                                    *
**************************************
*FUNCTIONS                           *
**************************************
*                                    *
 EXPO  FUNCTION    RN2,C24           EXPONENTIAL DISTRIBUTION
```

```
0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/
.6,.915/.7,1.2/.75,1.38/.8,1.6/.84,1.83/.88,2.12/
.9,2.3/.92,2.52/.94,2.81/.95,2.99/.96,3.2/.97,3.5/
.98,3.9/.99,4.6/.995,5.3/.998,6.2/.999,7/.9997,8

 CTXF   FUNCTION    V$CPCI,C11              CONTEXT PAGING FACTOR
0.1/10,.9/20,.8/30,.7/40,.6/50,.5/
60,.4/70,.3/80,.2/90,.1/100,0
*                                       *
*****************************************
*MODEL FOR MONITOR                      *
*****************************************
*                                       *
        SIMULATE
        GENERATE    V$MESTI,FN$EXPO      GENERATE INCOMING MESSAGES
        PRIORITY    10                    MAKE HIGHER THAN BATCH
        ASSIGN      1,V$CTEXT            NUMBER OF CONTEXT BLOCKS NEEDED TO BRING
*                                          IN
        ASSIGN       2,V$CPUTI,1
*                                         TOTAL CPU TIME NEEDED
        ASSIGN      3,V$SWAPS,1           TOTAL PAGE SWAPS NEEDED
        ASSIGN      4,V$DBIOS,1           TOTAL DBIO'S NEEDED
        ASSIGN      5,V$OMSGL             AVG OUTPUT MESSAGE LENGTH(CHAR)
        QUEUE       MONO                  JOIN MONITOR QUEUE
        SEIZE       MON                  REQUEST MONITOR
        DEPART      MONO                  LEAVE MONITOR QUEUE
        TRANSFER    SBR,SKED1,11          GET CPU
        SEIZE       CPUM                  COUNT MONITOR CPU TIME
*
*READ IN CONTEXT BLOCKS IF NEEDED
*
        TEST G      P1,KO,CKSW            ANY BLOCKS?
 CTXLP  RELEASE     CPU1                  YES
        RELEASE     CPUM
        SEIZE       20
        ADVANCE     V$DSKAC               PAGE OUT OLD CONTEXT BLOCK
        ADVANCE     V$DSKXF
        ADVANCE     V$DSKAC               PAGE IN NEW CONTEXT BLOCK
        ADVANCE     V$DSKXF
        RELEASE     20
        TRANSFER    SBR,SKED1,11          GET CPU AGAIN
        SEIZE       CPUM                  COUNT MONITOR CPU TIME
        LOOP        1,CTXLP              LOOP IF MORE BLOCKS
*                                       *
*****************************************
*MAIN LOOP FOR COMM MONITOR             *
*****************************************
*                                       *
        ADVANCE     P2                    PROCESS MESSAGE FOR SOME TIME
 CKSW   TEST G      P3,KO,CKDB            IF P3 GT 0 WE NEED PAGE SWAP
*
*DO PROGRAM PAGE I/O'S
*
 CKSW1  RELEASE     CPU1
        RELEASE     CPUM
        SEIZE       20
        ADVANCE     V$DSKAC               DO PAGE SWAP I/O
        ADVANCE     V$DSKXF
        RELEASE     20
        TRANSFER    SBR,SKED1,11          GET CPU AGAIN
```

```
        SEIZE       CPUM                    COUNT MONITOR CPU TIME
        LOOP        3,CKSW1                 LOOP IF MORE SWAPS
*
*DO DATA BASE I/O'S
*
 CKDB   TEST G      P4,K0,MEND              IF P4 GT 0 WE NEED DBIO
 CKDB1  RELEASE     CPU1
        RELEASE     CPUM
        ADVANCE     V$COMM                  SEND DML TO HOST
        TRANSFER    SBR,DBMS,11             DO DATA BASE I/O
        ADVANCE     V$COMM                  GET DATA FROM HOST
        TRANSFER    SBR,SKED1,11            GET CPU
        SEIZE       CPUM                    COUNT MONITOR CPU TIME
        LOOP        4,CKDB1                 LOOP IF MORE DBIO'S
 MEND   RELEASE     CPU1                    DONE WITH CPU
        RELEASE     CPUM
        RELEASE     MON                    DONE WITH MONITOR
        ADVANCE     V$OTIME                TRANSMIT MESSAGE TO TERMINAL
        TABULATE    XTIME                  TABULATE TRANSIT TIMES
        TERMINATE   0                       REMOVE MESSAGE
 XTIME  TABLE       M1,0,1000,100
*
***************************************
*MODEL FOR BATCH JOBS                  *
***************************************
*
        GENERATE    ,,,V$BJOBS,,,F           GENERATE BACKGROUND JOBS
 BINIT  ASSIGN      1,V$BCPU1,1             TOTAL CPU TIME NEEDED
        ASSIGN      2,V$BPART,1             TOTAL MEMORY NEEDED
        QUEUE       JOBQ
        ENTER       BATCH,P2                REQUEST MEMORY
        DEPART      JOBQ
        TRANSFER    SBR,SKED1,11            SCHEDULE CPU
        SEIZE       CPUB                    COUNT BATCH CPU TIME
*
***************************************
*MAIN LOOP FOR BATCH JOBS              *
***************************************
*
 BSET   ASSIGN      3,V$BDBTI,1             CPU TIME BEFORE DATA BASE I/O
 BLOOP  TEST G      P3,V$BTSLC,BDBIO
        ADVANCE     V$BTSLC                 PROCESS BATCH FOR 1 TIME SLICE
*
*RELEASE CPU FOR TIME SLICE
*
        RELEASE     CPU1
        RELEASE     CPUB
        TRANSFER    SBR,SKED1,11            SCHEDULE CPU
        SEIZE       CPUB                    COUNT BATCH CPU TIME
        ASSIGN      1-,V$BTSLC
        ASSIGN      3-,V$BTSLC              DECREMENT COUNTERS
        TEST LE     P1,K0,BDONE             IF P1 EQ 0 WE'RE DONE
        TRANSFER    ,BLOOP
*
*DO A DATA BASE I/O
*
 BDBIO  ADVANCE     P3                      PROCESS BATCH FOR SOME TIME
        ASSIGN      1-,P3
        RELEASE     CPU1
        RELEASE     CPUB
```

```
        ADVANCE     V$COMM               SEND DML TO HOST
        TRANSFER    SBR,DBMS,11          DO DATA BASE I/O
        ADVANCE     V$COMM               GET DATA FROM HOST
        TRANSFER    SBR,SKED1,11         SCHEDULE CPU
        SEIZE       CPUB                 COUNT BATCH CPU TIME
 CKEN   TEST LE     P1,K0,BSET           IF P1 EQ 0 WE ARE DONE
*
*JOB COMPLETE, START ANOTHER
*
 BDONE  RELEASE     CPU1
        RELEASE     CPUB
        LEAVE       BATCH,P2             RELEASE MEMORY
        TABULATE    BTIME                TABULATE BATCH JOB TIME
        TRANSFER    ,BINIT
 BTIME  TABLE       M1,0,10000,100
*
*****************
*DBMS ROUTINE   *
*****************
*
 DBMS   QUEUE       DMSQ
        SEIZE       DMS                  GET DBMS ROUTINE
        DEPART      DMSQ
        TEST LE     Q18,Q19,GET1         PICK CHANNEL WITH SMALLEST QUEUE
        ASSIGN      10,K18               CHANNEL 1
        TRANSFER    ,GET2
 GET1   ASSIGN      10,K19               CHANNEL 2
 GET2   ASSIGN      9,V$DISKN            ASSIGN DISK NUMBER
        ASSIGN      12,P11               SAVE RETURN ADDR.
        TRANSFER    SBR,SKED2,11         GET CPU
        SEIZE       CPUD                 COUNT DMS OVERHEAD
        ADVANCE     V$DMSE,FN$EXPO       DO OVERHEAD
        RELEASE     CPUD
        RELEASE     CPU2
        RELEASE     DMS
        QUEUE       P9                   ENTER DISK QUEUE
        SEIZE       P9                   GET DISK
        DEPART      P9                   LEAVE DISK QUEUE
        QUEUE       P10                  ENTER CHANNEL QUEUE
        SEIZE       P10                  GET CHANNEL
        DEPART      P10                  LEAVE CHANNEL QUEUE
        RELEASE     P10                  RELEASE CHANNEL
        ADVANCE     V$DSKAC              DO DISK ACCESS
        QUEUE       P10                  ENTER CHANNEL QUEUE
        SEIZE       P10                  GET CHANNEL
        DEPART      P10                  LEAVE CHANNEL QUEUE
        ADVANCE     V$DSKXF              TRANSFER DATA
        RELEASE     P9                   RELEASE DISK
        RELEASE     P10                  RELEASE CHANNEL
        TRANSFER    P,12,1               RETURN TO CALLER
*
****************
*CPU SCHEDULERS*
****************
*
 SKED1  QUEUE       CPQ1                 ENTER CPU QUEUE
        SEIZE       CPU1                 GET CPU
        DEPART      CPQ1                 LEAVE CPU QUEUE
        SEIZE       OVH1                 COUNT CPU OVERHEAD
        ADVANCE     V$OVHD,FN$EXPO       DO OVERHEAD
```

```
        RELEASE     OVH1
        TRANSFER    P.11.1                  GIVE CPU TO CALLER
*
 SKED2  QUEUE       CPQ2                    ENTER CPU QUEUE
        SEIZE       CPU2                    GET CPU
        DEPART      CPQ2                    LEAVE CPU QUEUE
        SEIZE       OVH2                    COUNT CPU OVERHEAD
        ADVANCE     V$OVHD,FN$EXPO          DO OVERHEAD
        RELEASE     OVH2
        TRANSFER    P.11.1                  GIVE CPU TO CALLER
*
**********
*TIMING
**********
*
        GENERATE    60000
        TERMINATE   1
*                                      *
****************************************
*                                      *
        START       1
        END
/*
```

```
*************************************
*BATCH ON CPU1,MONITOR + DBMS ON CPU2*
*************************************
*
*
************************************
*VARIABLES FOR COMM MONITOR         *
************************************
*                                   *
 TERMS VARIABLE    100              NUMBER OF TERMINALS
 AVGRE VARIABLE    40               AVG OPERATOR RESPONSE TIME(SECONDS)
 MESTI FVARIABLE   (V$AVGRE/V$TERMS)*1000
*                                   TIME BETWEEN MESSAGES
 AVGTX VARIABLE    10               AVG NUMBER OF CONTEXT BLOCKS PER TERMINAL
 CTXBF VARIABLE    300              NUMBER OF CONTEXT BLOCKS  IN CORE
 CPCT  FVARIABLE   (V$CTXBF/(V$AVGTX*V$TERMS))*100
*                                   PERCENT OF CONTEXT BLOCKS IN CORE
 CTEXT FVARIABLE   (V$AVGTX*FN$EXPD)*FN$CTXF
*                                   NUMBER OF CONTEXT BLOCKS TO BRING IN
 CPUTI VARIABLE    10               AVG CPU TIME NEEDED
 SWAPS VARIABLE    1                AVG PAGE SWAPS PER TRANSACTION
 DBIOS VARIABLE    5                AVG DATA BASE I/O 'S PER TRANSACTION
 AVGML VARIABLE    200              AVG OUTPUT MESSAGE LENGTH
 OMSGL VARIABLE    V$AVGML*FN$EXPD
*                                   OUTPUT MESSAGE LENGTH
 AVGSP VARIABLE    4800            AVG LINE SPEED(BITS/SEC)
 SPEED FVARIABLE   (1/(V$AVGSP/8))*1000
*                                   LINE SPEED
 OTIME FVARIABLE   P5*V$SPEED       TIME FOR OUTPUT MESSAGE
************************************
*VARIABLES FOR BACKGROUND JOBS      *
************************************
 BJOBS VARIABLE    20               NUMBER OF BACKGROUND JOBS
 BCPUT VARIABLE    250              AVG CPU TIME NEEDED
 BPART VARIABLE    300              AVG PARTITION SIZE NEEDED
 BDBTI VARIABLE    5                AVG CPU TIME BETWEEN DATA BASE I/O'S
 BTSLC VARIABLE    100              TIME SLICE
************************************
*OTHER VARIABLES                    *
************************************
 DMSF  VARIABLE    10               DBMS OVERHEAD FACTOR
 OVHD  VARIABLE    1                TASK SWITCH TIME
 DSKXF VARIABLE    2                DISK TRANSFER TIME
 DSKAC VARIABLE    30               DISK ACCESS TIME
 DISKS VARIABLE    12               TOTAL DATA BASE DISK DRIVES
 DISKN VARIABLE    (RN3@V$DISKS+1)+20
*                                   DISK DRIVE
 COMM  VARIABLE    1                COMM TIME BETWEEN CPU'S
*
************************************
*STORAGES                          *
************************************
*
 BATCH STORAGE     2048             MEMORY AVAILABLE TO BATCH
*                                   *
************************************
*FUNCTIONS                          *
************************************
*                                   *
 FXPD  FUNCTION    RN2,C24          EXPONENTIAL DISTRIBUTION
```

```
0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/
.6,.915/.7,1.2/.75,1.38/.8,1.6/.84,1.83/.88,2.12/
.9,2.3/.92,2.52/.94,2.81/.95,2.99/.96,3.2/.97,3.5/
.98,3.9/.99,4.6/.995,5.3/.998,6.2/.999,7/.9997,8

 CTXF   FUNCTION    V$CPCT,C11             CONTEXT PAGING FACTOR
0,1/10,.9/20,.8/30,.7/40,.6/50,.5/
60,.4/70,.3/80,.2/90,.1/100,0
*                                    *
**************************************
*MODEL FOR MONITOR                   *
**************************************
*                                    *
       SIMULATE
       GENERATE   V$MESTI,FN$EXPO     GENERATE INCOMING MESSAGES
       PRIORITY   10                   MAKE HIGHER THAN BATCH
       ASSIGN     1,V$CTEXT           NUMBER OF CONTEXT BLOCKS NEEDED TO BRING
*                                      IN
       ASSIGN      2,V$CPUTI,1
*                                      TOTAL CPU TIME NEEDED
       ASSIGN     3,V$SWAPS,1          TOTAL PAGE SWAPS NEEDED
       ASSIGN     4,V$DBIOS,1          TOTAL DBIO'S NEEDED
       ASSIGN     5,V$OMSGL            AVG OUTPUT MESSAGE LENGTH(CHAR)
       QUEUE      MONO                 JOIN MONITOR QUEUE
       SEIZE      MON                 REQUEST MONITOR
       DEPART     MONO                 LEAVE MONITOR QUEUE
       TRANSFER   SBR,SKED2,11         GET CPU
       SEIZE      CPUM                 COUNT MONITOR CPU TIME
*
*READ IN CONTEXT BLOCKS IF NEEDED
*
       TEST G     P1,K0,CKSW           ANY BLOCKS?
 CTXLP RELEASE    CPU2                 YES
       RELEASE    CPUM
       SEIZE      20
       ADVANCE    V$DSKAC              PAGE OUT OLD CONTEXT BLOCK
       ADVANCE    V$DSKXF
       ADVANCE    V$DSKAC              PAGE IN NEW CONTEXT BLOCK
       ADVANCE    V$DSKXF
       RELEASE    20
       TRANSFER   SBR,SKED2,11         GET CPU AGAIN
       SEIZE      CPUM                 COUNT MONITOR CPU TIME
       LOOP       1,CTXLP              LOOP IF MORE BLOCKS
*                                    *
**************************************
*MAIN LOOP FOR COMM MONITOR           *
**************************************
*                                    *
       ADVANCE    P2                   PROCESS MESSAGE FOR SOME TIME
 CKSW  TEST G     P3,K0,CKDB           IF P3 GT 0 WE NEED PAGE SWAP
*
*DO PROGRAM PAGE I/O'S
*
 CKSW1 RELEASE    CPU2
       RELEASE    CPUM
       SEIZE      20
       ADVANCE    V$DSKAC              DO PAGE SWAP I/O
       ADVANCE    V$DSKXF
       RELEASE    20
       TRANSFER   SBR,SKED2,11         GET CPU AGAIN
```

```
       SEIZE      CPUM                  COUNT MONITOR CPU TIME
       LOOP       3,CKSW1               LOOP IF MORE SWAPS
*
*DO DATA BASE I/O'S
*
 CKDB  TEST G     P4,K0,MEND            IF P4 GT 0 WE NEED DBIO
 CKDB1 RELEASE    CPU2
       RELEASE    CPUM
       TRANSFER   SBR,DBMS,11           DO DATA BASE I/O
       TRANSFER   SBR,SKED2,11          SCHEDULE CPU
       SEIZE      CPUM                  COUNT MONITOR CPU TIME
       LOOP       4,CKDB1               LOOP IF MORE DBIO'S
 MEND  RELEASE    CPU2                  DONE WITH CPU
       RELEASE    CPUM
       RELEASE    MON                  DONE WITH MONITOR
       ADVANCE    V$OTIME              TRANSMIT MESSAGE TO TERMINAL
       TABULATE   XTIME                TABULATE TRANSIT TIMES
       TERMINATE  0                     REMOVE MESSAGE
 XTIME TABLE      M1,0,1000,100
*
*************************************
*MODEL FOR BATCH JOBS                 *
*************************************
*
       GENERATE   ,,,V$BJOBS,,,F         GENERATE BACKGROUND JOBS
 BINIT ASSIGN     1,V$BCPUT,1           TOTAL CPU TIME NEEDED
       ASSIGN     2,V$BPART,1           TOTAL MEMORY NEEDED
       QUEUE      JOBQ
       ENTER      BATCH,P2             REQUEST MEMORY
       DEPART     JOBQ
       TRANSFER   SBR,SKED1,11          SCHEDULE CPU
       SEIZE      CPUB                  COUNT BATCH CPU TIME
*
*************************************
*MAIN LOOP FOR BATCH JOBS             *
*************************************
*
 BSET  ASSIGN     3,V$BDBTI,1           CPU TIME BEFORE DATA BASE I/O
 BLOOP TEST G     P3,V$BTSLC,BDBIO
       ADVANCE    V$BTSLC               PROCESS BATCH FOR 1 TIME SLICE
*
*RELEASE CPU FOR TIME SLICE
*
       RELEASE    CPU1
       RELEASE    CPUB
       TRANSFER   SBR,SKED1,11          SCHEDULE CPU
       SEIZE      CPUB                  COUNT BATCH CPU TIME
       ASSIGN     1-,V$BTSLC
       ASSIGN     3-,V$BTSLC            DECREMENT COUNTERS
       TEST LE    P1,K0,BDONE           IF P1 EQ 0 WE'RE DONE
       TRANSFER   ,BLOOP
*
*DO A DATA BASE I/O
*
 BDBIO ADVANCE    P3                    PROCESS BATCH FOR SOME TIME
       ASSIGN     1-,P3
       RELEASE    CPU1
       RELEASE    CPUB
       ADVANCE    V$COMM                SEND DML TO HOST
       TRANSFER   SBR,DBMS,11           DO DATA BASE I/O
```

```
        ADVANCE     V$COMM                  GET DATA FROM HOST
        TRANSFER    SBR,SKED1,11            SCHEDULE CPU
        SEIZE       CPUB                    COUNT BATCH CPU TIME
 CKEN   TEST LE     P1,K0,BSET              IF P1 EQ 0 WE ARE DONE
*
*JOB COMPLETE, START ANOTHER
*
 BDONE  RELEASE     CPU1
        RELEASE     CPUB
        LEAVE       BATCH,P2                RELEASE MEMORY
        TABULATE    BTIME                   TABULATE BATCH JOB TIME
        TRANSFER    ,BINIT
 BTIME  TABLE       M1,0,10000,100
*
*****************
*DBMS ROUTINE   *
*****************
*
 DBMS   QUEUE       DMSQ
        SEIZE       DMS                     GET DBMS ROUTINE
        DEPART      DMSQ
        TEST LE     Q18,Q19,GET1            PICK CHANNEL WITH SMALLEST QUEUE
        ASSIGN      10,K18                  CHANNEL 1
        TRANSFER    ,GET2
 GET1   ASSIGN      10,K19                  CHANNEL 2
 GET2   ASSIGN      9,V$DISKN               ASSIGN DISK NUMBER
        ASSIGN      12,P11                  SAVE RETURN ADDR.
        TRANSFER    SBR,SKED2,11            GET CPU
        SEIZE       CPUD                    COUNT DMS OVERHEAD
        ADVANCE     V$DMSE,FN$EXPO          DO OVERHEAD
        RELEASE     CPUD
        RELEASE     CPU2
        RELEASE     DMS
        QUEUE       P9                      ENTER DISK QUEUE
        SEIZE       P9                      GET DISK
        DEPART      P9                      LEAVE DISK QUEUE
        QUEUE       P10                     ENTER CHANNEL QUEUE
        SEIZE       P10                     GET CHANNEL
        DEPART      P10                     LEAVE CHANNEL QUEUE
        RELEASE     P10                     RELEASE CHANNEL
        ADVANCE     V$DSKAC                 DO DISK ACCESS
        QUEUE       P10                     ENTER CHANNEL QUEUE
        SEIZE       P10                     GET CHANNEL
        DEPART      P10                     LEAVE CHANNEL QUEUE
        ADVANCE     V$DSKXF                 TRANSFER DATA
        RELEASE     P9                      RELEASE DISK
        RELEASE     P10                     RELEASE CHANNEL
        TRANSFER    P,12,1                  RETURN TO CALLER
*
****************
*CPU SCHEDULERS*
****************
*
 SKED1  QUEUE       CPQ1                    ENTER CPU QUEUE
        SEIZE       CPU1                    GET CPU
        DEPART      CPQ1                    LEAVE CPU QUEUE
        SEIZE       OVH1                    COUNT CPU OVERHEAD
        ADVANCE     V$OVHD,FN$EXPO          DO OVERHEAD
        RELEASE     OVH1
        TRANSFER    P,11,1                  GIVE CPU TO CALLER
```

```
*
 SKED2 QUEUE       CPQ2                   ENTER CPU QUEUE
       SEIZE       CPU2                   GET CPU
       DEPART      CPQ2                   LEAVE CPU QUEUE
       SEIZE       OVH2                   COUNT CPU OVERHEAD
       ADVANCE     V$OVHD,FN$EXPO         DO OVERHEAD
       RELEASE     OVH2
       TRANSFER    P,11,1                 GIVE CPU TO CALLER
*
**********
*TIMING
**********
*
       GENERATE    60000
       TERMINATE   1
*                                      *
****************************************
*                                      *
       START       1
       END
/*
```

```
*********************************************
*BATCH ON CPU1, MONITOR ON CPU2,DBMS ON CPU3*
*********************************************
*
*
*************************************
*VARIABLES FOR COMM MONITOR         *
*************************************
*                                   *
 TERMS VARIABLE    100                NUMBER OF TERMINALS
 AVGRE VARIABLE    40                 AVG OPERATOR RESPONSE TIME(SECONDS)
 MESTI FVARIABLE   (V$AVGRE/V$TERMS)*1000
*                                     TIME BETWEEN MESSAGES
 AVGTX VARIABLE    10                 AVG NUMBER OF CONTEXT BLOCKS PER TERMINAL
 CTXBF VARIABLE    300                NUMBER OF CONTEXT BLOCKS  IN CORE
 CPCT  FVARIABLE   (V$CTXBF/(V$AVGTX*V$TERMS))*100
*                                     PERCENT OF CONTEXT BLOCKS IN CORE
 CTEXT FVARIABLE   (V$AVGTX*FN$EXPO)*FN$CTXF
*                                     NUMBER OF CONTEXT BLOCKS TO BRING IN
 CPUTI VARIABLE    10                 AVG CPU TIME NEEDED
 SWAPS VARIABLE    1                  AVG PAGE SWAPS PER TRANSACTION
 DBIOS VARIABLE    5                  AVG DATA BASE I/O 'S PER TRANSACTION
 AVGML VARIABLE    200                AVG OUTPUT MESSAGE LENGTH
 OMSGL VARIABLE    V$AVGML*FN$EXPO
*                                     OUTPUT MESSAGE LENGTH
 AVGSP VARIABLE    4800              AVG LINE SPEED(BITS/SEC)
 SPEED FVARIABLE   (1/(V$AVGSP/8))*1000
*                                     LINE SPEED
 OTIME FVARIABLE   P5*V$SPEED         TIME FOR OUTPUT MESSAGE
*************************************
*VARIABLES FOR BACKGROUND JOBS      *
*************************************
 BJOBS VARIABLE    20                 NUMBER OF BACKGROUND JOBS
 BCPUT VARIABLE    250                AVG CPU TIME NEEDED
 BPART VARIABLE    300                AVG PARTITION SIZE NEEDED
 BDBTI VARIABLE    5                  AVG CPU TIME BETWEEN DATA BASE I/O'S
 BTSLC VARIABLE    100                TIME SLICE
*************************************
*OTHER VARIABLES                    *
*************************************
 DMSF  VARIABLE    10                 DBMS OVERHEAD FACTOR
 OVHD  VARIABLE    1                  TASK SWITCH TIME
 DSKXF VARIABLE    2                  DISK TRANSFER TIME
 DSKAC VARIABLE    30                 DISK ACCESS TIME
 DISKS VARIABLE    12                 TOTAL DATA BASE DISK DRIVES
 DISKN VARIABLE    (RN3@V$DISKS+1)+20
*                                     DISK DRIVE
 COMM  VARIABLE    1                  COMM TIME BETWEEN CPU'S
*
*************************************
*STORAGES                           *
*************************************
*
 BATCH STORAGE     2048               MEMORY AVAILABLE TO BATCH
*                                   *
*************************************
*FUNCTIONS                          *
*************************************
*                                   *
 EXPO  FUNCTION    RN2,C24           EXPONENTIAL DISTRIBUTION
```

```
0.0/.1..104/.2..222/.3..355/.4..509/.5..69/
.6..915/.7.1.2/.75.1.38/.8.1.6/.84.1.83/.88.2.12/
.9.2.3/.92.2.52/.94.2.81/.95.2.99/.96.3.2/.97.3.5/
.98.3.9/.99.4.6/.995.5.3/.998.6.2/.999.7/.9997.8

 CTXF  FUNCTION   V$CPCT.C11          CONTEXT PAGING FACTOR
0.1/10..9/20..8/30..7/40..6/50..5/
60..4/70..3/80..2/90..1/100.0
*                                    *
**************************************
*MODEL FOR MONITOR                   *
**************************************
*                                    *
       SIMULATE
       GENERATE    V$MESTI,FN$EXPO   GENERATE INCOMING MESSAGES
       PRIORITY    10                 MAKE HIGHER THAN BATCH
       ASSIGN      1.V$CTEXT         NUMBER OF CONTEXT BLOCKS NEEDED TO BRING
*                                     IN
       ASSIGN       2.V$CPUTI.1
*                                     TOTAL CPU TIME NEEDED
       ASSIGN      3.V$SWAPS.1        TOTAL PAGE SWAPS NEEDED
       ASSIGN      4.V$DBIOS.1        TOTAL DBIO'S NEEDED
       ASSIGN      5.V$OMSGL          AVG OUTPUT MESSAGE LENGTH(CHAR)
       QUEUE       MONQ               JOIN MONITOR QUEUE
       SEIZE       MON               REQUEST MONITOR
       DEPART      MONQ               LEAVE MONITOR QUEUE
       TRANSFER    SBR.SKED2.11       GET CPU
       SEIZE       CPUM               COUNT MONITOR CPU TIME
*
*READ IN CONTEXT BLOCKS IF NEEDED
*
       TEST G      P1.K0.CKSW         ANY BLOCKS?
 CTXLP RELEASE     CPU2               YES
       RELEASE     CPUM
       SEIZE       20
       ADVANCE     V$DSKAC            PAGE OUT OLD CONTEXT BLOCK
       ADVANCE     V$DSKXF
       ADVANCE     V$DSKAC            PAGE IN NEW CONTEXT BLOCK
       ADVANCE     V$DSKXF
       RELEASE     20
       TRANSFER    SBR.SKED2.11       GET CPU AGAIN
       SEIZE       CPUM               COUNT MONITOR CPU TIME
       LOOP        1.CTXLP            LOOP IF MORE BLOCKS
*                                    *
**************************************
*MAIN LOOP FOR COMM MONITOR          *
**************************************
*                                    *
       ADVANCE     P2                 PROCESS MESSAGE FOR SOME TIME
 CKSW  TEST G      P3.K0.CKDB         IF P3 GT 0 WE NEED PAGE SWAP
*
*DO PROGRAM PAGE I/O'S
*
 CKSW1 RELEASE     CPU2
       RELEASE     CPUM
       SEIZE       20
       ADVANCE     V$DSKAC            DO PAGE SWAP I/O
       ADVANCE     V$DSKXF
       RELEASE     20
       TRANSFER    SBR.SKED2.11       GET CPU AGAIN
```

```
        SEIZE      CPUM                     COUNT MONITOR CPU TIME
        LOOP       3,CKSW1                  LOOP IF MORE SWAPS
*
*DO DATA BASE I/O'S
*
 CKDB   TEST G     P4,KO,MEND               IF P4 GT 0 WE NEED DBIO
 CKDB1  RELEASE    CPU2
        RELEASE    CPUM
        ADVANCE    V$COMM                   SEND DML TO HOST
        TRANSFER   SBR,DBMS,11              DO DATA BASE I/O
        ADVANCE    V$COMM                   GET DATA FROM HOST
        TRANSFER   SBR,SKED2,11             SCHEDULE CPU
        SEIZE      CPUM                     COUNT MONITOR CPU TIME
        LOOP       4,CKDB1                  LOOP IF MORE DBIO'S
 MEND   RELEASE    CPU2                     DONE WITH CPU
        RELEASE    CPUM
        RELEASE    MON                     DONE WITH MONITOR
        ADVANCE    V$OTIME                 TRANSMIT MESSAGE TO TERMINAL
        TABULATE   XTIME                   TABULATE TRANSIT TIMES
        TERMINATE  0                        REMOVE MESSAGE
 XTIME  TABLE      M1,0,1000,100
*
***************************************
*MODEL FOR BATCH JOBS                 *
***************************************
*
        GENERATE   ,,,V$BJOBS,,,F            GENERATE BACKGROUND JOBS
 BINIT  ASSIGN     1,V$BCPUT,1              TOTAL CPU TIME NEEDED
        ASSIGN     2,V$BPART,1              TOTAL MEMORY NEEDED
        QUEUE      JOBQ
        ENTER      BATCH,P2                 REQUEST MEMORY
        DEPART     JOBQ
        TRANSFER   SBR,SKED1,11             SCHEDULE CPU
        SEIZE      CPUB                     COUNT BATCH CPU TIME
*
***************************************
*MAIN LOOP FOR BATCH JOBS             *
***************************************
*
 BSET   ASSIGN     3,V$BDBTI,1              CPU TIME BEFORE DATA BASE I/O
 BLOOP  TEST G     P3,V$BTSLC,BDBIO
        ADVANCE    V$BTSLC                  PROCESS BATCH FOR 1 TIME SLICE
*
*RELEASE CPU FOR TIME SLICE
*
        RELEASE    CPU1
        RELEASE    CPUB
        TRANSFER   SBR,SKED1,11             SCHEDULE CPU
        SEIZE      CPUB                     COUNT BATCH CPU TIME
        ASSIGN     1-,V$BTSLC
        ASSIGN     3-,V$BTSLC               DECREMENT COUNTERS
        TEST LE    P1,KO,BDONE              IF P1 EQ 0 WE'RE DONE
        TRANSFER   ,BLOOP
*
*DO A DATA BASE I/O
*
 BDBIO  ADVANCE    P3                       PROCESS BATCH FOR SOME TIME
        ASSIGN     1-,P3
        RELEASE    CPU1
        RELEASE    CPUB
```

```
        ADVANCE     V$COMM                  SEND DML TO HOST
        TRANSFER    SBR,DBMS,11             DO DATA BASE I/O
        ADVANCE     V$COMM                  GET DATA FROM HOST
        TRANSFER    SBR,SKED1,11            SCHEDULE CPU
        SEIZE       CPUB                    COUNT BATCH CPU TIME
 CKEN   TEST LE     P1,K0,BSET              IF P1 EQ 0 WE ARE DONE
*
*JOB COMPLETE, START ANOTHER
*
 BDONE  RELEASE     CPU1
        RELEASE     CPUB
        LEAVE       BATCH,P2                RELEASE MEMORY
        TABULATE    BTIME                   TABULATE BATCH JOB TIME
        TRANSFER    ,BINIT
 BTIME  TABLE       M1,0,10000,100
*
******************
*DBMS ROUTINE    *
******************
*
 DBMS   QUEUE       DMSQ
        SEIZE       DMS                     GET DBMS ROUTINE
        DEPART      DMSQ
        TEST LE     Q18,Q19,GET1            PICK CHANNEL WITH SMALLEST QUEUE
        ASSIGN      10,K18                  CHANNEL 1
        TRANSFER    ,GET2
 GET1   ASSIGN      10,K19                  CHANNEL 2
 GET2   ASSIGN      9,V$DISKN               ASSIGN DISK NUMBER
        ASSIGN      12,P11                  SAVE RETURN ADDR.
        TRANSFER    SBR,SKED3,11            GET CPU
        SEIZE       CPUD                    COUNT DMS OVERHEAD
        ADVANCE     V$DMSF,FN$EXPO          DO OVERHEAD
        RELEASE     CPUD
        RELEASE     CPU3
        RELEASE     DMS
        QUEUE       P9                      ENTER DISK QUEUE
        SEIZE       P9                      GET DISK
        DEPART      P9                      LEAVE DISK QUEUE
        QUEUE       P10                     ENTER CHANNEL QUEUE
        SEIZE       P10                     GET CHANNEL
        DEPART      P10                     LEAVE CHANNEL QUEUE
        RELEASE     P10                     RELEASE CHANNEL
        ADVANCE     V$DSKAC                 DO DISK ACCESS
        QUEUE       P10                     ENTER CHANNEL QUEUE
        SEIZE       P10                     GET CHANNEL
        DEPART      P10                     LEAVE CHANNEL QUEUE
        ADVANCE     V$DSKXF                 TRANSFER DATA
        RELEASE     P9                      RELEASE DISK
        RELEASE     P10                     RELEASE CHANNEL
        TRANSFER    P,12,1                  RETURN TO CALLER
*
****************
*CPU SCHEDULERS*
****************
*
 SKED1  QUEUE       CPQ1                    ENTER CPU QUEUE
        SEIZE       CPU1                    GET CPU
        DEPART      CPQ1                    LEAVE CPU QUEUE
        SEIZE       OVH1                    COUNT CPU OVERHEAD
        ADVANCE     V$OVHD,FN$EXPO          DO OVERHEAD
```

```
         RELEASE      OVH1
         TRANSFER     P,11,1                  GIVE CPU TO CALLER
*
 SKED2   QUEUE        CP02                    ENTER CPU QUEUE
         SEIZE        CP02                    GET CPU
         DEPART       CP02                    LEAVE CPU QUEUE
         SEIZE        OVH2                    COUNT CPU OVERHEAD
         ADVANCE      V$OVHD,FN$EXPD          DO OVERHEAD
         RELEASE      OVH2
         TRANSFER     P,11,1                  GIVE CPU TO CALLER
*
 SKED3   QUEUE        CP03                    ENTER CPU QUEUE
         SEIZE        CP03                    GET CPU
         DEPART       CP03                    LEAVE CPU QUEUE
         SEIZE        OVH3                    COUNT CPU OVERHEAD
         ADVANCE      V$OVHD,FN$EXPD          DO OVERHEAD
         RELEASE      OVH3
         TRANSFER     P,11,1                  GIVE CPU TO CALLER
*
**********
*TIMING
**********
*
         GENERATE     60000
         TERMINATE    1
*                                      *
****************************************
*                                      *
         START        1
         END
/*
```

A SIMULATION STUDY OF ALTERNATIVES TO UPGRADING LARGE COMPUTER SYSTEMS

by

DANIEL E. KREIMER

B.S. University of Missouri, Columbia, 1970

AN ABSTRACT OF A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

Kansas State University
Manhattan, Kansas

1977

ABSTRACT

This report describes a simulation study of alternatives to upgrading large computer systems. A model of a system with a telecommunications monitor, a batch job stream and a data base manager is presented. The parameters of this model are adjusted so that the system resources are saturated and the throughput of the system is then measured.

The use of an additional processor(s) to increase throughput is considered. Models of the following configurations are presented:

1) Telecommunications monitor on a front-end processor.
2) Data base manager on a back-end processor.
3) Telecommunications monitor and data base manager on a front-end processor.
4) Telecommunications monitor on a front-end processor and data base manager on a back-end processor.

The throughput of each of these models is measured and compared to the throughput of the original system. The advantages and disadvantages of each configuration are discussed in terms of cost-effectiveness and increase in system capability.