A SIMULATION STUDY COMPARING FIVE CONSISTENCY ALGORITHMS FOR A MULTICOMPUTER-REDUNDANT DATA BASE ENVIRONMENT

by

CALVIN A. BUZZELL

B.S., California State Polytechnic University, Pomona, 1964

A MASTER'S REPORT

submitted in partial fulfillment of the

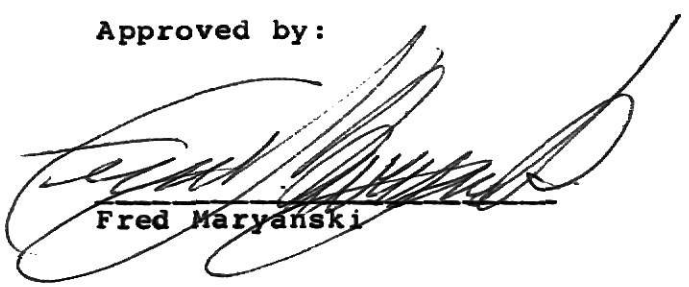
requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas
1979

Approved by:



Fred Maryanski

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

Advances in computer hardware, increasing availability of minicomputers, and the advantages of a distributed data base networks have resulted in significant interest in the efficiency of methodologies to maintain consistent copies of geographically separate yet redundant data in a data base network.

This is a simulation study comparing the responsiveness of five update algorithms in a multiple host, multiple back-end processor, redundant data base environment.

This chapter gives a brief survey of the advantages of a Distributed Data Base Management System (DBMS), the advantages of maintaining redundant copies of data in a data base network, and the problems associated with update algorithms to maintain consistent data bases. The purpose of this study is defined: to compare the performance of five update methodologies and to examine the parameters that affect the performance of each algorithm.

1.2 ADVANTAGES OF DISTRIBUTED DATA BASE MANAGEMENT SYSTEMS (DBMS)

Rapid advances in microelectronics and recent increased availability of relatively low cost minicomputers have increased the potential for multiple separate data base

systems to be combined into one network. Despite the overhead of intercomputer communications and software management, a network has significant advantages when compared to a single, large, conventional data base machine or to maintaining separate, redundant data base machines. Definitions of the various forms of a distributed DBMS are given in Chapter 2, Terminology and Parameters.

Several authors (11,17) describe the advantages of a distributed versus a centralized DBMS. These advantages are summarized below:

1. Increased reliability.

   Failure at one geographical location results in degraded performance, not total system support loss. Multiple copies of the data assure a reliable backup in case of system failure at one node.

2. Faster access.

   Communications delays are reduced for dispersed systems because data can be stored near the users.

3. Ease of expansion.

   A network is amenable to modular stepwise scaling of data base capability.

4. Increased access efficiency.

   Data may be accessed from a machine closest to the user or by the least busy machine in the network.

5. *Sharing of the work load.*

   The computational load is shared by all machines in the network, resulting in increased distribution of the computational work load.

Disadvantages or problems of a distributed DBMS depend on the implementation concerned. Several authors discuss the advantages and problems of distributed networks (11,17).

1.3 ADVANTAGES OF MINICOMPUTERS AS BACK-END PROCESSORS

There are obvious economic advantages of using minicomputers to process data base applications. Maryanski et al. (13), Maryanski (11), and Canaday et al. (4) discuss advantages of using minicomputers for data base functions. Included among these advantages are:

1. Resources of host freed for less time consuming tasks.
2. Reduction in requirement for software overhead for data base functions at the host.
3. Greater concurrency is achieved within the system.
4. Increased security by carefully protecting access of an application program on the back-end.
5. Increased integrity with the ability of the host and back-end to verify each other's operations.

6. An economical alternative to upgrading a mainframe computer.

1.4 THE BACK-END DBMS

In a multiprocessor back-end system, a host machine is coupled to a back-end machine with multiple processors. A significant difference in this configuration from the normal distributed DBMS network is that (1) the external links between back-ends are eliminated and (2) requests cannot originate at the back-end nodes.

Maryanski et al. (12) describe a system architecture where the back-end DBMS can be extended into a multi-computer environment with both multiple hosts and multiple back-ends. This configuration is the environment of this study. A more complex architecture is the inclusion of machines that can serve as both a host and as a back-end, or a bi-functional machine. The above report (12) lists the required steps to complete execution of a data base command in a back-end DBMS network (see Figure 1.1):

1. A data base command in the application program on the host machine produces a call to the host interface (HINT).
2. HINT formats the data base command into a message and instructs the communication system.
3. The communication link transmits the message.
4. BINT receives and unpacks the message.

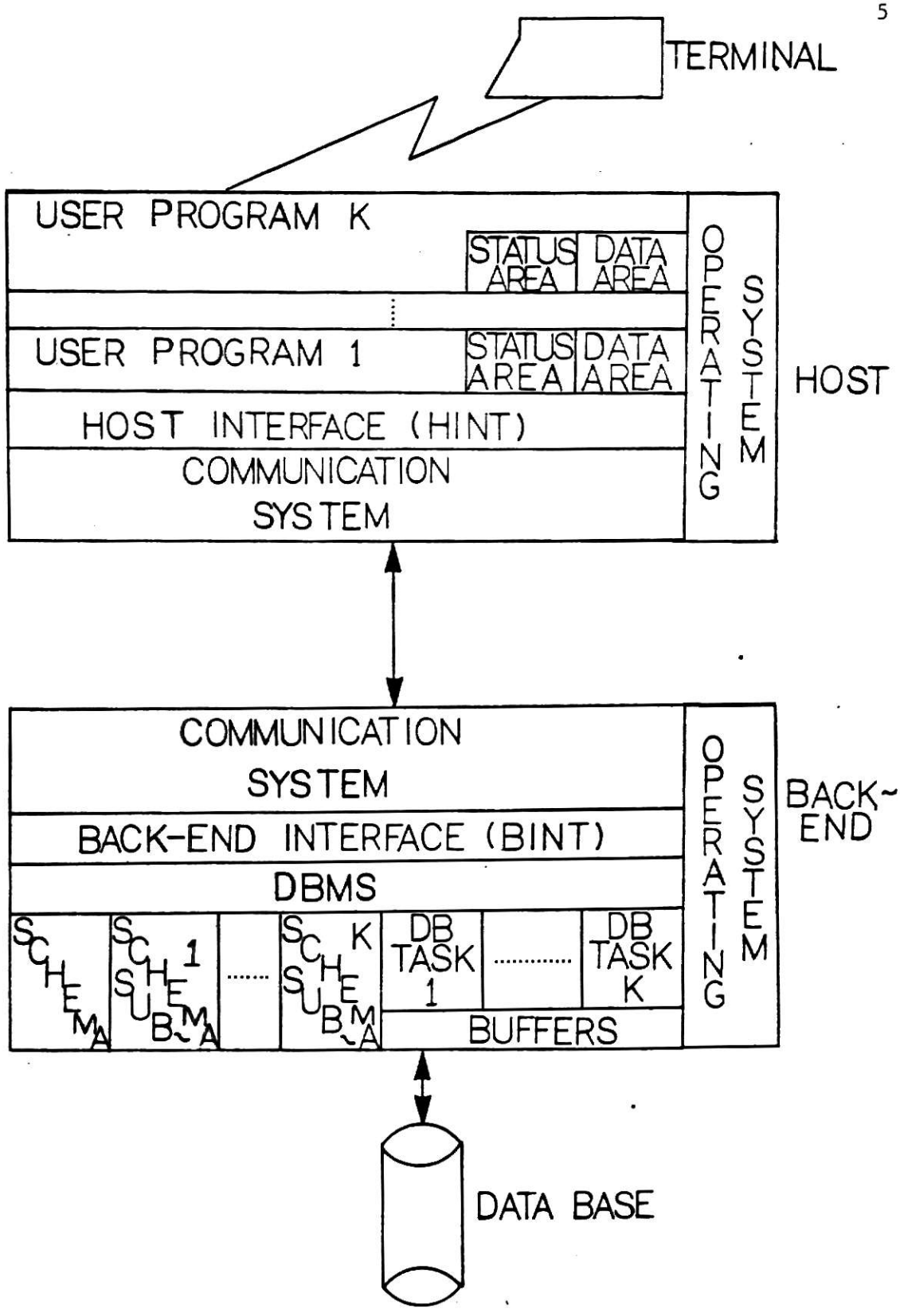


FIGURE 1.1

BACK-END DATA BASE MANAGEMENT SYSTEM (DBMS)

5. BINT transmits the command to the appropriate data base task.
6. The data base task calls DBMS to perform the operation.
7. DBMS executes the data base command, perhaps referencing secondary storage through the operation system.
8. Upon completion of the data base command, the data base task transmits the data and status to BINT for eventual return to the application program.
9. The result of the command passes through BINT, the communication systems, and HINT before reaching the application program.

Maryanski and Kreimer (14) in a simulation study of a back-end DBMS concluded that adding processors provides performance benefits if the demands for the function are high. Performance factors of various consistency algorithms were studied by Norsworthy (15) who also concluded that addition of back-end processors increased system throughput in a heavily utilized system. (Norsworthy's work is discussed in Chapter 3).

1.5 DISTRIBUTION OF DATA

Various approaches may be used to store data within the system architecture, as discussed by Rothnie and Goodman

(17). For example:

1. Unique Subsets:

   Each data base contains a unique subset of the data base files.

2. Partially Redundant:

   Each data base contain subsets of the data base that may overlap or contain data stored redundantly in other data bases.

3. Fully Redundant:

   Each data base contains a complete copy of the entire data base. This is referred to as the fully redundant approach.

The fully redundant case is the approach simulated by this study. The advantages include those discussed in Section 1.2. As the approach to the distribution of data varies from the fully redundant approach, reliability decreases because failure at the node that stores the unique files will result in failure of all applications that require access to the files. As utilization of a network increases, an additional node may be added to absorb the work load within a redundant data base. A system with increased demand for unique files must increase its capacity of the data base machine at the storage site with increased activity. See references (3,10,17) for a more complete discussion of this comparison.

1.6 SYNCHRONIZING UPDATES OF REDUNDANT DATA BASES

Synchronization of updates in a distributed DBMS is necessary to ensure consistency of the data bases and limit excessive delays while updates are correctly made at each data base.

A simple locking of the updated portion of the data base (as is done in a single, general purpose computer) becomes impractical as the number of machines (and copies of the data) increase. The communications delays become expensive to request permission to lock, grant permission, to lock, send update, acknowledge update, and to release the locks (17).

The problem of distributed DBMS update synchronization methodology is discussed by Alsberg et al. (1), Bernstein et al. (3), Ellis (5), Johnson and Thomas (9), Rosenkrantz et al. (16), Rothnie and Goodman (17), Stearns et al. (19), Stonebraker and Neuhold (20), and Thomas (21).

By examining a single host, multiple back-end environment in a simulations study at Kansas State University, Norsworthy (15) compared five methodologies which deal with the consistency problem of redundant data bases. The five models were the CODASYL (6), Bernstein (3), Johnson and Thomas (9), Ellis (5), and Hybrid (10,15).

As a continuation of the Norsworthy simulation study, this report examines the five consistency algorithms, expanding the environment to include multiple host networks. The purpose of this study is to examine the effect of

multiple hosts on performance of the DBMS network.

1.7 SUMMARY

The advantages of a back-end DBMS have been discussed. Although few data base networks can be expected to be fully redundant or to have homogeneous hardware configurations, a simulations study assuming these conditions allows the performance of the update methodologies to be compared and their similarities and differences to be studied.

The parameters of this study are defined in Chapter 2, the five simulation models are described in Chapter 3, and the implementation of the models is described in Chapter 4. The data obtained by this study is combined with the data obtained by the Norsworthy study, and using multiple linear regression, mathematical models of the simulation models were developed. A description of the mathematical models are presented in Chapter 5 and an analysis of the results of the study is contained in Chapter 6. Chapter 7 explains the conclusions of the study.

CHAPTER 2

TERMINOLOGY AND PARAMETERS

2.1 OVERVIEW

Terminology is discussed, followed by definitions of system and experimental parameters used in this study. Terms are defined in Section 2.2; constants used in the simulation are discussed in Section 2.3; and the simulation's variables are defined in Section 2.4.

2.2 TERMINOLOGY

Distributed Data Base Management System (DBMS) - defined by architecture (11,17)

1) Single, General Purpose Computers.
2) Data Base Machines.
   Special-purpose processors whose function is data management. A data base machine can be a back-end processor (see below).
3) Back-end Machine.
   When the above two computers are combined into one network, the dedicated data base processor is referred to as a back-end machine. A back-end processor frees the resources of the mainframe (host) CPU.
4) Special Purpose Distributed DBMS.

A network of identical data base machines, designed specifically for the data base system.

5) Homogeneous Distributed DBMS.

Conventional facilities for communications between machines. Identical computers with specialized software that allows communication between tasks residing in separate machines.

6) Heterogeneous Distributed DBMS.

A network of processors from different vendors.

7) Multiple Software, Heterogeneous Distributed DBMS.

A network comprised of processors from different vendors and having the ability for users to access the data base with more than one language.

Components of a Distributed DBMS - defined by function (11)

1) Front-end Machine.

A machine interfaces with the user and the host machine; receives input, transmits output.

2) Host Machine.

Executes application programs.

3) Backend Machine.

Controls data access by execution of data base operations.

3.1) Primary Back-end.

The back-end processor that is selected to receive the specific modification request

being transmitted to the DBMS. Although only one back-end is designated primary for a specific user request, some update algorithms designate only one back-end to be the primary back-end for all modifications.

3.2) Secondary Back-end.

A secondary back-end is a back-end processor that receives a specific modification request after the data at the primary back-end has been updated. Depending on the update algorithm considered, a back-end may be the primary back-end for some modification requests and the secondary back-end for others.

4) Bi-functional Machine.

Combines host and back-end functions.

Timestamp

A timestamp is the absolute clock time that a specific request is assigned or "tagged" when it enters the data base system.

Redundant (Replicated) Data Bases

A data base is either partitioned or replicated. A partitioned data base is spread across several computers. A replicated data base stores some portions of the data base redundantly at different nodes in the network (10). A fully redundant data base stores a complete copy of the data base at all

nodes in the network. The data base system in this work assumes the fully redundant data base in all simulation experiments.

2.3 SYSTEM PARAMETERS

The parameters discussed in this section were constant throughout the simulation experiments discussed in this paper. The variable parameters are discussed in Section 2.4.

Requests Per User

Each user issued from one to seventeen requests (average being eight).

User Terminals

The number of user terminals attached to the data base network for all simulations was held constant at eight.

Users

Normally, there were fifteen users per host assigned randomly to the eight available terminals for each run. All user requests were routed through a host machine, selected randomly, and the structure of the data base was considered transparent to the user.

Partitions

Each backend processor had two partitions allocated to executing jobs and a buffer to hold queued jobs.

Velocity of Communications

50 K Baud.

Buffered Transmissions per Request

Ninety percent of all requests fit in one buffered transmission and the remaining ten percent of the requests required two transmissions. The requests were randomly assigned a length as they entered the network.

Time

The basic time unit during the simulations was one millisecond.

2.4 EXPERIMENTAL PARAMETERS

The parameters discussed in this section include dependent and independent variables in the models of the data base system.

Response Time, t_R

The response time, t_R, is the dependent variable measured during each GPSS simulation. The response time is the absolute clock time given in the GPSS output when all

requests to the data base have been answered and all data base copies are identical.

Mean Response Time, $\bar{t}_R$

The arithmetic mean response time $\bar{t}_R$, is the average of six samples of one simulation model. The samples vary only in the random number generator multipliers used (RMULTs). The RMULTs used in this experiment were 31, 37, 743, 6352, 92576, and 14523.

Requests, R

The number of requests, or R, varied as follows:

1 host CPU	15 requests per simulation
2 host CPU's	30 requests per simulation
4 host CPU's	60 requests per simulation

The number of requests were varied as above in order for the requests per host remain constant and the length of the job stream also remain constant (see Section 2.3).

Request Interval, f

The frequency of requests to the data base, f, is defined as the seconds per request to the data base. The variable f was varied to maintain a constant work load per host CPU. The interval was varied as follows:

1 host CPU	3/2 sec/request
2 host CPU's	3/4 sec/request
4 host CPU's	3/8 sec/request

During the simulations, the intervals were varied as above, + or - 50 milliseconds.

Elapsed Time, t_S

With respect to the number of hosts, the workload was held constant for each simulation run. The elapsed time was a constant 22.5 seconds and the number of requests and requests per second were varied to produce 15 requests per host machine for each simulation (except two simulations discussed in Section 4.2). Elapsed time is computed during some SAS PLOT programs (Appendix 7) as follows:

$$t_S = f * R$$

Average Time Lag $\bar{t}_d$

The average time per request to complete all tasks requested by the job stream, beginning at the end of the job stream and ending when all requests have been answered and all copies of the data base are identical. The average time lag is defined as follows:

$$\bar{t}_d = (\bar{t}_R - t_S) * 1000 / R \quad \text{(in milliseconds)}$$

In an ideal system with infinitely fast updates, $\bar{t}_d$ would approach zero.

Back-end, B

The number of Back-end processors were varied in the simulations as follows: 2, 4, and 8.

Host CPU, H

The number of host CPU's were varied as follows: 1, 2, and 4.

Modification Requests, M

Modifications are requests to update, insert, or delete records from the data base. Possible values of M in this study are 20%, 35%, and 50%. The unit of M when not given is percent.

Priority Requests, P

Priority requests are requests that are randomly assigned a higher priority than normal requests to the data base. Possible values of P in this study are zero or no priority and 10%. The unit of P where not given is percent.

CHAPTER 3

THE SIMULATION MODELS

3.1 OVERVIEW

A description of each of the five GPSS simulations of the five update algorithms for redundant data bases is given in successive sections of this chapter.

The five models of the data base network are based on the following previously presented algorithms: Bernstein (3), CODASYL (6), Ellis (5), Hybrid (10,15), and Johnson and Thomas (9). The GPSS V simulation models were developed by previous researchers at Kansas State, primarily by Norsworthy (15). The models described in this paper are extensions of the previous simulation models. Instead of single host data base network, all models were simulated in a multi-host environment (2-4 hosts). Listings of the GPSS V programs are at Appendices 1-5. Analysis of the data obtained from this study in subsequent chapters includes data from the previous simulation experiments. Data was extracted from the Norsworthy Master's Report (15).

3.2 CODASYL MODEL

The CODASYL Model differs considerably from the remaining models discussed. Only one primary data base is updated until all requests have been received by the system (end of the work day). Subsequently, all modification

requests are sent from the single primary data base to the secondary back-end processors. Therefore, the CODASYL Model performs updates on only one data base and only one data base is current throughout the operational work day.

The obvious disadvantage of this model is in the delay in updating all data bases. As a consequence, the secondary data bases responding to queries respond with progressively outdated data until the beginning of a new work period.

A high level description of the simulation model is at Figure 3.1. A listing of the GPSS program is at Appendix 1.

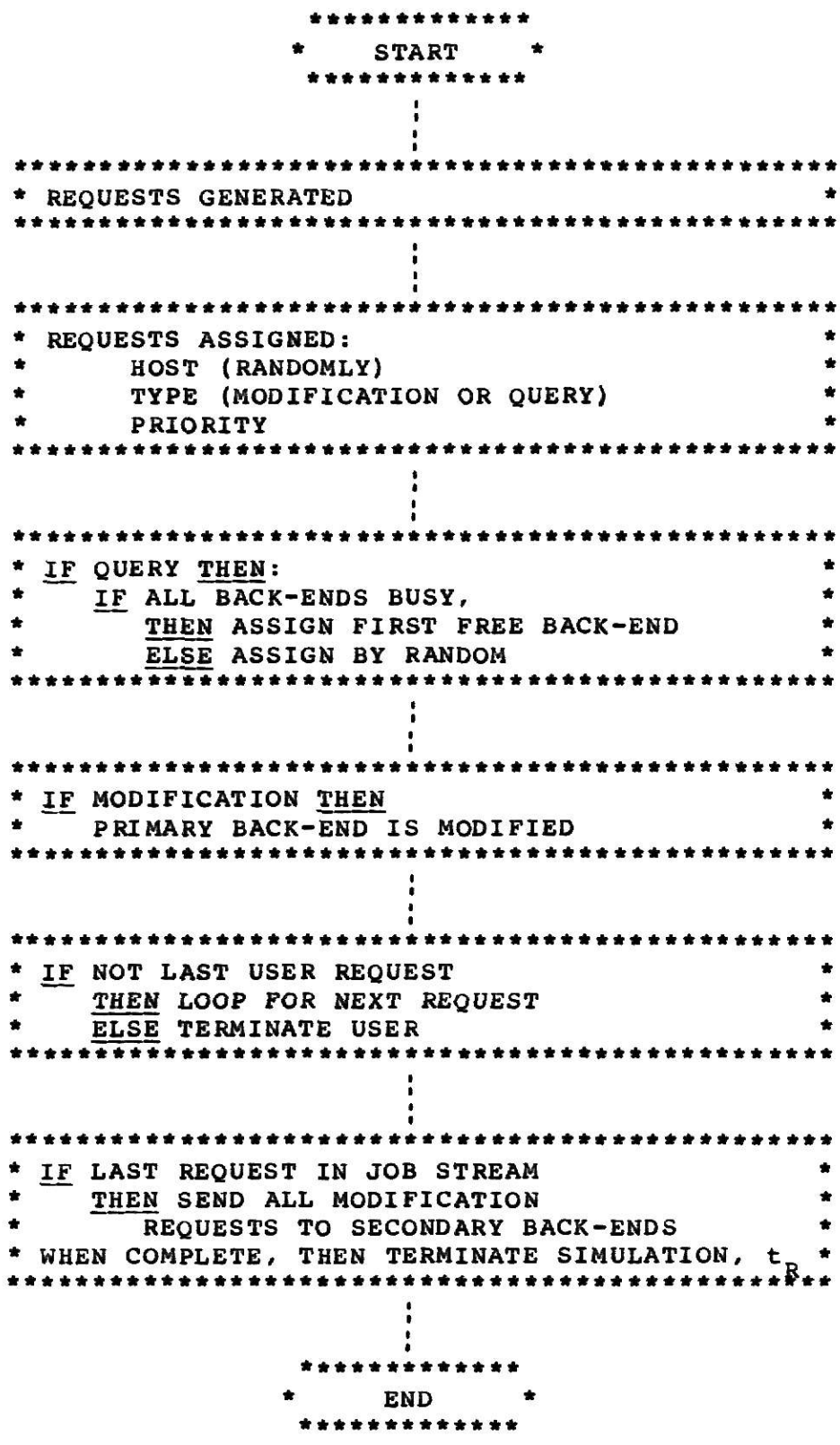
For comparison of this model with the others tested, the end of the job stream signaled the end of the work day and start of the secondary updates.

3.3 ELLIS MODEL

The Ellis Model (5) allows any data base to accept a modification request. The back-end requests permission to modify the specific record and the secondary back-ends grant their approval if not currently processing a modification request on the same record. If the primary back-end receives a negative acknowledgment from any other node, it then delays the request and resubmits the request later.

The simulation implementation differs from the Ellis algorithm, in that, instead of a modification request being processed directly by the node receiving the request, each request is sent to a host and then simultaneously sent to

Figure 3.1
CODASYL MODEL



join all job queues. The assumption is made that the requests will not get out of sequence if they are processed through a host CPU. This assumption was made in both the single host environment studied by Norsworthy (15) and in the multi-host environment of this study.

The algorithm for the Ellis Model is at Figure 3.2. During modification the back-end processors block all read requests until modifications are complete and, consequently, an expected delay results with increased modification requests.

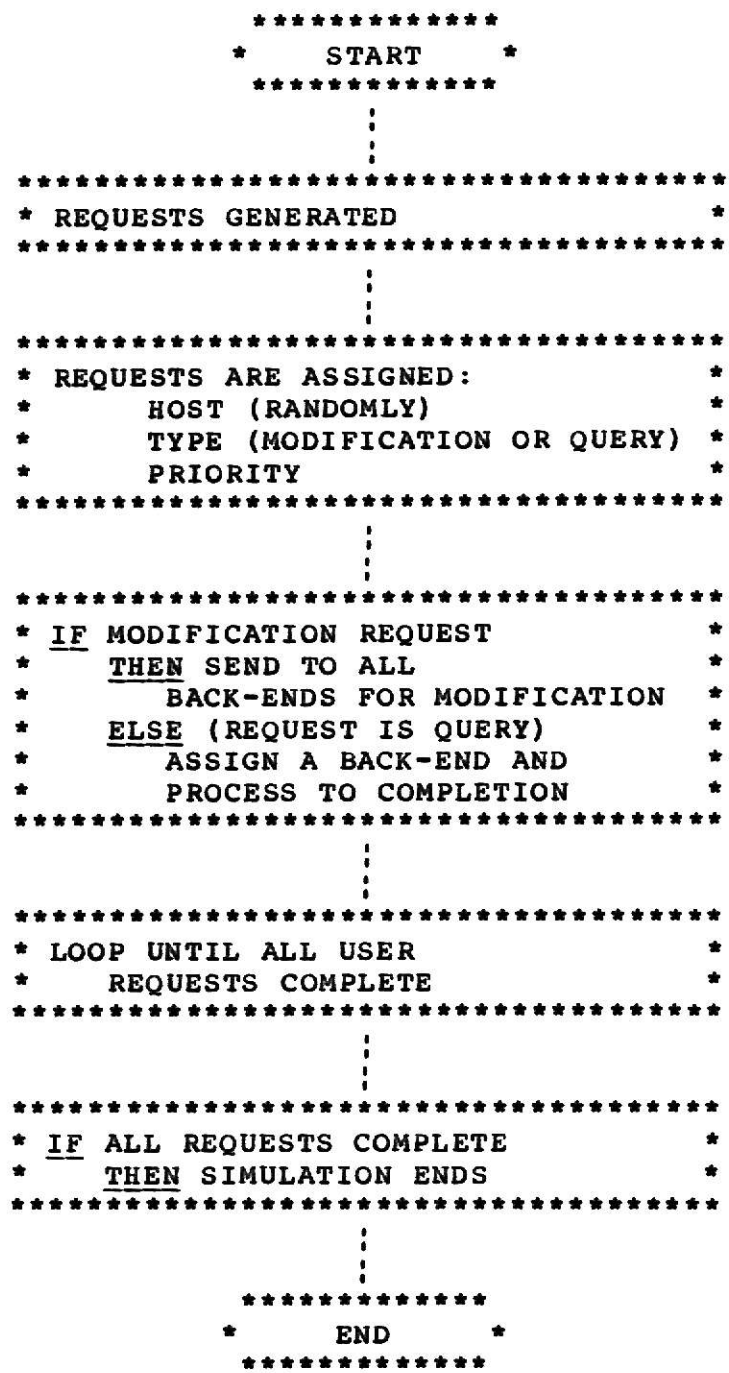
A listing of the GPSS program is at Appendix 2.

3.4 JOHNSON AND THOMAS MODEL

Johnson and Thomas proposed a model for maintenance of duplicate data bases using timestamps (9). Requests are timestamped as they enter the job stream, then sent to an assigned primary back-end where the update is performed. Subsequently, the request is added to the primary back-end's modification table.

When the secondary back-end is free, the primary back-end sends its oldest candidate modification to the secondary back-end. The secondary back-end compares the new request with its update list. If the request is the most recent for the record, then it is added to the secondary back-end's update list. If the back-end's modification list has a more recent timestamp, for the record specified, then

Figure 3.2
ELLIS MODEL



the request from the primary back-end is ignored. Finally, if the modification from the primary back-end is more recent than modification request in the secondary back-end's job queue, then the modification is deleted from the job queue in favor of the more recent modification. This last case discussed can result in duplicate data bases being inconsistent.

The Johnson and Thomas algorithm can be viewed as an improvement on the CODASYL algorithm by utilizing available processing time to perform secondary updates to the data bases. The timestamp prevents modification by a request that is older than the most recent modification request.

The algorithm discussed adds an overhead in that an update list must be maintained by each back-end. A disadvantage of this algorithm is that after completion of all tasks in the job stream, all copies of a redundant data base may not be identical.

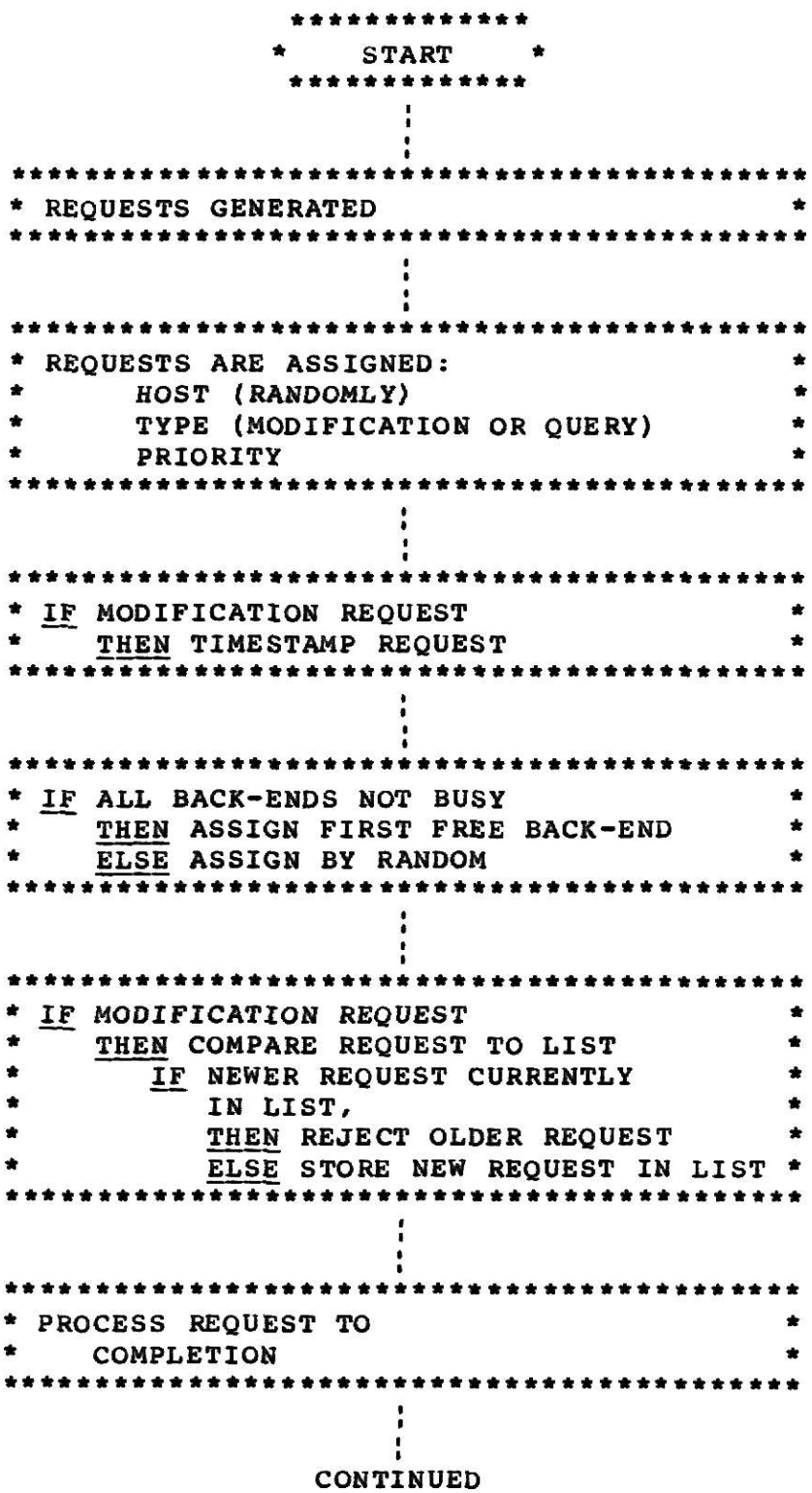
A high level description of the GPSS program of the Johnson and Thomas model is at Figure 3.3. A listing of the program is at Appendix 3.

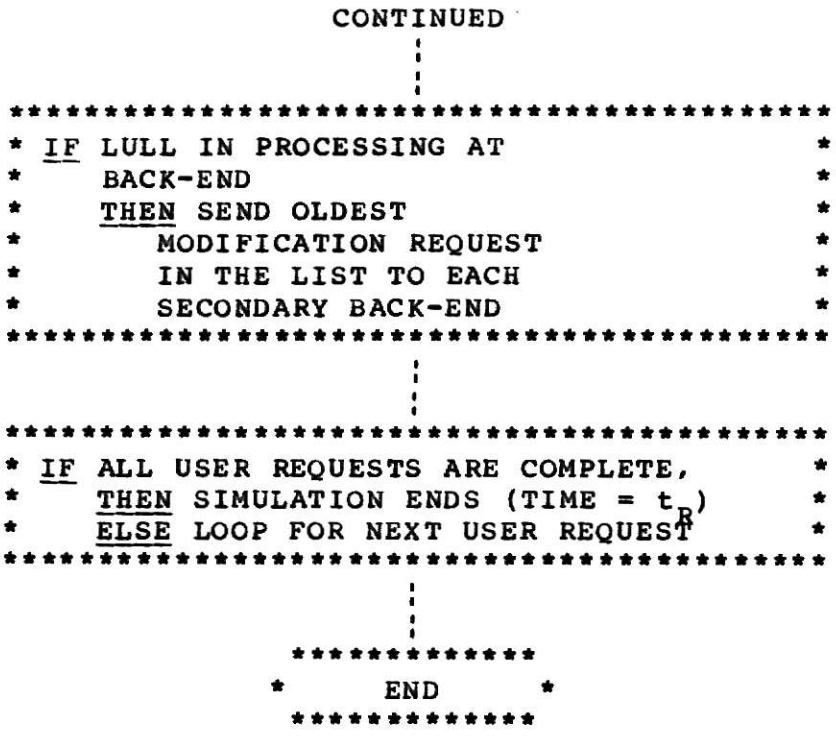
3.5 BERNSTEIN MODEL

Bernstein, et al (3), described a methodology for ensuring mutual consistency of data base copies by use of timestamps and a voting process.

Modification requests are tagged with a timestamp as

Figure 3.3
JOHNSON AND THOMAS MODEL




CONTINUED
IF LULL IN PROCESSING AT BACK-END
THEN SEND OLDEST MODIFICATION REQUEST IN THE LIST TO EACH SECONDARY BACK-END
IF ALL USER REQUESTS ARE COMPLETE,
THEN SIMULATION ENDS (TIME = t_R)
ELSE LOOP FOR NEXT USER REQUEST
END

they enter the job stream, then they are sent to an assigned primary back-end for processing. At the primary back-end all other requests are blocked until the request is processed. Voting takes place wherein each secondary back-end in the data base network must grant approval to update the data base with the new update. Approval is granted only when there are no "older" requests in the secondary's modification list. After all secondary back-ends grant the primary back-end their approval, the modification is performed by the primary back-end and the request is then sent to the secondary back-ends to join their job queues.

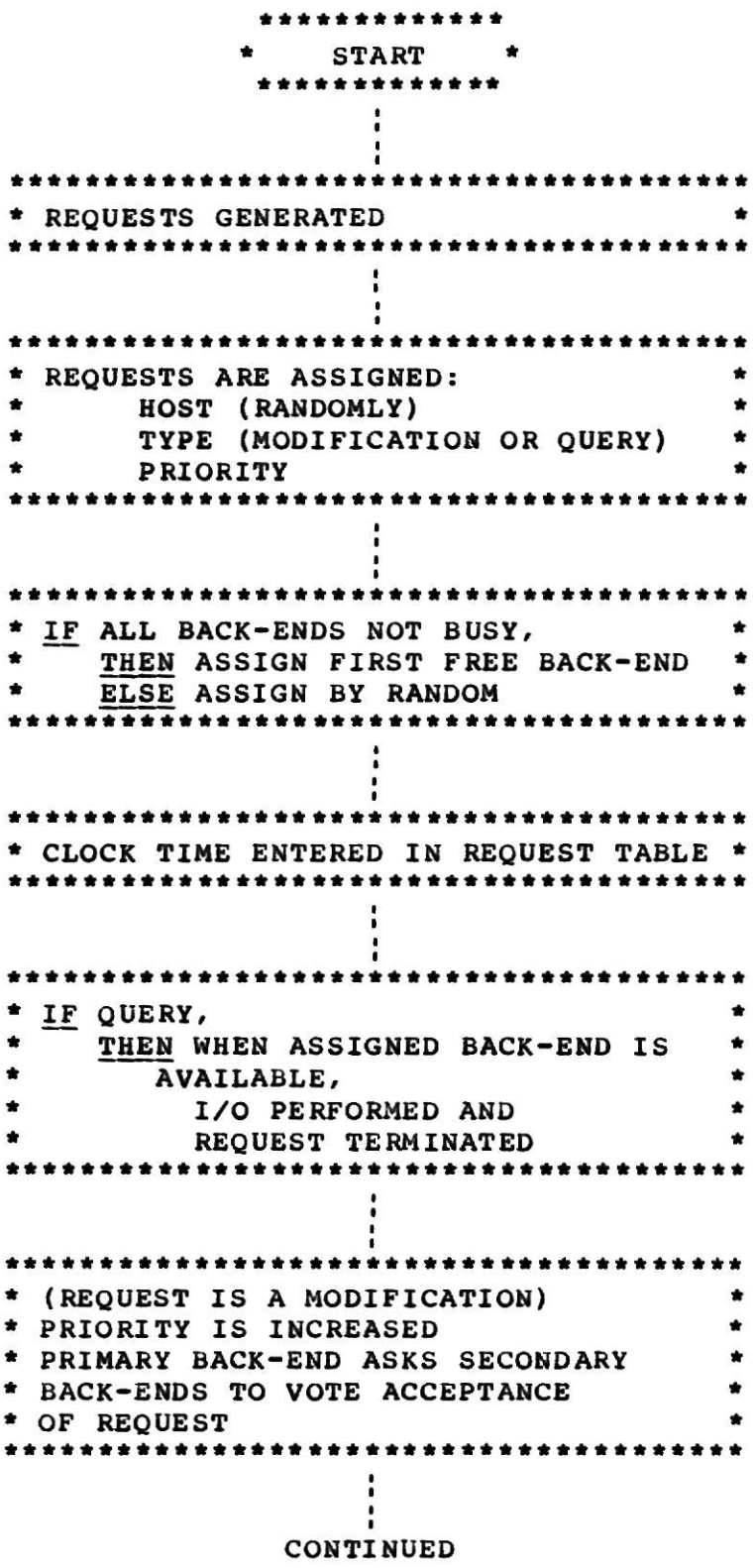
The high level description of the Bernstein algorithm is given in Figure 3.4. A listing of the GPSS program is at Appendix 4.

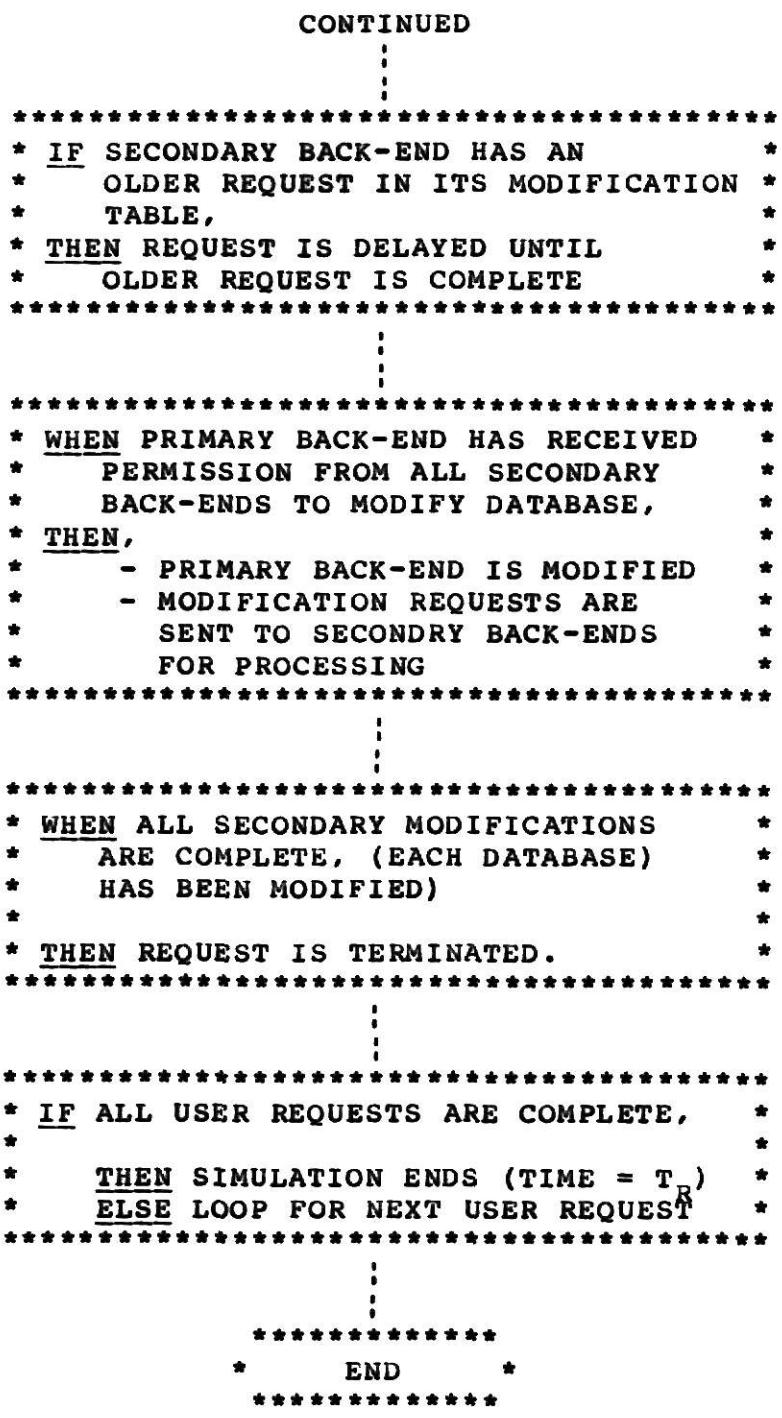
The primary advantage of the Bernstein model over previous methodologies is that mutual consistency of data bases is formally proven, ensuring that each data base converges on the same final state after all tasks have been completed in the job stream. The method requires considerable overhead in that a clock time must be stored for each modified record at each data base.

3.6 HYBRID MODEL

The Hybrid Model is a methodology developed during previous research by Maryanski (10) and Norsworthy (15).

Figure 3.4
BERNSTEIN MODEL




CONTINUED
IF SECONDARY BACK-END HAS AN OLDER REQUEST IN ITS MODIFICATION TABLE,
THEN REQUEST IS DELAYED UNTIL OLDER REQUEST IS COMPLETE
WHEN PRIMARY BACK-END HAS RECEIVED PERMISSION FROM ALL SECONDARY BACK-ENDS TO MODIFY DATABASE,
THEN,
- PRIMARY BACK-END IS MODIFIED
- MODIFICATION REQUESTS ARE SENT TO SECONDRY BACK-ENDS FOR PROCESSING
WHEN ALL SECONDARY MODIFICATIONS ARE COMPLETE, (EACH DATABASE) HAS BEEN MODIFIED)
THEN REQUEST IS TERMINATED.
IF ALL USER REQUESTS ARE COMPLETE,
THEN SIMULATION ENDS (TIME = T_R)
ELSE LOOP FOR NEXT USER REQUEST
END

Similar to the CODASYL model, the Hybrid Model sends all modifications to one primary back-end which logs the request for processing. However, instead of delaying updates until the end of the incoming job stream, as in the CODASYL model, the modification requests are sent to the secondary back-ends whenever the primary back-end detects that it has an empty back-end partition.

The algorithm for the Hybrid model is at Figure 3.5. A listing of the GPSS program is at Appendix 5.

The model is not as complex as the Bernstein model and, unlike the Johnson and Thomas model, ensures consistent copies of the data bases upon completion of processing (15).

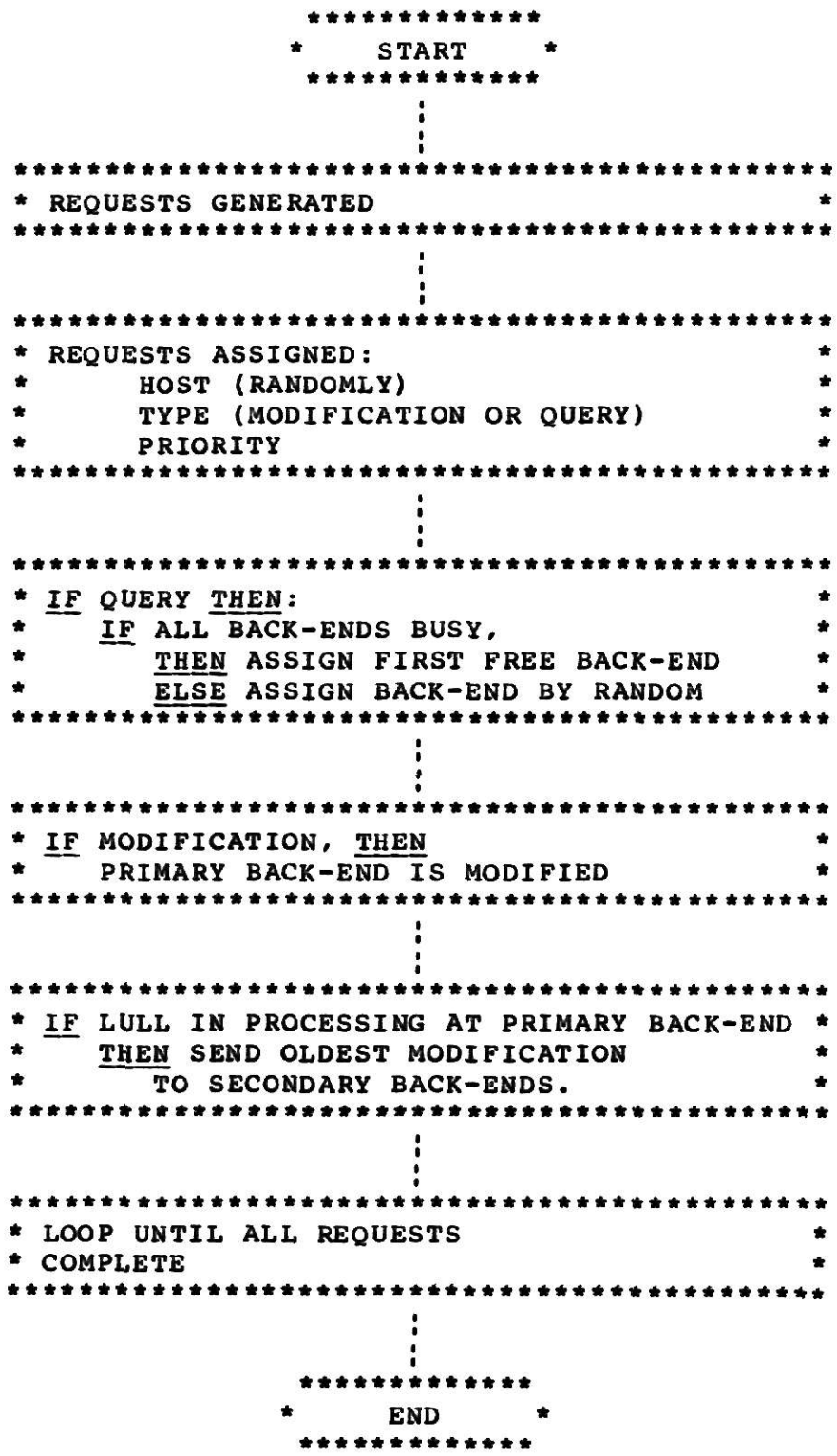
3.7 SUMMARY OF THE SIMULATION MODELS

The five models described in this chapter simulate models of methodologies that have been developed to update redundant data bases consistently.

The simulation models of the algorithms are useful in that a comparison can be made of the methodologies in a controlled environment. The implementation of this is discussed in Chapter 4.

The methodologies described require time to update duplicate data bases consistently, to vote, and/or to check modification tables. The simulation models were designed to test the time required by each of the models to update all data bases. As an overview, techniques used by the five

Figure 3.5
HYBRID MODEL



methodologies which directly affect their response times, are listed below:

	CODASYL	JOHNSON & THOMAS	ELLIS	BERN-STEIN	HYBRID
Use of time-stamps		X		X	
Primary back-end asks permission or vote to update			X	X	
Single modification table at primary back-end	X				
Modification tables at all back-ends		X	X	X	X

Methodologies
Figure 3.6

An analysis of the techniques above are discussed in relation to the results of this study in Chapter 6. Analysis of the time required by the models is used as the dependent variable of this study and the implementation is discussed in Chapter 4.

It is important to note that there are other considerations that must be considered when comparing the five methodologies, primarily the memory requirement of each of the models that is used to store clock times, tables, and record numbers. These additional factors for communications transmission time is another factor that must be considered

when comparing the algorithms. This parameter was not varied in this study of the models and the network configurations were considered to be located at the same installation. When the communication time is varied for dispersed networks, the models can be expected to be affected in various ways.

The additional factors for comparing the methodologies are included in Results and Analysis, Chapter 6.

CHAPTER 4

SIMULATION IMPLEMENTATION

4.1 OVERVIEW

This chapter contains a discussion of the simulation conducted using GPSS V to obtain response time, t_R, data for multi-host, multi back-end data base systems. A simulation was conducted for each of the five methodologies discussed in Chapter 3, for each combination of 2 and 4 host CPUs and 2, 4, and 8 back-end processors in the data base network, keeping a constant workload per host CPU. The percent of modification and high priority requests were kept constant at 50 percent and 10 percent, respectively. Some additional simulations were made to test the effect of the workload and priority requests.

The models simulated resulted in a large GPSS core memory requirement and the memory allocated by the GPSS system required careful management to ensure GPSS core memory was not exceeded and to reduce the computing cost of the simulations.

Data obtained by this study and data used from earlier studies are listed in Appendices IX and X.

4.2 SIMULATION PROGRAMS

The simulation parameters are defined in Chapter 2. Appendices 1-5 contain listings of the GPSS simulation

programs used.

In all but one case, each model simulated was run six times--each time varying the RMULTs, thereby changing the internal random number generators.

The following two sets of parameters were each varied for 2, 4, and 8 back-end processors:

2 HOSTs

f = 3/4 sec/request

M = 50 (50% modifications)

P = 10 (10% priority requests)

4 HOSTs

f = 3/8 sec/request

M = 50

P = 10

Consequently, mean response times, $\bar{t}_R$, were obtained for six multi-host environments of each methodology (except CODASYL). The 4 host, 8 back-end configuration of the CODASYL model was not simulated due to the large requirement for memory, the long execution time, and the expected expense. However, sufficient data was obtained with which to compare the CODASYL model with the other methodologies discussed herein (see discussion of results in Chapter 6).

Additional data was also obtained for single-host networks of the Bernstein model. The following set of parameters was varied for 2, 4, and 8 back-end processors:

1 HOST

f = 3/2 sec/request

M = 50

P = 10

The above simulations were conducted to ensure valid single-host data in the analysis due to several changes in the Norsworthy version of the Bernstein simulation that were required for a multi-host environment.

Additional simulations were conducted with the purpose of testing the effects of several independent variables on the model's response time. The following simulations were conducted to test (1) the effect of the work load per host and (2) the affect of priority requests included in the job stream.

H = 1 ; B = 2 ; f = 3/4 sec/req ; R = 30 ; M = 50 ; P = 10

H = 4 ; B = 2 ; f = 3/4 sec/req ; R = 30 ; M = 50 ; P = 10

H = 1 ; B = 2 ; f = 3/2 sec/req ; R = 30 ; M = 50 ; P = 0

Analysis of the data obtained from the additional simulations is given in Chapters 5 and 6.

An explanation of the GPSS code for each of the five basic simulation models is provided in the Norsworthy Master's Report (15) and it is considered unnecessary to repeat his description of the simulation models in this Master's Report.

The multi-host network models developed by this author are implemented as described in Chapter 3. The network model for the environments is as shown in Figure 4.1. Upon entering the network, requests are randomly assigned a host machine and continue to be processed by that host until

USERS
8 TERMINALS
HOST 1
OPER SYS
USER PROGRAMS
HOST INTERFACE
COMMUNICATIONS
HOST 2
OPER SYS
USER PROGRAMS
HOST INTERFACE
COMMUNICATIONS
HOST 3
OPER SYS
USER PROGRAMS
HOST INTERFACE
COMM SYS
HOST 4
OPER SYS
USER PROGRAMS
HOST INTERFACE
COMM SYS
INTER BACKEND COMMUNICATION SYSTEM
BACKEND 1
BACKEND 2
BACKEND3
BACKEND4
BACKEND 5
BACKEND 6
BACKEND 7
BACKEND 8
DBMS
TASK 1 TASK
1 . . . K
buffer
DBASE DATA 1
DBASE DATA 2
DBASE DATA 3
DBASE DATA 4
DBASE DATA 5
DBASE DATA 6
DBASE DATA 7
DBASE DATA 8

FIGURE 4.1

processing is complete by that request. This selection method was used for all models.

With the exception of two additional simulation models for the Bernstein algorithm (discussed above), the work load per host machine was held constant throughout the simulations experiments in order to compare the effects of adding multiple host CPUs to the distributed data base network.

4.3 LARGE GPSS CORE MEMORY REQUIREMENT

Addition of multiple hosts and the consequent increased rate of requests to the network resulted in heavy core memory requirements of the GPSS V system. Further, execution times were frequently over two minutes in duration for a single simulation program. This effect required careful reallocation of memory from the standard allocation provided by GPSS V. Auxilliary storage of modification tables was also used as a memory allocation technique, however, it became impractical when many accesses were required. Reallocation of memory is described in reference (8) pages 326-329 and in reference (7). It was found that all multi-host models could be simulated using parameter B in the GPSS control statement that loads the GPSS program, excepting the CODASYL model which required parameter C.

4.4 DATA

Reponse times, t_R, for each simulation are listed in Appendix 8. Mean response times and their standard deviations of the simulations conducted by this study are listed in Appendix 9.

Data for single-host data base network systems obtained by Norsworthy and used for analysis in this report is listed in Appendix 10.

4.5 SUMMARY OF IMPLEMENTATION

The Data Base network models with multiple host CPUs were simulated varying the number of host CPUs and back-end processors for constant work loads. By making multiple simulations and varying the random numbers generated by GPSS, the response times were obtained for each of the methodologies. Response time was the time taken by the algorithm to answer the user's requests and for the duplicate data bases to reach a steady state, having consistent data bases. The number of host CPUs and back-end processors significantly affected the response times, $\bar{t}_R$. Also, the frequency of requests to the data base and the percentage of modifications and high priority requests affected $\bar{t}_R$.

The data obtained from this study is discussed and analyzed in Chapter 5, Mathematical Models, and in Chapter 6, Results and Analysis.

CHAPTER 5

MATHEMATICAL MODELS

5.1 OVERVIEW

The data obtained from the simulations of multi-host data base systems were combined with the data obtained from previous single-host data base networks. Multiple linear regression procedures of Statistical Analysis System (SAS), reference (2), were used to obtain mathematical models of the five simulated metodologies in this study. This chapter describes the procedures used to select the mathematical models and discusses their ability and limitations to describe the algorithms.

5.2 DATA AND VARIABLES

Preliminary analysis of the data obtained in this study (Appendix 8) indicated that a relationship existed between the dependent variable, t_R, and the following independent variables:

B - number of back-end processors

H - number of host machines

f - frequency of requests

R - number of requests in the job stream

The Correlation and RSQUARE procedures of SAS were used to determine that relationships existed between the independent variables of the simulation. The General Linear

Model procedure of SAS was used to determine mathematical relationships between the independent variable, t_R, and B, the number of back-ends. Also, mathematical models were found with t_R, and H. However, a model could not be determined with a high correlation with variables t_R, B, H, R, and f, until additional simulations were used to test the variables (discussed in next section).

5.3 MULTIPLE LINEAR REGRESSION

In order to test the relationship of the work load the frequency of requests to the dependent variable t_R and the other independent variables, two additional simulations were made of the Bernstein model, with the following parameters:

(1) H = 1 ; B = 2 ; f = 3/4 ; R = 30

(2) H = 4 ; B = 2 ; f = 3/4 ; R = 30

The above simulations differed from all previous simulations in that the requests per host was not equal 15. Subsequently, the GLM and Stepwise procedures of SAS were used to develop one mathematical model for each of the methodologies.

It was decided to pull all available data of the five models together to, first, test the models that were developed and, second, to strengthen the formulas if a general relationship existed. A problem with this approach was that the previous data obtained by Norsworthy (15) varied percent modifications (20, 35, and 50 percent). Also

the Norsworthy data did not include a 10 percent priority request rate. Modification and priority were added as independent variables in the linear regression. Using the SAS stepwise procedure, a strong correlation was found between four of the five models when a multiple regression was made using one general model. The fifth model, CODASYL, was found to fit a separate mathematical model. Instead of each methodology being represented by five separate mathematical models, only two are required. The two models and the coefficients for each of the models are listed in Tables 5.1 and 5.2.

The coefficient of determination, R^2, for the five simulation models ranged between 0.993690 and 0.999655. The SAS program giving the models' R^2, coefficient of the general mathematical model, and the probability of fit is listed at Appendix 6.

5.4 LIMITS OF THE MATHEMATICAL MODELS

As stated in Chapter 1, the purpose of this study is to determine the effect of multiple-hosts on the response time, t_R, for the five update algorithms described herein. The mathematical models described in this chapter were developed, in large part, after the simulation data had been collected. As a result, insufficient data was collected to fully test each of the independent variables of the mathematical models. For example, in the Bernstein model

TABLE OF MATHEMATICAL MODELS I

Models for BERNSTEIN, ELLIS, HYBRID, and JOHNSON AND THOMAS Methodologies

$$\bar{t}_R = C1 + C2*B + C3*R*f^{-1} + C4*B^{-1} + C5*H^{-2} + C6*f*B^{-1} + C7*B*f + C8*M/(B*f) + C9*P/(B*f)$$

Where:

$\bar{t}_R$ mean response time in seconds

B number of back-ends
H number of host CPUs
R total number of requests
f request interval(sec/request)
M percent modification requests
P percent high priority requests

	BERNSTEIN	ELLIS	JOHNSON & THOMAS	HYBRID
C1	79.13563680	145.05242827	55.51611866	-11.35832609
C2	9.003336688	-6.43088333	6.18675417	17.55630833
C3	0.33138173	0.75011348	0.47046118	0.33556617
C4	-51.74670317	-168.45220745	16.21106783	173.31042058
C5	4.96598796	-79.16173356	0.35511998	82.77174692
C6	-5.94198065	107.21754881	-2.36432766	-135.01332697
C7	-6.86593821	4.06155556	-4.30727778	-12.55277778
C8	1.42773218	0.47112568	-0.32782457	0.89594651
C9	1.33701028	0.72173694	6.09821369	2.80743509

Table 5.1

TABLE OF MATHEMATICAL MODELS II

Model for CODASYL Methodology

$$\bar{t}_R = C1 + C2*H*f^{-1} + C3*B^2 + C4*f*B + C5*B*P*f^{-1} + C6*B*M*f^{-1}$$

Where:

$\bar{t}_R$ mean response time in seconds

B number of back-ends
H number of host CPUs
R total number of requests
f request interval (sec/request)
M percent *modification requests*
P percent high priority requests

	CODASYL
C1	89.2074747299
C2	1.31191234
C3	0.98374036
C4	-6.69186434
C5	0.39499193
C6	-0.03016417

Table 5.2

all simulations were made with M = 50 and in all but one simulation P = 10. Further, in almost every case the work load per host was a constant 15 requests per host.

The benefits of the capability for expressing the results of the simulation in mathematical terms will become evident in later chapters. One of the primary benefits is that the relationship between the independent variables, such as H and B, can be examined graphically in relationship to the dependent variable t_R. It will be shown that there are optimum numbers of both back-end processors and numbers of hosts, depending on the work load of the system.

The analysis of the results of this study in Chapter 6 considers the adequacy of the data to substantiate the mathemtical models. Care is taken not to extend the mathematical models beyond their limits.

CHAPTER 6

RESULTS AND ANALYSIS

6.1 OVERVIEW

The effects of the experimental parameters on the mean lag time, $\bar{t}_d$, are examined for each of the five update methodologies. The effect of adding additional back-ends, while keeping the number of host machines constant, is analyzed graphically for each methodology. The results are presented with graphical plots using a SAS program. The impact of varying the number of hosts for each model and the effect of keeping the workload per back-end constant are examined. Finally, the optimum number of back-ends for each model and number of hosts is plotted graphically, keeping the workload per host constant. A summary of the plots is given in Sections 6.6. A discussion of the model's queueing behavior in the GPSS simulations is given in Section 6.7. An overall summary of the analysis in Section 6.8.

6.2 MODELS COMPARED, VARYING NUMBER OF BACK-ENDS; CONSTANT WORKLOAD PER HOST

Using the mathematical models described in Chapter 5, $\bar{t}_d$ is plotted varying the number of back-ends, holding constant: H, f, M, and P. Table 6.1 tabulates the figures, the values of the experimental parameters, and the symbols of the algorithms that are represented in the plots.

	FIGURE NUMBERS									
	6.1	6.2	6.3	6.4	6.5	6.6	6.7	6.8	6.9	6.10
ALGORITHM Bernstein		B						B		B
ALGORITHM CODASYL	C		C		C		C		C	
ALGORITHM Ellis		E		E		E		E		E
ALGORITHM Hybrid		H		H		H		H		H
ALGORITHM Johnson and Thomas		J		J		J		J		J
Hosts	1	1	1	1	1	1	2	2	4	4
f (sec/ request)	3/2	3/2	3/2	3/2	3/2	3/2	3/4	3/4	3/8	3/8
M (percent)	50	50	35	35	20	20	50	50	50	50
P (percent)	10	10	10	10	10	10	10	10	10	10

Plot Parameters
Table 6.1

The points of the plots correspond to the models as follows:

B Bernstein Model

C CODASYL Model

E Ellis Model

H Hybrid Model

J Johnson and Thomas Model

The plots of the graphs were made with a Procedure PLOT of SAS. An example of the SAS PLOT program procedure used in

this section to provide the graphical relationships of the plots is at Appendix 7. For information on use of the PLOT module see reference (18).

Each plot is discussed in a subsequent subsection. Note that the percent of modification requests were plotted for one host only. This is due to the fact that M was constant at 50 percent for simulations of multiple hosts. Accurate effects of variable M is believed to be represented by the single host network. Note also that the percent of high priority requests is never compared with the various models. This is due to the face that data was collected for P = 10 for multiple hosts and for P = 0 for single host networks, and data was never obtained with all variables constant except priority.

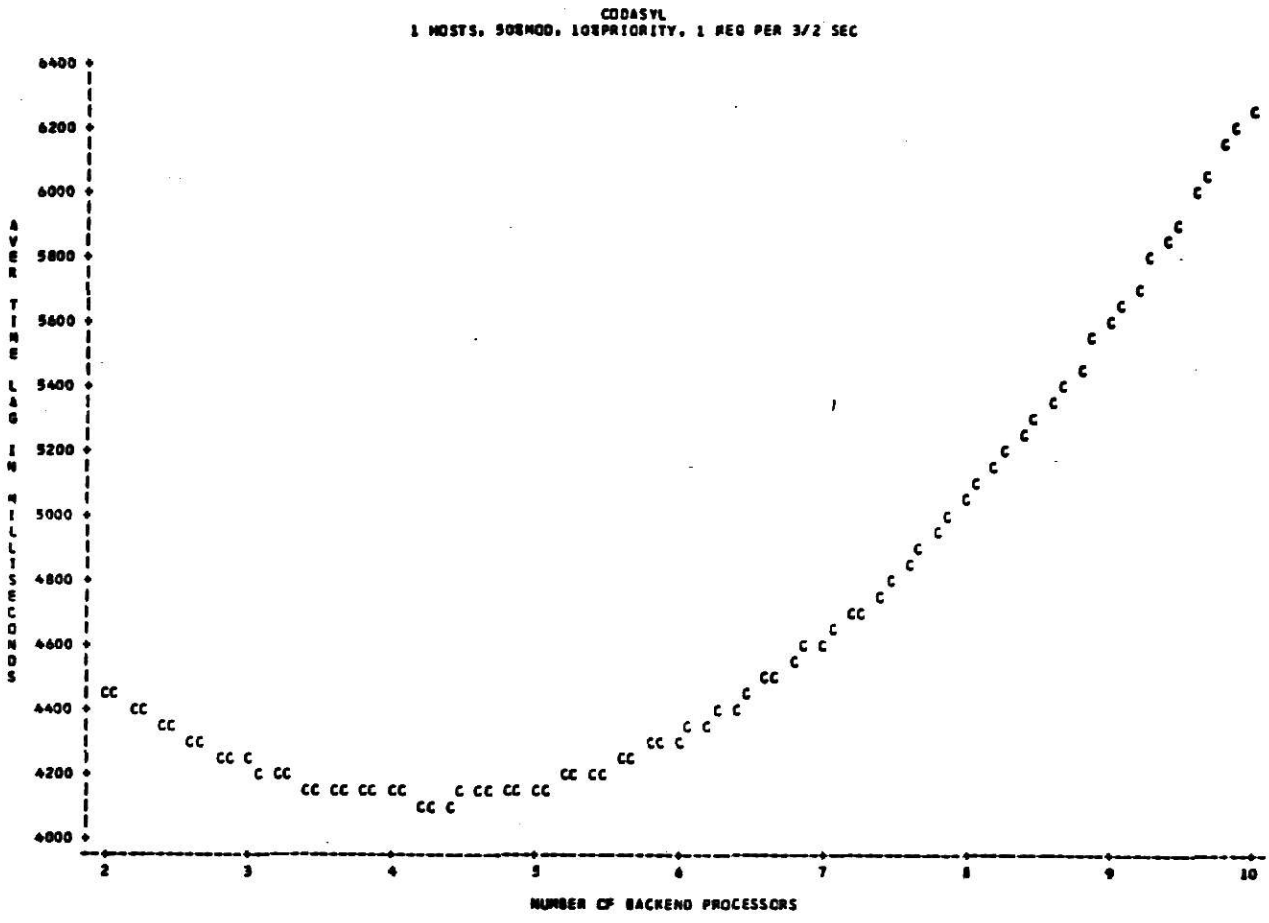


Figure 6.1

6.2.1 CODASYL MODEL, 1 HOST, 50% MODIFICATIONS

The CODASYL model is plotted in Figure 6.1 showing the mean time lag, $\bar{t}_d$, as the number of back-end is varied. Values of the experimental parameters are as shown in the legend of the plot. The CODASYL model performs with a higher $\bar{t}_d$ than the other four models studied (see next section). In order to show more graphically the differences of the other models, the CODASYL model will normally be shown separately. As indicated in the plot above, $\bar{t}_d$ decreases as the number of back-ends increase until B = 5, then $\bar{t}_d$ increases with each

addition of a back-end processor. As discussed in Section 3.3, the CODASYL algorithm updates only the primary back-end until all requests have been received. Adding back-ends results in a proportional increase in workload for the primary back-end.

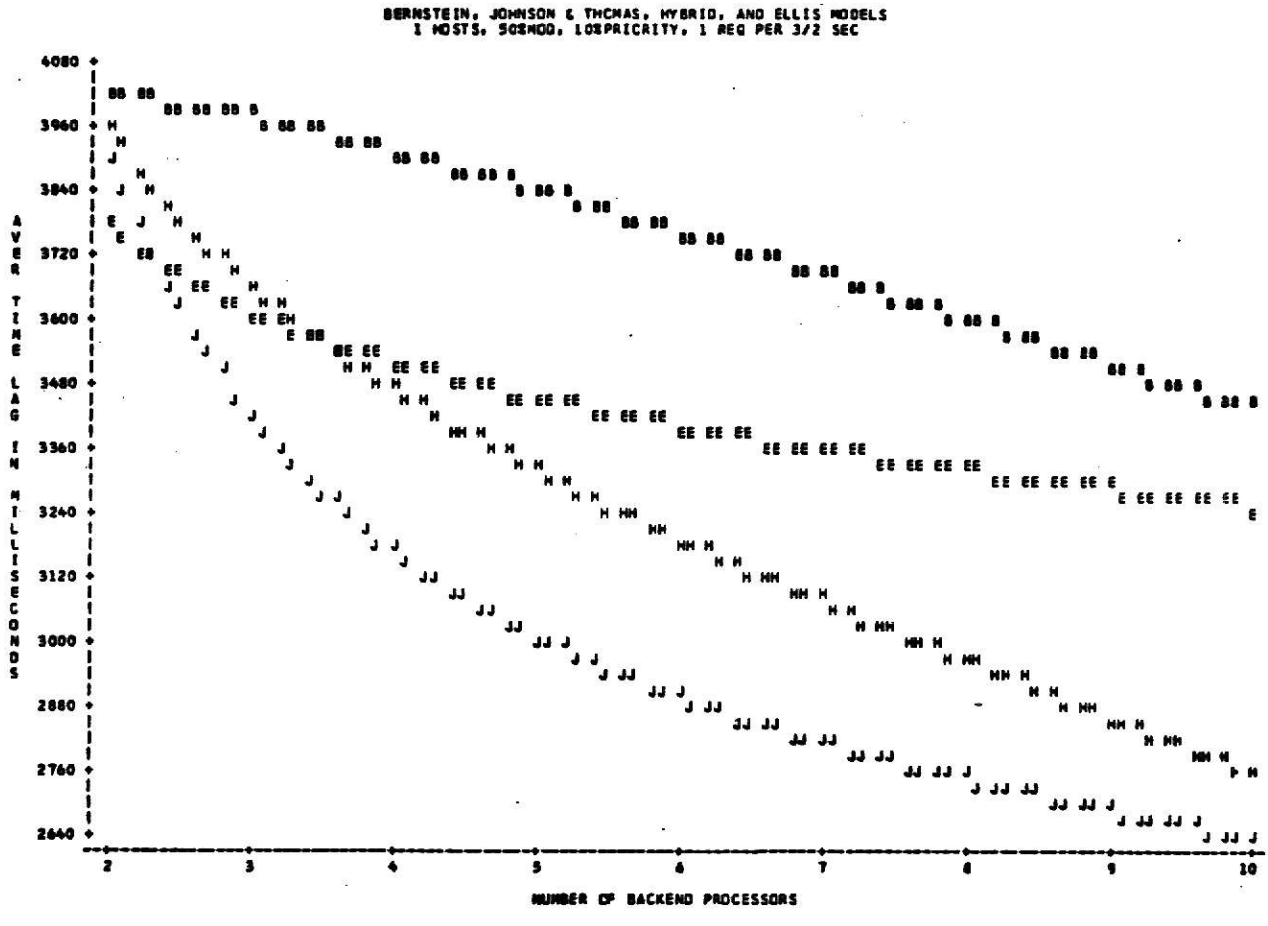


Figure 6.2

6.2.2 MULTIPLE MODELS, 1 HOST, 50% MODIFICATIONS

For the four models plotted in the figure above, the workload in the network is relatively light and as the number of back-ends increase, the mean system time lay, $\bar{t}_d$, steadily decreases for each of the methodologies. The Johnson and Thomas model performs with the least delay in updating the data bases in the plot.

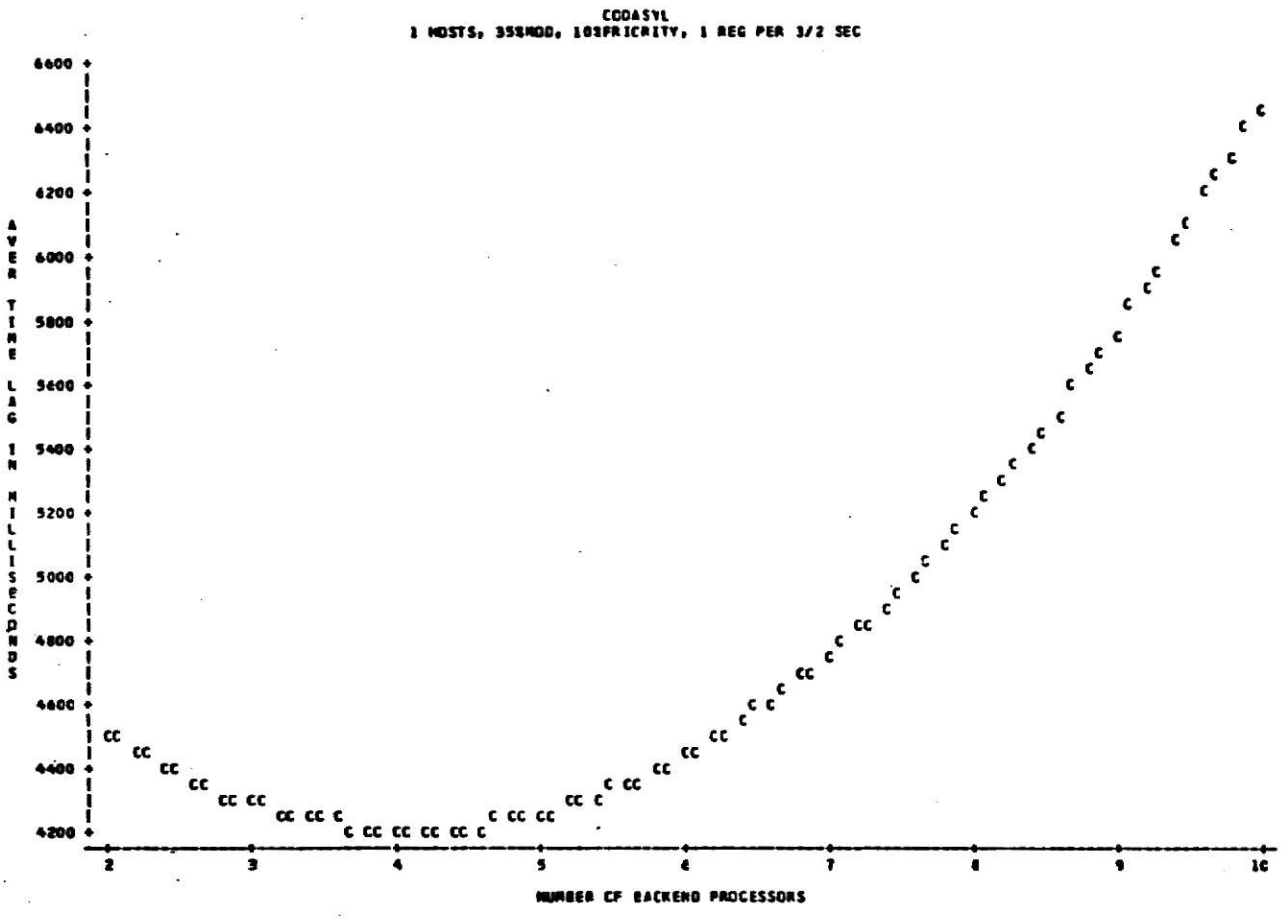


Figure 6.3

6.2.3 CODASYL MODEL, 1 HOST, 35% MODIFICATIONS

The CODASYL model responds only slightly better with a decrease in modifications from 50 percent to 35 percent (compare with Figure 6.1). The $\bar{t}_d$ decreased 200 milliseconds with respect to Figure 6.1 after the number of back-ends increased to B > 4.

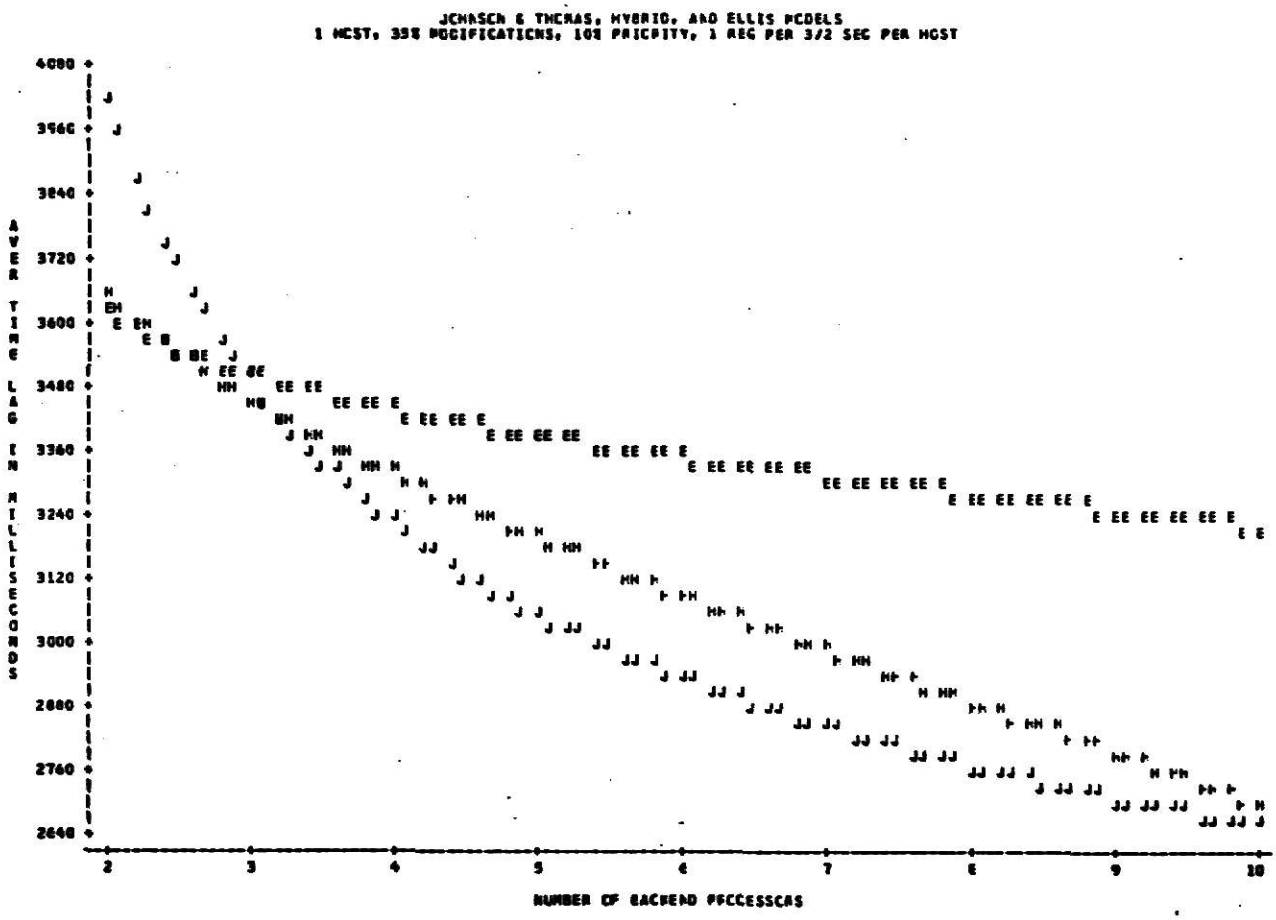


Figure 6.4

6.2.4 MULTIPLE MODELS, 1 HOST, 35% MODIFICATIONS

The plot above differs from Figure 6.2 by only a reduction in percent modifications (M = 35). (The Bernstein model is not included beacause simulation data was obtained only for M = 50 for the Bernstein model). The mean time lags, t_d, are similar to Figure 6.2. As B increases, $\bar{t}_d$ decreases for each of the models plotted. Performance improved as back-ends were added because the total system was able to progressively process requests more efficiently through concurrent operations.

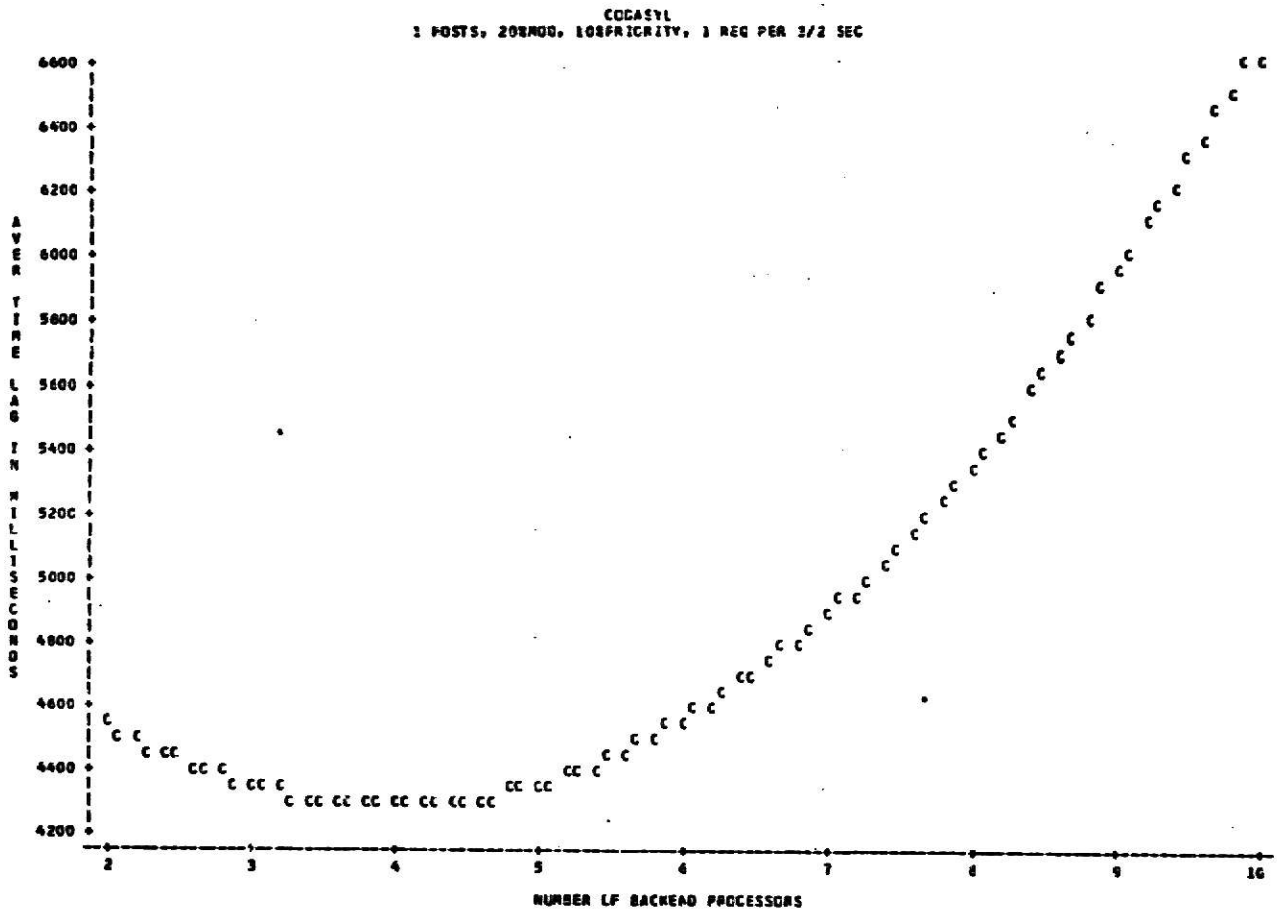


Figure 6.5

6.2.5 CODASYL MODEL, 1 HOST, 20% MODIFICATIONS

A decrease in percent modifications from 35 percent to 20 percent had virtually no effect on the average time lag for the CODASYL model (compare above plot with Figure 6.3).

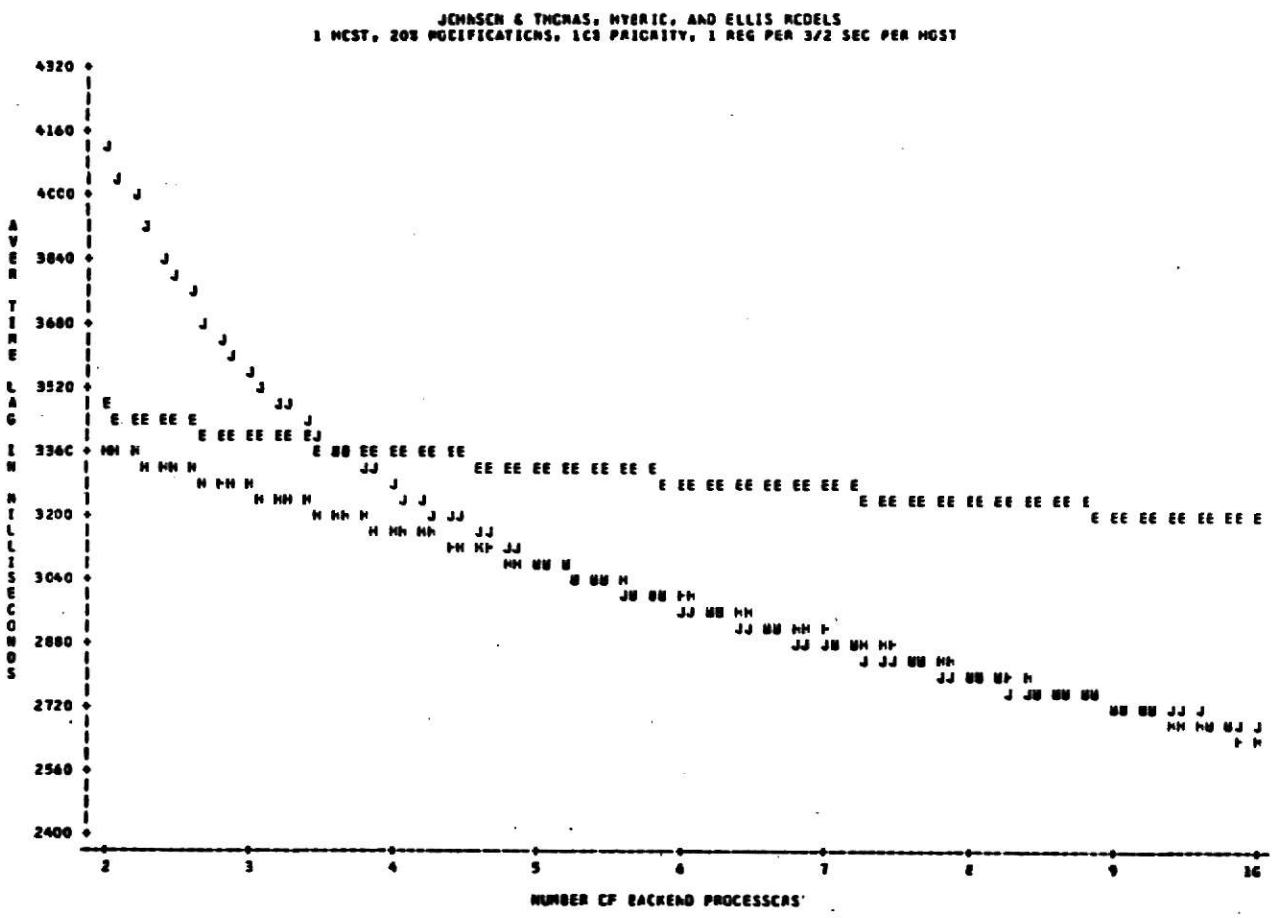


Figure 6.6

6.2.6 MULTIPLE MODELS, 1 HOST, 20% MODIFICATIONS $\bar{t}_d$ decreases slightly as the percent modification decreases from 35 percent to 20 percent (compare Figure 6.6 to above figure). Note that the $\bar{t}_d$ for the Johnson and Thomas model and the Hybrid model became similar as the percent modifications decreased.

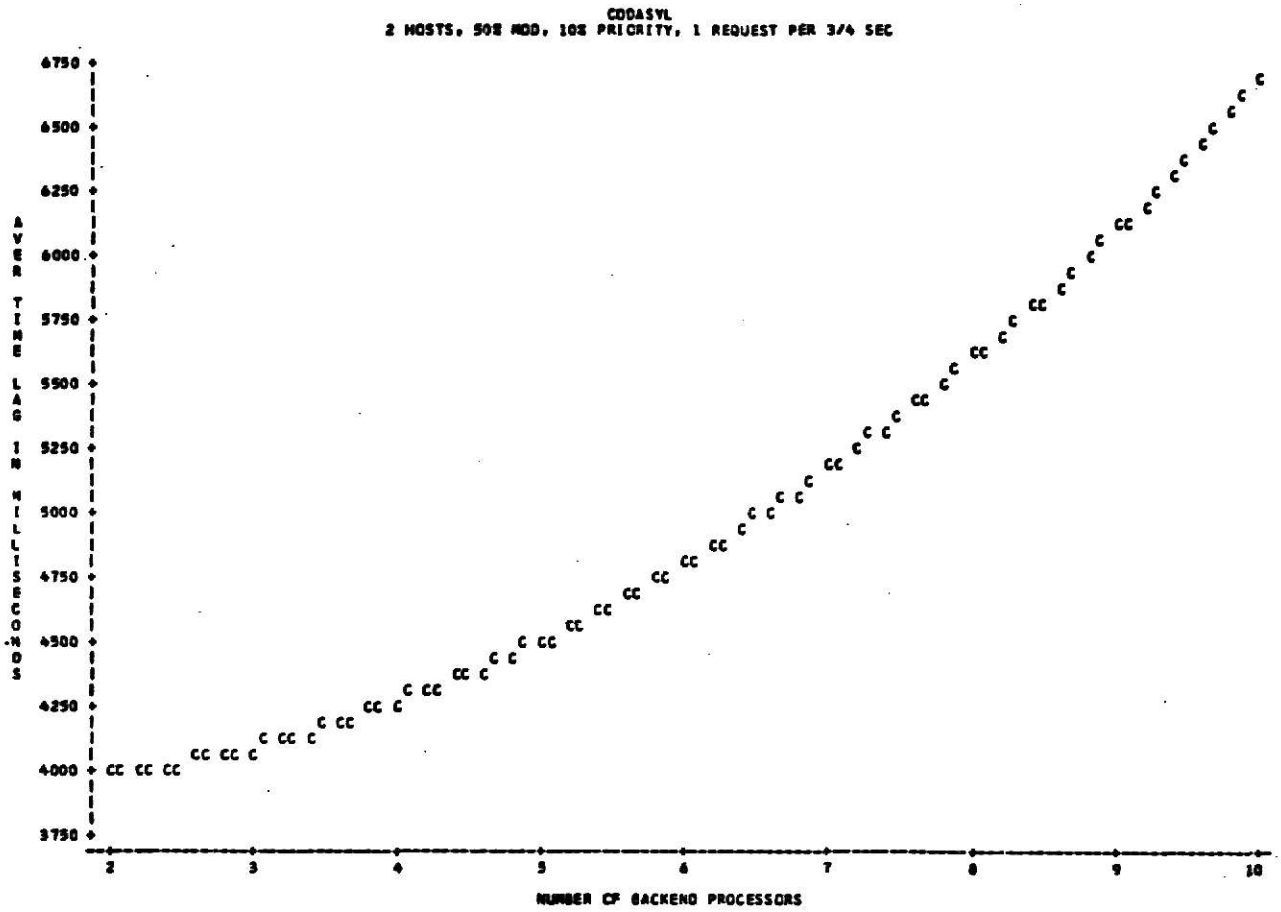


Figure 6.7

6.2.7 CODASYL MODEL, 2 HOSTS

The above plot differs from that in Figure 6.1 by the number of host machines and the frequency of requests. The total requests per host and frequency of requests per host (time interval between requests is 3/2 divided by H) remains constant in both environments; however, the workload per back-end has increased. In essence the network's workload is doubled, keeping the same number of back-ends and all other parameters resulted in an increase in $\bar{t}_d$. Note that there is no efficiency gained by adding back-ends at any point in the plot at the additional workload. This result is expected

because as the workload increases in the network the time required to update all data bases will progressively increase. The CODASYL model has only one primary back-end and additional back-ends results in added workload for the primary back-end.

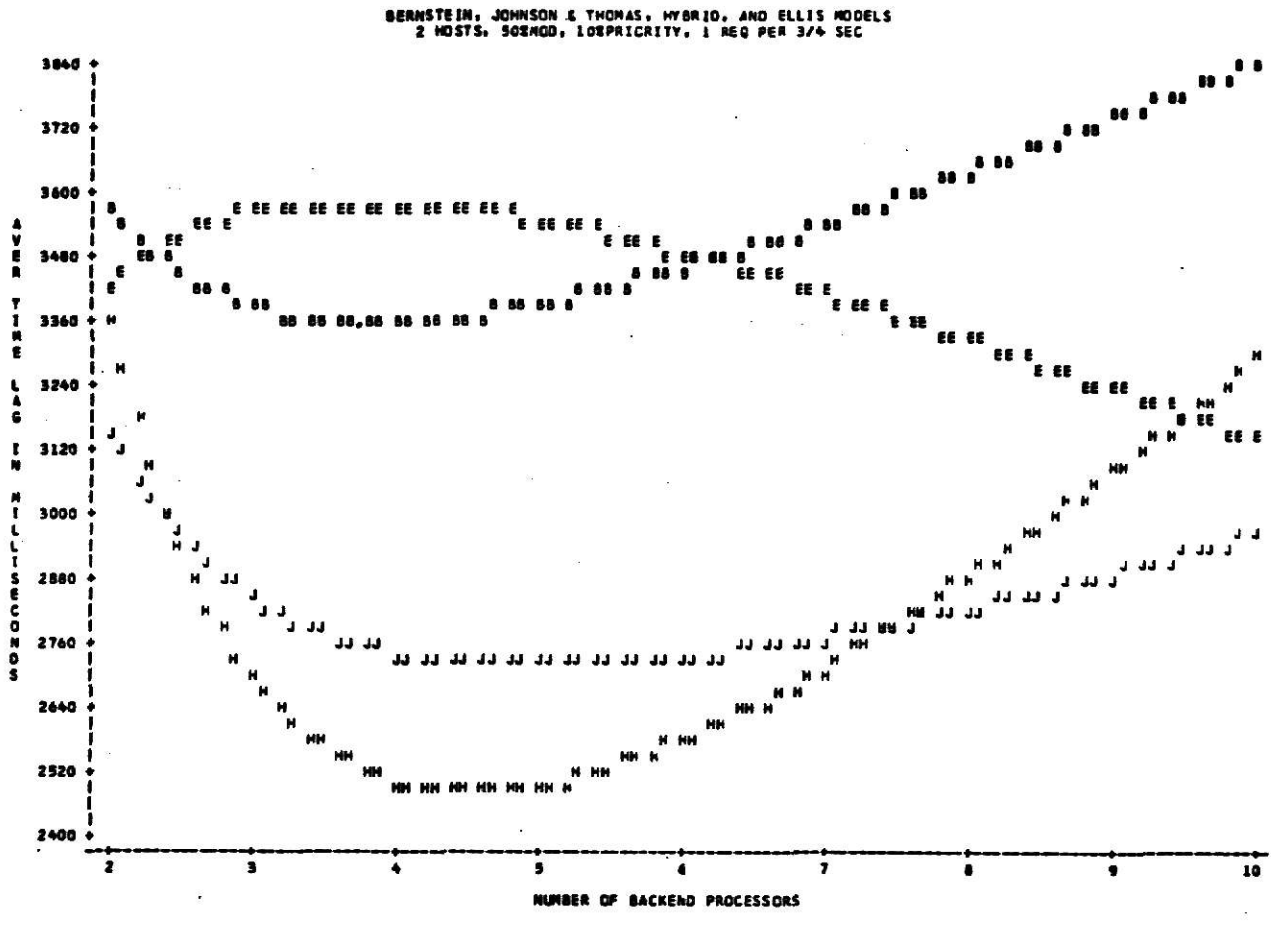


Figure 6.8

6.2.8 MULTIPLE MODELS, 2 HOSTS

Compare the above plot to Figure 6.2. The number of hosts is doubled and the workload per host is constant. An interesting transformation in the curves of the models has taken place. As B approaches 10, $\bar{t}_d$ is between 0 and 10 percent greater (and slower) than with H = 1; however, for B = 8, $\bar{t}_d$ is significantly reduced (at B = 5 there is over a 20 percent reduction for the Bernstein, Johnson and Thomas, and the Hybrid models). The Ellis model is not dramatically affected; however, at B = 2 Ellis responds much faster with

H = 2. The Ellis model does not react in the same manner as the other models. The Ellis model is much slower than the other models for B $\leq$ 5; however, it is less affected by adding additional back-ends. Another significant effect is noticeable for the Bernstein, Hybrid, and Johnson and Thomas models: instead of $\bar{t}_d$ steadily decreasing as B increases, the algorithms reach an optimum $\bar{t}_d$ at approximately B = 5, then $\bar{t}_d$ increases in lag time. The overhead of the algorithms, such as voting and use of modification tables begin to become a significant factor and the benefits of adding a second host are lost as B > 8.

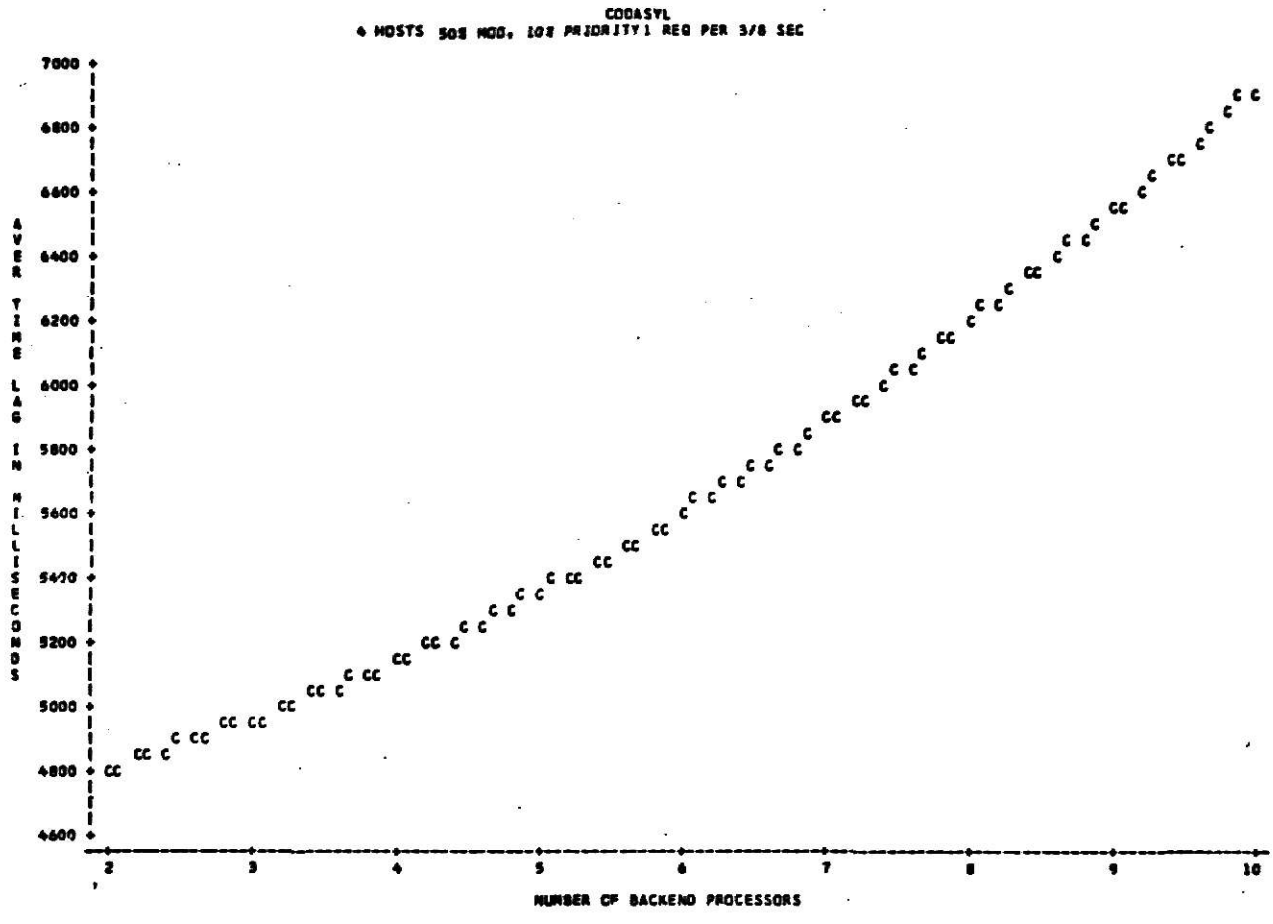


Figure 6.9

6.2.9 CODASYL MODEL, 4 HOSTS

Compare the above plot to Figure 6.7. The addition of back-end processors in a 4 host network increases the lag time, $\bar{t}_d$, in a direct relationship. Throughout the range of B, $\bar{t}_d$ is greater than in a 2 host network. Increasing the number of back-ends in the network increases $\bar{t}_d$ throughout the range of B.

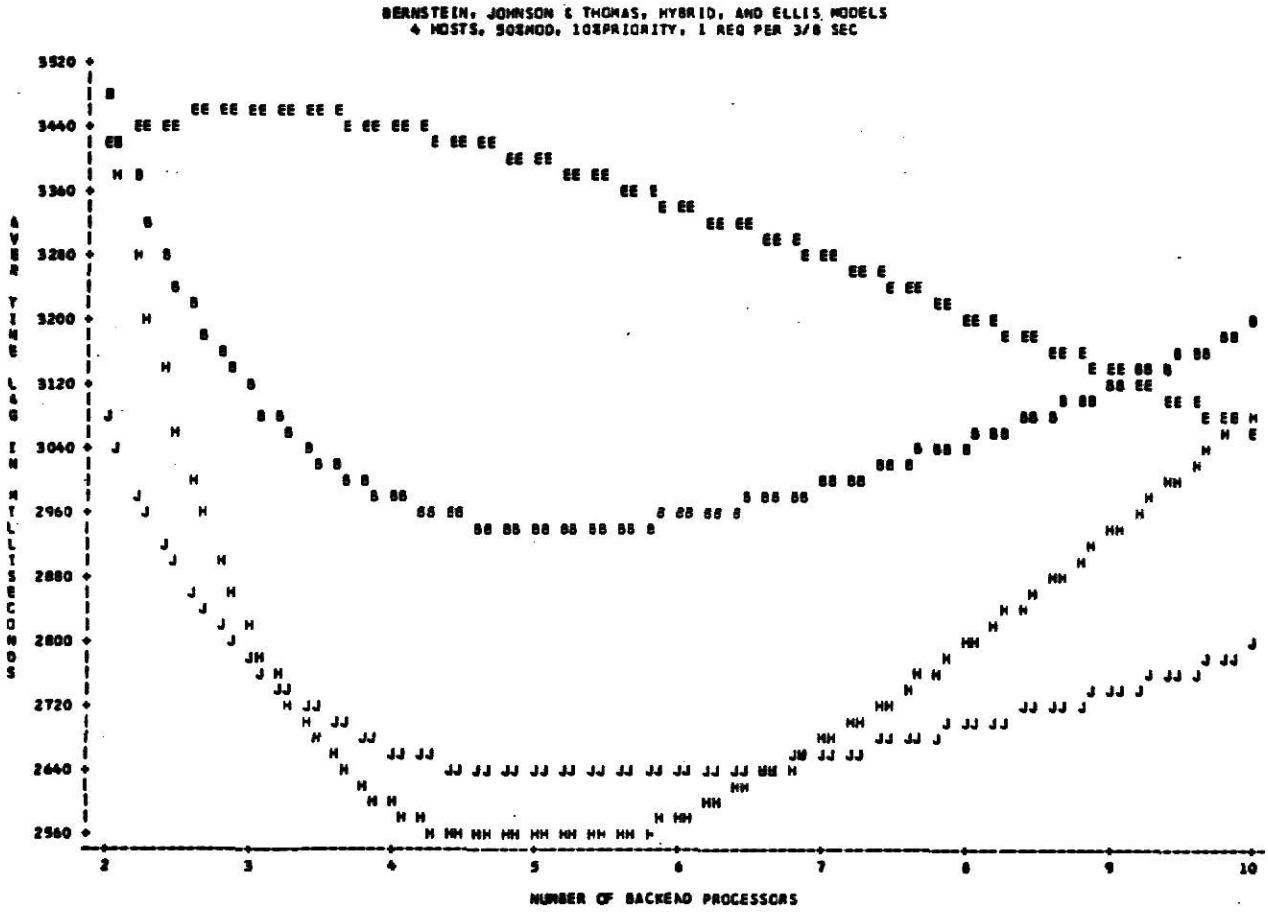


Figure 6.10

6.2.10 MULTIPLE MODELS, 4 HOSTS

Compare the above plot in Figure 6.8. Except for the Bernstein model, an increase in the number of host machines to four, does not significantly affect $\bar{t}_d$. As the number of back-ends increase, the performance of the Bernstein model improves significantly (approximately 20 percent better in performance). For the Bernstein model with four hosts and five back-ends, the optimum $\bar{t}_d$ is plotted when comparing Figure 6.2, 6.8, and 6.10. This observation is again noted in the discussion in Section 6.4 where the $\bar{t}_d$ of the optimum

number of back-ends is plotted for configurations of host machines.

6.3 EFFECT OF NUMBER OF HOSTS ON PERFORMANCE, VARYING BACK-ENDS, AND CONSTANT WORKLOAD

The plots presented in Section 6.2 were discussed for the purpose of comparing each model for specific combination of back-ends and hosts keeping the workload per host constant. It was difficult to compare how a specific model changed as the number of hosts increased because the scale of $\bar{t}_d$ on the vertical axis would vary with each plot. In this section an analysis of the effect of the number of hosts is made for each of the models. Four plots illustrate the effects for four of the models, as follows:

Bernstein	Figure 6.11
Ellis	Figure 6.12
Hybrid	Figure 6.14

where, H = 1, 2, and 4

f = 3/2, 3/4, and 3/8 (respectively)

M = 50

P = 10

The CODASYL model is not plotted because the single plots given in Section 6.2 provide an adequate base for comparison.

6.3.1 CODASYL MODEL

(See Figures 6.1, 6.3, 6.5, 6.7, and 6.9)

When the total number of requests are relatively low for the CODASYL model, as in Figures 6.1.6.2, and 6.5, increases in the number of back-ends have the effect of improving performance. However, the overhead of the additional work

per back-end becomes significant at B = 6 and addtional back-ends result in increases time delay. An effect that does not hold for the other models but does hold for the CODASYL model is that doubling the workload and doubling the number of hosts results in greater average time lag.

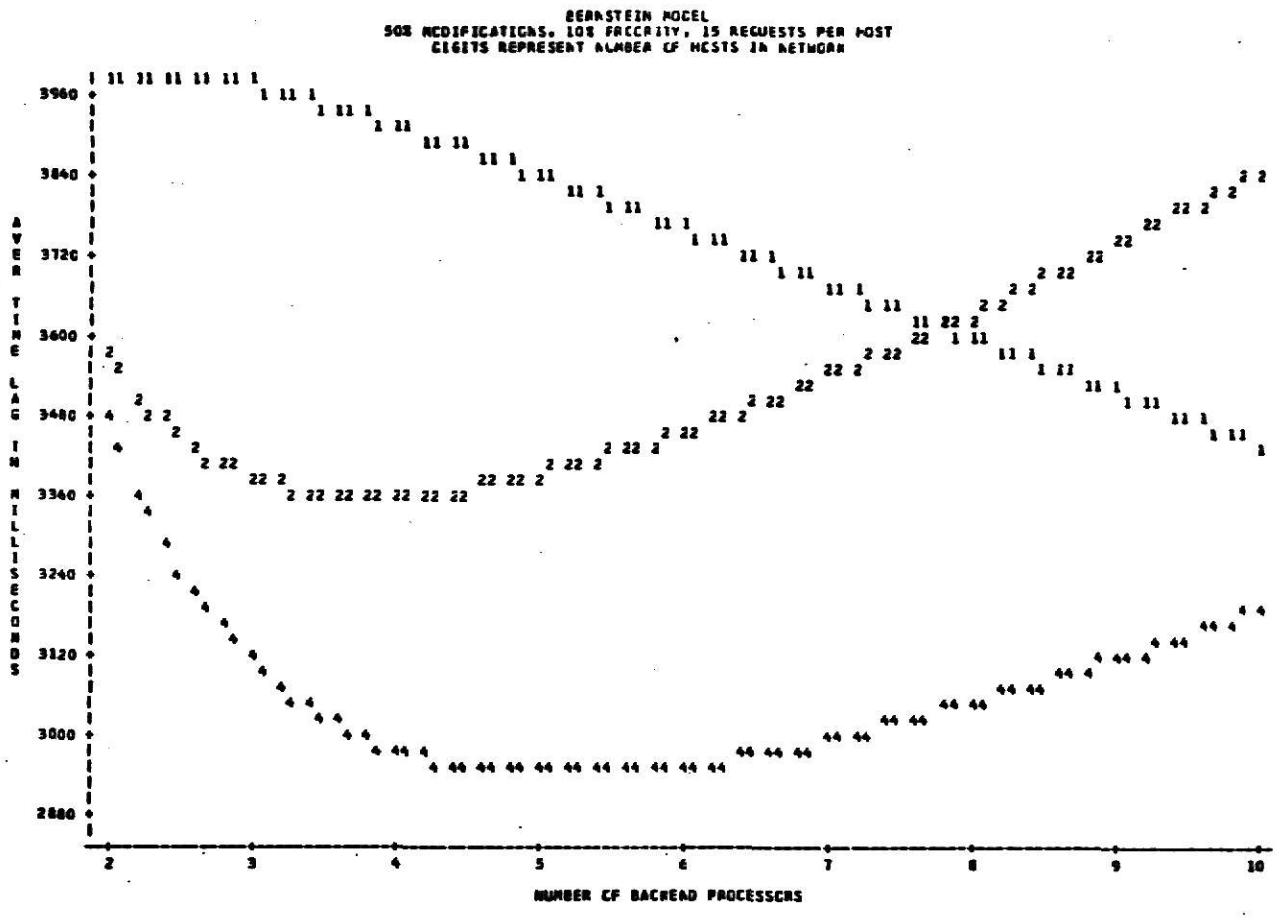


Figure 6.11

6.3.2 BERNSTEIN MODEL VARYING HOSTS

The plot above gives the average time lag, $\bar{t}_d$, for the Bernstein model for networks of 1, 2, and 4 hosts. The Bernstein model is unique among the models studied in that an increase in hosts to both two and four result in significant improvements in performance for low numbers of back-ends despite the consecutive doubling of the workload. The plot in Figure 6.11 clearly demonstrates the advantage of adding multiple hosts using the Bernstein algorithm.

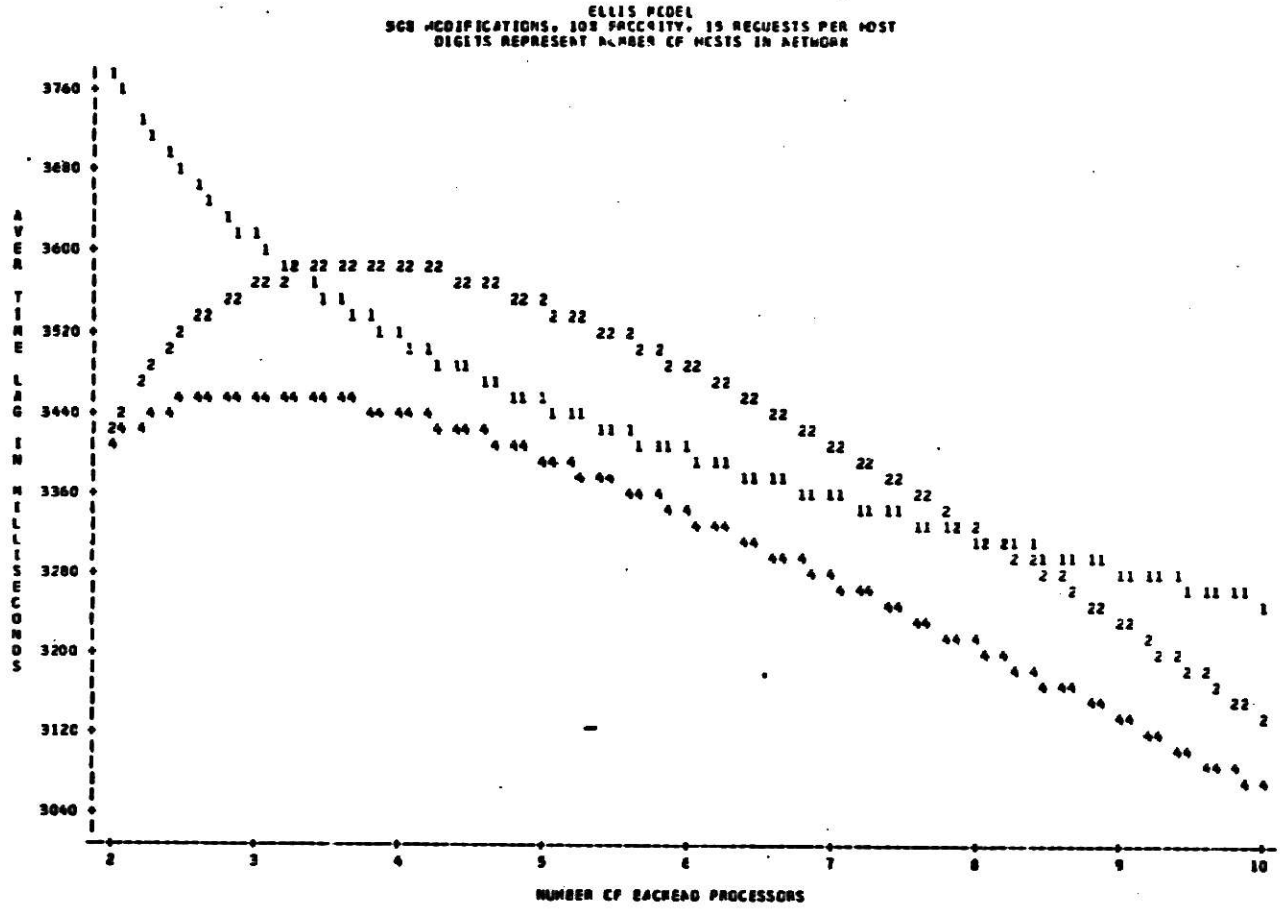


Figure 6.12

6.3.3 ELLIS MODEL VARYING HOSTS

Except for the performance at B = 2, the time lag for the Ellis model is not greatly affected by an increase in the number of hosts. A significant trend in the plot is that, after B = 4, the Ellis model steadily increases in performance as the number of back-ends are added to the network.

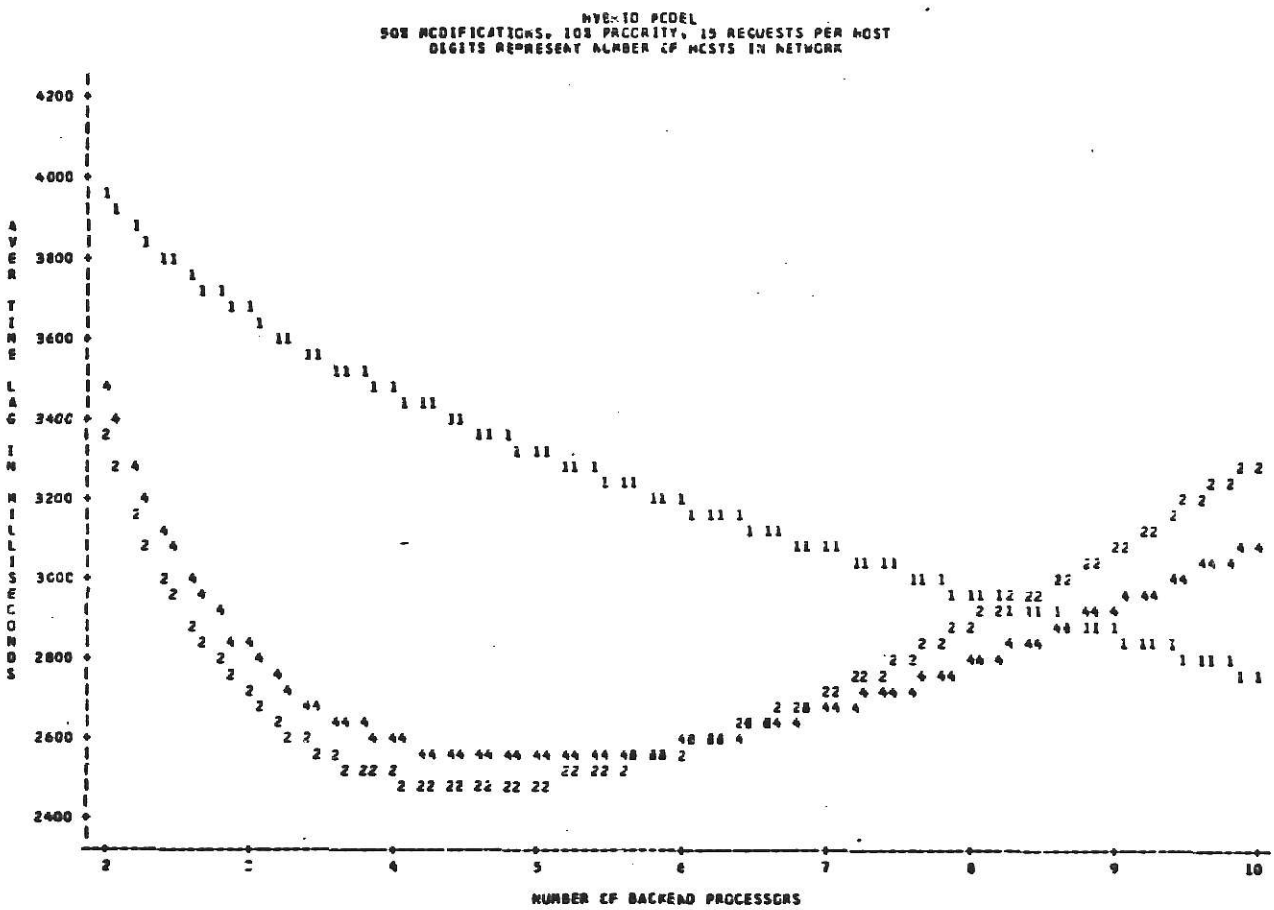


Figure 6.13

6.3.4 HYBRID MODEL VARYING HOSTS

As indicated in Figure 6.13, the effect of the increase from one host to two hosts significantly improves performance. Adding additional hosts; however, has little effect. The workload per host is constant for each plot; however, the increased workload per back-end begins to have an effect on performance for B > 5 and H > 1.

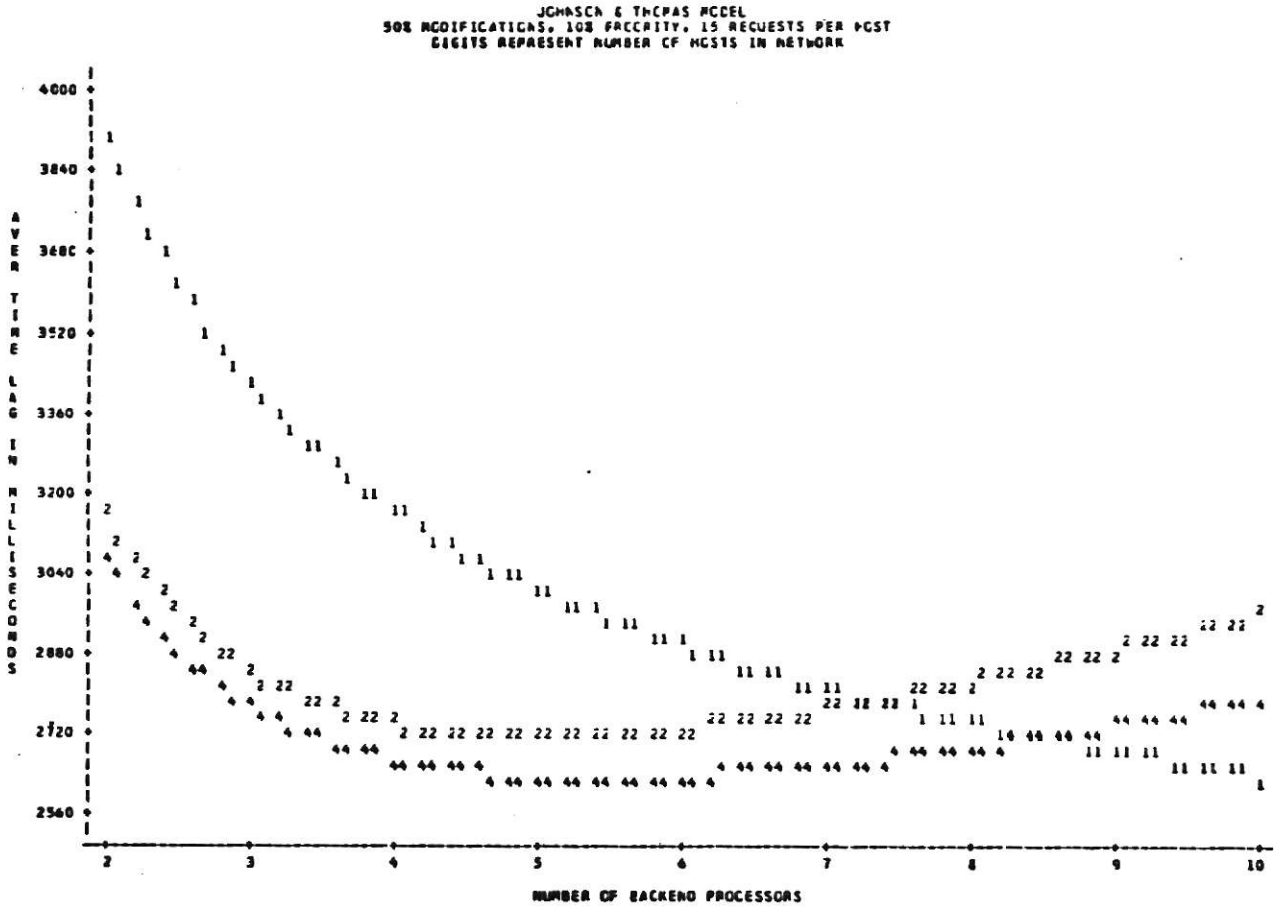


Figure 6.14

6.3.5 JOHNSON AND THOMAS MODEL VARYING HOSTS

As seen in the above figure, the Johnson and Thomas model responds in a manner similar to the Hybrid model as the number of hosts are varied. The increase of one host to two hosts has the most significant effect.

6.4 MODELS COMPARED VARYING BACK-ENDS WITH CONSTANT WORKLOAD PER BACK_END

All plots in the previous sections have been with a constant workload per host. As discussed in Chapter 5, the mathematical models were developed by first varying the workload per back-end for the Bernstein model. To examine the relationship of the parameters, keeping the workload per back-end constant, a SAS program was used to plot $\bar{t}_d$ versus B for each model. The plots are discussed in this section. Note that in this study only the Bernstein model was simulated with varying workload per host and constant workload per back-end. Models other than the Bernstein model are plotted in this section. The models have demonstrated that they react to the experimental parameters in a very similar manner, especially the four models with the same general mathematical models. To keep the workload constant at a value that was very close to the average for the simulations conducted, the workload was computed as follows for all data reflected in the plots of this section:

$$f = (3/2) * B$$

$$R = 15 * B$$

$$S = f * R$$

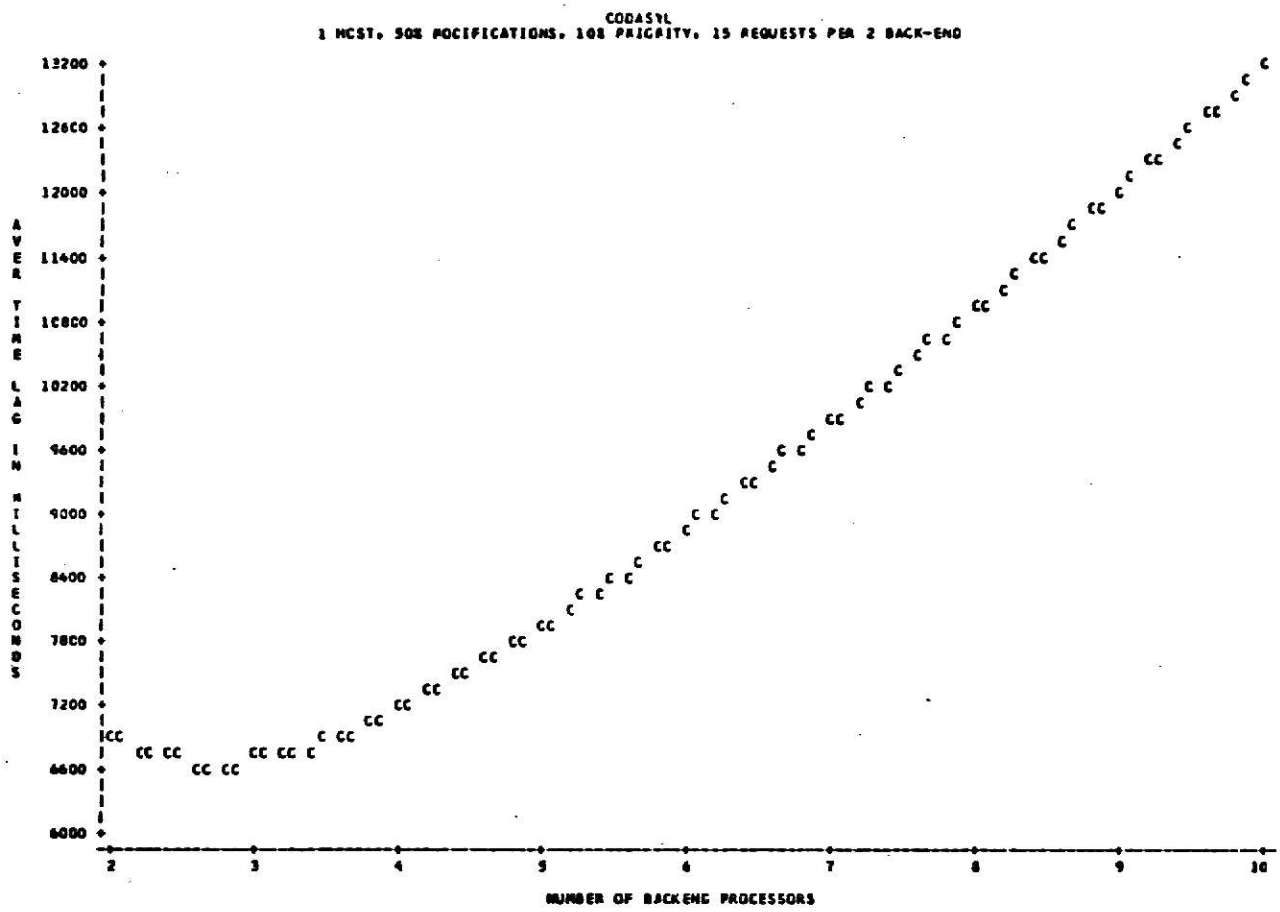


Figure 6.15

6.4.1 CODASYL MODEL, 1 HOST

As seen in Figure 6.15, above, t_d is related almost directly to the number of back-ends in the network if the workload per back-end is kept constant. The lag time, $\bar{t}_d$, for the CODASYL model was also closely related to the number of back-ends for a constant work-load per host as seen in Section 6.2.

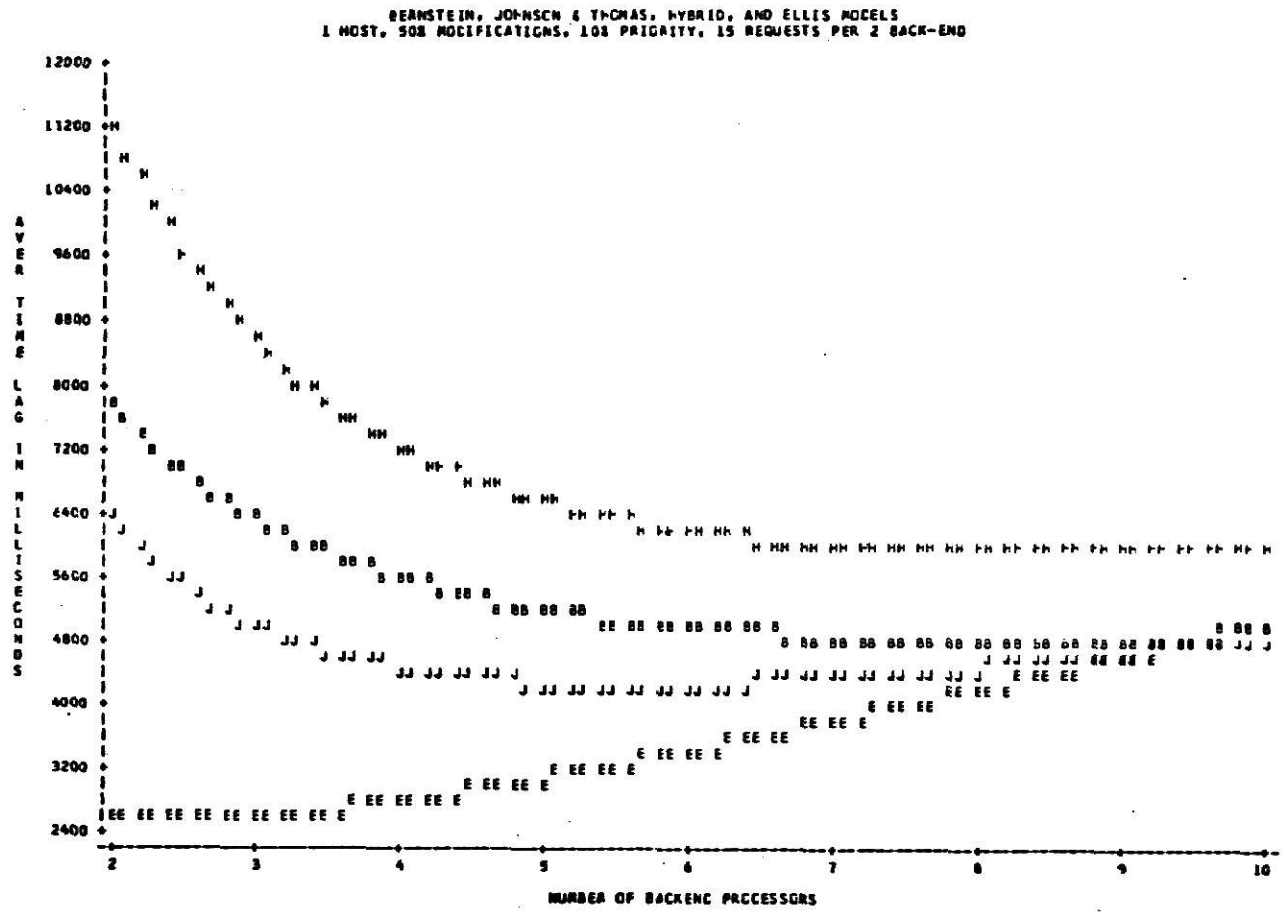


Figure 6.16

6.4.2 MULTIPLE MODELS, 1 HOST

The plot above is for a one host network. The Hybrid, Johnson and Thomas and Bernstein models each respond in a similar manner as the number of back-ends are increased. Performance is enhanced as B increases with a constant workload per back-end in a one host network. The Ellis model steadily increases in lag time as B increases. Note that the low $\bar{t}_d$ for the Ellis model at B = 2 may be an inconsistency in the mathematical model, as discussed earlier, due to insufficient data. The Ellis model reacts more consistently

in the plots in Sections 6.4.4 and 6.4.6.

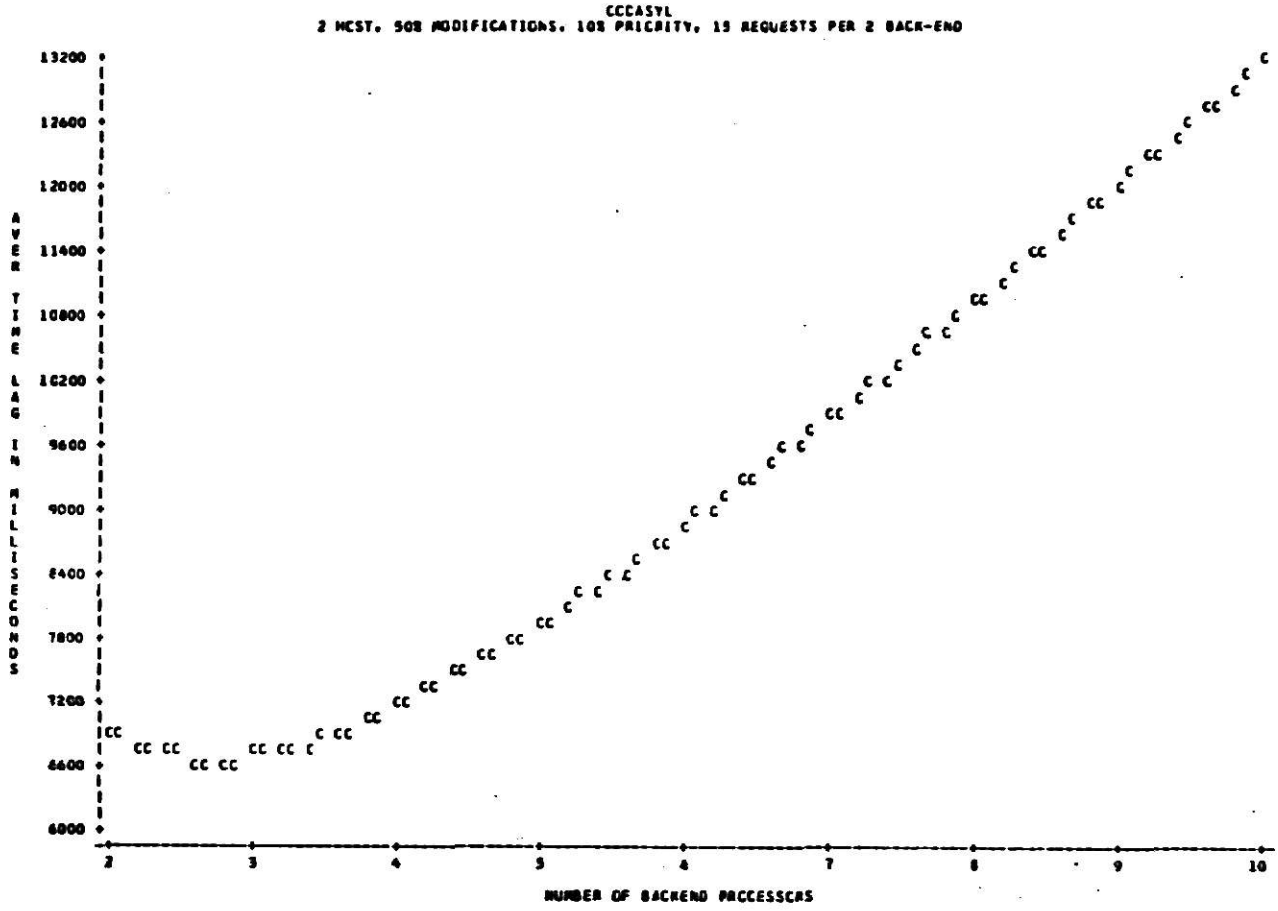


Figure 6.17

6.4.3 CODASYL MODEL, 2 HOSTS

The plot of the CODASYL model above is virtually unchanged from the plot in Figure 6.15. The CODASYL algorithm, performs updates after the last request is made in the job stream and the effect of an additional host is seen to add little to the efficiency of the modification updates.

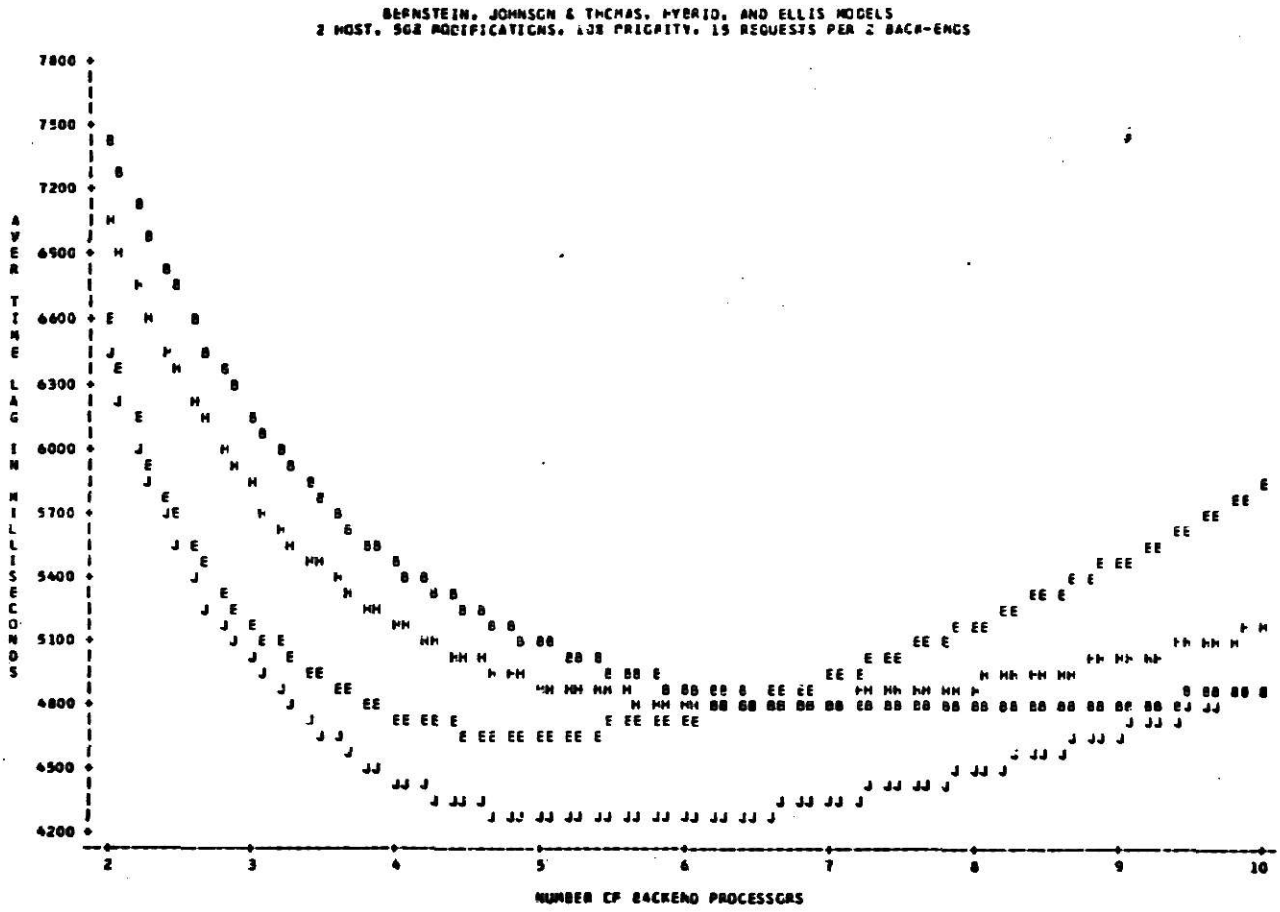


Figure 6.18

6.4.4 MULTIPLE MODELS, 2 HOSTS

The addition of a second host above changes the plot from that in Figure 6.16. All of the models plotted respond in very much the same manner. With a constant workload per back-end, all models respond better with the increase of back-ends and all models, at some point, are affected by the overhead of adding additional back-ends.

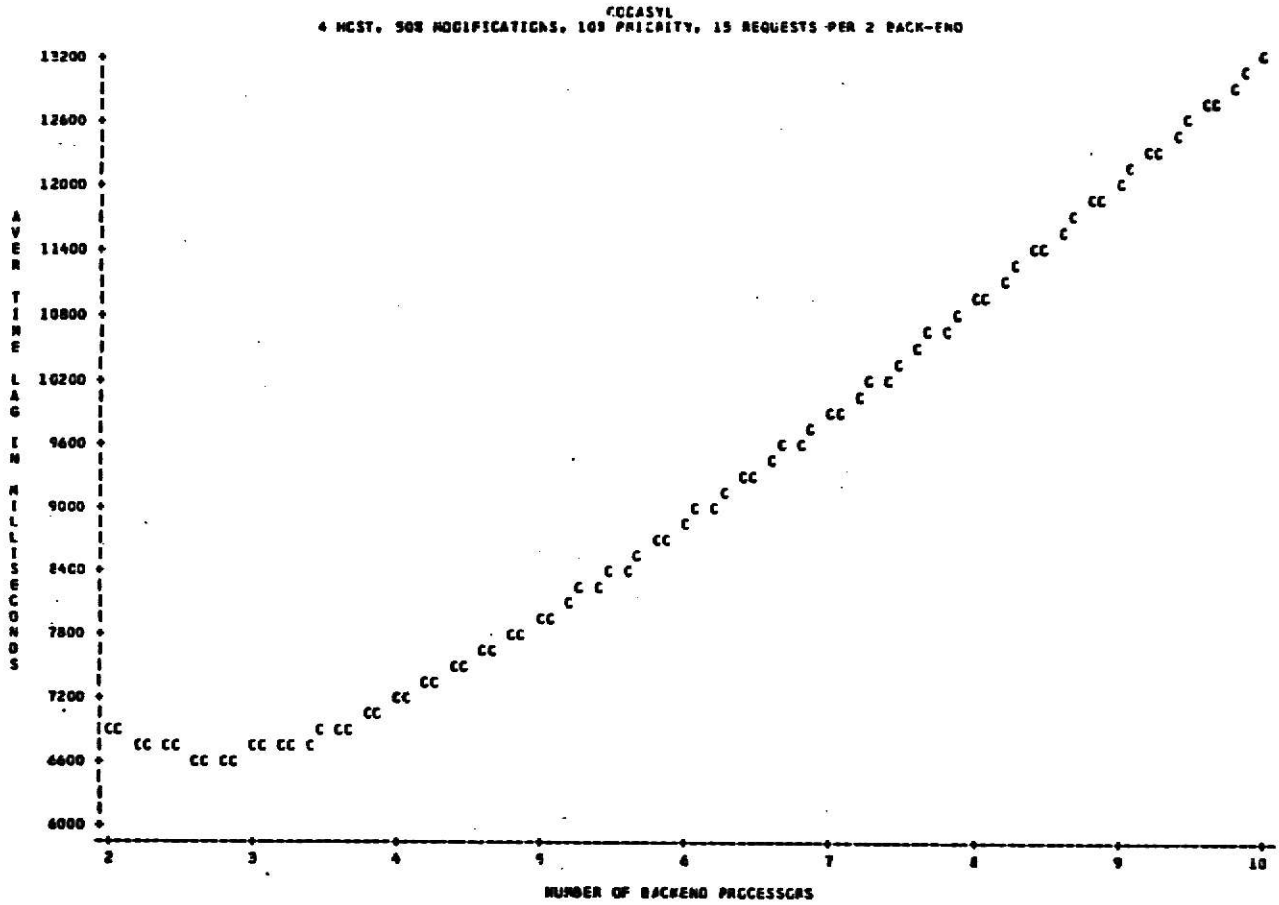


Figure 6.19

6.4.5 CODASYL MODEL, 4 HOSTS

As noted in Section 6.4.3, the CODASYL model is affected little by adding hosts if the workload per host is constant.

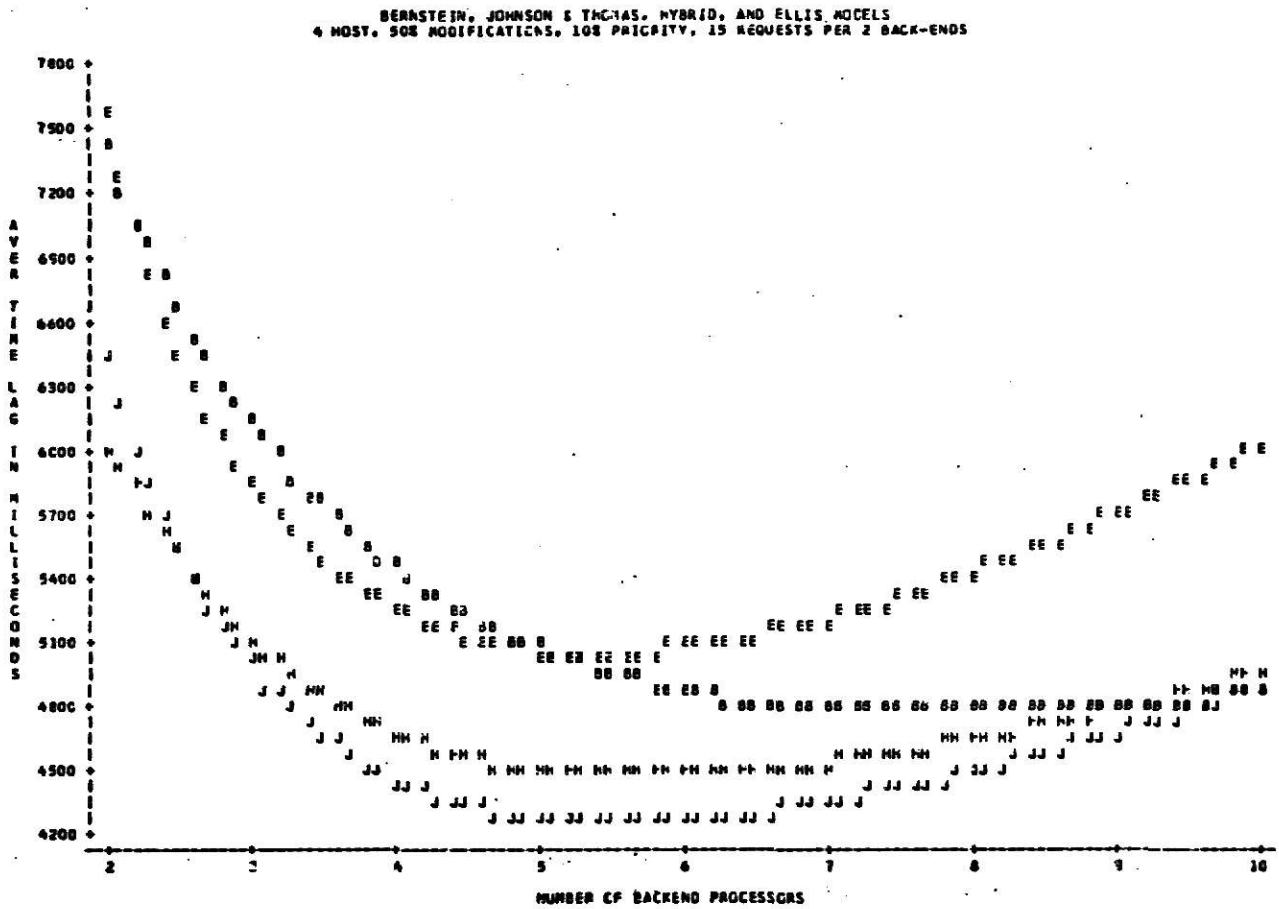


Figure 6.20

6.4.6 MULTIPLE MODELS, 4 HOSTS

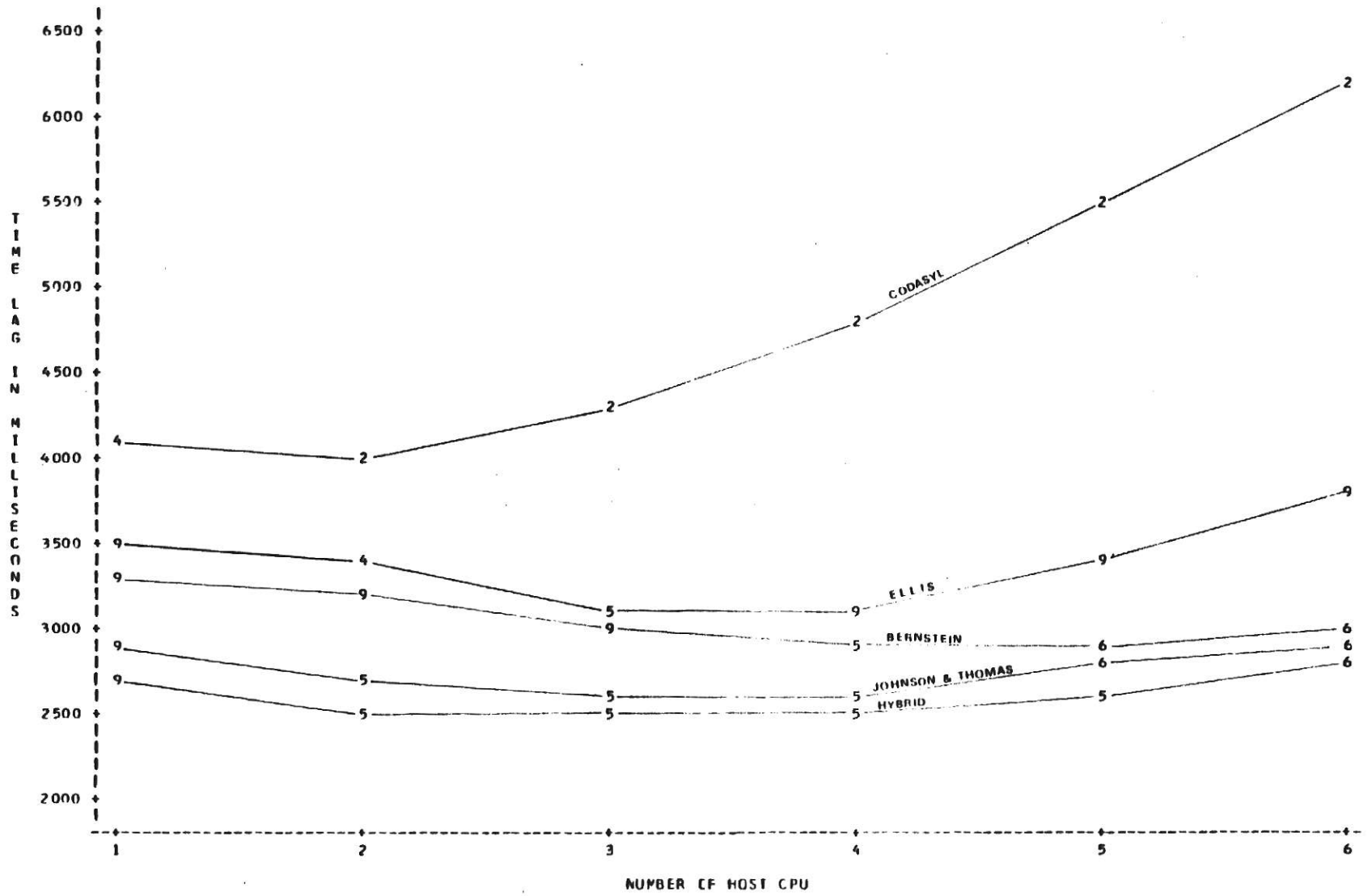
The plot in Figure 6.20 reflects the effects of the number of back-ends with four hosts. The $\bar{t}_d$ of the Hybrid model decreases, however, the other three models respond in essentially the same fashion as with two hosts.

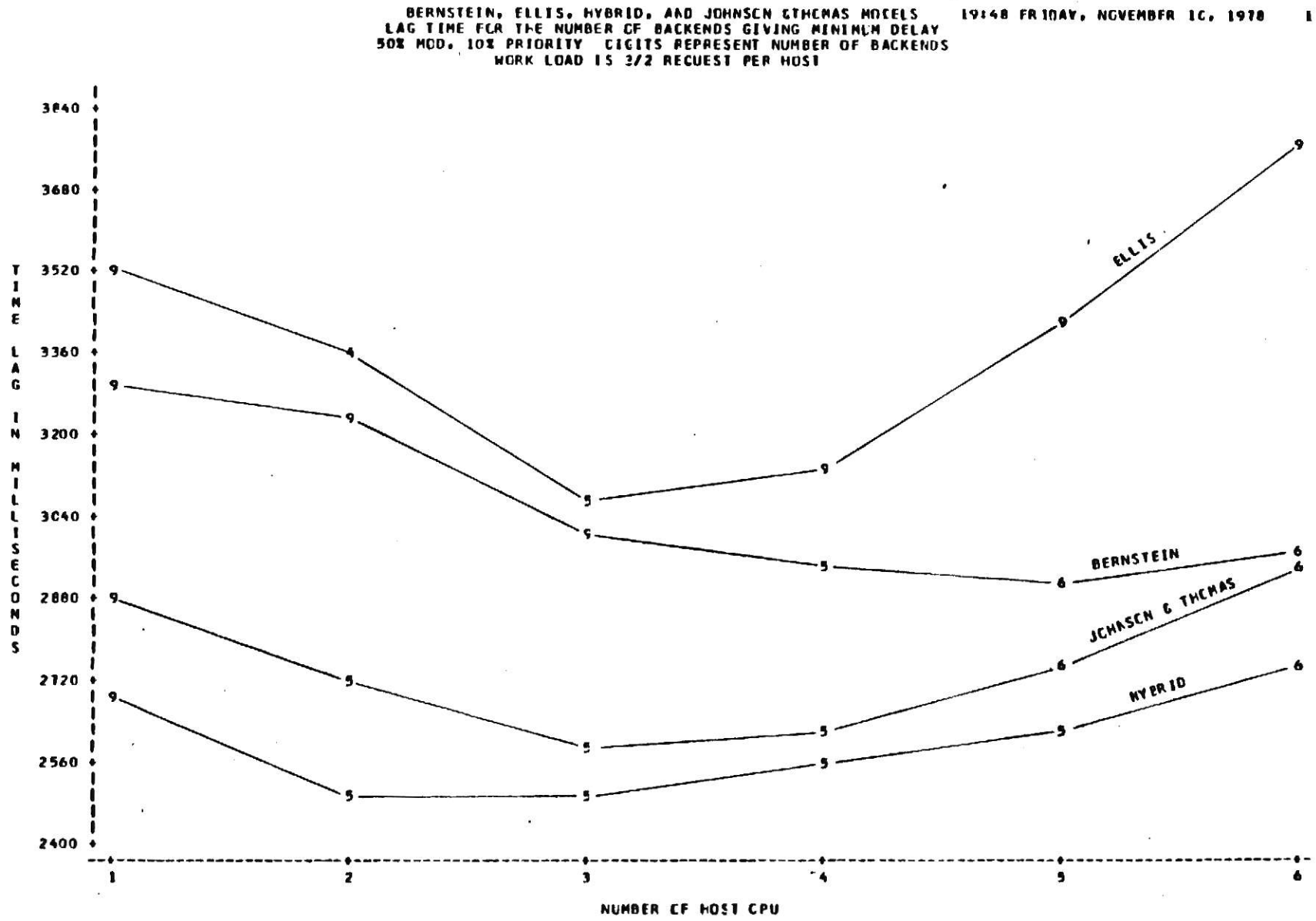
6.5 EFFECT OF NUMBER OF HOSTS ON THE OPTIMUM NUMBER OF BACK-ENDS WITH CONSTANT WORKLOAD PER HOST

As noted in Section 6.2, depending on the number of host machines in the network, a unique number of back-end processors can be computed which will give the optimum response time, $\bar{t}_d$, for a constant host machine work load. This section presents a discussion of plots which graphically represent this relationship. Using a SAS program the number of back-ends that provided the minimum time lag for each model was computed for each number of hosts. An example of the SAS program used is at Appendix 7. Figure 6.21 and 6.22 are the plots of the optimum numbers of back-ends plotted. The first figure includes all five models; the second figure excludes the CODASYL model in order to reduce the range of $\bar{t}_d$ on the vertical axis and compare the differences in the remaining models. As seen in the two plots, the Hybrid and the Johnson and Thomas models provide the best performance with fewer back-ends throughout most of the range as the workload per host is kept constant.

6.6 SUMMARY OF PLOTS

In Section 6.2 plots of the parameters of the update algorithms were compared keeping the workload per host constant. The Johnson and Thomas model and the Hybrid model perform significantly better than the other models studied at higher workloads. In Section 6.3 each of the models were plotted to examine the effect of adding additional hosts as


BERNSTEIN, CODASYL, ELLIS, HYBRID, AND JOHNSON & THOMAS MODELS
LAG TIME FOR THE NUMBER OF BACKENDS GIVING MINIMUM DELAY
50% MOD, 10% PRIORITY DIGITS REPRESENT NUMBER OF BACKENDS
TIME LAG IN MILLISECONDS
6500
6000
5500
5000
4500
4000
3500
3000
2500
2000
1
2
3
4
5
6
NUMBER OF HOST CPU
CODASYL
ELLIS
BERNSTEIN
JOHNSON & THOMAS
HYBRID


BERNSTEIN, ELLIS, HYBRID, AND JOHNSON &THOMAS MODELS 19:48 FRIDAY, NOVEMBER 10, 1978 1
LAG TIME FOR THE NUMBER OF BACKENDS GIVING MINIMUM DELAY
50% MOD, 10% PRIORITY CIGITS REPRESENT NUMBER OF BACKENDS
WORK LOAD IS 3/2 REQUEST PER HOST
TIME LAG IN MILLISECONDS
3840
3680
3520
3360
3200
3040
2880
2720
2560
2400
ELLIS
BERNSTEIN
JOHNSON & THOMAS
HYBRID
1
2
3
4
5
6
NUMBER OF HOST CPU

the workload per host remain constant. The Bernstein model improved significantly with the additiion of hosts. The Hybrid and the Johnson and Thomas model increased in performance with the increase in one additional host to the network but were little affected when hosts were further increased. In Section 6.4 the models were compared with the workload per back-end constant instead of the workload per host constant as in previous comparisons. Each model was similar in that as back-ends were added, performance improved until the overhead of additional back-ends forced the time lag to increase.

In Section 6.5 it was seen that for a constant workload per host, there is a unique number of back-ends that give the optimum $\bar{t}_d$. The Hybrid, Johnson and Thomas, and the Bernstein models performed best and each performed with optimum response times at five to six back-ends as the workload per host increased. The Bernstein model did not perform as well as the Hybrid and Johnson and Thomas models in networks if the number of hosts decreased.

6.7 GPSS STATISTICAL OUTPUT

As seen in the plots of the parameters of the models' variables in this chapter, each of the five models, at various specific workloads, demonstrate increased performance as back-ends are added, then the trend is reversed at an optimum point and performance is degraded. This degradation

is due to overhead of the algorithm required to update the additional data bases. The statistical output of the GPSS simulations were compared in an attempt to correlate the trend described to queuing behavior of the models. The comparison was limited by the data available. Except for the Bernstein model, statistical output was available for only 2 and 4 host networks. (Data available from the Norsworthy study of one host networks included mean response times and standard deviations as listed in his Master's Report [15].) Although the statistical data to examine the non-linear relationships was limited, the following demonstrates that the number of required accesses to the data bases by the back-end processors may have a significant correlation to the model's performance.

Table 6.2
Bernstein Model

BACK-ENDS	1 HOST	2HOST	4HOST
2	1,050	1,500	3,400
4	800	1,300	2,500
8	500	1,200	3,100

Average Number of Entries of Each
Back-end to Data Base Queue
(Constant Work Load Per Host)

Note that as the total workload increases, the back-end's entries to its data base also increases. The

addition of back-ends for one and two hosts decreases the number of accesses in a near linear manner. The data above for four hosts is of particular interest. Instead of decreasing in number of data base accesses, the number increases as the total back-ends increase from four to eight.

For the other models studied, as the number of back-ends were increased the total number of accesses to each data base decreased. There was no observed trend where the queueing or the number of accesses to a resource was reversed as in the Bernstein model's statistical output.

6.8 ANALYSIS OF THE MODEL'S PERFORMANCE

The mathematical models were developed from fifteen data samples of the CODASYL model and a total of fifty-seven of the remaining models studied. Each response time, $\bar{t}_R$, is a mean of six simulations, each with different random number within the simulation. The data resulting from over 300 simulations were used to formulate the mathematical models to describe the model's performance within a multi-host, multi-back-end environment. Some of the limitations of the mathematical models have been discussed previously.

Now that each of the relationships plotted have been discussed, there are some additional limitations that should be considered. Data was obtained for each of the models at $B = 2$, 4, and 8. When observing the curves in the previous sections, it is important to note that where the optimum

number of back-ends normally is 5 to 6, the minimum of the curve was extrapolated without obtaining any data from any of the models within that optimum range of back-ends. The optimum B for each model may vary from that presented in this chapter. Further, the workload per host was held constant for most of the simulations. The plots with a constant workload per back-end are presented so that the function of that parameter could be demonstrated and is not as significant as the plots with constant workload per host.

From the results presented, it is significant that some algorithms may benefit by adding back-ends and hosts and that others do not notably improve. Also depending on the workload of the algorithm, there exists an optimum number of back-ends in a back-end DBMS. Definite advantages can be achieved if the optimum number of back-ends and hosts can be theoretically determined in the design phase of a back-end DBMS. Hopefully, the results of research in this area of study will allow designers to better estimate the optimum configuration of a proposed back-end network.

It would be difficult, based on the results of this study, to state one algorithm is best because of its performance in the simulation experiments. As discussed before, this study indicates where performance may increase by adding hosts and/or back-ends to the network.

It is helpful in the comparison of the algorithms to look at factors other than performance.

The CODASYL model guarantees consistency in data base update but may provide out-dated information during a work day.

As discussed in reference (15), the Johnson and Thomas algorithm may result in inconsistency in the data bases.

The Bernstein model has been formally proven and its performance is comparable to the Hybrid and Johnson and Thomas algorithms in a multi-host environment. The cost in memory to maintain clock times for each record must be an important factor to be considered, particularly for very large data bases.

The Hybrid algorithm is less complex than the Bernstein algorithm and performs consistently better than all other models in this study.

Concurrency of update processing is the primary factor that appears to enhance performance of a back-end DBMS. CODASYL's lack of concurrency in updating data bases is cause for its failure to improve performance by adding back-end processors. The Hybrid model's performance is most readily enhanced by additional back-ends and hosts due primarily to its ability to maximize the computing power available. The Bernstein methodology requires significant overhead, yet benefits from added back-ends and hosts due to its ability to be able to efficiently overcome the overhead of the algorithm.

Chapter 7

CONCLUSION

7.1 SUMMARY

Using simulation models, the performance of five consistency algorithms were compared in varying architecture configurations. The response time of each methodology was measured in relation to the parameters of a varied multi-host multi-back-end architecture. Because the methodologies responded in similar ways to the experimental parameters, general mathematical models were able to be formulated to describe the performance of the models.

The mathematical models were utilized to graphically demonstrate that optimum architectural configurations can be estimated, if a constant workload is assumed. The mathematical models were also used to illustrate that data base performance is enhanced in many cases by addition of back-end and/or host machines to the DBMS architecture. The trend of increasing performance by adding back-ends to a DBMS is reversed at a point where the methodologies' overhead becomes a significant factor in the synchronizing process of updating the redundant data bases. Workload of the back-ends was found to be a significant factor in the degree to which performance could be enhanced by additional hardware.

Performance of the CODASYL model was not significantly improved by adding back-ends to the DBMS network. The Bernstein, Ellis, Hybrid, and the Johnson and Thomas

methodologies were each enhanced to various degrees by addition of additional host machines and back-end processors even though the workload remained constant. Based on the proposition that architecture of a back-end DBMS should be designed based on heavy workload per machine in the network, configurations can be designed in a back-end DBMS to optimize performance in an environment that provides the benefits of concurrency of updates and the security of redundant data bases.

7.2 FUTURE CONSIDERATIONS

7.2.1 MATHEMATICAL MODELS

As discussed in previous chapters, the original purpose of this study was to measure the effect of multiple hosts on a back-end DBMS using simulation models. During the analysis phase of the experiment, general mathematical models were developed. The mathematical models are not robust and, in fact, are very limited in the range of parameters that can be used to express performance of the consistency models. A significant finding of this study is that general mathematical models can be developed to describe a methodology as complicated as the models studied. The parameters that must be tested to adequately measure the effect on the models are as follows:

Workload per back-end

Performance of the models adjusting the number of back-ends
--near the optimum number of back-ends (approximately 6)

--for large number of back-ends (> 10).

Modification (percent) varied during heavy workload

Priority (percent) varied during heavy workload

Additional data to adequately test the above parameters would allow general mathematical models to be developed that are capable of becoming a design tool rather than a means of describing experimental results.

7.2.2 Real System Performance

The mathematical models developed in this experiment could perhaps be refined and validated by obtaining comparable response times from actual operating back-end DBMS networks. In this way the models may be able to be of utility in examining working systems and proposing improvements in the network architecture.

7.2.3 Simulation Model Measurement

A perhaps less costly method of measuring performance of the simulation models than the method used in this study would be to mark individual requests when entering the

network and time required to process the request to completion. An advantage to this approach is that large amounts of data could be obtained from one simulation program; for example, the frequency of input requests could be varied during one simulation.

7.3 ROUND-ROBIN METHODOLOGY PROPOSAL

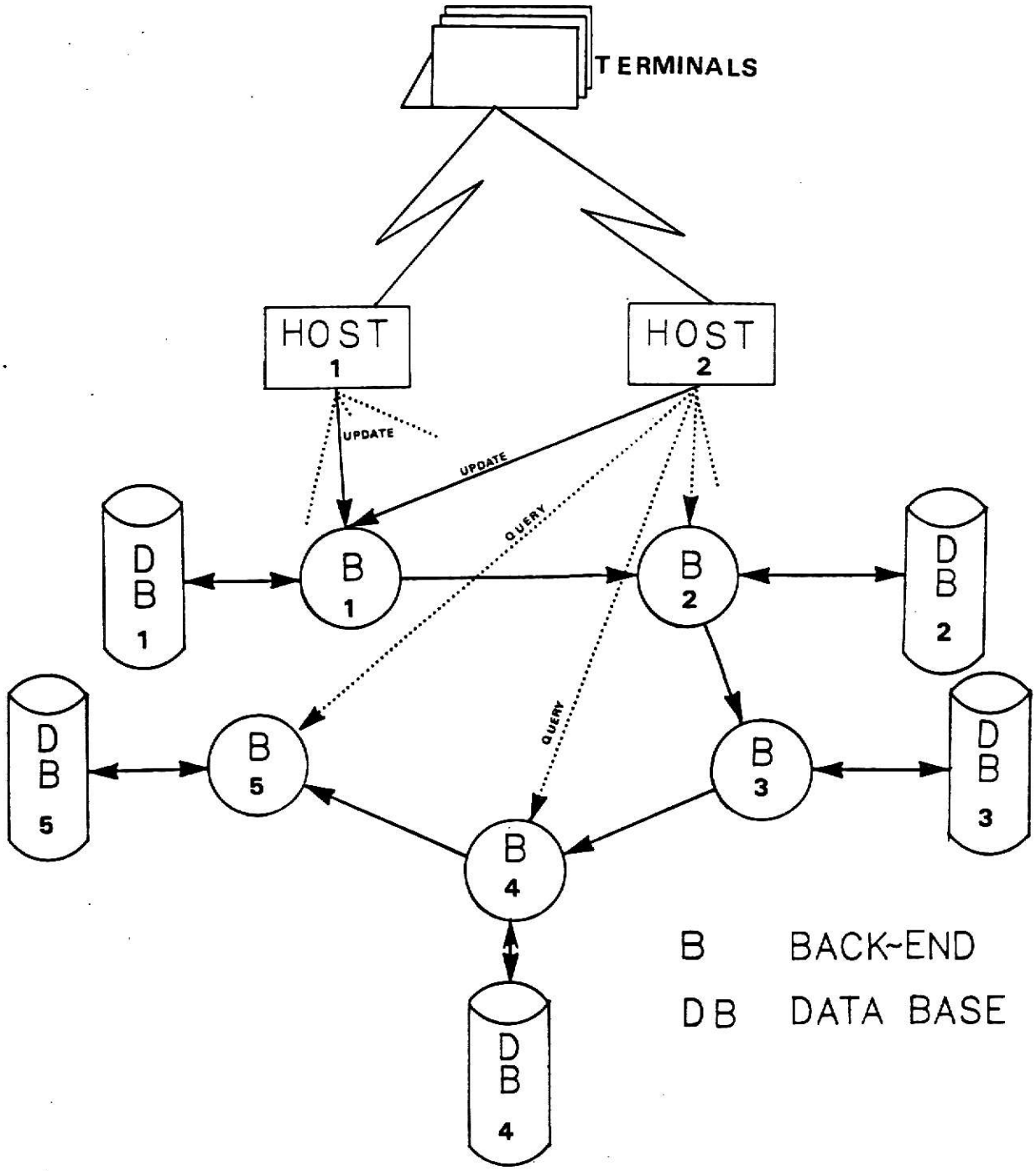
As a possible alternative consistency algorithm, the following model is proposed for consideration by future researchers. The round-robin update algorithm is represented in the network in Figure 7.1 and in the simulation flow chart in Figure 7.2.

Only one back-end is designated as primary. When query requests are received in the network, they are randomly assigned a back-end which responds to the request. However, if the request is to modify the data base, modifications are randomly sent to a host, but, after raising priority of the request, are transmitted to the primary back-end. The flow of updating is in an established order from the lowest numbered back-end to the highest-numbered back-end.

When an update request is sent to the primary back-end, the request is placed in the modification table of the back-end. If the record to be modified is currently being modified by a prior request, the request waits in a queue until continued. When continued, the back-end (1) sends a copy of the update request to the next back-end in the

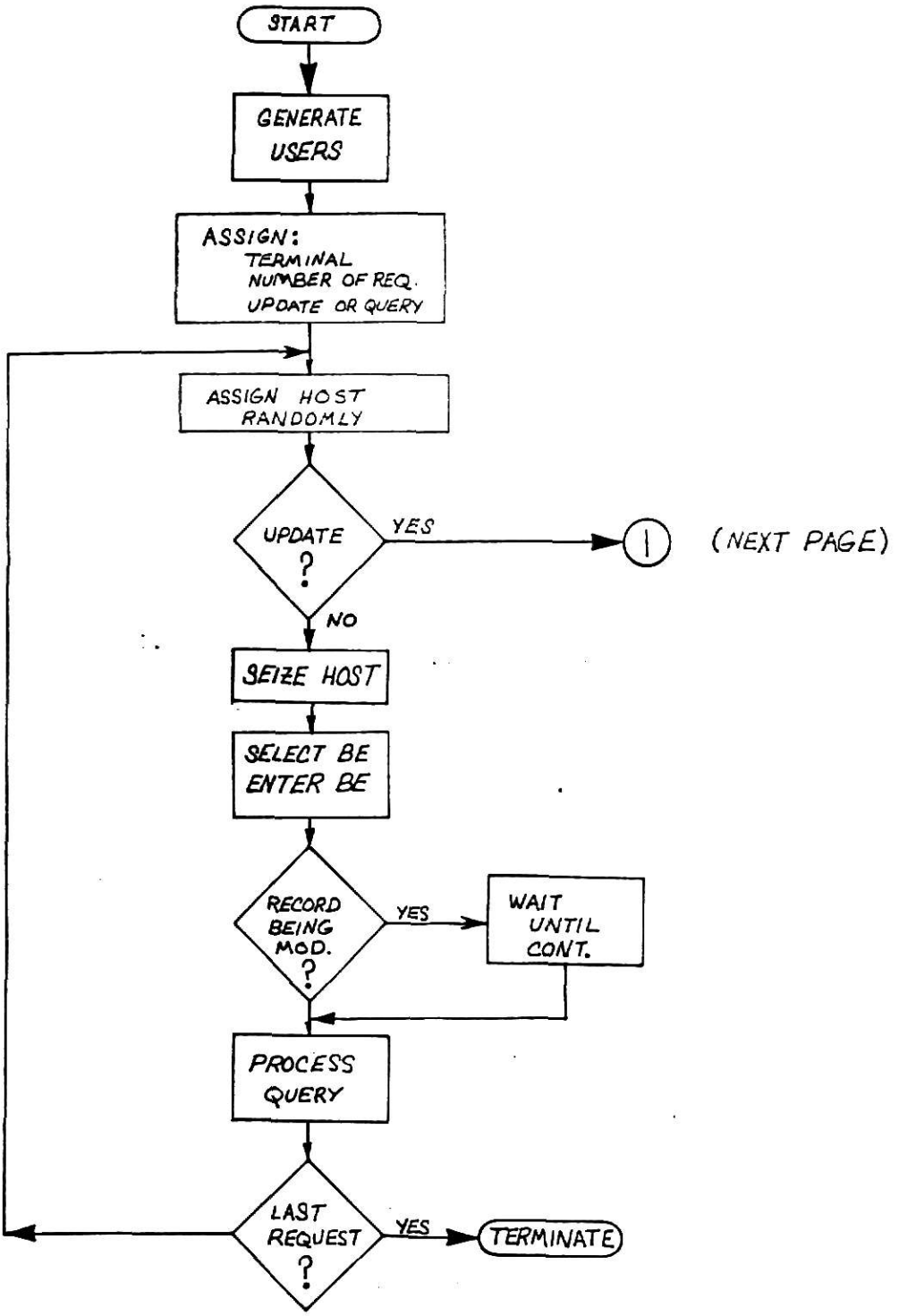
FIGURE 7.1

ROUND ROBIN

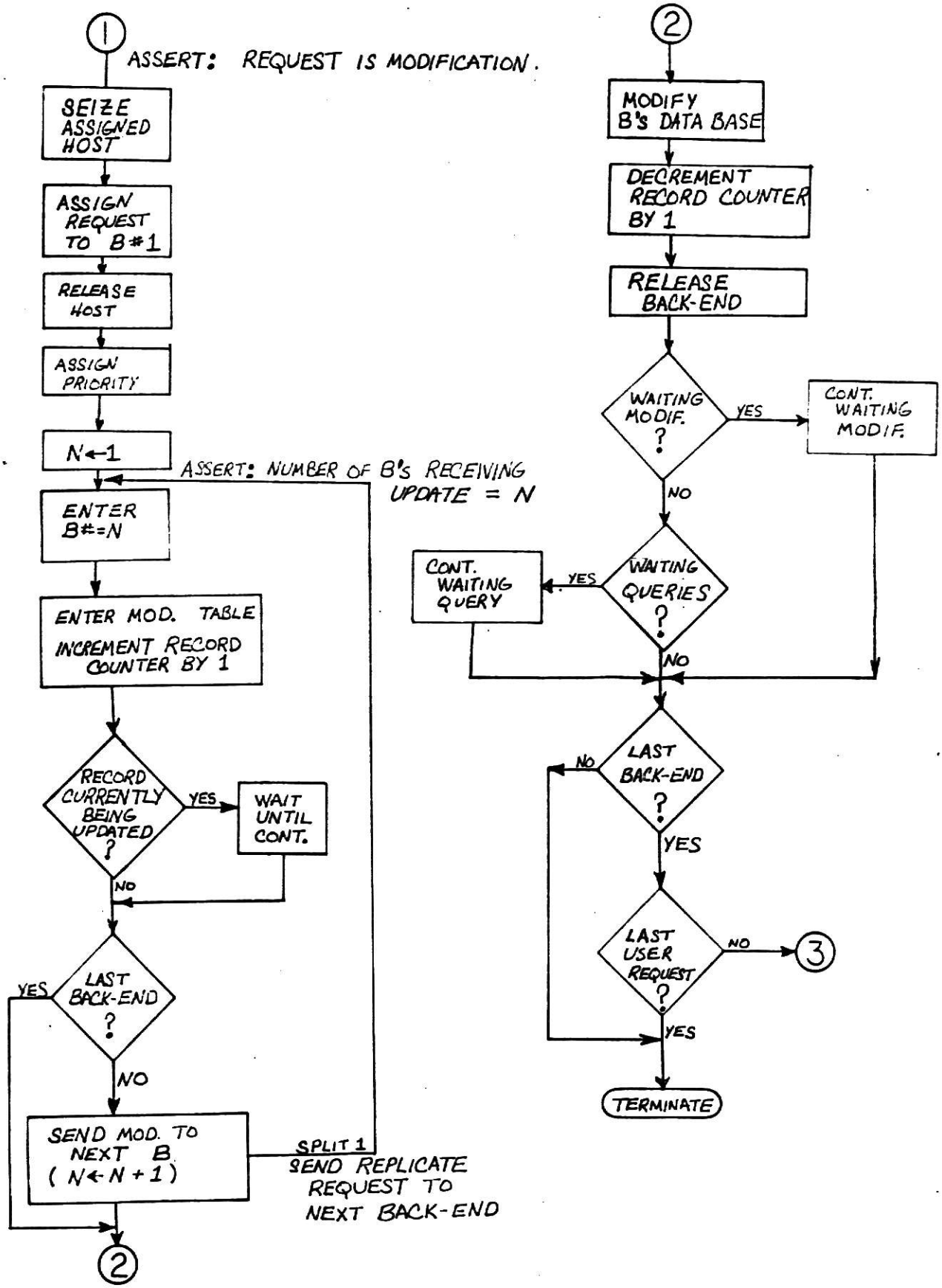


ROUND ROBIN FLOW CHART

FIGURE 7.2



CONTINUATION FIG. 7.2



network, then (2) proceeds to update the primary data base to completion. Modification requests waiting to update the data base are continued prior to the request terminating. The update procedure described at the primary back-end is continued for the remaining back-ends.

The expected advantages of the described algorithm are:

(1) Clock times are not required because update requests are always sent in the same order and requests are not able to get out of sequence.

(2) Delays for voting are unnecessary.

(3) Concurrency is enhanced because update requests are sent to the next data base prior to the update of the current data base.

(4) If a back-end is nonoperational, the modification requests may easily be stored until it becomes operational without losing consistency in the data base.

(5) Each back-end in the network sends an update request to only one other back-end. Permission is not required from any back-end to send the request.

(6) The Round-Robin algorithm allows the designer to specify the order in which the data bases are updated. A hierarchical organization, such as a military force structure, may prefer this algorithm because the order in which each data base is updated may be critical--particularly

during crisis situations when workload may peak.

(7) Unlike other algorithms, it is possible for all back-ends to be performing an update on the same record concurrently.

During analysis of the performance of the five methodologies of this study, it was noted that adding back-ends normally increased performance; however, the trend would reverse when overhead of the methodologies became significant. A major part of the overhead was due to one designated primary back-end communicating with many back-ends; the larger the number the greater the overhead. Poor performance is predicted for the methodologies studied in this report if a great number of back-ends (over 12) are added to the networks. Alternatively, as back-ends are added to the Round-Robin network, the performance is expected to decrease to a steady value and <u>not</u> lose the gained performance at high values of B. There is additional workload only for the last back-end in the network as an additional back-end is added.

The overhead of the proposed algorithm is similar to the Hybrid algorithm studied in this report; modification tables are required at each back-end. The primary difference is that instead of waiting for a back-end partition to be free as in the Hybrid algorithm, the update requests are sent immediately and always in the same direction in the network.

REFERENCES

1. Alsberg, P. A., et al., "Synchronization and Deadlock," CAC Document Number 185, CCTC-WAD 6503, Center for Advanced Computation, University of Illinois at Urbana-Champagne, Urbana, Illinois, March 1, 1976.

2. Barr, A. J.; Goodnight, J. H.; Sall, J. P.; and Helwig, J. T., SAS, SAS Institute Inc., January 1977.

3. Bernstein, P. A.; Rothnie, J. B.; Goodman, N.; Papadimitrou, C. A., "The Concurrency Control Mechanism of SDD-1: A System for Redundant Databases (The Fully Redundant Case)," IEEE Transactions of Software Engineering, Vol. SE-4, No. 3, May 1978, pp. 158-167.

4. Canady, R. H.; Harrison, R. D.; Ivie, E. L.; Ryder, J. L.; Wehr, L. A., "A Backend Computer for Data Base Management," Communications of the ACM, Vol. 17, 10, 1974, pp. 575-582.

5. Ellis, C. A., "A Robust Algorithm for Updating Duplicate Data Bases," Proceedings of the Second Annual Berkeley Workshop on Distributed Data Management and Computer Networks, 1977, pp. 146-158.

6. Everest, G., Personal Communications, Fall, 1977.

7. General Purpose Simulation System V User's Manual, (SH20-0866) IBM Corporation Third Edition, Sept. 1977.

8. Gordon, G., The Application of GPSS V to Discrete System Simulation, Prentice Hall, Inc., 1975.

9. Johnson, P. R., and Thomas, R. H., "The Maintenance of Duplicate Databases," Network Working Group RFC 677, 1977.

10. Maryanski, F. J., "The Management of Redundant Data in a Distributed Data Base," TR 78-21, Computer Science Department, Kansas State University, Manhattan, KS 66506, Sept. 1978.

11. Maryanski, F. J., "A Survey of Developments in Distributed Data Base Management Systems," IEEE Computer (11,2) February 1978, pp. 28-38.

12. Maryanski, F. J.; Norsworthy, K. E.; and Northsworthy, K. A.; and Ratliff, J. R., "A System Architecture for Distributed Data Base Management," Proceedings IEEE COMPCON, March 1977.

13. Maryanski, F. J.; Wallentine, V. E.; Fisher, P. S.; and Calhoun, M. A., "A Distributed Data Base Management System Using Minicomputer," in Infotech State of the Art Report, Minis Versus Mainframes, 1978.

14. Maryanski, F. J. and Kreimer, D. E., "Effects of Distributed Processing in a Data Processing Environment," Proceedings of the 11th Annual Simulation Symposium, March 1978, pp. 183-197.

15. Norsworthy, K. E., "A Simulation Study Comparing Five Consistency Algorithms for Redundant Data Bases," A Master's Report, Kansas State University, 1978.

16. Rosenkrantz, D. J.; Stearns, R. E.; and Lewis, P. M., "A System Level Concurrency Control for distributed Database Systems," Berkeley Workshop on Distributed Data Management and Computer Networks, Lawrence Berkeley Laboratory, University of California, Berkeley, California, May 1977, pp. 132-145.

17. Rothnie, J. B. and Goodman, N., "A Survey of Research and Development in Distributed Database Management," Proceedings of the Third International Conference on Very Large Data Bases, TOKYO, Oct. 1977, pp. 48-62.

18. SAS Supplemental User's Guide, SAS Institute Inc., July 1977, pp. 130-141.

19. Stearns, R. E.; Lewis, P. M. II; and Rosenkrantz, D. J., "Concurrency Controls for Database Systems," Proceedings of the 17th Annual Symposium on Foundations of Computer Science IEEE 1976, pp. 19-32.

20. Stonebraker, M.; Wong, E.; Kreps, P.; and Held, G., "The Design and Implementation of INGRES," ACM Transactions on Database Systems, Vol. 1, No. 3, Sept. 1976, pp. 189-222.

21. Thomas, R. H., "A Resource Sharing Executive for the Arpanet," Proceedings AFIPS National Computer Conference, AFIPS Press, Vol. 42, 1973, pp. 155-163.

```
      REALLOCATE FAC,200,QUE,200,BLC,250
      REALLOCATE STO,10,XAC,200,VAR,10,FUN,15,TAB,0,LOG,90,COM,33300
      SIMULATE
*
*       THE BERNSTEIN MODEL
*       MODEL OF 4 HOST CPU'S, 2 BE'S, 50% UPDATES, & 10% PRIORITY
*       20 TRANX PER SEC PER BE
*
*
* **    ENTITY DEFINITIONS
*
*   PARAMETERS
*       PH1: TERMINAL ASSIGNMENT NUMBER
*       PF1: HALFWORD SAVEVALUE WHOSE NUMBER IS 10+PH5
*       PH2: NUMBER OF I/O REQUESTS NECESSARY FOR EACH USER RWQUEST
*       PF2: THE SPECIFIC RECORD NUMBER
*       PH3: SIZE OF MESSAGE BUFFER
*       PF3: THE ABSOLUTE CLOCK TIME
*       PH4: MARKED 0 IF READ REQUEST
*                   1 IF PRIMARY UPDATE REQUEST
*                   2 IF SECONDARY UPDATE REQUEST
*       PF4: MARKED 0 IF UPDATE REQUEST HAS NOT BEEN SPOOLED
*                   1 IF UPDATE REQUEST HAS BEEN SPOOLED
*       PH5: ID NUMBER OF THE BACKEND MACHINE BEING USED
*       PF5: NUMBER OF LAST TABLE ENTRY
*       PH6: NUMBER OF BACKEND MACHINES USED IN THE SIMULATION
*       PF6: PRIMARY OR SECONDARY UPDATE FLAG
*       PH7: A COUNTER
*       PH8: NUMBER OF REQUESTS MADE BY THE USER
*       PH9: CHANNEL ID USED BETWEEN TWO BACKEND MACHINES
*       PF11: TRANSMISSION SPEED TO CHANNEL IN BITS/SEC
*       PF12: CHANNEL TRANSMISSION SPEED IN BITS/SEC
*       PH13: TIME USED BY HINT OR BINT
*       PH14: TIME USED BY THE MESSAGE SYSTEM
*       PH15: ID NUMBER OF THE BE THAT THE SECONDARY UPDATE IS SENT TO
*
*   STORAGES REPRESENT PARTITIONS IN EACH BACKEND
*
*   HALFWORD MATRIX
*       1: UPDATE TABLE FOR BE1
*       2: UPDATE TABLE FOR BE2
*       3: UPDATE TABLE FOR BE3
*       4: UPDATE TABLE FOR BE4
*       5: UPDATE TABLE FOR BE5
*       6: UPDATE TABLE FOR BE6
*       7: UPDATE TABLE FOR BE7
*       8: UPDATE TABLE FOR BE8
 1      MATRIX      MH,250,2
 2      MATRIX      MH,250,2
 1      MATRIX      MX,250,1
 2      MATRIX      MX,250,1
 1      MATRIX      MB,250,1
 2      MATRIX      MB,250,1
        STORAGE     S1-S8,2
 PLUS   VARIABLE    10+PH1
 MINUS  VARIABLE    PH6-1
 TRHNL  FVARIABLE   PH3*8/PF11*1000
 CHANL  FVARIABLE   PH3*8/PF12*1000
 INCRE  VARIABLE    PH7+1
 BECNL  VARIABLE    10*PH5+PH15
```

```
 HCBE  VARIABLE   PF10+PH5
 PRIOR VARIABLE   PF13+2
*  50% UPDATE
 TYPE  FUNCTION   RN2,D2
.50,0/1.00,1
 TERM  FUNCTION   RN2,D8
.125,101/.25,102/.375,103/.5,104/.625,105/.75,106/.875,107/1.0,108
 RECRD FUNCTION   RN2,D10
.10,1/.20,2/.30,3/.40,4/.50,5/.60,6/.70,7/.80,8/.90,9/1.0,10
 REQNO FUNCTION   RN2,D10
.10,1/.20,2/.30,4/.40,5/.50,7/.60,9/.70,11/.80,12/.90,13/1.00,14
 LNGTH FUNCTION   RN2,D2
.90,128/1.00,256
*  90% FIT IN ONE BUFFER
 BEDEV FUNCTION   PH5,D8          BE TO DEVICE CHANNELS
1,11/2,22/3,33/4,44/5,55/6,66/7,77/8,88
 USREC FUNCTION   RN2,D9          # OF REQUESTS BY A USER
.10,1/.20,3/.40,5/.50,7/.60,9/.70,11/.80,13/.90,15/1.00,17
*
 RNDM  FUNCTION   RN2,D4
.25,1/.5,1/.75,2/1.0,2
 RNDM2 FUNCTION   RN2,D4
.25,111/.50,112/.75,113/1.0,114
 HOST  FUNCTION   PH10,D4
111,150/112,160/113,170/114,180
 PRIOR FUNCTION   RN2,D2
.9,1/1.0,11      ********      10% HIGHER PRICRITY REQUESTS  *********
*
       INITIAL    XH1-XH8,0      INITIALIZE SAVEVALUES 1-10 TC ZERO
       INITIAL    XF1-XF8,0      INITIALIZE XF1-8 TC ZERO
       INITIAL    XH11-XH18,0    INITIALIZE SAVEVALUES 11-14 TO ZERO
*
       GENERATE   750,50,,30,,15PF,15PH  GENERATE 30 TRANX 1 PER 3/2 SEC
       ASSIGN     15,,PF          NUMBER OF HOST CPU'S IN THE SIMULATION
       ASSIGN     6,2,PH          NUMBER OF BACKENDS IN SIMULATION
       ASSIGN     1,FN$TERM,PH    ASSIGN TERMINAL # IN PH1
       ASSIGN     8,FN$USREQ,PH   STORE # OF USER'S COMMANDS IN PH8
       ASSIGN     11,50000,PF        TRANSMISSION SPEED TO CHANNEL
       ASSIGN     12,50000,PF        TRANSMISSION SPEED OF CHANNEL
       ASSIGN     13,5,PH         ADVANCE TIME FOR HINT OR BINT
       ASSIGN     14,5,PH         ADVANCE TIME FOR MESSAGE SYSTEM
       QUEUE      PH1             QUEUE FOR TERMINAL
       SEIZE      PH1             SEIZE THE TERMINAL
       DEPART     PH1             DEPART QUEUE FOR TERMINAL
*
*              MAIN LOOP FOR USER REQUESTS
*
 MORE  ADVANCE    1000,500        TIME TAKEN TO TYPE REQUEST
       ASSIGN     2,FN$REQNO,PH   ASSIGN # OF IO REQUESTS IN PH 2
       ASSIGN     3,FN$LNGTH,PH   LENGTH OF BUFFER TRANSMISSION
       ASSIGN     4,FN$TYPE,PH    ASSIGN UPDATE OR READ
       ASSIGN     6,2,PF          PRIMARY/SECONDARY UPDATE CODE.INIT 2 OR READ
       ASSIGN     8-,1,PH         DECREMENT # OF REQUESTS MADE BY USER
       ASSIGN     13,FN$PRIOR,PF  ASSIGN THE REQUEST A PRIORITY I OR II
       PRIORITY   PF13            ASSIGN A PRICRITY TO THE  REQUEST
       ASSIGN     10,FN$RNDM2,PH  RANDOMLY SELECT A HOST CPU CHANNEL
       ASSIGN     10,FN$HOST,PF   ASSIGN THE REQUEST TO CPU OF HOST CPU-TERMIN.
       QUEUE      PH10
       SEIZE      PH10            TERMINAL - HOST CHANNEL
       DEPART     PH10            LEAVE QUEUE HOST-TERMINAL CHANNEL
```

```
         ADVANCE   V$TRNNL          LINE TRANSMISSION
         RELEASE   PH10             RELEASE THE HOST CPU-TERMINAL CHANNEL
         QUEUE     PF10             QUEUE FOR THE ASSIGNED HOST CPU
         SEIZE     PF10             SEIZE HOST CPU
         DEPART    PF10             LEAVE THE QUEUE
         ADVANCE   PH13             HINT
         ADVANCE   PH14             MESSAGE SYSTEM
         TEST E    PH4,1,*+2        IF UPDATE MARK PF6 A 1
         ASSIGN    6,1,PF
         ASSIGN    2,FN$RECRD,PF    MAKE RECORD ASSIGNMENT
         ASSIGN    3,C1,PF          MAKE CLOCK TIME ASSIGNMENT
         ASSIGN    5,1,PH
*
*              CHECK FOR A FREE MACHINE
*
LOCPA    RELEASE   PF10             FREE HOST CPU
         ASSIGN    14,V$HOBE,PF     SELECT A HOST-BE CHANNEL
         QUEUE     PF14             WAIT IN THE SELECTED HOST CPU-BE CHANNEL QUEU
         SEIZE     PF14             SEIZE HOST-BE CHANNEL
         DEPART    PF14             DEPART THE QUEUE
         ADVANCE   V$CHANL
         ADVANCE   1                TIME TAKEN TO CHECK IF FREE
         GATE SNE  PH5,BECB         GO TO BEOB IF BE FREE
         RELEASE   PF14             RETURN THE CHANNEL
         QUEUE     PF10
         SEIZE     PF10             SEIZE HOST CPU
         DEPART    PF10
         ASSIGN    5+,1,PH
         TEST LE   PH5,PH6,FULL     SEND TO FULL IF ALL BE'S ARE BUSY
         TRANSFER  ,LOCPA
FULL     ASSIGN    5,FN$RNDM,PH     RANDOMLY ASSIGN BE MACHINE
         ADVANCE   1                ASSIGNMENT TIME
         RELEASE   PF10             RELEASE HOST CPU
*              BACKEND HAS BEEN SELECTED
*
*
*
         ASSIGN    14,V$HOBE,PF     SELECT A HOST-BE CHANNEL
         QUEUE     PF14
         SEIZE     PF14
         DEPART    PF14
         ADVANCE   V$CHANL
BECB     RELEASE   PF14
         SAVEVALUE PH1,0,XF         INITIALIZE THE SAVEVAL USED IN VOTEA
         ASSIGN    1,V$PLUS,PF      PLACE VALUE OF VARIABLE IN PF1
         ADVANCE   PH13             HINT
         QUEUE     PH5              QUEUE FOR BE PARTITION
         ENTER     PH5              ENTER BE PARTITION
         DEPART    PH5
BEGIN    TEST E    PH4,1,*+2        CHK IF UPDATE
         PRIORITY  V$PRIOR
         SEIZE     PH5              SEIZE BE CPU
*
*              ENTER REQUESTS IN MODIFICATION TABLE
*
REQ1     SAVEVALUE PH5+,1,XH        INCREMENT # OF REQUESTS MADE
         ASSIGN    9,XH*PH5,PF      STORE LAST POINTER NUMBER IN PF9
         TEST NE   PH5,1,ONE        DETERMINE BE YOU ARE REQUESTING ON
         TEST NE   PH5,2,TWO        SO YOU CAN UPDATE APPROPRIATE TABLE.
         TEST NE   PH5,3,THREE
```

```
        TEST NE     PH5,4,FOUR
FOUR    MSAVEVALUE  4,XH4,1,PH2,MH  STORE THE NUMBER OF IO REQUESTS
        MSAVEVALUE  4,XH4,2,PF2,MH  STORE THE RECORD NUMBER
        MSAVEVALUE  4,XH4,1,PF3,MX  STORE THE ABSOLUTE CLOCK TIME OF UPDATE REQ
        MSAVEVALUE  4,PF9,1,1,MB     MARK REQUEST AS PENDING
        TRANSFER    ,SKIP1
THREE   MSAVEVALUE  3,XH3,1,PH2,MH  STORE THE NUMBER OF IO REQUESTS
        MSAVEVALUE  3,XH3,2,PF2,MH  STORE THE RECORD NUMBER
        MSAVEVALUE  3,XH3,1,PF3,MX  STORE THE ABSOLUTE CLOCK TIME OF UPDATE REQ
        MSAVEVALUE  3,PF9,1,1,MB     MARK REQUEST AS PENDING
        TRANSFER    ,SKIP1
TWO     MSAVEVALUE  2,XH2,1,PH2,MH  STORE THE NUMBER OF IO REQUESTS
        MSAVEVALUE  2,XH2,2,PF2,MH  STORE THE RECORD NUMBER
        MSAVEVALUE  2,XH2,1,PF3,MX  STORE THE ABSOLUTE CLOCK TIME OF UPDATE REQ
        MSAVEVALUE  2,PF9,1,1,MB     MARK REQUEST AS PENDING
        TRANSFER    ,SKIP1
ONE     MSAVEVALUE  1,XH1,1,PH2,MH  STORE THE NUMBER OF IO REQUESTS
        MSAVEVALUE  1,XH1,1,PF3,MX  STORE THE ABSOLUTE CLOCK TIME OF UPDATE REQ
        MSAVEVALUE  1,PF9,1,1,MB     MARK REQUEST AN PENDING
        TRANSFER    ,SKIP1
SKIP1   ADVANCE     1               TIME TAKEN TO WRITE TO TABLE
VOTE    TEST E      PH4,1,*+2       IF NOT AN UPDATE, SKIP 2 LINES
        TEST E      PF6,0,UPDT1     IF PRIMARY UPDATE GO TO UPDT1
LABL    ASSIGN      7,0,PH          PH7 IS A COUNTER
*
*               PROCESSING OF IO REQUESTS
*
LOOPB   TEST NE     PH2,PH7,FINIO   IF IO DONE GO TO FINIO
        TEST NE     PH4,1,*+2       IF UPDATE DON'T RELEASE DATABASE
        RELEASE     PH5             RETURN BE CPU
        QUEUE       FN$BEDEV
        SEIZE       FN$BEDEV        SEIZE BE DEVICE FOR IO
        DEPART      FN$BEDEV
        ADVANCE     52              TIME FOR EACH IO
        RELEASE     FN$BEDEV        RELEASE DEVICE
        TEST NE     PH4,1,*+2       IF UPDATE, BE IS ALREADY SEIZED
        SEIZE       PH5             SEIZE BE CPU
        ADVANCE     1               PROCESSING TIME
        ASSIGN      7+,1,PH         INCREMENT COUNTER
        TRANSFER    ,LOOPB
*
*               IO COMPLETED
*
FINIO   TEST NE     PF5,0,UPDT2     IF SECONDARY UPDATE GO TO UPDT2
        TEST E      PF6,1,READ1     IF NOT A PRIMARY UPDATE, SKIP AROUND
        SAVEVALUE   PF1+,1,XF       RELEASE THE SECONDARY MODIFICATIONS
*
*               RETURN INFORMATION TO THE HOST AND TERMINALS
*
READ1   MSAVEVALUE  PH5,PF9,1,0,MB  MARK REQUEST AS BEING COMPLETE
        ADVANCE     PH13            BINT
        RELEASE     PH5
        LEAVE       PH5             LEAVE PARTITION IN BE
        QUEUE       PF14            WAIT FOR HOST-BE CHANNEL
        SEIZE       PF14            SEIZE HOST - BE CHANNEL
        DEPART      PF14
        ADVANCE     V$CHANL
        ADVANCE     PH14            MESSAGE SYSTEM
        RELEASE     PF14
        QUEUE       PF10            QUEUE FOR HOST CPU
```

```
        PREEMPT     PF10,PR          PREEMPT THE HOST CPU
        DEPART      PF10
        ADVANCE     PH13             HINT
        RETURN      PF10
        QUEUE       PH10             WAIT FOR CPU-TERMINAL CHANNEL
        PREEMPT     PH10,PR
        DEPART      PH10
        ADVANCE     V$TRMNL
        RETURN      PH10
        TEST E      PH8,0,MORE       MORE REQUESTS BY THIS USER?
        RELEASE     PH1
        TERMINATE   1
*
*               ROUTINE CALLED FOR MODIFICATION REQUESTS
*
 UPDT1  TEST NE     PF6,1,SKIP5      SKIP SECTION IF  PRIMARY UPDATE
*
 UPDT2  MSAVEVALUE  PH15,XH*PH15,1,0,MB  MARK REQUEST AS COMPLETE
        LEAVE       PH5              LEAVE BE PARTITION
        RELEASE     PH5
        ASSEMBLE    V$MINUS          ASSEMBLE ALL SECONDARY UPDATES
        SAVEVALUE   PF1,0,XF
        TERMINATE   0
*
 SKIP5  ASSIGN      7,1,PH           A COUNTER
*
 SKIP6  TEST LE     PH7,PH6,VOTIN    IF COUNTR GT # OF BES, GO TO VOTEIN
        TEST NE     PH7,PH5,ELSE
        ASSIGN      15,PH7,PH        DETERMINE # OF BE TO BE UPDTED
        TRANSFER    ,*+3
 ELSE   ASSIGN      7+,1,PH
        TRANSFER    ,SKIP6
        ASSIGN      7+,1,PH          INCREMENT COUNTER
        SPLIT       1,*+2
        TRANSFER    ,SKIP6
*
        QUEUE       V$BECNL          QUEUE FOR BE TO BE CHANNEL
        SEIZE       V$BECNL          SEIZE THE BE TO BE CHANNEL
        DEPART      V$BECNL          LEAVE THE BE TO BE CHANNEL QUEUE
        ADVANCE     V$CHANL          TIME TAKEN TO SEND MESSAGE ACROSS THE CHANNEL
        RELEASE     V$BECNL          FREE THE BE TO BE CHANNEL
        ASSIGN      5,XH*PH15,PF     ASSIGN INTO PF5 # OF LAST ENTRY IN TABLE
*
*               SEARCH MODIFICATION TABLE
*
 LOOPM  TEST NE     PF5,0,VOTEA      IF THERE ARE NO ENTRIES  VOTE ACCEPT
        TEST L      MX*PH5(PF9,1),MX*PH15(PF5,1),VOTEA    TIMESTAMP LESS-VOTE ACPT
        TEST NE     MH*PH5(PF9,2),MH*PH15(PF5,2),THERE    IF RECRD # DIFFERENT-LOOP
        ASSIGN      5-,1,PF          DECREMENT TABLE POINTER
        TRANSFER    ,LOOPM
 THERE  TEST NE     MB*PH15(PF5,1),0,VOTEA   IF REQUEST PENDING - WAIT
        ADVANCE     5                TIME OUT 5 MSEC
        TRANSFER    ,THERE
*
 VOTEA  ADVANCE     10               VOTING TIME
        QUEUE       V$BECNL          QUEUE FOR BE TO BE CHANNEL
        SEIZE       V$BECNL          SEIZE THE BE TO BE CHANNEL
        DEPART      V$BECNL          LEAVE THE BE TO BE CHANNEL QUEUE
        ADVANCE     V$CHANL          TIME TAKEN TO SEND MESSAGE ACROSS THE CHANNEL
        RELEASE     V$BECNL          FREE THE BE TO BE CHANNEL
```

```
        SAVEVALUE   PH1+,1,XF       INCREMENT VOTE TALLYIER
        TEST E      XF*PH1,PH6      HOLD TILL ALL VOTES COUNTED
        ADVANCE     1               DELAY NEEDED TO ASSEMBLE THE REQUESTS
        SAVEVALUE   PF1+,1,XF       INCREMENT # OF SECONDARY MODS READY
        ADVANCE     10              TIME TAKEN TO 'WRITE' OK MESSAGE
        TEST E      XF*PF1,PH6      WAIT FOR PRIMARY MOD FINISHED(SEE FINIO)
        QUEUE       V$BECNL         QUEUE FOR BE TO BE CHANNEL
        SEIZE       V$BECNL         SEIZE THE BE TO BE CHANNEL
        DEPART      V$BECNL         LEAVE THE BE TO BE CHANNEL QUEUE
        ADVANCE     V$CHANL         TIME TAKEN TO SEND MESSAGE ACROSS THE CHANNEL
        RELEASE     V$BECNL         FREE THE BE TO BE CHANNEL
        ASSIGN      5,PH15,PH       ASSIGN BE # INTO PH5
        ASSIGN      6,0,PF          MARK AS SECONDARY UPDATE
        QUEUE       PH5             QUEUE FOR BE PARTITION
        ENTER       PH5             ENTER BE PARTITION
        DEPART      PH5
        TRANSFER    ,BEGIN          GO TO BEGIN AND BEGIN SECONDARY UPDATING
VOTIN   SAVEVALUE   PH1+,1,XF       INCREMENT VOTE TALLYIER
        TEST E      XF*PH1,PH6      HOLD TILL ALL VOTES IN
        TRANSFER    ,LABL
*
        START       30
        END
```

```
      REALLOCATE FAC,200,QUE,200,FSV,0,HSV,10,EVR,0,CHA,6,BSV,0
      REALLOCATE STO,10,VAR,10,FUN,15,TAB,0,BLC,170,LOG,90,COM,73620
      SIMULATE
*
*       CODASYL MODEL
*       MODEL OF 2 HOST CPU'S, 8 BE'S, 50% UPDATES, & 10% PRIORITY
*
** ENTITY DEFINITIONS
*
*  PARAMETERS
*
*
** ENTITY DEFINITIONS
*
*  PARAMETERS
*       PH1: TERMINAL ASSIGNMENT NUMBER
*       PF1: HALFWORD SAVEVALUE WHOSE NUMBER IS 10+PH5
*       PH2: NUMBER OF I/O REQUESTS NECESSARY FOR EACH USER RWQUEST
*       PH3: SIZE OF MESSAGE BUFFER
*       PH4: MARKED 0 IF READ REQUEST
*                   1 IF PRIMARY UPDATE REQUEST
*                   2 IF SECONDARY UPDATE REQUEST
*       PF4: MARKED 0 IF UPDATE REQUEST HAS NOT BEEN SPOOLED
*                   1 IF UPDATE REQUEST HAS BEEN SPOOLED
*       PH5: ID NUMBER OF THE BACKEND MACHINE BEING USED
*       PH6: NUMBER OF BACKEND MACHINES USED IN THE SIMULATION
*       PH7: A COUNTER
*       PF7: LAST TRANX NUMBER
*       PH8: NUMBER OF REQUESTS MADE BY THE USER
*       PH9: CHANNEL ID USED BETWEEN THE BACKEND MACHINES
*       PH10: ID NUMBER OF TERMINAL TO HOST CHANNEL
*       PF10: NUMBER OF HOST CPU
*       PF11: TRANSMISSION SPEED TO CHANNEL IN BITS/SEC
*       PF12: CHANNEL TRANSMISSION SPEED IN BITS/SEC
*       PH13: TIME USED BY HINT OR BINT
*       PF13: PRIORITY OF REQUESTS
*                   1 IF NORMAL REQUEST
*                   11 IF HIGH PRIORITY REQUEST
*       PH14: TIME USED BY THE MESSAGE SYSTEM
*       PF14: ID NUMBER OF HOST CPU-BE CHANNEL
*       PH15: ID NUMBER OF THE BE THAT THE SECONDARY UPDATE IS SENT TO
*       PF15: NUMBER OF HOST CPU'S IN THE SIMULATION MODEL
*
*  STORAGES REPRESENT PARTITIONS IN EACH BACKEND
*
*  FACILITIES
*
*       1 - 8: CPU'S OF THE EIGHT BACKEND'S
*       11,22,33,44,55,66,77,88 : CHANNELS FROM BACKEND TO DEVICE
*       12->18: CHANNELS FROM BE1 TO BE2,....,BE8
*       21,23....28: CHANNELS FROM BE2 TO OTHER BE'S,RESPECTIVELY
*       101->108: TERMINALS CONNECTED TO HOST
*  DEFINITIONS
*       UPDAT1: PRIMARY UPDATE
*       UPDAT2: SECONDARY UPDATE
*       LAST: ROW OF MATRIX LAST USED IN UPDATING
*       CURRENT: NUMBER OF UPDATES SPOOLED TO THAT BE
*
*
*  HALFWORD MATRIX
```

```
*         1:  MODIFICATION TABLE FOR PRIMARY BACKEND
*
 1      MATRIX      MH,400,1
*
*    STORAGES REPRESENT PARTITIONS IN EACH BACKEND
        STORAGE     S1-S8,2
 TRMNL  FVARIABLE   PH3*8/PF11*1000
 CHANL  FVARIABLE   PH3*8/PF12*1000
 LAST   VARIABLE    10*PH5
 INCRE  VARIABLE    PH7+1
 BECNL  VARIABLE    10*PH5+PH15
 HOBE   VARIABLE    PF10+PH5
 PRIOR  VARIABLE    PF13+2
*  50% UPDATE  ****************************************
 TYPE   FUNCTION    RN2,D2
.50,C/1.00,1
 TERM   FUNCTION    RN2,D8
.125,101/.25,102/.375,103/.5,104/.625,105/.75,106/.875,107/1.0,108
 RECRD  FUNCTION    RN2,D10
.10,1/.20,2/.30,3/.40,4/.50,5/.60,6/.70,7/.80,8/.90,9/1.0,10
 REQNO  FUNCTION    RN2,D10
.10,1/.20,2/.30,4/.40,5/.50,7/.60,9/.70,11/.80,12/.90,13/1.00,14
 LNGTH  FUNCTION    RN2,D2
.90,128/1.00,256
*  90% FIT IN ONE BUFFER
 BEDEV  FUNCTION    PH5,D8              BE TO DEVICE CHANNELS
1,11/2,22/3,33/4,44/5,55/6,66/7,77/8,88
 USREC  FUNCTION    RN2,D9              # OF REQUESTS BY A USER
.10,1/.20,3/.40,5/.50,7/.60,9/.70,11/.80,13/.90,15/1.00,17
*
 RNDM   FUNCTION    RN2,C8
.125,1/.25,2/.375,3/.5,4/.625,5/.75,6/.875,7/1.0,8
 RNDM2  FUNCTION    RN2,D4
.25,111/.50,111/.75,112/1.0,112
 HOST   FUNCTION    PH10,D4
111,150/112,160/113,170/114,180
 PRIOR  FUNCTION    RN2,D2
.9,1/1.0,11          **************   10% HIGH PRIORITY REQUESTS *********
*
        GENERATE    750,50,,30,,15PF,15PH GENERATE 30 TRANX 1 PER 3/4 SEC
        ASSIGN      13,FN$PRIOR,PF    ASSIGN THE REQUEST A PRIORITY 1 OR 11
        PRIORITY    PF13              ASSIGN A PRIORITY TO THE  REQUEST
        ASSIGN      15,2,PF           NUMBER OF HOST CPU'S IN THE SIMULATION
        ASSIGN      6,8,PH            NUMBER OF BACKENDS IN SIMULATION
        ASSIGN      1,FN$TERM,PH      ASSIGN TERMINAL # IN PH1
        ASSIGN      8,FN$USREQ,PH     STORE # OF USER'S COMMANDS IN PH8
        ASSIGN      11,50000,PF          TRANSMISSION SPEED TO CHANNEL
        ASSIGN      12,50000,PF          TRANSMISSION SPEED OF CHANNEL
        ASSIGN      13,5,PH           ADVANCE TIME FOR HINT OR BINT
        ASSIGN      14,5,PH           ADVANCE TIME FOR MESSAGE SYSTEM
        SAVEVALUE   9+,1,XH           INCREMENT USER COUNTER
        QUEUE       PH1               QUEUE FOR TERMINAL
        SEIZE       PH1               SEIZE THE TERMINAL
        SAVEVALUE   5+,1,XH
        TEST E      XH5,30,*+2        IF THIS IS THE LAST TRANSX, MARK IT
        ASSIGN      7,1,PF            MARK AS THE LAST TRANSACTION
        DEPART      PH1               DEPART QUEUE FOR TERMINAL
*
*             MAIN LOOP FOR USER REQUESTS
*
```

```
MORE   ADVANCE    1000,500          TIME TAKEN TO TYPE REQUEST
       ASSIGN     8-,1,PH           DECREMENT # OF REQUESTS MADE BY USER
       ASSIGN     2,FN$REQNO,PH     ASSIGN # OF IO REQUESTS IN PH 2
       ASSIGN     3,FN$LNGTH,PH     LENGTH OF BUFFER TRANSMISSION
       ASSIGN     4,FN$TYPE,PH      ASSIGN UPDATE OR READ
       ASSIGN     10,FN$RNUM2,PH    RANDOMLY SELECT A HOST CPU CHANNEL
       ASSIGN     10,FN$HOST,PF     ASSIGN THE REQUEST TO CPU OF HOST/CPU TERMINA
       QUEUE      PH10
       SEIZE      PH10              TERMINAL - HOST CHANNEL
       DEPART     PH10              LEAVE QUEUE HOST-TERMINAL CHANNEL
       ADVANCE    V$TRMNL           LINE TRANSMISSION
       ADVANCE    V$CHANL
       RELEASE    PH10              RELEASE THE HOST CPU-TERMINAL CHANNEL
       QUEUE      PF10              QUEUE FOR THE ASSIGNED HOST CPU
       SEIZE      PF10              SEIZE HOST CPU
       DEPART     PF10              LEAVE THE QUEUE
       ADVANCE    PH13              HINT
       ADVANCE    PH14              MESSAGE SYSTEM
       ASSIGN     5,1,PH
       TEST NE    PH4,1,ASIGN       IF UPDATE GO TO ASIGN FOR ASSIGNMENT
*
*        CHECK FOR A FREE MACHINE
*
LOOPA  RELEASE    PF10              FREE HOST CPU
       ASSIGN     14,V$HOBE,PF      SELECT A HOST-BE CHANNEL
       QUEUE      PF14              WAIT IN THE SELECTED HOST CPU-BE CHANNEL QUEU
       SEIZE      PF14              SEIZE HOST-BE CHANNEL
       DEPART     PF14              DEPART THE QUEUE
       ADVANCE    V$CHANL
       ADVANCE    1                 TIME TAKEN TO CHECK IF FREE
       GATE SNE   PH5,BEDB          GO TO BEDB IF BE FREE
       RELEASE    PF14              RETURN THE CHANNEL
       QUEUE      PF10
       SEIZE      PF10              SEIZE HOST CPU
       DEPART     PF10
       ASSIGN     5+,1,PH
       TEST LE    PH5,PH6,FULL      SEND TO FULL IF ALL BE'S ARE BUSY
       TRANSFER   ,LOOPA
FULL   ASSIGN     5,FN$RNUM,PH      RANDOMLY ASSIGN BE MACHINE
ASIGN  ADVANCE    1                 ASSIGNMENT TIME
       RELEASE    PF10              RELEASE HOST CPU
*            BACKEND HAS BEEN SELECTED
*
*
*
       ASSIGN     14,V$HOBE,PF      SELECT A HOST-BE CHANNEL
       QUEUE      PF14
       SEIZE      PF14
       DEPART     PF14
       ADVANCE    V$CHANL
BEDB   RELEASE    PF14
       ADVANCE    PH13              HINT
       QUEUE      PH5               QUEUE FOR BE PARTITION
       ENTER      PH5               ENTER BE PARTITION
       DEPART     PH5
       TEST E     PH4,1,*+2
       PRIORITY   V$PRIOR
       SEIZE      PH5               SEIZE BE CPU
       ASSIGN     7,0,PH            PH7 IS A COUNTER
*
```

```
*              PROCESSING OF IO REQUESTS
*
 LOOPB TEST NE     PH2,PH7,FINIO     IF IO DONE GO TO FINIO
       TEST NE     PH4,1,**2         IF UPDATE DON'T RELEASE DATABASE
       RELEASE     PH5               RETURN BE CPU
       QUEUE       FN$BEDEV
       SEIZE       FN$BEDEV          SEIZE BE DEVICE FOR IO
       DEPART      FN$BEDEV
       ADVANCE     52                TIME FOR EACH IO
       TEST E      PH4,1,**4         IS REQUEST A PRIMARY UPDATE?
       TEST NE     PF4,1,**3         HAS THIS UPDATE ALREADY BEEN SPOOLED?
       ADVANCE     30                IF NOT, SPOOL THE UPDATE
       ASSIGN      4,1,PF            MARK THE UPDATE AS SPOOLED
       RELEASE     FN$BEDEV          RELEASE DEVICE
       TEST NE     PH4,1,**2         IF UPDATE, BE IS ALREADY SEIZED
       SEIZE       PH5               SEIZE BE CPU
       ADVANCE     1                 PROCESSING TIME
       ASSIGN      7+,1,PH           INCREMENT COUNTER
       TRANSFER    ,LOOPB
*
*              IO COMPLETED
*
*        ENTER MODIFICATION REQUESTS IN THE MODIFICATION TABLE
*
 FINIO TEST NE     PH4,1,UPDT1       IF UPDATE, GO TO UPDT1-CHANGE TABLE
       TEST E      PH4,2,READ        IF QUERY ONLY COMMAND, GO TO READ
*
*        REQUEST IS A SECONDARY MODIFICATION
*
 SNDUP RELEASE     PH5
       LEAVE       PH5               RELEASE THE BE PARTITION
 FINUP TEST G      S*PH5,0,UPDT2     IF BE PARTITION EMPTY GO TO UPDATES
       TERMINATE   0
*
*              RETURN INFORMATION TO THE HOST AND TERMINALS
*
 READ  ADVANCE     PH13              HINT
       RELEASE     PH5
       LEAVE       PH5               LEAVE PARTITION IN BE
       QUEUE       PF14              WAIT FOR HOST-BE CHANNEL
       SEIZE       PF14              SEIZE HOST - BE CHANNEL
       DEPART      PF14
       ADVANCE     V$CHANL
       ADVANCE     PH14              MESSAGE SYSTEM
       RELEASE     PF14
       QUEUE       PF10              QUEUE FOR HOST CPU
       PREEMPT     PF10,PR           PREEMPT THE HOST CPU
       DEPART      PF10
       ADVANCE     PH13              HINT
       RETURN      PF10
       QUEUE       PH10              WAIT FOR CPU-TERMINAL CHANNEL
       PREEMPT     PH10,PR
       DEPART      PH10
       ADVANCE     V$TRMNL
       RETURN      PH10
       TEST E      PH8,0,MORE        MORE REQUESTS BY THIS USER?
       RELEASE     PH1               RELEASE OCCUPIED TERMINAL
*
*        IF LAST USER, BEGIN SECONDARY UPDATES
*
```

```
         TEST E   PF7,1,END1
         TRANSFER   ,UPT2            IF LAST TRANX, SEND SECONDARY MOD
 END1    TERMINATE  1
 END2    TERMINATE  0
*
*          ADD REQUEST TO MODIFICATION TABLE
*
 UPDT1   SAVEVALUE  PH5+,1,XH         INCREMENT # OF UPDATES SPOOLED
 ONE     MSAVEVALUE 1,XH1,1,PH2,MH    STORE THE NUMBER OF IO REQUESTS
 SKIP1   ADVANCE    1                 TIME TAKEN TO WRITE TO TABLE
         TRANSFER   ,READ
*
*               ROUTINE CALLED FOR SECONDARY MODIFICATION REQUESTS
*
 UPT2    SAVEVALUE  2+,1,XH
         ASSIGN     7,0,PF
 UPDT2   ASSIGN     5,1,PH
         TEST L     XH2,XH*PH5,END2   CONTINUE IF FENCING UPDATES
         ASSIGN     1,XH2,PF          ASSIGN UPDATE TABLE ROW # INTO PF1
         SAVEVALUE  2+,1,XH           INCREMENT UPDT LIST COUNTER
         ASSIGN     7,1,PH
*
*          SEND THE SECONDARY MODIFICATIONS
*
 LOOPC   TEST LE    PH7,PH6,UPDT2     IF COUNTER GT # OF BE GO TO FINLP
         TEST NE    PH7,PH5,ELSE      GO TO ELSE IF CNTR = # OF BE
         ASSIGN     15,PH7,PH         DETERMINE # OF BE TO BE UPDATED
         TRANSFER   ,SKIP2
 ELSE    ASSIGN     7+,1,PH           INCREMENT COUNTER
         TRANSFER   ,LOOPC
 SKIP2   ASSIGN     9,V$BECNL,PH
         ASSIGN     7+,1,PH           INCREMENT COUNTER
         ADVANCE    4000              4000 MSEC PAUSE BETWEEN SENDING OF UPDATES
         SPLIT      1,LOOPC           SEND OUT ANOTHER UPDATE TO NEXT BE
*
         QUEUE      PH5
         ENTER      PH5
         DEPART     PH5
         SEIZE      PH5
         ADVANCE    PH13 .            BINT
         RELEASE    PH5
         LEAVE      PH5
         QUEUE      PH9               QUEUE FOR CHANNEL BETWEEN 2 BE'S
         SEIZE      PH9               BE TO BE CHANNEL
         DEPART     PH9
         ADVANCE    V$CHANL
         ADVANCE    PH14              MESSAGE SYSTEM
         RELEASE    PH9
         QUEUE      PH15
         ENTER      PH15
         DEPART     PH15
         SEIZE      PH15
         ADVANCE    PH13              BINT ON NEXT BE
         ASSIGN     4,2,PH            CODE UPDATE AS SECONDARY
         ASSIGN     7,0,PH            COUNTER
         ASSIGN     2,MH*PH5(PF1,1),PH   ASSIGN INTO PH2 THE # OF IO REQ
         ASSIGN     5,PH15,PH         ASSIGN PH15 INTO PH5
         TRANSFER   ,LOOPB
         NOXREF
         END
```

```
      REALLOCATE LSV,0,CRP,C,FMS,0,HMS,1,BMS,C,LMS,C
      REALLOCATE FAC,200,QUE,200,BLC,250
      REALLOCATE STO,10,XAC,200,VAR,10,FUN,15,TAB,0,CCM,34080
      SIMULATE
*
*       ELLIS MODEL
*       MODEL OF 2 HOST CPU'S, 8 BE'S, 50% UPDATES, & 10% PRIORITY
** ENTITY DEFINITIONS
*
*  PARAMETERS
*
*       PH1: TERMINAL ASSIGNMENT NUMBER
*       PF1: HALFWORD SAVEVALUE WHOSE NUMBER IS 10+PH5
*       PH2: NUMBER OF I/O REQUESTS NECESSARY FOR EACH USER RWQUEST
*       PF2: MODIFICATION REQUEST NUMBER
*       PH3: SIZE OF MESSAGE BUFFER
*       PH4: MARKED 0 IF READ REQUEST
*                   1 IF PRIMARY UPDATE REQUEST
*                   2 IF SECONDARY UPDATE REQUEST
*       PF4: MARKED 0 IF UPDATE REQUEST HAS NOT BEEN SPOOLED
*                   1 IF UPDATE REQUEST HAS BEEN SPOOLED
*       PH5: ID NUMBER OF THE BACKEND MACHINE BEING USED
*       PH6: NUMBER OF BACKEND MACHINES USED IN THE SIMULATION
*       PH7: A COUNTER
*       PH8: NUMBER OF REQUESTS MADE BY THE USER
*       PH9: CHANNEL ID USED BETWEEN TWO BACKEND MACHINES
*       PH10: ID NUMBER OF TERMINAL TO HOST CHANNEL
*       PF10: NUMBER OF HOST CPU
*       PF11: TRANSMISSION SPEED TO CHANNEL IN BITS/SEC
*       PF12: CHANNEL TRANSMISSION SPEED IN BITS/SEC
*       PH13: TIME USED BY HINT OR BINT
*       PF13: PRIORITY OF REQUESTS
*                   1 IF NORMAL REQUEST
*                   10 IF HIGH PRIORITY REQUEST
*       PH14: TIME USED BY THE MESSAGE SYSTEM
*       PF14: ID NUMBER OF HOST CPU-BE CHANNEL
*       PH15: ID NUMBER OF THE BE THAT THE SECONDARY UPDATE IS SENT TO
*       PF15: NUMBER OF HOST CPU'S IN THE SIMULATION MODEL
*
*  FACILITIES
*
*       1 - 8: CPU'S OF THE EIGHT BACKEND'S
*       11,22,33,44,55,66,77,88 : CHANNELS FROM BACKEND TO DEVICE
*       12->18: CHANNELS FROM BE1 TO BE2,...,BE8
*       21,23,...28: CHANNELS FROM BE2 TO OTHER BE'S,RESPECTIVELY
*       101->108: TERMINALS CONNECTED TO HOST
*
*  HALFWORD SAVEVALUES
*       XH1:  NUMBER OF MODIFICATION REQUSTS ACCEPTED BY HOST
*
*  DEFINITIONS
*       UPDAT1: PRIMARY UPDATE
*       UPDAT2: SECONDARY UPDATE
*       LAST: ROW OF MATRIX LAST USED IN UPDATING
*       CURRENT: NUMBER OF UPDATES SPOOLED TO THAT BE
*
*  STORAGES REPRESENT PARTITIONS IN EACH BACKEND
*
     STORAGE    S1-S8,2
*
```

```
 PLUS  VARIABLE   10+PH1
 MINUS VARIABLE   PH6-1
 TRMNL FVARIABLE  PH3*8/PF11*1000
 CHANL FVARIABLE  PH3*8/PF12*1000
 LAST  VARIABLE   10+PH5
 INCRE VARIABLE   PH7+1
 HOBE  VARIABLE   PF10+PH5
 BECNL VARIABLE   10*PH5+PH15
 PRIOR VARIABLE   PF13+2
*  50% UPDATE  ************************************
 TYPE  FUNCTION   RN2,D2
.50,0/1.00,1
 TERM  FUNCTION   RN2,D8
.125,101/.25,102/.375,103/.5,104/.625,105/.75,106/.875,107/1.0,108
 REQNO FUNCTION   RN2,D10
.10,1/.20,2/.30,4/.40,5/.50,7/.60,9/.70,11/.80,12/.90,13/1.00,14
 LNGTH FUNCTION   RN2,D2
.90,128/1.00,256
*  90% FIT IN ONE BUFFER
 BEDEV FUNCTION   PH5,D8              BE TO DEVICE CHANNELS
1,11/2,22/3,33/4,44/5,55/6,66/7,77/8,88
 USREC FUNCTION   RN2,D9              # OF REQUESTS BY A USER
.10,1/.20,3/.40,5/.50,7/.60,9/.70,11/.80,13/.90,15/1.00,17
*
 RNDM  FUNCTION   RN2,D8
.125,1/.25,2/.375,3/.5,4/.625,5/.75,6/.875,7/1.0,8
 RNDM2 FUNCTION   RN2,D4
.25,111/.50,111/.75,112/1.0,112
 HOST  FUNCTION   PH10,D4
111,150/112,160/113,170/114,180
 PRIOR FUNCTION   RN2,D2
.9,1/1.0,11          **************   10% HIGH PRIORITY REQUESTS *********
*
       INITIAL    XH1-XH114,0      INITIALIZE SAVEVALUES TO 0
*
       GENERATE   750,50,,30,,15PF,15PH   GENERATE 30 TRANSX-AVE 1/.75 SEC
       ASSIGN     13,FN$PRIOR,PF    ASSIGN THE REQUEST A PRIORITY 1 OR 11
       PRIORITY   PF13              ASSIGN A PRICRITY TO THE  REQUEST
       ASSIGN     1,FN$TERM,PH      ASSIGN TERMINAL # IN PH1
       ASSIGN     15,2,PF           NUMBER OF HOST CPU'S IN THE SIMULATION
       ASSIGN     1,FN$TERM,PH      ASSIGN TERMINAL # IN PH1
       ASSIGN     6,8,PH            NUMBER OF BACKENDS IN SIMULATION
       ASSIGN     8,FN$USREQ,PH     STORE # OF USER'S COMMANDS IN PH8
       ASSIGN     11,50000,PF          TRANSMISSION SPEED TO CHANNEL
       ASSIGN     12,50000,PF          TRANSMISSION SPEED OF CHANNEL
       ASSIGN     13,5,PH           ADVANCE TIME FOR HINT OR BINT
       ASSIGN     14,5,PH           ADVANCE TIME FOR MESSAGE SYSTEM
       QUEUE      PH1               QUEUE FOR TERMINAL
       SEIZE      PH1               SEIZE THE TERMINAL
       DEPART     PH1               DEPART QUEUE FOR TERMINAL
*
*             MAIN LOOP FOR USER REQUESTS
*
 MORE  ADVANCE    1000,500          TIME TAKEN TO TYPE REQUEST
       ASSIGN     2,FN$REQNO,PH     ASSIGN # OF IO REQUESTS IN PH 2
       ASSIGN     3,FN$LNGTH,PH     LENGTH OF BUFFER TRANSMISSION
       ASSIGN     4,FN$TYPE,PH      ASSIGN UPDATE OR READ
       ASSIGN     8-,1,PH           DECREMENT # OF REQUESTS MADE BY USER
       ASSIGN     10,FN$RNDM2,PH    RANDOMLY SELECT A HOST CPU CHANNEL
       ASSIGN     10,FN$HOST,PF     ASSIGN THE REQUEST TO CPU OF HOST/CPU TERMINA
```

```
         QUEUE     PH10
         SEIZE     PH10              TERMINAL - HOST CHANNEL
         DEPART    PH10              LEAVE QUEUE HOST-TERMINAL CHANNEL
         ADVANCE   V$TRMNL           LINE TRANSMISSION
         RELEASE   PH10              RELEASE THE HOST CPU-TERMINAL CHANNEL
         QUEUE     PF10              QUEUE FOR THE ASSIGNED HOST CPU
         SEIZE     PF10              SEIZE HOST CPU
         DEPART    PF10              LEAVE THE QUEUE
         ADVANCE   PH13              HINT
         ADVANCE   PH14              MESSAGE SYSTEM
         TEST NE   PH4,1,UPDT        IF UPDATE GO TO UPDT FOR ASSIGNMENT
         ASSIGN    5,1,PH
*
*          CHECK FOR A FREE MACHINE
*
LOOPA    RELEASE   PF10              FREE HOST CPU
         ASSIGN    14,V$HCBE,PF      SELECT A HOST-BE CHANNEL
         QUEUE     PF14              WAIT IN THE SELECTED HOST CPU-BE CHANNEL QUEU
         SEIZE     PF14              SEIZE HOST-BE CHANNEL
         DEPART    PF14              DEPART THE QUEUE
         ADVANCE   V$CHANL
         ADVANCE   1                 TIME TAKEN TO CHECK IF FREE
         GATE SNE  PH5,BECB          GO TO BECB IF BE FREE
         RELEASE   PF14              RETURN THE CHANNEL
         QUEUE     PF10
         SEIZE     PF10              SEIZE HOST CPU
         DEPART    PF10
         ASSIGN    5+,1,PH
         TEST LE   PH5,PH6,FULL      SEND TO FULL IF ALL BE'S ARE BUSY
         TRANSFER  ,LOOPA
FULL     ASSIGN    5,FN$RNDM,PH      RANDOMLY ASSIGN BE MACHINE
         ADVANCE   1                 ASSIGNMENT TIME
         RELEASE   PF10              RELEASE HOST CPU
*
*               BACKEND HAS BEEN SELECTED
*
ENTER    ASSIGN    14,V$HCBE,PF      SELECT A HOST-BE CHANNEL
         QUEUE     PF14
         SEIZE     PF14
         DEPART    PF14
         ADVANCE   V$CHANL
BECB     RELEASE   PF14
         ADVANCE   PH13              BINT
         QUEUE     PH5               QUEUE FOR BE PARTITION
         ENTER     PH5               ENTER BE PARTITION
         DEPART    PH5
         SEIZE     PH5               SEIZE THE BE CPU
         ASSIGN    7,0,PH            PH7 IS A COUNTER
*
*               PROCESSING OF IO REQUESTS
*
LOOPB    TEST NE   PH2,PH7,READ      IF IO DONE GO TO READ
         TEST NE   PH4,1,**2         IF UPDATE DON'T RELEASE DATABASE
         RELEASE   PH5               RETURN BE CPU
         QUEUE     FN$BEDEV
         SEIZE     FN$BEDEV          SEIZE BE DEVICE FOR IO
         DEPART    FN$BEDEV
         ADVANCE   52                TIME FOR EACH IO
         TEST E    PH4,1,**4         IS REQUEST A PRIMARY UPDATE?
         TEST NE   PF4,1,**3         HAS THIS UPDATE ALREADY BEEN SPOOLED?
```

```
         ADVANCE    30                IF NOT, SPOOL THE UPDATE
         ASSIGN     4,1,PF            MARK THE UPDATE AS SPOOLED
         RELEASE    FN$BEDEV          RELEASE DEVICE
         TEST NE    PH4,1,*+2
         SEIZE      PH5               SEIZE THE BE CPU
         ADVANCE    1                 PROCESSING TIME
         ASSIGN     7+,1,PH           INCREMENT COUNTER
         TRANSFER   ,LOOPB
*
*
*              IC COMPLETED
*
*
*              RETURN INFORMATION TO THE HOST AND TERMINALS
*
 READ    ADVANCE    PH13              HINT
         RELEASE    PH5
         LEAVE      PH5               LEAVE PARTITION IN BE
         QUEUE      PF14              WAIT FOR HOST-BE CHANNEL
         SEIZE      PF14              SEIZE HOST - BE CHANNEL
         DEPART     PF14
         ADVANCE    V$CHANL
         ADVANCE    PH15              MESSAGE SYSTEM
         RELEASE    PF14
         TEST E     PH4,1,*+3
         LOGIC R    PF2               RESET THE LOGIC SWITCH TO FREE HELD TRANSX
         TRANSFER   ,ENDIT
 RETRN   QUEUE      PF10              QUEUE FOR HOST CPU
         PREEMPT    PF10,PR           PREEMPT THE HOST CPU
         DEPART     PF10
         ADVANCE    PH13              HINT
         RETURN     PF10
         QUEUE      PH10              WAIT FOR CPU-TERMINAL CHANNEL
         PREEMPT    PH10,PR
         DEPART     PH10
         ADVANCE    V$TRNNL
         RETURN     PH10
         TEST E     PH8,0,MORE        MORE REQUESTS BY THIS USER?
         RELEASE    PH1               RELEASE OCCUPIED TERMINAL
         TERMINATE  1
 ENDIT   TERMINATE  0
*
*
*              ROUTINE CALLED FOR MODIFICATION REQUESTS
*
 UPDT    ASSIGN     7,1,PH
         SAVEVALUE  PH1C+,1,XH        INCREMENT XH - UPDATE CNTR FOR HOST
         ASSIGN     2,XH*PH1C,PF      ASSIGN UPDATE # TO PF2
         LOGIC S    PF2               SET LOGIC SWITCH PF2
 LOOPC   TEST LE    PH7,PH6,WAIT      IF COUNTER=# OF BE GO TO WAIT
         ASSIGN     5,PH7,PH          DETERMINE # OF BE TO BE UPDATED
         ASSIGN     7+,1,PF           INCREMENT COUNTER
         SPLIT      1,ENTER           SEND OUT TRANSX COPY TO BE UPDATED
         TRANSFER   ,LOOPC            SEND THE PARENT TRANSX TO LOOPC
*
 WAIT    RELEASE    PF10              RELEASE THE HOST PROCESSOR
         GATE LR    PF2               WAIT UNTIL LOGIC SWITCH PF2 IS RESET
         TRANSFER   ,RETRN            GO TO RETRN WHEN THE MODIFICATION REQUEST
*                                     HAS BEEN COMPLETED BY ONE OF THE BACKENDS
*
         NOXREF
         END
```

```
      REALLOCATE FAC,200,QUE,200,STO,10,LOG,200,FUN,25,TAB,0,COM,21000
      SIMULATE
*
*       HYBRID MODEL
*       MODEL OF 4 HOST CPU'S, 8 BE'S, 50% UPDATES, & 10% PRIORITY
* **   ENTITY DEFINITIONS
*
*   PARAMETERS
*
*        PH1: TERMINAL ASSIGNMENT NUMBER
*        PF1: HALFWORD SAVEVALUE WHOSE NUMBER IS 10+PH5
*        PH2: NUMBER OF I/O REQUESTS NECESSARY FOR EACH USER RWQUEST
*        PH3: SIZE OF MESSAGE BUFFER
*        PH4: MARKED 0 IF READ REQUEST
*                     1 IF PRIMARY UPDATE REQUEST
*                     2 IF SECONDARY UPDATE REQUEST
*        PF4: MARKED 0 IF UPDATE REQUEST HAS NOT BEEN SPOOLED
*                     1 IF UPDATE REQUEST HAS BEEN SPOOLED
*        PH5: ID NUMBER OF THE BACKEND MACHINE BEING USED
*        PH6: NUMBER OF BACKEND MACHINES USED IN THE SIMULATION
*        PH7: A COUNTER
*        PH8: NUMBER OF REQUESTS MADE BY THE USER
*        PH9: CHANNEL ID USED BETWEEN TWO BACKEND MACHINES
*        PH10: ID NUMBER OF TERMINAL TO HOST CHANNEL
*        PF10: NUMBER OF HOST CPU
*        PF11: TRANSMISSION SPEED TO CHANNEL IN BITS/SEC
*        PF12: CHANNEL TRANSMISSION SPEED IN BITS/SEC
*        PH13: TIME USED BY HINT OR BINT
*        PF13: PRIORITY OF REQUESTS
*                     1 IF NORMAL REQUEST
*                     10 IF HIGH PRIORITY REQUEST
*        PH14: TIME USED BY THE MESSAGE SYSTEM
*        PF14: ID NUMBER OF HOST CPU-BE CHANNEL
*        PH15: ID NUMBER OF THE BE THAT THE SECONDARY UPDATE IS SENT TO
*        PF15: NUMBER OF HOST CPU'S IN THE SIMULATION MODEL
*   FACILITIES
*
*   HALFWORD SAVEVALUES
*        XH1->XH8: CURRENT TABLE POINTERS FOR BE1->BE8
*        XH11->XH18: LAST TABLE POINTERS FOR BE1->BE8
*
*   DEFINITIONS
*        UPDAT1: PRIMARY UPDATE
*        UPDAT2: SECONDARY UPDATE
*        LAST: ROW OF MATRIX LAST USED IN UPDATING
*        CURRENT: NUMBER OF UPDATES SPOOLED TO THAT BE
*
*
*   FACILITIES
*
*        1 - 8: CPU'S OF THE EIGHT BACKEND'S
*        11,22,33,44,55,66,77,88 : CHANNELS FROM BACKEND TO DEVICE
*        12->18: CHANNELS FROM BE1 TO BE2,....,BE8
*        21,23,...28: CHANNELS FROM BE2 TO OTHER BE'S,RESPECTIVELY
*
*   HALFWORD SAVEVALUES
*        XH1->XH8: CURRENT TABLE POINTERS FOR BE1->BE8
*        XH11->XH18: LAST TABLE POINTERS FOR BE1->BE8
*
*   DEFINITIONS
```

```
*         UPDAT1: PRIMARY UPDATE
*         UPDAT2: SECONDARY UPDATE
*         LAST: ROW OF MATRIX LAST USED IN UPDATING
*         CURRENT: NUMBER OF UPDATES SPOOLED TO THAT BE
*
*     FULLWORD,HALFWORD, AND BYTE MATRICES
*         1: UPDATE TABLE FOR BE1
*         2: UPDATE TABLE FOR BE2
*         3: UPDATE TABLE FOR BE3
*         4: UPDATE TABLE FOR BE4
*         5: UPDATE TABLE FOR BE5
*         6: UPDATE TABLE FOR BE6
*         7: UPDATE TABLE FOR BE7
*         8: UPDATE TABLE FOR BE8
*
*
*
 1      MATRIX      MH,200,1
 2      MATRIX      MH,200,1
 3      MATRIX      MH,200,1
 4      MATRIX      MH,200,1
 5      MATRIX      MH,200,1
 6      MATRIX      MH,200,1
 7      MATRIX      MH,200,1
 8      MATRIX      MH,200,1
*
*    STORAGES REPRESENT PARTITIONS IN EACH BACKEND
*
        STORAGE     S1-S8,2
 MINUS  VARIABLE    PH6-1
 PLUS   VARIABLE    10+PH1
 TRMNL  FVARIABLE   PH3*8/PF11*1000
 CHANL  FVARIABLE   PH3*8/PF12*1000
 LAST   VARIABLE    10+PH5
 INCRE  VARIABLE    PH7+1
 BECNL  VARIABLE    10*PH5+PH15
 HOBE   VARIABLE    PF10+PH5
 PRIOR  VARIABLE    PF13+2
*  50% UPDATE  ****************************************
 TYPE   FUNCTION    RN2,D2
.50,0/1.00,1
 TERM   FUNCTION    RN2,D8
.125,101/.25,102/.375,103/.5,104/.625,105/.75,106/.875,107/1.0,108
 RECNO  FUNCTION    RN2,D10
.10,1/.20,2/.30,4/.40,5/.50,7/.60,9/.70,11/.80,12/.90,13/1.00,14
 LNGTH  FUNCTION    RN2,D2
.90,128/1.00,256
*  90% FIT IN ONE BUFFER
 USREC  FUNCTION    RN2,D9              # OF REQUESTS BY A USER
.10,1/.20,3/.40,5/.50,7/.60,9/.70,11/.80,13/.90,15/1.00,17
 BEDEV  FUNCTION    PH5,D8              BE TO DEVICE CHANNELS
1,11/2,22/3,33/4,44/5,55/6,66/7,77/8,88
 RNDM   FUNCTION    RN2,D8
.125,1/.25,2/.375,3/.5,4/.625,5/.75,6/.875,7/1.0,8
 RNDM2  FUNCTION    RN2,D4
.25,111/.50,112/.75,113/1.0,114
 HOST   FUNCTION    PH10,04
111,150/112,160/113,170/114,180
 PRIOR  FUNCTION    RN2,D2
.9,1/1.0,11          ***************   10% HIGH PRIORITY REQUESTS *********
```

```
         INITIAL    XH1-XH8,C        INITIALIZE SAVEVALUES 1-10 TO ZERO
         INITIAL    XH11-XH18,0      INITIALIZE SAVEVALUES 11-18 TO ZERO
*
         GENERATE   375,50,,60,,15PF,15PH   GENERATE 60 TRANSX-AVE 3/8 SEC
         ASSIGN     13,FN$PRIOR,PF   ASSIGN THE REQUEST A PRIORITY 1 OR 11
         PRIORITY   PF13             ASSIGN A PRICRITY TO THE  REQUEST
         ASSIGN     15,4,PF          NUMBER OF HOST CPU'S IN THE SIMULATION
         ASSIGN     6,8,PH           NUMBER OF BACKENDS IN SIMULATION
         ASSIGN     1,FN$TERM,PH     ASSIGN TERMINAL # IN PH1
         ASSIGN     8,FN$USREQ,PH    STORE # OF USER'S COMMANDS IN PH8
         ASSIGN     11,50000,PF      TRANSMISSION SPEED TO CHANNEL
         ASSIGN     12,50000,PF      TRANSMISSION SPEED OF CHANNEL
         ASSIGN     13,5,PH          ADVANCE TIME FOR HINT OR BINT
         ASSIGN     14,5,PH          ADVANCE TIME FOR MESSAGE SYSTEM
         QUEUE      PH1              QUEUE FOR TERMINAL
         SEIZE      PH1              SEIZE THE TERMINAL
         DEPART     PH1              DEPART QUEUE FOR TERMINAL
*
*              MAIN LOOP FOR USER REQUESTS
*
MORE     ADVANCE    1000,500         TIME TAKEN TO TYPE REQUEST
         ASSIGN     2,FN$RECNO,PH    ASSIGN # OF IO REQUESTS IN PH 2
         ASSIGN     3,FN$LNGTH,PH    LENGTH OF BUFFER TRANSMISSION
         ASSIGN     4,FN$TYPE,PH     ASSIGN UPDATE OR READ
         ASSIGN     8-,1,PH          DECREMENT # OF REQUESTS MADE BY USER
         ASSIGN     10,FN$RNDM2,PH   RANDOMLY SELECT A HOST CPU CHANNEL
         ASSIGN     10,FN$HOST,PF    ASSIGN THE REQUEST TO CPU OF HOST/CPU TERMINA
         QUEUE      PH10
         SEIZE      PH10             TERMINAL - HOST CHANNEL
         DEPART     PH10             LEAVE QUEUE HOST-TERMINAL CHANNEL
         ADVANCE    V$TRMNL          LINE TRANSMISSION
         RELEASE    PH10             RELEASE THE HOST CPU-TERMINAL CHANNEL
         QUEUE      PF10             QUEUE FOR THE ASSIGNED HOST CPU
         SEIZE      PF10             SEIZE HOST CPU
         DEPART     PF10             LEAVE THE QUEUE
         ADVANCE    PH13             HINT
         ADVANCE    PH14             MESSAGE SYSTEM
         ASSIGN     5,1,PH
         TEST NE    PH4,1,FULL       IF UPDATE GO TO FULL FOR ASSIGNMENT
*
*              CHECK FOR A FREE MACHINE
*
LOOPA    RELEASE    PF10             FREE HOST CPU
         ASSIGN     14,V$HCBE,PF     SELECT A HOST-BE CHANNEL
         QUEUE      PF14             WAIT IN THE SELECTED HOST CPU-BE CHANNEL QUEU
         SEIZE      PF14             SEIZE HOST-BE CHANNEL
         DEPART     PF14             DEPART THE QUEUE
         ADVANCE    V$CHANL
         ADVANCE    1                TIME TAKEN TO CHECK IF FREE
         GATE SNE   PH5,BECB         GO TO BECB IF BE FREE
         RELEASE    PF14             RETURN THE CHANNEL
         QUEUE      PF10
         SEIZE      PF10             SEIZE HOST CPU
         DEPART     PF10
         ASSIGN     5+,1,PH
         TEST LE    PH5,PH6,FULL     SEND TO FULL IF ALL BE'S ARE BUSY
         TRANSFER   ,LOOPA
FULL     ASSIGN     5,FN$RNDM,PH     RANDOMLY ASSIGN BE MACHINE
ASIGN    ADVANCE    1                ASSIGNMENT TIME
         RELEASE    PF10             RELEASE THE HOST PROCESSOR
```

```
*
*              PACKEND HAS BEEN SELECTED
*
        ASSIGN      14,V$HOBE,PF      SELECT A HOST-BE CHANNEL
        QUEUE       PF14
        SEIZE       PF14
        DEPART      PF14
        ADVANCE     V$CHANL
 BEDB   RELEASE     PF14
        ADVANCE     PH13              BINT
        QUEUE       PH5               QUEUE FOR BE PARTITION
        ENTER       PH5               ENTER BE PARTITION
        DEPART      PH5
        SEIZE       PH5               SEIZE BE CPU
        ASSIGN      7,0,PH            PH7 IS A COUNTER
*
*              PROCESSING OF IO REQUESTS
*
 LOOPB  TEST NE     PH2,PH7,FINIO     IF IO DONE GO TO FINIO
        TEST NE     PH4,1,*+2
        RELEASE     PH5
        QUEUE       FN$BEDEV
        SEIZE       FN$BEDEV          SEIZE BE DEVICE FOR IO
        DEPART      FN$BEDEV
        ADVANCE     52                TIME FOR EACH IO
        TEST E      PH4,1,*+4         IS REQUEST A PRIMARY UPDATE?
        TEST NE     PF4,1,*+3         HAS THIS UPDATE ALREADY BEEN SPOOLED?
        ADVANCE     30                IF NOT, SPOOL THE UPDATE
        ASSIGN      4,1,PF            MARK THE UPDATE AS SPOOLED
        RELEASE     FN$BEDEV          RELEASE DEVICE
        TEST NE     PH4,1,*+2
        SEIZE       PH5               SEIZE THE BE CPU
        ADVANCE     1                 PROCESSING TIME
        ASSIGN      7+,1,PH           INCREMENT COUNTER
        TRANSFER    ,LOOPB
*
*              IO COMPLETED
*
*
*
*              ENTER MODIFICATION REQUESTS IN MOD. TABLES
*
 FINIO  TEST NE     PH4,1,UPDT1       IF UPDATE, GO TO UPDT1-CHANGE TABLE
        TEST E      PH4,2,READ        IF QUERY ONLY COMMAND, GO TO READ
*
*              REQUEST IS A SECONDARY UPDATE
*
 RECPU  RELEASE     PH5
        LEAVE       PH5               RELEASE THE BE PARTITION
 FINUP  TEST G      S*PH5,1,UPDT2     IF BE PARTITION EMPTY GO TO UPDATES
        TRANSFER    ,ENDIT            ELSE END TRANSACTION
*
*              RETURN INFORMATION TO THE HOST AND TERMINALS
*
 READ   ADVANCE     PH13              BINT
        RELEASE     PH5
        LEAVE       PH5               LEAVE PARTITION IN BE
        TEST E      S*PH5,0,*+2       IF PARTITIONS EMPTY, SPLIT
        SPLIT       1,UPDT2           SPLIT AND SEND 1 TO UPDT2
        QUEUE       PF14              WAIT FOR HOST-BE CHANNEL
```

```
        SEIZE       PF14              SEIZE HOST - BE CHANNEL
        DEPART      PF14
        ADVANCE     V$CHANL
        ADVANCE     PH14              MESSAGE SYSTEM
        RELEASE     PF14
        QUEUE       PF10              QUEUE FOR HOST CPU
        PREEMPT     PF10,PR           PREEMPT THE HOST CPU
        DEPART      PF10
        ADVANCE     PH13              HINT
        RETURN      PF10
        QUEUE       PH10              WAIT FOR CPU-TERMINAL CHANNEL
        PREEMPT     PH10,PR
        DEPART      PH10
        ADVANCE     V$TRMNL
        RETURN      PH10
        TEST E      PH8,0,MORE        MORE REQUESTS BY THIS USER?
        RELEASE     PH1               RELEASE OCCUPIED TERMINAL
 ENDIT  TERMINATE   1
*
*               ADD REQUEST TO MODIFICATION TABLE
*
 UPDT1  SAVEVALUE   PH5+,1,XH         INCREMENT # OF UPDATES SPECIED
        TEST NE     PH5,1,ONE         DETERMINE BE YOU ARE UPDATING ON
        TEST NE     PH5,2,TWO         SO YOU CAN UPDATE APPROPRIATE TABLE.
        TEST NE     PH5,3,THREE
        TEST NE     PH5,4,FOUR
        TEST NE     PH5,5,FIVE
        TEST NE     PH5,6,SIX
        TEST NE     PH5,7,SEVEN
        TEST NE     PH5,8,EIGHT
 EIGHT  MSAVEVALUE  8,XH8,1,PH2,MH    STORE THE NUMBER OF IO REQUESTS
        TRANSFER    ,SKIP1
 SEVEN  MSAVEVALUE  7,XH7,1,PH2,MH    STORE THE NUMBER OF IO REQUESTS
        TRANSFER    ,SKIP1
 SIX    MSAVEVALUE  6,XH6,1,PH2,MH    STORE THE NUMBER OF IO REQUESTS
        TRANSFER    ,SKIP1
 FIVE   MSAVEVALUE  5,XH5,1,PH2,MH    STORE THE NUMBER OF IO REQUEXTS
        TRANSFER    ,SKIP1
 FOUR   MSAVEVALUE  4,XH4,1,PH2,MH    STORE THE NUMBER OF IO REQUEXTS
        TRANSFER    ,SKIP1
 THREE  MSAVEVALUE  3,XH3,1,PH2,MH    STORE THE NUMBER OF IO REQUESTS
        TRANSFER    ,SKIP1
 TWO    MSAVEVALUE  2,XH2,1,PH2,MH    STORE THE NUMBER OF IO REQUESTS
        TRANSFER    ,SKIP1
 ONE    MSAVEVALUE  1,XH1,1,PH2,MH    STORE THE NUMBER OF IO REQUESTS
        TRANSFER    ,SKIP1
 SKIP1  ADVANCE     1                  TIME TAKEN TO WRITE TO TABLE
        TRANSFER    ,READ
*
*               ROUTINE CALLED FOR SECONDARY MODIFICATION REQUESTS
*
 UPDT2  TEST L      XH*V$LAST,XH*PH5,ENDIT   CONTINUE IF UPDATES PENDING
        SAVEVALUE   V$LAST+,1,XH      UPDATE TABLE POINTER LAST
        ASSIGN      7,1,PH
*
*               SEND THE SECONDARY MODIFICATIONS
*
 LOOPC  TEST LE     PH7,PH6,FINUP     IF COUNTER GT # OF BE GO TO FINLP
        TEST NE     PH7,PH5,ELSE
        ASSIGN      15,PH7,PF         DETERMINE # OF BE TO BE UPDATED
```

```
      TRANSFER   ,SKIP2
ELSE  ASSIGN     7+,1,PH
      TRANSFER   ,LOOPC
SKIP2 ASSIGN     9,V$BECNL,PH
      ASSIGN     7+,1,PH          INCREMENT COUNTER
      SPLIT      1,LOOPC          SEND OUT ANOTHER UPDATE TO NEXT BE
*
      QUEUE      PH5
      ENTER      PH5
      DEPART     PH5
      SEIZE      PH5
      ADVANCE    PH13             BINT
      RELEASE    PH5
      LEAVE      PH5
      QUEUE      V$BECNL          QUEUE FOR BE TO BE CHANNEL
      SEIZE      V$BECNL          SEIZE THE BE TO BE CHANNEL
      DEPART     V$BECNL          LEAVE THE BE TO BE CHANNEL QUEUE
      ADVANCE    V$CHANL          TIME TAKEN TO SEND MESSAGE ACROSS THE CHANNEL
      ADVANCE    PH14             MESSAGE SYSTEM
      RELEASE    V$BECNL          FREE THE BE TO BE CHANNEL
      QUEUE      PH15
      ENTER      PH15
      DEPART     PH15
      SEIZE      PH15
      ADVANCE    PH13             BINT ON NEXT BE
      ASSIGN     4,2,PH           CODE UPDATE AS SECONDARY
      ASSIGN     7,0,PH          COUNTER
      ASSIGN     1,XH*V$LAST,PF
      ASSIGN     2,MH*PH5(PF1,1),PH   ASSIGN INTO PH2 THE # OF IC REQ
      ASSIGN     5,PH15,PH        ASSIGN PH15 INTO PH5
      TRANSFER   ,LOOPB
*
      START      9000
      NOXREF
      END
```

```
      REALLCCATE FAC,200,CUE,200,BLC,250
      REALLOCATE STO,10,XAC,4CO,VAR,1C,FUN,15,TAB,0,CCM,30880
      SIMULATE
*
*       JOHNSON & THOMAS
*       MCDEL CF 4 HCST CPU'S, 8 BE'S, 50% UPDATES, & 10% PRIORITY
*
*
* **    ENTITY DEFINITICNS
*
*   PARAMETERS
*       PH1: TERMINAL ASSIGNMENT NUMBER
*       PF1: HALFWORD SAVEVALLE WHOSE NUMBER IS 10+PH5
*       PH2: NUMBER OF I/O REQUESTS NECESSARY FCR EACH USER RWQLEST
*       PF2: THE SPECIFIC RECORD NUMBER
*       PH3: SIZE OF MESSAGE BUFFER
*       PF3: THE ABSCLUTE CLOCK TIME
*       PH4: MARKED 0 IF READ REQUEST
*                   1 IF PRIMARY UPDATE REQUEST
*                   2 IF SECONDARY UPDATE REQUEST
*       PF4: MARKED 0 IF UPDATE REQUEST HAS NCT BEEN SPCCLED
*                   1 IF UPDATE REQUEST HAS BEEN SPCCLED
*       PH5: ID NUMBER CF THE BACKEND MACHINE BEING USED
*       PF5: NUMBER CF LAST TABLE ENTRY
*       PH6: NUMBER CF BACKEND MACHINES USED IN THE SIMULATICN
*       PF6: PRIMARY CR SECONDARY UPDATE FLAG
*       PH7: A CCUNTER
*       PH8: NUMBER CF REQUESTS MADE BY THE USER
*       PH9: CHANNEL ID USED BETWEEN THE BACKEND MACHINES
*       PH10: ID NUMBER OF TERMINAL TO HCST CHANNEL
*       PF10: NUMBER OF HCST CPU
*       PF11: TRANSMISSION SPEED TO CHANNEL IN BITS/SEC
*       PF12: CHANNEL TRANSMISSION SPEED IN BITS/SEC
*       PH13: TIME USED BY HINT OR BINT
*       PF13: PRIORITY CF REQUESTS
*                   1 IF NORMAL REQUEST
*                   10 IF HIGH PRIORITY REQUEST
*       PH14: TIME USED BY THE MESSAGE SYSTEM
*       PF14: ID NUMBER OF HOST CPU-BE CHANNEL
*       PH15: ID NUMBER CF THE BE THAT THE SECONDARY UPDATE IS SENT TC
*       PF15: NUMBER OF HCST CPU'S IN THE SIMULATICN MODEL
*
*   STORAGES REPRESENT PARTITICNS IN EACH BACKEND
*
*   FACILITIES
*
*       1 - 8: CPU'S CF THE EIGHT BACKENC'S
*       11,22,33,44,55,66,77,88 : CHANNELS FRCM BACKEND TC DEVICE
*       12->18: CHANNELS FRCM BE1 TC BE2,...,BE8
*       21,23,...28: CHANNELS FRCM BE2 TC CTHER BE'S,RESPECTIVELY
*       101->1C8: TERMINALS CONNECTED TC HOST
*   DEFINITICNS
*       UPDAT1: PRIMARY UPDATE
*       UPDAT2: SECONDARY UPDATE
*       LAST: ROW OF MATRIX LAST USED IN UPDATING
*       CURRENT: NUMBER OF UPDATES SPCOLEC TC THAT BE
*
 1      MATRIX      MX,200,1
 2      MATRIX      MX,200,1
 3      MATRIX      MX,200,1
```

```
 4      MATRIX      MX,200,1
 5      MATRIX      MX,200,1
 6      MATRIX      MX,200,1
 7      MATRIX      MX,200,1
 8      MATRIX      MX,200,1
 1      MATRIX      MH,200,2
 2      MATRIX      MH,200,2
 3      MATRIX      MH,200,2
 4      MATRIX      MH,200,2
 5      MATRIX      MH,200,2
 6      MATRIX      MH,200,2
 7      MATRIX      MH,200,2
 8      MATRIX      MH,200,2
        STORAGE     S1-S8,2
        STORAGE     S9,2000
 PLUS   VARIABLE    10+PH1
 MINUS  VARIABLE    PH6-1
 TRMNL  FVARIABLE   PH3*8/PF11*1000
 CHANL  FVARIABLE   PH3*8/PF12*1000
 LAST   VARIABLE    10+PH5
 INCRE  VARIABLE    PH7+1
 BECNL  VARIABLE    10*PH5+PH15
 HCBE   VARIABLE    PF10+PH5
 PRIOR  VARIABLE    PF13+2
*  50% UPDATE  ****************************************
 TYPE   FUNCTION    RN2,D2
.50,0/1.00,1
 TERM   FUNCTION    RN2,D8
.125,101/.25,102/.375,103/.5,104/.625,105/.75,106/.875,107/1.0,108
 RECRD FUNCTION     RN2,D10
.10,1/.20,2/.30,3/.40,4/.50,5/.60,6/.70,7/.80,8/.90,9/1.0,10
 RECNO FUNCTION     RN2,D10
.10,1/.20,2/.30,4/.40,5/.50,7/.60,9/.70,11/.80,12/.90,13/1.00,14
 LNGTH FUNCTION     RN2,C2
.90,128/1.00,256
*  90% FIT IN ONE BUFFER
 BEDEV FUNCTION     PH5,C8              BE TO DEVICE CHANNELS
1.11/2.22/3.33/4.44/5.55/6.66/7.77/8.88
 USREQ FUNCTION     RN2,D9              # OF REQUESTS BY A USER
.10,1/.20,3/.40,5/.50,7/.60,9/.70,11/.80,13/.90,15/1.00,17
*
 RNDM   FUNCTION    RN2,C8
.125,1/.25,2/.375,3/.50,4/.625,5/.75,6/.875,7/1.0,8
 RNDM2 FUNCTION     RN2,C4
.25,111/.50,112/.75,113/1.0,114
 HOST   FUNCTION    PH10,D4
111,150/112,160/113,170/114,180
 PRIOR FUNCTION     RN2,D2
.9,1/1.0,11          **************   10% HIGH PRIORITY REQUESTS *********
*
       INITIAL      XH1-XH8,0         INITIALIZE SAVEVALUES 1-10 TO ZERO
       INITIAL      XF1-XF8,0          INITIALIZE XF1-8 TO ZERO
       INITIAL      XH11-XH18,0       INITIALIZE SAVEVALUES 11-14 TO ZERO
*
       GENERATE     375,50,,60,,15PF,15PH   GENERATE 60 TRANX 1 PER 3/8 SEC
       ASSIGN       13,FN$PRIOR,PF     ASSIGN THE REQUEST A PRIORITY 1 OR 11
       PRIORITY     PF13               ASSIGN A PRIORITY TO THE  REQUEST
       ASSIGN       15,4,PF            NUMBER OF HOST CPU'S IN THE SIMULATION
       ASSIGN       6,8,PH             NUMBER OF BACKENDS IN SIMULATION
       ASSIGN       1,FN$TERM,PH       ASSIGN TERMINAL # IN PH1
```

```
        ASSIGN      8,FN$USREC,PH     STORE # OF USER'S COMMANDS IN PH8
        ASSIGN      11,50000,PF         TRANSMISSION SPEED TO CHANNEL
        ASSIGN      12,50000,PF         TRANSMISSION SPEED OF CHANNEL
        ASSIGN      13,5,PH           ADVANCE TIME FOR HINT OR BINT
        ASSIGN      14,5,PH           ADVANCE TIME FOR MESSAGE SYSTEM
        SAVEVALUE   9+,1,XH           INCREMENT USER COUNTER
        QUEUE       PH1               QUEUE FOR TERMINAL
        SEIZE       PH1               SEIZE THE TERMINAL
        DEPART      PH1               DEPART QUEUE FOR TERMINAL
*
*             MAIN LOOP FOR USER REQUESTS
*
MORE    ADVANCE     1000,500          TIME TAKEN TO TYPE REQUEST
        ASSIGN      2,FN$RECNO,PH     ASSIGN # OF ID REQUESTS IN PH 2
        ASSIGN      3,FN$LNGTH,PH     LENGTH OF BUFFER TRANSMISSION
        ASSIGN      4,FN$TYPE,PH      ASSIGN UPDATE OR READ
        ASSIGN      8-,1,PH           DECREMENT # OF REQUESTS MADE BY USER
        ASSIGN      10,FN$RNDM2,PH    RANDOMLY SELECT A HOST CPU CHANNEL
        ASSIGN      10,FN$HOST,PF     ASSIGN THE REQUEST TO CPU OF HOST/CPU TERMINA
        QUEUE       PH10
        SEIZE       PH10              TERMINAL - HOST CHANNEL
        DEPART      PH10              LEAVE QUEUE HOST-TERMINAL CHANNEL
        ADVANCE     V$TRMNL           LINE TRANSMISSION
        ADVANCE     V$CHANL
        RELEASE     PH10              RELEASE THE HOST CPU-TERMINAL CHANNEL
        QUEUE       PF10              QUEUE FOR THE ASSIGNED HOST CPU
        SEIZE       PF10              SEIZE HOST CPU
        DEPART      PF10              LEAVE THE QUEUE
        ADVANCE     PH13              HINT
        ADVANCE     PH14              MESSAGE SYSTEM
        TEST E      PH4,1,*+3         IF UPDATE MAKE RECORD & CLOCK TIME ASSIGNMENT
        ASSIGN      2,FN$RECRD,PF     MAKE RECORD ASSIGNMENT
        ASSIGN      3,C1,PF           MAKE CLOCK TIME ASSIGNMENT
        ASSIGN      5,1,PH
*
*             CHECK FOR A FREE MACHINE
*
LOOPA   RELEASE     PF10              FREE HOST CPU
        ASSIGN      14,V$HCBE,PF      SELECT A HOST-BE CHANNEL
        QUEUE       PF14              WAIT IN THE SELECTED HOST CPU-BE CHANNEL QUEU
        SEIZE       PF14              SEIZE HOST-BE CHANNEL
        DEPART      PF14              DEPART THE QUEUE
        ADVANCE     V$CHANL
        ADVANCE     1                 TIME TAKEN TO CHECK IF FREE
        GATE SNE    PH5,BEDB          GO TO BEDB IF BE FREE
        RELEASE     PF14              RETURN THE CHANNEL
        QUEUE       PF10
        SEIZE       PF10              SEIZE HOST CPU
        DEPART      PF10
        ASSIGN      5+,1,PH
        TEST LE     PH5,PH6,FULL      SEND TO FULL IF ALL BE'S ARE BUSY
        TRANSFER    ,LOOPA
FULL    ASSIGN      5,FN$RNDM,PH      RANDOMLY ASSIGN BE MACHINE
        ADVANCE     1                 ASSIGNMENT TIME
        RELEASE     PF10              RELEASE HOST CPU
*             BACKEND HAS BEEN SELECTED
*
*
*
        ASSIGN      14,V$HCBE,PF      SELECT A HOST-BE CHANNEL
```

```
        QUEUE       PF14
        SEIZE       PF14
        DEPART      PF14
        ADVANCE     V$CHANL
BEDB    RELEASE     PF14
        ADVANCE     PH13            BINT
        QUEUE       PH5             QUEUE FOR BE PARTITION
        ENTER       PH5             ENTER BE PARTITION
        DEPART      PH5
        TEST E      PH4,1,*+2
        PRIORITY    V$PRIOR
        SEIZE       PH5             SEIZE BE CPU
        ASSIGN      7,0,PH          PH7 IS A COUNTER
*
*               PROCESSING OF IO REQUESTS
*
LOOPB   TEST NE     PH2,PH7,FINIO   IF IO DONE GO TO FINIO
        TEST NE     PH4,1,*+2       IF UPDATE DON'T RELEASE DATABASE
        RELEASE     PH5             RETURN BE CPU
        QUEUE       FN$BEDEV
        SEIZE       FN$BEDEV        SEIZE BE DEVICE FOR IO
        DEPART      FN$BEDEV
        ADVANCE     52              TIME FOR EACH IO
        TEST E      PH4,1,*+4       IS REQUEST A PRIMARY UPDATE?
        TEST NE     PF4,1,*+3       HAS THIS UPDATE ALREADY BEEN SPOOLED?
        ADVANCE     30              IF NOT, SPOOL THE UPDATE
        ASSIGN      4,1,PF          MARK THE UPDATE AS SPOOLED
        RELEASE     FN$BEDEV        RELEASE DEVICE
        TEST NE     PH4,1,*+2       IF UPDATE, BE IS ALREADY SEIZED
        SEIZE       PH5             SEIZE BE CPU
        ADVANCE     1               PROCESSING TIME
        ASSIGN      7+,1,PH         INCREMENT COUNTER
        TRANSFER    ,LOOPB
*
*               IO COMPLETED
*
*
FINIO   TEST NE     PH4,1,UPDT1     IF UPDATE, GO TO UPDT1-CHANGE TABLE
        TEST E      PH4,2,READ      IF QUERY ONLY COMMAND, GO TO READ
*
*         REQUEST IS A SECONDARY MODIFICATION
*
RECPU   RELEASE     PH5
        LEAVE       PH5             RELEASE THE BE PARTITION
        LEAVE       9               LEAVE SECONDARY UPDATE STORAGE
*
FINUP   TEST G      S*PH5,1,UPDT2   IF BE PARTITION EMPTY GO TO UPDATES
        TRANSFER    ,END2           ELSE END TRANSACTION
*
*               RETURN INFORMATION TO THE HOST AND TERMINALS
*
READ    ADVANCE     PH13            BINT
        RELEASE     PH5
        LEAVE       PH5             LEAVE PARTITION IN BE
        TEST E      S*PH5,0,*+2     IF PARTITIONS EMPTY, SPLIT
        SPLIT       1,UPDT2         SPLIT AND SEND 1 TO UPDT2
        QUEUE       PF14            WAIT FOR HOST-BE CHANNEL
        SEIZE       PF14            SEIZE HOST - BE CHANNEL
        DEPART      PF14
        ADVANCE     V$CHANL
```

```
         ADVANCE    PH14              MESSAGE SYSTEM
         RELEASE    PF14
         QUEUE      PF10              QUEUE FOR HOST CPU
         PREEMPT    PF10,PR           PREEMPT THE HOST CPU
         DEPART     PF10
         ADVANCE    PH13              HINT
         RETURN     PF10
         QUEUE      PH10              WAIT FOR CPU-TERMINAL CHANNEL
         PREEMPT    PH10,PR
         DEPART     PH10
         ADVANCE    VSTRMNL
         RETURN     PH10
         TEST E     PH8,0,MORE        MORE REQUESTS BY THIS USER?
         RELEASE    PH1               RELEASE OCCUPIED TERMINAL
         GATE SE    9                 NO ENTRY TILL ALL UPDATES HAVE BEEN COMPLETED
         TERMINATE  1
*
*          ENTER MODIFICATION REQUESTS IN BE MODIFICATION TABLE
*
UPDTI    SAVEVALUE  PH5+,1,XH         INCREMENT # OF UPDATES SPECIFIED
         TEST NE    PH5,1,ONE         DETERMINE BE YOU ARE UPDATING ON
         TEST NE    PH5,2,TWO         SO YOU CAN UPDATE APPROPPIATE TABLE.
         TEST NE    PH5,3,THREE
         TEST NE    PH5,4,FOUR
         TEST NE    PH5,5,FIVE
         TEST NE    PH5,6,SIX
         TEST NE    PH5,7,SEVEN
         TEST NE    PH5,8,EIGHT
EIGHT    MSAVEVALUE 8,XH8,1,PH2,MH    STORE THE NUMBER OF IO REQUESTS
         MSAVEVALUE 8,XH8,2,PF2,MH    STORE THE RECORD NUMBER
         MSAVEVALUE 8,XH8,1,PF3,MX    STORE THE ABSOLUTE CLOCK TIME OF UPDATE REQ
         TRANSFER   ,SKIP1
SEVEN    MSAVEVALUE 7,XH7,1,PH2,MH    STORE THE NUMBER OF IO REQUESTS
         MSAVEVALUE 7,XH7,2,PF2,MH    STORE THE RECORD NUMBER
         MSAVEVALUE 7,XH7,1,PF3,MX    STORE THE ABSOLUTE CLOCK TIME OF UPDATE REQ
         TRANSFER   ,SKIP1
SIX      MSAVEVALUE 6,XH6,1,PH2,MH    STORE THE NUMBER OF IO REQUESTS
         MSAVEVALUE 6,XH6,2,PF2,MH    STORE THE RECORD NUMBER
         MSAVEVALUE 6,XH6,1,PF3,MX    STORE THE ABSOLUTE CLOCK TIME OF UPDATE REQ
         TRANSFER   ,SKIP1
FIVE     MSAVEVALUE 5,XH5,1,PH2,MH    STORE THE NUMBER OF IO REQUESTS
         MSAVEVALUE 5,XH5,2,PF2,MH    STORE THE RECORD NUMBER
         MSAVEVALUE 5,XH5,1,PF3,MX    STORE THE ABSOLUTE CLOCK TIME OF UPDATE REQ
         TRANSFER   ,SKIP1
FOUR     MSAVEVALUE 4,XH4,1,PH2,MH    STORE THE NUMBER OF IO REQUESTS
         MSAVEVALUE 4,XH4,2,PF2,MH    STORE THE RECORD NUMBER
         MSAVEVALUE 4,XH4,1,PF3,MX    STORE THE ABSOLUTE CLOCK TIME OF UPDATE REQ
         TRANSFER   ,SKIP1
THREE    MSAVEVALUE 3,XH3,1,PH2,MH    STORE THE NUMBER OF IO REQUEXTS
         MSAVEVALUE 3,XH3,2,PF2,MH    STORE THE RECORD NUMBER
         MSAVEVALUE 3,XH3,1,PF3,MX    STORE THE ABSOLUTE CLOCK TIME OF UPDATE REQ
         TRANSFER   ,SKIP1
TWO      MSAVEVALUE 2,XH2,1,PH2,MH    STORE THE NUMBER OF IO REQUEXTS
         MSAVEVALUE 2,XH2,2,PF2,MH    STORE THE RECORD NUMBER
         MSAVEVALUE 2,XH2,1,PF3,MX    STORE THE ABSOLUTE CLOCK TIME OF UPDATE REQ
         TRANSFER   ,SKIP1
ONE      MSAVEVALUE 1,XH1,1,PH2,MH    STORE THE NUMBER OF IO REQUEXTS
         MSAVEVALUE 1,XH1,2,PF2,MH    STORE THE RECORD NUMBER
         MSAVEVALUE 1,XH1,1,PF3,MX    STORE THE ABSOLUTE CLOCK TIME OF UPDATE REQ
         TRANSFER   ,SKIP1
```

```
 SKIP1 ADVANCE    1                  TIME TAKEN TO WRITE TO TABLE
*
       TRANSFER   ,READ
*
*        ROUTINE CALLED TO PERFORM SECONDARY MODIFICATION REQUESTS
*
 UPDT2 TEST L     XH*V$LAST,XH*PH5,END2    CONTINUE IF UPDATES PENDING
       SAVEVALUE  V$LAST+,1,XH       UPDATE TABLE POINTER LAST
       ASSIGN     7,1,PH
*
 LOOPC TEST L     PH7,PH6,FINUP      IF COUNTER=# OF BE GO TO FINUP
       TEST L     PH7,PH5,ELSE
       ASSIGN     15-PH7,PH          DETERMINE # OF BE TO BE UPDATED
       TRANSFER   ,SKIP2
 ELSE  ASSIGN     15,V$INCRE,PF
 SKIP2 ASSIGN     9,V$BECNL,PH
       ASSIGN     7+,1,PF            INCREMENT COUNTER
       SPLIT      1,LOOPC            SEND OUT ANOTHER UPDATE TO NEXT BE
*
       QUEUE      PH5
       ENTER      PH5
       DEPART     PH5
       SEIZE      PH5                SEIZE THE BE CPU
       ADVANCE    PH13               BINT
       RELEASE    PH5
       LEAVE      PH5
        ENTER     9                  ENTER UPDATE PENDING STORAGE
       QUEUE      PH9               QUEUE FOR CHANNEL BETWEEN 2 BE'S
       SEIZE      PH9                BE TO BE CHANNEL
       DEPART     PH9
       ADVANCE    V$CHANL
       ADVANCE    PH14               MESSAGE SYSTEM
       RELEASE    PH9
       QUEUE      PH15
       ENTER      PH15
       DEPART     PH15
       ASSIGN     1,XH*V$LAST,PF
       PRIORITY   V$PRIOR
       SEIZE      PH15                SEIZE THE BE CPU
*
*        CHECK THE TIMESTAMPS OF SAME RECORD MODIFICATIONS AT THAT BE
*
 TIMCK ASSIGN     5,XH*PH15,PF       INITIALIZE CNTR TO FINAL ENTRY IN UPDT TABLE
       TEST NE    PF5,0,CONTU        IF THERE ARE NO ENTRIES,  CONTINUE.
 LOOPD TEST L     MX*PH5(PF1,1),MX*PH15(PF5,1),CONTU   IS TIMESTAMP LESS?
*
*        IF TIMESTAMP OLDER, DON'T PERFORM MODIFICATION
*
       TEST E     MH*PH5(PF1,2),MH*PH15(PF5,2),*+5   IS IT THE SAME RECORD?
       RELEASE    PH15               RELEASE THE BACKEND PROCESSOR
       LEAVE      PH15               LEAVE THE BE PARTITION
       LEAVE      9                  LEAVE SECONDARY UPDATE STORAGE LIST
*
 END2  TERMINATE  0
*
       ASSIGN     5-,1,PF            DECREMENT COUNTER
       TEST E     PF5,0,LOOPC        LOOP IF MORE IN TABLE
 CONTU ADVANCE    PH13               BINT ON NEXT BE
       ASSIGN     4,2,PH             CODE UPDATE AS SECONDARY
       ASSIGN     7,0,PH            COUNTER
```

```
ASSIGN      2,NH*PH5(PF1,1),PH   ASSIGN INTO PH2 THE # OF IC REQ
ASSIGN      5,PH15,PH            ASSIGN PH15 INTO PH5
TRANSFER    ,LOOPB
NOXREF
END
```

```
NOTE: THE JOB SPEC0064 HAS BEEN RUN UNDER RELEASE 76.6C OF SAS AT KANSAS STATE UNIVERSITY.

1          DATA CODY;  INPUT ALGOR $ H B RESP_TI R F P M ;
2          FAC1=R/F;
3          B_SQ=B*B;
4          F_X_B=F*B;
5          FAC2=(B*P)/F;
6          FAC3=(B*M)/F;
7          CARDS;

NOTE: DATA SET WORK.CODY HAS 15 OBSERVATIONS AND 13 VARIABLES. 120 OBS/TRK.
NOTE: THE DATA STATEMENT USED 0.39 SECONDS AND 102K.

23         PROC PRINT;
24         TITLE1 CODASYL MODEL;
25         TITLE2 MODEL  MEAN RESPONSE TIME = FAC1  B_SQ  F_X_B FAC2 FAC3;
26         TITLE3 FAC1 = B/F        B_SQ = B**2          F_X_B = F*B;
27         TITLE4 FAC2 = (B*P)/F    FAC3 = (B*M)/F     ;
28         TITLE5 B = NUMBER OF BACK-ENDS                F = SEC PER REQUEST;
29         TITLE6 P = PERCENT PRIORITY REQUESTS          M = PERCENT MODIFICATIONS;

NOTE: THE PROCEDURE PRINT USED 0.56 SECONDS AND 122K AND PRINTED PAGE 1.

30         PROC GLM;
31         MODEL RESP_TI = FAC1  B_SQ F_X_B FAC2 FAC3 ;

NOTE: THE PROCEDURE GLM USED 0.73 SECONDS AND 160K AND PRINTED PAGES 2 TO 3.

32         DATA ALL;INPUT ALGOR $ H B RESP_TI R F P M ;
33         FAC1=R/F;
34         OVER_B=1/B;
35         OVER_HSQ=1/(H**2);
36         FAC2=F/B;
37         B_X_F=B*F;
38         FAC3=M/(B*F);
39         FAC4=P/(B*F);
40         CARDS;

NOTE: DATA SET WORK.ALL HAS 57 OBSERVATIONS AND 15 VARIABLES. 105 OBS/TRK.
NOTE: THE DATA STATEMENT USED 0.37 SECONDS AND 102K.

98         PROC PRINT;
99         TITLE BERNSTEIN, ELLIS, HYBRID, AND JOHNSON & THOMAS MODELS;
100        TITLE2 MODEL  MEAN RESPONSE TIME = B FAC1 OVER_B OVER_HSQ FAC2 B_X_F FAC3 FAC4;
101        TITLE3 FAC1 = B/F        OVER_B = 1/B         OVER_HSQ = 1/H**2;
102        TITLE4 FAC2 = F/B        B_X_F = B*F          FAC3 = M/(B*F)    ;
103        TITLE5 FAC4 = P/(B*F)                         H = NUMBER OF HOSTS     ;
104        TITLE6 B = NUMBER OF BACK-ENDS                F = SEC PER REQUEST;
105        TITLE7 P = PERCENT PRIORITY REQUESTS          M = PERCENT MODIFICATIONS;

NOTE: THE PROCEDURE PRINT USED 0.95 SECONDS AND 122K AND PRINTED PAGES 4 TO 5.

106        PROC GLM;
107        CLASS ALGOR;
108        BY ALGOR;
```

```
109       MODEL RESP_TI = B FAC1 OVER_B OVER_HSQ FAC2 B_X_F FAC3 FAC4 / SOLUTION;

NOTE: THE PROCEDURE GLM USED 2.37 SECONDS AND 162K AND PRINTED PAGES 6 TO 13.

NOTE: SAS USED 162K MEMORY.

NOTE: BARR, GOODNIGHT, SALL AND HELWIG
      SAS INSTITUTE INC.
      P.O. BOX 10066
      RALEIGH, N.C. 27605
```

COCASYL MODEL

MODEL MEAN RESPONSE TIME = FAC1 B_SQ F_X_B FAC2 FAC3

FAC1 = B/F B_SQ = B**2 F_X_B = F*B

FAC2 = (B*P)/F FAC3 = (B*M)/F

B = NUMBER OF BACK-ENDS F = SEC PER REQUEST

P = PERCENT PRIORITY REQUESTS M = PERCENT MODIFICATIONS

OBS	ALGOR	M	B	RESP_TI	R	F	P	M	FAC1	B_SQ	F_X_B	FAC2	FAC3
1	C	1	2	86.066	15	1.500	10	50	10	4	3.00	13.333	66.67
2	C	1	4	93.641	15	1.500	10	50	10	16	6.00	26.667	133.33
3	C	1	8	98.029	15	1.500	10	50	10	64	12.00	53.333	266.67
4	C	2	2	145.283	30	0.750	10	50	40	4	1.50	26.667	133.33
5	C	2	4	152.316	30	0.750	10	50	40	16	3.00	53.333	266.67
6	C	2	8	191.103	30	0.750	10	50	40	64	6.00	106.667	533.33
7	C	4	2	317.230	60	0.375	10	50	160	4	0.75	53.333	266.67
8	C	4	4	323.542	60	0.375	10	50	160	16	1.50	106.667	533.33
9	C	4	8	.	60	0.375	10	50	160	64	3.00	213.333	1066.67
10	C	1	2	78.990	15	1.500	0	20	10	4	3.00	0.000	26.67
11	C	1	2	88.520	15	1.500	0	35	10	4	3.00	0.000	46.67
12	C	1	4	68.245	15	1.500	0	20	10	16	6.00	0.000	53.33
13	C	1	4	75.141	15	1.500	0	35	10	16	6.00	0.000	93.33
14	C	1	8	77.068	15	1.500	0	20	10	64	12.00	0.000	106.67
15	C	1	8	84.298	15	1.500	0	35	10	64	12.00	0.000	186.67

COCASYL MODEL

MODEL MEAN RESPONSE TIME = FAC1 B_SQ F_X_B FAC2 FAC3

FAC1 = B/F B_SQ = B**2 F_X_B = F*B

FAC2 = (B*P)/F FAC3 = (B*M)/F

B = NUMBER OF BACK-ENDS F = SEC PER REQUEST

P = PERCENT PRIORITY REQUESTS M = PERCENT MODIFICATIONS

GENERAL LINEAR MODELS PROCEDURE

DEPENDENT VARIABLE INFORMATION

NUMBER OF OBSERVATIONS IN DATA SET = 15

NOTE: ALL DEPENDENT VARIABLES ARE CONSISTENT WITH RESPECT TO THE PRESENCE OR ABSENCE OF MISSING VALUES. HOWEVER, ONLY 14 OBSERVATIONS IN DATA SET CAN BE USED IN THIS ANALYSIS.

CODASYL MODEL

MODEL MEAN RESPONSE TIME = FAC1 B_SQ F_X_B FAC2 FAC3
FAC1 = B/F B_SQ = B**2 F_X_B = F*B
FAC2 = (B*P)/F FAC3 = (B*M)/F
B = NUMBER OF BACK-ENDS F = SEC PER REQUEST
P = PERCENT PRIORITY REQUESTS M = PERCENT MODIFICATIONS

GENERAL LINEAR MODELS PROCEDURE

DEPENDENT VARIABLE: RESP_TI

SOURCE	DF	SUM OF SQUARES	MEAN SQUARE	F VALUE	PR > F	R-SQUARE	C.V.
MODEL	5	96667.99811157	19333.59962231	419.52	0.0001	0.996201	5.0568
ERROR	8	368.68196243	46.08524530		STD DEV		RESP_TI MEAN
CORRECTED TOTAL	13	97036.68007400			6.78861144		134.24800000

SOURCE	DF	TYPE I SS	F VALUE	PR > F	DF	TYPE IV SS	F VALUE	PR > F
FAC1	1	92652.27575465	2010.45	0.0001	1	19003.82758259	412.36	0.0001
B_SQ	1	638.68000968	13.86	0.0058	1	406.09720440	8.81	0.0179
F_X_B	1	3013.45265002	65.39	0.0001	1	573.38216567	12.44	0.0078
FAC2	1	354.39378123	7.69	0.0242	1	124.76433736	2.71	0.1385
FAC3	1	9.19591599	0.20	0.6669	1	9.19591599	0.20	0.6669

PARAMETER	ESTIMATE	T FOR HO: PARAMETER=0	PR > |T|	STD ERROR OF ESTIMATE
INTERCEPT	89.20747299	12.32	0.0001	7.23833163
FAC1	1.31191234	20.31	0.0001	0.06460487
B_SQ	0.98374036	2.97	0.0179	0.33139537
F_X_B	-6.69186434	-3.53	0.0078	1.89716887
FAC2	0.39499193	1.65	0.1385	0.24006229
FAC3	-0.03016417	-0.45	0.6669	0.06752659

BERNSTEIN, ELLIS, HYBRID, AND JOHNSON & THOMAS MODELS

MODEL MEAN RESPONSE TIME = B FAC1 OVER_B OVER_HSQ FAC2 B_X_F FAC3 FAC4

FAC1 = B/F OVER_B = 1/B OVER_HSQ = 1/H**2

FAC2 = F/B B_X_F = B*F FAC3 = M/(B*F)

FAC4 = P/(B*F) H = NUMBER OF HOSTS

B = NUMBER OF BACK-ENDS F = SEC PER REQUEST

P = PERCENT PRIORITY REQUESTS M = PERCENT MODIFICATIONS

OBS	ALGOR	H	B	RESP_TI	R	F	P	M	FAC1	OVER_B	OVER_HSQ	FAC2	B_X_F	FAC3	FAC4
1	B	4	2	232.369	60	0.375	10	50	160	0.500	0.0625	0.187500	0.75	66.6667	13.3333
2	B	4	4	197.920	60	0.375	10	50	160	0.250	0.0625	0.093750	1.50	33.3333	6.6667
3	B	4	8	207.853	60	0.375	10	50	160	0.125	0.0625	0.046875	3.00	16.6667	3.3333
4	B	2	2	125.280	30	0.750	10	50	40	0.500	0.2500	0.375000	1.50	33.3333	6.6667
5	B	2	4	124.612	30	0.750	10	50	40	0.250	0.2500	0.187500	3.00	16.6667	3.3333
6	B	2	8	129.305	30	0.750	10	50	40	0.125	0.2500	0.093750	6.00	8.3333	1.6667
7	B	1	2	81.840	15	1.500	10	50	10	0.500	1.0000	0.750000	3.00	16.6667	3.3333
8	B	1	4	83.104	15	1.500	10	50	10	0.250	1.0000	0.375000	6.00	8.3333	1.6667
9	B	1	8	76.339	15	1.500	10	50	10	0.125	1.0000	0.187500	12.00	4.1667	0.8333
10	B	1	2	133.729	30	0.750	10	50	40	0.500	1.0000	0.375000	1.50	33.3333	6.6667
11	B	4	2	132.897	30	0.750	10	50	40	0.500	0.0625	0.375000	1.50	33.3333	6.6667
12	B	1	2	78.290	15	1.500	0	50	10	0.500	1.0000	0.750000	3.00	16.6667	0.0000
13	E	4	2	228.949	60	0.375	10	50	160	0.500	0.0625	0.187500	0.75	66.6667	13.3333
14	E	4	4	229.890	60	0.375	10	50	160	0.250	0.0625	0.093750	1.50	33.3333	6.6667
15	E	4	8	212.411	60	0.375	10	50	160	0.125	0.0625	0.046875	3.00	16.6667	3.3333
16	E	2	2	122.394	30	0.750	10	50	40	0.500	0.2500	0.375000	1.50	33.3333	6.6667
17	E	2	4	128.560	30	0.750	10	50	40	0.250	0.2500	0.187500	3.00	16.6667	3.3333
18	E	2	8	126.328	30	0.750	10	50	40	0.125	0.2500	0.093750	6.00	8.3333	1.6667
19	E	1	2	72.547	15	1.500	10	50	10	0.500	1.0000	0.750000	3.00	16.6667	3.3333
20	E	1	4	82.234	15	1.500	10	50	10	0.250	1.0000	0.375000	6.00	8.3333	1.6667
21	E	1	8	78.655	15	1.500	10	50	10	0.125	1.0000	0.187500	12.00	4.1667	0.8333
22	E	1	2	78.942	15	1.500	0	20	10	0.500	1.0000	0.750000	3.00	6.6667	0.0000
23	E	1	2	74.978	15	1.500	0	35	10	0.500	1.0000	0.750000	3.00	11.6667	0.0000
24	E	1	4	68.969	15	1.500	0	20	10	0.250	1.0000	0.375000	6.00	3.3333	0.0000
25	E	1	4	69.110	15	1.500	0	35	10	0.250	1.0000	0.375000	6.00	5.8333	0.0000
26	E	1	8	69.882	15	1.500	0	20	10	0.125	1.0000	0.187500	12.00	1.6667	0.0000
27	E	1	8	64.039	15	1.500	0	35	10	0.125	1.0000	0.187500	12.00	2.9167	0.0000
28	H	1	2	61.346	15	1.500	0	20	10	0.500	1.0000	0.750000	3.00	6.6667	0.0000
29	H	1	2	69.179	15	1.500	0	35	10	0.500	1.0000	0.750000	3.00	11.6667	0.0000
30	H	1	4	66.908	15	1.500	0	20	10	0.250	1.0000	0.375000	6.00	3.3333	0.0000
31	H	1	4	66.779	15	1.500	0	35	10	0.250	1.0000	0.375000	6.00	5.8333	0.0000
32	H	1	8	62.362	15	1.500	0	20	10	0.125	1.0000	0.187500	12.00	1.6667	0.0000
33	H	1	8	64.166	15	1.500	0	35	10	0.125	1.0000	0.187500	12.00	2.9167	0.0000
34	H	1	2	82.648	15	1.500	10	50	10	0.500	1.0000	0.750000	3.00	16.6667	3.3333
35	H	1	4	74.143	15	1.500	10	50	10	0.250	1.0000	0.375000	6.00	8.3333	1.6667
36	H	1	8	66.509	15	1.500	10	50	10	0.125	1.0000	0.187500	12.00	4.1667	0.8333
37	H	2	2	124.844	30	0.750	10	50	40	0.500	0.2500	0.375000	1.50	33.3333	6.6667
38	H	2	4	96.589	30	0.750	10	50	40	0.250	0.2500	0.187500	3.00	16.6667	3.3333
39	H	2	8	108.884	30	0.750	10	50	40	0.125	0.2500	0.093750	6.00	8.3333	1.6667
40	H	4	2	230.908	60	0.375	10	50	160	0.500	0.0625	0.187500	0.75	66.6667	13.3333
41	H	4	4	178.845	60	0.375	10	50	160	0.250	0.0625	0.093750	1.50	33.3333	6.6667
42	H	4	8	190.029	60	0.375	10	50	160	0.125	0.0625	0.046875	3.00	16.6667	3.3333
43	J	1	2	68.574	15	1.500	0	20	10	0.500	1.0000	0.750000	3.00	6.6667	0.0000
44	J	1	2	60.785	15	1.500	0	35	10	0.500	1.0000	0.750000	3.00	11.6667	0.0000
45	J	1	4	59.578	15	1.500	0	20	10	0.250	1.0000	0.375000	6.00	3.3333	0.0000
46	J	1	4	56.694	15	1.500	0	35	10	0.250	1.0000	0.375000	6.00	5.8333	0.0000
47	J	1	8	63.818	15	1.500	0	20	10	0.125	1.0000	0.187500	12.00	1.6667	0.0000
48	J	1	8	63.775	15	1.500	0	35	10	0.125	1.0000	0.187500	12.00	2.9167	0.0000

BERNSTEIN, ELLIS, HYBRID, AND JOHNSON & THOMAS MODELS
MODEL MEAN RESPONSE TIME = B FAC1 OVER_B OVER_HSQ FAC2 B_X_F FAC3 FAC4
FAC1 = B/F OVER_B = 1/B OVER_HSQ = 1/H**2
FAC2 = F/B B_X_F = B*F FAC3 = M/(B*F)
FAC4 = P/(B*F) H = NUMBER OF HOSTS
B = NUMBER OF BACK-ENDS F = SEC PER REQUEST
P = PERCENT PRIORITY REQUESTS M = PERCENT MODIFICATIONS

OBS	ALGOR	H	B	RESP_TI	R	F	P	M	FAC1	OVER_B	OVER_HSQ	FAC2	B_X_F	FAC3	FAC4
49	J	4	2	208.942	60	0.375	10	50	160	0.500	0.0625	0.187500	0.75	66.6667	13.3333
50	J	4	4	182.203	60	0.375	10	50	160	0.250	0.0625	0.093750	1.50	33.3333	6.6667
51	J	4	8	182.747	60	0.375	10	50	160	0.125	0.0625	0.046875	3.00	16.6667	3.3333
52	J	2	2	114.477	30	0.750	10	50	40	0.500	0.2500	0.375000	1.50	33.3333	6.6667
53	J	2	4	105.402	30	0.750	10	50	40	0.250	0.2500	0.187500	3.00	16.6667	3.3333
54	J	2	8	109.435	30	0.750	10	50	40	0.125	0.2500	0.093750	6.00	8.3333	1.6667
55	J	1	2	79.509	15	1.500	10	50	10	0.500	1.0000	0.750000	3.00	16.6667	3.3333
56	J	1	4	75.864	15	1.500	10	50	10	0.250	1.0000	0.375000	6.00	8.3333	1.6667
57	J	1	8	53.809	15	1.500	10	50	10	0.125	1.0000	0.187500	12.00	4.1667	0.8333

BERNSTEIN, ELLIS, HYBRID, AND JOHNSON & THOMAS MODELS

MODEL MEAN RESPONSE TIME = B FAC1 OVER_B OVER_HSQ FAC2 B_X_F FAC3 FAC4

FAC1 = B/F OVER_B = 1/B OVER_HSQ = 1/H**2

FAC2 = F/B B_X_F = B*F FAC3 = M/(B*F)

FAC4 = P/(B*F) H = NUMBER OF HOSTS

B = NUMBER OF BACK-ENDS F = SEC PER REQUEST

P = PERCENT PRIORITY REQUESTS M = PERCENT MODIFICATIONS

ALGOR=B

GENERAL LINEAR MODELS PROCEDURE

CLASS LEVEL INFORMATION

CLASS	LEVELS	VALUES
ALGOR	1	B

NUMBER OF OBSERVATIONS IN BY GROUP = 12

BERNSTEIN, ELLIS, HYBRID, AND JOHNSON & THOMAS MODELS

MODEL MEAN RESPONSE TIME = B FAC1 OVER_B OVER_HSQ FAC2 B_X_F FAC3 FAC4

FAC1 = B/F OVER_B = 1/B OVER_HSQ = 1/H**2

FAC2 = F/B B_X_F = B*F FAC3 = M/(B*F)

FAC4 = P/(B*F) H = NUMBER OF HOSTS

B = NUMBER OF BACK-ENDS F = SEC PER REQUEST

P = PERCENT PRIORITY REQUESTS M = PERCENT MODIFICATIONS

ALGOR=B

GENERAL LINEAR MODELS PROCEDURE

DEPENDENT VARIABLE: RESP_TI

SOURCE	DF	SUM OF SQUARES	MEAN SQUARE	F VALUE	PR > F	R-SQUARE	C.V.
MODEL	8	31075.89560680	3884.48695085	176.82	0.0006	0.997884	3.5076
ERROR	3	65.90689886	21.96896629		STD DEV		RESP_TI MEAN
CORRECTED TOTAL	11	31141.80250567			4.68710639		133.62816667

SOURCE	DF	TYPE I SS	F VALUE	PR > F	DF	TYPE IV SS	F VALUE	PR > F
B	1	100.76754006	4.59	0.1217	1	372.02113290	16.93	0.0260
FAC1	1	29116.37230890	1325.34	0.0001	1	273.13654380	12.43	0.0387
OVER_B	1	188.02569233	8.56	0.0612	1	13.22954438	0.60	0.4943
OVER_HSQ	1	598.25238679	27.23	0.0137	1	12.93449549	0.59	0.4988
FAC2	1	664.34775314	30.24	0.0118	1	0.82188163	0.04	0.8590
B_X_F	1	189.50240555	8.63	0.0607	1	405.03968772	18.44	0.0232
FAC3	1	208.24885169	9.48	0.0542	1	113.66099149	5.17	0.1075
FAC4	1	10.37826834	0.47	0.5413	1	10.37826834	0.47	0.5413

PARAMETER	ESTIMATE	T FOR H0: PARAMETER=0	PR > |T|	STD ERROR OF ESTIMATE
INTERCEPT	79.13563680	4.82	0.0170	16.42706392
B	9.00333668	4.12	0.0260	2.18788503
FAC1	0.33138173	3.53	0.0387	0.09398174
OVER_B	-51.74670317	-0.78	0.4943	66.68298820
OVER_HSQ	4.96598796	0.77	0.4988	6.47155854
FAC2	-5.94198065	-0.19	0.8590	30.72060916
B_X_F	-6.86593821	-4.29	0.0232	1.59902742
FAC3	1.42773218	2.27	0.1075	0.62769091
FAC4	1.33701028	0.69	0.5413	1.94525755

BERNSTEIN, ELLIS, HYBRID, AND JOHNSON & THOMAS MODELS

MODEL MEAN RESPONSE TIME = B FAC1 OVER_B OVER_HSQ FAC2 B_X_F FAC3 FAC4

FAC1 = B/F OVER_B = 1/B OVER_HSQ = 1/H**2

FAC2 = F/B B_X_F = B*F FAC3 = M/(B*F)

FAC4 = P/(B*F) H = NUMBER OF HOSTS

B = NUMBER OF BACK-ENDS F = SEC PER REQUEST

P = PERCENT PRIORITY REQUESTS M = PERCENT MODIFICATIONS

ALGOR=E

GENERAL LINEAR MODELS PROCEDURE

CLASS LEVEL INFORMATION

CLASS	LEVELS	VALUES
ALGOR	1	E

NUMBER OF OBSERVATIONS IN BY GROUP = 15

BERNSTEIN, ELLIS, HYBRID, AND JOHNSON & THOMAS MODELS

MODEL MEAN RESPONSE TIME = B FAC1 OVER_B OVER_HSQ FAC2 B_X_F FAC3 FAC4

FAC1 = B/F OVER_B = 1/B OVER_HSQ = 1/H**2

FAC2 = F/B B_X_F = B*F FAC3 = M/(B*F)

FAC4 = P/(B*F) H = NUMBER OF HOSTS

B = NUMBER OF BACK-ENDS F = SEC PER REQUEST

P = PERCENT PRIORITY REQUESTS M = PERCENT MODIFICATIONS

ALGOR=E

GENERAL LINEAR MODELS PROCEDURE

DEPENDENT VARIABLE: RESP_TI

SOURCE	DF	SUM OF SQUARES	MEAN SQUARE	F VALUE	PR > F	R-SQUARE	C.V.
MODEL	8	51687.81563128	6460.97695391	134.28	0.0001	0.994446	6.0922
ERROR	6	288.69038512	48.11506419		STD DEV		RESP_TI MEAN
CORRECTED TOTAL	14	51976.50601640			6.93650230		113.85920000

SOURCE	DF	TYPE I SS	F VALUE	PR > F	DF	TYPE IV SS	F VALUE	PR > F
B	1	84.77633897	1.76	0.2327	1	80.26012809	1.67	0.2440
FAC1	1	50285.09999307	1045.10	0.0001	1	2051.69309853	42.64	0.0006
OVER_B	1	12.30939023	0.26	0.6310	1	76.40325814	1.59	0.2544
OVER_HSQ	1	1201.15472711	24.96	0.0025	1	219.11830695	4.55	0.0768
FAC2	1	6.51912858	0.14	0.7254	1	70.64999271	1.47	0.2712
B_X_F	1	47.72124700	0.99	0.3578	1	47.72124700	0.99	0.3578
FAC3	1	48.61387952	1.01	0.3536	1	6.87348328	0.14	0.7185
FAC4	1	1.62092680	0.03	0.8604	1	1.62092680	0.03	0.8604

PARAMETER	ESTIMATE	T FOR H0: PARAMETER=0	PR > |T|	STD ERROR OF ESTIMATE
INTERCEPT	145.05242827	4.87	0.0028	29.77732600
B	-6.43088333	-1.29	0.2440	4.97921915
FAC1	0.75011348	6.53	0.0006	0.11487124
OVER_B	-168.45220745	-1.26	0.2544	133.67841536
OVER_HSQ	-79.16173356	-2.13	0.0768	37.09509595
FAC2	107.21754881	1.21	0.2712	88.48106641
B_X_F	4.06155556	1.00	0.3578	4.01828001
FAC3	0.47112568	0.38	0.7185	1.24649022
FAC4	0.72173694	0.18	0.8604	3.93221961

BERNSTEIN, ELLIS, HYBRID, AND JOHNSON & THOMAS MODELS

MODEL MEAN RESPONSE TIME = B FAC1 OVER_B OVER_HSQ FAC2 B_X_F FAC3 FAC4

FAC1 = B/F OVER_B = 1/B OVER_HSQ = 1/H**2

FAC2 = F/B B_X_F = B*F FAC3 = M/(B*F)

FAC4 = P/(B*F) H = NUMBER OF HOSTS

B = NUMBER OF BACK-ENDS F = SEC PER REQUEST

P = PERCENT PRIORITY REQUESTS M = PERCENT MODIFICATIONS

ALGOR=H

GENERAL LINEAR MODELS PROCEDURE

CLASS LEVEL INFORMATION

CLASS	LEVELS	VALUES
ALGOR	1	H

NUMBER OF OBSERVATIONS IN BY GROUP = 15

BERNSTEIN, ELLIS, HYBRID, AND JOHNSON & THOMAS MODELS

MODEL MEAN RESPONSE TIME = B FAC1 OVER_B OVER_HSQ FAC2 B_X_F FAC3 FAC4

FAC1 = B/F OVER_B = 1/B OVER_HSQ = 1/H**2

FAC2 = F/B B_X_F = B*F FAC3 = M/(B*F)

FAC4 = P/(B*F) H = NUMBER OF HOSTS

B = NUMBER OF BACK-ENDS F = SEC PER REQUEST

P = PERCENT PRIORITY REQUESTS M = PERCENT MODIFICATIONS

ALGER=H

GENERAL LINEAR MODELS PROCEDURE

DEPENDENT VARIABLE: RESP_TI

SOURCE	DF	SUM OF SQUARES	MEAN SQUARE	F VALUE	PR > F	R-SQUARE	C.V.
MODEL	8	41457.03628302	5182.12953538	2170.76	0.0001	0.999965	1.5009
ERROR	6	14.32343458	2.38723910		STD DEV		RESP_TI MEAN
CORRECTED TOTAL	14	41471.35971760			1.54506929		102.94260000

SOURCE	DF	TYPE I SS	F VALUE	PR > F	DF	TYPE IV SS	F VALUE	PR > F
B	1	426.32865617	178.59	0.0001	1	598.17049286	250.57	0.0001
FAC1	1	38640.85422170	16186.42	0.0001	1	410.55607426	172.0[illegible]	0.0001
OVER_B	1	462.85828663	193.91	0.0001	1	80.87379417	33.88	0.0011
OVER_HSQ	1	577.95409456	242.10	0.0001	1	239.55850180	100.35	0.0001
FAC2	1	569.31255344	238.48	0.0001	1	112.02982339	46.93	0.0005
B_X_F	1	455.83395089	190.95	0.0001	1	455.83395089	190.95	0.0001
FAC3	1	299.32861774	125.39	0.0001	1	24.85807422	10.41	0.0180
FAC4	1	24.52590189	10.27	0.0185	1	24.52590189	10.27	0.0185

PARAMETER	ESTIMATE	T FOR H0: PARAMETER=0	PR > |T|	STD ERROR OF ESTIMATE
INTERCEPT	-11.35832609	-1.71	0.1377	6.63274224
B	17.55630833	15.83	0.0001	1.10909479
FAC1	0.33556617	13.11	0.0001	0.02558696
OVER_B	173.31042058	5.82	0.0011	29.77616252
OVER_HSQ	82.77174692	10.02	0.0001	8.26273689
FAC2	-135.01332697	-6.85	0.0005	19.70869064
B_X_F	-12.55277778	-13.82	0.0001	0.90841535
FAC3	0.89594651	3.23	0.0180	0.27754912
FAC4	2.80743509	3.21	0.0185	0.87587117

BERNSTEIN, ELLIS, HYBRID, AND JOHNSON & THOMAS MODELS

MODEL MEAN RESPONSE TIME = B FAC1 OVER_B OVER_HSQ FAC2 B_X_F FAC3 FAC4

FAC1 = B/F OVER_B = 1/B OVER_HSQ = 1/H**2

FAC2 = F/B B_X_F = B*F FAC3 = M/(B*F)

FAC4 = P/(B*F) H = NUMBER OF HOSTS

B = NUMBER OF BACK-ENDS F = SEC PER REQUEST

P = PERCENT PRIORITY REQUESTS M = PERCENT MODIFICATIONS

ALGOR=J

GENERAL LINEAR MODELS PROCEDURE

CLASS LEVEL INFORMATION

CLASS	LEVELS	VALUES
ALGOR	1	J

NUMBER OF OBSERVATIONS IN BY GROUP = 15

BERNSTEIN, ELLIS, HYBRID, AND JOHNSON & THOMAS MODELS

MODEL MEAN RESPONSE TIME = B FAC1 OVER_B OVER_HSQ FAC2 B_X_F FAC3 FAC4

FAC1 = B/F OVER_B = 1/B OVER_HSQ = 1/H**2

FAC2 = F/B B_X_F = B*F FAC3 = M/(B*F)

FAC4 = P/(B*F) H = NUMBER OF HOSTS

B = NUMBER OF BACK-ENDS F = SEC PER REQUEST

P = PERCENT PRIORITY REQUESTS M = PERCENT MODIFICATIONS

ALGOR=J

GENERAL LINEAR MODELS PROCEDURE

DEPENDENT VARIABLE: RESP_TI

SOURCE	DF	SUM OF SQUARES	MEAN SQUARE	F VALUE	PR > F	R-SQUARE	C.V.
MODEL	8	37341.78141220	4667.72267652	118.10	0.0001	0.993690	6.3477
ERROR	6	237.14178620	39.52363103		STD DEV		RESP_TI MEAN
CORRECTED TOTAL	14	37578.92319840			6.28678225		99.04080000

SOURCE	DF	TYPE I SS	F VALUE	PR > F	DF	TYPE IV SS	F VALUE	PR > F
B	1	276.50504267	7.00	0.0383	1	74.28212271	1.88	0.2195
FAC1	1	35552.62675121	899.53	0.0001	1	807.06041059	20.42	0.0040
OVER_B	1	139.83048893	3.54	0.1090	1	0.70759004	0.02	0.8979
OVER_HSQ	1	932.94601430	23.60	0.0028	1	0.00440559	0.00	0.9919
FAC2	1	81.70095007	2.07	0.2005	1	0.03435545	0.00	0.9774
B_X_F	1	53.67014251	1.36	0.2881	1	53.67014251	1.36	0.2881
FAC3	1	188.78138483	4.78	0.0715	1	3.32802308	0.08	0.7814
FAC4	1	115.72063767	2.93	0.1379	1	115.72063767	2.93	0.1379

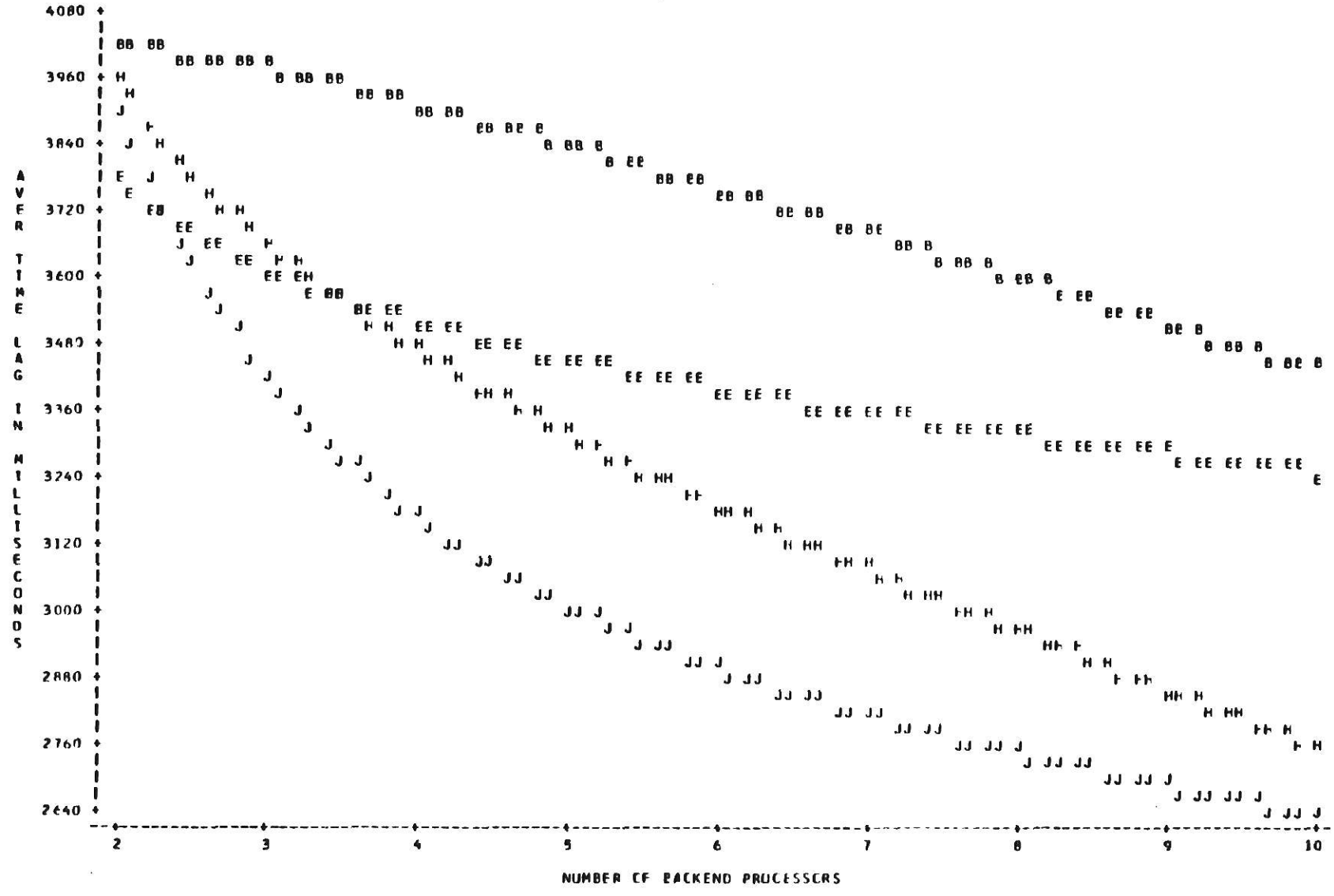
PARAMETER	ESTIMATE	T FOR HO: PARAMETER=0	PR > |T|	STD ERROR OF ESTIMATE
INTERCEPT	55.51611866	2.06	0.0854	26.98817882
B	6.18675417	1.37	0.2155	4.51283157
FAC1	0.47046118	4.52	0.0040	0.10411162
OVER_B	16.21106783	0.13	0.8979	121.15718804
OVER_HSQ	0.35511998	0.01	0.9919	33.62051662
FAC2	-2.36432766	-0.03	0.9774	80.19332704
B_X_F	-4.30727778	-1.17	0.2881	3.69628053
FAC3	-0.32782457	-0.29	0.7814	1.12973546
FAC4	6.09821369	1.71	0.1379	3.56390114

```
1                                   S T A T I S T I C A L   A N A L Y S I S   S Y S T E M

NOTE: THE JOB SPEC0072 HAS BEEN RUN UNDER RELEASE 76.6C OF SAS AT KANSAS STATE UNIVERSITY.


1          DATA PROGRAM:
2          *  1 HOST CPU, VARIABLE BACK-END PROCESSORS (1-8);
3          *  STREAM OF REQUESTS ENTER DATA BASE SYSTEM AT CONSTANT RATE ;
4          *  RATE OF REQUESTS PER HOST IS CONSTANT;
5          *  MODIFICATION REQUESTS   50%  ;
6          *  HIGH PRIORITY REQUESTS   10%  ;
7          *  DEFINITION OF CONSTANTS AND PARAMETERS;
8          *  H  NUMBER OF HOSTS;
9          *  R  TOTAL NUMBER OF REQUESTS IN THE WORK LOAD;
10         *  F  SECONDS PER REQUEST;
11         *  P  PRIORITY IN PERCENT;
12         *  M  MODIFICATION REQUESTS IN PERCENT;
13         *  S  DURATION OF THE JOB STREAM IN SECONDS;
14         *  ;

NOTE: DATA SET WORK.PROGRAM HAS 1 OBSERVATIONS AND 0 VARIABLES. 3256 OBS/TRK.
NOTE: THE DATA STATEMENT USED 0.40 SECONDS AND 102K.

15         DATA PROGRAM;
16         H=1;
17         P=10;
18         M = 50;
19         R=15;
20         F=3/2;
21         S=22.5;
22         *
23         *BERNSTEIN MODEL;
24         *          ;
25         LABEL D_TIME=AVER TIME LAG IN MILLISECONDS;
26         LABEL BACK_END= NUMBER OF BACKEND PROCESSORS;
27         C1=79.13563680 ;
28         C2= 9.00333668B;
29         C3= 0.33138173 ;
30         C4=-51.74670317;
31         C5= 4.96598796 ;
32         C6= -5.94198065;
33         C7= -6.86593821;
34         C8= 1.42773218 ;
35         C9= 1.33701028 ;
36         BACK_END=2;
37         LOOP:
38         B=BACK_END;
39         RESP_TI=  C1 + C2*B +C3*R/F+ (C4/B) + (C5/H**2) + (C6*F/B) + (C7*B*F) +
40          C8*M/(B*F)  + (C9*P)/(B*F) ;
41         D_TIME=(RESP_TI - S)*1000/R;
42         BACK_END=B;
43         OUTPUT;  BACK_END = BACK_END + 0.1;
44         IF BACK_END <=10 THEN GO TO LOOP;
45         *
46         *       ;
47         *  ELLIS ;
48         *;
49         C1=   145.05242827;
```

```
50        C2=     -6.43088333;
51        C3=      0.75011348;
52        C4=   -168.45220745;
53        C5=    -79.16173356;
54        C6=    107.21754881;
55        C7=      4.06155556;
56        C8=      0.47112568;
57        C9=      0.72173654;
58        BE_E=2;
59        LOOPE:
60        B=BE_E;
61        RESP_TI=  C1 + C2*B +C3*R/F+ (C4/B) + (C5/H**2) + (C6*F/B) + (C7*B*F) +
62         C8*M/(B*F)  + (C9*P)/(B*F) ;
63        D_TIME_E= (RESP_TI - S)*1000/R;
64        BE_E=B;
65        OUTPUT;  BE_E= BE_E + 0.1;
66        IF BE_E <=10 THEN GO TO LOOPE;
67        * JOHNSON & THOMAS MODEL;
68        *                 ;
69        C1= 55.51611866;
70        C2=  6.18675417;
71        C3=  0.47046118;
72        C4= 16.21106783;
73        C5=  0.35511998;
74        C6= -2.36432766;
75        C7= -4.30727778;
76        C8= -0.32782457;
77        C9=   6.09821369;
78        BE_J=2;
79        LOOPJ:
80        B=BE_J;
81        RESP_TI=  C1 + C2*B +C3*R/F+ (C4/B) + (C5/H**2) + (C6*F/B) + (C7*B*F) +
82         C8*M/(B*F)  + (C9*P)/(B*F) ;
83        D_TIME_J= (RESP_TI - S)*1000/R;
84        BE_J=B;
85        OUTPUT;  BE_J = BE_J + 0.1;
86        IF BE_J <= 10 THEN GO TO LOOPJ;
87        *;
88        *  HYBRID;
89        *;
90        BE_H=2;
91        C1=-11.35832609;
92        C2= 17.55630833;
93        C3=  0.33556617;
94        C4=173.31042058;
95        C5= 82.77174692;
96        C6=-135.01332697;
97        C7= -12.55277778;
98        C8=  0.89594651;
99        C9=  2.80743509;
100       LOOPH:
101       B=BE_H;
102       RESP_TI=  C1 + C2*B +C3*R/F+ (C4/B) + (C5/H**2) + (C6*F/B) + (C7*B*F) +
103        C8*M/(B*F)  + (C9*P)/(B*F) ;
104       D_TIME_H= (RESP_TI - S)*1000/R;
105       BE_H=B;
```

```
3                                   S T A T I S T I C A L   A N A L Y S I S   S Y S T E M

106         OUTPUT:  BE_H = BE_H + 0.1;
107         IF BE_H <=10 THEN GO TO LOOPH;

NOTE: DATA SET WORK.PROGRAM HAS 324 OBSERVATIONS AND 25 VARIABLES. 63 OBS/TRK.
NOTE: THE DATA STATEMENT USED 1.24 SECONDS AND 102K.

108         PROC PLOT   NOLEGEND;
109         TITLE1 BERNSTEIN, JOHNSON & THOMAS, HYBRID, AND ELLIS MODELS;
110         TITLE2 1 HOST, 50% MODIFICATIONS, 10% PRIORITY, 1 REC PER 3/2 SEC PER HOST;
111         PLOT D_TIME*BACK_END='B' C_TIME_E*BE_E='E' D_TIME_H*BE_H='H'
112         D_TIME_J*BE_J='J' / OVERLAY HAXIS=2 TO 10 BY 1;

NOTE: THE PROCEDURE PLOT USED 2.43 SECONDS AND 146K AND PRINTED PAGE 1.

NOTE: SAS USED 146K MEMORY.

NOTE: BARR, GOODNIGHT, SALL AND HELWIG
      SAS INSTITUTE INC.
      P.O. BOX 10066
      RALEIGH, N.C. 27605
```

1 HOST, 50% MODIFICATIONS, 10% PRIORITY, 1 REQ PER 3/2 SEC PER HOST



```
1                                   S T A T I S T I C A L   A N A L Y S I S   S Y S T E M

NOTE: THE JOB SPEC0097 HAS BEEN RUN UNDER RELEASE 76.6C OF SAS AT KANSAS STATE UNIVERSITY.

1          DATA PROGRAM;
2          *  HOST CPU VS  LAG TIME OF NUMBER OF BACKENDS GIVING MIN DELAY;
3          *  CONSTANT WORK LOAD PER HOST;
4          *  MODIFICATION REQUESTS   50%  ;
5          *  HIGH PRIORITY REQUESTS   10% ;
6          *  NUMBER OF BACKENDS WITH MIN TIME LAG PLOTTED AGAINST NUMBER OF HOSTS;
7          *  DEFINITION OF VARIABLES AND CONSTANTS:
8          *  H  NUMBER OF HOST CPU;
9          *  B  NUMBER OF BACK-END PROCESSORS;
10         *  M  MODIFICATION REQUESTS IN PERCENT;
11         *  P  PRIORITY REQUESTS IN PERCENT;
12         *  S  DURATION OF THE JOB STREAM IN SECONDS;
13         *  F  INTERVAL BETWEEN REQUESTS IN SECONDS PER REQUEST;
14         *  R  TOTAL NUMBER OF REQUESTS SUBMITTED IN THE JOB STREAM;
15         *;
16         *  THE NUMBER OF HOST CPU ARE INITIATED TO 1     ;
17         H=1;
18         *  THERE ARE 50% MODIFICATIONS AND 10% HIGH PRIORITY REQUESTS    ;
19         M=50;
20         P=10;
21         S = 22.5 ;
22         *
23         *BERNSTEIN MODEL;
24         *          ;
25         LABEL TIME_B=TIME LAG IN MILLISECONDS;
26         LABEL HOST=NUMBER OF HOST CPU;
27         *  C1-C9 ARE CONSTANTS ;
28         C1=79.13563680 ;
29         C2= 9.003336688;
30         C3= 0.33138173 ;
31         C4=-51.74670317;
32         C5= 4.96598756 ;
33         C6= -5.94198065;
34         C7= -6.86593821;
35         C8= 1.42773218 ;
36         C9= 1.33701028 ;
37         HOST=1;
38         *  THE NUMBER OF BACKEND PROCESSORS THAT GIVE THE MINIMUM LAG TIME    ;
39         *  IS CALCULATED FOR EACH VALUE OF HOST;
40         LOOP:
41         H=HOST;
42         B=2;
43         F=(3/2)/H;
44         R=15*H;
45         RESP_TI = C1 + C2*B +C3*R/F+ (C4/B) + (C5/H**2) + (C6*F/B) + (C7*B*F) +
46          C8*M/(B*F)  + (C9*P)/(B*F) ;
47         NEW_LAG=(RESP_TI -S)*1000/R;
48         MIN_LAG = NEW_LAG;
49         MIN_BE=2;
50         LOOP_MIN:
51         F=(3/2)/H;
52         R=15*H;
53         B=B+0.1;
```

```
2                                   S T A T I S T I C A L   A N A L Y S I S   S Y S T E M

54        RESP_TI = C1 + C2*B +C3*R/F+ (C4/B) + (C5/H**2) + (C6*F/B) + (C7*B*F) +
55         C8*M/(B*F)  + (C9*P)/(B*F) ;
56        NEW_LAG=(RESP_TI -S)*1000/R;
57        IF NEW_LAG < MIN_LAG THEN MIN_BE = B;
58        IF NEW_LAG < MIN_LAG THEN MIN_LAG = NEW_LAG;
59        IF B<=9  THEN GO TO LOOP_MIN;
60        MIN_BE = MIN_BE + 0.5; MIN_BE = INT(MIN_BE);  B = MIN_BE;
61        RESP_TI = C1 + C2*B +C3*R/F+ (C4/B) + (C5/H**2) + (C6*F/B) + (C7*B*F) +
62         C8*M/(B*F)  + (C9*P)/(B*F) ;
63        MIN_LAG=(RESP_TI - S)*1000/R;
64        TIME_B = MIN_LAG;
65        HOST=H;
66        OUTPUT; HOST=HOST + 1;
67        IF HOST<=6 THEN GO TO LOOP;
68        *
69        *  ELLIS :
70        *        :
71        *;
72        M=50;
73        P=10;
74        S=22.5;
75        C1=    145.05242827;
76        C2=     -6.43088333;
77        C3=      0.75011348;
78        C4=   -168.45220745;
79        C5=    -79.16173356;
80        C6=    107.21754881;
81        C7=      4.06155556;
82        C8=      0.47112568;
83        C9=      0.72173694;
84        M_E=1;
85        LOOPE:
86        H=M_E;
87        B=2;
88        F=(3/2)/H;
89        R=15*H;
90        RESP_TI = C1 + C2*B +C3*R/F+ (C4/B) + (C5/H**2) + (C6*F/B) + (C7*B*F) +
91         C8*M/(B*F)  + (C9*P)/(B*F) ;
92        NEW_LAG=(RESP_TI -S)*1000/R;
93        MIN_LAG = NEW_LAG;
94        MIN_BE_E = 2;
95        MIN_E:
96        F=(3/2)/H;
97        R=15*H;
98        B=B+0.1;
99        RESP_TI = C1 + C2*B +C3*R/F+ (C4/B) + (C5/H**2) + (C6*F/B) + (C7*B*F) +
100        C8*M/(B*F)  + (C9*P)/(B*F) ;
101       NEW_LAG=(RESP_TI -S)*1000/R;
102       IF NEW_LAG < MIN_LAG THEN MIN_BE_E = B;
103       IF NEW_LAG < MIN_LAG THEN MIN_LAG = NEW_LAG;
104       IF B<=9  THEN GO TO MIN_E;
105       MIN_BE_E = MIN_BE_E + 0.5;  MIN_BE_E = INT(MIN_BE_E);  B=MIN_BE_E;
106       RESP_TI = C1 + C2*B +C3*R/F+ (C4/B) + (C5/H**2) + (C6*F/B) + (C7*B*F) +
107        C8*M/(B*F)  + (C9*P)/(B*F) ;
108       MIN_LAG=(RESP_TI - S)*1000/R;
109       TIME_E = MIN_LAG;
```

```
110       H_E=H;
111       OUTPUT; H_E=H_E+1;
112       IF H_E <=6 THEN GO TO LOOPE;
113       *;
114       *  HYBRID;
115       *;
116       LABEL TIME_H=TIME LAG IN MILLISECONDS;
117       M=50;
118       P=10;
119       S=22.5;
120       C1=-11.358326C9;
121       C2= 17.55630E23;
122       C3=  0.33556617;
123       C4=173.31042058;
124       C5= 82.77174692;
125       C6=-135.01322697;
126       C7= -12.55277778;
127       C8=  0.89594651;
128       C9=  2.80743509;
129       H_H=1;
130       LOOPH:
131       H=H_H;
132       B=2;
133       F=(3/2)/H;
134       R=15*H;
135       RESP_TI = C1 + C2*B +C3*R/F+ (C4/B) + (C5/H**2) + (C6*F/B) + (C7*B*F) +
136        C8*M/(B*F)  + (C9*P)/(B*F) ;
137       NEW_LAG=(RESP_TI -SI*1000/R;
138       MIN_LAG = NEW_LAG;
139       MIN_BE_H = 2;
140       MIN_H:
141       F=(3/2)/H;
142       R=15*H;
143       B=B+0.1;
144       RESP_TI = C1 + C2*B +C3*R/F+ (C4/B) + (C5/H**2) + (C6*F/B) + (C7*B*F) +
145        C8*M/(B*F)  + (C9*P)/(B*F) ;
146       NEW_LAG=(RESP_TI -SI*1000/R;
147       IF NEW_LAG < MIN_LAG THEN MIN_BE_H = B;
148       IF NEW_LAG < MIN_LAG THEN MIN_LAG = NEW_LAG;
149       IF B<=9  THEN GO TO MIN_H;
150       MIN_BE_H = MIN_BE_H + 0.5;  MIN_BE_H = INT(MIN_BE_H);  B=MIN_BE_H;
151       RESP_TI = C1 + C2*B +C3*R/F+ (C4/B) + (C5/H**2) + (C6*F/B) + (C7*B*F) +
152        C8*M/(B*F)  + (C9*P)/(B*F) ;
153       MIN_LAG=(RESP_TI - SI*1000/R;
154       TIME_H = MIN_LAG;
155       H_H=H;
156       OUTPUT;  H_H=H_H+1;
157       IF H_H <=6 THEN GO TO LOOPH;
158       * JOHNSON & THOMAS MODEL;
159       *              ;
160       LABEL TIME_J=TIME LAG IN MILLISECONDS;
161       M=50;
162       P=10;
163       S=22.5;
164       C1= 55.51611866;
165       C2=  6.18675417;
```

```
4                                   S T A T I S T I C A L   A N A L Y S I S   S Y S T E M

166        C3=  0.47046118;
167        C4= 16.21106783;
168        C5=  0.35511598;
169        C6= -2.36432766;
170        C7= -4.30727778;
171        C8= -0.32782457;
172        C9=   6.09821369;
173        H_J=1;
174        LOOPJ:
175        H=H_J;
176        B=2;
177        F=(3/2)/H;
178        R=15*H;
179        RESP_TI = C1 + C2*B +C3*R/F+ (C4/B) + (C5/H**2) + (C6*F/B) + (C7*B*F) +
180         C8*M/(B*F)  + (C9*P)/(B*F) ;
181        NEW_LAG=(RESP_TI -S)*1000/R;
182        MIN_LAG = NEW_LAG;
183        MIN_BE_J = 2;
184        MIN_J:
185        F=(3/2)/H;
186        R=15*H;
187        B=B+0.1;
188        RESP_TI = C1 + C2*B +C3*R/F+ (C4/B) + (C5/H**2) + (C6*F/B) + (C7*B*F) +
189         C8*M/(B*F)  + (C9*P)/(B*F) ;
190        NEW_LAG=(RESP_TI -S)*1000/R;
191        IF NEW_LAG < MIN_LAG THEN MIN_BE_J = B;
192        IF NEW_LAG < MIN_LAG THEN MIN_LAG = NEW_LAG;
193        IF B<=9  THEN GO TO MIN_J;
194        MIN_BE_J = MIN_BE_J + 0.5;  MIN_BE_J = INT(MIN_BE_J);  B=MIN_BE_J;
195        RESP_TI = C1 + C2*B +C3*R/F+ (C4/B) + (C5/H**2) + (C6*F/B) + (C7*B*F) +
196         C8*M/(B*F)  + (C9*P)/(B*F) ;
197        MIN_LAG=(RESP_TI - S)*1000/R;
198        TIME_J = MIN_LAG;
199        H_J=H;
200        OUTPUT; H_J=H_J+1;
201        IF H_J <=6 THEN GO TO LOOPJ;

NOTE: DATA SET WORK.PROGRAM HAS 24 OBSERVATIONS AND 31 VARIABLES. 51 OBS/TRK.
NOTE: THE DATA STATEMENT USED 2.52 SECONDS AND 102K.

202        PROC PLOT   NOLEGEND;
203        TITLE1 BERNSTEIN, ELLIS, HYBRID, AND JOHNSON &THOMAS MODELS;
204        TITLE2 LAG TIME FOR THE NUMBER OF BACKENDS GIVING MINIMUM DELAY;
205        TITLE3 50% MOD, 10% PRIORITY  DIGITS REPRESENT NUMBER OF BACKENDS;
206        TITLE4 WORK LOAD IS 3/2 REQUEST PER HOST;
207        PLOT TIME_B*HOST=MIN_BE                              TIME_H*H_H=MIN_BE_H
208        TIME_J*H_J=MIN_BE_J TIME_E*H_E=MIN_BE_E
209        / OVERLAY HAXIS=1 TO 6 BY 1  ;

NOTE: THE PROCEDURE PLOT USED 0.83 SECONDS AND 144K AND PRINTED PAGE 1.

NOTE: SAS USED 144K MEMORY.
```

LAG TIME FOR THE NUMBER OF BACKENDS GIVING MINIMUM DELAY
50% MOD, 10% PRIORITY DIGITS REPRESENT NUMBER OF BACKENDS
WORK LOAD IS 3/2 REQUEST PER HOST

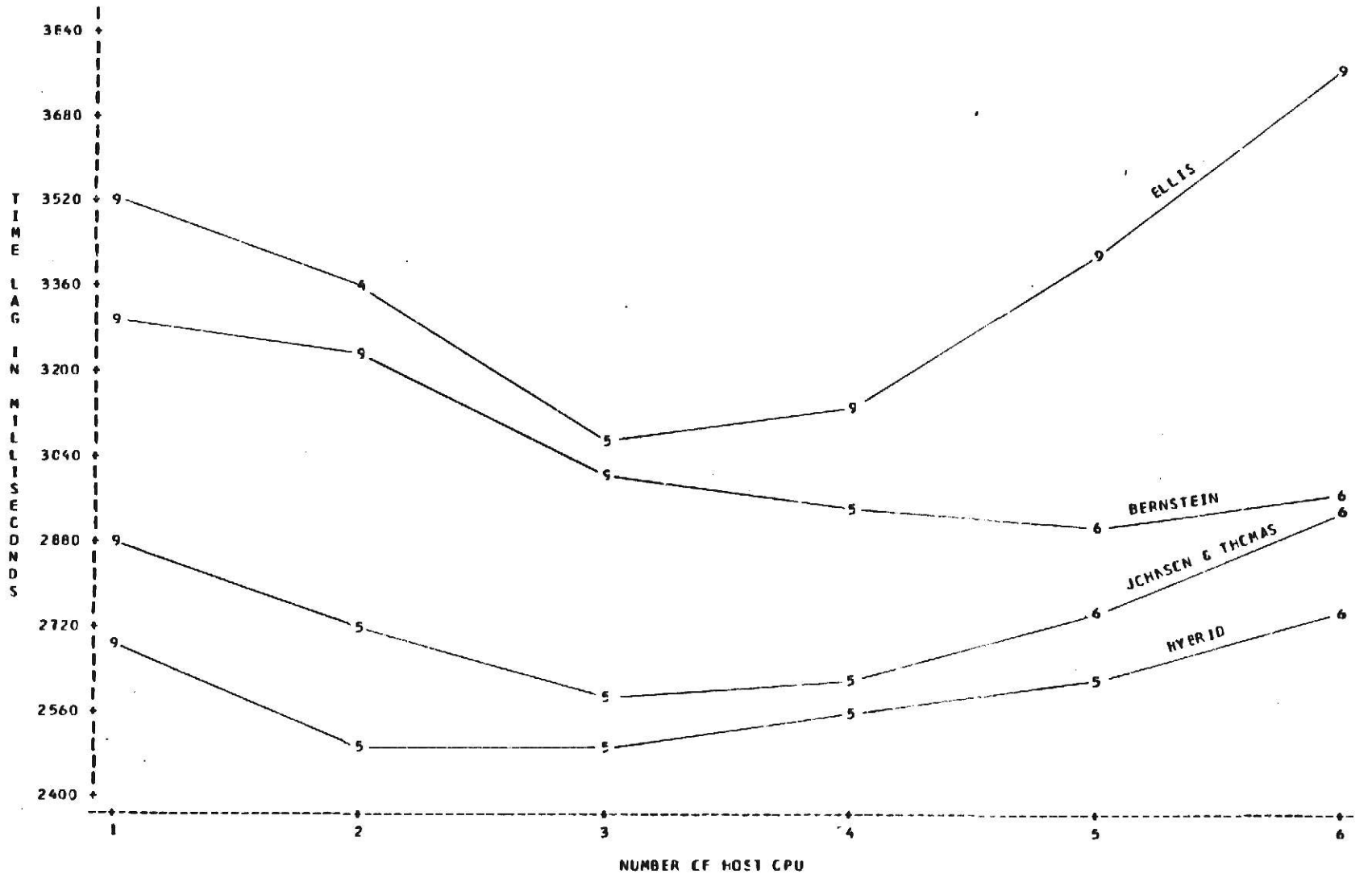


TABLE IX.I, BERNSTEIN DATA I

M=50, P=10
Units are seconds
S = Standard deviation

		H=1, R=15, f=3/2	H=2, R=30, f=3/4	H=4, R=60, f=3/8
B=2	t_R	108.803	121.075	225.072
		66.576	127.326	217.750
		70.800	128.101	264.013
		105.094	152.404	185.936
		74.792	106.708	254.485
		64.977	116.064	216.956
	$\bar{t}_R$	81.840	125.280	232.369
	S	18.059	14.115	27.728
B=4	t_R	99.095	112.106	185.268
		70.388	130.904	214.287
		84.078	114.372	208.217
		94.167	124.656	165.402
		81.430	142.928	246.381
		69.464	122.708	167.965
	$\bar{t}_R$	83.104	124.612	197.920
	S	11.027	10.328	28.394
B=8	t_R	86.680	126.042	195.807
		74.646	143.994	210.682
		77.754	123.133	210.682
		69.432	101.867	214.370
		72.589	137.811	192.165
		76.932	142.985	223.411
	$\bar{t}_R$	76.339	129.305	207.853
	S	53.840	14.588	10.737

TABLE IX.I BERNSTEIN DATA II

Units are seconds
S = Standard deviation

B=2	t_R	110.033
H=1		76.238
R=15		69.987
f=3/2		73.397
P=0		70.458
M=50		69.625
	$\bar{t}_R$	78.290
	S	14.383
B=2	t_R	131.100
H=1		119.184
R=30		166.153
f=3/4		122.569
P=10		139.893
M=50		123.476
	$\bar{t}_R$	133.729
	S	16.008
B=2	t_R	150.815
H=4		135.541
R=30		135.446
f=3/4		111.609
P=10		127.630
M=50		136.338
	$\bar{t}_R$	132.897
	S	11.742

TABLE IX.II CODASYL DATA

M=50, P=10
Units are seconds
S = Standard deviation

		H=2, R=30, f=3/4	H=4, R=60, f=3/8
B=2	t_R	130.966	295.914
		150.792	357.353
		154.580	305.901
		141.619	277.464
		141.619	277.464
		151.094	344.466
		142.648	322.283
	$\bar{t}_R$	145.283	317.230
	S	7.915	27.511
B=4	t_R	173.178	336.444
		128.968	366.070
		179.501	318.248
		140.167	278.568
		117.555	299.617
		174.525	342.302
	$\bar{t}_R$	152.316	323.542
	S	24.388	28.723
B=8	t_R	208.049	
		172.486	
		222.978	
		184.207	
		160.441	
		198.450	
	$\bar{t}_R$	191.103	
	S	21.182	

TABLE IX.III ELLIS DATA

M=50, P=10
Units are seconds
S = Standard deviation

		H=2, R=30, f=3/4	H=4, R=60, f=3/8
B=2	t_R	125.419	228.233
		141.619	256.832
		140.415	221.551
		100.856	227.159
		93.783	223.211
		132.269	216.708
	$\bar{t}_R$	122.394	228.949
	S	18.635	13.030
B=4	t_R	149.382	253.247
		113.243	245.883
		154.232	235.444
		110.210	183.012
		114.592	207.593
		129.702	254.161
	$\bar{t}_R$	128.560	229.890
	S	17.608	26.191
B=8	t_R	124.085	222.675
		130.044	206.891
		121.274	246.845
		144.696	182.532
		92.682	201.447
		145.186	214.074
	$\bar{t}_R$	126.328	212.411
	S	11.225	19.737

TABLE IX.IV HYBRID DATA

M=50, P=10
Units are seconds
S = Standard deviation

		H=2, R=30, f=3/4	H=4, R=60, f=3/8
B=2	t_R	125.410	203.755
		110.522	232.493
		138.171	236.550
		149.829	234.567
		110.066	263.313
		115.066	214.772
	$\bar{t}_R$	124.844	230.908
	S	14.840	18.688
B=4	t_R	88.680	174.587
		84.242	158.409
		102.984	195.134
		99.170	162.236
		96.510	165.982
		107.947	216.724
	$\bar{t}_R$	96.589	178.845
	S	8.078	20.725
B=8	t_R	124.427	171.049
		108.333	181.946
		106.263	198.536
		119.476	193.412
		99.626	197.095
		95.179	198.136
	$\bar{t}_R$	108.884	190.029
	S	10.283	10.207

TABLE IX.V JOHNSON AND THOMAS DATA

M=50, P=10
Units are seconds
S = Standard deviation

		H=2, R=30, f=3/4	H=4, R=60, f=3/8
B=2	t_R	115.550	258.967
		110.099	173.876
		94.905	187.884
		135.467	201.718
		126.467	204.634
		104.376	226.576
	$\bar{t}_R$	114.477	208.942
	S	13.477	27.560
B=4	t_R	98.306	201.824
		86.670	171.453
		115.497	203.692
		120.272	155.219
		85.806	185.286
		125.862	175.745
	$\bar{t}_R$	105.402	182.203
	S	15.952	17.038
B=8	t_R	85.760	217.434
		100.415	135.777
		123.200	202.299
		87.355	184.900
		124.489	160.672
		135.392	195.400
	$\bar{t}_R$	109.435	182.747
	S	19.232	27.214

TABLE X DATA FROM NORSWORTHY MASTER'S REPORT (15)

H=1, R=15, P=0, f-3/2 sec
Units are seconds
S = Standard deviation

MODEL	M	B	$\bar{t}_R$
CODASYL	50	2	86.066
		4	93.641
		8	98.029
	35	2	88.520
		4	75.141
		8	84.298
	20	2	78.990
		4	68.245
		8	77.068
Hybrid	50	2	82.648
		4	74.143
		8	66.599
	35	2	69.179
		4	66.779
		8	64.166
	20	2	61.346
		4	66.908
		8	62.362

MODEL	M	B	$\bar{t}_R$
Ellis	50	2	72.547
		4	82.234
		8	78.655
	35	2	74.978
		4	69.110
		8	64.039
	20	2	78.942
		4	68.969
		8	69.882
Johnson and Thomas	50	2	79.509
		4	75.864
		8	53.809
	35	2	60.785
		4	56.694
		8	63.775
	20	2	68.574
		4	59.578
		8	63.818

A SIMULATION STUDY COMPARING FIVE CONSISTENCY ALGORITHMS FOR A MULTICOMPUTER-REDUNDANT DATA BASE ENVIRONMENT

by

CALVIN A. BUZZELL

B.S., California State Polytechnic University, Pomona, 1964

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
1979

ABSTRACT

This report describes a simulation study comparing performance factors of five methodologies designed to update redundant data bases and maintain consistent data.

The following methodologies are studied in the computer simulation: (1) Bernstein, (2) CODASYL, (3) Ellis, (4) Hybrid, and (5) Johnson and Thomas.

The methodologies are compared in various system configurations.

Using mathematical models formulated from the study's simulation data, methodologies are graphically compared varying system and experimental parameters.

A new consistency algorithm for redundant data bases is proposed.