AD-A056 048 OHIO STATE UNIV COLUMBUS COMPUTER AND INFORMATION SC--ETC F/G 9/2
SIMULATION STUDIES OF THE DATABASE COMPUTER (DBC).(U)
FEB 78 D K HSIAO, K KANNAN N00014-75-C-0573
UNCLASSIFIED NL

| OF |
ADA
056048

END
DATE
FILMED
9-78
DDC

TECHNICAL REPORT SERIES

LEVEL # 0

COMPUTER & INFORMATION SCIENCE RESEARCH CENTER

08 25 049

THE OHIO STATE UNIVERSITY COLUMBUS, OHIO

LEVEL II

(OSU-CISRC-TR-78-1)

1

SIMULATION STUDIES OF THE DATABASE COMPUTER (DBC)

by

David K. Hsiao and Krishnamurthi Kannan

Work performed under

Contract N00014-75-C-0573

Office of Naval Research

Computer and Information Science Research Center

The Ohio State University

Columbus, Ohio 43210

February, 1978

78 08 25 049

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER OSU-CISRC-TR-78-1	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) "Simulation Studies of the Database Computer (DBC)"		5. TYPE OF REPORT & PERIOD COVERED Technical Report
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) David K. Hsiao and Krishnamurthi Kannan		8. CONTRACT OR GRANT NUMBER(s) N00014-75-C-0573
9. PERFORMING ORGANIZATION NAME AND ADDRESS		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 784115-A1
11. CONTROLLING OFFICE NAME AND ADDRESS		12. REPORT DATE February , 1978
		13. NUMBER OF PAGES 34
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimite

Scientific Officer	DDC New York Area
ONR BRO	ONR 437
ACO	ONR, Boston
NRL 2627	ONR, Chicago
ONR 1021P	ONR, Pasadena

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Database computer, structure and data loops, structure and mass memories; simulation, throughput, response time, queuing; FIFO, MFD and MAU-first disciplines; short, medium-sized, long queries.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report consists of a series of simulation studies of the hardware throughput of a specialized machine known as the database computer (DBC). Although raw hardware performance of the DBC can be estimated in an order of 80 to 160 times that of conventional software-oriented database management systems, there is the need of knowning the DBC's response time to user access requests, load condition to various request types, and potential bottlenecks which may be caused by uneven performance matching of the key DBC components. The simulation studies are intended to address the need.

The simulation programs are written in GPSS and run on an IBM 370/168. All of these results are plotted in graph form. Prediction of the results, interpretation of the graphs, and observation of the deviations are the main effort of the simulation studies. Very good conclusions are obtained concerning the DBC's response time, load condition, and throughput bottleneck.

TABLE OF CONTENTS

PREFACE

This work was supported by Contract N00014-75-C-0573 from the Office of Naval Research to Dr. David K. Hsiao, Associate Professor of Computer and Information Science, and conducted at the Computer and Information Science Research Center of The Ohio State University. The Computer and Information Science Research Center of The Ohio State University is an interdisciplinary research organization which consists of the staff, graduate students, and faculty of many University departments and laboratories. This report is based on research accomplished in cooperation with the Department of Computer and Information Science. The research contract was administered and monitored by The Ohio State University Research Foundation.

Appendix I, referred to in this report, consists of program listings of three simulation programs which are not included herein. The reader may obtain the program listings by writing to the first author of the report.

1. INTRODUCTION

In the previous technical reports [1,2,3], we presented the functional design of a database computer known as the DBC. The present report attempts to determine the hardware performance of the DBC and compare it to existing database systems. A first-order analysis of the DBC hardware as depicted in Figure 1 proceeds as follows: The mass memory (MM) logic is designed to process an entire cylinder in one revolution. Because a cylinder generally consists of between 20 and 40 tracks, and because conventional systems process one track at a time, we can expect a performance improvement factor of between 20 and 40 over conventional systems. Furthermore, since the structure loop can be processing a current request while the mass memory is processing a previous request, a performance improvement factor of 2 can be expected over systems which store structural information and raw data on the same storage medium. Furthermore, the high degree of pipelining of the DBC components and clear delineation of front-end general-purpose processing and back-end special-purpose database management may allow at least a performance improvement factor of 2. Thus, the proposed database computer is likely to have a raw hardware processing power which is 80 to 160 times that of existing software-oriented systems.

The above analysis falls short of predicting what the absolute performance figures are likely to be under various load conditions. And in the absence of reliable performance data on conventional systems, the relative figures do not give us any clue as to the likely performance of the DBC. In this report, we therefore have made an attempt to answer some fundamental questions such as: What is the probable response time of the DBC for a user access request? What is the throughput of the DBC? How do response time and throughput depend on the various parameters of the DBC components? Can the track information

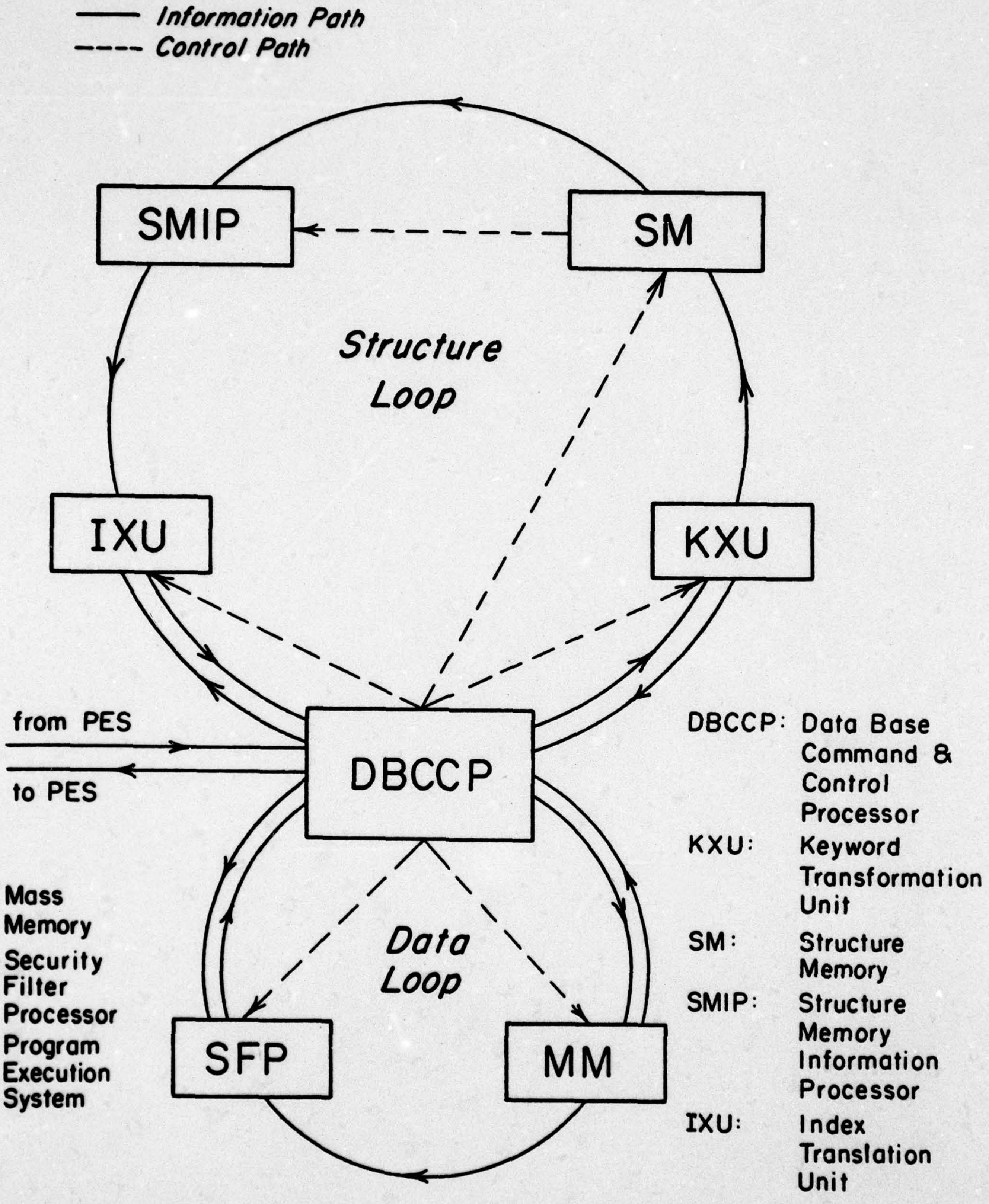
Information Path
Control Path
SMIP
SM
Structure Loop
IXU
KXU
from PES
to PES
DBCCP
Data Loop
SFP
MM
MM: Mass Memory
SFP: Security Filter Processor
PES: Program Execution System
DBCCP: Data Base Command & Control Processor
KXU: Keyword Transformation Unit
SM: Structure Memory
SMIP: Structure Memory Information Processor
IXU: Index Translation Unit

Figure 1. Architecture of DBC

processors (TIPs) in the mass memory (MM) retrieve all the relevant information in one disk revolution as anticipated? What is the effect of the look-aside buffer on the performance of the structure memory (SM)?

In order to answer these questions, we examined two lines of approach. First, one could use analytical tools based on queueing theory to analyze the flow of information within the DBC. Second, one could use simulation techniques to study the behavior of the DBC under various input conditions and for various values of the DBC design parameters. It would be ideal if both the approaches could be pursued to a point where meaningful results are obtained. We could then proceed to compare the results of the two approaches and obtain a high degree of confidence in our performance predictions, if the results agree.

In the case of the DBC, however, the complexity of the problem at hand and the limitations of queueing theory made it altogether difficult, if not impossible, to obtain meaningful theoretical results. Thus, the employment of simulation techniques became the only feasible approach. The simulation effort that was undertaken was restricted to a consideration of the questions raised earlier in our discussion. No attempt was made to validate the DBC design in terms of the verification of the various algorithms proposed in the previous reports. Such an attempt, which would require a full-scale logic simulation of the DBC, is not intended for this first study of the DBC performance.

The organization of the remainder of this report is as follows: In the next section we propose a simulation model of the DBC. In Section 3, we present the results of the simulation and their interpretation. In the last section, we make some general remarks on the performance of the DBC and the limitations of the current design.

2. A SIMULATION MODEL OF THE DBC

As illustrated in Figure 1, the DBC consists of two loops of processors and memories, namely, the structure loop and the data loop. The simulation model depicts a sequence of events that takes place between the time a user request enters the DBC and the time the response data for the request is sent out of the DBC. There are two possible sequences of events that can take place for a request. First, the request can be sent directly to the mass memory (MM) without being processed by the structure loop components. This is the case when the user issues the retrieve-by-pointer, delete-by-pointer or retrieve-within-bounds command. The second course of events involves processing by the structure loop components followed by processing by the data loop components. This is the case when the user issues a retrieve-by-query, delete-by-query, load-records or replace-record command.

2.1 The Simulation Model of the Structure-Loop Components

Let us now describe the events that take place for requests that involve processing by the structure-loop components. Such requests have either queries or records as arguments and are received first by the database command and control processor (DBCCP) as depicted in Figure 1. A query, when specified as an argument, is decomposed into its constituent query conjuncts. Each conjunct is treated as an independent job by the DBCCP for processing by the structure-loop components. The priority with which such jobs are scheduled for processing is specified by the user request. Within each priority class, jobs are processed on a first-in-first-out (FIFO) basis. In case a record is specified as the argument, record processing by the structure-loop components is also considered an independent job by the DBCCP.

When a job is scheduled for processing by the structure loop, the DBCCP sends the keyword predicates of a query conjunct (or the keywords of a record) to the keyword transformation unit (KXU). We shall refer to these predicates and keywords as tasks. The tasks are placed in an input queue by the KXU, and processed sequentially. When a task has been processed by the KXU, it is sent to the structure memory (SM). At the structure memory it is placed in another queue for processing by the processing elements. The output from the structure memory for each task is sent to the structure memory information processor (SMIP). The SMIP acts as a sink for the outputs of all the tasks of a given job. The output from the SMIP is usually much smaller (in terms of the number of bytes) than the size of the input stream. The output of SMIP is sent to the index transformation unit (IXU) for further processing. We shall consider the SMIP output for a job as a single entity as far as the IXU processing is concerned.

In Figure 2, we have summarized the above discussion. We have also indicated the major parameters of the structure-loop model. Access commands that are received by the DBC hav e an average inter-arrival time of QTime. In the absence of any data on the arrival pattern of DBC requests, we assume that the inter-arrival time has an exponential distribution. Access commands are divided into two categories - those that require structure-loop processing (i.e., query-based and record-based commands) and those that do not require structure-loop processing (i.e., pointer-based commands). The distribution of access commands is assumed to be the following:

- Query-based Commands:	50%
- Record-based Commands:	20%
- Other (that do not require structure-loop processing):	30%

The rationale for the above distribution is as follows: First, the DBC is designed to accept commands which refer to data by their contents. Therefore, it is reasonable to assume a high proportion of query-based commands. Second, the simulation model assumes that DBC loading operations would be relatively infrequent under normal conditions. This means that record-based commands such as load-records and insert-records will be significantly smaller in number than the query-based commands.

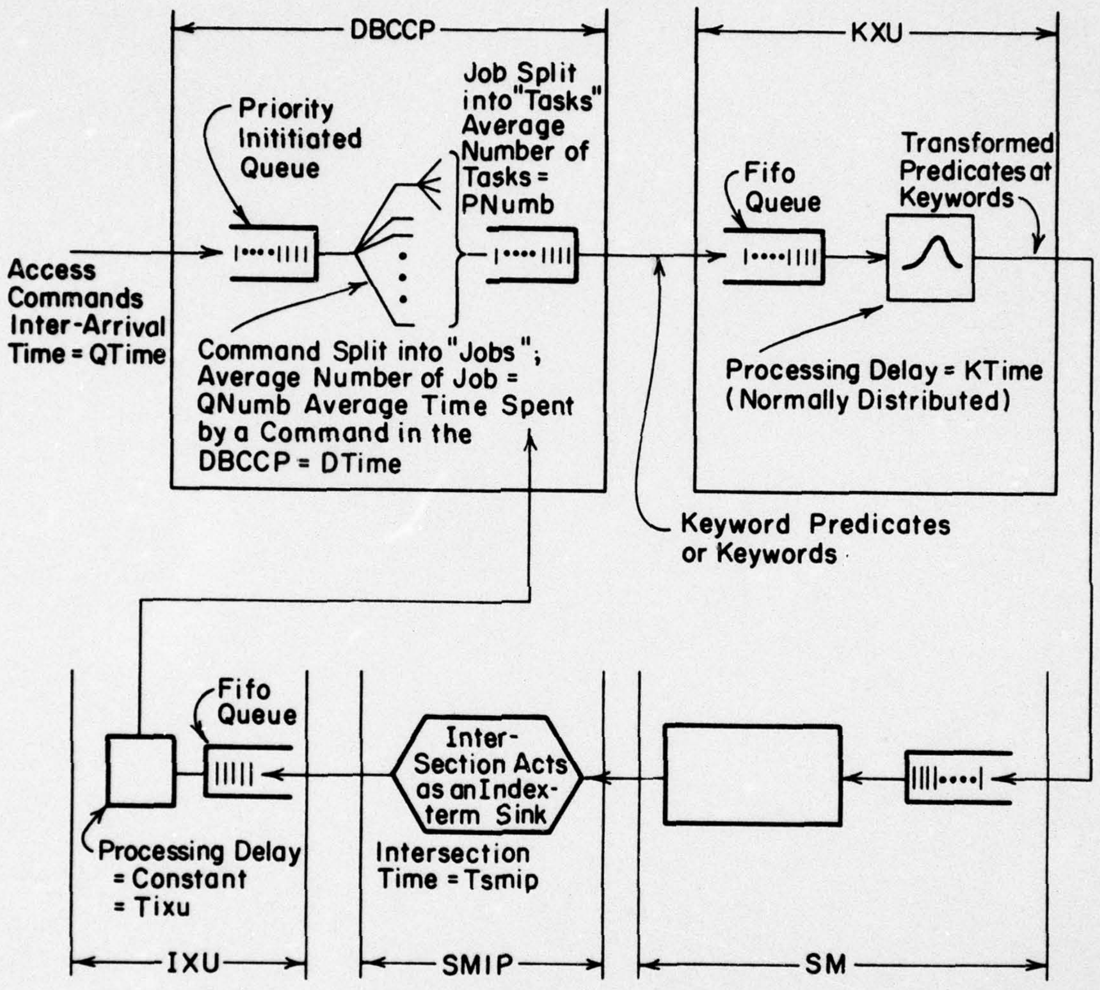
DBCCP
KXU
Priority Inititiated Queue
Job Split into "Tasks" Average Number of Tasks = PNumb
Access Commands Inter-Arrival Time = QTime
Command Split into "Jobs"; Average Number of Job = QNumb Average Time Spent by a Command in the DBCCP = DTime
Fifo Queue
Transformed Predicates at Keywords
Processing Delay = KTime (Normally Distributed)
Keyword Predicates or Keywords
Fifo Queue
Inter-Section Acts as an Index-term Sink
Processing Delay = Constant = Tixu
Intersection Time = Tsmip
IXU
SMIP
SM

Figure 2: Simulation Model of the Structure Loop

A command spends an average of DTime in the DBCCP before it is split into jobs and sent into the pipeline consisting of the KXU, SM, SMIP and IXU. Each command generates on the average, ANumb jobs. Each job has an average of PNumb tasks. Both the quantities are assumed to be normally distributed.

The keyword transformation unit (KXU) is primarily a processing engine. We model the KXU by a FIFO queue and a delay unit labeled KTime in Figure 2. The determination of the structure memory processing time for a keyword predicate is more complex. The processing time depends on the number of buckets that have to be searched in order to find the directory entries of keywords satisfying the predicate. Since the physical blocks of a bucket are generally uniformly distributed across the memory units of the structure memory, the number of accesses to the SM for a bucket is assumed to be 1. Thus, the number of accesses for a predicate is equal to the number of buckets to be searched. We assume this number to be normally distributed with an average of BNumb and a standard deviation of 2 x BNumb x KTime.

In computing the total processing time in the SM, one must take into account not only the search time, but also the time to transmit the index terms out of the processing elements. We recall [2] that index terms are transferred from the processing elements to the SMIP: the transfer rate BSped (bytes per second) of the SMPEBUS, the average number of index terms INumb per keyword directory entry and the average number of keywords per bucket KNumb. The transfer time of index terms of keywords satisfying a keyword predicate is then given by the following expression:

$$\text{Transfer Time} = \text{BNumb} \times \text{INumb} \times \text{KNumb}/\text{BSped}$$

The total SM processing time for a keyword predicate, then, is the sum of the search time and the transfer time. The index terms for the predicate are assumed to be collected by the SMIP at a rate commensurate with the rate at which they can be retrieved from the SM. The time taken by the SMIP to perform intersection is parameterized by the variable Tsmip. Finally, the time taken by the IXU to decode the index terms relevant to a query conjunct (i.e., a job) is given by the constant Tixu.

2.2 The Simulation Model of the Data-Loop Components

Either after a query conjunct (or a record) is processed by the structure-loop components or when a user command does not need to be processed by the structure loop components, the conjunct (or the record) in the user command is sent to the mass memory (MM). The mass memory processes the command and transmits the output (i.e., data elements) to the security filter processor (SFP) which after a delay sends them to the DBCCP for onward transmission to the user.

In Figure 3, we have shown the model used for simulating the MM and the SFP. User requests which have been pre-processed by the structure loop or which do not require such pre-processing are placed in an output queue by the DBCCP. In this queue, the requests are known as MM orders. The MM maintains a fixed number of hardware queues. Orders pending outside the MM are placed in one of the queues according to the queueing discipline chosen for the MM implementation. We shall discuss three strategies for the placement and movement of orders in these queues.

2.2.1 Three Queueing Disciplines

The first discipline is the well-known FIFO discipline. In this case, each of the hardware queues (called the seek queues) contains orders on a particular disk drive. An order pending outside the MM is moved into one of these seek queues on a first-come-first-served basis if there exists a queue which is empty or which contains orders for the same disk drive as the one referenced by the pending order. If there is no such queue, the order waits outside the MM until one is av ailable. Orders within a seek queue are moved on a first-

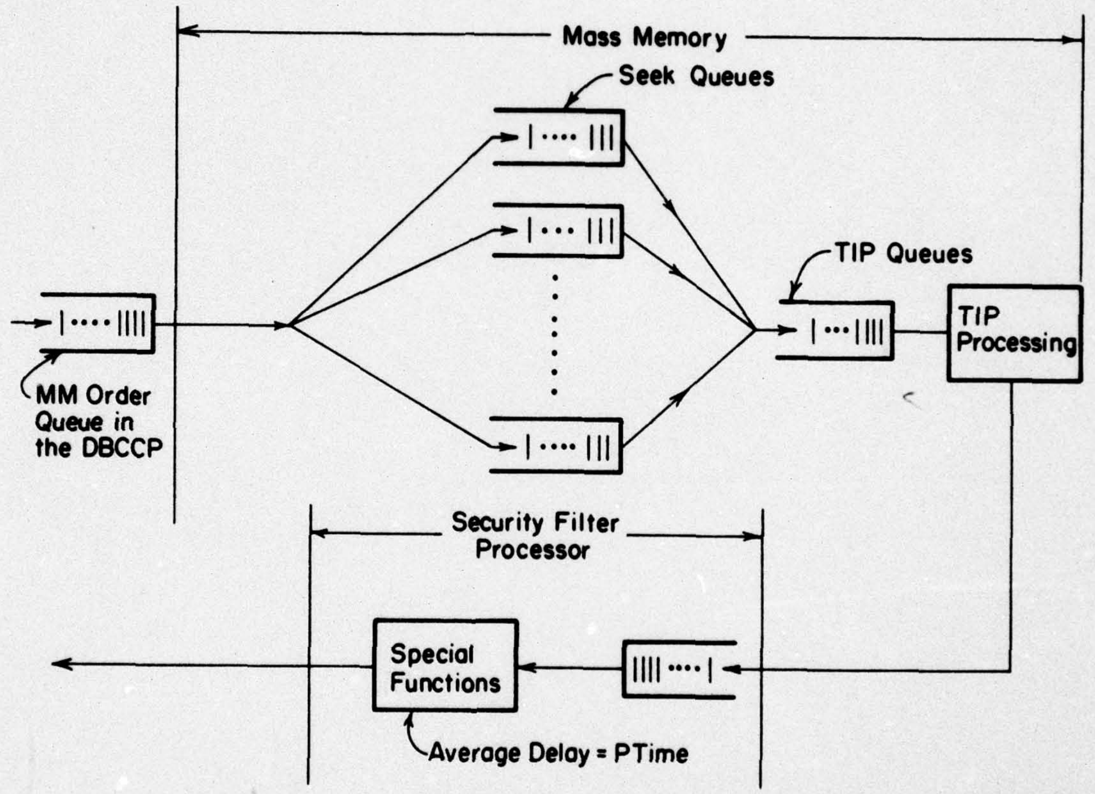

Figure 3: Simulation Model of the Data Loop

in-first-out basis. Each order requires a cylinder seek. When the cylinder seek has been completed, the order is moved into another queue for the track information processors (TIPs). This queue is called the TIP queue. When an order in the TIP queue has been serviced by the TIPs, then the next order waiting in the TIP queue is considered for service by the TIPs. A timing diagram depicting the wait times and service times for an MM order is shown in Figure 4.

The second discipline, which we define as the MAU-first discipline, differs from the FIFO discipline in the following way: Orders in the seek queues are not moved on a first-in-first-out basis; when an order at the head of a seek queue is moved to the TIP queue as a result of a seek completion, other orders on the same cylinder which exist in the seek queue are also moved to the TIP queue. This is done by linking all orders on a given cylinder within a seek queue. The projected advantage of this discipline over the FIFO discipline is that the MM will save a few cylinder seeks by "batching" all the orders on a cylinder.

The third discipline, which we call the MAU-first-file-next-drive-last (MFD) discipline, moves orders within the seek queues in the following manner: As in the case of the MAU-first discipline, orders on the same cylinder are moved as a group to the TIP queue. In addition, when a new seek is to be initiated on the disk drive, preference is then given to an order which references the same file as the last block of orders moved out of the queue. The projected advantage here is as follows: A file is normally allocated contiguous cylinders on a disk drive. Therefore, by giving preference to orders on the same file over orders on different files, it is conceivable that the disk arm movement can be sharply reduced.

2.2.2 The Distribution of Order Type and Parameters

Let us now describe the parameters involved in the simulation of the data

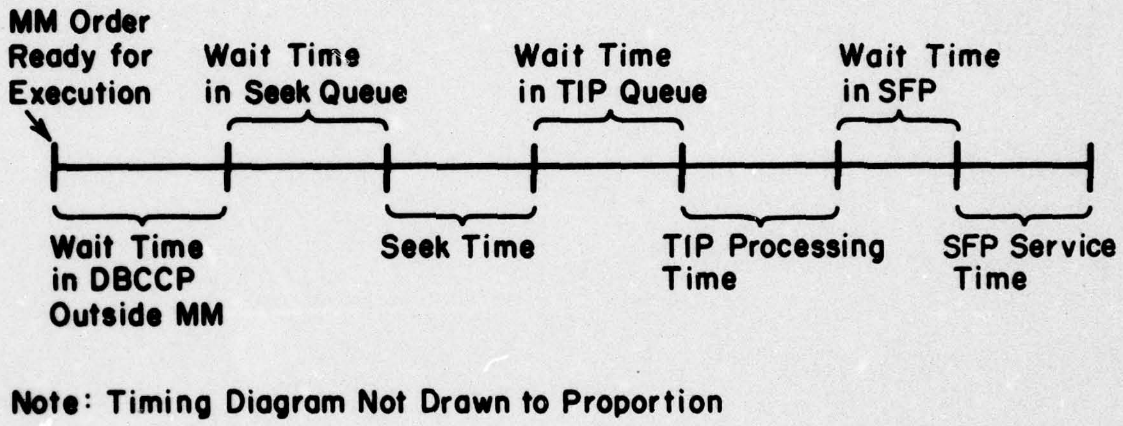

Figure 4: Timing Diagram Depicting the Sequence of Events Taking Place in the Data Loop

loop components. The distribution of order type is assumed to be as follows:

35%:	Retrieve-by-Query
5%:	Retrieve-by-Pointer
5%:	Retrieve-by-Query-with-Pointer
5%:	Retrieve-within-Bounds
20%:	Insert-Records
10%:	Delete-by-Query
5%:	Delete-by-Pointer
15%:	Replace-Keywords

Note that the simulation model can accommodate any distribution or order types. We have chosen the above distribution as one that is likely to be encountered in an environment where updates (i.e., deletions, insertions and replacements) and retrievals have the same probability of occurrence. The average seek time in which the access arm is moved from one cylinder to another within the same file is assumed to be 15 milliseconds. When the access arm has to be moved from one file to another, the average seek time is assumed to be 35 milliseconds. The rotation time for the disk drives is assumed to be 20 milliseconds.

The variable parameters are as follows: The average inter-arrival time of MM orders is MTime. The number of hardware seek queues maintained by the MM controller is given by QNo. The number of active files (i.e., the number of files referenced by the MM orders) is likely to be smaller than the total number of files resident in the mass memory and is given by FiLst. The time taken by the security filter processor to massage the data elements retrieved by the mass memory is modelled by the parameter PTime (average processing time).

2.2.3 The Simulation of the Track Information Processors

In performing the simulation of the MM, it was assumed that a retrieval, deletion or insertion order can be processed by the TIPs in one revolution and that a replace-record order can be processed in two revolutions. However, this assumption is based on the ability of the TIPs to store all the relevant data elements in their local buffers and on the ability of the IOBUS (see [3]) to transfer all the data elements out of the TIP buffers within one revolution. In order to examine how good these assumptions are, a second-level simulation was carried out. This simulation process will now be described.

The simulation model of a TIP is shown in Figure 5. This model depicts only the retrieval process, since the retrieval process is most likely to cause the local TIP buffer to overflow resulting in extra revolutions for its completion. The local buffer consists of two modules. Each module can be accessed by either of two processors, namely the drive interface processor (DIP) and the controller interface processor (CIP). Each module consists of a set of sequentially accessed segments.

The logic that is simulated by the model may be summarized as follows: Records on a track are read by the drive interface processor in a sequential manner. Before a record is read, the DIP attempts to gain access to one of the two buffer modules. If both are full, or if one is being emptied by the CIP and the other is full, the DIP discontinues processing for the rest of the current revolution and resumes processing when the record reappears under the read head. When both modules can be access, the DIP gains access to the module which has the larger number of empty segments.

The parameters of the TIP simulation can now be discussed. The load Ltip on the TIP is characterized by the percentage of records on each track that satisfy the retrieval criterion. Based on this percentage, the simulation model can statistically determine if a record on a track satisfies the selection criterion (i.e., the query conjunct) or not. The segment size is parameterized by the variable SSize. The length of the record LRec is another important parameter of the model.

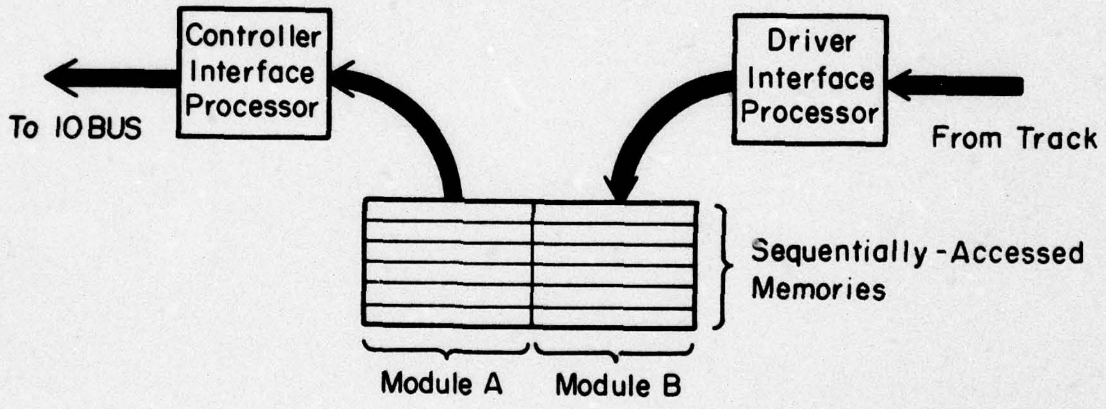
Controller Interface Processor
To IO BUS
Driver Interface Processor
From Track
Sequentially-Accessed Memories
Module A
Module B

Figure 5: Simulation Model of the Track Information Processor

3. SIMULATION RESULTS AND THEIR INTERPRETATION

The simulation model described in the last section was programmed on an IBM 370/168 using the GPSS/360 simulation language [4,5]. The GPSS programs are given in Appendix I. Because the model (and, therefore, the GPSS program) is highly parameterized, the number of distinct cases that can be formulated is extremely large. The combinatorics inherent in such modeling makes any "exhaustive" simulation infeasible. Fortunately, it turns out that a great deal can be learnt by simulating just a small, 'judiciously' chosen, subset of all the possible cases involved. In the following section, we discuss the experiments and results of the structure loop simulation. Then in Section 3.2 we discuss the experiments and results of the structure loop simulation.

3.1 The Behavior of the Structure-Loop Components

For studying the behavior of the structure-loop components, we divided the access commands requiring processing by the structure-loop components into three categories: short requests, medium-sized requests and long requests. The short requests are defined as those requests which involve queries with 3 conjuncts (on the average), with each conjunct having 4 predicates (on the average) or those requests that involve records with 3 clustering conditions and 4 directory keywords. The medium-sized requests are defined as those that involve queries with 6 conjuncts (on the average), with each conjunct having 8 predicates (on the average) or those requests that involve 6 clustering conditions and 8 directory keywords. Finally, the long requests are defined to be requests with queries having 10 conjuncts (on the average) and 14 predicates per conjunct, or requests with 10 clustering conditions and 14 directory keywords (on an average).

It is easy to observe that as the requests get longer (in terms of number of predicates and keywords that need to be processed), the load on the structure-loop components also progressively increases. Therefore, it became necessary to assume different request arrival rates for the three categories. In particular, an inter-arrival time of 50 milliseconds was assumed for short requests; an inter-arrival time of 100 milliseconds was assumed for medium-sized requests and an inter-arrival time of 200 milliseconds was assumed for large requests. These figures were determined by trial runs in which several different inter-arrival times were tested. For each category of requests, the smallest inter-arrival time which did not result in DBC instability was chosen for conducting other experiments. By DBC instability we mean the indefinite growth of various queues within the simulation program, resulting in its abnormal termination. In subsequent discussions, we shall refer to such an occurrence as "choking".

Since the structure memory (SM) and the keyword transformation unit (KXU) were perceived to be the most critical units of the structure loop, the main objective of the simulation study was to observe response times for queries, conjuncts and records as a function of the parameters associated with the structure memory and the keyword transformation unit. The time required to access a block of memory in the SM and the KXU processing time were both varied from 1 millisecond to 3 milliseconds in steps of 0.25 milliseconds. For each set of values, the simulation program was run with and without enabling the look-aside buffer logic in the SM. This was done in an attempt to determine the effect of the look-aside buffer on the response times. Each simulation program run was allowed to proceed for a period of (simulated) time long enough for the results to stabilize (i.e., long enough for the effects of the initial conditions to become vanishingly small). This was done by allowing the simulation program to run for increasing periods of time and by comparing results until they (i.e., the results) showed very little change.

In Figure 6, we have plotted the results for short requests. The utilization of the structure memory as a function of the access time is plotted in Figure 6c. The utilization was found to be relatively insensitive to the KXU processing delay, and depended only on the access time of the structure memory and request length. Since the SM is the most expensive component of the structure loop, it is imperative that we strive to maintain a high level of utilization of SM. We observe from Figure 6a, that the utilization tends to become a constant beyond an access time of 2 milliseconds, indicating that there is very little idle capacity in the SM when its access time is greater than 2 milliseconds. Looking at Figure 6a and 6b, we find that at about the same access time (i.e., 2 milliseconds), the response times begin to climb rather steeply. Clearly, we have to operate the SM with access times below 2 milliseconds if we wish to keep the structure loop from "choking."

In Figure 7, we have plotted the results for medium-sized requests. As mentioned before, for medium-sized requests, it became impossible to maintain the inter-arrival time at 50 milliseconds. This can be explained by observing that the processing load presented by a medium-sized request is approximately four times that of a short request. The simulation program was run with an average request inter-arrival time of 100 milliseconds. Comparing the utilization curves for small and medium-sized requests, we find that the utilization is consistently higher for medium-sized requests. This is as it should be, since we have increased the processing load four times, but reduced the arrival rate only by half. Also, looking at Figures 6a and 7b, we find that the response times for medium-sized requests are higher than those for short requests. However, the interesting aspect of the comparison is that the response times for medium-sized requests are only 10% to 35% higher than those for short requests (when the SM access time is below 2 milliseconds), although we would expect a 100% increase.

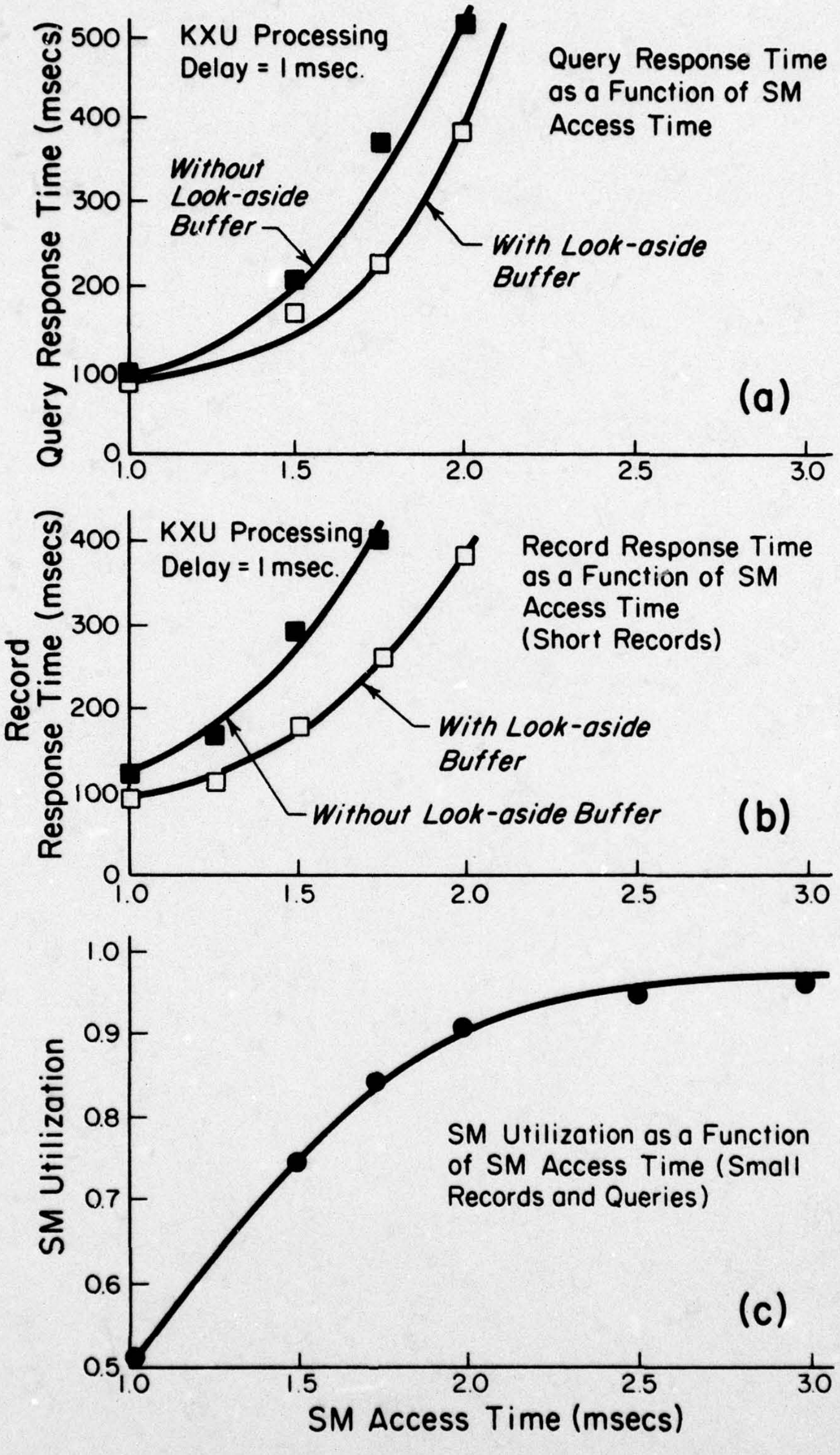

Figure 6: Simulation Results for Short Requests

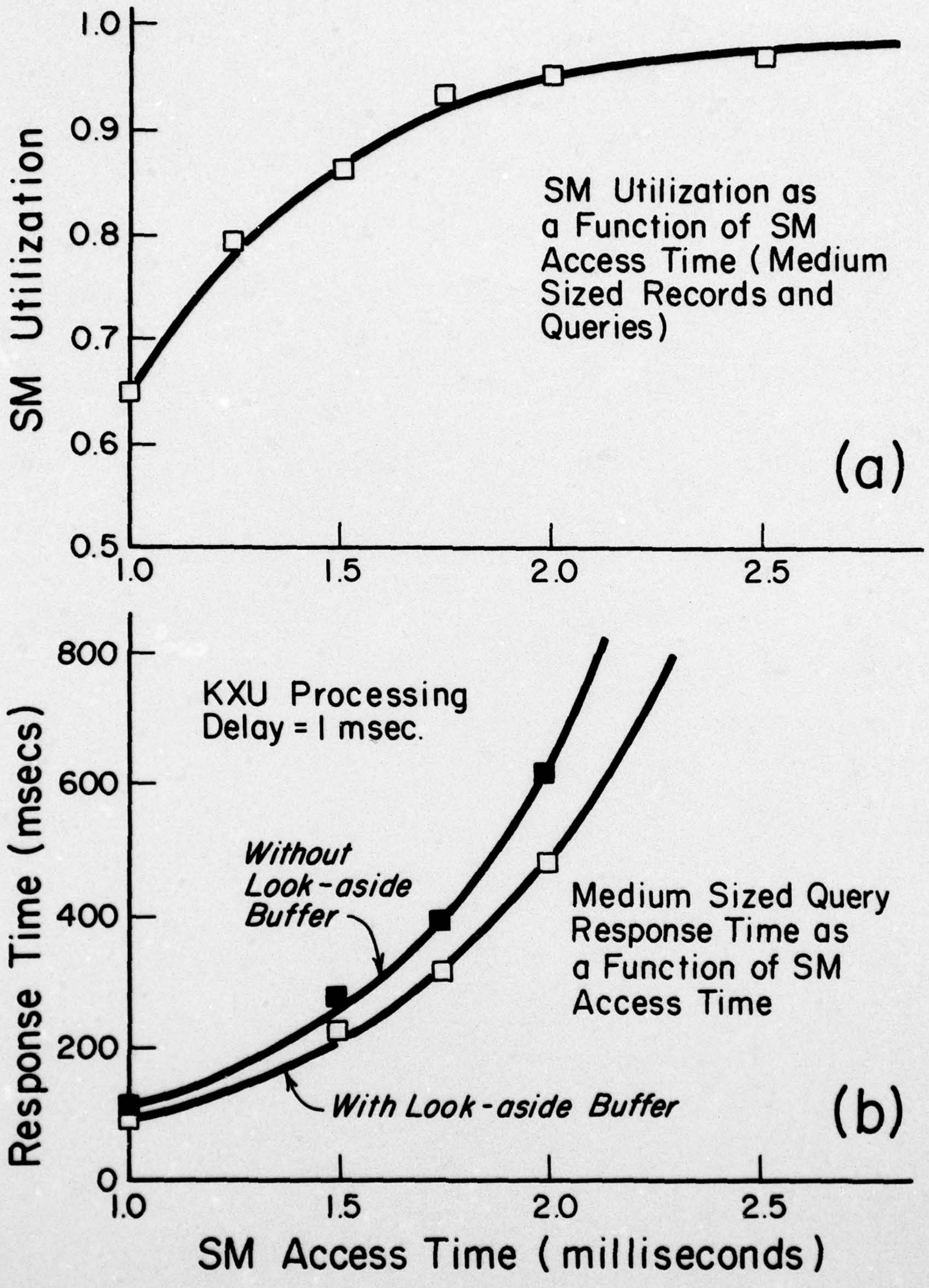

Figure 7: Simulation Results for Medium-Sized Requests

This phenomenon can be explained by observing that the structure memory was operating well below its maximum capacity in the case of short requests (as can be seen from the utilization figures). The idle capacity is absorbed in the case of medium-sized requests with only a small increase in response time.

In Figure 8, we have plotted the utilization and response times as a function of the SM access time for large requests. For reasons mentioned before, the simulation program was run with an average request inter-arrival time of 200 milliseconds. In spite of the slower arrival rate, the utilization is seen to be significantly higher for large requests than for the small and medium-sized requests. Also, the response times are significantly higher than in the previous two cases. For example, the response time with an access time of 0.5 milliseconds is 192.86 milliseconds. This exceeds the response times for short and medium-sized requests with an access time of 1 millisecond. The conclusion to be drawn from these observations is that the structure loop cannot be operated without possible "choking" if the requests are long and the arrival rate exceeds 5 requests per second or if the requests are long and the SM is operated with an access time greater than 1 millisecond.

An important observation that can be made in regard to Figures 6, 7 and 8 is that in each case, the response time for a structure memory with look-aside buffer is much better than for a structure memory without look-aside buffer. More specifically, an improvement of 10-20% in response times may be expected by using the look-aside buffer. (We assume, in the simulation model, that the look-aside buffer is large enough to hold all update requests that are made to the structure memory during the time a retrieve request is being serviced.)

We would now like to summarize our discussion thus far. For short and medium-sized requests, an SM access time of 1.5 milliseconds and a KXU

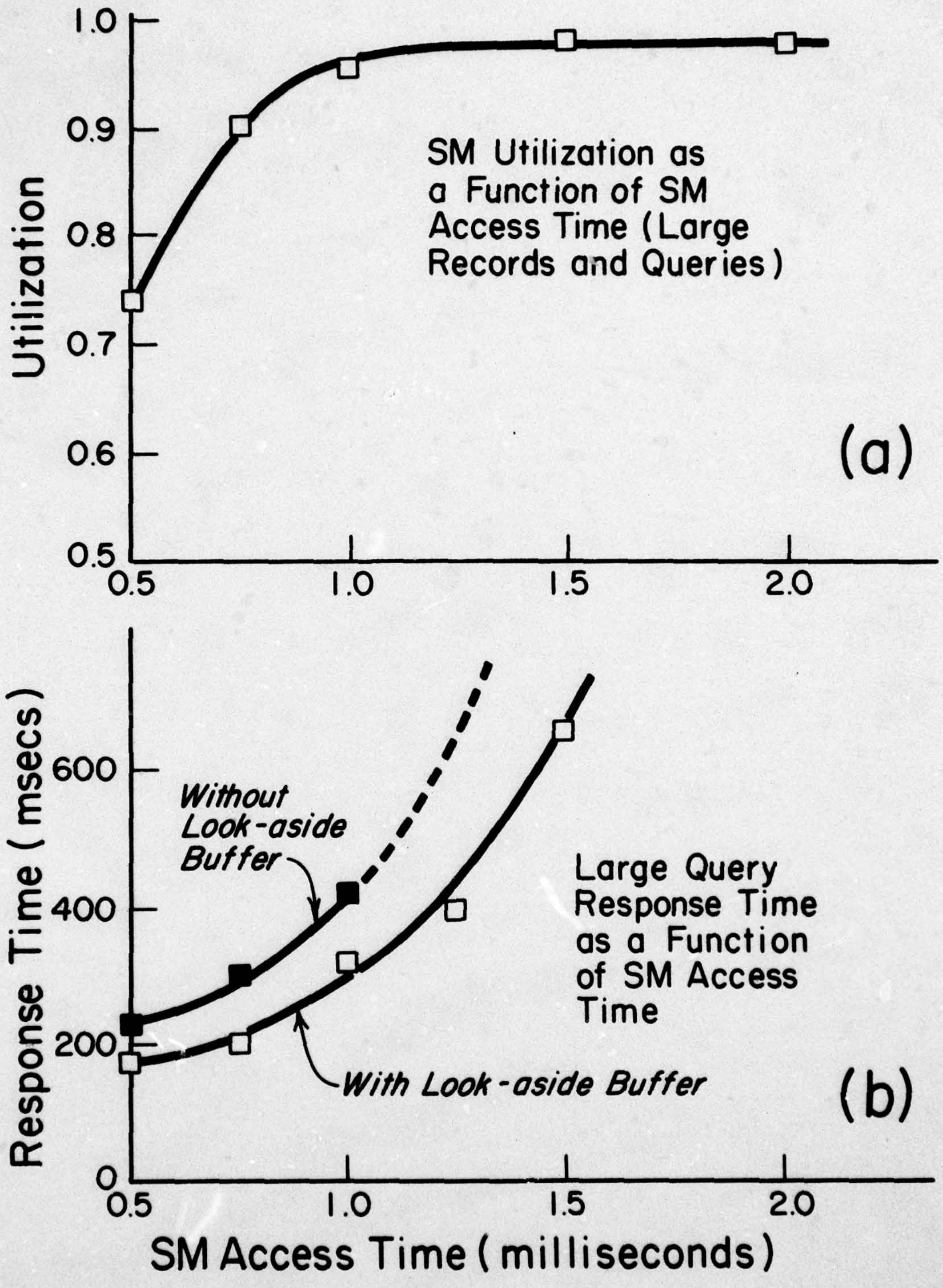

Figure 8: Simulation Results for Large Requests

processing delay of 2 milliseconds per task will yield a response time of under 200 milliseconds, an SM utilization factor of between 0.7 and .9, and an average throughput of between 20 and 10 access requests per second. For long requests, the observations indicate that an SM access speed of under 1 millisecond is necessary for a response time of about 200 milliseconds and a throughput of 5 requests per seocnd. In all cases, the use of a look-aside is recommended.

3.2 The Behavior of the Data-Loop Components

We begin our study of the performance of the data-loop components by observing that in order to ensure that the data loop does not become a bottleneck within the DBC, its throughput must match the rate at which orders are created for it by the structure loop and the DBCCP. We can determine the rate at which orders are likely to be sent to the data loop by consulting the results provided at the end of the last section.

A reasonable starting point is to assume that over a long period of time, the average request received by the DBC will closely resemble a medium-sized request as defined in the last section. Since the structure loop was seen to be able to handle about 10 such requests per second, the data loop should be in a position to handle orders generated from these requests in the same time period. The number of orders generated from each request can be determined as follows: Recall from Section 2.1 that 50% of all access requests are query-based requests; furthermore, a medium-sized request has on the average 6 conjuncts (if the request is query-based). Since each conjunct generates at least one mass memory order, the total number of orders is given by: Probable number of orders generated by 10 requests = $10 \times 0.5 \times 6 + 10 \times 0.5 = 35$. Therefore, the data loop should have a throughput of about 35 orders per second.

This translates into an inter-order arrival time of about 28.5 milliseconds.

A simple summation of the average times involved in the processing of a MM order by the mass memory and the security filter processor (SFP) shows that the data loop is incapable of meeting such a requirement. For example, a retrieve-by-query order needs a minimum of 15 milliseconds for a seek operation. This is followed by a 20 millisecond processing time by the track information processors. The SFP takes an additional time of say, 10 milliseconds. The total adds up to 45 milliseconds. We have not included any of the waiting times that may be involved in the data loop (see Figure 4). Thus, a more realistic throughput rate would allow the data loop to process orders at an average of 50 milliseconds per order. In the remaining part of this section, we shall assume an average inter-arrival time of 50 milliseconds per order.

The first experiment we carried out with the above inter-arrival rate was to compare the three queueing disciplines proposed in Section 2.2 for the movement of orders within the seek queues. In Section 2.2 we had anticipated that the MFD discipline would result in minimizing the access times associated with each request. In Figure 9, we have plotted the data loop response time as a function of the number of seek queues in the mass memory for the three queueing disciplines. From these figures, we find that for all the three disciplines the response time tends to reach the same value as the number of seek queues increases. The rate at which the response time declines is different for each of the disciplines. The response time curve fur the MFD discipline declines rather sharply when the number of seek queues is in the range 2-4, and thereafter stabilizes at around 100 milliseconds. The response time curve for the MAU-first discipline and FIFO discipline decline more gently and stabilize at around the same 100 millisecond value when the number of seek queues is increased to 8. This behavior of the three queueing disciplines would seem to agree with our reasoning in an earlier section. The MFD discipline

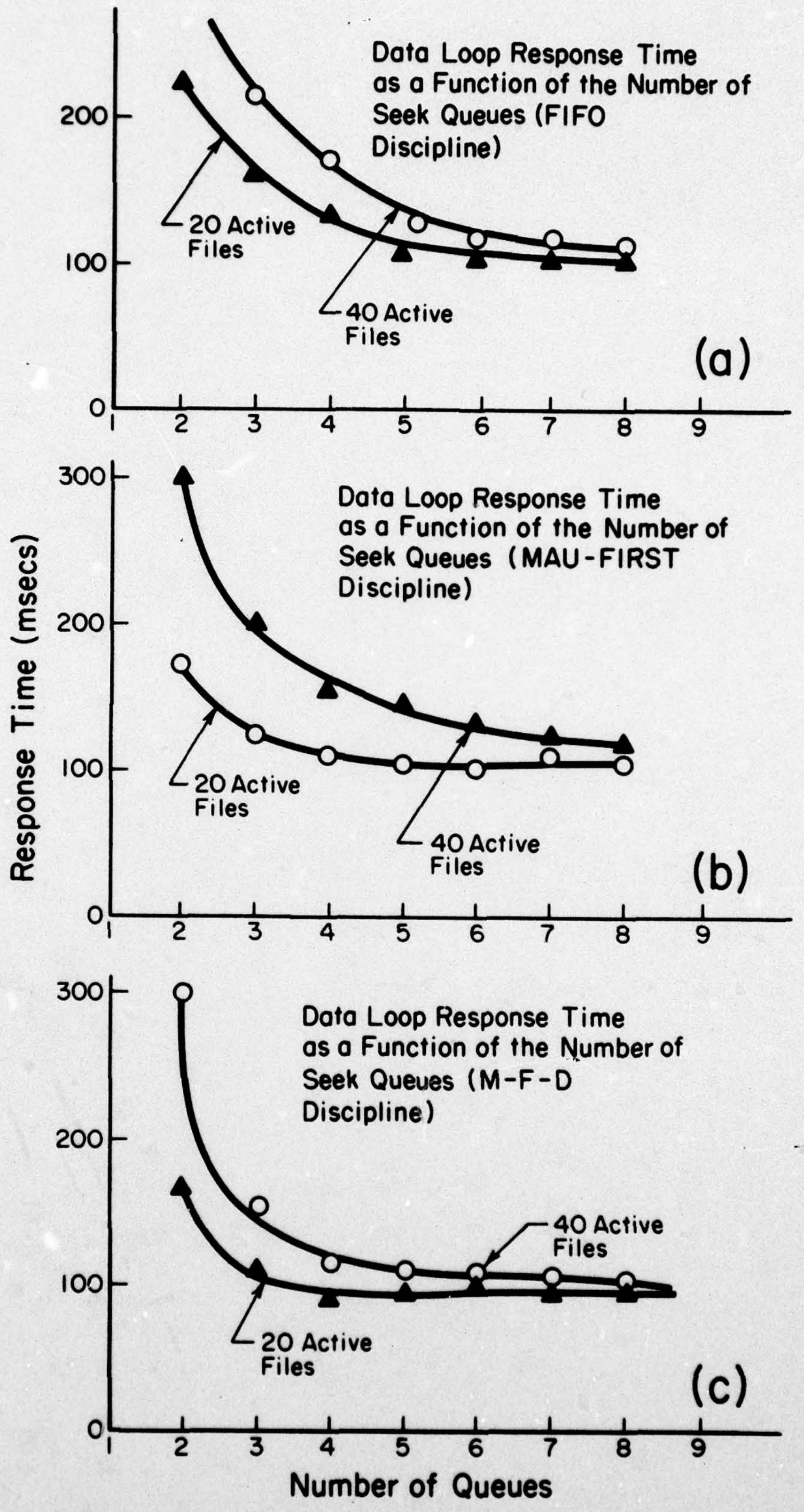

Figure 9: Results of Data Loop Simulation

reaches stabilization at a smaller number of seek queues because the average seek time associated with an order under this discipline is small. As a result, complete overlap of TIP processing and seek operations can be reached with a small number of queues. Increasing the number of queues beyond this point cannot bring about any further reduction in response time.

The number of seek queues to be maintained is dictated not only by a desire to minimize the average response time but also by a desire to minimize the average waiting time outside the mass memory. In Figure 10, we have plotted the average waiting time of a mass memory order as a function of the number of seek queues. Comparing Figures 9a and 10, we find that although the response time curve stabilizes around 100 milliseconds when the number of queues is 4, the average waiting time outside the MM attains a value of zero only when the number of queues is increased to 8. Since it is important that orders do not wait in the DBCCP for entry into the mass memory, we favor the higher number of seek queues.

We now turn to the results of the simulation of the track information processors (TIPs). One of the primary goals of the simulations study was to test the hypothesis that a TIP can retrieve all relevant information in one disk revolution. In Figure 11, we have plotted the average number of disk revolutions required to retrieve data elements as a function of the retrieval percentage for various segment sizes and record sizes. These plots show that for any record size, the smaller the segment size, the better the performance of the TIP. This may be explained by taking a closer look at the TIP logic.

Records that satisfy a query conjunct are stored in segments of the TIP buffer. Recall from [3] that these segments are sequentially accessed memories. During the time that a TIP is comparing a record's keywords with the predicates of a query conjunct the part of the record which has moved past the read head is stored in a segment in anticipation of a successful comparison. Now, if the

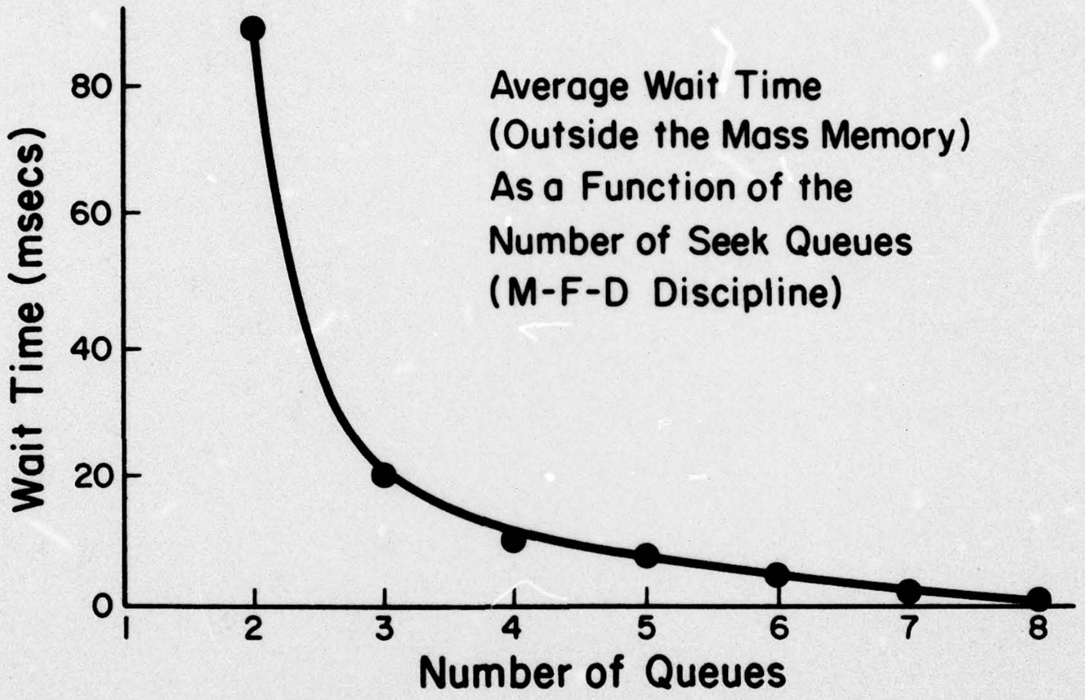

Figure 10: Mass Memory Order Wait Time as a Function of the Number of Queues within the Mass Memory

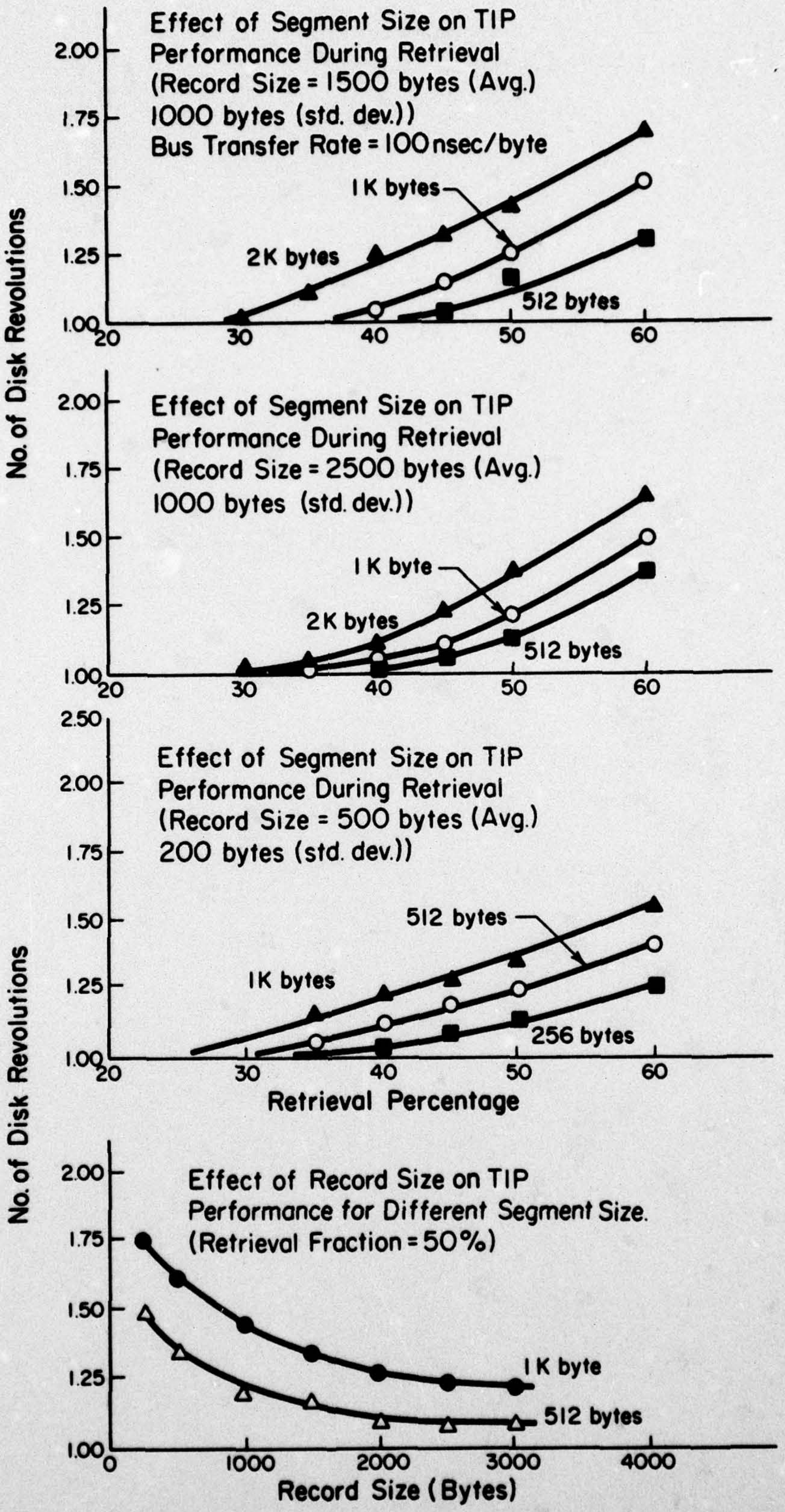

Figure 11: Results of Simulating the Track Information Processors during Retrieval

comparison fails in the middle of a record, then the part of the record already stored in the segment must be discarded. This is done by circulating the segment (forward of backward) to restore it to its position at the time the record first appeared at the read head. The segment is essentially unavailable for input or output during the recovery process. The larger the segment memory, the longer it will take to complete the recovery procedure. The non-availability of the segment can force the TIP to postpone the processing of a record by one revolution. Thus, there is reason to believe that a smaller segment would perform better than a relatively larger segment. This observation is borne out by the simulation result presented in Figure 11.

A further observation can be made is that the number of revolutions required by the TIPs to process retrieval orders being close to one, if the retrieval fraction is under 50%. Beyond 50% the performance begins to degrade significantly. In the light of the fact that a track usually has a capacity of 20K bytes, it would seem unlikely that the retrieval fraction would exceed 50% for most retrieval orders. Thus our initial hypothesis is seen to hold for most retrieval orders.

We now summarize our discussions of the data loop simulations studies. It is probable that the data loop, with the current design proposal may not be able to match the throughput of the structure loop. Two suggestions can be made to remedy the situation. First, the mass memory can be speeded up by extending the cylinder content-addressability concept to more than one cylinder at a time. This involves employing multiple sets of TIPs. The second suggestion involves slowing down the structure memory by increasing the access time of the structure memory. The first suggestion involves additional hardware, while the second affects the throughput of the DBC itself.

4. SUMMARY

We have presented in this report some of the useful results obtained from extensive simulation studies carried out on the DBC. The structure memory is capable of handling medium-sized requests at the rate of 10 requests per second. The response time of the structure memory to a query or a record can be held to below 200 milliseconds by carefully choosing the access time of the structure memory. In order for the structure loop to handle truly large requests, it may perhaps become necessary to employ random access memories instead of sequentially accessed memories.

The data loop has a throughput of about 20 mass memory orders per second and an average response time of 100 milliseconds for an order. However, as was shown in Section 3, this performance may be insufficient to avoid a bottleneck in the data loop. The performance of the mass memory may be substantially improved by incorporating a second set of track information processors. The design of the mass memory presented in [3] does not preclude the inclusion of such additional hardware; indeed it can easily be modified to handle two sets of track information processors instead of the one set currently envisioned.

REFERENCES

[1] Baum, R.I., Hsiao, D.K., and Kannan, K., "The Architecture of a Database Computer - Part I: Concepts and Capabilities", The Dept. of Computer and Information Science, The Ohio State University, OSU-CISRC-TR-76-1, (September 1976).

[2] Hsiao, D.K. and Kannan, K., "The Architecture of a Database Computer - Part II: The Design of the Structure Memory and its Related Processors", The Dept. of Computer and Information Science, The Ohio State University, OSU-CISRC-TR-76-2, (October 1976).

[3] Hsiao, D.K. and Kannan, K., "The Architecture of a Database Computer - Part III: The Design of the Mass Memory and its Related Components", The Dept. of Computer and Information Science, The Ohio State University, OSU-CISRC-TR-76-3, (December 1976).

[4] <u>IBM 360 OS GPSS User's Manual</u>, Form GH20-0326.

[5] <u>IBM 360 OS GPSS Application Description Manual</u>, Form GH20-0327.