PERFORMANCE CHARACTERISTICS OF DATA BASE MACHINES

by

RANDALL LEE MEHARG

B.S., Kansas State University, 1976

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
1980

Approved by:

Major Professor

TABLE OF CONTENTS

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

THIS IS AS RECEIVED FROM CUSTOMER.

LIST OF ILLUSTRATIONS

ACKNOWLEDGEMENTS

Five persons were highly instrumental in the preparation of this paper: Dr. P.S. Fisher, whose years of tolerance and help allowed me to complete this study; Dr. D.A. Gustafson, from whom my initial learnings of DBMS came; Dr. F.J. Maryanski, whose able guidance and work made this project possible; Dr. E.A. Unger, whose patience and willpower I tested severely; and my wife, Marjorie, whose support was invaluable. To each of you I extend my sincerest thanks.

CHAPTER 1

INTRODUCTION

Today, managerial trends indicate an ever increasing dependency upon nearly instantaneous accessibility of massive volumes of data. In answer to this requirement of the managerial world has come significant advancements in technology of large-scale high-speed central processing units (CPUs). Additionally, storage device technologies have achieved capacities nearly beyond comprehension to store these volumes of data.

Paralleling the advances achieved in both CPU and storage technology has come more sophisticated software data access methodologies (later termed file access methods). However, one truism is readily apparent: software is at best only as efficient as the hardware upon which it executes. Thus, if some method could be devised whereby hardware CPU execution speeds, massive storage capacities and software file access flexibility could be effectively merged, many of management's current and future requirements could be easily met. One feasible method of achieving this "merger" is a data base machine (DBMS). This DBM would provide hardware level primitives which would approximate very closely the current software retrieval functions. Additionally, the DBM would incorporate a specialized arithmetic logic unit (ALU) much like today's general purpose microprocessors. The ALU would provide greatly enhanced usability of the hardware primitives, thereby providing flexibility previously available only via software. The DBM would utilize CCD type memories to provide for a storage medium.

Initial work [3,12,24] has been undertaken to determine the efficiency and overall performance characteristics inherent in and vital

to this DBM. Basic prototypes of the DBM hardware have been developed [2,5,16]. However, only skeletal, heavily theoretical studies have been undertaken to determine performance characteristics of the DBM. This paper will undertake the presentation of statistics based upon application data bases used by a wholesale company. From the statistics provided, conclusions will be drawn concerning DBM performance.

Chapter 2 provides a technical description of a DBM based on relational associative processing (RAP) extended from Schusters model [28]. Chapter 3 traces the development of a mathematical model which compares the DBM execution times and Cullinane's Integrated Data Management System (IDMS) executing on a general purpose von-Neumann type computer execution times. Chapter 4 presents the results of the comparison. Finally, Chapter 5 provides an accounting of the uses of this study and suggests possible extensions.

Appendices included are devoted to presentation of details of the study. Appendix A provides detailed formulas utilized in the mathematical model. Appendix B summarizes the data bases, inputs used in the comparison and the results obtained.

CHAPTER 2

DATA BASE MACHINE

With the advent of powerful Data Base Management Systems (DBMS) has come ever increasingly complex software file access methods and larger percentages of time accumulated in overhead functions due to this software. Secondarily, existing DBMS software requires very large amounts of storage to function with any time efficiency [15]. Typically, these systems tend to be cumbersome to maintain. These problems are but a few of those attacked by researchers attempting to develop techniques to provide the services of the conventional software DBMS without the inherent overheads and deficiencies. The DBM discussed herein is one solution which provides specialized hardware controlled by relatively small, simple software packages. The need for the DBM and the associated hardware required is discussed in the following subsections.

The Need for DBM

It is readily apparent that the data processing world of today and the future will require greatly enhanced accessibility to massive volumes of data. Two factors which are critical to the effective accessibility of this data are reliability and performance.

In discussing reliability, two aspects are relevant: size and complexity. Conventional DBMS are massive pieces of software which require large amounts of storage [10,15]. This problem arises from the requirements of flexibility to access data. First, to provide flexible access to the data, conventional DBMS all use multiple methods of file access [33]. For example, under IBM's Information Management System,

HDAM, HSAM, HIDAM, HISAM, TOSAM, ISAM, OSAM, and GSAM access methods are used [15]. Second, conventional DBMS employ numerous control structures to maintain positioning while accessing data in a data base. Examples of these control structures are inverted lists, scratch pad areas and currency pointers. All these control structures require storage and software to maintain [33]. Third, with the file access flexibility provided through software, the algorithm implemented for parsing of requests to determine what is to be accessed and by what means often are sizable units of software [15]. Fourth, with multiple users accessing multiple files comes the requirement to control the users for security reasons as well as the files for data integrity. Last, all of these controllers must be integrated under a master controller. This multiplicity of control functions can currently be implemented only by software which is voluminous [15].

In discussing complexity of program logic as it relates to reliability, one need only understand the previous examples. Indeed, with the sheer size of current DBMS has come a complexity of internal coding rivaled only by operating systems [15,33]. When developing a necessarily large, complex piece of software, researchers are at another disadvantage: few effective software verification or proving techniques are available. Those that are have been effectively applied in production only to simple small systems. No efforts have been successful in proving the validity of anything as massive as a complex DBMS [14]. Thus, software implemented DBMS are large and necessarily complex software and current technology provides programmers with little or no help in assuring the correct functioning of this code.

The DBM can alleviate a significant portion of the reliability

problems of a conventional DBMS. First, the flexibility of access is provided by a hardware unit (discussed in subsection Hardware of this chapter). This unit provides numerous text processing capabilities (discussed in Chapter 3). The unit allows the multiple file access methods to be eliminated from software. Under the DBM concept, the logical data structure is stored directly on the storage device and requires no transformations. This storage technique makes the hardware primitives adequate for the access operation [16,17]. Second, the DBM does not utilize extensive control structures. Thus, the control structures and controller structure software required by conventional DBMS is simplified nearly to the point of being effectively eliminated. Since the multiplicity of file access methods and the compound multiplicity of control structures is eliminated, a simplified request processing is possible. This occurs because the hardware unit and its text processing logic require very limited software command processing. Thus, the task of integrating the controllers is also made simpler. Last, multiuser control remains a software controlled element. Overall, the software required for most significant functions of a conventional DBMS is either eliminated entirely or greatly simplified by the DBM.

By redesigning numerous elements of the conventional DBMS into hardware, a degree of reliability is achieved through verification of the correctness of hardware. While portions of the DBM environment are still software, they are smaller and simpler. The more complex elements are supported directly in hardware which is currently verifiable [14]. Additionally, trends indicate that future research will provide better hardware verification techniques [14]. Thus, with size and complexity lessened and the hardware verification capability, reliability is

greatly enhanced.

The second factor critical to effective access of data is performance. General purpose von-Neumann type computers were designed for sequential data processing and numerical computing. These general purpose computers were not designed to provide for the concurrency and random text manipulation required in a DBMS. Compounding this design misorientation problem is the fact that under conventional DBMS both the control for locating a unit of data and the actual accessing of files is channeled through the CPU, which is a potential bottleneck in most systems [14,15,16]. The concurrency available through multiple channels and storage devices is minimal, restrictive and not well utilized. Thus, not only is the design of today's popular CPUs inefficient for data base processing, the software DBMS must utilize the CPU for almost all of its functions.

The second set of reasons for utilizing DBM environments relates to actual run-time performance. First, the DBM is as its name implies a specialized data base processor, not a von-Neumann type general purpose CPU. The DBM's hardware design provides three capabilities [2,25,29]:

1. text-oriented processing
2. rapid, independent, sequential retrieval
3. concurrency of operations

The text oriented processing is provided via the microprocessors specialized logic units. The rapid retrieval is provided by the relational associated processor, RAP, arrays. The concurrency of operation is a two-fold benefit. Initially, as in most storage units the main CPU issues a request for I/O. Under a conventional DBMS, only one logical record is requested per I/O request. Under the DBM, concept

all logical records required can be available to the main CPU in only one I/O request. This benefit derives from having logical capabilities built into the storage device. This concurrency is appropriately labeled CPU-DBM concurrency. The second form of concurrency exists in the operation of the RAP arrays. These multiple microprocessors built into the storage device provide for concurrent access to different portions of a single file. This type of concurrency can be labeled DBM-RAP concurrency.

In summation, the needs for DBM concepts are twofold, reliability and performance. Enhanced reliability is obtained in the move from unverifiable software to verifiable hardware. Enhanced performance is achieved by designing hardware specifically for the task, i.e., utilizing specialized text processing systems and capitalizing on the concurrency inherent in the DBM's operation.

Hardware

The specialized hardware involved in the data base machine utilizes components used widely today, but utilizes them in a somewhat different manner [5,16,23]. Figure 1 provides an overview of a data base machine relational associative processor (RAP). The reader should note the RAP unit is a separate hardware storage device. The RAP unit contains three basic components: the controller, the set function unit and a variable number of parallel cells.

The controller is responsible for coordination of cells, for controlling the set function unit, for decision processing and the handling of RAP functions. It is via the controller that a RAP unit would receive requests for data from the host computer. The set

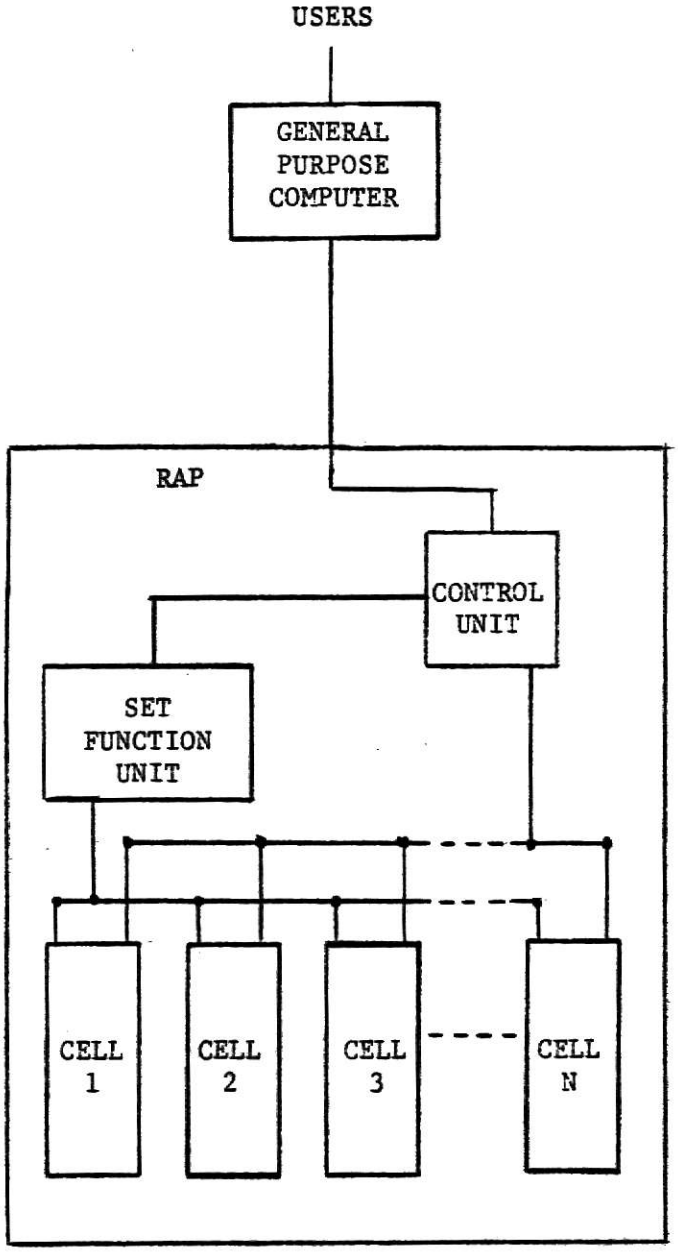


Figure 1

RAP Architecture Overview

function unit provides the means to logically combine the results received via the cells. The cells are tiny microprocessors, as can be seen in Figure 2. Each cell consists of six components: an information search and manipulation unit (ISMU), an arithmetic logic unit (ALU), a buffer area, a read head, a write head and a memory unit. Additionally, the ISMUs of adjacent cells are connected to facilitate further information/control flow. When basic functions are deciphered and search arguments established by the controller, the ISMU will evaluate the search criteria, effect I/O transfers and control the ALU during modifications. In the ALU would reside serial adders, multipliers, counters and arithmetic logic. These units provide for summation and minimization/maximization primitives. The buffer area would be of a size appropriate for the data structures to be supported.

The RAP cell allows assimilation of data from different portions of the storage at exactly the same time, i.e., true concurrency. This concurrency is provided by multiple "intelligent" cells accessing distinct portions of the data base via the coordinating controller. Additionally, multiple RAP units may be utilized.

In implementing data base systems on a data base machine, some logical organization of the data is required. Due to the nature of RAP, normalized relational data structuring (very similar to the normalized relations as defined by Codd) tends to be regarded as the most appropriate structure [23,29]. The normalized relations described provide a tabular view of the data base. The tabular relationship of tuples tends directly toward a linear mapping as is required, for example, on a disk-like unit. Herein lies the method by which RAP eliminates the need for mapping and transformation of logical structures

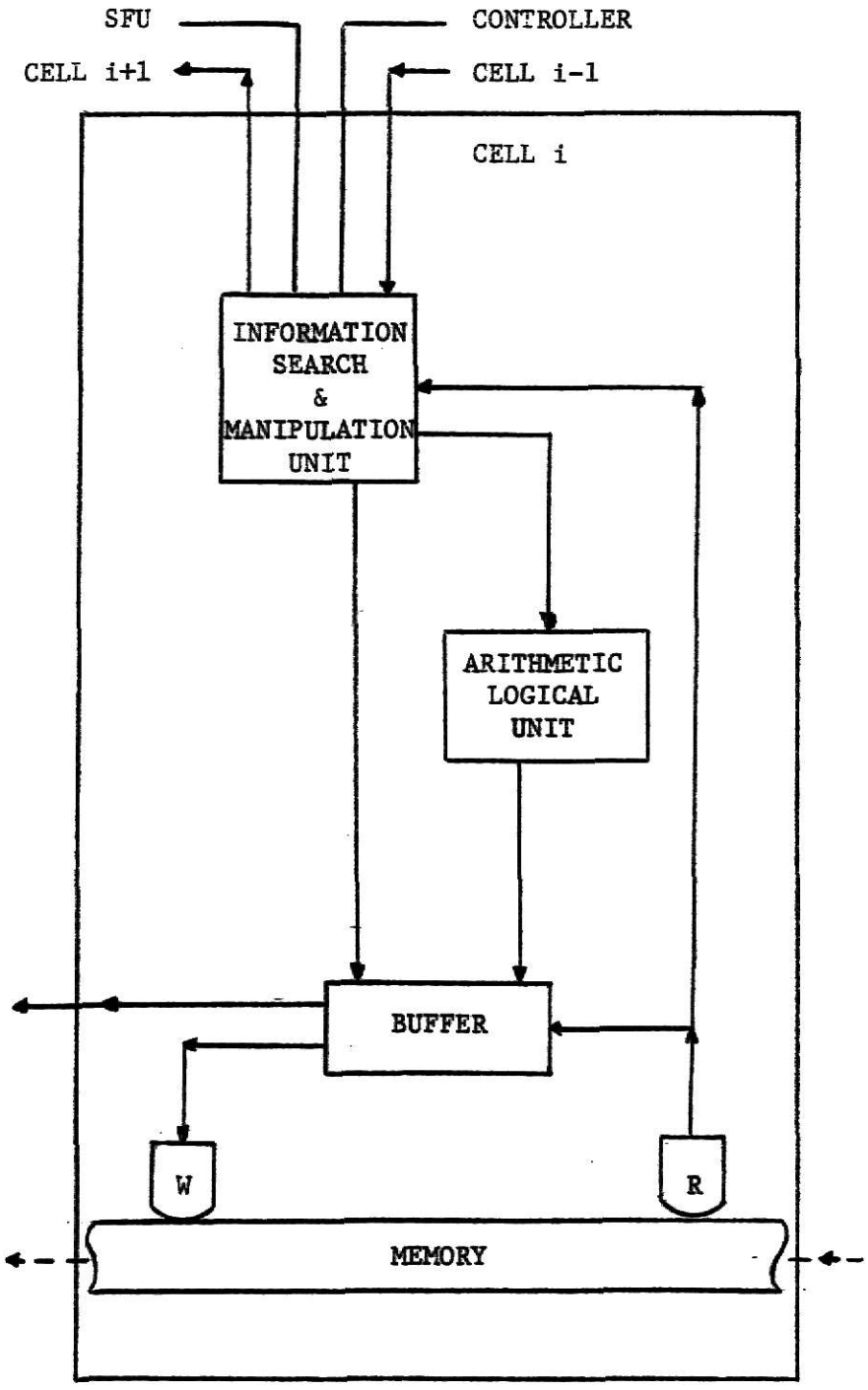
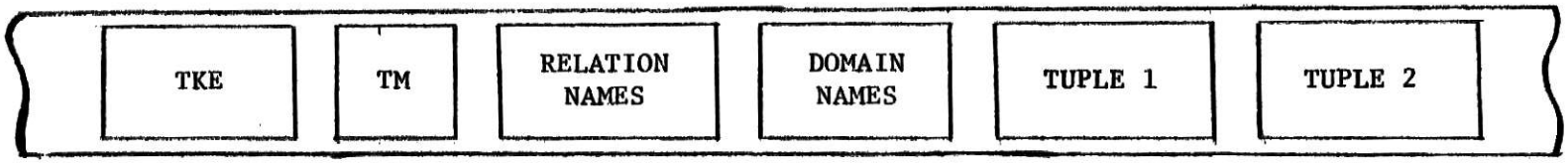


Figure 2

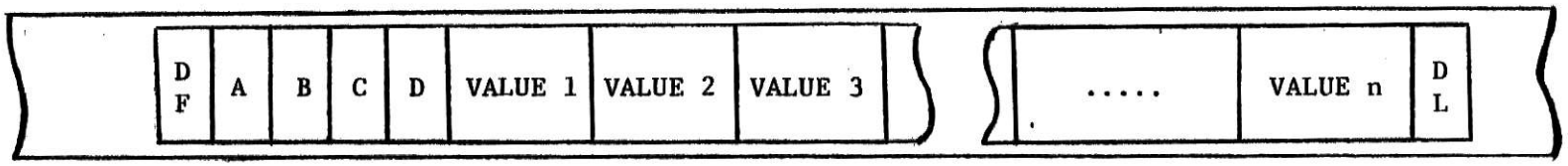
Cellular Architecture

to physical devices. As is illustrated in Figure 3, the tuples of the relations can be stored one after the other on a physical track [23]. This file structure is quite similar to a file on a conventional disk device. However, the data machine performs a rapid, concurrent sequential access of distinct portions of the data base. As illustrated, the first two blocks of data contain relational and domain names. Succeeding data blocks contain the tuple values. In the illustration, "DL" represents delimiters. The beginning and end of track are indicated as "TKE". Several flag and mark bits are illustrated. These allow for logical control information to be stored right with the associated tuple. For example, the delete flag "DF" is set on to indicate a deleted tuple. Dynamic garbage collection is utilized with data compressed toward the front of the track. It should be pointed out that the length of a RAP relational tuple is physically limited to the cell buffer length (typically 1K bits).

RAP is designed in such a manner that relation tracks need not be sequential or contiguous [24]. In earlier access methods, this has provided increased input/output efficiency and thus was incorporated in RAP [23]. With the functional capabilities incorporated into the RAP cell, output only occurs when a set of tuples satisfying the request are sent down the channel to the host computer. The parallel cells are controlled so as to achieve a pipeline transmission.



Track Data Structure



Tuple Data Block

Figure 3

RAP Data Format

CHAPTER 3

DBM/IDMS MATHEMATICAL MODEL

Functions of the DBM

The DBM architecture, as described thus far, has always indicated a highly specialized functional capability. These specialized capabilities are quite logical extensions when considering the deficiencies of the von-Neumann type general purpose computer in handling extensive text oriented processing. Typical specialized functions of a data base machine are [23,24]:

1. Retrieval
2. Modification
3. Deletion
4. Insertion
5. Selection of a Minimum (or Maximum) Value
6. Selection and Counting of Specific Data Values
7. Selection and Summing of Specific Data Values
8. Selection and Averaging of Specific Data Values
9. Cross Retrieval
10. Projection
11. Sorting

In noting these functions, special attention must be given to the point that the DBM performs the functions in hardware. The need for the highly specialized hardware described earlier becomes readily apparent in view of the number and complexity of these functions. While various of these functional capabilities are emulated through software in the available software DBMS, they require large overhead times in maintenance of control structures (for example, inverted lists) and must

utilize the less efficient general purpose computer architecture in order to provide the emulation [4].

DBM and DBMS Functional Capabilities

Few software DBMS provide most of the specialized functions of the DBM. To compare all DBM functions, a cross section of DBMS would be required for the modeling process. In an effort to provide a direct accounting of both types of data base handlers, conventional and DBM, the study was restricted to more direct comparisons. Eventually, the paring process indicated the best understanding of the data base machine was derivable from the simplest and most widely utilized functions.

Thus, the results presented later are for the four most basic operations necessary for even the least complex data base handler to be useful. The functions are:

1. Retrieval
2. Modification
3. Deletion
4. Insertion

In preparing final phases of this study, the study was specialized to a point where "real-world" applicability could be obtained. To this end, the internal workings of Cullinane's IDMS data base management system were studied. IDMS was selected for three basic reasons:

1. It is based on a CODASYL network model
2. Reasonably wide-spread use in the "real-world"
3. Availability of documentation on the internal processes of IDMS

Following the study of IDMS, formulas for the software DBMS utilized in earlier phases of this study were modified to reflect the actual methods

used by IDMS. As a second support of the "real-world" applicability of this report, the characteristics of functioning data bases from a wholesalers application were used as the target data bases for the modeling process (see Appendix B, "Modeling Environment and Results"). Further support of "real-world" applicability is given because the DBM formulas were tailored to represent the recently developed NCR DBM.

The following subsection provides insight into how the actual model was developed using the data provided.

Parameterization of the Model

To provide a statistical base from which to derive conclusions concerning the performance characteristics of the DBM, a mathematical model was developed. This model was simulated to provide ratios of times to perform given functions on the DBM versus the software DBMS performance times. To enhance portability for further research and because of its mathematical functionability, FORTRAN was selected as the source language for the computer simulation program. Two key factors in developing the model were data transfer rates and access times of the storage devices. Following are the relevant parameter values utilized throughout the analysis:

1. CCD cycle time - 2 msec., 4k memory with 2M hertz rate
2. Transfer rate between CCD and MO5 processor at the data base machine - 1M hertz
3. Disk access time on conventional computer - 50 msec. for 6144 bytes + 25 msec. software time.

Rotational time was left as a variable in the model so as to be easily modifiable for future use.

The DBM can have a variable number of microprocessor controllers

operating upon the data base. Controller units numbering from two to 128 were modeled in the study.

For each of the files utilized in the study, statistics were gathered for operations involving variable numbers of records. These record range categories are as follows:

1. 1 record
2. 10 records
3. 100 records (where applicable)
4. 1000 records (where applicable)
5. 20% of the total records
6. 40% of the total records
7. 60% of the total records
8. 80% of the total records
9. All of the records

CHAPTER 4

RESULTS

The results presented in this section are the distillations of volumes of statistical data from the modeling studies. To enhance the visibility of trends in performance characteristics, a brief summation of the results for the functions simulated is provided graphically. These graphs provide a mapping of the ratio (R) versus number of records accessed. The ratio, R, is derived from the execution time of the DBM for the given function and number of records of a given data base divided by the execution time of IDMS performing the identical function on the same number of records of the same data base. Mathematically, this can be represented as:

$$R = \frac{t_{IDMS}\ (m_i,\ db_j,\ f_k)}{t_{DBM}\ (m_i,\ db_j,\ f_k)}$$

where

t_{DBM} - the data base machine execution time

t_{IDMS} - the IDMS execution time

m_i - the number of records involved

db_j - the data base containing the records accessed

f_k - the function being performed

Each of the graphs, contains the results for four ratio/controller configurations. This technique allows for visual comparison of the effect of adding additional controllers. It must be noted that to accommodate such a wide range of results, the graphs will not have identical scales. In all cases, the same data base was utilized. The

data base utilized contains 4000 230 byte records. For further details concerning the data bases and input parameters to the mathematical model, see Appendix B, "Modeling Environment and Results."

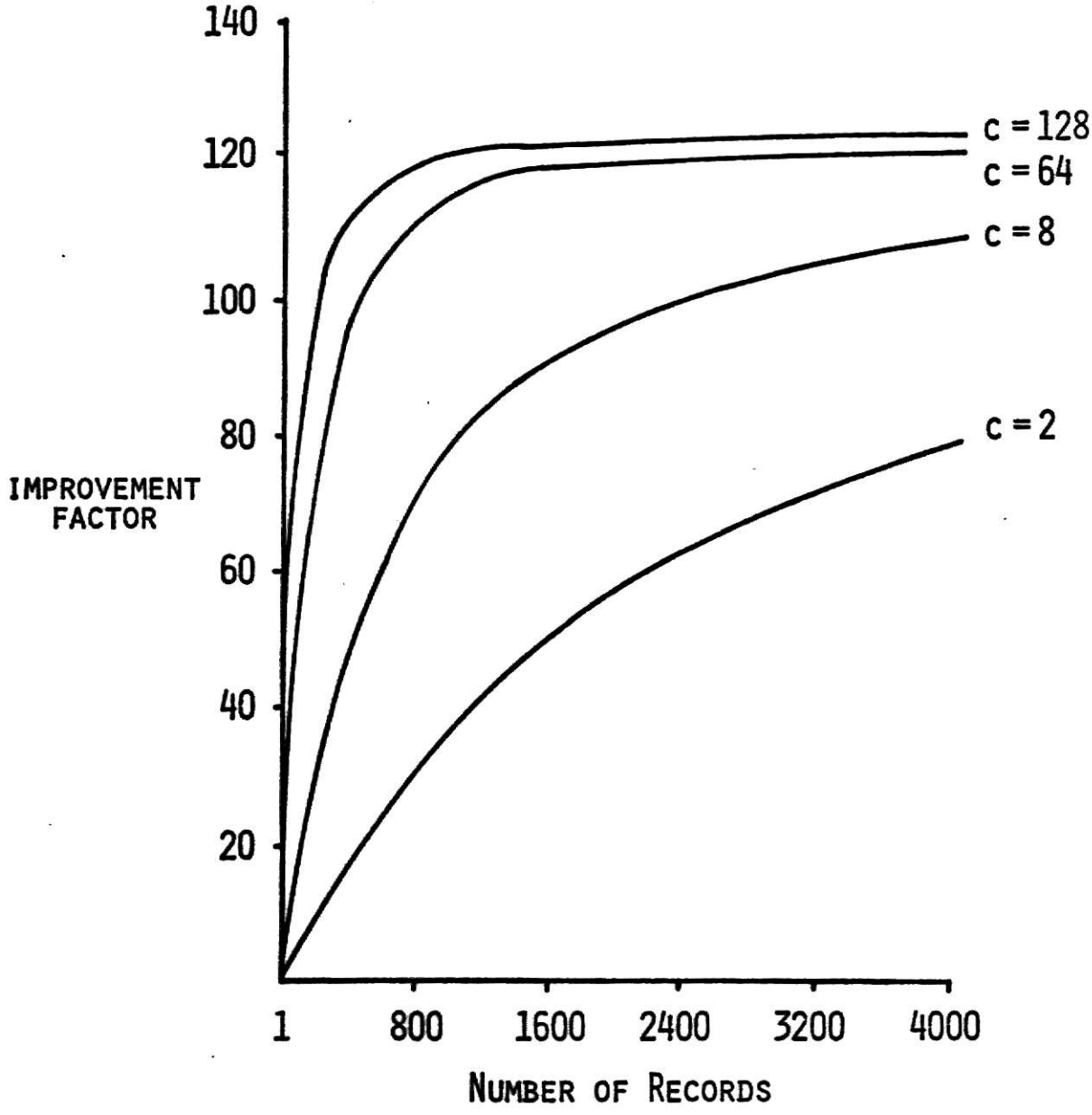
In the graphs a ratio of 7.50 indicates that in order to fulfill the specified request, IDMS would require seven and one half times as much time as would the DBM.

In the following subsections, the phrase "control structures" will be used often. IDMS utilizes currency pointers and interrecord pointers as its control structure.

Retrieval

A retrieval operation consists of reading all records from secondary storage to main memory which satisfy a specified criteria. Under conventional DBMS, such as IDMS, the select keys must be predefined. The DBM content addressability allows any field to be a key. To provide for more direct comparison, the same fields were used as keys for both data base handlers. The keys were assumed to be predefined. This same predefined key structure restriction is very significant. If a nonkey field under IDMS or any conventional DBMS is to be checked for record selection, records would have to be checked in core and included or excluded accordingly. Thus, records which may not be exactly the ones desired must be read. The DBM alleviates this deficiency via content addressability at the storage device.

Figure 4 illustrates the performance ratio projections for the retrieval operation. An example request for this type of retrieval might request all customer records with customer number 87654321, balance of zero, size of 600, class of 'K' and gross of $1,000,000



RETRIEVAL PERFORMANCE

WHERE C IS NUMBER OF REPLICATED
RELATIONAL ARRAY PROCESSORS

Figure 4

(note: five predefined key fields).

As can be seen on the graphs, as the number of records retrieved increases so does the relative performance of the DBM. Next, as the number of controllers is increased the performance curves increase also. These factors are produced by several features of the DBM. First, as number of records accessed increases the content addressability capability of the DBM reduces significantly the search time, since no control structures must be accessed as in conventional DBMS. Also, the searching for data in the DBM is performed at hardware speeds. The control structure manipulations must occur at software speeds and, for example, in the case of large inverted list structures may additionally require I/O to complete, e.g., the inverted list may not be wholly core resident. Second, one request for I/O is required by the DBM irrespective of the number of records required. Under the conventional DBMS, such as IDMS, multiple I/O requests must be issued. Third, as the number of controllers increases a compounding of concurrency benefits occurs. All benefits of retrieval are amplified by additional concurrent processors.

From the graph, a plateau effect is evident. This suggests that for retrieval there is a point of diminishing returns after large numbers of records are accessed or large numbers of controllers are used. While important to note, this point is not as detrimental as it might seem; the DBM is still capable of operating two orders of magnitude faster than a conventional DBMS.

Modification

All DBM updates can be performed by the specialized DBM hardware. The modification operation is a two function operation. First, a record must be located which satisfies the key requirements and then the record will be modified. Second, the record must be rewritten on secondary storage. Thus, statements quite similar to the DBM retrieval function apply for the modification record location step. More importantly, however, under conventional DBMS, the record must be brought to main storage and modified one record per I/O request. Under the DBM, a single update request can update multiple records of the data base. Additionally, under the DBM, multiple field updates require at worst one additional revolution per update to complete.

Figure 5 illustrates the performance ratio projections for the modification operation on one field. An example modification request is for all customer records with credit status 'A' set credit limit to 100,000. Note that for the DBM the requested records would be updated at the storage unit. IDMS would locate a single record, bring it to core, allow an application program to test and possibly modify the record, IDMS would rewrite the modified records and update all control structures. If any selection criteria fields were not key fields, additional in-core tests would have to occur for the IDMS system.

As stated, the benefits accrued under retrieval apply to the modification location step. This benefit is enhanced significantly by the DBM hardware update capabilities. The use of inefficient in-core software updates and one update per I/O request severely limits the conventional DBMS. When control structure manipulation inefficiencies

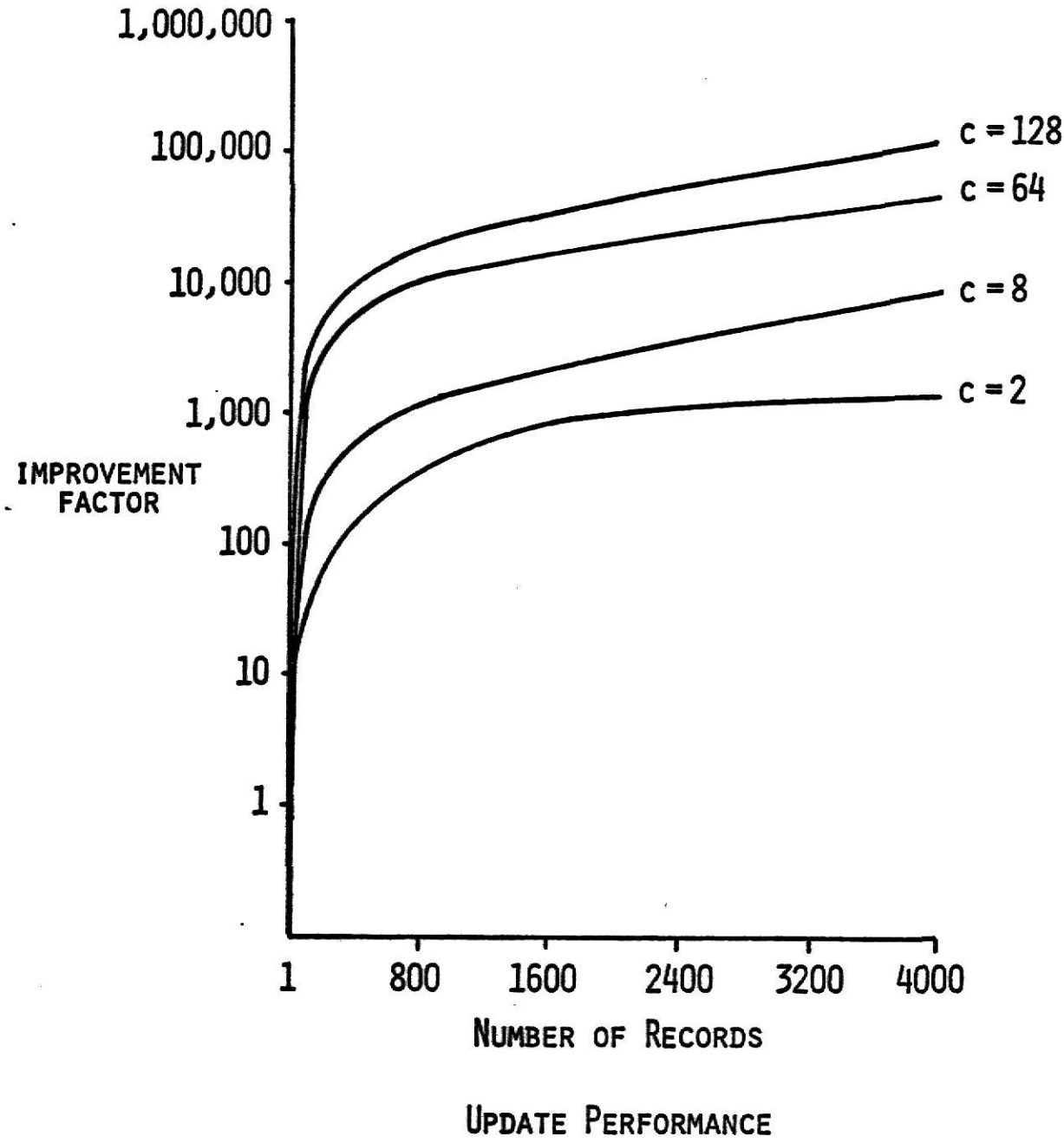

1,000,000
100,000
10,000
1,000
100
10
1
IMPROVEMENT FACTOR
c = 128
c = 64
c = 8
c = 2
1
800
1600
2400
3200
4000
NUMBER OF RECORDS
UPDATE PERFORMANCE

Figure 5

are also considered, it becomes easier to understand the scaling necessary to show the rather substantial performance factors of the DBM on the graphed performance measures. It should be noted that multifield updating will lessen the performance benefits of the DBM; however, since the initial performance ratio indicates four orders of magnitude, this slight deficiency is acceptable.

As the number of records accessed increases, the specialized hardware benefits (no control structure, storage device updating, etc.) compound quickly. As the number of controllers is increased, a second compounding effect occurs; the performance ratio is enhanced substantially by the greater levels of concurrency when using greater numbers of controllers. A general flattening of the curves again seems to indicate a point of diminishing returns when large numbers of records are accessed and/or more controllers are added. However, the scaling factors exaggerate this flattening effect. For the probable ranges of records accessed, the DBM is capable of performing on the order of 140,000 times faster than IDMS.

Deletion

Typically, deletion operations are performed by marking a given piece of data as deleted and then reusing the available space. In the case of conventional DBMS, this marking is preceeded by access to a control structure(s) to locate the record. An I/O request is initiated to mark the desired record. Then the control structure(s) is updated to reflect the new deleted record. The first and last steps are oftimes combined into a single step. Under the DBM, there is no control structure to access and update. Multiple deletions can be performed in

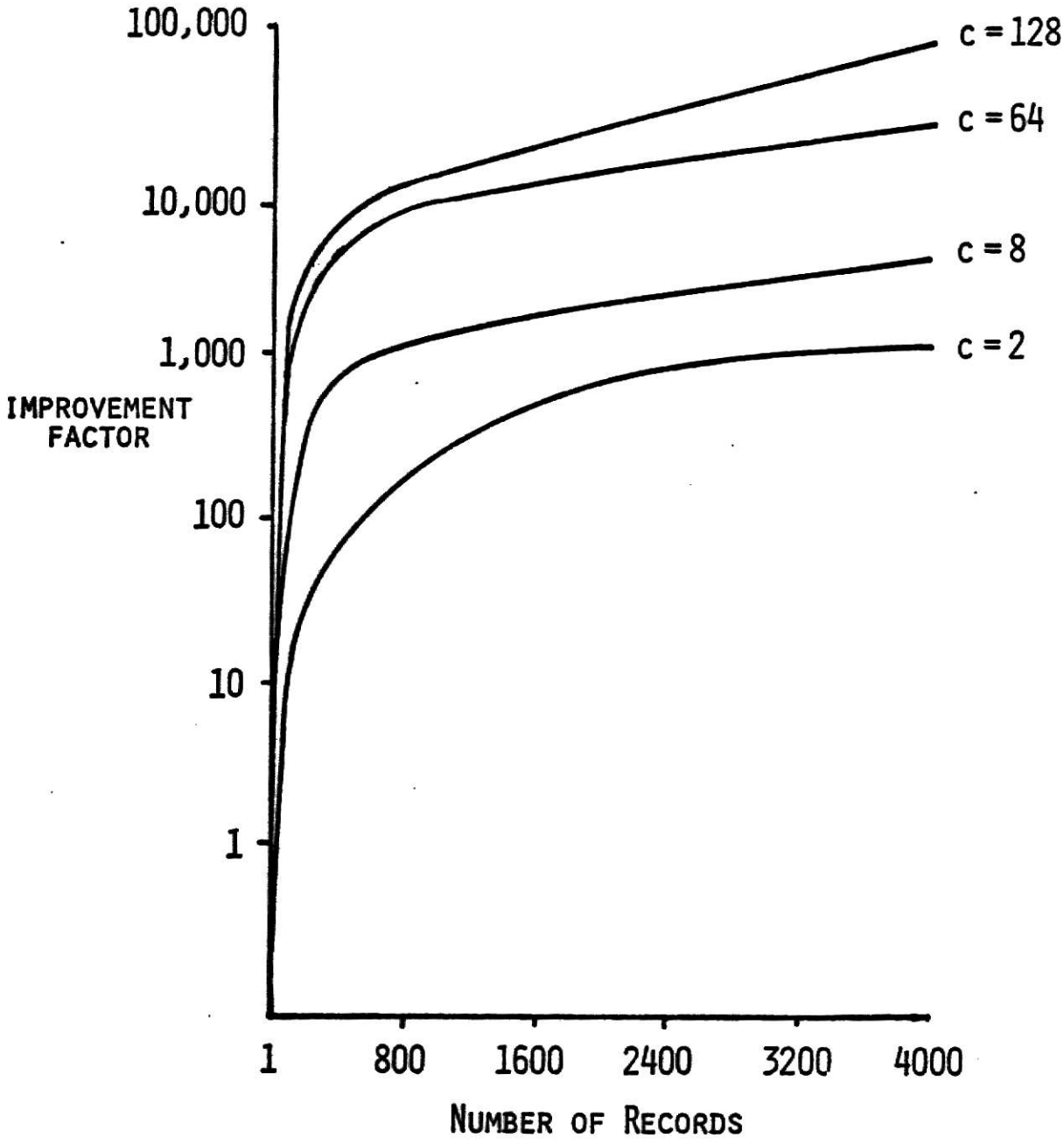
a single request.

Figure 6 illustrates the performance ratio projections for the deletion operation. An example deletion request is delete all customer records with a balance of zero and credit limit of zero. Under IDMS, this possibly multirecord update would require multiple accesses/updates of the control structures and multiple I/O requests to perform the deletion on all records. If either balance field and/or credit limit were not key fields, all records would be read into core, tested and possibly marked. This method is clearly lacking when compared to simply requesting the deletion be done by the DBM in a single request.

Again the statements made under the retrieval subsections generally apply. In the case of the DBM, each tuple's delete flag is turned on. The actual marking operation for the DBM or DBMS are not significantly different, since both require a writing operation. However, compounding the benefits accrued by a DBM retrieval-like access are the benefits of concurrently updating the flags of more than one tuple. The more records and/or controllers involved, the greater the benefits accrued.

As was the case in retrieval and updating, a point of diminishing returns is evidenced in the graphs. Again, caution must be exercised due to the scaling factors involved. The DBM can delete larger groups of records up to 90,000 times faster than IDMS. Therefore, the points of diminishing returns are not likely to be highly important.

One last point concerning deletions is garbage collection. Under the DBM, the buffers can be "short circuited" processing an immediate read then write, thus, moving all data ahead one position per revolution.



DELETION PERFORMANCE

Figure 6

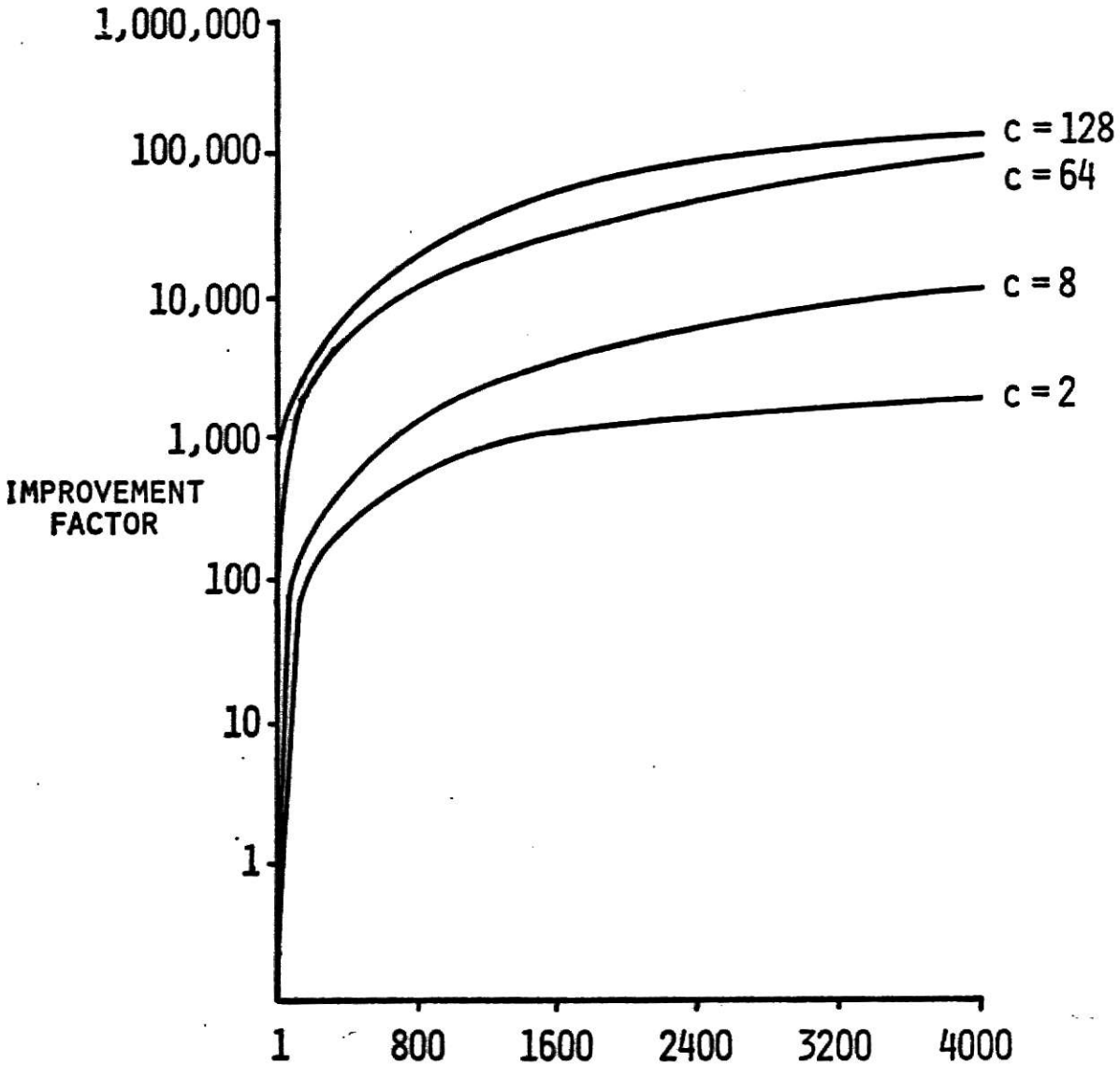
Insertion

An insertion operation consists of locating the area for a given tuple and writing it. With conventional DBMS, this operation will consist of updating control structures and issuing write requests for each record. The locating of a given free position for conventional DBMS can be highly time consuming, especially in the cases of large inverted lists inverted on multiple keys and requiring secondary storage to maintain these lists. Indeed I/O and manipulation of inverted lists could take longer than I/O for the records. With the DBM, new tuples are inserted at the end of a given cell. It is most important to remember the DBM RAP cells can communicate directly and concurrently to provide the required location for the insertion.

Figure 7 illustrates average insertion time performance ratio projections. The average labeling is due to the condition that a RAP cell of the DBM may begin writing at one of three times:

1. Immediately upon receipt of the request
2. On the average of 1/2 revolution
3. After some portion of a revolution

The first time described above denotes the case where a given insert request can fill the memory area currently being read. The second time is the average time for the cell's free position to be located. If the DBM's data base is optimally loaded for the query environment, this average time will be roughly one-half revolution of the memory unit (addressing a disk-like unit). Last, the third time is the actual time to locate first available slot at the end of the cell. This time is based on the currently existing state of the cell. It



AVERAGE INSERTION PERFORMANCE

Figure 7

could conceivably be either of the other two. For the purposes of this study, it is assumed to be the last possible location in a cell. Thus, a best case/average case/worst case analysis could be taken. For the purpose of this discussion, the average case is presented. Details of both other cases can be found in Appendix B, "Modeling Environment and Results."

While proper ordering of records in the data base is critical to a conventional DBMS because of its control structures, the DBM is an associative device; thus, ordering is not relevant. It is for this reason that no mention was made of searching for a logical sequence under insertions via the DBM.

In studying the graphs for insertion, the effects of maintaining control structures and performing multiple I/O requests for multiple record insertions are again observed. When inserted records have multiple keys, the control structure inefficiencies are again compounded. Conversely, the DBM requires no control structures to maintain and only one I/O request per multiple records is necessary. Thus, as number of records increases, relative overhead times are decreased. As number of controllers increases, the concurrency effect provides a substantial compounding effect.

Special care should be exercised when studying this graph. The flattening curve/diminishing return effect is illustrated, but amplified by the exponential scaling. Secondarily, also due to scaling the curves appear to converge when actually they do not. Even if the diminishing return appearance of the graph is not discounted, the DBM is capable of performance five orders of magnitude greater than IDMS.

CHAPER 5

APPLICATIONS AND EXTENSIONS

Applying the Analysis

The results presented in the previous section provide relative performance of the DBM versus IDMS. These results can be used to provide further insight for potential DBM users. This insight would be in the form of a performance coefficient (PC). In order to determine the relative performance of the DBM in a given environment, three pieces of job mix information are required:

- n - the number of distinct operations to be performed on the data base
- p_i - the probability of the ith operation being performed
- r_i - the performance ratio of the ith operation as determined in the manner of the preceeding section

The performance coefficient formula is:

$$PC = \sum_{i=i}^{n} (p_i * r_i) \text{ where } \sum_{i=i}^{n} p_i = 1$$

It is important to note that operations are defined as distinct if they access different files, perform a different function or involve a different number of records from a file.

For the purpose of illustration, two examples of the performance coefficient equation results using the wholesaler data bases are presented in this section. The results are presented only for the case when 16 controllers are present.

The job mix of example one indicates that for systems having a more even distribution of functions, the DBM will outperform IDMS 8985.08 to

EXAMPLE 1
Job Mix Data

Operation

Function	File	No. of Records	Frequency
Retrieval	Inventory	100%	.25
Retrieval	Customer	100%	.20
Insertion	Inventory	100%	.25
Deletion	Inventory	40%	.30

Data Base Machine vs IDMS

(See Appendix B for r_i values)

PC = .25 * (117.21) + .20 * (116.64) + .25 * (29629) + .30 * (5084)
= 8985.08 improvement factor for DBM over IDMS

EXAMPLE 2
Job Mix Data

Operation

Function	File	No. of Records	Frequency
Retrieval	Cutomer	1	.30
Retrieval	Inventory	10	.20
Retrieval	Customer	10	.20
Retrieval	Inventory	100%	.10
Modification	Inventory	1	.10
Modification	Customer	20%	.10

Data Base Machine vs. IDMS

PC = .30 * (2.21) + .20 * (14.12) + .20 * (5.30) + .10 * (117.21)
+ .10 * (3788) + .10 * (16.45)
= 396.713 improvement factor for DBM over IDMS

one. An extremely substantial savings by the DBM.

The job mix of example two also indicates that for systems using the appropriate function mix the DBM will allow overall data base operations to occur 396.713 times faster than would IDMS. With the savings in time and resources combined, this PC indicates a very strong perference for the DBM.

One important factor which is considered constant between the examples is, of course, the application software time, since this is both totally variable and unpredictable and not a part of the actual data base operations.

In both examples, the DBM shows a clearly superior performance coefficient. Upon deriving performance coefficients for various environments, the author found this clear superiority in all but the most simplistic of job mix configurations.

Extensions of the Analysis

Several areas of this study could be extended to provide more insight into the DBM's performance characteristics and make the study more "real-world" applicable. Five of these possible extensions follow. First, it is known generally what pieces of software are still necessary for the functioning of the DBM. However, at the time of the study, the actual execution times were not available. Hence, selecting a particular mainframe on which to use the DBM and timing the software portion of the DBM could provide valuable insight into the possibly greater efficiencies by the DBM. Second, throughout this study only one DBM was utilized. Multiple DBMs would assuredly produce even more favorable results than the single DBM environment. Third, while many

users utilize IDMS, many do not. The formulas for the conventional machine could be modified to model other common DBMS, e.g., IBM's Information Management System or Cincom's TOTAL. Thus, more users could derive comparisons. Fourth, relatively small data bases were utilized since the DBM will most probably be utilized on smaller systems initially. Utilizing larger data bases with varying keys may provide valuable knowledge on tuning the DBM environment for large (as well as smaller) system applicability. Additionally, larger data bases would provide larger shops with more direct comparative information. Last, but certainly not least important, a cost/benefit study would be undertaken; potential users must determine their benefits and associated costs of using the DBM over use of the conventional DBMS.

CHAPTER 6

CONCLUSION

One general conclusion can be drawn from the results, the DBM will clearly out perform conventional DBMS for multirecord manipulations. This peformance can be attributed to four basic factors. First, fewer I/O requests must be performed to accomplish multirecord manipulations in the DBM environment. Second, no time is spent in updating control structures under the DBM environment. Third, concurrent operating levels generated by multiple controller units under the DBM are unattainable in IDMS on a von-Neumann machine. Fourth, text processing capabilities on the storage device as in the DBM provides numerous performance benefits.

From the performance coefficient formula, potential users can determine how a given job mix run under IDMS will perform in the DBM environment. The techniques, results, formulas and conclusions drawn from this study, potential DBM users who are not using IDMS can develop studies for their situations.

SELECTED BIBLIOGRAPHY

1. Anderson, G.A., and R.Y. Kain, "A Content Addressed Memory Designed for Data Base Applications," Proceedings of 1976 International Conference on Parallel Processors, 1976, pp. 191-195.

2. Babb, E., "Implementing a Relational Database by Means of Specialized Hardware," ACM Transactions on Database Systems, Vol. 4, No. 1, March 1979.

3. Bannerjee, J., R. Baum, and D.K. Hsiao, "Concepts and Capabilities of a Database Computer," ACM Transactions on Database Systems, Vol. 3, No. 4, December 1978, pp. 347-384.

4. Berra, P.B., and E. Oliver, "The Role of Associative Array Processors in Data Base Machine Architecture," Computer, March 1979, pp. 53-60.

5. Bray, O., and K.J. Thurber, "What's Happening with Data Base Processors?" Datamation, January 1979, pp. 146-156.

6. Canaday, R.H., R.D. Harrison, E.L. Ivie, J.L. Ryder, and L.A. Wehr, "A Back-end Computer for Data Base Management," CACM, Vol. 17, No. 10, October 1974, pp. 575-582.

7. Champine, G.A., "Four Approaches to a Data Base Computer," DATAMATION, December 1978, pp. 101-106.

8. Codd, E.F., "A Relational Model of Data for Large Shared Data Banks," Communications of the ACM, Vol. 13, No. 6, June 1970, pp. 377-387.

9. Copeland, G.P., G.J. Lipovski, and S.Y. Su, "The Architecture of CASSM: A Cellular System for Non-Numeric Processing," Proceedings of the 1st Annual Symposium on Computer Architecture, December 1973, pp. 121-128.

10. Cullinane Corporation, IDMS Programmers Reference Guide, Cullinane Corporation, 1977.

11. DeFiore, C.F. and P.B. Berra, "A Data Management System Utilizing an Associative Memory," Proceedings of AFIPS 1973 NCC, Vol. 42, AFIPS Press, Montvale, N.J., pp. 181-185.

12. DeFiore, C.F. and P.B. Berra, "A Quantitative Analysis of the Utilization of Associative Memories in Data Management," IEEE Transactions on Computers C-23, Vol. 2, February 1974, pp. 121-133.

13. Hsiao, D.K., K. Kannan, and D.S. Kerr, "Structure Memory Designs for a Database Computer," Proceedings of ACM 77.

14. Hsiao, D.K., "Data Base Machines are Coming, Data Base Machines are

Coming!" Computer, March 1979, pp. 7-9.

15. International Business Machines Corporation, IMS/VS Version 1 System Programming Reference Manual, San Jose: International Business Machines Corporation, 1978.

16. Kerr, D.S., "Data Base Machines with Large Content Addressable Blocks and Structural Information Processor," Computer, March 1979, pp. 64-79.

17. Lin, C.S., D.C.P. Smith, and J.M. Smith, "The Design of a Rotating Associative Memory for Relational Database Applications," ACM Transactions on Database Systems, Vol. 1, No. 1, March 1976, pp. 53-65.

18. Lipovski, G.J., "Architectural Features of CASSM: A Content Addressed Segment Sequential Memory," Proceedings of 5th Annual Symposium on Computer Architecture, Palo Alto, CA, April 1978, pp. 31-38.

19. Love, H.H., "An Efficient Associative Processor Using Bulk Storage," Proceedings of Sagamore Computer Conference on Parallel Processing, 1973, pp. 103-112.

20. Lowenthal, E.I., "The Backend (Data Base) Computer, Part I and II," Auerbach (Data Management) Series, 24-01-04 and 24-01-05, 1976.

21. Maryanski, F.J., P.S. Fisher, and V.E. Wallentine, "Evaluation of Conversion to a Back-End Data Base Management System," Proceedings of ACM 76, pp. 293-297.

22. Moulder, R., "An Implementation of a Data Management System on an Associative Processor," AFIPS Conference Proceedings, Vol. 42, 1973, pp. 171-176.

23. Ozkarahan, E.A., S.A. Schuster, and K.C. Smith, "RAP-An Associative Processor for Data Base Management," Proceedings of 1975 NCC, Vol. 45, AFIPS Press, Montvale, N.J., pp. 379-387.

24. Ozkarahan, E.A., S.A. Schuster, and K.C. Sevcik, "Performance Evaluation of a Relational Associative Processor," ACM Transactions on Database Systems, Vol. 2, No. 2, June 1977, pp. 175-195.

25. Ozkarahan, E.A. and K.C. Sevcik, "Analysis of Architectural Features for Enhancing the Performance of a Database Machine," ACM Transactions of Database Systems, Vol. 4, No. 2, December 1977, pp. 297-316.

26. Parker, J.L., "A Logic Track Retrieval System," Proceedings of IFIP Congress 1971, pp. TA-4-146 - TA-4-150.

27. Schuster, S.A., E.A. Ozkarahan, and K.C. Smith, "A Virtual Memory System for a Relational Associative Processor," Proceedings of 1976 NCC, Vol. 45, AFIPS Press, Montvale, NJ, pp. 855-862.

28. Schuster, S.A., H.B. Nguyen, E.A. Ozkarahan, and K.C. Smith, "RAP2-An Associative Processor for Data Base," Proceedings of 5th Annual Symposium on Computer Architecture, April 1978, pp. 52-59.

29. Smith, D.C.P. and J.M. Smith, "Relational Database Machines," Computer, March 1979, pp. 28-37.

30. Su, S.Y., G.P. Copeland, and G.J. Lipovski, "Retrieval Operations and Data Representations in a Content-Addressed Disc System," Proceedings of SIGPLAN and SIGIR Interface Meeting, November 1973, pp. 144-156.

31. Su, S.Y., "Cellular Logic Devices: Concepts and Application," Computer, March 1979, pp. 11-25.

32. Thurber, K.J. and L.D. Wald, "Associative and Parallel Processors," Computing Surveys, Vol. 7, No. 4, April 1975, pp. 215-255.

33. Tsichritzis, D.C. and F.H. Lochovsky, Data Base Management Systems, New York: Academic Press, 1977.

APPENDIX A

FORMULAS FOR DATA MACHINE ANALYSIS/STUDY

APPENDIX A

FORMULAS FOR DATA MACHINE ANALYSIS/STUDY

Variable Definition

Nmemonic	Usage
AVEINS	Average Insertion Time for Data Machine
BMD	Bytes in Maximum Domain
BQD	Bytes in Qualifying Domain
BR	Bytes per Relation
BRD	Bytes in Read Domain
BT	Bytes per Tuple
BWD	Bytes in Write Domain
C	Number of Controller Units
CCDR	CCD Transfer Rate (1 Mh)
CL	Number of Qualifying Domains/Complexity Level
COUNTC	Count Command Time
COUNTF	Count Function Time
DELDM	Data Machine Deletion Time
DELETEF	Delete Function Time
GETSOF	Software, Rotational Delay and Seek Time
GETTIM	Rotational Delay and Seek Time
IDELCN	Simple Random Access Deletion Time
IMODCN	Simple Random Access Modification Time

Variable Definition

Nmemonic	Usage
INSC	Insertion Command Time
INSCN	Conventional Machine Insertion Time
INSF	Insertion Function Time
IRETCON	Simple Random Access Retrieval Time
LDELF	Logical Delete Function
LT	Loop Time/CCD Cycle Time (2 msec.)
LOG2C	Logarithm base 2 of C
MARK C	Mark Command
MARKF	Mark Function
MAXI	Temporary Variable
MAXINS	Maximum Insertion Time on Data Machine
MAXKEY	Maximum Number of Keys in File
MINC	Minimum Command Time
MINF	Minimum Function Time
MININS	Minimum Insertion Time on Data Machine
MODDM	Data Machine Modification Time
MO5R	MO5R Processor Transfer Rate (1 Mh)
ND	Number of Inverted Attributes
NR	Tuples/Relation (File Size--Records)
OPENST	Search Time for First Open Location in Relation
OPS	Number of Arithmetic Operations in a Function

Variable Definition

Nmemonic	Usage
PART	Temporary Variable
READC	Read Command Time
READF	Read Command Function
RETDM	Data Machine Retrieval Time
REPLACEC	Replace Command Time
REPLACEF	Replace Function Time
RTIME	Disk Access Time
SI	Simple Random Access Time over Data Machine Time Ratio
SOFT	Software Processing Time Constant
TR	Tuples per Relation

FORMULAS FOR DATA MACHINE

ANALYSIS/STUDY

(cont.)

ADC Command Times

$$\text{COUNTC} = \frac{2}{\text{MO5R}}$$

$$\text{INSERTC} = \frac{\text{BT}}{\text{MO5R}} + 1.5 * \text{LT} + \text{OPENST}$$

$$\text{MARKC} = \text{RST} + 1.5 * \text{LT} + \frac{\text{BQD}}{\text{MO5R}}$$

$$\text{MINC} = \text{MARKC} + \frac{\text{BMD}}{\text{MO5R}} + \frac{\text{BMD}}{\text{CCDR}} - \text{LT}$$

$$\text{READC} = 1.5 * \text{LT} + \frac{\text{RST}}{(\text{NR}+1)} + \frac{\text{BRD}}{\text{MO5R}} + \frac{\text{BT}}{\text{CCDR}} + \frac{\text{BQD}}{\text{MO5R}}$$

$$\text{REPLACEC} = \text{MARKC} + \frac{\text{BWD}}{\text{MO5R}}$$

FORMULAS FOR DATA MACHINE

ANALYSIS/STUDY

(cont.)

Primitive Function Times

$$MARKF = MARKC$$

$$READF = READC + (NR-1) * \left[\frac{BRD}{MO5R} + MAX \left[0, \ .5 * LT + \frac{RST}{NR} - \frac{BRD}{MO5R} \right] \right]$$

$$REPLACEF = REPLACEC$$

$$INSERTF = INSERTC$$

$$DELETEF = MARKF$$

$$LDELETEF = MARKF$$

$$COUNTF = MARKF + \frac{C * 2}{MO5R}$$

$$MINF(MAXF) = CL * (MINC + (\log_2 C - (MINC)) + READC$$

FORMULAS FOR DATA MACHINE

ANALYSIS/STUDY

(cont.)

Relational Operation Times

1. RETRIEVAL

   a. CONVENTIONAL

   IRECTCON = (NR/2*RTIME) + (CL*GETSOF)

   = (NR/2*RTIME) + (CL*(GETTIM+SOFT))

   = (NR/2*RTIME) + (CL*(RTIME+SEEK+SOFT))

   b. DATA MACHINE

   RETDM = READF

   $$= \text{READC} + (\text{NR}-1) * \left[\frac{\text{BT}}{\text{MO5R}} + \text{MAX}\left[0, .5*\text{LT} + \frac{\text{RST}}{\text{NR}} - \frac{\text{BT}}{\text{MO5R}}\right]\right]$$

   $$= .5 * \text{LT} + \frac{\text{RST}}{\text{NR}} + \frac{\text{BRD}}{\text{MO5R}} + \frac{\text{BT}}{\text{CCDR}} + \frac{\text{BQD}}{\text{MO5R}} + (\text{NR}-1) * \left[\frac{\text{BRD}}{\text{MO5R}} + \text{MAX}\left[0, .5*\text{LT} + \frac{\text{RST}}{\text{NR}} - \frac{\text{BT}}{\text{MO5R}}\right]\right]$$

   $$= .5 * \text{LT} + \frac{\text{RST}}{\text{NR}} + \frac{\text{BRD+BQD}}{\text{MO5R}} + \frac{\text{BT}}{\text{CCDR}}$$

2. UPDATE

   a. CONVENTIONAL

   IMODCN = (NR*(1.5+4*OPS)) * RTIME + CL * GETSOF

   = (NR*(1.5+4*OPS)) * RTIME + CL * (GETTIM+SOFT)

   = (NR*(1.5+4*OPS)) * RTIME + CL * (RTIME+SEEK+SOFT)

Relational Operation Times

b. DATA MACHINE

MODDM = REPLACEC

3. DELETE

a. CONVENTIONAL

IDELCN = NR * (1.5+2*ND) * RTIME + CL * GETSOF

= NR * (1.5+2*ND) * RTIME + CL * (GETTIM+SOFT)

= NR * (1.5+2*ND) * RTIME + CL + (RTIME+SEEK+SOFT)

b. DATA MACHINE

DELDM = DELETEF

= MARKF

= MARKC

$$= RST + .5 * LT + \frac{BQD}{MO5R}$$

4. INSERT

a. CONVENTIONAL

INSCN = NR * (2*ND+1.5) * GETTIME

= NR * (2*ND+1.5) * (RTIME+SEEK)

b. DATA MACHINE

INSDM =

i. Minimum

$$MININS = \frac{BT}{MO5R} + 1.5 * LT + OPENST$$

ii. Average

$$AVEINS = \frac{BT}{MO5R} + 1.5 * LT + \frac{OPENST}{2}$$

Relational Operation Times

iii. Maximum

$$\text{MAXINS} = \frac{\text{BT}}{\text{MO5R}} + 1.5 * \text{LT} + \frac{\text{OPENST}}{\text{TR}}$$

APPENDIX B

MODELING ENVIRONMENT AND RESULTS

Data Bases Used in Analysis

Name	Bytes per Logical Record	No. Records	Total Size (Bytes)
Inventory	214	1500	321000
Customer	230	4000	920000

Input Data Set -- Inventory File

RTIME = 0.25

TR = 1500.00

BT = 214.00

BWD = 2.00

MAXKEY = 6.00

BRD = 2.00

BQD = 2.00

BMD = 22.00

OPS = 1.00

Number of Records	Retrieval	Modification	Deletion
1	0.80	2.11	1.49
10	1.95	17.47	11.27
100	14.18	171.10	109.03
300	34.24	512.51	326.29
600	53.63	1024.61	652.17
900	66.19	1536.72	978.06
1000	69.45	1707.42	1086.69
1200	74.99	2048.82	1303.94
1500	81.50	2560.93	1629.83

Number of Records	Insertion Minimum	Insertion Average	Insertion Maximum
1	1.30	2.60	297.03
10	13.04	25.97	2970.30
100	130.35	259.74	29702.96
300	391.06	779.22	89108.81
600	782.12	1558.44	178217.70
900	1173.18	2337.66	267326.60
1000	1303.54	2597.40	297029.60
1200	1564.25	3116.88	356435.50
1500	1955.31	3896.10	445544.30

Number of Controllers = 2

Number of Records	Retrieval	Modification	Deletion
1	1.60	4.21	2.97
10	3.84	34.82	22.45
100	25.48	340.94	217.25
300	53.78	1021.21	650.15
600	75.07	2041.62	1299.51
900	86.57	3062.03	1948.86
1000	89.30	3402.17	2165.31
1200	93.76	4082.44	2598.21
1500	98.68	5102.84	3247.56

Number of Records	Insertion Minimum	Insertion Average	Insertion Maximum
1	2.60	5.16	321.35
10	25.97	51.57	3213.47
100	259.74	515.65	32134.65
300	779.22	1546.96	96403.88
600	1558.44	3093.92	192807.80
900	2337.66	4640.88	289211.80
1000	2597.40	5156.54	321346.50
1200	3116.88	6187.84	385615.80
1500	3896.10	7734.80	482019.70

Number of Controllers = 4

Number of Records	Retrieval	Modification	Deletion
1	3.14	8.35	5.89
10	7.47	69.12	44.57
100	42.37	676.86	431.31
300	75.24	2027.38	1290.73
600	93.84	4053.16	2579.87
900	102.31	6078.95	3869.00
1000	104.20	6754.21	4298.71
1200	107.17	8104.73	5158.13
1500	110.31	10130.51	6447.27

Number of Records	Insertion Minimum	Insertion Average	Insertion Maximum
1	5.16	10.16	335.06
10	51.57	101.63	3350.62
100	515.65	1016.33	33506.17
300	1546.96	3049.00	100518.40
600	3093.92	6098.00	201037.00
900	4640.88	9147.01	301555.50
1000	5156.54	10163.34	335061.70
1200	6187.84	12196.01	402074.10
1500	7734.80	15245.01	502592.50

Number of Controllers = 8

Number of Records	Retrieval	Modification	Deletion
1	6.10	16.45	11.62
10	14.12	136.24	87.84
100	63.36	1334.06	850.09
300	93.99	3995.89	2543.98
600	107.24	7988.63	5084.82
900	112.55	11981.37	7625.65
1000	113.68	13312.29	8472.60
1200	115.42	15974.12	10166.49
1500	117.21	19966.86	12707.32

Number of Records	Insertion Minimum	Insertion Average	Insertion Maximum
1	10.16	19.75	342.37
10	101.63	197.53	3423.68
100	1016.33	1975.31	34236.79
300	3049.00	5925.92	102710.30
600	6098.00	11851.85	205420.70
900	9147.01	17777.78	308131.10
1000	10163.34	19753.09	342367.90
1200	12196.01	23703.70	410841.50
1500	15245.01	29629.63	513551.90

Number of Controllers = 16

Number of Records	Retrieval	Modification	Deletion
1	11.53	31.98	22.57
10	25.48	264.79	170.72
100	84.23	2592.83	1652.20
300	107.38	7766.25	4944.38
600	115.48	15526.39	9882.66
900	118.48	23286.54	14820.93
1000	119.10	25873.25	16467.02
1200	120.04	31046.68	19759.20
1500	121.00	38806.82	24697.47

Number of Records	Insertion Minimum	Insertion Average	Insertion Maximum
1	19.75	37.40	346.14
10	197.53	373.96	3461.42
100	1975.31	3739.57	34614.19
300	5925.92	11218.69	103842.50
600	11851.85	22437.39	207685.00
900	17777.77	33656.09	311527.60
1000	19753.08	37395.66	346141.80
1200	23703.70	44874.79	415370.20
1500	29629.62	56093.48	519212.70

Number of Controllers = 32

Number of Records	Retrieval	Modification	Deletion
1	20.77	60.55	42.74
10	42.62	501.28	323.21
100	100.84	4908.63	3127.88
300	115.61	14702.73	9360.49
600	120.10	29393.88	18709.40
900	121.69	44085.04	28058.32
1000	122.01	48982.09	31174.63
1200	122.50	58776.19	37407.24
1500	122.99	73467.31	46756.15

Number of Records	Insertion Minimum	Insertion Average	Insertion Maximum
1	37.40	67.57	348.06
10	373.96	675.72	3480.60
100	3739.57	6757.16	34806.02
300	11218.69	20271.48	104418.00
600	22437.39	40542.98	208836.00
900	33656.09	60814.48	313254.10
1000	37395.66	67571.63	348060.10
1200	44874.79	81085.94	417672.20
1500	56093.48	101357.40	522090.20

Number of Controllers = 64

Number of Records	Retrieval	Modification	Deletion
1	34.66	109.40	77.22
10	64.20	905.78	584.01
100	111.87	8869.58	5651.89
300	120.21	26566.92	16913.83
600	122.55	53112.93	33806.74
900	123.35	79658.94	50699.66
1000	123.52	88507.56	56330.63
1200	123.76	106204.90	67592.56
1500	124.01	132750.90	84485.44

Number of Records	Insertion Minimum	Insertion Average	Insertion Maximum
1	67.57	113.27	349.03
10	675.72	1132.74	3490.27
100	6757.16	11327.43	34902.73
300	20271.48	33982.29	104708.10
600	40542.98	67964.56	209416.30
900	60814.48	101946.80	314124.60
1000	67571.63	113274.30	349027.30
1200	81085.94	135929.10	418832.80
1500	101357.40	169911.50	523541.00

Number of Controllers = 128

Input Data Set -- Customer File

RTIME = 0.25

TR = 4000.00

BT = 230.00

BWD = 2.00

MAXKEY = 5.00

BRD = 2.00

BQD = 2.00

BMD = 80.00

OPS = 1.00

Number of Records	Retrieval	Modification	Deletion
1	0.28	0.74	0.52
10	0.69	6.11	3.94
100	5.34	59.85	38.14
800	32.34	477.78	304.09
1000	37.95	597.19	380.08
1600	51.34	955.42	608.04
2400	63.87	1433.06	912.00
3200	72.76	1910.69	1215.95
4000	79.39	2388.33	1519.90

Number of Records	Insertion Minimum	Insertion Average	Insertion Maximum
1	0.46	0.91	293.71
10	4.56	9.11	2937.06
100	45.59	91.07	29370.62
800	364.74	728.53	234964.90
1000	455.93	910.67	293706.10
1600	729.48	1457.07	469929.90
2400	1094.23	2185.60	704895.00
3200	1458.97	2914.14	939860.00
4000	1823.71	3642.67	1174824.00

Number of Controllers = 2

Number of Records	Retrieval	Modification	Deletion
1	0.56	1.47	1.04
10	1.37	12.21	7.87
100	10.24	119.54	76.17
800	51.39	954.32	607.39
1000	58.23	1192.82	759.17
1600	72.79	1908.35	1214.51
2400	84.55	2862.38	1821.62
3200	91.98	3816.42	2428.73
4000	97.11	4770.45	3035.84

Number of Records	Insertion Minimum	Insertion Average	Insertion Maximum
1	0.91	1.82	319.39
10	9.11	18.17	3193.92
100	91.07	181.66	31939.16
800	728.53	1453.29	255513.20
1000	910.67	1816.61	319391.50
1600	1457.07	2906.57	511026.50
2400	2185.60	4359.86	766539.80
3200	2914.14	5813.15	1022053.00
4000	3642.67	7266.43	1277565.00

Number of Controllers = 4

Number of Records	Retrieval	Modification	Deletion
1	1.12	2.94	2.08
10	2.71	24.35	15.70
100	18.94	238.45	151.95
800	72.85	1903.68	1211.63
1000	79.46	2379.46	1514.40
1600	92.01	3806.79	2422.71
2400	100.88	5709.90	3633.78
3200	105.99	7613.02	4844.85
4000	109.31	9516.13	6055.93

Number of Records	Insertion Minimum	Insertion Average	Insertion Maximum
1	1.82	3.61	334.00
10	18.17	36.14	3339.96
100	181.66	361.45	33399.59
800	1453.29	2891.57	267196.70
1000	1816.61	3614.46	333995.90
1600	2906.57	5783.13	534393.50
2400	4359.86	8674.70	801590.30
3200	5813.14	11566.26	1068787.00
4000	7266.43	14457.82	1335982.00

Number of Controllers = 8

Number of Records	Retrieval	Modification	Deletion
1	2.21	5.85	4.13
10	5.30	48.45	31.24
100	32.92	474.44	302.32
800	92.07	3787.69	2410.76
1000	97.18	4734.34	3012.17
1600	106.01	7574.27	4820.39
2400	111.66	11360.84	7230.04
3200	114.72	15147.42	9639.67
4000	116.64	18933.99	12049.31

Number of Records	Insertion Minimum	Insertion Average	Insertion Maximum
1	3.61	7.16	341.81
10	36.14	71.55	3418.11
100	361.45	715.50	34181.07
800	2891.57	5724.02	273448.50
1000	3614.46	7155.30	341810.60
1600	5783.13	11448.04	546897.00
2400	8674.70	17172.07	820345.70
3200	11566.26	22896.09	1093794.00
4000	14457.82	28620.09	1367241.00

Number of Controllers = 16

Number of Records	Retrieval	Modification	Deletion
1	4.33	11.58	8.18
10	10.19	95.91	61.84
100	52.17	939.18	598.47
800	106.07	7497.96	4772.23
1000	109.37	9371.89	5964.74
1600	114.74	14993.70	9542.25
2400	117.97	22489.45	14312.27
3200	119.65	29985.19	19082.29
4000	120.68	37480.93	23852.31

Number of Records	Insertion Minimum	Insertion Average	Insertion Maximum
1	7.16	14.02	345.86
10	71.55	140.23	3458.57
100	715.50	1402.34	34585.68
800	5724.02	11218.70	276685.40
1000	7155.03	14023.38	345856.80
1600	11448.04	22437.41	553370.80
2400	17172.07	33656.11	830056.40
3200	22896.09	44874.82	1106741.00
4000	28620.00	56093.48	1383426.00

Number of Controllers = 32

Number of Records	Retrieval	Modification	Deletion
1	8.32	22.70	16.03
10	18.90	187.98	121.20
100	73.71	1840.74	1172.96
800	114.79	14695.50	9353.26
1000	116.69	18368.29	11690.49
1600	119.67	29386.67	18702.18
2400	121.39	44077.83	28051.11
3200	122.27	58768.98	37400.03
4000	122.81	73460.13	46748.95

Number of Records	Insertion Minimum	Insertion Average	Insertion Maximum
1	14.02	26.97	347.92
10	140.23	269.66	3479.16
100	1402.34	2696.63	34791.60
800	11218.70	21573.03	278332.80
1000	14023.38	26966.29	347916.00
1600	22437.41	43146.07	556665.60
2400	33656.11	64719.11	834998.50
3200	44874.82	86292.13	1113331.00
4000	56093.48	107865.00	1391663.00

Number of Controllers = 64

Number of Records	Retrieval	Modification	Deletion
1	15.41	43.66	30.82
10	33.00	361.48	233.07
100	92.90	3539.65	2255.54
800	119.71	28258.75	17985.87
1000	120.73	35321.35	22480.25
1600	122.29	56509.15	35963.40
2400	123.18	84759.50	53940.93
3200	123.63	113009.90	71918.44
4000	123.90	141260.30	89895.94

Number of Records	Insertion Minimum	Insertion Average	Insertion Maximum
1	26.97	50.07	348.95
10	269.66	500.75	3489.55
100	2696.63	5007.45	34895.48
800	21573.03	40059.61	279163.80
1000	26966.29	50074.51	348954.80
1600	43146.05	80119.19	558327.70
2400	64719.09	120178.80	837491.70
3200	86292.06	160238.40	1116655.00
4000	107865.00	200297.80	1395818.00

Number of Controllers = 128

PERFORMANCE CHARACTERISTICS OF DATA BASE MACHINES

by

RANDALL LEE MEHARG

B.S., Kansas State University, 1976

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
1980

ABSTRACT

The performance characteristics of a data base machine (DBM) based on relational associative processing (RAP) are described herein. The specialized hardware functions and design characteristics of the DBM are provided. A model is presented which facilitates the comparison of functions common to both the DBM and Cullinane Corporations Integrated Data Management System (IDMS). The utilization of this model under varying environments indicates that reduced software overhead, concurrent access operations and elimination of data base control structures allow the DBM to operate at speeds approaching six orders of magnitude faster than IDMS. A performance coefficient (PC) formula permits the application of the models results to a wide variety of IDMS supported installations.