IN-60-CR
210937
38P.

ICASE REPORT

EXPERIMENTS CONCERNING BUFFER MANAGEMENT IN A VIRTUAL MEMORY SYSTEM

Stephen W. Sherman

Richard S. Brice

Report Number 75-20

October 6, 1975

INSTITUTE FOR COMPUTER APPLICATIONS

IN SCIENCE AND ENGINEERING

Operated by the

UNIVERSITIES SPACE RESEARCH ASSOCIATION

at

NASA'S LANGLEY RESEARCH CENTER

Hampton, Virginia

(NASA-CR-180233) EXPERIMENTS CONCERNING BUFFER MANAGEMENT IN A VIRTUAL MEMORY SYSTEM (ICASE) 38 p

N89-70666

Unclas

00/60 0210937

Experiments Concerning Buffer Management in a Virtual Memory System

Stephen W. Sherman*

Institute for Computer Applications in Science and Engineering

Richard S. Brice

Department of Civil, Mechanical and Environmental Engineering
George Washington University

Buffer pools are created and managed in data base systems in order to reduce the total amount of accesses to the I/O devices. The useof the buffer pool in a virtual storage system may cause an increase in the page fault rate. We examine this phenomenon through the analysis of empirical data gathered in a multifactor experiment. The factors we considered were memory size, buffer size, replacement algorithm for memory and buffer management algorithm.

*On leave from the University of Houston

This paper was prepared as a result of work performed under NASA Grant NGR 47-102-001 while the first author was in residence at ICASE, NASA Langley Research Center, Hampton, VA 23665. The work of the second author was supported by NASA Grant NGR-09-010-078.

I. Introduction

Computer programs with a large amount of I/O often employ storage buffers to hold one or more I/O records for future reference. If we assume that the overhead to operate the storage manager is negligible and the buffer area consumes previously unrequired memory, then the use of storage buffers can only improve performance due to the faster access to main memory than to the storage device. In systems where the unrequired memory is virtual memory instead of real memory, an increase in the size of the buffer area may cause a decrease in performance due to a phenomenon known as double paging. The dynamics of double paging was characterized by Goldberg and Hassinger [1] as the running of a paged operating system under a paged virtual machine monitor. We define double paging as the management of buffer storage under the control of a paged virtual memory environment.

In this paper we present a series of statistically designed and analyzed experiments. The statistical approach has been shown [2] to be effective for analyzing the effects of a number of factors and their interactions. These experiments were constructed to study the effects of four factors upon the cost of running an I/O oriented application program. The factors are storage buffer replacement algorithm, storage buffer size, size of real memory and paging replacement algorithm. Cost is defined as a function of buffer I/O and paging. The application program is a data base management system.

These experiments were conducted on a minicomputer with a virtual memory capability. The use of a minicomputer as a test bed [13] in a dedicated environment significantly reduces the cost of experiments and allows a complete set of experiments to be conducted. The similarity between the more powerful minicomputers and most large scale computers allow these results to retain significance for a broad class of computing systems. In this paper we develop

and empirically verify a simple theoretical model of the behavior of the I/O buffer in a paged environment. We quantify the effects of the factors in this set of experiments on the total I/O in the system and the virtual buffer where total I/O is a function of paging and disk I/O. We demonstrate the existence of situations where an increase in the size of the virtual buffer area causes an increase in performance by decreasing the total I/O. The performance is increased in the virtual buffer and in the system when the relative costs of paging and disk I/O are considered. We also show empirically that the interaction of the paging algorithm and the buffer algorithm does not cause a significant amount of variation in the double paging rate.

II. Previous Studies and Theory

The possibility of double paging in a data base system running on a virtual memory machine was discussed by Tuel [3] using the IMS system running on the IBM Virtual System/2. Tuel postulated a model to explain double paging and compared the model with some experimental data obtained from running IMS Version 2.4 on VS/2 Release 1.6. The agreement between Tuel's model and his experimental results is reasonably close when only the buffer paging is considered [4]. Tuel concludes that optimum performance is achieved by reducing the buffer size to fit the number of page frames available since the total I/O activity increases with increasing buffer size if the buffer is allowed to page.

The early version of IMS used in Tuel's study kept the information about the contents of the individual buffers in those buffers. Therefore, even when the requested data was not in the buffer, the complete buffer had to be searched to determine that fact. We would expect that the number of page faults in a virtual storage system would be excessive when the buffer management strategy

of that early IMS system is used. Even if the requested data was found in the buffer, half of the buffer would have to be read under Tuel's assumption of random assignment. IMS now has a small pointer array that keeps information on the contents of the buffers [5] and our data base system employs the same technique. Searching this pointer array causes much less paging activity than reading the information from the individual buffers.

With a small pointer array to handle the information on the contents of the buffers, the improvement in the buffer handling algorithm might allow other factors to become more significant in determining the effects of double paging. The previous analysis by Goldberg and Hassinger suggested that the interaction between the page replacement and buffer management algorithms could cause a variation in total paging activity in the buffer.

If we examine just the paging in the data base buffer under some simple assumptions, then we can construct a counterexample to Tuel's results which showed the I/O activity increases with increasing buffer size if the buffer is allowed to page. Although these assumptions are quite simple, they are very similar to the assumptions that Tuel made. The primary difference is that we assume the search of the information pointer array generates no page faults. This might be reasonable if the array is sufficiently small and searched sufficiently often using some type of stacked paging algorithm.

Using the same notation as Tuel, let N be the buffer pages available to hold the data base. Let M be the pages of real storage available for the buffer. Let D be the pages in the data base. All information is read from the data base and stored in pages. For uniformly distributed data base requests, the probability d that a requested data base page is in the buffer is

$$d = 1 \quad \text{for} \quad D < N$$

and

$$d = N/D \quad \text{for} \quad D \geq N.$$

For a random page replacement algorithm, the probability n that a buffer page is in real storage is

$$n = 1 \quad \text{for} \quad N < M$$

and

$$n = M/N \quad \text{for} \quad N \geq M$$

We are assuming that data base pages are assigned randomly to the buffer and buffer pages are assigned randomly to main storage.

The total I/O activity for each data base request is a function of the number of page faults and the number of data base accesses. Consider the number of page faults for each data base request. The probability of a page fault if a request is found in the buffer is $d\,(1-n)$ which is the probability that a requested data base page is in the buffer and the probability that the buffer is not in real memory. If a request is not found in the buffer, the probability of page fault is $(1-d)(1-n)$ which is the probabiltiy that the requested data base page is not found in the buffer times the probability that the buffer selected to read the page into is not in real memory. Therefore, the probability of a page fault for each request is $d(1-n) + (1-d)\,(1-n) = 1-n$.

Consider the probability of a data base access for each request. An access is only made if the data base request is not in the buffer. The probability that it is not in the buffer is $1-d$.

Let us define the total I/O cost (T) per data base request as the expected number of page faults plus the expected number of data base accesses.

Therefore

$$T(N,M,D) = (1-n) \text{ faults} + (1-d) \text{ accesses}$$

$$= (1-\frac{M}{N}) \text{ faults} + (1-\frac{N}{D}) \text{ accesses for } 1 \leq M \leq N \leq D.$$

Let us further assume one fault equals X accesses where $0 < X < \infty$. Then $T(N,M,D) = X - \frac{XM}{N} + 1 - \frac{N}{D}$. If we hold M and D fixed and allow N to increase the change in total I/O cost is

$$\Delta T(N) = T(N+1) - T(N) = \frac{XM}{N^2+N} - \frac{1}{D}$$

If $D \left(\overset{=}{\gtrless}\right) \frac{N^2+N}{XM}$ then $\Delta T(N) \left(\overset{=}{\gtrless}\right) 0$ and $\Delta T(N) \left(\overset{=}{\gtrless}\right) 0$ = implies that $T(N)$ is a $\begin{pmatrix} \text{constant} \\ \text{increasing} \\ \text{decreasing} \end{pmatrix}$ function of buffer size.

In general the size of D is relatively fixed. If D is very large, then $T(N)$ is an increasing function of buffer size. This implies that there may be an advantage in extracting a sub data base [6]. In large data bases, resource requirements for data base requests usually do not include the complete data base. They can be considered in terms of the working set [7] of data base records that they invoke. This reduces the actual size of D and could allow $T(N)$ to be a decreasing function of buffer size. As an example of increasing N and reducing the total I/O, let $X = 1$, $D = 15$, $N = 6$, $M = 2$. Then $T(N,M,D) = \frac{19}{15}$ accesses. If N is increased to 10 then $T(N,M,D) = \frac{17}{15}$ accesses per data base request and all of these values satisfy the inequality $1 \leq M \leq N \leq D$. Figure I shows how the cost of total I/O can change for a range of N given M and D. Further, the minimum cost of total I/O could be shifted to either end of the range depending on the value of X.

With this simple model, we have shown that there is a possibility that different combinations of values for M and N can yield a reduction in total I/O for given D and X. Some of the assumptions in our model are overly

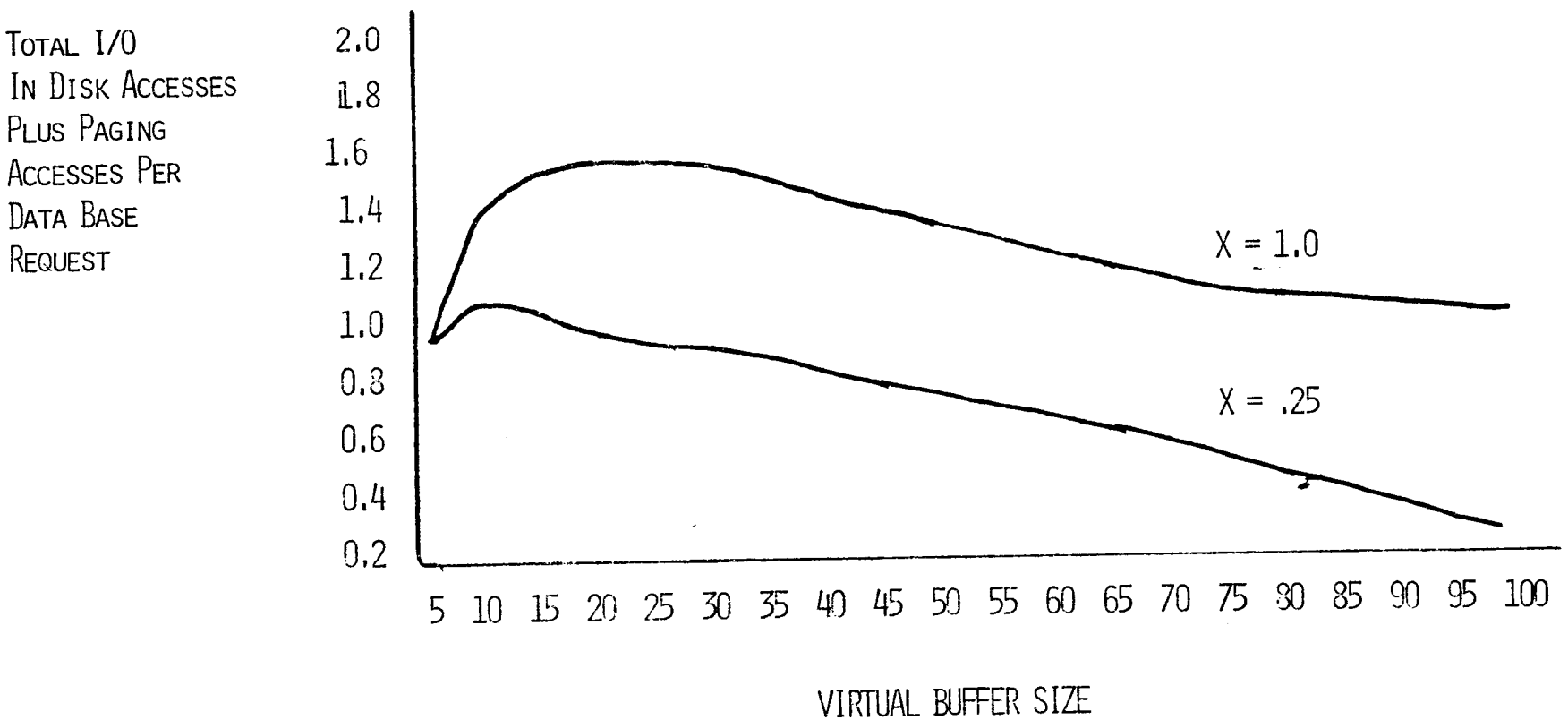


FIGURE 1

TOTAL I/O FOR THE BUFFER WHERE: D = 100 , M = 5 , X = (1.0,.25)
N RANGING FROM 5 TO 100

simplistic and ignore the effect of the operating system, the data base program, the page management strategy of the program, the buffer management strategy of the system and the distribution of data base requests. We will examine the effects of some of these factors in our empirical study.

III. Experiments

There are several techniques that might be used in a controlled environment to examine the effects of double paging. The user space could be split between user program and buffer area. Each space might then be managed by an appropriate paging algorithm in the operating system. Further, the split could be a fixed partition or a dynamic partition. If a dynamic partition is used, then the paging algorithm might employ a working set strategy. If a fixed partition is used then the virtual buffer size should be at least as large as the partition. The current IMS strategy [8] for improving performance is to permit the user to lock some number of IMS programs and buffer pages into core. The IMS strategy is similar to the fixed partition with only one paging algorithm. Due to the poor performance that usually accompanies fixed partition strategies, our initial investigations focus on dynamic partition strategies.

In this initial study, we investigate the performance of global paging algorithms on the user program, user buffer area and that part of the operating system that is not resident in dedicated memory. The page replacement algorithms we examined are first in - first out (FIFO), random (RAND), second chance (SCH), and least recently used (LRU). These techniques are all well known with the possible exception of the second chance algorithm. The second chance algorithm which was used on the multics system [9] combines the simplicity of FIFO with the predictive power of LRU. The FIFO replacement algorithm can be

implemented by a circular pointer to the table of pages in real memory. As each page being pointed to is replaced, the pointer is moved to point to the next page. The second chance algorithm requires the addition of a bit associated with each page in real memory that is set whenever there is a reference to a particular page. Instead of automatically replacing the page indicated by the circular pointer, the bit associated with that page is inspected. If the bit is not set, the page is replaced and the pointer is moved ahead. If the bit is set, the bit is reset and the pointer is moved ahead repeating the process until a page can be replaced.

The record replacement strategies used for the buffer managers consist of the same algorithms used for the paging. The buffer sizes in records were 1, 5, 10, 15, and 20. The total amount of real memory available to the system and users ranged from 36K to 48K in increments of 4K. The range of buffer sizes and main memory sizes were chosen after examination of some preliminary experiments.

The four experimental factors, virtual buffer size, real memory size, buffer management algorithm and page replacement algorithm have five, four, four, and four levels of interest respectively. There are 320 ($5 \cdot 4^3$) possible combinations of one virtual buffer size, one real memory size, one buffer management algorithm and one page replacement algorithm. The experiment run was a complete 5 by 4 by 4 by 4 factorial experiment consisting of all 320 combinations. All of the experiments were run on a dedicated system.

IV. Environment

The experiments described in this paper were conducted on a PRIME 300 minicomputer. An illustration of our configuration is presented in Figure II.

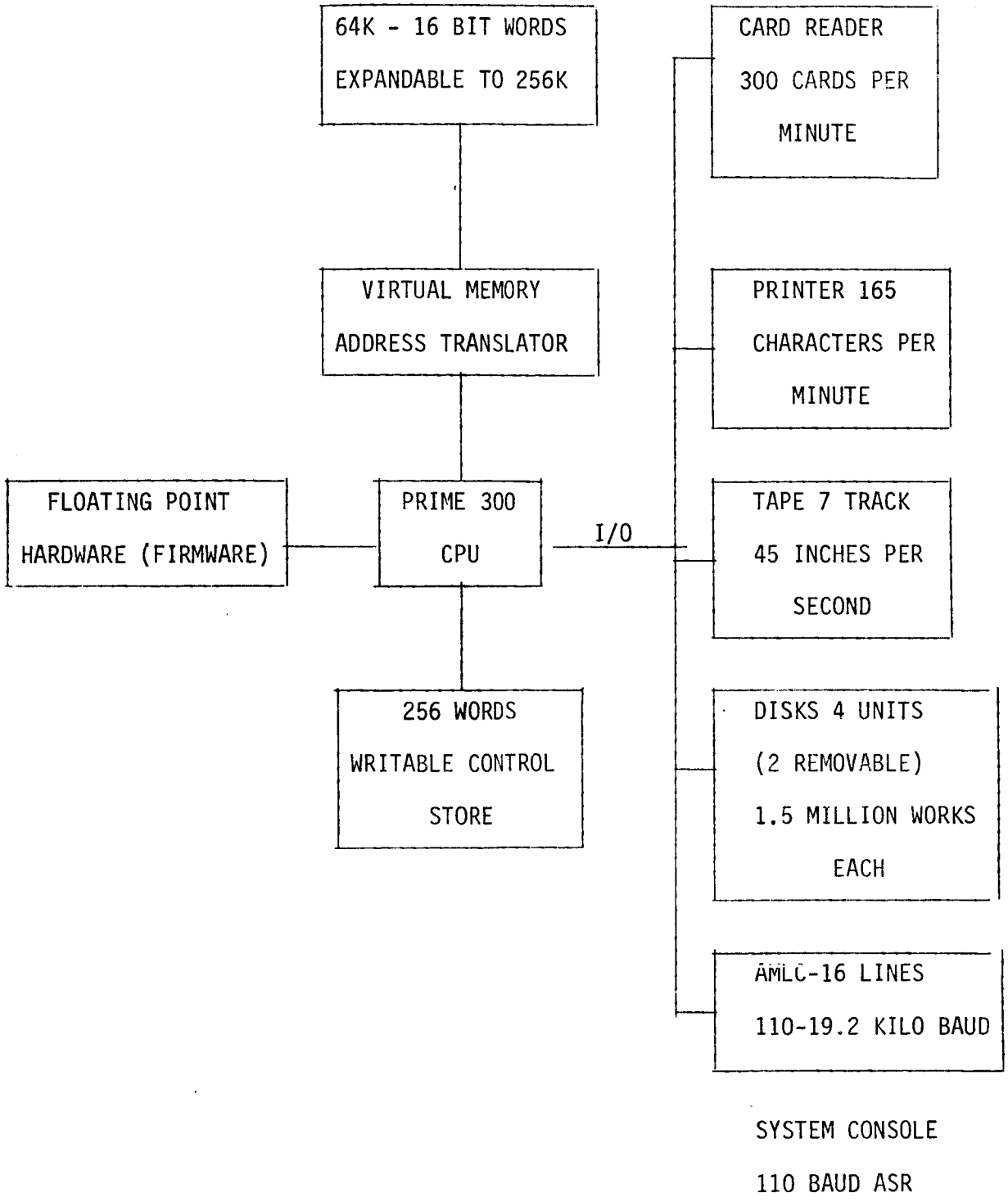


FIGURE II

The minicomputer comfiguration

The PRIME 300 has a 16 bit word size and can support up to 256,000 words. Our system has 64,000 words. The PRIME has a virtual memory hardware capability that maps virtual addresses into a real address space of up to 512 pages of length 512 words each. The PRIME 300 peripherals of interest to this experiment consisted of 2 moving head disks which are each divided into 1.5 million words of permanent storage and 1.5 million words of removable storage. In speed and power the PRIME 300 could be reasonably compared to a Digital Equipment PDP 11/70.

The multi-user disk operating system for the PRIME [10] makes use of the virtual memory capability to multiprogram up to 16 users in an interactive environment. Each user is given a virtual address space of 64K words. In our system, the pages are swapped to the permanent storage areas of one of the disks. The operating system is almost 32K words in size and approximately 24K is locked into central memory and unavailable for paging.

We have instrumented the operating system with a software probe. The probe reserves a buffer in which events of interests are recorded. These are the events that cause a significant change in the state of the system. Examples of these events are: 1) a user issues an SVC (service request) call to the system; 2) a request is made to a peripheral device, or; 3) a page fault has occurred. Information about the event, the time, and the user that was in control when the event occurred are also recorded with the event. When the buffer is sufficiently full, it is dumped to tape for later analysis. Since the probe operates within pages locked in core, no extra paging is caused by its operation and it slows the system down by less than 10%.

The application we have chosen to instrument is a prototype data base management system (DBM) which was written at the Institute for Computer Applications in Science and Engineering by D. Loendorf and S. Sherman. The DBM

organizes data in a tree structured format. There is no formal query language at this time and requests are made in a series of interactive primitive functions that perform elementary operations upon the data base. Examples of primitive functions are: select data base, build node of tree, move up the tree, move down the tree to a particular node, insert data at a node, delete data from a node. The data, names, and pointers are encoded in 40 word segments. One physical record on disk consists of eleven 40 word segments (440 words in 1 record) and information is read from the disk in physical records. All I/O requests from the DBM to the disk are direct access requests. The DBM program is written in FORTRAN and consumes almost 29K of memory. The DBMS buffer area can consume up to an additional 32K words of memory. The virtual buffer size is fixed when the DBM system is initially loaded.

The data base used in this experiment is an engineering data base generated from the input/output of SNAP [11], a finite element analysis program. The tree structure is 7 levels deep and contains 40 nodes. The size of the data base is 45 records. Most of the data consists of integer and floating point numbers describing the geometry of an aircraft structure.

All experiments are controlled by a script of data base requests. The script is contained in and executed from a disk file. The script completely traverses the data base and causes reads, inserts and deletes of data as it is executed. The script sequence moves through the data as an engineer might if he were interacting with graphical displays produced by a program running intermittently with the DBM. The engineer might display the complete aircraft structure, zoom in on a particular part, change some values in that part, re-examine the changed part, examine another part, etc.

V. Results

The appendix and Table II contain all the information required to derive the figures, numbers, and tables used to interpret the factorial experiment. The LRU paging algorithm was omitted from the appendix and from the analysis of results presented here due to the similarity between the results of the LRU and SCH paging algorithms. In general the LRU paging algorithm was a slight improvement over the SCH paging algorithm in terms of total paging. In order to omit unnecessary congestion, we usually present figures that are representative rather than exhaustive in terms of the number of levels of factors available. These figures have been presented to show trends in the data and the appendix and Table II contain all the data necessary to construct an exhaustive set of figures. The model presented in the theoretical section only considered the buffer paging and I/O requests. Our paging algorithms were global and did not distinguish missing buffer faults from program faults where the program includes both the user program and the operating system. The data gathered by the software probe allows us to separate paging and I/O in the buffer from other paging and I/O. Pages were free to migrate between buffer space and program space. In every test the migration of pages from the buffer space to the program space was approximately equal to the migration in the reverse direction. The number of pages that migrated was an increasing function of virtual buffer size and a decreasing function of real memory size. The interference of the program paging on the buffer was balanced by the interference of the buffer paging on the program if the amount of page migration is used is a measure.

The model allocated a fixed amount of real memory M to the virtual buffer. The paging algorithms we used permitted the program and buffer to compete for real memory pages. Our analysis produced the average quantity of

real pages allocated to the buffer. The averages were an increasing function of virtual buffer size. The analysis of variance of the average buffer sizes presented in Table I shows that virtual buffer size causes over 90% of the variability.

The disk I/O in the buffer is dependent only on buffer size and buffer management algorithm. Table II presents the number of disk accesses for all combinations of buffer manager and buffer size. The optimal column represents a simulation of Belady's [12] optimal algorithm using the data base reference string as input. The random buffer management algorithm was also simulated with various seeds in the random generator to verify reproducibility. The random algorithm was significantly better than the others at virtual buffer sizes of 10 and 15. All of the others exhibited similar performance and were noticeably better than the random algorithm at the virtual buffer size of 20. Our preliminary experiments showed that the relatively poor performance of the random algorithm at the largest buffer size would have continued if the virtual buffer size were expanded. An analysis of the reference string shows that there were a few records used very frequently as shown in the histogram in Figure III. The references to these records were usually separated by a string of references to other records. Figure IV shows that the lengths of these strings was typically between 10 and 15. While the random algorithm might allow the frequently referenced records to remain in the buffer, the other buffer management algorithms were often forced to throw out the frequently referenced records.

Any record in the data base can contain both data and structural information. The frequently referenced records contained information describing the organization of that portion of the tree structured data base most often required. The strings of intervening record references were the result of data transfers once the desired node in the tree had been reached.

SOURCE OF VARIATION	DEGREE OF FREEDOM	SUM OF SQUARES	MEAN SQUARES
Total	239	3364.2	14.08
Main Effects	12	3209.7	267.48
V	4	3042.8	760.69
R	3	157.8	52.59
B	3	8.3	2.78
P	2	.8	.42
1st order interactions	53	146.0	2.75
VR	12	122.0	10.16
VB	12	17.0	1.42
VP	8	2.5	.31
RB	9	1.0	.12
RP	6	3.2	.54
2nd order interactions	102	7.1	.07
VRB	36	1.6	.04
VRP	24	4.3	.18
3rd order interactions	72	1.4	.2
VRBP	72	1.4	.2

V = Virtual buffer size
R = Real memory in the system
B = Buffer Manager
P = Paging Algorithm

TABLE I

Analysis of variance of average virtual buffer size in pages of real memory

(Any interactions with a sum of squares < 1 are not presented)

Virtual Buffer Sizes in Records	Buffer Managers				
	LRU	SCH	RAND	FIFO	Belady Optimal
1	1075	1075	1075	1075	1075
5	784	791	774	794	551
10	684	687	559	688	311
15	418	422	270	434	138
20	93	98	154	103	82

TABLE II

Number of I/O requests to read records into the virtual buffer

```
Number of
References

161     ****************************************
119     ******************************
 61     ***************
 52     *************
 50     ************
 45     ***********
 43     **********
 32     ********
 30     *******
 30     *******
 29     *******
 29     *******
 29     *******
 29     *******
 26     ******
 25     ******
 23     *****
 22     *****
 20     *****
 20     *****
 12     ***
 12     ***
 12     ***
 12     ***
 12     ***
 12     ***
 12     ***
 12     ***
 11     **
 11     **
 10     **
  9     **
  8     **
  8     **
  8     **
  4     *
  4     *
  4     *
  4     *
  4     *
  4     *
  4     *
  4     *
  4     *
  4     *
```

FIGURE III

Histogram of Record References

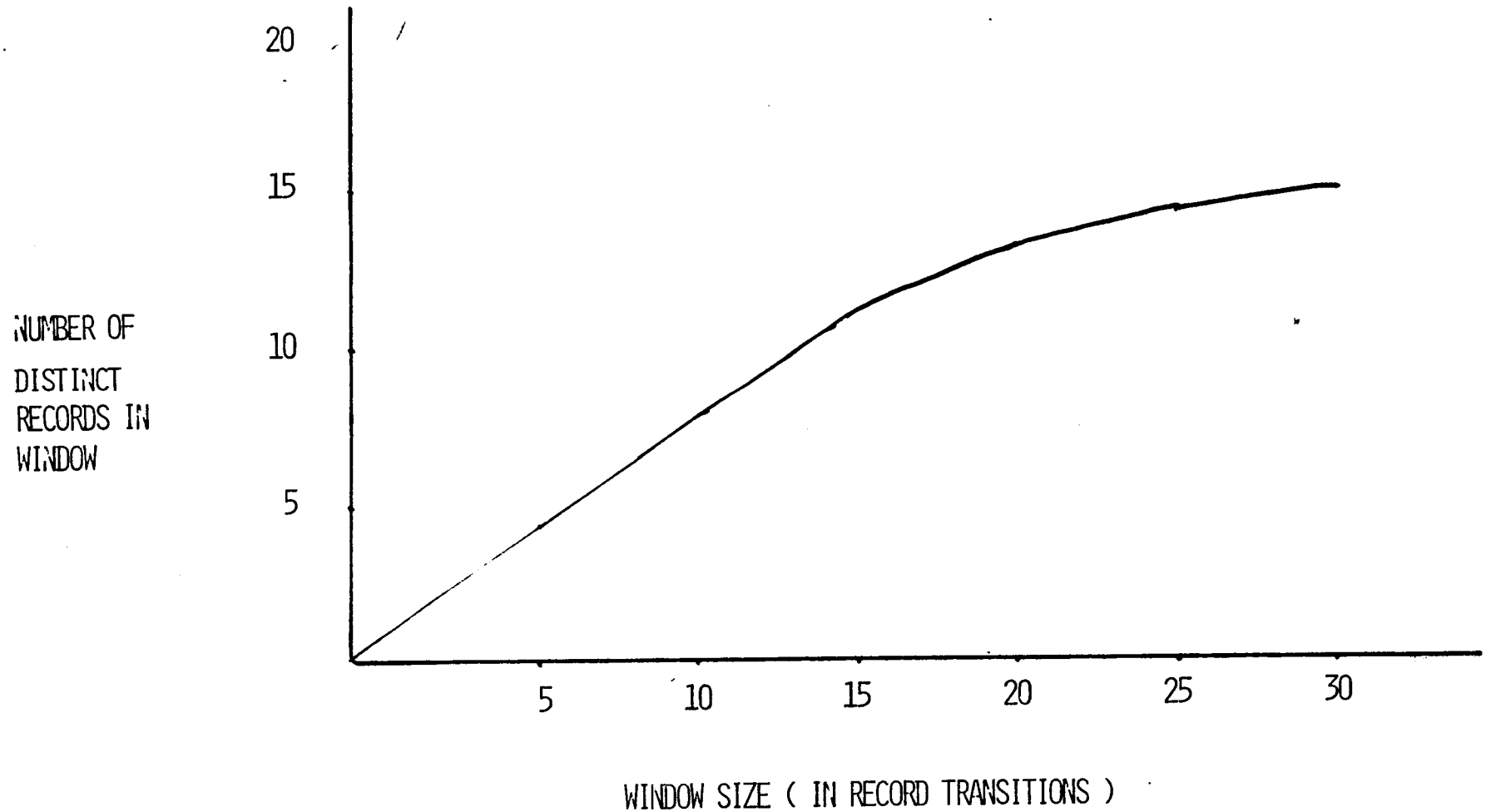


FIGURE IV

AVERAGE NUMBER OF DISTINCT RECORDS IN A WINDOW LENGTH N .

In addition to the cost of I/O in the buffer, use of data in the buffer incurs a paging cost. We divided buffer paging into paging caused by the disk I/O manager and paging caused by attempts to reference the data in the buffer. The paging in the buffer induced by the disk I/O manager represents a form of double paging. The double paging rate presented in Figure V has been calculated by dividing the number of double page turns by the number of I/O requests to the disk. The normalized double paging rate is an increasing function of buffer size and a decreasing function of real memory size. The choice of buffer management algorithm appears to have little effect. Most of the time the FIFO buffer manager performs slightly better than the rest. The results obtained in [1] suggest that the combinations of SCH paging and SCH buffer algorithm should create a higher double paging rate than, for example, SCH paging combined with the RAND buffer algorithm. We suspect that the lack of influence of the buffer manager on the variability of the double paging rate in these experiments is due to the paging interference between the program space and the buffer space. Further experiments are being conducted to test this hypothesis.

The reference paging rate, paging caused by attempts to reference the data in the buffer, was also an increasing function of buffer size and a decreasing function of memory size. The reference paging rate presented in Figure VI has been computed by dividing the number of reference page turns by the number of record transitions minus the number of I/O requests to the disk. Each record transition causes an I/O request to the disk in the single buffer case. There were 1075 transitions in every experiment.

The total paging in the buffer (reference paging + double paging) appears to be unaffected by the buffer algorithm in most cases. The random buffer algorithm is noticeably worse than the others when the buffer size is large.

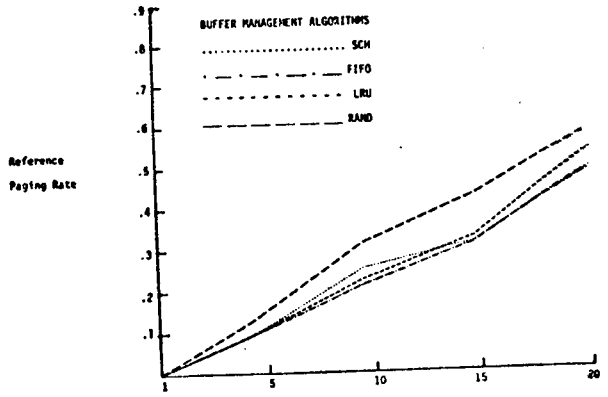


Figure VI (Graph 1)

Reference paging rate for SCH paging algorithm with 40K of real memory

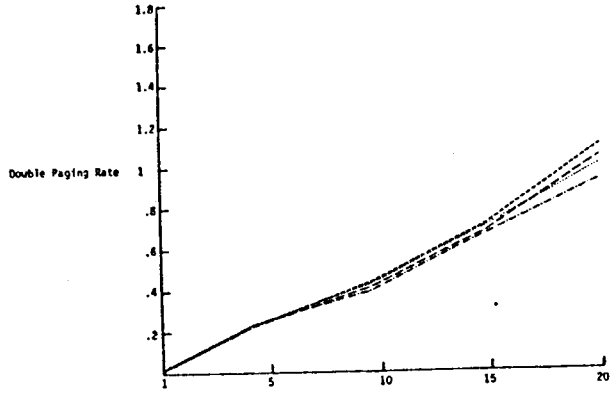


Figure V (Graph 3)

Double paging rate for SCH paging algorithm with 40K of real memory

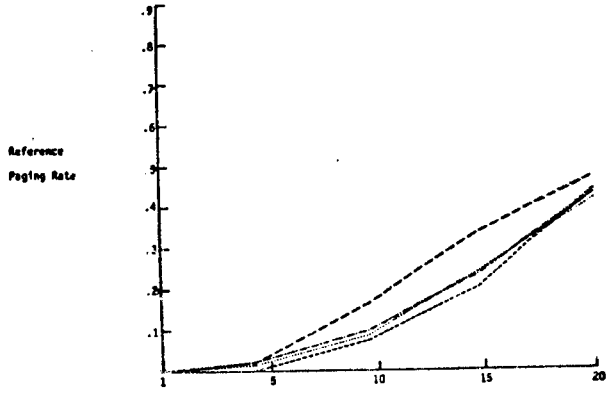


Figure VI (Graph 2)

Reference paging rate for SCH paging algorithm with 44K of real memory

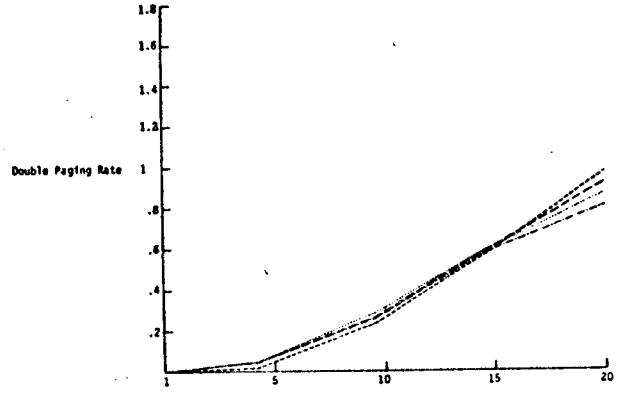


Figure V (Graph 2)

Double paging rate for SCH paging algorithm with 44K of real memory

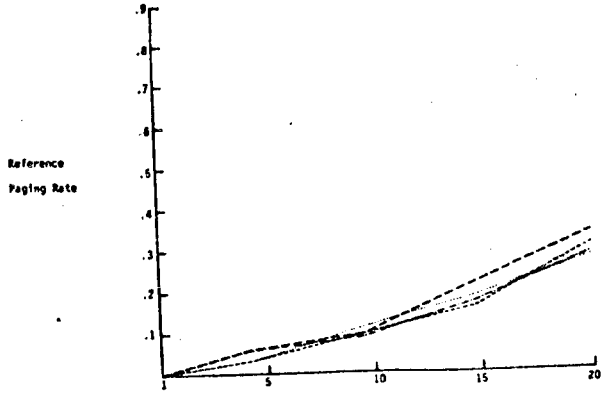


Figure VI (Graph 3)

Reference paging rate for RAND paging algorithm with 44K of real memory

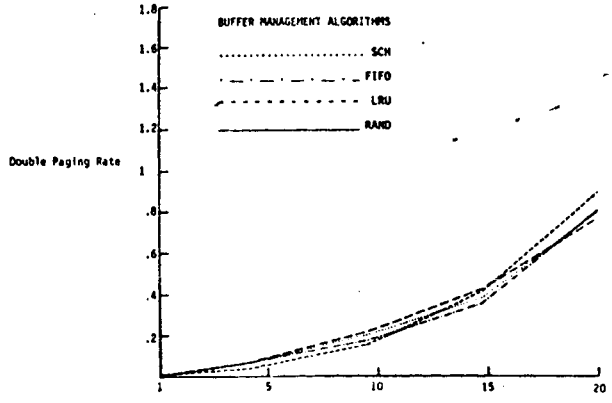


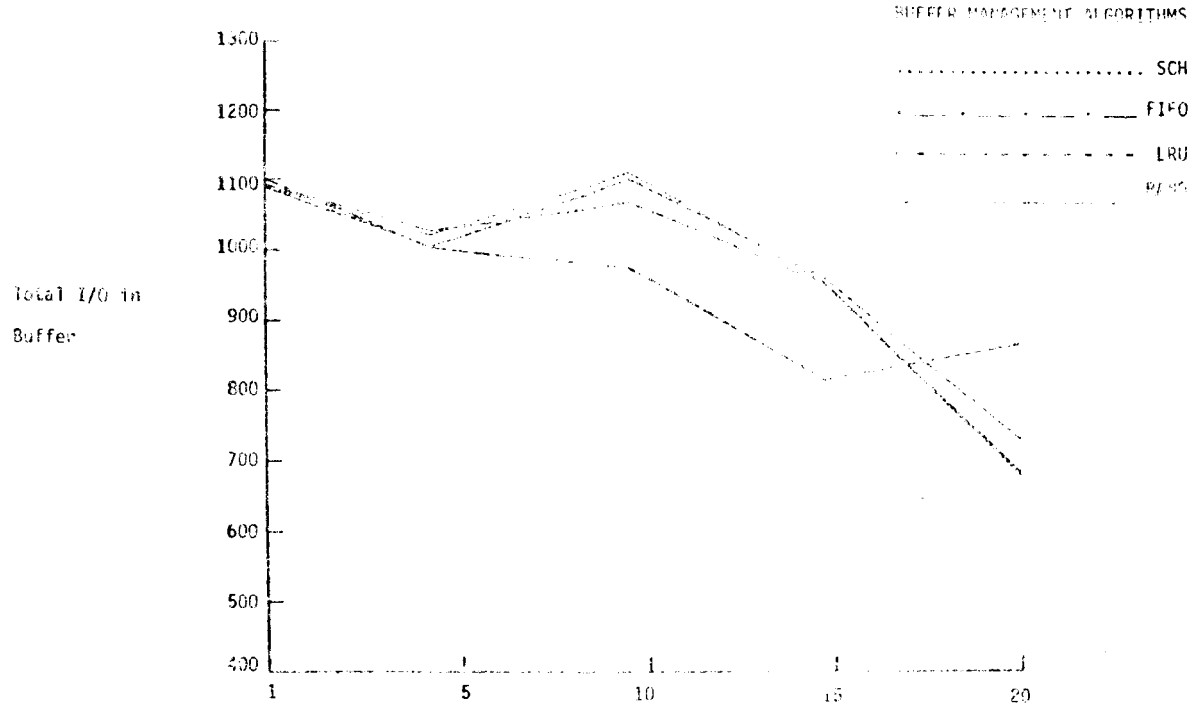
Figure V (Graph 1)

Double paging rate for RAND paging algorithm with 44K of real memory

In general the random algorithm has a lower total number of double page faults and a much higher number of reference page faults.

The total I/O in the buffer consists of the total paging in the buffer plus the requests for I/O in the buffer. A comparison among the buffer managers is shown in the graph of total I/O using SCH paging in Figure VII. The major differences in the total I/O were primarily attributable to the differences in the I/O requests in the buffer. These differences made the random buffer manager significantly better than the others when 10 or 15 buffers were allocated. Figure VIII contains the total I/O in the buffer for all real memory sizes with a combination of random paging and random buffer manager. A range of model results are also presented for comparison.

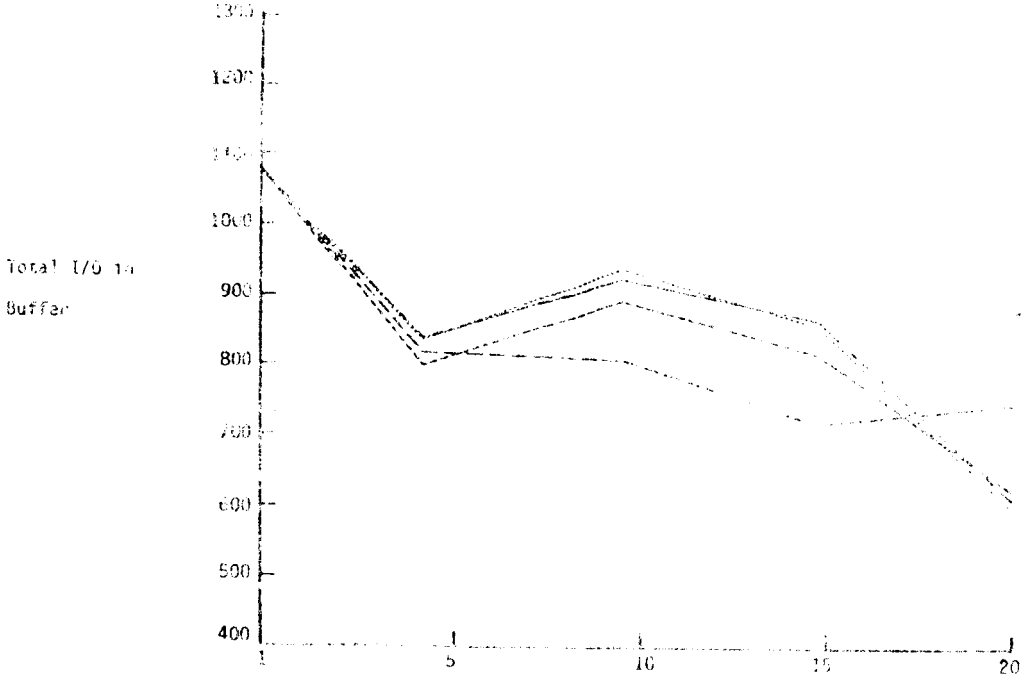
The model values were computed using the average real buffer size for M and the minimum number of virtual pages needed to accommodate the records in the virtual buffer. The values of N were 1, 5, 9, 13, and 18 pages for buffer sizes of 1, 5, 10, 15, and 20 records respectively. In the model, the record references were randomly distributed in D with equal probability. The histogram in Figure III shows that in our test environment, the records were not referenced with equal probability. A more reasonable comparison of the model with the empirical results can be obtained by allowing D to be a smaller subset of the real D containing those records for which the probability of reference was more evenly distributed while still keeping the number of references as large as possible. The value of D must be at least as large as the virtual buffer size for the model results to be meaningful. We have presented in Figure VIII plots of the model results using values of D from 20 to 28. This range of D represents from 80% to 90% of the references to data base records. Figure VIII shows that a trend predicted by this very simple model was observed in the experiments.



Virtual Buffer Size

Figure VII (Graph 1)

Total I/O in buffer for SCH paging algorithm with 40K of real memory



Virtual Buffer Size

Figure VII (Graph 2)

Total I/O in buffer for SCH paging algorithm with 44K of real memory

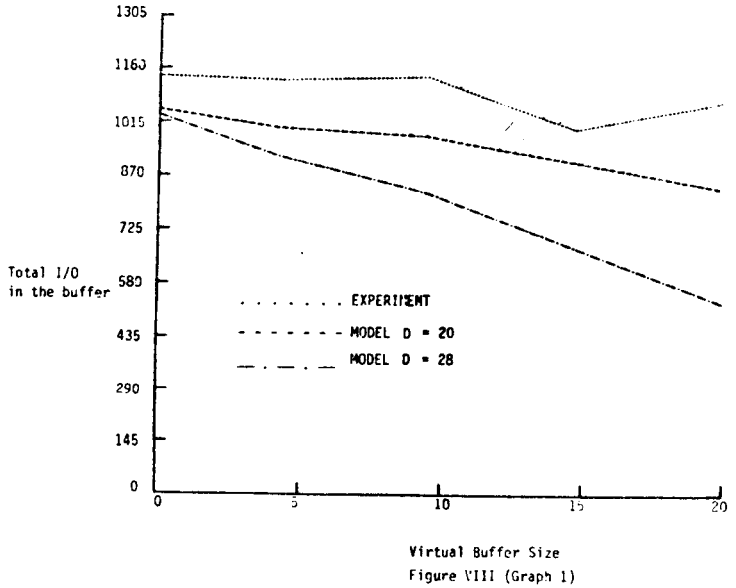


Figure VIII (Graph 1)

A comparison of empirical results using RAND paging algorithm and RAND buffer manager with 36K of real memory and theoretical results with values of D set to 20 and 28. (The model used the average sizes for M)

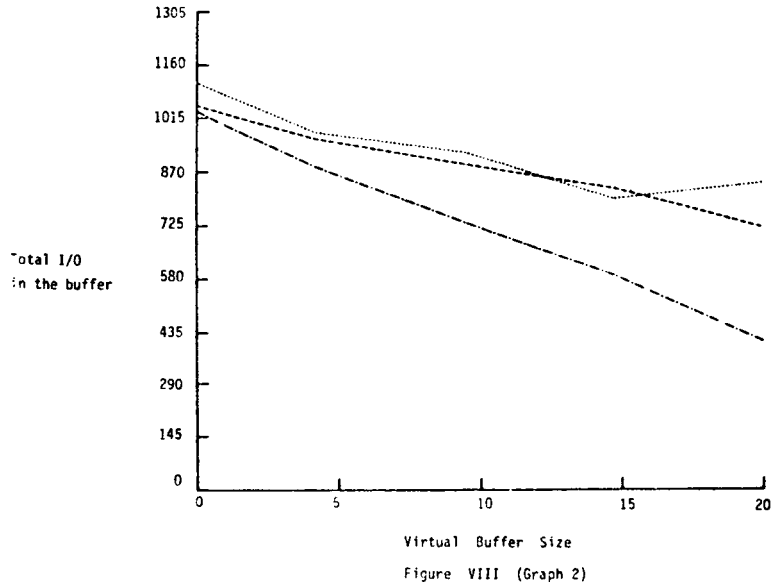


Figure VIII (Graph 2)

A comparison of empirical results using RAND paging algorithm and RAND buffer manager with 40K of real memory and theoretical results with values of D set to 20 and 28. (The model used the average sizes for M)

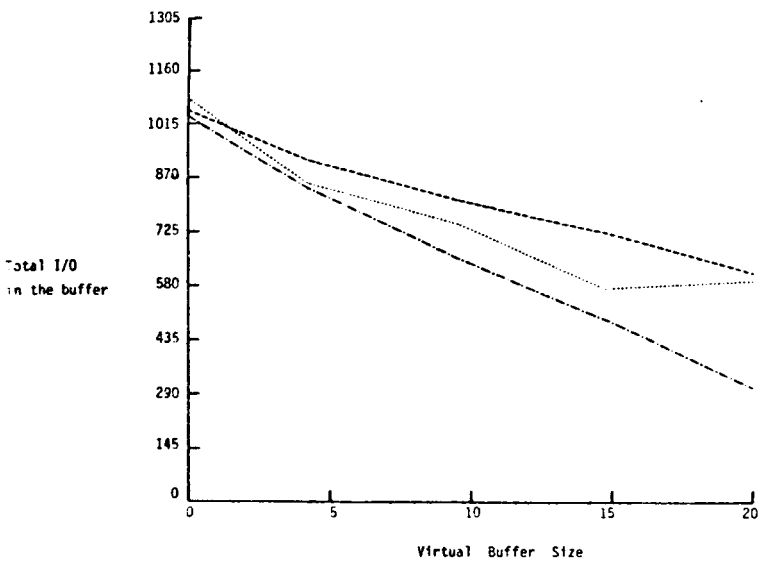


Figure VIII (Graph 3)

A comparision of empirical results using RAND paging algorithm and RAND buffer manager with 44K of real memory and theoretical results with values of D set to 20 and 28. (The model used the average sizes for M)

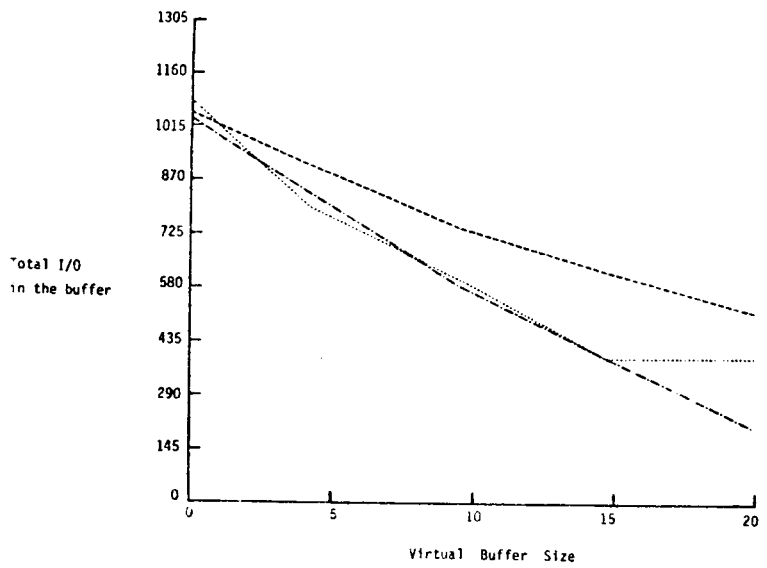


Figure VIII (Graph 4)

A comparison of empirical results using RAND paging algorithm and RAND buffer manager with 48K of real memory and theoretical results with values of D set to 20 and 28. (The model used the average sizes for M)

The total I/O in the system that is affected by the variations of the factors in these experiments is composed of disk I/O in the buffer, paging in the buffer and paging in the program. Other disk I/O in the program is not affected by any variations of factors in these experiments. The empirical and theoretical data presented in Figure VIII showed that total I/O in the buffer can decrease when virtual buffer size is increased. The increase in virtual buffer size was accompanied by an increase in real buffer memory size and a corresponding decrease in real memory available to the program. The increased program paging caused by the decrease in real memory available to the program could cause the total I/O in the system to increase even if the total I/O in the buffer decreases.

The paging in the buffer and the paging in the program can be combined into total system paging. The increase in total system paging is illustrated in Figure IX for the second chance paging algorithm. The increase in total paging displayed similar characteristics using the FIFO and RAND paging algorithms. The variation in total system paging was most significantly affected by memory size, virtual buffer and paging algorithm in that order. These results are similar to other results presented in the literature [2] in that real memory size dominated the effects of the other factors. An analysis of variance of the total system paging appears in Table III. The virtual buffer size made the most difference when the real memory size was 44K. It appears that at 44K of real memory the increase in virtual memory size is affecting the ability of the program to keep its working set in real memory.

The total cost of accesses to secondary memory while executing the script is a function of total paging and the number of I/O requests to the disk. The disk I/O resulting from the data base manager accessing data outside the data base was a small fixed quantity throughout all the experiments and was not

SOURCE OF VARIATION	DEGREE OF FREEDOM	SUM OF SQUARES*10^{-5}	MEAN SQUARE*10^{-5}
Total	239	6475.8	27.09
Main Effects	12	6199.2	516.60
V	4	716.4	179.10
R	3	5270.2	1756.70
B	3	16.0	5.33
P	2	196.6	98.30
1st order interactions	53	239.9	4.53
VR	12	115.9	9.66
RP	6	104.0	17.34
2nd order interactions	102	33.42	.33
VRP	24	28.7	1.20
3rd order interactions	72	3.25	.04

V = Virtual buffer size
R = Real memory in the system
B = Buffer Manager
P = Paging Algorithm

TABLE III

Analysis of variance of the total paging in the system

(Any interactions with a sum of squares < 10^6 are not presented)

included in our computation of total cost. One measure of the cost of accessing secondary storage is the time required. These times are affected by many factors such as speed of the device, size of data transfer, length of the queue, and the queuing algorithm. As in the model, we assign a fixed unit cost to a disk reference and let the cost of a page turn be X times the cost of a disk reference. Typically, paging is handled by a faster device than disk references, e. g. fixed head disk versus moving head disk. Even with the other time factors considered, X is usually less than 1. Figure X shows the total costs for the SCH paging algorithm, the SCH buffer manager and a range of real memory sizes. Other combinations of paging algorithms and buffer managers show similar trends. We chose values of 1 and .33 for X to demonstrate the potential variation in cost. With reasonable values of X, the cost of I/O is reduced in our experiments when the virtual buffer size is increased despite the observed increases in total paging. The cost of I/O was reduced the most where memory was either abundant or in very short supply. The cost was reduced the least where an increase in virtual buffer size forced a transition from abundant memory to scarce memory.

In each experiment, an average buffer size was determined. If the average buffer size is a reasonable measure of the relative importance of buffer and program pages, then a partition of memory into program pages and buffer pages should yield better results as the number of pages in the buffer partition approaches the average buffer size. In order to investigate this aspect, we created a system which partitioned the real memory into a buffer area and a program area. These are preliminary experiments and only a small number of tests were made. A more complete set of tests will be presented in a later paper. We fixed the real memory size at 44K, used the SCH paging algorithm, set the buffer sizes at 10 and 15 records, and used the SCH and RAND buffer algorithms.

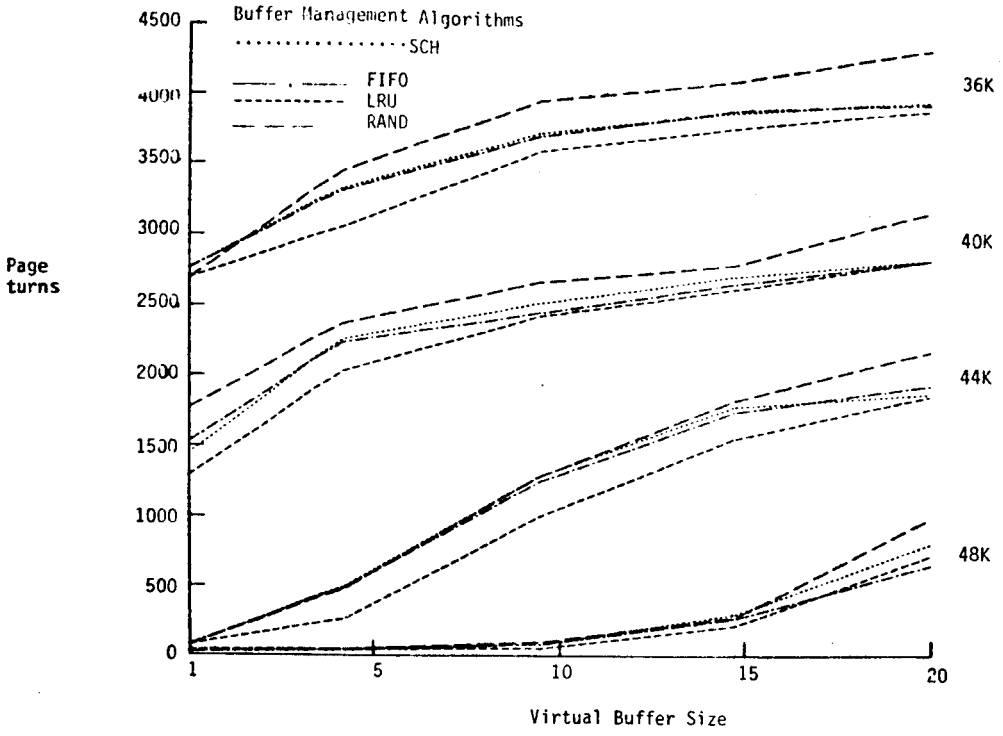


Figure IX

Total paging in the system using SCH paging for all sizes of real memory

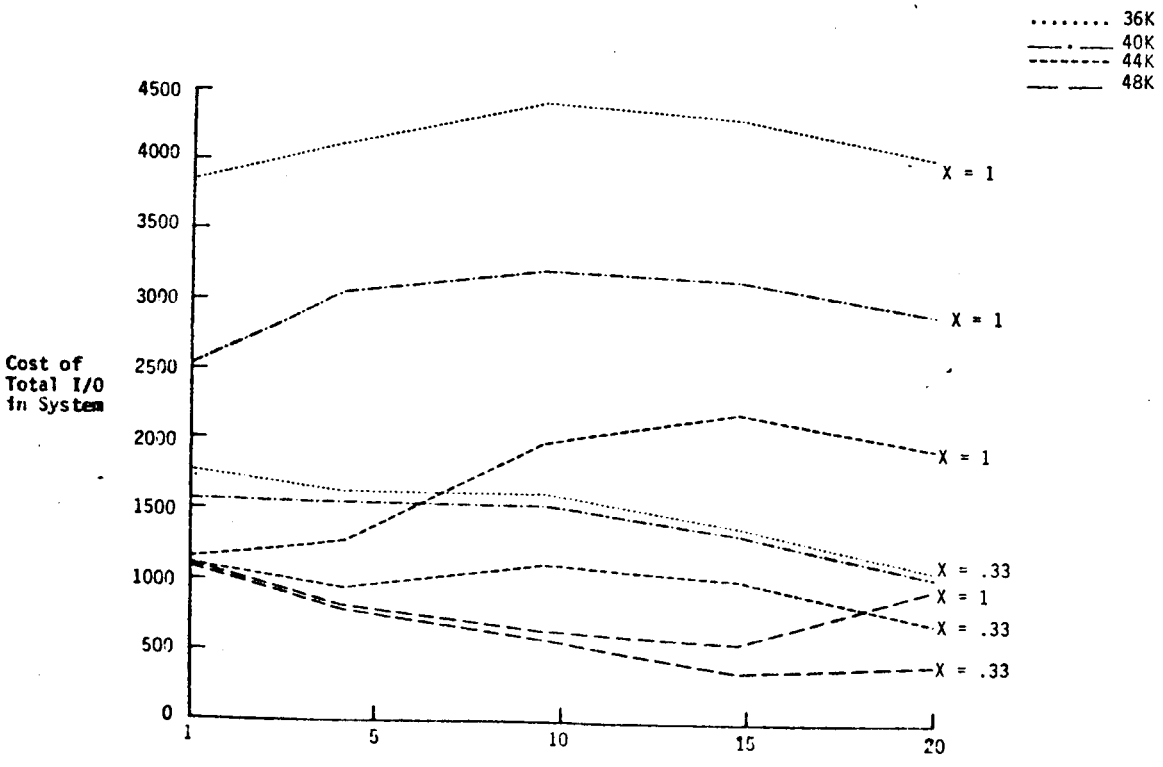


Figure X

Cost of total I/O in the system for the SCH paging algorithm and SCH buffer manager for a range of memory sizes and values of X

The SCH paging algorithm treated each partition separately. The results are presented in Table IV in terms of total I/O in the system and total I/O in the buffer as contrasted with the unpartitioned case. Table IV shows that total I/O in the buffer decreases as the real memory in the buffer increases which the model predicts. The best performance in total I/O for the system was achieved at a partition size significantly smaller than the average buffer size of the corresponding unpartitioned system. These preliminary results suggest that the average buffer size is not a good measure of the relative importance of buffer and program pages.

Our model predicts that increasing the virtual memory in the buffer can lead to a decrease in total I/O in the buffer when real memory in the buffer is held fixed. A set of experiments has been initiated to test this prediction. Again, memory was partitioned into buffer pages and program pages. The RAND paging algorithm and the RAND buffer manager was chosen to correspond with the model. The memory in the buffer was held fixed at 5 pages and 3 pages while the virtual buffer size was allowed to vary from 5 to 20 records. The results presented in Table V include a comparison with the model where the size of the data base ranges from 20 records to 28 records. While the experimental results differ from our simple model noticeably in the 20 buffer case, the empirical results conform to the predicted results in the other cases. We are currently investigating the implications of these results.

VI. Conclusions

A set of experiments has been conducted and analyzed to investigate the effect of several factors on the total I/O of an I/O bound program in a virtual memory system. The real memory used for the I/O buffer and the program was not

BUFFER MAN-AGERS	AVE-RAGE UN-PARTI-TIONED BUFFER SIZE IN PAGES	VIRTUAL BUFFER SIZE IN RECORDS (PAGES)	TOTAL I/O IN SYSTEM (S) AND BUFFER (B)	NUMBER OF PAGES DEDICATED TO THE VIRTUAL BUFFER						
				3	5	7	9	11	13	UNPAR-TITIONED
RAND	7.16	10 (9)	S	1874	1714	1951	1955	----	----	1850
			B	1625	1243	918	568	----	----	805
SCH	7.84	10 (9)	S	1837	1904	2318	2052	----	----	1974
			B	1570	1425	1301	696	----	----	936
RAND	9.77	15 (13)	S	1629	1625	2042	2201	2415	2336	2093
			B	1378	1167	968	816	588	283	717
SCH	9.45	15 (13)	S	1504	1715	2106	2368	2627	2616	2196
			B	1237	1165	1089	1002	745	435	856

TABLE IV

A comparison of the partitioned and unpartitioned total I/O using the SCH paging algorithm with 44K of real memory

	Number of Pages Dedicated to the Buffer	Virtual Buffer Size in Records 5	10	15	20
Experiment	3	1575	1749	1477	1500
	5	788	1298	1216	1317
Model with D = 20 records	3	1236	1254	1095	895
	5	806	1015	930	776
Model with D = 28 records	3	1313	1407	1326	1202
	5	883	1168	1160	1083

TABLE V

A comparison between the model and a partitioned experiment using the RAND paging algorithm, RAND buffer manager and 44K of real memory

partitioned by the paging algorithm. We showed empirically that the total I/O in the buffer can be reduced when the virtual memory size is increased and the real memory size in the buffer is controlled by the paging algorithm. The interaction between the paging algorithm and the buffer algorithm did not have a significant effect on the total I/O in the buffer. While a noticeable amount of paging in the buffer could be attributed to double paging, the double paging rate was not significantly different for any combination of paging algorithm and buffer algorithm.

The trends of the results on total I/O in the buffer were predicted quite well by a very simplistic model developed in the paper. The model also predicted that the total I/O in the buffer would decrease under certain conditions if the real memory in the buffer were held constant and the virtual memory was increased. This predicted decrease in total I/O is contrary to the results of a different model of an I/O bound program which showed that total I/O increased monotonically under approximately the same conditions.

Some preliminary experiments were run with the real memory in the buffer held fixed and separated from the real memory available to the program. The results of those experiments indicated that the total I/O in the buffer could decrease when the virtual buffer size was increased. These experiments will be expanded and investigated further in a later paper.

It was shown empirically that as the virtual buffer size increased, the average amount of real memory in the buffer increased thereby decreasing the real memory available to the program. Even when the total I/O in the buffer was decreased, the total I/O of the experiment could possibly be increased as a result of increased program paging. We showed that the total I/O in the experiment can decrease when the virtual buffer size is increased but that the results depend strongly on the amount of real memory available.

BIBLIOGRAPHY

1. Goldberg, R. and R. Hassinger, "The Double Paging Anomaly," Proc. 1974 National Computer Conference, Chicago, May 6-8, 1974.

2. Tsao, R. F.; L. W. Comeau, and B. H. Margolin, "A Multi-Factor Paging Experiment I, II," Statistical Computer Performance Evaluation, edited by Walter Freiberger, Academic Press, pp. 103-158.

3. Tuel, W. G., "An Analysis of Buffer Paging in Virtual Storage Systems," IBM Report RJ 1421, July 1974.

4. Tuel, W. G., "An Analysis of Buffer Paging in Virtual Storage Systems," Proc. Third Texas Conference on Computing Systems, Austin, Texas, Nov. 1974.

5. Personal Correspondence with Juan Rodriguez-Rosell, IBM, San Jose, CA

6. Carlson, E. D., "Using Large Data Bases for Interactive Problem Solving," Proc. International Conference on Very Large Data Bases, Framingham, Mass., September 1975, pp. 499-501.

7. Rodriguez-Rosell, J., and D. Hilderbrand, "A Framework for Evaluation of Data Base Systems," IBM Report RJ 1587, May 1975.

8. Personal Correspondence with Mark Sherman, Standard Oil of Cal., San Francisco,CA.

9. Corbato', R. J., "A Paging Experiment with the Multics System," In Honor of P. M. Morse, M.I.T. Press, Cambridge, Mass., 1969, pp. 217-228.

10. PRIME Computer User Guide, PRIME Computer Inc., Framingham, Mass., 1974.

11. Whetstone, W. D., "Computer Analysis of Large Linear Frames," Journal of Structures Division, ASCE, November 1969.

12. Belady, L. A., "A study of Replacement Algorithms for a Virtual Storage Computer," IBM Systems Journal 5 (1966), pp. 78-101.

13. Schwetman, H. D. and J. C. Browne, "An Experimental Study of Computer System Performance," Proc. National ACM Conference, Boston, Mass., August 1972, pp. 693-703.

APPENDIX

VIRTUAL BUFFER SIZE IN RECORDS

	1	5	10	15	20
FIFO PAGING ALGORITHM--FIFO BUFFER MANAGER--36K REAL MEMORY					
TOTAL PAGE TURNS	3640	4332	4940	5101	5092
DOUBLE PAGING	35	266	414	370	113
REFERENCE PAGING	41	71	137	305	626
REAL BUFFER SIZE	1	4.62	6.73	7.92	8.41
FIFO PAGING ALGORITHM--FIFO BUFFER MANAGER--40K REAL MEMORY					
TOTAL PAGE TURNS	2634	3153	3404	3590	3658
DOUBLE PAGING	17	210	296	326	110
REFERENCE PAGING	32	45	103	196	475
REAL BUFFER SIZE	1	4.68	5.36	8.57	9.09
FIFO PAGING ALGORITHM--FIFO BUFFER MANAGER--44K REAL MEMORY					
TOTAL PAGE TURNS	181	841	2102	2454	2859
DOUBLE PAGING	3	59	216	220	95
REFERENCE PAGING	0	8	59	185	411
REAL BUFFER SIZE	1	4.86	7.78	9.59	10.18
FIFO PAGING ALGORITHM--FIFO BUFFER MANAGER--48K REAL MEMORY					
TOTAL PAGE TURNS	66	88	178	583	997
DOUBLE PAGING	1	6	19	74	77
REFERENCE PAGING	0	1	3	32	130
REAL BUFFER SIZE	1	4.93	8.70	11.73	13.48
FIFO PAGING ALGORITHM--SCH BUFFER MANAGER--36K REAL MEMORY					
TOTAL PAGE TURNS	3579	4409	5015	5089	5010
DOUBLE PAGING	31	270	420	370	112
REFERENCE PAGING	43	73	156	326	603
REAL BUFFER SIZE	.99	4.62	6.82	8.20	8.21
FIFO PAGING ALGORITHM--SCH BUFFER MANAGER--40K REAL MEMORY					
TOTAL PAGE TURNS	2618	3182	3445	3630	3650
DOUBLE PAGING	14	203	318	336	112
REFERENCE PAGING	35	53	108	204	471
REAL BUFFER SIZE	.99	4.68	7.28	8.80	8.81
FIFO PAGING ALGORITHM--SCH BUFFER MANAGER--44K REAL MEMORY					
TOTAL PAGE TURNS	236	823	1581	2612	2995
DOUBLE PAGING	4	54	150	287	96
REFERENCE PAGING	0	12	62	139	427
REAL BUFFER SIZE	1	4.85	8.07	9.67	9.69
FIFO PAGING ALGORITHM--SCH BUFFER MANAGER--48K REAL MEMORY					
TOTAL PAGE TURNS	68	87	263	632	1094
DOUBLE PAGING	1	7	27	79	89
REFERENCE PAGING	0	0	8	31	145
REAL BUFFER SIZE	1	4.93	8.67	11.75	13.26
FIFO PAGING ALGORITHM--RAND BUFFER MANAGER--36K REAL MEMORY					
TOTAL PAGE TURNS	3640	4250	4997	5251	5441
DOUBLE PAGING	35	253	328	233	186
REFERENCE PAGING	41	72	234	503	726
REAL BUFFER SIZE	1	4.52	6.69	8.36	10.13
FIFO PAGING ALGORITHM--RAND BUFFER MANAGER--40K REAL MEMORY					
TOTAL PAGE TURNS	2633	3157	3422	3650	4019
DOUBLE PAGING	17	194	261	202	179
REFERENCE PAGING	32	59	156	344	537
REAL BUFFER SIZE	1	4.58	7.15	9.04	10.92

FIFO PAGING ALGORITHM--RAND		BUFFER MANAGER--44K		REAL MEMORY	
TOTAL PAGE TURNS	180	828	1696	2744	3147
DOUBLE PAGING	3	44	145	183	157
REFERENCE PAGING	0	22	74	280	465
REAL BUFFER SIZE	1	4.82	7.91	9.83	11.47
FIFO PAGING ALGORITHM--RAND		BUFFER MANAGER--48K		REAL MEMORY	
TOTAL PAGE TURNS	67	91	182	615	1288
DOUBLE PAGING	1	9	19	55	104
REFERENCE PAGING	0	0	4	50	179
REAL BUFFER SIZE	1	4.93	8.72	11.94	14.53
FIFO PAGING ALGORITHM--LRU		BUFFER MANAGER--36K		REAL MEMORY	
TOTAL PAGE TURNS	3337	4031	4744	4786	5007
DOUBLE PAGING	38	249	436	361	120
REFERENCE PAGING	31	78	136	326	714
REAL BUFFER SIZE	.99	4.68	7.19	8.80	9.53
FIFO PAGING ALGORITHM--LRU		BUFFER MANAGER--40K		REAL MEMORY	
TOTAL PAGE TURNS	2489	3129	3413	3619	3696
DOUBLE PAGING	16	196	326	337	117
REFERENCE PAGING	31	62	110	222	527
REAL BUFFER SIZE	.99	4.73	7.56	9.41	10.09
FIFO PAGING ALGORITHM--LRU		BUFFER MANAGER--44K		REAL MEMORY	
TOTAL PAGE TURNS	236	622	1712	2480	2714
DOUBLE PAGING	4	35	166	242	106
REFERENCE PAGING	0	15	61	190	424
REAL BUFFER SIZE	1	4.87	8.05	10.15	11.18
FIFO PAGING ALGORITHM--LRU		BUFFER MANAGER--48K		REAL MEMORY	
TOTAL PAGE TURNS	70	135	259	544	1103
DOUBLE PAGING	1	9	26	59	61
REFERENCE PAGING	0	1	10	37	180
REAL BUFFER SIZE	1	4.92	8.64	11.85	13.90
SCH PAGING ALGORITHM--FIFO		BUFFER MANAGER--36K		REAL MEMORY	
TOTAL PAGE TURNS	2762	3302	3689	3874	3902
DOUBLE PAGING	32	272	392	362	109
REFERENCE PAGING	24	62	151	353	682
REAL BUFFER SIZE	.99	4.61	6.73	7.67	8.13
SCH PAGING ALGORITHM--FIFO		BUFFER MANAGER--40K		REAL MEMORY	
TOTAL PAGE TURNS	1530	2226	2446	2646	2797
DOUBLE PAGING	18	208	298	318	106
REFERENCE PAGING	0	26	82	198	473
REAL BUFFER SIZE	.99	4.72	7.20	8.45	8.99
SCH PAGING ALGORITHM--FIFO		BUFFER MANAGER--44K		REAL MEMORY	
TOTAL PAGE TURNS	78	489	1255	1740	1910
DOUBLE PAGING	1	36	195	281	92
REFERENCE PAGING	0	6	38	149	417
REAL BUFFER SIZE	1	4.89	7.69	9.26	9.8
SCH PAGING ALGORITHM--FIFO		BUFFER MANAGER--48K		REAL MEMORY	
TOTAL PAGE TURNS	26	41	81	269	648
DOUBLE PAGING	1	5	18	86	84
REFERENCE PAGING	0	0	0	6	167
REAL BUFFER SIZE	1	4.93	8.73	11.87	12.95
SCH PAGING ALGORITHM--SCH		BUFFER MANAGER--36K		REAL MEMORY	
TOTAL PAGE TURNS	2762	3314	3714	3865	3918
DOUBLE PAGING	32	271	421	367	112
REFERENCE PAGING	24	65	158	374	670
REAL BUFFER SIZE	.99	4.62	6.82	7.89	7.87

SCH PAGING ALGORITHM--SCH		BUFFER MANAGER--40K		REAL MEMORY	
TOTAL PAGE TURNS	1451	2249	2515	2699	2800
DOUBLE PAGING	15	205	324	326	109
REFERENCE PAGING	0	26	98	204	470
REAL BUFFER SIZE	.99	4.70	7.29	8.74	8.70
SCH PAGING ALGORITHM--SCH		BUFFER MANAGER--44K		REAL MEMORY	
TOTAL PAGE TURNS	75	492	1287	1774	1849
DOUBLE PAGING	1	39	216	278	94
REFERENCE PAGING	0	4	33	156	407
REAL BUFFER SIZE	1	4.90	7.84	9.45	9.58
SCH PAGING ALGORITHM--SCH		BUFFER MANAGER--48K		REAL MEMORY	
TOTAL PAGE TURNS	26	45	85	303	798
DOUBLE PAGING	1	5	14	80	88
REFERENCE PAGING	0	0	0	17	215
REAL BUFFER SIZE	1	4.93	8.72	11.87	12.61
SCH PAGING ALGORITHM--RAND		BUFFER MANAGER--36K		REAL MEMORY	
TOTAL PAGE TURNS	2690	3440	3943	4084	4291
DOUBLE PAGING	36	251	333	232	186
REFERENCE PAGING	25	79	243	513	735
REAL BUFFER SIZE	.99	4.52	6.74	8.22	9.84
SCH PAGING ALGORITHM--RAND		BUFFER MANAGER--40K		REAL MEMORY	
TOTAL PAGE TURNS	1767	2364	2661	2786	3132
DOUBLE PAGING	17	193	253	202	178
REFERENCE PAGING	9	38	162	342	532
REAL BUFFER SIZE	.99	4.61	7.14	9.08	10.89
SCH PAGING ALGORITHM--RAND		BUFFER MANAGER--44K		REAL MEMORY	
TOTAL PAGE TURNS	75	477	1291	1823	2152
DOUBLE PAGING	1	35	162	177	157
REFERENCE PAGING	0	6	84	270	434
REAL BUFFER SIZE	1	4.87	7.66	9.77	11.57
SCH PAGING ALGORITHM--RAND		BUFFER MANAGER--48K		REAL MEMORY	
TOTAL PAGE TURNS	26	42	100	276	978
DOUBLE PAGING	1	5	20	55	104
REFERENCE PAGING	0	0	0	25	248
REAL BUFFER SIZE	1	4.93	8.72	11.94	13.90
SCH PAGING ALGORITHM--LRU		BUFFER MANAGER--36K		REAL MEMORY	
TOTAL PAGE TURNS	2690	3047	3583	3749	3858
DOUBLE PAGING	36	260	425	364	119
REFERENCE PAGING	25	63	143	404	751
REAL BUFFER SIZE	.99	4.67	7.26	8.58	9.23
SCH PAGING ALGORITHM--LRU		BUFFER MANAGER--40K		REAL MEMORY	
TOTAL PAGE TURNS	1290	2022	2425	2613	2796
DOUBLE PAGING	14	195	329	327	113
REFERENCE PAGING	0	27	88	213	522
REAL BUFFER SIZE	1	4.75	7.54	9.39	10.08
SCH PAGING ALGORITHM--LRU		BUFFER MANAGER--44K		REAL MEMORY	
TOTAL PAGE TURNS	75	265	1007	1552	1836
DOUBLE PAGING	1	13	176	266	100
REFERENCE PAGING	0	0	29	132	430
REAL BUFFER SIZE	1	4.93	8.13	10.18	10.99
SCH PAGING ALGORITHM--LRU		BUFFER MANAGER--48K		REAL MEMORY	
TOTAL PAGE TURNS	34	43	56	216	712
DOUBLE PAGING	1	5	11	60	77
REFERENCE PAGING	0	0	0	8	183
REAL BUFFER SIZE	1	4.93	8.70	12.11	13.93

RAND PAGING ALGORITHM--FIFO		BUFFER MANAGER--36K		REAL MEMORY	
TOTAL PAGE TURNS	3602	4520	5118	5404	5187
DOUBLE PAGING	40	286	396	380	117
REFERENCE PAGING	38	68	157	305	575
REAL BUFFER SIZE	.99	4.51	6.59	7.23	7.99
RAND PAGING ALGORITHM--FIFO		BUFFER MANAGER--40K		REAL MEMORY	
TOTAL PAGE TURNS	1474	2348	2908	3224	3385
DOUBLE PAGING	17	149	260	285	104
REFERENCE PAGING	8	34	92	189	414
REAL BUFFER SIZE	.99	4.69	7.34	8.78	9.71
RAND PAGING ALGORITHM--FIFO		BUFFER MANAGER--44K		REAL MEMORY	
TOTAL PAGE TURNS	272	890	1298	1803	2042
DOUBLE PAGING	5	61	133	171	92
REFERENCE PAGING	1	16	34	110	275
REAL BUFFER SIZE	.99	4.85	8.04	10.16	11.58
RAND PAGING ALGORITHM--FIFO		BUFFER MANAGER--48K		REAL MEMORY	
TOTAL PAGE TURNS	106	157	274	540	928
DOUBLE PAGING	1	12	42	71	68
REFERENCE PAGING	1	2	1	29	131
REAL BUFFER SIZE	1	4.92	8.62	11.79	13.86
RAND PAGING ALGORITHM--SCH		BUFFER MANAGER--36K		REAL MEMORY	
TOTAL PAGE TURNS	3517	4444	5062	5309	5216
DOUBLE PAGING	44	249	418	368	119
REFERENCE PAGING	27	79	150	322	552
REAL BUFFER SIZE	.99	4.58	6.80	7.45	7.90
RAND PAGING ALGORITHM--SCH		BUFFER MANAGER--40K		REAL MEMORY	
TOTAL PAGE TURNS	1489	2420	2981	3299	3292
DOUBLE PAGING	13	144	268	277	108
REFERENCE PAGING	10	33	86	205	431
REAL BUFFER SIZE	.99	4.73	7.49	8.95	9.34
RAND PAGING ALGORITHM--SCH		BUFFER MANAGER--44K		REAL MEMORY	
TOTAL PAGE TURNS	287	716	1434	1785	2015
DOUBLE PAGING	3	59	154	178	87
REFERENCE PAGING	0	9	43	121	268
REAL BUFFER SIZE	.99	4.88	8.08	10.33	11.37
RAND PAGING ALGORITHM--SCH		BUFFER MANAGER--48K		REAL MEMORY	
TOTAL PAGE TURNS	99	149	400	576	907
DOUBLE PAGING	2	10	49	76	62
REFERENCE PAGING	0	1	8	34	130
REAL BUFFER SIZE	.99	4.93	8.52	11.54	13.65
RAND PAGING ALGORITHM--RAND		BUFFER MANAGER--36K		REAL MEMORY	
TOTAL PAGE TURNS	3649	4483	5122	5498	5713
DOUBLE PAGING	39	236	331	244	197
REFERENCE PAGING	22	114	240	478	714
REAL BUFFER SIZE	.99	4.47	6.64	8.11	9.18
RAND PAGING ALGORITHM--RAND		BUFFER MANAGER--40K		REAL MEMORY	
TOTAL PAGE TURNS	1544	2448	2936	3413	3754
DOUBLE PAGING	19	148	229	195	169
REFERENCE PAGING	16	51	129	325	511
REAL BUFFER SIZE	.99	4.66	7.39	9.14	11.15
RAND PAGING ALGORITHM--RAND		BUFFER MANAGER--44K		REAL MEMORY	
TOTAL PAGE TURNS	281	786	1329	1730	2326
DOUBLE PAGING	3	59	135	127	131
REFERENCE PAGING	0	18	49	174	307
REAL BUFFER SIZE	.99	4.85	8.03	10.32	12.87

RAND PAGING ALGORITHM--RAND		BUFFER MANAGER--48K		REAL MEMORY	
TOTAL PAGE TURNS	90	201	314	711	1053
DOUBLE PAGING	2	15	28	67	89
REFERENCE PAGING	2	1	11	52	150
REAL BUFFER SIZE	.99	4.90	8.64	11.54	14.55
RAND PAGING ALGORITHM--LRU		BUFFER MANAGER--36K		REAL MEMORY	
TOTAL PAGE TURNS	3055	4057	4721	4959	5174
DOUBLE PAGING	36	244	403	369	117
REFERENCE PAGING	29	83	150	321	679
REAL BUFFER SIZE	.99	4.62	7.12	8.32	8.90
RAND PAGING ALGORITHM--LRU		BUFFER MANAGER--40K		REAL MEMORY	
TOTAL PAGE TURNS	1148	2066	2706	3109	3293
DOUBLE PAGING	17	159	264	272	113
REFERENCE PAGING	9	35	73	230	478
REAL BUFFER SIZE	.99	4.75	7.65	9.44	10.59
RAND PAGING ALGORITHM--LRU		BUFFER MANAGER--44K		REAL MEMORY	
TOTAL PAGE TURNS	328	507	1041	1619	1888
DOUBLE PAGING	7	35	116	191	92
REFERENCE PAGING	1	9	37	105	298
REAL BUFFER SIZE	.99	4.91	8.13	10.61	12.21
RAND PAGING ALGORITHM--LRU		BUFFER MANAGER--48K		REAL MEMORY	
TOTAL PAGE TURNS	85	145	198	437	782
DOUBLE PAGING	1	9	19	49	55
REFERENCE PAGING	0	3	7	34	119
REAL BUFFER SIZE	1	4.92	8.66	11.68	14.10

TOTAL PAGE TURNS CONSIST OF THE PAGING IN THE BUFFER AND PROGRAM
DOUBLE PAGING REFERS TO PAGING IN THE VIRTUAL BUFFER
REFERENCE PAGING REFERS TO PAGING IN THE VIRTUAL BUFFER
TOTAL PAGING IN THE BUFFER = DOUBLE PAGING + REFERENCE PAGING
REAL BUFFER SIZE IS THE AVERAGE NUMBER OF PAGES OF REAL MEMORY IN THE VIRTUAL BUFFER