EIS-TR-76-2

SIMULATION OF A COMPUTER SYSTEM WITH BUFFER MEMORY

by

Tomoo Nakamura

Hajime Kitagawa

Masanori Kanazawa

Hiroshi Hagiwara

February 23, 1976

INSTITUTE
OF
ELECTRONICS AND INFORMATION SCIENCE

UNIVERSITY OF TSUKUBA

EIS-TR-76-2

SIMULATION OF A COMPUTER SYSTEM WITH BUFFER MEMORY*

Tomoo Nakamura**

Hajime Kitagawa***

Masanori Kanazawa***

Hiroshi Hagiwara****

February 23, 1976

Abstract

In recent years buffer memory has become more popular in order to attain high CPU performance in large systems. In the analysis of the effectiveness of buffer memory in multiprogramming environments, it is important to take the influence of task-switching into consideration as well as design parameters of buffer memory.

We have made to evaluate the effectiveness of buffer memory by means of simulation. In this simulation a multiprogramming model considering task-switch control is proposed and the patterns of memory reference taken by tracing are used as input data. This paper presents the method and results of the simulation in the analysis of buffer memory.

*This report is a revised edition of our paper appeared in Journal of Information Processing Society of Japan,Vol.15,No.1(1974).
**Institute of Electronics and Information Science,University of Tsukuba, Niihari-Gun, Ibaraki 300-31, Japan
***Data Processing Center, Kyoto University, Sakyo-Ku, Kyoto 606, Japan
****Department of Information Science, Kyoto University, Sakyo-Ku, Kyoto 606, Japan

Introduction

In recent years buffer memory, or cache memory, has become a popular and indispensable function in order to attain high CPU-performance in large-scale computer systems.

Buffer memory is high-speed memory of small capacity that is used to compensate for the difference in operating speed between main memory and CPU. It is organized into fixed-size blocks of information, and automatically keeps the information most recently used by CPU.

The transfer of blocks between buffer memory and main memory is controlled by hardware. The effectiveness of buffer memory depends on such design parameters; the block size of information, buffer memory capacity, replacement algorithm of blocks and the block transfer rate. In addition, in multiprogramming environments, it is important to take the influence of task-switching into consideration as well as the design parameters.

We have made to evaluate the effectiveness of buffer memory by simulation. In this simulation a multiprogramming model considering task-switch control is proposed and the address patterns of memory reference taken by tracing in an actual computer system are used as input data. This report presents the method and results of the simulation in the analysis of buffer memory.

Adress tracing

To measure the dynamic behavior of programs and to make the input data for memory reference simulation, we have prepared the address tracer by which the data referencing pattern of a user program during execution can be recorded. The address tracer runs under the FACOM 230-60 batch operating systems. The trace program is link-edited with the object program being traced, and only user program mode instructions are recorded.*

The unit of data which is to be sequentially recorded on magnetic tape consists of,

(1) address of the instruction which references memory or causes transfer of control (jump or monitor-call), and its instruction code,

(2) address of the referenced memory (including all the referenced addresses in the case of indirect addressing), and

(3) type of the reference; i.e. load/store as an operand or instruction fetch.

We selected several FORTRAN programs as the benchmarks and got the traced data while they were being compiled and being executed. (See Table 1.)

---

* The moderate slow-down of execution under tracing is obtained 50:1 to 80:1. The address tracer is written in FASP assembler language and its size is about 700 words.

Table 1

trace data	compile /execute	comp. type	prog. size (kilo-w)	f ($\times 10^4$)	P_R %	P_W %	P_X %	
Differential equation RKG	C	FORTRAN-C non-optimize	*38	22	20.0	13.0	67.0	a1
	E		9	47	17.1	14.8	68.1	a2
EIGEN VALUE 10x10 symmetric matrix Jacobi	C		*38	83	21.8	11.2	67.0	b1
	E		14	79	31.0	10.5	58.5	b2
	C	FORTRAN-D optimize	*38	177	21.7	11.3	67.0	c1
	E		14	39	25.8	11.6	62.6	c2
Sorting	C		*38	38	21.0	12.8	66.2	d1
	E		9	**50	22.3	18.8	58.9	d2
BM Simulator	E		16	**50	26.7	13.8	59.5	e2

f : frequency of data reference by CPU,

P_R : frequency of operand fetch / f,

P_W : frequency of operand store / f,

P_X : frequency of instruction fetch / f,

P_F : frequency of instruction or operand fetch / f,

and $P_R + P_W + P_X = 1$, $P_F = 1 - P_W$.

*the amount of core used

**Tracing was stopped when f = 500,000.

Simulation of Buffer Memory System

As a simulation model of memory references we consider a computer system with buffer memory(BM) which is controlled by the set-associative mapping algorithm.[2] As shown in Fig. 1, in this model buffer memory and main memory(MM) are both divided into blocks of m words and set-associative algorithm maps each group of blocks in the same column of MM into the corresponding column of BM. When a request for data is made by the CPU, it is determined whether the requested word is in BM or not. If the word is in BM, it is transmitted to the CPU without accessing MM. If it is not in BM, a MM-fetch is initiated and send directly to the CPU. The CPU must be delayed while it is obtained from MM. When this occurs the entire block containing the requested word is also transferred into BM. If the corresponding column in BM is already full, an appropriate block must be replaced. We take the least recently used (LRU) algorithm[3] for replacement of blocks. When a word is stored by the CPU, only MM is updated if the corresponding location is not in BM.[4] If the location is currently contained in BM, we adopt the following data storing algorithms; (a) storing the data to MM as well as to BM concurrently (through-storing) and (b) storing the complete blocks that have been updated only when they are displaced from BM (post-storing).

The computer system in this model is a multiprogrammed single-CPU system. The influence of task-switch control on the effectiveness of buffer memory is considered in this simulation. In a multiprogramming system, task-switching is usualy caused by the run of

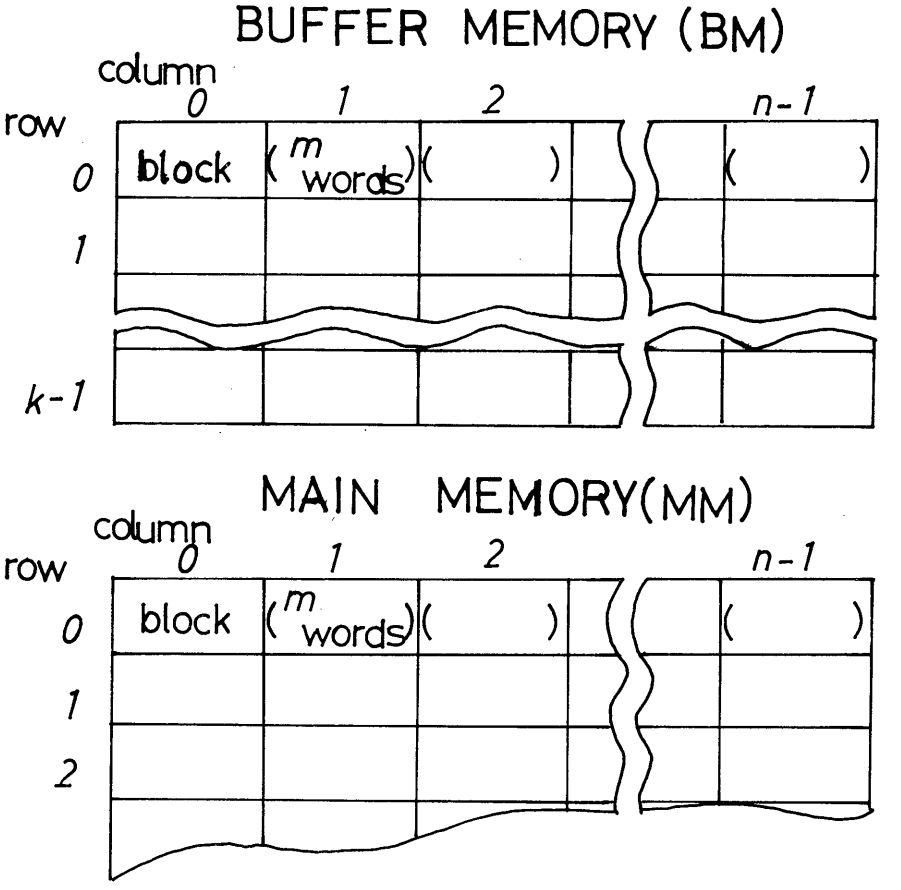

Fig. 1

Structure of Buffer Memory and Main Memory.

monitor-task which is initiated by a monitor-call or an interruption. Then, the monitor-call by TRAP instruction in the traced data which mainly requests I/O operation can be used in the simulator as one of the initiation of a task-switch. As the information on interruption is not, however, contained in the traced data, we prepare another simulation parameter; i.e. the average interval of task-switches (the number of reference f_t) caused by interruption.

We have developed the following two kinds of method of simulation considering the task-switch control:

Method 1 (BM reset method). In multiprogramming environments, one program is paid attention to; the data of the program in BM become invalid because of task-switch whenever (a) the CPU has made the references of f_t times to that program, or (b) a TRAP instruction has been executed in that program before (a) occurs.

This method is considered as the worst case in which all of the data in BM are displaced by the run of other programs before the the program gets the CPU again.

Method 2 (program switching method). Multiprograms running in the system are processed by CPU in order and a task-switch is generated whenever (a) the CPU has made the references of f_t times to one program, or (b) a TRAP instruction in that program has been executed before (a) occurs.

In this method the data in BM may be displaced by the run of other programs, and when some program gets the CPU again the data in BM which have not been displaced by others can be used. This method is considered the rather good case because the run of monitor-tasks can be disregarded.

Finally, we adopt the assumption that the CPU in this model does not take the advanced control. Therefore we do not consider that the CPU transmits the blocks from MM to BM in the stage of pre-fetching an instruction or operands, and that the CPU performs store to MM and execution of the next instruction concurrently.

The parameters in this simulation are as follows; block size (m words), number of columns (n), column size of BM (k blocks), capacity of BM ($C = m \times n \times k$ words) and average interval between task-switches (f_t references).

C, m, n and k are the design parameters of the BM structure, and f_t is the parameter which is dependent on the system behavior. We got the following kinds of data on the effectiveness of BM as the outputs of the simulator while varying the combination of parameter values using the traced data as the inputs:

P_B : frequency of access to BM / f,

P_{BR} : frequency of operand fetch from BM / f,

P_{BW} : frequency of operand store to BM / f,

P_{BX} : frequency of instruction fetch from BM / f,

($P_B = P_{BR} + P_{BW} + P_{BX}$)

P_{PS} : frequency of the case when the block updated on BM is displaced from BM / f,

P_{BL} : frequency of block transfer to BM from MM / f

where

f : frequency of memory reference to user programs by the CPU.

Effectiveness of buffer memory

Results of simulation

We have adopted the following values that can be important measures on the performance evaluation of the systems with BM:

(1) P_B : relative frequency(probability) of access to and from BM,

(2) in the case of through-storing,

$P_T = P_B - P_{BW}$: relative frequency of memory reference at the speed of the BM access time,

(The effectiveness of BM-access decreases by operand store.)

(3) in the case of post-storing,

$P_P = P_B - P_{PS}$: relative frequency of memory reference at the speed of the BM access time,

(The effectiveness of BM-access decreases because of neccesity for transfer block to MM from BM when the updated block in BM is displaced by another block. It is assumed that both transfer times to BM from MM and to MM from BM are equal.)

(4) effective cycle time ratio α ; average BM cycles per a reference,

$\alpha_T = P_T + \beta \times (P_W + d \times P_{BL})$: through-storing,

$\alpha_P = P_B + \beta \times ((P_W - P_{BW}) + d \times (P_{BL} + P_{PS}))$: post-storing,

where β is the cycle time ratio of MM/BM and d is MM cycles required to transfer a complete block.

Some results are shown in the following figures.

1. Influence of mean task-switch interval upon BM-effectiveness (comparison between method 1 and method 2) (Fig.2).
2. Variation of BM-effectiveness with block size (Fig.3).
3. Variation of BM-effectiveness with BM capacity (Fig.4).

4. Variation of BM-effectiveness with column size (Fig.5).
5. Variation of BM-effectiveness with BM capacity and column size (Fig.6).
6. Variation of BM-effectiveness with address pattern (Fig.7).
7. Variation of Effective cycle time ratio α with block size if the data transfer width (m/d) is constant (Fig.8).

Influence of task-swithing upon BM-effectiveness

The following fact can be said from the results of this simulation. (1) Fig.2 indicates the variation of P_B (access rate to BM) with the average number of references f_t between successive task-switches. In the case of method 1 the value of P_B tends to decrease rapidly as f_t gets small. However, in the case of method 2 the decrease in the value of P_B as f_t gets small is slight. The degree of multiprogramming is 2 or 3 in this case. If the degree of multiprogramming increases the results by method 2 would have the tendency like those by method 1 because almost all the data of some task are displaced by other tasks before control of CPU returns to the task.

It can be said that the influence of task-switching upon BM-effectiveness is less than or equal to about 5% of P_B if the interval f_t gets greater than several thousands memory references.* One of the reason is, we suppose, that almost all the blocks required by a certain program for running are transferred to BM by the first several thousands memory references after the occurrence of a task-switch.

---

* Roughly speaking, the average task-switch interval is several thousands memory references in most acutual operating systems.

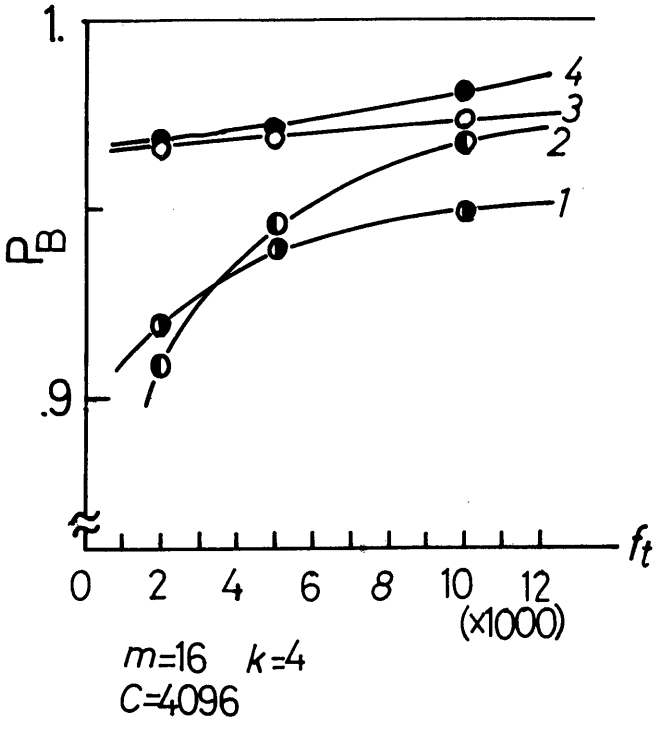

Fig.2

Effect of mean task-switch interval upon BM-effectiveness

1. *(a1)* by method 1
2. *(a2)* by method 1
3. *(a1,a2)* by metod 2
4. *(a1,a2,c2)* by method 2

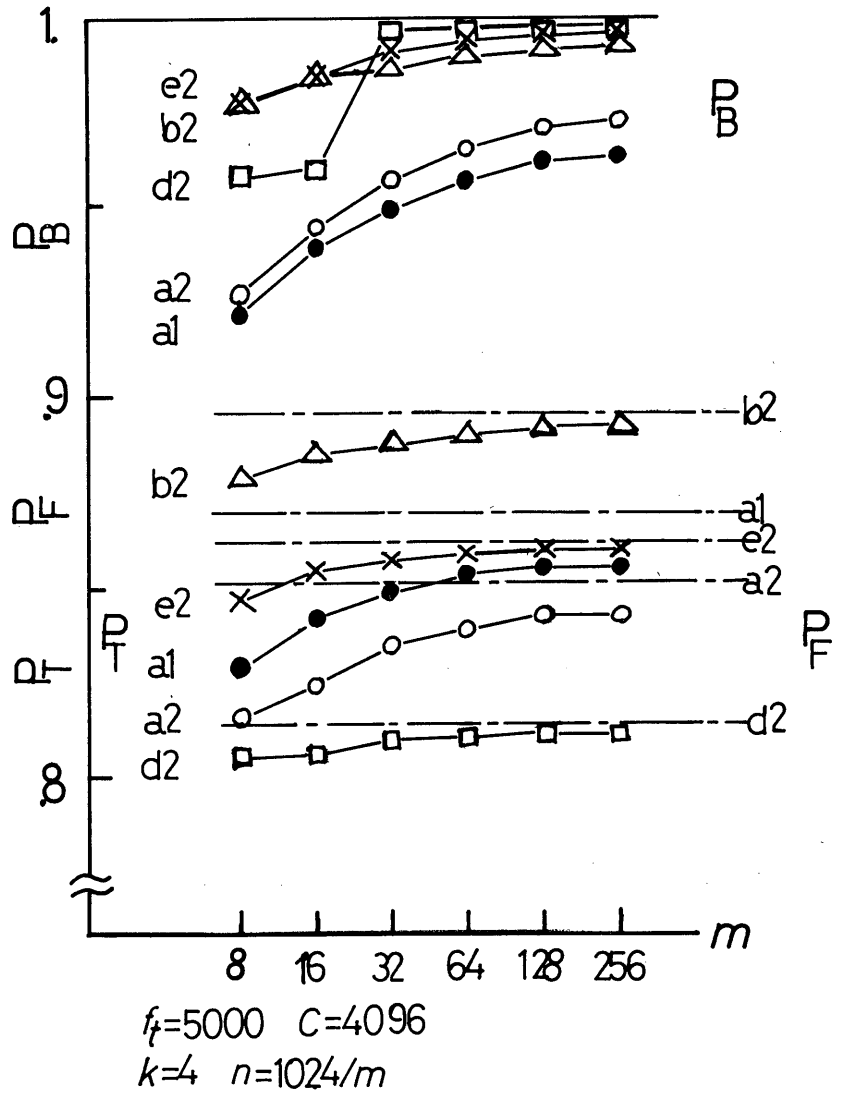

Fig.3

Variation of BM-effectiveness with block size.

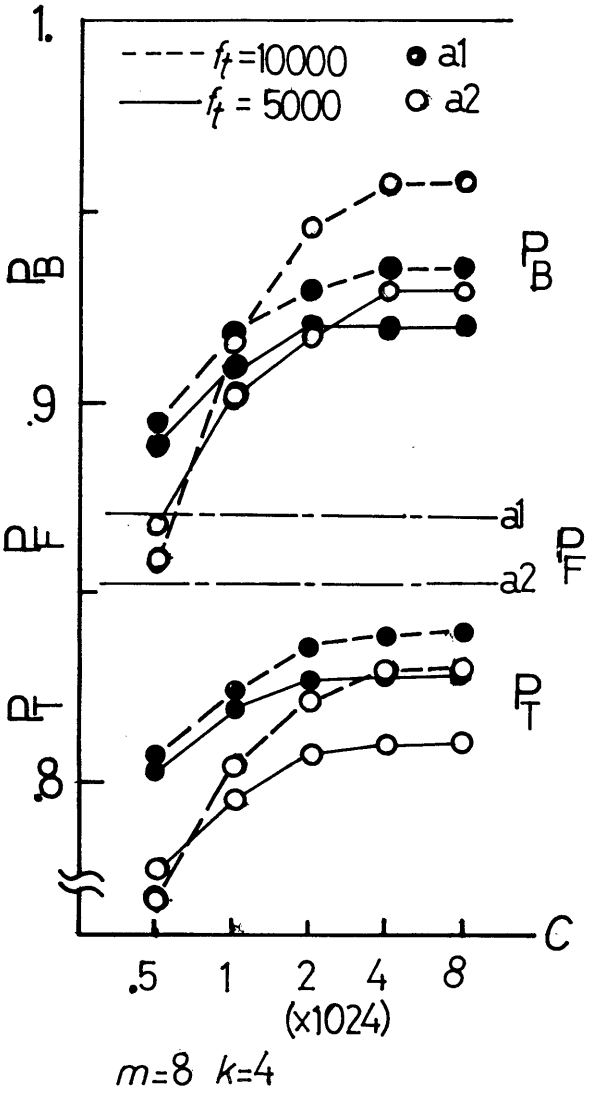

Fig.4
Variation of
BM-effectiveness
with BM capacity.

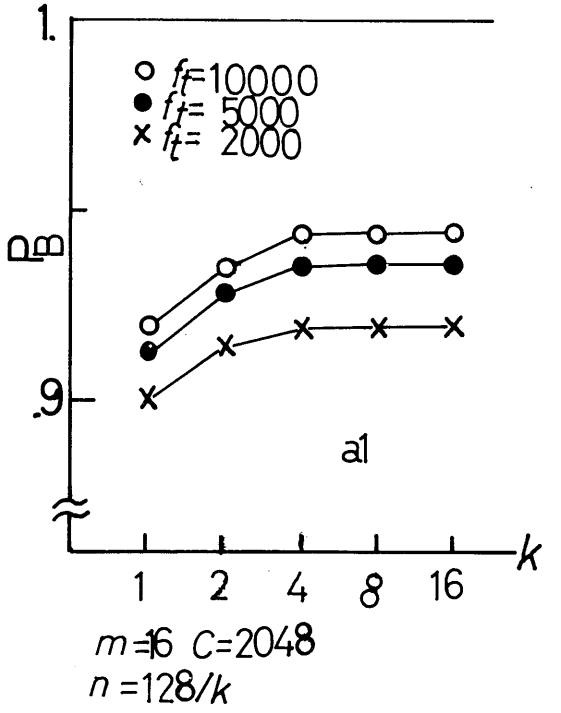

Fig.5
Variation of
BM-effectiveness
with column size.

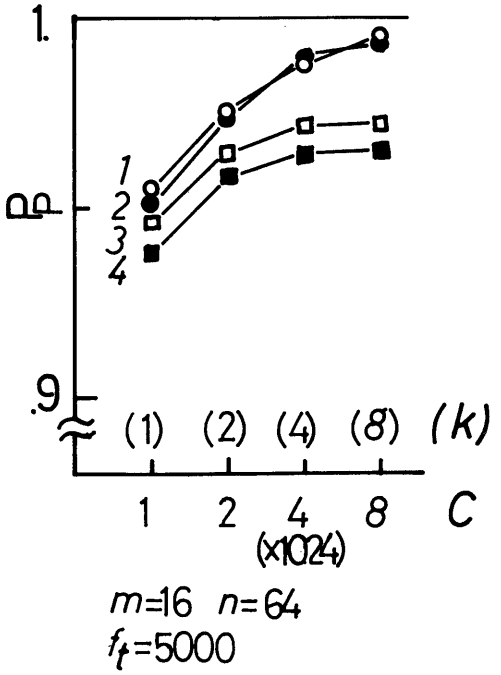

Fig.6

Variation of BM-effectiveness with capacity and column size.

1. *(a2,b2,e2)* by method 2
2. *(a2,b2)* by method 2
3. the average of *(a2,b2,e2)* by method 1
4. the average of *(a2,b2)* by mrthod 1

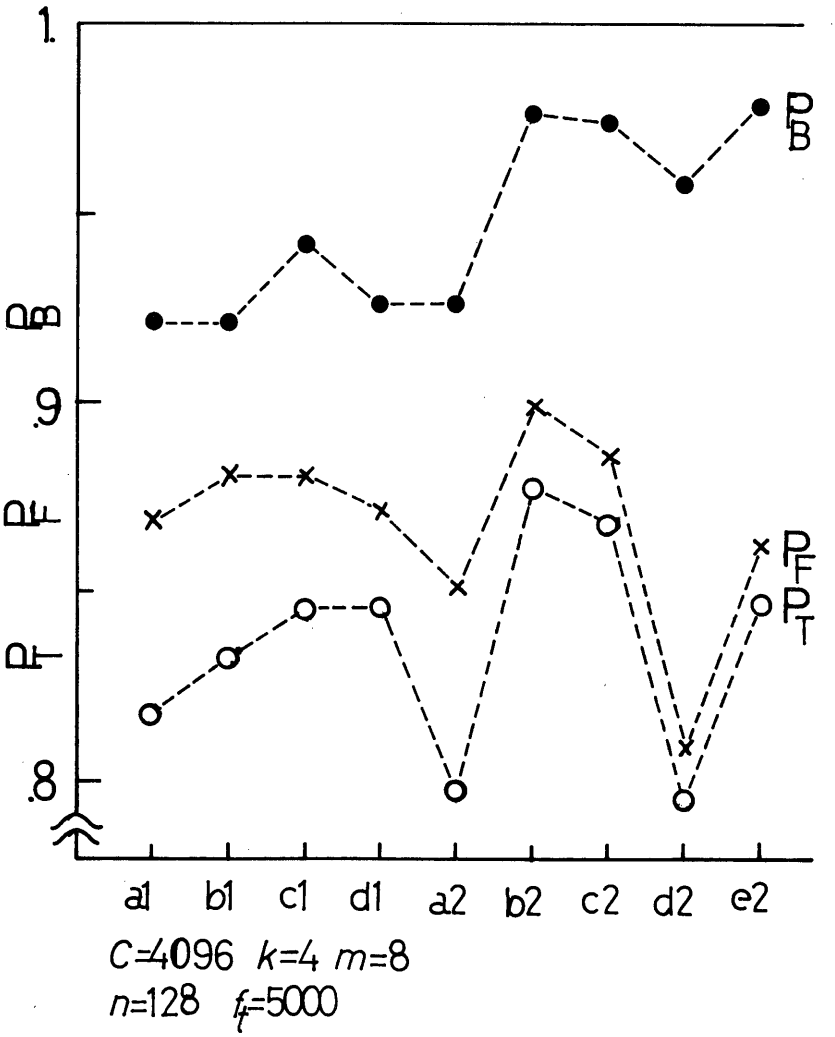

Fig.7

Variation of BM-effectiveness with address pattern.

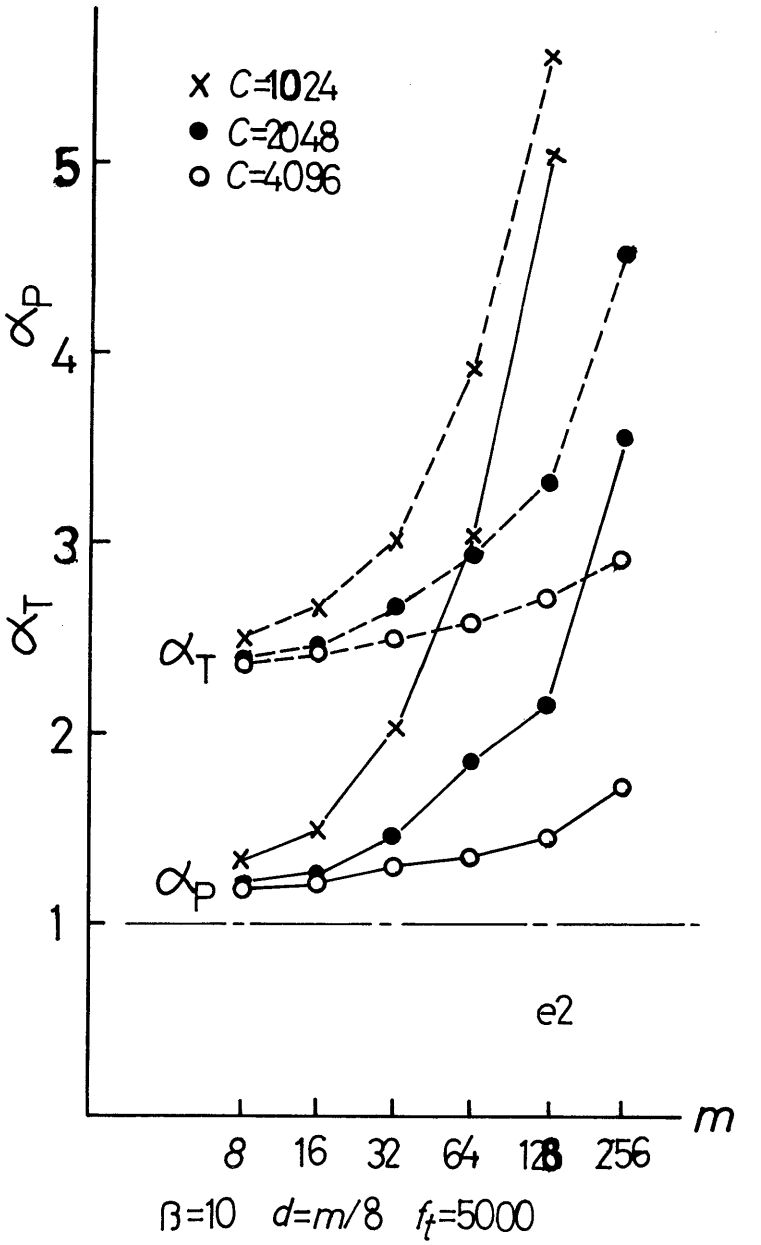

Fig.8

Effective cycle time ratio vs. block size.

(2) As shown in Fig.3, the values of P_B ($\fallingdotseq P_P$) and P_T increase as block size m increases if BM capacity C is kept constant. It can be said that this fact is caused by localities of memory references in the mean task-switch interval because BM is reset by task-switching in method 1 (in Fig.3 $f_t = 5,000$).Though not shown in this report, when programs run without task-switces ($f_t = \infty$), the value of P_B or P_T keeps almost constant (P_B = 0.99--0.96) while block size m increases in the case of 4 kilo-words (kw) BM capacity, and P_B or P_T decreases a little as m increases in the case of the smaller BM capacity.

(3) The effective cycle time ratio α_T and α_P do not alwalys become smaller by the increase of the access rate P_B : they are greatly influenced by the block transfer time. As shown in Fig.8, the increase of block size m causes the increase of α_T and α_P in the case that the data transfer width between BM and MM is narrow, and consequently BM-effectiveness decreases. In such a case, it may be said that the smaller block size (8 or 16) is better in multi-programming environments.

(4) It is clear that as BM capacity increases the value of P_B or P_T increases in multiprogramming environments if the other parameters are kept constant. However, Fig.4 shows that the value of P_B or P_T is saturated at about 4 kw BM capacity even if f_t is 10,000 references in the results by method 1. Because the amount of data transferred to BM between succesive task-switches is less than 2 kw when block size is 8 words as for the traced data in this simulation. Therefore it may be expected that in such a large BM capacity as 4 or 8 kw there is a difference in BM-effectiveness between the results by method 1 and method 2. Fig.6 shows an example comparing the results

by the two methods with the same address patterns as input data. The results by other address patterns would indicate the same tendency. It can be said that in the case of large BM capacities task-switching will have a relatively great influence on BM-effectiveness by task-switching becomes.

(5) The variation of the value of P_B with column size k for an address pattern $a1$ while BM capacity is constant is shown in Fig.5. When column size k is 1, the value of P_B is low particularly, and when k goes greater than 4, it becomes saturated. This means that 2 or 4 is nearly optimal as column size in the set-associative mapping algorithm in the view of cost/performance evaluation.

Remarks

We have simulated a simple model of a multiprogrammed single-CPU computer system with buffer memory and studied about the effectiveness of buffer memory. When the effectiveness of buffer memory in actual computers is to be evaluated, we shoud take into acount in the analysis (a) the advanced control by hardware and (b) the task management under the control of its operating system. If the advanced control is taken, though not considered in this simulation, pre-transfer of blocks from main memory to buffer memory is performed in the instruction or operand prefetch stage by the CPU even if the block may not be used, and the performance of the system may differ from the results to some extent. Moreover, by the advanced control of data storing to main memory, it may also be assumed that the degradation of the effectiveness of buffer memory becomes smaller,

and in the case of through-storing the deviation of the effectiveness caused by the different frequency of store-instructions in each program is less than that of the results. We have analyzed the effectiveness of buffer memory with multiprogramming environments in two typical cases, that is; the worst case that tha data in buffer memory are completely displaced by task-switching and the rather better casethat the data in buffer memory are replaced by other user programs while they are running. But in order to analyze the actual systems in detail the influence of the data in main memory being updated by I/O operations and of the run of monitor-tasks or interrupt handling routines will be important factors. Should the performance of computer systems be evaluated in consideration of these factors, it would be analyzed through simulating in more detailed model or through monitoring actual computer systems.

This work was supported in part by Project of Data Processing Center, Kyoto University.

References

1) Liptay,J.S. Structural aspects of the System/360,Model 85 II The cache, IBM Systems J.,Vol.7,No.1,pp.15-21,1968.

2) Conti,C.J. Concepts for buffer storage, IEEE Computer Group News, Vol.2,No.8,pp.9-13,1969.

3) Mattson,R.L,Gecsei,J.,Slutz,D.R.and Traiger,I.L. Evaluation techniques for storage hierarchies, IBM Systems J.,Vol.9,No.2, pp.78-117,1970.

4) Katzan, H.Jr. Computer Organization and the System/370, Van Nostrand Reinhold, 1971.

5) Sisson,S.S. and Flynn,M.J. Addressing patterns and memory handling algorithms, Proc. AFIPS 1968 FJCC,Vol.33,pp.957-967.

6) Meade,R.M. Design approaches for cache memory control, Computer Design,Vol.10,No.1,pp.87-93,1971.

7) Nakamura,T., Kitagawa,H. and Hagiwara,H. An analysis of program behavior, EIS-TR-76-3 (to appeare).

INSTITUTE OF ELECTRONICS AND INFORMATION SCIENCE
UNIVERSITY OF TSUKUBA
SAKURA-MURA, NIIHARI-GUN, IBARAKI JAPAN

REPORT DOCUMENTATION PAGE	REPORT NUMBER TR-76-2

TITLE

Simulation of a Computer System with Buffer Memory

AUTHOR(s)

Tomoo Nakamura (Institute of Electronics and Information Science, University of Tsukuba)
Hajime Kitagawa, Masanori Kanazawa
(Data Processing Center, Kyoto University)
Hiroshi Hagiwara
(Faculity of Engineering, Kyoto University)

REPORT DATE	NUMBER OF PAGES
February 23,1976	17
MAIN CATEGORY	CR CATEGORIES
Computer Systems	6.20, 8.1, 4.32

KEY WORDS

memory hierarchy, buffer memory, multiprogramming system, program behavior

ABSTRACT

In recent years buffer memory has become more popular in order to attain high CPU performance in large systems. In the analysis of the effectiveness of buffer memory in multiprogramming environments, it is important to take the influence of task-switching into consideration as well as design parameters of buffer memory.

We have made to evaluate the effectiveness of buffer memory by means of simulation. In this simulation a multiprogramming model considering task-switch control is proposed and the patterns of memory reference taken by tracing are used as input data. This paper presents the method and results of the simulation in the analysis of buffer memory.

SUPPLEMENTARY NOTES