The Comparison of Swapping Algorithms and Some Program Behaviors under a Paging Environment

TAKASHI MASUDA*, NOBUMASA TAKAHASHI* AND YASUFUMI YOSHIZAWA*

1. *Introduction*

This paper reports the comparison of swapping algorithms and some program behaviors under a paging environment. Address trace system called PATTERN (Program Address Trace patTERN) was developed for these purposes, which executes interpretively any program under HITAC 5020 Time-Sharing System instruction by instruction, recording the instruction code, the instruction address and the operand addresses of each instruction on the magnetic tape.

The H-5020TSS is a time-sharing system which has segmentation and paging features and has been in operation with 22 terminals at our Central Research Laboratory since March 1968.

Four typical programs under H-5020TSS were traced by PATTERN, and used as inputs for the analysis of swapping algorithms and program behaviors under a paging environment.

2. *Properties of the Programs Traced*

Four typical programs under H-5020TSS were traced. These programs are of reentrant codings and the different segments are assigned for a procedure part and a data part. The properties of the four programs traced are listed in Table 1. Some comments for these programs are given.

BASIC is a popular conversational language in our TSS. It is an incremental compiler. The process of the compilation of a circuit analysis program, which has 663 source steps by BASIC, was traced.

PL/1W is a subset of PL/1 and was developed as a system implementation language of H-5020TSS. It consists of two phases. A source program is compiled into a machine-independent intermediate language in the first phase and a machine code is generated in the second phase. The compiling process, at the first phase, of a supervisor module whose size is 90 steps was traced.

CONCISE is a conversational interpreter. The model program traced executes a simulation of swapping algorithms in the paging system. Its source code is 70 steps.

EIGEN is a program which finds eigenvalues and eigenvectors of a symmetric

This paper first appeared in Japanese in Joho-Shori (Journal of the Information Processing Society of Japan), Vol. 13, No. 2 (1972), pp. 81~88.
* Central Research Laboratory, Hitachi, Ltd.

Table 1. Programs to be analyzed.

	page size (word)	program size (page)			number of segments used			program steps (on Mag. Tape)
		proc.	data	sum	proc.	data	sum	
BASIC	64 256 1,024	44 15 7	547 143 43	591 158 50	5	8	13	549,105
PL/1W	64 256 1,024	148 47 18	194 70 38	342 117 56	12	19	31	356,173
CONCISE	64 256 1,024	56 16 8	67 25 12	123 41 20	6	7	13	514,972
EIGEN	64 256 1,024	16 6 5	171 47 16	187 53 21	5	5	10	195,296

matrix by JACOBI method. The program traced finds those values of a 50×50 symmetric matrix.

3. *Comparison of Swapping Algorithms*

Main memory size of H-5020TSS is 64K words and the page size is 256 words. Every programs but the resident operating system are loaded on-demand into the main memory. The resident operating system occupies about 20K words. Measurements from software monitor [1] show that more than 50% of a running time of the operating system is due to the processing of the page-fault interrupt and 30% of the total time is a pure idle time during which cpu has no task to execute and is waiting for the interrupt of the I/O termination.

It is apparent from these measurements that the frequency of page faults effects essentially the performance of a paging-machine. So we tried to find the relation among the swapping algorithm, the main memory size available for the program examined, the page size and the mean executing steps between successive page faults (msbf). The four programs mentioned are selected as model programs, and the address traces of those programs are used as inputs to an analyzer. The analyzer simulates four swapping algorithms (FIFO, LRU, RANDOM, OPTIMUM [2]). Any page size and main memory size can be selected. We assumed 16, 64, 256, 1024 words as a page size. Fig. 1 shows an output of the analyzer when the page size is 256 words. The mean executing steps between successive page faults increase sharply when the main memory size assigned for the program exceeds some critical value. This value when normalized by dividing its program size differs by each program, but it seems, regardless of swapping

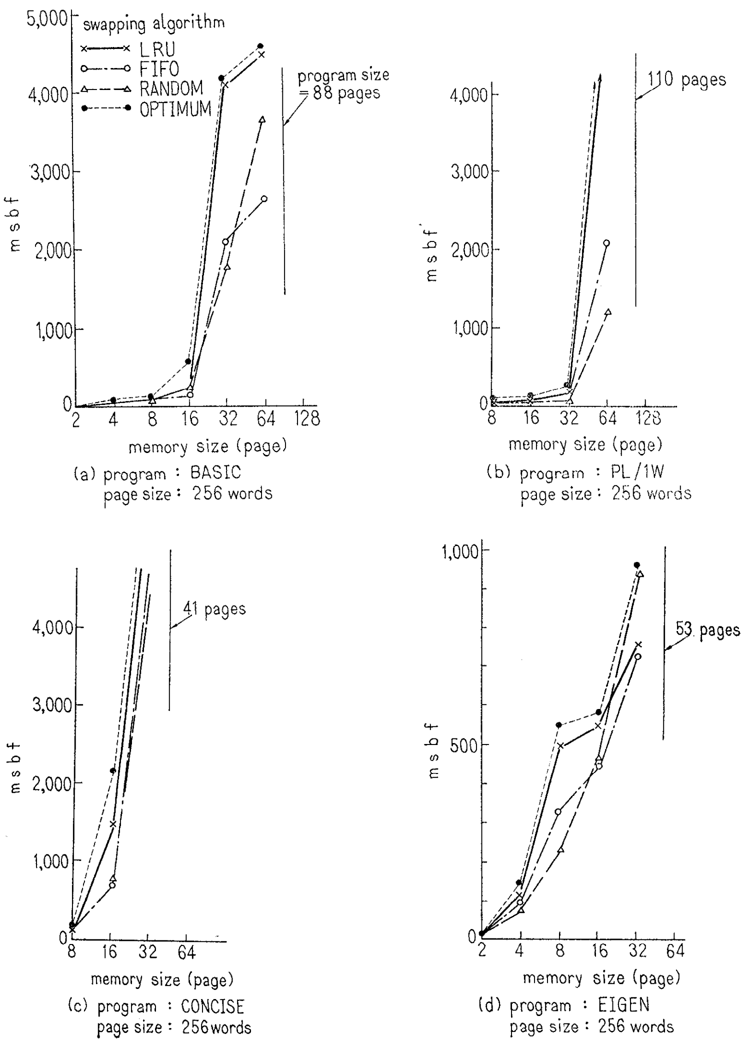

Fig. 1. msbf (mean executing steps between successive page faults) vs. memory size, page size and swapping algorithm.

algorithms, that the mean executing steps between successive page faults are enough large when the size of the virtual memory requirements is less than the twice of the real main memory size assigned for each program.

Table 2 compares the number of page faults for the four swapping algorithms. The values normalized by OPTIMUM algorithm are also showed. Sample program EIGEN shows quite different behaviors for swapping algorithms from other three programs. RANDOM algorithm is better than FIFO in case of EIGEN. It will be because EIGEN refers very frequently a large range of logical addresses rather at random. In EIGEN most of the page faults will occur at data-refer. FIFO or LRU does not show good effects in this case.

Table 2. Comparison of swapping algorithms

	page size (word)	memory size (page)	msbf						
			RANDOM (*R*)	FIFO (*F*)	LRU (*L*)	OPTI-MUM (*O*)	*R/O*	*F/O*	*L/O*
BASIC	64	64	351	920	920	984	0.36	0.94	0.94
	256	32	1,790	2,117	4,188	4,188	0.43	0.51	1.00
	1,024	16	5,282	8,643	17,286	17,286	0.31	0.50	1.00
PL/1W	64	256	1,079	1,891	2,584	2,645	0.41	0.72	0.98
	256	64	1,232	2,076	5,645	5,750	0.21	0.36	0.98
	1,024	32	6,129	26,667	26,667	29,792	0.21	0.89	0.89
CONCISE	64	64	1,940	2,554	6,315	6,520	0.30	0.39	0.97
	256	32	8,906	19,296	28,944	28,944	0.31	0.66	0.66
	1,024	16	297,050	297,050	297,050	297,050	1.00	1.00	1.00
EIGEN	64	128	396	227	337	425	0.93	0.53	0.79
	256	32	941	728	758	950	0.99	0.77	0.80
	1,024	8	1,133	1,386	2,065	2,120	0.53	0.65	0.97

4. *Relation of Pages used in the Past and in Future*

The degree of coincidence between the pages used in the past and the pages used in future was found. It will give the effective information for the pre-paging.

At some points $s_{(t)}$ of the program address trace, the following steps are executed.

(1) We find i-different pages $p_{1(t)}, p_{2(t)}, \cdots, p_{i(t)}$ by seeing backwards from $s_{(t)}$.

(2) We find j-different pages $f_{1(t)}, f_{2(t)}, \cdots, f_{j(t)}$ by seeing forewards from $s_{(t)}$.

(3) We find $e_{j(t)}$ which is the number of instructions between $s_{(t)}$ and the point of which $f_{j(t)}$ is refered.

(4) We find $a_{ij(t)}$ which is the number of the coincident pages between $(p_{1(t)}, p_{2(t)}, \cdots, p_{i(t)})$ and $(f_{1(t)}, f_{2(t)}, \cdots, f_{j(t)})$.

(5) We put $A_{(t)}=(a_{ij(t)})$

We found $A_{(t)}$ for $i, j=5, 10, 20, 40, 60$ and for $s_{(1)}=50{,}000$, $s_{(2)}=100{,}000$, $s_{(3)}=150{,}000$ and $s_{(4)}=200{,}000$. The following variables were calculated.

$$A=\frac{\sum_{t=1}^{4} A_{(t)}}{4}, \text{ where } \sum A_{(t)} \text{ means } (\sum_{t} a_{ij(t)}).$$

$$\bar{e}_j=\frac{\sum_{t=1}^{4} e_{j(t)}}{4}.$$

$$C=\frac{A}{j}\times 100, \text{ where } \frac{A}{j}\times 100 \text{ means } \left(\frac{a_{ij}}{j}\times 100\right).$$

Fig. 2 shows the results when the page size is 256 words. In Fig. 2, c in the

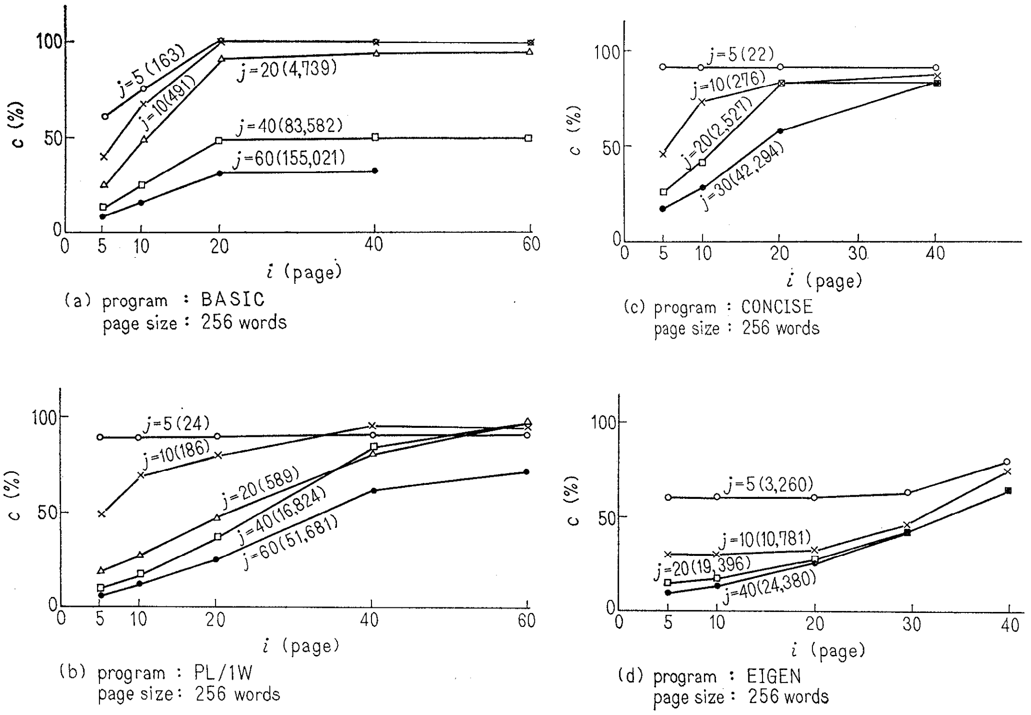

Fig. 2. The degree of coincidence between the pages used in the past and the pages used in future.

vertical axis is $\frac{a_{ij}}{j}\times 100$ for each (i, j). Some comments are given about Fig. 2. For example, in Fig. 2 (b), let us see the curve of $j=40$. Then PL/1W can be executed 16,824 instruction steps as a mean before it requires 40 pages. For $j=40$ and $i=40$, about 75% of 40 pages used in future from some point will be coincident with the 40 pages which have been used immediately before that point in the past. For $j=40$ and $i=60$, about 90% of the 40 pages used in future will be coincident with the 60 pages used in the past.

Coincidence coefficient c in Fig. 2 will converge to some constant value α as i increases. For the program which requires no new virtual pages from some point of execution, α will converge almost 100%. In case of BASIC, α converges to a rather small value because the compiling process of BASIC requires constantly new pages for the source program from a terminal and for the object code. In EIGEN c is very small when i is small because EIGEN refers data areas almost at random and there is little causal relation between the pages used in the near past and the pages used in the near future. LRU swapping algorithm does not show a good performance in such cases.

5. *Conclusion*

Four typical programs under H-5020TSS which has a segmentation and paging mechanism were traced for the analysis of swapping algorithms and program behaviors under a paging environment.

The following results were obtained.

(1) The relation among the swapping algorithm, the main memory size, the page size and the mean executing steps between successive page faults was obtained. A surprising result is that RANDOM swapping algorithm is better than FIFO for some program. It seems, regardless of swapping algorithms, that the mean executing steps between successive page faults are enough large when the size of virtual memory requirements is less than the twice of real main memory size assigned for each program.

(2) The degree of coincidence between the pages which have been used in the past and the pages which will be used in future was found. It will give the effective information for the pre-paging.

References

[1] T. Masuda, Y. Yoshizawa, T. Hirosawa and N. Takahashi: "System Data and Its Evaluation of Time Sharing System with Virtual Memory Concept", Proc. of the MEXICO 1971 International IEEE Conference, pp. 8-11, 1971.

[2] L. A. Belady: "A Study of Replacement Algorithms for a Virtual Storage Computer", *IBM System Journal*, Vol. 5, No. 2, pp. 78-101, 1966.

[3] P. J. Denning: "The Working Set Model for Program Behavior", *CACM*, Vol. 11, No. 5, pp. 323-333, 1968.