Memory Hierachies of a virtual Memory Computer

Testunori NISHIMOTO*

1. Preface

A time sharing system provides for simultaneous utilization of a computer by a number of users, increasing utilization efficiency. In the virtual memory computer system, the more tasks there are which run in parallel, the more the overhead required due to the shorter period between missing page faults.

This report illustrates the above mechanism with a model of the virtual memory computer and also clarifies how to design the system in order to minimize overhead, especially in regard to selection of main memory capacity, CPU speed, speed of auxiliary memory units (such as magnetic drums and/or magnetic disks), and the number of tasks.

2. Building a Time Sharing System Model

The overhead of the system may be divided into the following three components:

(1) CPU idle time, resulting in particular from time required to swap between the main memory and auxiliary memory units. (T1)

(2) Time required by the an operating system for controling tasks. (T2)

(3) CPU idle time caused in particular by the following factors. (T3)
The number of active tasks is 0.

In this report, OH_1 indicates the ratio of CPU idle time (as mentioned in (1)) to total overhead (except for the time required for task control as mentioned in (2)), while OH_2 indicates the ratio of idle time to total overhead, including the time required to run a program for correction of a missing page fault (interruption due to required page not being in main memory) and time required for running of trap drum channel (interruption originated by channel upon completion of one-page swap between secondary memory and main memory), but not including idle time

This paper first appeared in Japanese in Joho－Shori (Journal of the Information Processing Society of Japan), Vol. 14, No. 10 (1973), pp762～768.

* Central Research Laboratory, Hitachi Ltd.

mentioned in (1).

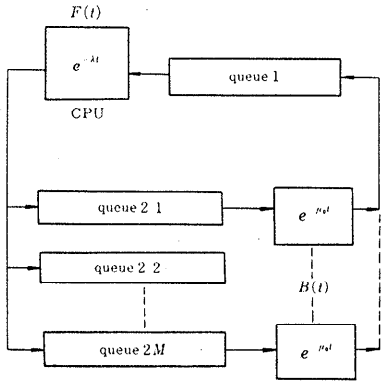

Fig. 3 Block diagram

Fig. 1 shows the transformation of task movement into a model. Each of the N tasks enters the box written as CPU, and it moves to either of Queue 21, Queue 22, ..., or Queue 2M in order to transfer pages of information from the secondary memory to the main memory whenever any information page not exsisting in the main memory is demanded while the tasks are being run. The aforementioned M, represents the number of secondary memories which can be independently operated.

Upon completion of the page swap from secondary memory to main memory, the task will moved into Queue 1. After all the preceding tasks in Queue 1 have been executed, the task will be performed in CPU.

Now, if the average distribution function of the period between each task swap from Queue 1 to CPU and each occurrence of missing page fault is assumed to be an exponetial distribution of $1/\lambda$; the average distribution function, $B(t)$, of the page swapping time from a unit of the secondary memory to the main memory is assumed to be an exponetial distribution of $1/\mu_0$; and, in movement of tasks from CPU to Queue 21, Queue 22, ..., Queue M, if the transferring probability is considered to be the same as far as the number of 2M queues concerned; then, OH_1 based on the above-mentioned model is expressed as follows,[1)2)]

$$OH1=\frac{\binom{N+M-1}{N}\left(\frac{\lambda}{\mu}\right)^N}{\sum_{l=0}^{N}\binom{l+M-1}{l}\left(\frac{\lambda}{\mu}\right)^l} \qquad (1)$$

where $\mu \equiv M \cdot \mu_0$

Analyses similar to equation (1) has been known from early days, but OH_1 was considered to be constant without relating to the number of tasks, N, and the capacity of the main memory, C, in those analyses. The problem was therefor how to express the fact that the main memory is being used in the model.

In order to furnish a solution, the author has tried to use λ as a function of N and C so that the capacity of the main memory can be taken into consideration. The function form of λ is obtained by means of a simulation and applied to the analysis.

If g is used for the average instruction execution time of CPU, α for the number of average collating times to the memories per instruction, C (page) for the main memory capacity, N for the number of tasks, and $E(C,N)$ for the probability per virtual storage reference that the referent address will not be included in a page already in main memory, then λ can be expressed by the following equation.

$$\lambda=\frac{\alpha}{g}(C,N) \qquad (3)$$

Since the available pages for a task becomes C/N, the following equation may be justified.

$$E(C,N)=E\left(\frac{C}{N},1\right) \qquad (4)$$

Now, it should be known which function will satisfy $E(C/N,1)$. If the capacity is represented as x page and the probability of the missing page fault as y, and if the missing page fault is assumed to be $y+dy$ when the main memory capacity is $x+dx$, then, considering that missing page fault will be reduced because pages not existing in x will be in section dx, the equation mentioned below should hold true. (Refer to Fig. 2)

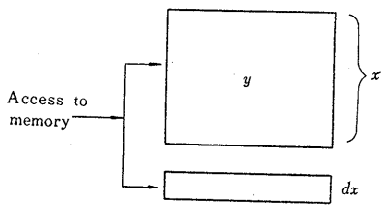

Fig. 4 Memory capacity and missing page fault

$$dy = -(\beta dx)$$

where β denotes the probability that one page will exsist in a main memory unit. If β is assumed to be a constant, the above equation is solved as follows,

$$y = A\cdot e^{-\beta x}$$

and $E(x,1)$ will be the exponetial function of x.

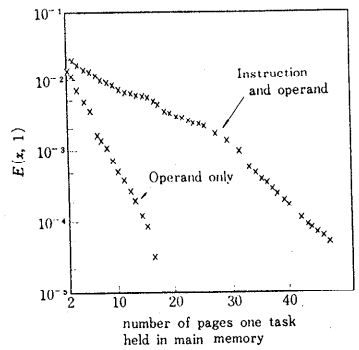

Fig. 5 Memory capacity (page) and swapping probability

The author has tried (in the case of the number of task = 1) to obtain $E(x,1)$ by means of adapting x from the magnetic tape in which the address pattern HITAC 5020 (when it is used as the virtual memory computer) is traced. (Refer to Fig. 3)

As seen in the above graph, both plottings give essentially straight lines. Accordingly, the function form of $E(x,1)$ can be expressed as follows,

$$E(x,1)=A\cdot e^{-\beta x} \qquad (5)$$

From equations (4) and (5), the following equation will be obtained.

$$E(C,N)=A\cdot e^{-B\frac{C}{N}} \tag{6}$$

And according to (3) and (6), λ will be expressed by the following equation.

$$\lambda=\frac{\alpha}{g}\cdot A\cdot e^{-B\frac{C}{N}} \tag{7}$$

If T is used as the swapping time of one page from a secondary memory to the main memory. μ0 can be expressed by T as follows,

$$\mu_0=\frac{1}{T} \tag{8}$$

In equation (1), Z is defined as follows,

$$Z\equiv\frac{\lambda}{\mu} \tag{9}$$

Therefore, OH_1 can be expressed as follows,

$$OH1=\frac{\binom{N+M-1}{N}Z^N}{\sum_{l=1}^{N}\binom{l+M-1}{l}Z^l} \tag{10}$$

From equations (7), (8), and (9), the following equation can be developed.

$$T=\frac{M\cdot g}{\alpha}\cdot\frac{Z}{A}\cdot e^{B\frac{C}{N}} \tag{11}$$

It is noted from equation (11) that Z is the function of C, T, N, and M, and it is also understood from (10) that OH_1 becomes the function of Z. On the contrary, however, solve equation (10) in respect to Z so that Z may be obtained as a function of OH_1, N, and M and substitute the result into equation (11), then, in the case of a constant OH_1, the relation among C, T, N, and M will be understood. And, through such relation, it can be said from equation (11) that, when the overhead and the number of tasks are kept constant, the page swapping time, T, is the one given as an exponential function of the number of pages, C, in the main memory.

As a next step, the author has conducted the following attempt to obtain OH_2.

To begin with, "Tos" used in the following discussion is defined to be the total of steps required for one run on each program of processing the missing page fault and the trap drum channel. The average time between the moment when tasks move into the CPU from Queue 1 and an occurrence of the mapping fault is $1/\lambda$, that is,

. Accordingly, OH_2 can be expressed by the following equation.

$$OH2=\frac{g\cdot Tos}{\frac{g}{\alpha\cdot A}\cdot e^{B\frac{C}{N}}+g\cdot Tos}=\frac{1}{\frac{1}{\alpha\cdot Tos\cdot A}e^{B\frac{C}{N}}+1} \tag{12}$$

Equation (12) can be expressed in terms of Tos,

$$Tos=\frac{OH2}{\alpha\cdot A\cdot(1-OH2)}\cdot e^{B\frac{C}{N}} \tag{13}$$

3. Results and Their Evaluations

An attempt to obtain a function having the form $y=A\cdot\exp(-Bx)$ and the most approximate value is tried using a polynomial regression equation in order to approximate plotted point in Fig. 3 by means of strait lines. A and B of $E(x,1)$ for the address pattern of accesses to all the main memories obtained from this approximation is as follows.

$$A = 0.0416 \qquad (14)$$

$$B = 0.131 \qquad (15)$$

The effects on λ (that is, OH_1 according to the equations (10) and (11) due to the number of tasks, N, and the number of pages, C, in the main memory, using A and B above is exhibited below.

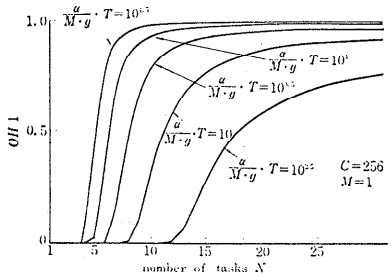

Fig. 4 Relation between the number of tasks and $OH1$

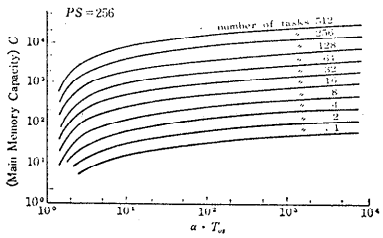

Fig. 5 Main memory capacity and relative secodndary memory speed at $OH1=0.05$

Fig. 4 shows the number of tasks, N, as the axis of abscissa and the number of pages, C, as the axis of ordinate, applying $C=256$ and $a/M\cdot T/g$ as a parameter.

Fig. 5 shows the contour lines for number of tasks when $OH_1=0.05$. That is to say, the graph exhibits the contour lines of N at the upper limits within which OH_1 is acceptable. The axis of abscissa represents $a/M\cdot T/g$ — the parameter obtained through dividing the time required for one-page swap to the secondary memory by CPU speed and the axis of ordinate represents C — the capacity of the main memory expressed by a unit of page (1 page = 256 words).

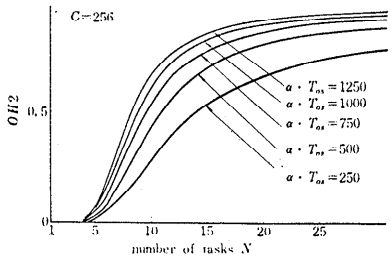

Fig. 6 Relation between the number of tasks and $OH2$

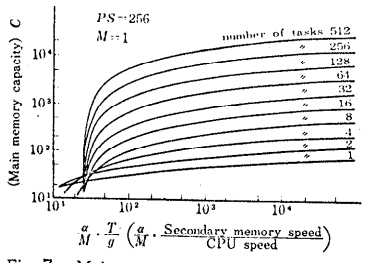

Fig. 7 Main memory capacity and T_{os} at $OH2=0.05$

Fig. 6 shows OH_2 as the axis of ordinate and the number of tasks, N, as the axis of abscissa, where C=256 words. Through the graph, it is well understood that OH_2 abruptly increases at a certain point with the larger number of tasks.

Fig. 7 shows the capacity of the main memory, C, as the axis of ordinate and $a \cdot Tos$ as the axis of abscissa, exhibiting the contour iso-lines of N when OH_2 attains 0.05.

4. Conclusion

In this report, the following has been discussed.

1. To divide the overall overhead of the time sharing system for which a virtual memory computer is employed into two types such as OH_1 and OH_2.
2. OH_1 is related to the idle time of CPU, and OH_2, which occupies the most part of the total overhead related to the running time on the operating system, is the time required for runnings on the programs of processing the missing page fault and the trap drum channel.
3. In respect to OH_2, the probability that the missing page fault will occur is approximated by an exponential function of the capacity in the main memory which holds one task.
4. Based on the assumption that the pages of the main memory are equally distributed to each task, analyses of OH_2 are conducted.

As a result, the following points are disclosed.

(1) If the number of multiple-running tasks increases, the overhead will be enlarged because the capacity of the main memory assigned to one task decreases and more swappings with the secondary memory is required.

(2) The system having a larger overhead, throughput will not be improved even when CPU running speed increases. In such system, either of the following remedies should be applied for improvement of throughput.

a) To increase running speed of the secondary memory.

b) To enlarge the capacity of main memory.

c) To minimize the number of tasks so that the overhead may decrease.

References

1) Tanaka: Analysis of Multiple Programing System by means of Cyclic Queue; The Trans. of The Institute of Electronics and Communication Engineers of Japan, Vol. 53-C, No. 2, pp. 57-64 (Jan., 1970).

2) Encyclopedia of Queuing Applications. (Published by Hirokawa Book, Ltd.)