T. Kaneko

Optimal Task Switching Policy for a Multilevel Storage System

Abstract: Capacity demands for computer memory are increasing. A multilevel storage system provides an economically feasible solution without seriously affecting the total response time. An M-level storage system is considered in this paper. The capability of a digital computer with a multilevel storage system is best enhanced in a multiprogramming environment. In a high level storage system, determination of a best task switching policy becomes an important consideration. In this paper a queuing network is introduced to describe distribution and flow of tasks in the system. An optimal switching policy is determined in relation to the system's overhead time. It is shown that in heavily CPU-limited cases the determination becomes a very simple one; namely, the best policy is given as the threshold level at which the accumulation of the average access time exceeds the overhead time.

Introduction

The trend in the design of computer storage systems is toward multilevel storage hierarchies. The first level or cache has a small capacity but a very low access time, which can be realized with expensive technology. Storage devices at the lower levels are realized with slower and less expensive technology and have larger capacities.

Our basic assumption, in this paper, is that with the development of new technologies we may reach a situation for which the CPU might become the bottleneck of the system. In such a case, an optimal task switching policy is needed to obtain maximum throughput.

We assume that only one switching policy exists in a two-level storage system and that there are two policies in a three-level storage system: The execution will be switched to another task only when 1) there is a page fault in the first level or 2) the referenced page is not found in either the first or the second level. We call the former the first policy and the latter the second policy. In the past, attention has been limited to the first policy only. Let us consider an M-level storage system as shown in Fig. 1. There are $M-1$ task switching policies in general. The mth policy in an M-level storage system is defined as follows: The present task is executed without interruption as long as the CPU references pages that are found within the first m levels, and the execution is switched to the next task only if the CPU requests a page that is found in one of the remaining $M-m$ levels.

The purpose of this paper is to analyze an M-level storage system in a multiprogramming environment and to search for an optimal task switching policy related to the system's overhead time. We will consider a multilevel linear storage system in which data transfers take place only between adjacent levels as indicated by the arrows in Fig. 1, although other types of data transfer are conceivable [1].

Each level is usually divided into a set of smaller units called "pages" or "page frames" [2, 3]; a page is also a unit of data transferred from one level to the next level. Careful consideration is needed in order to determine the suitable page size at each level [2]. Here we assume that the page size at each level is already determined and that its effect is included in the access times $\{t_i\}$. The hit ratio or the probability that a referenced page is found in level i is assumed to be given as $\{p_i\}$, where $i=1, 2, \cdots, M$ [3, 4].

In the past, performance evaluation for a multiprogramming environment has been carried out mostly for a two-level storage system [5, 6]. Quite recently there have appeared several works on the design of multilevel storage systems [3, 4, 7–9] that investigate technologies and capacities at each level of the storage hierarchy. To the best of the author's knowledge, however, there is no work which pays specific attention to a task switching policy for a multilevel storage system.

System overhead time

In the determination of an optimal task switching policy, the system overhead time plays a major role. Lewis and Shedler [10] studied the effect of the overhead in a two-level storage system. They pointed out four services as system overhead functions: 1) service for picking up the

next program for processing and restoring the machine state; 2) service for saving the machine state of the (present) program relinquishing the CPU, executing the replacement algorithm, constructing the channel control program for the required page, and placing an entry into the paging queue; 3) service for picking up the next page request and executing the channel control program; and 4) service for placing a new entry in the CPU queue. The first and second are concerned with service between the CPU and the first level of storage, and the last two are concerned with service between the first and the second levels. The major overhead activity is represented by the second service.

In a multilevel storage system with more than two levels, additional services are needed, e.g., the administration of page directories (if a page replacement algorithm such as the "least recently used" is invoked) and of the queues that transfer the data from one level to another. Comparing the mth and $(m+1)$th policies in an M-level storage system, the number of page directories is M for both but the number of transferring queues is decreased by one for the latter.

When the CPU is used to govern all the services, a very complex problem arises in establishing the CPU's priority rules [10]. Since we analyze a relatively high level storage system (say more than four levels), we assume here that dedicated special hardware is employed to carry out most of the services, so that the services assigned to the CPU are minimal.

Let us denote by $B_{m,M}$ the average overhead time in an M-level storage system when the mth task switching policy is used. Our problem is to determine the optimal policy under given overhead times $\{B_{i,M}\}$ and a fixed number of programs in an M-level storage system characterized with access times $\{t_i\}$ and hit ratios $\{p_i\}$. The criterion for optimality considered here is to increase the throughput or the number of useful instructions executed per unit of time.

Queuing model

In order to derive the expression for throughput we employ a queuing network model [3, 10–12]. To apply a queuing network model to an M-level storage system, the following assumptions are made;

1. The priority rule for service to the queue is First-Come-First-Served (FCFS).
2. The probability distribution for the service time at each server is negative exponential, i.e.

$$P(y < y_0) = 1 - \exp(-y_0/t), \tag{1}$$

where t is the average service time.

3. The pages at each level are referenced independently; p_i is the probability that a referenced page will be found in the ith level; every program is of an identical class. Note that $\Sigma_{i=1}^{M} p_i = 1$.
4. The data transfer from the $i+1$ level to the i level depends on the availability of a vacant slot. If there is no vacant slot in the ith level, a page is removed from it under the invoked page replacement algorithm.
5. When an mth task switching policy is employed in an M-level system, a task that is found at one of the $(M-m)$ slower levels migrates to the $m+1$ level, then further to the m level, and eventually to the first level. When such a transfer is taking place at one of the faster m levels, there is a chance of a conflict between the CPU requests and the task migration. In such a case the priority is given to the CPU. The resultant additional delay time to the task is ignored, however, because the probability of such a conflict is small and because the delay time is insignificant compared with the long waiting time at the CPU server in a CPU-limited system. (The term "CPU-limited" is explained in the following section.) The average transfer time from the mth level to the first is denoted by $C_{m,M}$. We have, in general,

$$C_{m,M} = t_2 + t_3 + \cdots + t_m. \tag{2}$$

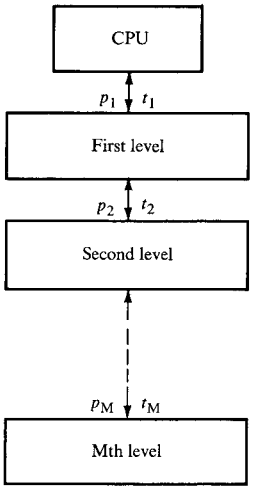

Figure 1 An M-level storage system.

Under the above assumptions we first obtain a queuing network for the mth task switching policy in an M-level storage system $(1 \leq m \leq M-1)$ and then derive an expression for the throughput by using Gordon and Newell's result [12].

Since there is no task switching as long as the CPU requests pages within the faster m levels, there are no queues for them. Therefore it is convenient to combine them into one server which we call the CPU server. Now let us compute its average service time.

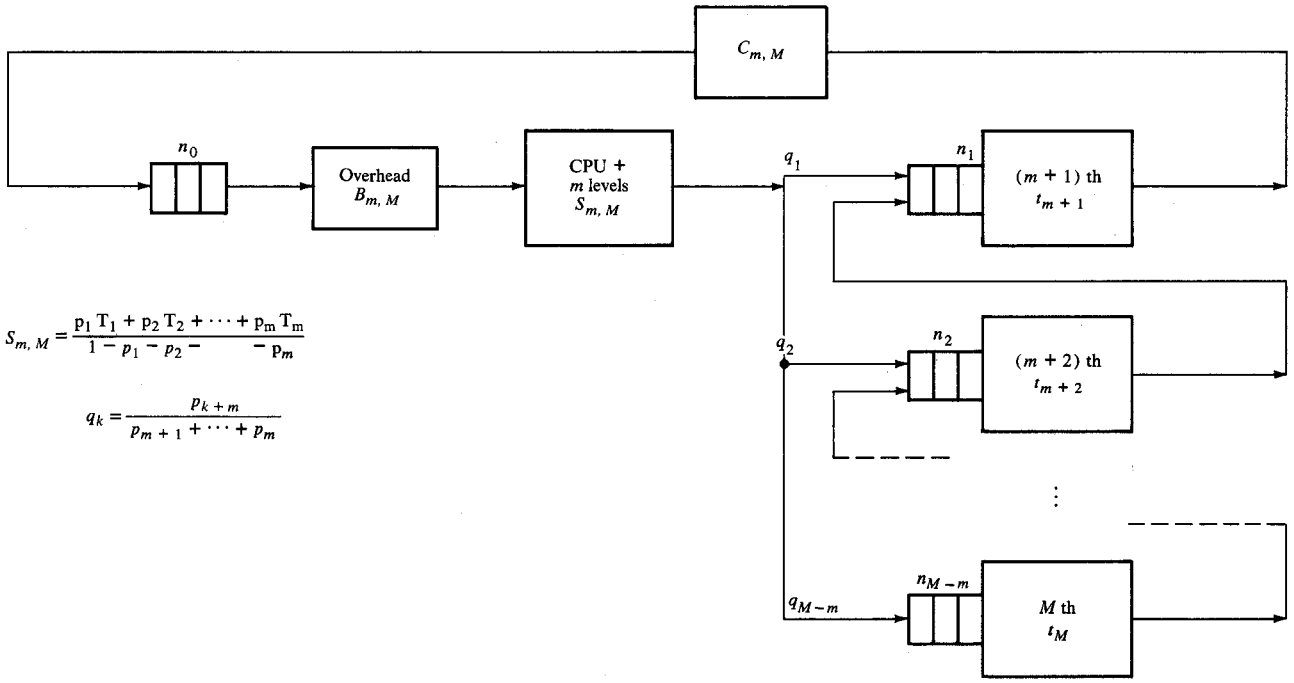

Figure 2 A queuing model for an M-level storage system when the mth task switching policy is used.

By T_i we denote the access time from the CPU to the ith level. Therefore we have

$$T_i = t_0 + \sum_{j=1}^{i} t_j, \tag{3}$$

where t_0 is the average execution time at the CPU. The probability that the CPU will call the first level k_1 times, the second k_2 times, $\cdots$, and the mth level k_m times before switching to another task due to a page fault in the first m levels is

$$f = \frac{(k_1 + k_2 + \cdots + k_m)!}{k_1! k_2! \cdots k_m!} {p_1}^{k_1} {p_2}^{k_2} \cdots {p_m}^{k_m} \times (1 - p_1 - p_2 - \cdots - p_m). \tag{4}$$

The average service time is

$$S_{m,M} = \sum_{k_1=0}^{\infty} \cdots \sum_{k_m=0}^{\infty} (k_1 T_1 + \cdots + k_m T_m) f.$$

After considerable algebraic manipulation we obtain

$$S_{m,M} = \frac{p_1 T_1 + p_2 T_2 + \cdots + p_m T_m}{1 - p_1 - p_2 - \cdots - p_m}. \tag{5}$$

The queuing network model used in our investigation is shown in Fig. 2. We define the state of the queuing network as the number of tasks in the queues at the servers, namely $(n_0, n_1, \cdots, n_J)$ where $J = M - m$ and n_0 is the number of tasks at the CPU server and $n_i (i \geq 1)$ is that at the $m + i$ level. From Gordon and Newell's paper we obtain the probability of the state $(n_0, n_1, \cdots, n_J)$ as

$$P(n_0, n_1, \cdots, n_J) = {d_1}^{n_1} {d_2}^{n_2} \cdots {d_J}^{n_J} / G_{m,M}, \tag{6}$$

where

$$d_1 = \frac{t_{m+1} + C_{m,M}}{B_{m,M} + S_{m,M}},$$

$$d_k = \frac{r_k t_k}{B_{m,M} + S_{m,M}},$$

$$r_k = \frac{\sum_{j=k+m}^{M} p_j}{\sum_{j=m+1}^{M} p_j} \text{ for } 2 \leq k \leq J, \text{ and} \tag{7}$$

$$n_0 + \sum_{j=1}^{J} n_j = N.$$

The normalization factor $G_{m,M}$ is calculated as

$$G_{m,M} = \sum_{j=1}^{J} \frac{{d_j}^{J-1}}{\prod_{\substack{k=1 \\ k \neq j}}^{J} (d_j - d_k)} \left(\frac{1 - {d_j}^{N+1}}{1 - d_j} \right). \tag{8}$$

The probability that the CPU is idle is

$$Q_{m,M} = \frac{1}{G_{m,M}} \cdot \sum_{j=1}^{J} \frac{{d_j}^{N+J-1}}{\prod_{\substack{k=1 \\ j \neq k}}^{J} (d_j - d_k)}. \tag{9}$$

The throughput is

$$R_{m,M} = \left(1 - \sum_{j=1}^{m} p_j\right)^{-1} \left(\frac{1 - Q_{m,M}}{B_{m,M} + S_{m,M}}\right). \tag{10}$$

The derivations of (8) and (9) are omitted here, but we check their validity by obtaining $G_{2,3}$ and $Q_{2,3}$:

$$G_{2,3} = \frac{d_1}{d_1 - d_2} \frac{1 - d_1^{N+1}}{1 - d_1} + \frac{d_2}{d_2 - d_1} \frac{1 - d_2^{N+1}}{1 - d_2}; \tag{11}$$

$$Q_{2,3} = \frac{1}{G_{2,3}} \left(\frac{d_1^N}{d_1 - d_2} + \frac{d_2^N}{d_2 - d_1}\right). \tag{12}$$

The correctness of the above two results can be checked by calculating them directly from (6).

Comparison of task switching policies

We have calculated the throughput for an M-level storage system. Therefore, comparison can be readily carried out by evaluating the expression for a given hit ratio $\{p_i\}$, access or service time $\{t_i\}$, CPU overhead time $\{B_{i,M}\}$, and number of tasks N. When d_k in (7) is less than unity for every k, a long waiting line will probably be formed at the CPU server and we will say that the system is "CPU-limited." If d_k is larger than unity for some k, a long waiting line is formed at the $m + k$ level and the system is said to be "storage-limited" (or the $m + k$ level is the "bottleneck"). Since in a storage-limited system the throughput is mostly determined by the flow rate in the storage hierarchy, little effect on throughput is expected with different task switching policies. In other words, the study of optimal task switching policies is meaningful only for a CPU-limited system. Therefore, we are concerned only with this case.

Sekino [6], Wallace and Mason [13], as well as others, have studied the effect that the number of tasks has on the throughput in a two-level storage system. A large N will cause an undesirably long delay in program completion; moreover, the number of pages in the first level allotted to a particular program decreases as N increases. Therefore, the hit ratio p_1 is a decreasing function of N. Since $(1 - Q_{m,M})$ is an increasing function in N, there is an optimal N that produces the maximum throughput. In this paper, we assume that a reasonably large N is chosen so that the approximation $Q_{m,M} \ll 1$ holds reasonably well.

Using (10) and the approximation $Q_{m,M} \ll 1$, the difference in the throughput between the mth and $(m+1)$th task switching policy is

$$R_{m+1,M} - R_{m,M} = \left[\left(1 - \sum_{j=1}^{m+1} p_j\right)\left(B_{m+1,M} + S_{m+1,M}\right)\right]^{-1} - \left[\left(1 - \sum_{j=1}^{m} p_j\right)\left(B_{m,M} + S_{m,M}\right)\right]^{-1} \tag{13}$$

or

$$R_{m+1,M} - R_{m,M} = \left[\sum_{i=1}^{m+1} p_i T_i + B_{m+1,M} \sum_{j=m+2}^{M} p_j\right]^{-1} - \left[\sum_{i=1}^{m} p_i T_i + B_{m,M} \sum_{j=m+1}^{M} p_j\right]^{-1}. \tag{13'}$$

The polarity depends upon

$$\begin{aligned} A &= \sum_{i=1}^{m} p_i T_i + B_{m,M} \sum_{j=m+1}^{M} p_j - \sum_{i=1}^{m+1} p_i T_i - B_{m+1,M} \sum_{j=m+2}^{m} p_j \\ &= B_{m,M} \sum_{j=m+1}^{M} p_j - B_{m+1,M} \sum_{j=m+2}^{M} p_j - p_{m+1} T_{m+1} \\ &= p_{m+1}(B - T_{m+1}) + b_{m,M} \sum_{j=m+1}^{M} p_j - b_{m+1,M} \sum_{j+m+2}^{M} p_j, \end{aligned} \tag{14}$$

where $B_{i,M} = B + b_{i,M}$. In general an optimal m is found by computing the above. However, if $b_{i,M} \ll B$ for every i (i.e., the overhead time is nearly constant), the polarity depends only upon $(B - T_{m+1})$. Therefore, if we define an m' such that

$$T_{m'} < B < T_{m'+1}, \tag{15}$$

then we have

$$R_{1,M} < \cdots < R_{m'-1,M} < R_{m',M} > R_{m'+1,M} > \cdots R_{M-1,M}. \tag{16}$$

Therefore the optimal task switching is the m'th policy.

Discussion

Let us assume that we investigate the case where M may have the value four, five or six. In this case the choice of a task switching policy is generally between the first and the second. We will investigate these two specifically.

Assuming no change in the overhead time with the two task switching policies, we know that the optimal switching policy is the first one if the average overhead time B is less than T_2. When $B = T_2$, the two policies yield the same throughput. If B is increased further, the second will yield a larger throughput. When $B = T_3$, we have the relationship

$$\frac{R_{2,M} - R_{1,M}}{R_{2,M}} = \frac{p_2(T_3 - T_2)}{(1 - p_1)T_3 + p_1 T_1}. \tag{17}$$

Since usually $T_2/T_3 \ll 1$ and $p_1 T_1/(1 - p_1)T_3 \ll 1$, (17) becomes

$$\frac{R_{2,M} - R_{1,M}}{R_{2,M}} = \frac{p_2}{1 - p_1}\left(1 + \frac{p_1}{1 - p_1} \cdot \frac{T_1}{T_3}\right)^{-1} \approx \frac{p_2}{1 - p_1}, \tag{18}$$

which can be close to 1. In other words a wrong switching policy can reduce the throughput significantly.

One may be very eager to pay attention only to the first task switching policy because it is the simplest. In this case the paging overhead time should be less than the

access time to the second level. If this is impossible, the use of the second task switching policy is advisable.

Example

As an example, we take a four-level storage system specified by $t_1 = 0.05\mu s$, $t_2 = 1\mu s$, $t_3 = 15\mu s$, $t_4 = 100\mu s$, $p_1 = 0.9$, $p_2 = 0.09$, $p_3 = 0.009$, $p_4 = 0.001$. For convenience we set $t_0 = 0$. We study the system performance for the fixed overhead time $B = 5\mu s$, although it may be artificially large. By applying the first switching policy, we can compute the average service time for the CPU server from (5). That is,

$$S_{1,4} = \left(\frac{0.9 \times 0.05}{1 - 0.9}\right)\mu s = 0.45\ \mu s.$$

The values of d_k's which are significant in determining the asymptotic system performance are computed from (7):

$$d_1 = \frac{1}{5 + 0.45};$$

$$d_2 = \frac{0.1 \times 15}{5.45} = \frac{1.5}{5.45};$$

$$d_3 = \frac{0.01 \times 100}{54.5} = \frac{10}{54.5}.$$

Since d_1, d_2 and d_3 are in such a range that d_1^N, d_2^N and d_3^N are by far less than unity for reasonably large N, the system is heavily CPU-limited. The asymptotic throughput is obtained from (10) by setting $Q_{m,M} = 0$:

$$R_{1,4} = \frac{1}{(1 - 0.9)(5 + 0.45)} \times 10^6$$

$$= \frac{1}{5.45} \times 10^7 \text{ instructions/second.}$$

When the second switching policy is applied, we have

$$S_{2,4} = \frac{0.9 \times .05 + 0.09 \times 1.05}{1 - 0.9 - 0.09}\ \mu s = 13.95\mu s,$$

and

$$d_1 = \frac{16}{5 + 13.95};\ d_2 = \frac{10}{5 + 13.95}.$$

Again d_1 and d_2 are less than unity. The asymptotic throughput is given by

$$R_{2,4} = \frac{1}{0.01 \times (5 + 13.95)} \times 10^6$$

$$= \frac{1}{1.895} \times 10^7 \text{ instructions/second.}$$

In case the third switching policy is applied, we have

$$S_{3,4} = \frac{0.9 \times 0.05 + 0.09 \times 1.05 + 0.009 \times 16.05}{1 - 0.9 - 0.09 - 0.009}$$

$$= 284\mu s;$$

$$d_1 = \frac{116}{5 + 284}.$$

Again d_1 is less than unity. The asymptotic throughput is

$$R_{3,4} = \frac{1}{0.001 \times (5 + 284)} \times 10^6$$

$$= \frac{1}{2.89} \times 10^7 \text{ instructions/second.}$$

As expected, the second switching policy yields the maximum throughput, approximately three times more than the first policy and 50 percent more than the third policy.

Comments

Some comments are in order concerning the drawbacks of the model employed in the determination of an optimal switching policy for a multilevel storage system in a multiprogramming environment. As stated at the beginning, it was assumed that the references made by the CPU are independent, stationary, and of one class. The stationary assumption is a necessary one for such statistical analysis. Regarding program classes, the choice of a task switching policy is independent of the hit ratios $\{p_i\}$ as long as the overhead time does not differ much for different switching policies. It may be conjectured that the result of this paper would be valid for a mixture of different program classes. The independence property might be replaced by introducing a Markovian dependency as presented by Shedler and Tung [14]. The first-order Markov dependence implies that the address at time n depends only upon the address of the call at its prior time $n - 1$. This assumption will be more realistic but the analysis will be more complex.

The negative exponential assumption is another drawback of the present model. The assumption is relatively good for random access storage devices but is not acceptable for sequential accesses. A departure from the negative exponential assumption will make the analysis very difficult.

In conclusion, we feel that in spite of the simple model used, the results obtained in this paper provide a meaningful basis for choosing an optimal switching policy in a multiprogrammed multilevel storage system.

Acknowledgment

The author thanks C. K. Chow for suggesting the investigation of multilevel storage systems and W. D. Frazer for his valuable comments.

References

1. E. Morenoff and J. B. McLean, "Application of Level Changing to a Multilevel Storage Organization," *Comm. ACM* **10,** 149 (1967).
2. P. J. Denning, "Virtual Memory," *Computing Surveys* **2,** 153 (1970).

3. I. L. Traiger and R. L. Mattson, "The Evaluation and Selection of Technologies for Computer Storage Systems," Research Report RJ967. IBM Research Laboratory, Monterey and Cottle Roads, San Jose, California, February 1972.
4. R. L. Mattson, J. Gecsei, D. R. Slutz and I. L. Traiger, "Evaluation Techniques for Storage Hierarchies," *IBM Systems Journal* **9,** 78 (1970).
5. J. L. Smith, "Multiprogramming under a Page on Demand Strategy," *Comm. ACM* **10,** 636 (1967).
6. A. Sekino, "Performance Evaluation of Multiprogrammed Time-shared Computer Systems," Massachusetts Institute of Technology, Project MAC, TR-103, Cambridge, Sept. 1972.
7. Y. S. Lin and R. L. Mattson, "Cost-Performance Evaluation of Memory Hierarchies," Research Report RC3781, IBM Thomas J. Watson Research Center, Yorktown Heights, N.Y. 10598, March 1972.
8. C. V. Ramamoorthy and K. M. Chandy, "Optimization of Memory Hierarchies in Multiprogramming," *J. ACM* **17,** 426 (1970).
9. C. K. Chow, "Optimization of Storage Hierarchies," *IBM J. Res. Develop.* **18,** 194 (1974).
10. P. A. W. Lewis and G. S. Shelder, "A Cyclic-Queue Model of System Overhead in Multiprogrammed Computer Systems," *J. ACM* **18,** 199 (1971).
11. D. P. Gaver, "Probability Models for Multiprogramming Computer Systems," *J. ACM* **14,** 423 (1967).
12. W. J. Gordon and G. F. Newell, "Closed Queueing Systems with Exponential Servers," *Oper. Res.* **15,** 254 (1967).
13. V. L. Wallace and D. L. Mason, "Degree of Multiprogramming in Page-on-Demand Systems," *Comm. ACM* **12,** 305, 318 (1969).
14. G. S. Shedler and C. Tung, "Locality in Page Reference Strings," Research Report RJ932, IBM Research Laboratory, Monterey and Cottle Roads, San Jose, California, October 1971.

Received March 6, 1974

The author is located at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York 10598.