Optimization Controls of Computer Systems — Particularly on the Control of a TSS —

SETSUO OSUGA*

Introduction

One of the recent major problems concerning to the computer systems with wide spread use of on-line systems is how to operate the system more efficiently. To achieve the efficient use of it, a particular program system called optimization control system becomes necessary. This program system looks upon the system including optimization control system itself as an object and controls it. Such a program system must not only be able to control overall system so as to prevent any loss but also be simple enough because the overhead must be small. Thus the problem lies in finding of the optimal control algorithm and the program itself will take only a few part of a system program.

1. *Optimization control of computer systems*

1.1. *The concept of optimization of computer systems*

For optimizing a system, an appropriate real valued function of the system performance called an optimality criterion which measures the effectiveness of the system control must be defined. Generally it takes a functional form like $I=\int_0^T Qdt$. Fundamental concepts of optimization control of a computer system for itself do not differ from that of any other systems except that, in the former case the controlling and controlled part being same, the cost for the control (overhead) and the reward obtained (decrease of loss) have equivalent but opposite effects on the optimality criterion. This fact tells us that the control program must be powerful enough to achieve the optimization of a complex system on the one hand but as simple as possible on the other hand, and the optimization control should be defined at a point where the increase of the cost and the decrease of the loss are balanced.

1.2. *Modelling of the systems—the problem of finding a policy in stochastic decision processes with rewards*

A computer system usually consists of a number of devices or subsystems and there are many user's processes in the system in a variety of processing stages. These user's states as a whole represent a dynamically and randomly changing state of the system. Unless there are infinitely growing components in the system, this represents a stochastic process with finite states.

This paper first appeared in Japanese in Joho Shori (the Journal of the Information Processing Society of Japan), Vol. 9, No. 2 (1968), pp. 88-97.

* Institute of Space and Aeronautical Science, University of Tokyo.

When the system stays in a state i for an interval of dt, we suppose the system contributes an amount of $Q_i dt$ to the optimality criterion. Then with transition, there occurs a sequence of values $Q_i dt$ ($i=1, 2, 3, \ldots N$, where N is the number of states) that will be summed up to indicate a primary criterion. If the probabilities of the state transition are given, the mathematical expectation of this quantity can be found. Meanwhile, suppose that when the system is in a state i control program can choose one of m_i alternative transition vectors. Then a policy consists of the selection of one alternative in each state so as to optimize the system. Thus, the role of the control program is to detect the current state of the system and to find a specific policy corresponding to this detected state or the history of the states. The correspondence between the policy and the state that would be obtained by solving the problem of finding the optimal policy analytically or using simulations or any other methods, can be taught to the control program in the proper form in advance.

1.3. *Problems on programming*

There is no systematic way to obtain the best control program, and system designer's experiences, intuitions and abilities for system analysis and synthesis are always required. There are several problems to be considered, of which three most inportant are shown;

(1) How to compose the configuration of an operational system or a traffic control mechanism in the system, that is, how to arrange the queues for the processes, how to relate them each other, how to process them and so on. This establishment of the configuration of the system defines possible states of the system. Though there is no systematic way to achieve this, it seems useful to take the following step; (a) The least essential components and their relations to the traffic control, this is the on-demand method without the active control, are first established. (b) Then investigate what kind of features are effective to utilize the more elaborate informations and achieve the optimal control. With these steps, a special problem of the optimization control can be seperated from that of the basic traffic control.

(2) How to express and store the predetermined relations between the states and the policies so that the control program can find the optimal policy easily. The one to one correspondence table is not practical because the number of possible states of the system may be too large to find out one item from them rapidly and to store it in the computer memory. It is desirable that the amount of the memory space required to express this relation is as large as that proportional to the logarithm of the total number of possible states.

The procedure to get the optimal policy from this relational expression must be also as simple as possible. For example, if an appropriate simple function between state indices (S_i) and policy indices (P_i) can be found as $P_i=f(S_i)$, only that the control program must do is the calculation of the value of this function with the detected state index as the argument, and then the program becomes quite straightforward.

(3) How to make the program. In addition to prcccding problems, the control program must preferably be made independent of its control algorithm because (a) the establishment of this relationship always takes considerable times and the system developement will be subjected to undue delay if it must wait for the result and (b) the optimal policy must adapt to

the changes of the environment easily and preferably independently of the procedure of the control program.

This is a kind of the requirement to generalize the control program opposing to the requirement for the simplification since it can be achieved at the sacrifice of the generality. So there needs compromise though we must try to find the better method; for example, we may be able to introduce a decision function $P_i=f(S_i)$ so that the parameters included would be such that can be adjusted to adapt to the changes in the wide range without changing the form of the function.

As a conclusion, we can say that the features of the ideal control program will be as follows; (1) It is composed of a simple procedure and a few data or parameter words, (2) the optimal policy can be found using only these data from the state of the system quickly and (3) the control characteristics can be changed widely and continuously by changing only these data words.

2. *An optimal control of a time sharing system provided with a two-dimensional addressing system*

As a specific example of the above-mentioned general procedure to obtain the optimal control, we show here a case of the developement of a control program in a large scale TSS with two-dimensional addressing system called HITAC 5020 TSS.

2.1. *Outline of the system*

The one of the most novel fertures in this system is in the two-dimensional addressing that allows each user to write programs as though there is a virtual memory system of a large size. The concept of the segmentation and paging is introduced. A segment is an ordered collection of words with the associated segment name, in which a particular word is selected by the word address. The pair of this segment name and the word address is mapped into a physical location by a mapping hardware called Dynamic Address Translator (DAT). These segments which can vary in length are paged with 256-word and this allows flexible dynamic memory allocation. A descriptor is used to define and locate in a physical memory either a segment or a page, called a segment descriptor or a page descriptor respectively.

These descriptors are collected into a segment descriptor table (SDT) and a location page descriptor table (LPDT) respectively. These tables are called Mapping Tables. A SDT itself is a segment and is subjected to paging. Then there is a page descriptor table to locate each SDT page and this descriptor is collected to form the segment page descriptor table (SPDT).

A particular hardware register called Descriptor Base Register (DBR) is provided that points the location of the SPDT, specifying the base point of the user's name space. Switching of the CPU's control between processes can be achieved by replacing the content of the DBR. Fig. 1 illustrates these relations.

To reduce a number of memory cycles, a few associative registers are incorporated in the processor. When referred page is not in main memory, missing page fault occurs. There are many other features in the system such as a memory protection method, a dynamic linking, a transmission control, a file system etc.; however, they are not described here, because they

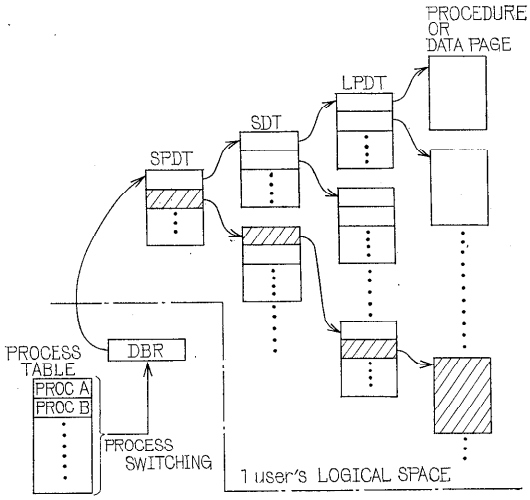


Fig. 1. Structure of a user's name space.

have no direct concerns with what described below.

2. 2. *The optimal control of TSS*

2. 2. 1. *A possibility of the optimal control*

Can we find any control algorithm to optimize a TSS?

Before answering to this, we reconsider the schenduling method in multi-programming wherein there are many user's processes concurrently in various processing stages. They receive one after another the CPU's service for a predetermined time between the slice-time-over interruptions. In the following, we assume for a while that to get the maximum utility of the system is most important. Every process is devided into either (1) what is ready for receiving CPU's service or (2) what is waiting to be waked up by any other process. The former is said "ready" state while the latter is "blocked" state. The one that is currently receiving the CPU's service is said in the "running" state, and included in the ready state.

As only a part of the programs can reside in the main memory, the programs must be swapped frequenty. The conventional scheduling methods are either (1) the one that reloads all programs and the data of one job into the main memory directly before the running (before the reload, purging must be done) or (2) the one that requires the swap dynamically only through the missing-page-fault.

Recalling that the optimal control is achieved by the effective use of the inherent informations of the system, both methods can not be thought of optimal. In the case (1), the control program does not note what part of the program really to be used nor does think of particular charasteristics of the secondary memory, and CPU may often be made idle waiting for the end of I/O operation. On the other hand, the case of (2) is the complete on-demand method without an active intervention of control. In this case, though some pages are certain to be referred to and can be provided in the main memory before the program runs, these knowledges are completely ignored by the scheduler.

Meanwhile, in regard to the swap-out, the pages that have not been used are usually chosen. However, the past knowledge of the page usage gives no direct informations to the future condition. Any possibility of a page usage must be estimated only through investigating (1) to what process a physical page belongs, (2) how is the probability of this process to come to the running state and (3) how is the probability of the page to be used when the process comes to running.

As a conclusion, we can say that both methods using the opposite algorithm each other, can not be said optimal and leaves room for improvement.

2. 2. 2. *The informations for the control*

The first step of the optimization is to find useful and obtainable informations. The characteristics of the drum of disk, the distribution of processes, the state of each process, the number of processes waiting for the in/out operation etc. are common examples, but, more particulary, the following information will be useful for the swap control.

(1) *The probability of the page usage*

A process usually uses a number of pages and some of them are referred to with the higher probability than the others. Then we classify all pages into several ranks by the probability of the usage, and identify them by the rank numbers. A page and its LPDT to which the processor certainly transfers its control, Active User's Table (AUT) that contains each user's inherent informations, Segment Name Table (SNT) that contains segment name currently being used, are a few examples that are cslassified into the higher rank.

(2) *The state and the priority of the process*

The process to be swapped is selected according to its probability of coming to the running state. The modified states of the processes are described later. Job Scheduler that is an another control system in the TSS decides the priority of the process. This is also mentioned later. The process to be swapped-in is the one with higher probability and that to be swapped-out is the one with lower probability.

To utilize such informations, the swap must be at least partially supervised by the control program. This is called "scheduled swap" or "controlled swap" (CS, or in the case when the input or output operation is to be designated, CSI or CSO respectively in the abbreviated form). Besidies this, the requested swap (RS) is a swap request through the missing-page fault. A page with uncertain or very small probability of usage is swapped only through RS. On the other hand, the CS is carried out as a control action through which the state of the system is intended to direct to or stay in an ideal state, while intending to adjust the load to the secondary memory. One of its main object is to control the main memory in order to keep the sufficient available pages to eliminate the extra time for handling of the missing-page-fault.

2. 2. 3. *The states of the processes*

An unsatisfactory control sometimes induces some useless or rather harmful actions. For example, the scheduler designated as case (1) in section 2. 2. 1 will read out the many useless words. Similarly, in the CS, to swap out the pages that have been just swapped in and are

not yet used, shows this kind of loss. To avoid this, the processes that have been subjected to the swap-in to some extent are made free from the CSO. Thus we redefine the states of the processes except that of "Blocked" as follows;

* "Ready" state—a state of the processes that are not only ready for receiving the CPU's service but provided with the pages with a higher rank number than the predetermined level (R) already in the main memory. (R is a control parameter that can be adjusted so as to change the control characteristics.)

* "Pending" state—a state of all the processes that are also ready to receive the CPU's service but not in the "ready" state.

The processes in the ready state are no more subjected to the swap-out, while those in the pending state are subjected both to the CSI and CSO. This allows the control program to control the memory through the control of the processes. Thus, there are three states, that is, "ready" (including running), "pending" and "blocked". The relations or the causes of the transfers of the processes among them are as follows (See Fig. 2).

1) From running to pending

a) The slice time over

b) The RS have occurred without the available page because of the random property of the system. By this transfer, the process that has demanded the new page turns to the supplying side, automatically absorbing this irregular condition.

2) From running to blocked—The process requested (a) to be blocked (including the "quit" request) or (b) the terminal in/out.

3) From running to ready—(a) The in/out request to the secondary memory including the RS. The processes issued the in/out request are not subjected to the CSO.

4) From blocked to pending—(a) The wake-up signal issued.

5) From pending to ready and control of pending process—A process in the pending state must be selected by the control program to be subjected to CS, that is, a process with highest priority is chosen for the CSI and a process with the lowest priority for the CSO. When the CS has been done, the rank of the process that is defined as the highest rank number of pages be longning to the process in the main memory may change. As soon as the rank of the process with the highest priority reaches R, this process is transfered to the ready state. In this way, new state is introduced and this makes a system a little complicated but enables

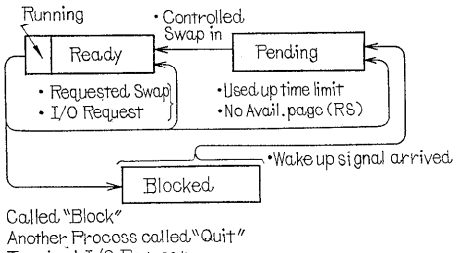


Fig. 2. States of the processes and their transitions.

the optimization of it.

2.2.4. *The control algorithm*

As an optimality criterion, we choose the effectiveness of the system. The control algorithm deduced from this criterion, cooperating with the "Job Scheduler", allows increased number of active users in the system without bringing any delay in the response time. In following three cases, the system would lose time.

(a) While there exist "ready" processes, CPU does nothing but wait for an external event to occur. For example, there are no available page and CPU waits for the completion of the swap-out.

(b) Though there are user's process, no "ready" process exists.

(c) The overhead due to the control

In order to reduce these chances to occur (particularly in the cases (a) and (b) because (c) concerns with the control program itself), the system must be controlled to keep the proper number of available pages and the ready processes. This can be carried out through the CS and then the control policy to be selected each time is one of three executions: They are (1) to carry out the CSI, (2) to carry out the CSO, and (3) to do nothing. The more elaborate use of informations, for example, for page selections, may be done after this policy selection.

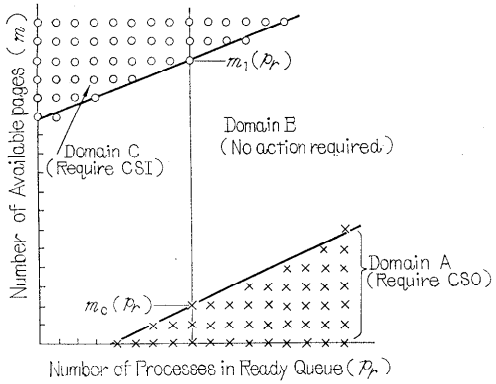
A control policy is selected on the basis of the system's state that is represented as the combination of (1) the number of currently available pages (m), (2) the number of ready processes (p_r), (3) the number of pending processes (p_p), (4) the number of blocked processes (p_b), (5) the number of processes in input/output waiting queue (p_c) and (6) the rank of the pending process with the highest priority (r) that shows the distance to the ready state of this process (when this rank number is r, it is thought there are more r/R ready processes). However, the number of possible states are too many to decide and remember a policy corresponding to each state. Then, also taking into account of the fact that the contributions of these informations to decision differ each other, we simplify this treatment:

Suppose there are only two sets of informations, say (1) and (2). Each state of the system consists of the combined pair of them to which an optimal policy corresponds; this is shown in Fig. 3, where each node represents a state and a corresponding policy is shown at each node. Every policy belongs to one of the three regions as shown, and the boundaries of these regions will be expressed as the function relations between two variables, as $m_0 = f_0(p_r)$ and $m_1 = f_1(p_r)$. In the more general cases of more than two variables, we can obtain a similar expression,

$$m_i = f_i(p_r, p_p, p_b, p_c, r) \qquad (i = 0 \text{ or } 1). \tag{1}$$

In general, the entire region will be divided into three comparatively simple shaped regions and the boundaries are expected to take also simple forms even in the n-dimensional space. Then the policy selection algorithm is rather simple,

1) if $m \geq m_1$ and $p_r \neq 0$, carry out the CSI
2) if $m \leq m_0$ and $p_p + p_b \neq 0$, carry out the CSO
3) do nothing for any other conditions.



CSI; Controlled swap in ○…Optimum policy is CSI
CSO; Controlled swap out ×…Optimum policy is CSO

Fig. 3. Optimum policy map in n-dimensional state space (The case of $n=2$ is shown where system's states are represented by the nodes. An optimum policy corresponds to each of them.)

The functions f_0 and f_1 can be determined through the procedure of the policy determination, but as they are supposed to be of comparatively moderate property, we approximate them as

$$m_i=A_ip_r+B_ip_p+C_ip_b+D_ip_c+E_ir+F_i \qquad (i=0 \text{ or } 1). \tag{2}$$

Here, as mentioned before, we let $E_i=A_i/R$ and as p_b has really few effect on m_i, we let $C_i=0$, then at last,

$$m_i=A_i(p_r+r/R)+B_ip_p+D_ip_c+F_i \qquad (i=0 \text{ or } 1). \tag{3}$$

The parameters A_i, B_i, D_i, F_i and R ($i=0$ or 1) must be decided analytically or experimentally, these parameters have large effects in the charasteristics of the control. The m_0 and m_1 must always be recomputed as the state changes. As the change of each variable in Eq. (3) is one at a time, we use an incremental form

$$\Delta m_i=A_i(\Delta p_r+\Delta r/R)+B_i\Delta p_p+D_i\Delta p_c \qquad (i=0 \text{ or } 1), \tag{4}$$

where Δp_r, Δr, Δp_p and Δp_c are either ±1 or 0. This means every time the state changes, relevant coefficients may be added to or subtracted from m_i by the control program. Besides these parameters, the number of pages swapped in one operation can also be included in the control parameters.

This optimization control is named "P-P control" because it controls the pages and processes simultaneously. Meanwhile, there must be Job Scheduler, the main role of which is to decide the priority and the slice time of the ready or pending process. This scheduling routine is called every time a process transfers to the ready or pending state. It is important to note that the Job Scheduler operates independently of the P-P control. These two controls may interlace or cooperate but do not interfere each other.

Conclusion

General concepts of optimization control in a computer system is discussed and a control

method in a TSS is described as an example. Though the optimization control is more and more important, there are no systematic ways to find the optimal control and then system designer's experiences, intuitions and ability of the analysis and synthesis are required without exception. The author tried to systemize at least a part of this procedure. However, there are many problems remained unsolved, for example, the way to find the best basic operational configuration of the system is quite important and more study is required.

References

[1] Howard, R. A., Dynamic Programming and Markov Process, Technology Press and Wiley, 1960.

[2] Osuga, S., An Internal Traffic Control in an Information Processing System, *Proc. 7th Conf. of JSIP*, (December 1966) 29–30.

[3] Saltzer, J. H., Traffic Control in a Multiplexed Computer System, MAC-TR-20, (June 1966)

[4] Corbató, F. J. and V. A. Vyssotsky, Introduction and Overview of the MULTICS System, *Proc. PJCC* 1965, 185–196.

[5] Glaser, E. L., J. F. Couleur and G. A. Oliver, System Design of a Computer for Time Sharing Application, *Proc. FJCC* 1965, 197–202.

[6] Vyssotsky, V. A., F. J. Corbató and R. M. Graham, Structure of the MULTICS Supervisor, *Proc. FJCC* 1965, 203–212.

[7] Gibson, C. T., Time-Sharing in the IBM System/360 Model 67, *Proc. SJCC* 1966, 61–78.

[8] Schwartz, J. I. and C. Weissman, The SDC Time-Sharing System Revisited, *Proc. A. C. M. National Meeting* 1967, 263–271.