Performance Monitoring of Operating System

Hajime Kitagawa*, Hiroshi Hagiwara**, Masanori Kanazawa*
and Yasuo Fujiwara***

ABSTRACT

In recent years, as computer systems have increased the complexity, it has become very important to evaluate the system performance by observing and analyzing behavior of an actual system. We have made to evaluate the system performance for FACOM 230-60 in Data Processing Center, Kyoto University by using software monitor as tool. In this paper we present the method to monitor the system and some results of performance evaluation.

1. INTRODUCTION

Monitoring is a method of collecting data on the performance of an existing system and useful in grasping dynamic behavior of system operation. System operation is not under equilibrium, and it is important to observe dynamically the relation between resource utilization and workload in order to locate the bottlenecks limiting performance. Considerable works have been done in the area of system performance monitoring in the past,[1] and they can be roughly categorized into three classes: hardware monitoring, software monitoring and monitoring by direct coupled computers.[2]

We evaluated the system performance of the operating system (MONITOR-V SYSTEM[4]) for FACOM 230-60 in Data Processing Center, Kyoto University by the software monitor. We monitored resource utilization and workload, and understood the actual state. A monitoring program operated in processing program mode and hence neither modification nor generation of control program was involved.

2. MONITORING TECHNIQUE

Monitoring program operates in processing program mode and its size is 1 K words which is a unit of core memory allocation. Informations which are observed at each selected interval consist basically of two kinds: status of resource

This paper first appeared in Japanese in Joho – Shori (Journal of the Information Processing Society of Japan), Vol. 13, No.11 (1972), pp. 789 ~ 797.

* Data Processing Center, Kyoto University
** Faculty of Engineering, Kyoto University
*** Osaka office, Fujitsu Co., Ltd.

utilization and system workload. Monitoring program records them on magnetic tape, accumulates statistics and outputs a useful part of results on console typewriter or character display device. Some parameters for monitoring such as an observation interval time can be changed from console dynamically.

Reference to control program area from monitoring program is possible in the following way, indicated in Fig.1. An interrupt handling routine for memory protection violation due to invalid data is declared with SPIE (Specify Program Interruption Exit) macro-instruction. If an instruction (LA P) trys to load the content of address P in control program area to any register (accumulator), interruption is occurred after the content of address P is loaded to the register. And control is transferred to the interrupt handling routine located at address L in monitoring program via handling routine in control program. Then, RESET macro-instruction makes control return to the next of register load instruction in the status that the content of address P remains in the register.

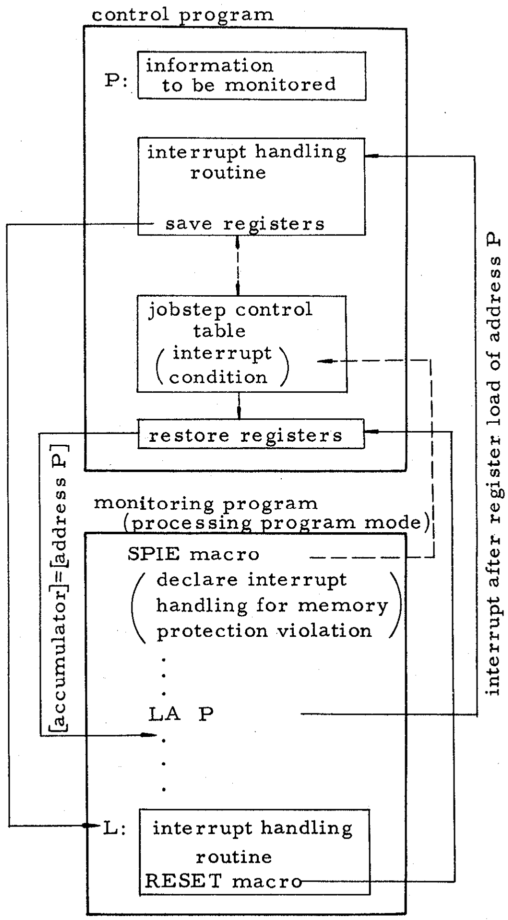

Fig.1. Reference to control program area in processing program mode.

Since interrupt handling routine runs at each reference to control program area in such a way, it is important to consider CPU time used by monitoring program. P_m, CPU utilization by the act of monitoring, is approximately given by

$$P_m = d \cdot f / t_m \ ,$$

where t_m: monitoring interval, d: number of words observed at each t_m, f: CPU time of 1 word reference to control program area. The greater part of f is the time of interruption handling routine and about 500 µs at the present system. So it is possible to observe about 20 words/sec on the condition of $P_m = 1$ %.

3. CORE MEMORY UTILIZATION AND REALLOCATION

It is important to use core memory effectively in multiprogramming environment because the throughput is influenced by its capacity and the cost is relatively high in comparison with other resources. In the case of the system without paging facility such as FACOM 230-60, it can not be prevented that useless empty areas remain between used domains. Thus reallocation of user's area becomes useful for improvement of performance. But FACOM 230-60 MONITOR-V SYSTEM has no such reallocation facility. We observed dynamically core memory utilization in user's area by refering memory control table in control program and made monitoring program to be possible to reallocate user's area in times of inefficient utilization.

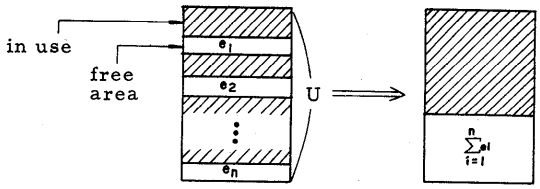

Fig.2. Reallocation.

An appropriate condition should be given on which the reallocation becomes effective for the increase of throughput. We gave the condition of reallocation in the following. If there are n empty areas $e_1, e_2, \cdots, e_n$ in core memory (size U) indicated in Fig.2, reallocation is triggered after continuous C_4 times of observation in the next status:

$$n \geq C_1, \quad E/U \geq C_2, \quad D/E \leq C_3,$$

where

$$E = \sum_{i=1}^{n} e_i, \quad D = \frac{1}{n}\sqrt{\sum_{i=1}^{n} e_i^2 - \left(\frac{E}{n}\right)^2}, \quad C_1 = 1,2,3\cdots, \quad 0 \leq C_2, C_3 \leq 1.$$

Reallocation trigger parameters, $C_1, \cdots, C_4$, are changeable dynamically from console, and meaningless reallocation can be prevented by giving the appropriate value to these parameters.

Fig.3 shows an example of core memory utilization monitoring. R gives utilization of core memory in user's area. This is the result of about 1 hour observation (t_m = 30 sec) at usual batch processing and the average of R is 84.5 %.

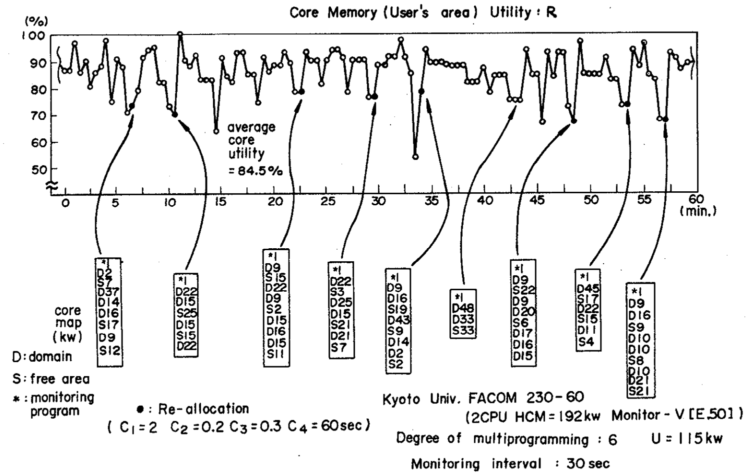

Fig.3. Core memory utilization monitoring.

Ordinarilly it can be said as the the result of monitoring that the average of R becomes 70~80 % without reallocation. Fig.3 also shows that 8 times of reallocation (C_1=2, C_2=0.2, C_3=0.3, C_4=2) prevent the system to be under equilibrium of the inefficient status. Such phenomenon is very effective to increase throughput in the case of batch processing containing large jobs.

4. CPU/CHANNEL/DRUM UTILIZATION

One of resources such as CPU, channel unit and i/o unit becomes often the bottleneck limiting performance. Hence, it is necessary to observe the utilization of these resources dynamically and concurrently. In the operating system like FACOM 230-60 MONITOR-V such observation with software monitor is possible by accumulating time or number of CPU idle task and sampling specific informations (device busy, channel path, etc.) in i/o unit control table.

We monitored such resource utilization in both FACOM 230-60 MONITOR-V batch processing and time-sharing system. In the case of the system in Kyoto University all system files are located in magnetic drum units, so we here consider only magnetic drums as i/o units. Fig.4 shows channel/drum configuration in the system monitored. And the results of observation are indicated in Fig.5 and Fig.6, where monitoring interval t_m is 2 sec, average values of 100 times (≃ 200 sec) of monitoring are plotted and CPU utilization by monitoring, P_m, is less than 1 %.

In the case of batch processing shown in Fig.5, CPU idle ratio is sufficiently small under the workload of the appropriate job-mix, and the system seems to be

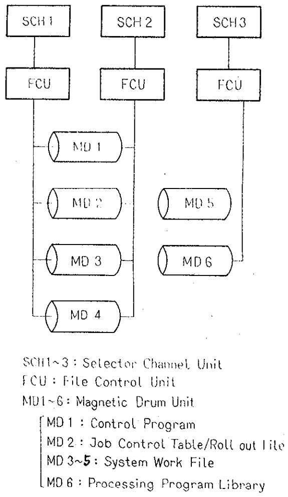

Fig.4. Channel/Drum Configuration.

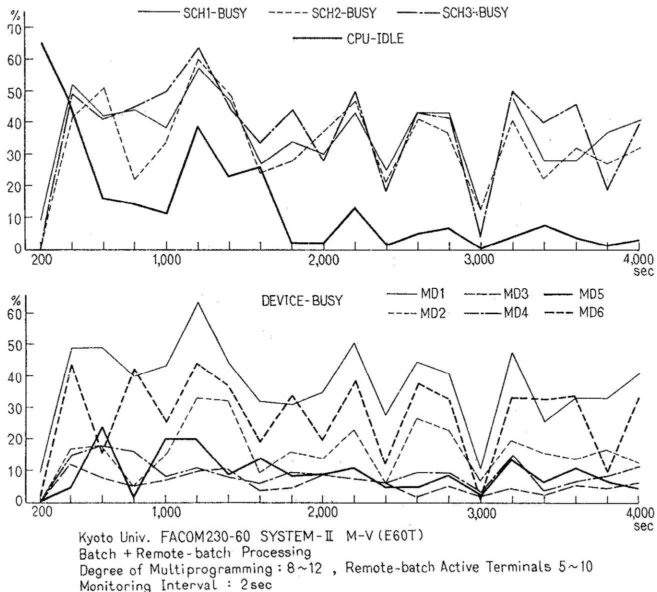

Fig.5. CPU/Channel/Drum utilization monitoring in batch processing system.

optimal. In the other case, time-sharing processing in Fig.6, the bottleneck limiting performance is located obviously in roll in/out traffic which is crowed by conversational and background processing. The conversational job is always rolled out immediately after they are put into the waiting status, however, the larger the number of active terminals and background jobs processed concurrently are, the more frequent roll in/out of batch jobs becomes. Thus, background processing which is intended to improve CPU utilization in time-sharing environment causes to increase swapping overhead time, consequently CPU utilization is not only improved but response time for terminals

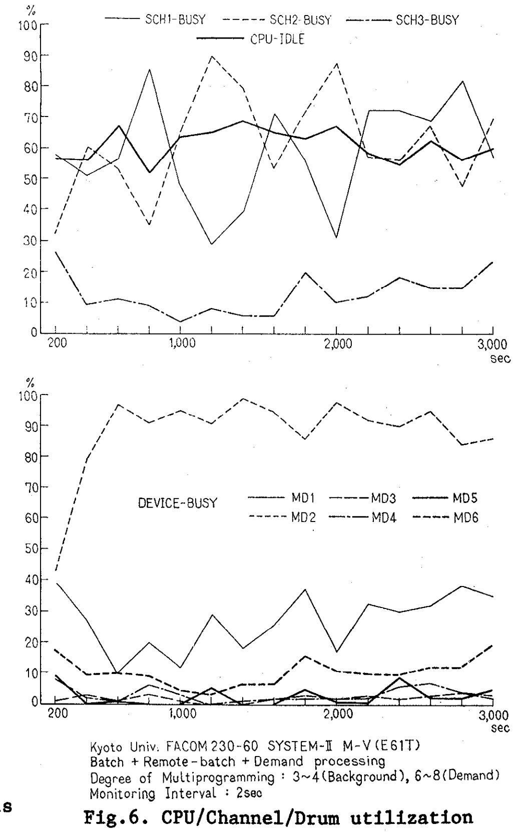

Fig.6. CPU/Channel/Drum utilization monitoring in time-sharing system.

becomes also worse in this system. This problem may be solved by installing large capacity core memory or highspeed channel/drum units.

5. CONCLUDING REMARKS

Addition to the above, we monitored the utilization of system work area and transient module loading area in control program dynamically and took in the effective utilization of core memory by feeding back their optimal size and frequency of use among transient modules to system generation.

Since the system performance is considerably influenced by environment and workload, the performance evaluation must be discussed in the context of a specific application or individual system, and is necessary at user side for an efficient use of computer system. With this point in view, the technique which was used in this software monitor is very useful because of ease.

Besides monitoring, we analyzed the system performance for various system parameters by the simulation,[3] and knew that the present hardware configuration and operating system seem to be optimal for batch processing in our center.

The drawback of monitoring method to evaluate any proposed system other than the actual one are i) necessity for changes of operating system and hardware configuration, ii) difficulty for reappearance of the same workload, and iii) time-consuming for real-time observation. And these problems could be made good by using together with simulation method.

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

REFERENCES

1) H.C. Lucas, Jr. : Performance Evaluation and Monitoring, Computing Surveys, Vol.3, No.3, (Sept. 1971).

2) R. Sedgewick, R. Stone and J.W. McDonald : SPY - A program to monitor OS/360, PROC. 1970 FJCC, Vol.37.

3) M. Kanazawa, H. Kitagawa and H. Hagiwara : Simulation of a Computer System, Journal of IPSJ, Vol.13, No.10, (Oct. 1972).

4) FACOM 230-60 MONITOR-V System, Fujitsu Co., Ltd., EX-041-2/EX-043-2.