Bulletin of Networking, Computing, Systems, and Software – www.bncss.org, ISSN 2186-5140
Volume 8, Number 2, pages 87–92, July 2019

Disk I/O Control Mechanism for Online and Batch Processing

Masanori Tanabe
Graduate School of Natural Science and Technology, Okayama University
Okayama, Japan
tanabems@s.okayama-u.ac.jp

Yoshinari Nomura
Graduate School of Natural Science and Technology, Okayama University
Okayama, Japan
nom@cs.okayama-u.ac.jp

Kazutoshi Yokoyama
School of Information, Kochi University of Technology
Kochi, Japan
yokoyama.kazutoshi@kochi-tech.ac.jp

Takashi Nagao
Graduate School of Natural Science and Technology, Okayama University
Okayama, Japan
takashi.nagao.xu@hitachi.com

Hiedo Taniguchi
Graduate School of Natural Science and Technology, Okayama University
Okayama, Japan
tani@cs.okayama-u.ac.jp

***Abstract* — Some IT systems, including banking systems, perform workload cycles on a daily basis. These workloads are *composite workloads* of online and batch processing. To separate the impact of these two workloads, typical banking systems employ individual computers dedicated to each type of workload. Since each dedicated system is designed to fit the peak times of their respective tasks, it is difficult to optimize the scale of the system as a whole. In this paper, we propose an effective method for maximizing disk I/O performance under composite workloads. Our basic idea depends on the fact that online workloads and batch workloads (1) have different peak times in a day, and (2) have different I/O patterns and lengths of disk access.**

***Keywords* — disk I/O control, I/O scheduling, online processing, batch processing**

I. Introduction

IT systems, like those used in banking systems, have two types of workloads: online processing workloads and batch processing workloads. Online processing workloads are short-time processing tasks that should be executed quickly by answering user's request. It is difficult to predict the number of online processing workloads. On the other hand, batch processing workloads are executed preemptively. Therefore, online processing workloads require elastic resource allocation while batch processing workloads require statically optimal resource allocation.

To handle these composite workloads, typical banking systems employ individual computers dedicated to each type of workload. Since each dedicated system is designed to fit the peak times of their respective tasks, it is difficult to optimize the scale of the system as a whole. However, we believe that executing online processing workloads and batch processing workloads on a single computer is possible if we schedule the workloads carefully. Basically, the two workloads have different peak times in a day (shown in Figure 1); if not, it is crucial to reduce the impact of batch processing workloads on the response time of online processing workloads. Therefore, it is important to prioritize the scheduling of online processing workloads over batch processing workloads. The scheduling of composite workloads using conventional priority control on a CPU will not work well because resource conflicts stem not from the CPU but from disk I/O requests. When the I/O request of batch processing workload is invoked before online processing, the online processing workload will be suspended for long time.

In this paper, we present how the disk I/O requests of batch processing workloads influence online processing workloads within *composite workloads*. We also propose a new disk I/O control mechanism that gives high-priority to I/O requests of online processing workloads. This mechanism reduces the waiting time of I/O requests of online processing workloads.

II. A System Environment to Execute Online Processing Workloads and Batch Processing Workloads

A. Delay of an I/O request to a disk

The execution time of online processing workloads should be short because the response time directly impacts the quality

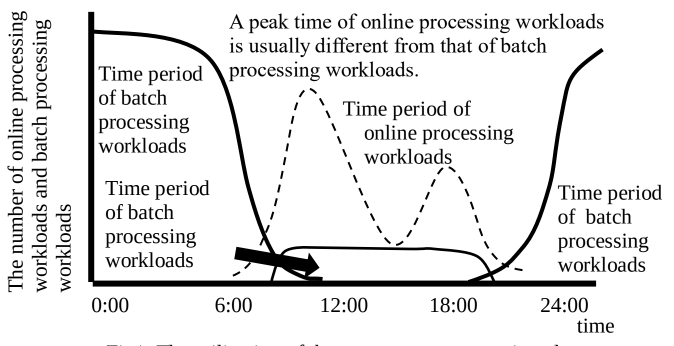


Fig1. The utilization of the computer resources in a day.

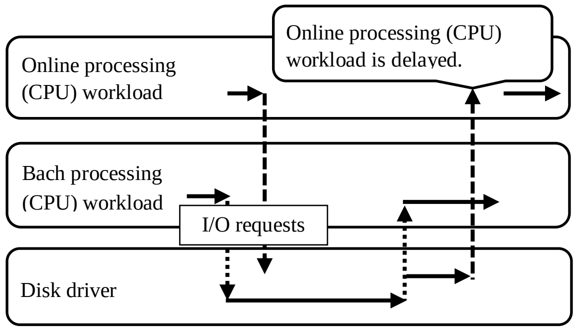

Online processing (CPU) workload is delayed.
Online processing (CPU) workload
Bach processing (CPU) workload
I/O requests
Disk driver

Fig. 2. I/O requests and delay of processing.

of the user experience. Therefore, online processing workloads often use CPUs for a short time and emit smaller disk I/O requests. On the other hand, the I/O requests and CPU usage of batch processing workloads are larger than that of online processing workloads. Therefore, not only priority control of the CPU but also priority control of I/O requests are necessary to guarantee the response time of online processing workloads. For example, when a batch processing workload requests a disk I/O that has a long execution time, the online processing workload will have to wait for this to be finished because the disk driver cannot handle I/O requests in parallel. Figure 2 shows this situation. In the figure, the online processing (CPU) workload and batch processing (CPU) workload execute their tasks by sharing the single CPU resource. The batch processing (CPU) workload calls the disk driver to execute an I/O request, and the disk driver handles the I/O request. Right after this I/O request, the online processing (CPU) workload also calls the disk driver. However, the I/O request of the online processing (CPU) workload is blocked until disk I/O of the batch processing workload is finished. To remedy this situation, we can assign the online workload a high CPU priority, which gives it a chance to call the disk driver prior to other batch I/O requests. However, when the disk I/O is emitted by the batch workload by chance, then the CPU priority will not work because the disk I/O is completely serialized on a first-come, first-served basis. Therefore, it is required to establish a disk I/O control mechanism to prioritize particular I/O requests.

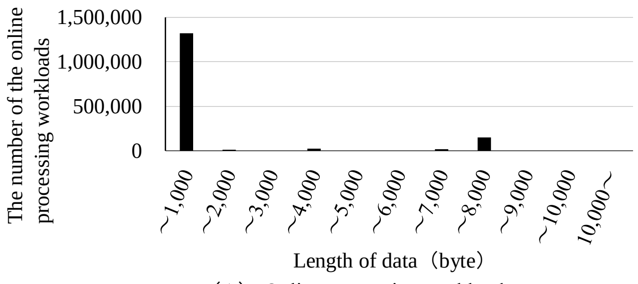
III. Execution time of the I/O requests with different length of data

A. *Comparison of the I/O requests*

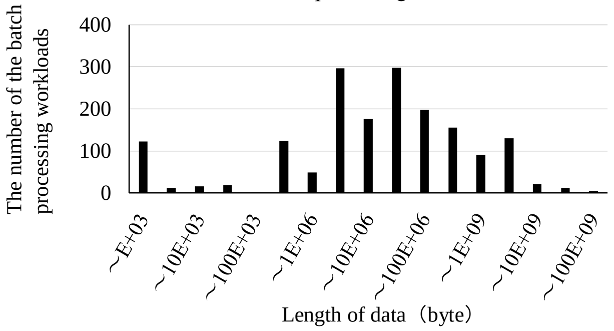
The typical flow of one transaction in an online processing proceeds as follows: (1) the contents of the user's request are written on a disk; (2) the process is executed; (3) the results are written on a disk; and (4) the results are returned to the user. For the batch processing, (1) the process is executed, and (2) and the results are written on a disk repeatedly.

TABLE I. Specifications of the computer.

CPU	Intel i5 3.2GHz 4 cores No hyper-threading	
Disk for data	I/O path	SATA3.0
	DISK	7,400 RPM 32MB cache
	File system	UFS
	Cache of file system	Not use
OS	FreeBSD 11.0-RELEASE-p1	



（A） Online processing workloads



(B) Batch processing workloads

Fig. 3. Distribution of length of data.

The length of data that batch processing workloads write on a disk is different from that of online processing workloads. As shown in Figure 3(A), the lengths of most data written by online processing workloads are less than 1,000 bytes. In contrast, Figure 3(B) shows that the length of data written by batch processing workloads range from over one million to several hundred million bytes.

B. *Analyzing relation between size of data and processing time of I/O request Units*

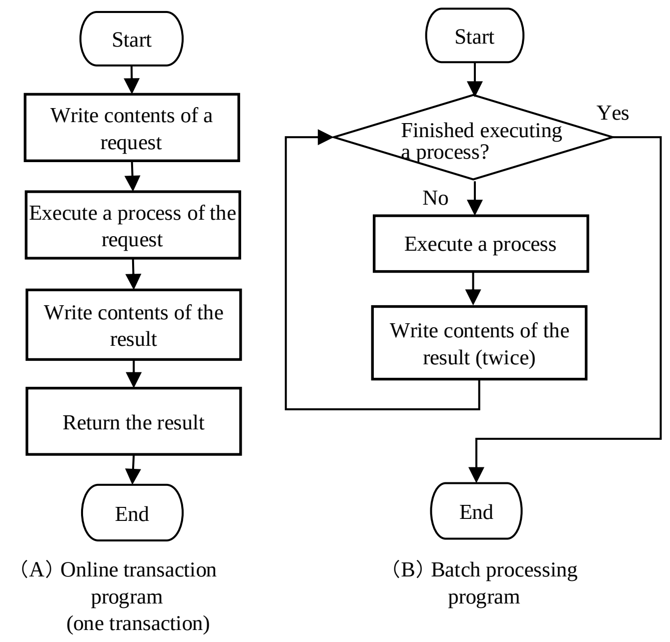
Table I shows the specifications of the computer we used to measure the relation between the length of the data and processing time.

C. *Evaluation programs*

We prepared two evaluation programs based on the characteristics of online processing workloads and batch processing workloads described in Section III(A). Figure 4(A) shows the flow of the pseudo online transaction program of the online processing workload. It consists of writing the contents of a request on a disk, executing a process for the request, writing data of the results on a disk, and finally returning the results to the user. This pseudo online transaction program is executed repeatedly. Figure 4(B) shows the flow of the pseudo batch processing program. It repeats the execution of a process and writes data (twice) on a disk. When the pseudo batch processing program writes data on the disk, it writes large data on the disk twice.

The pseudo online transaction program writes 1,000 bytes for each I/O request. The batch transaction program writes one million bytes each.

We determine the number of transactions of the pseudo online transaction program and the number of times the pseudo



Start
Write contents of a request
Execute a process of the request
Write contents of the result
Return the result
End
(A) Online transaction program (one transaction)
Start
Finished executing a process?
Yes
No
Execute a process
Write contents of the result (twice)
End
(B) Batch processing program

Fig. 4. Flow of evaluation programs.

batch processing program is repeated based on the following assumptions: (1) CPU time of the pseudo online transaction program and pseudo batch processing program is equal to the time to write all of the data; and (2) the pseudo online transaction program and pseudo batch processing program are executed at the same time slot. Specifically, we determine that the number of transactions of the pseudo online transaction program is 50, and the number of repetition times of the pseudo batch processing program is five. In the case of the pseudo online transaction program, if the CPU time is 0 s, then the execution time to write all the data is 92.63 ms when the pseudo online transaction program executes the transactions 50 times. In other words, the time to write data on the disk for one transaction is 1.85 ms. In the pseudo batch processing program, if the CPU processing time is 0 s, then the execution time to write data on the disk five times is 297.05 ms. In other words, the execution time to write data on the disk (twice) is 59.41 ms. We determine that the execution time (CPU) of the pseudo online transaction program (one transaction) is 2 ms while that of the pseudo batch processing program is 60 ms because we assume that the time to execute both pseudo programs are almost equal to the time for both pseudo programs to write data. In addition, we determine that the execution time to return a result is 0 ms because it is simpler than others. The execution time of the pseudo online transaction program with 50 transactions is 192.5 ms. The execution time of the pseudo batch processing program with five repetitions is 597.05 ms.

D. *Results of the execution*

Figure 5 shows the results of the evaluation, i.e., the time elapsed to write 1,000 bytes of data. We also performed the same evaluation by changing the parallelism of the transactions (1, 2, 5, and 10).

As the number of concurrencies of pseudo online transaction programs increase, the performance to write data decreases. On the other hand, pseudo batch processing programs are not affected by the concurrency. With the increase in the number of

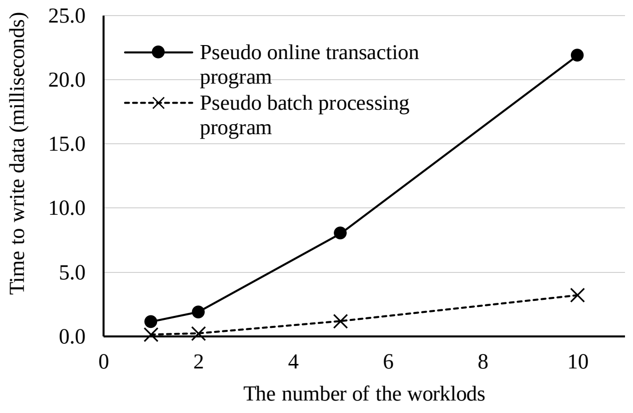


Fig.5 The relation between the number of workloads and the average time to write 1,000 bytes of data.

pseudo batch processing programs, the number of write() system calls with long execution time increase. Consequently, the disk driver is occupied with the workloads of batch processing programs. In other words, the possibility of the workloads of pseudo online transaction programs getting stuck is increased.

In the case of the 1,000-bytes transaction with the single concurrency program, the execution time is 1.23 ms in Figure 5. In the case of one million bytes of data, it becomes 0.032 ms per 1,000 bytes. For simple calculations, these two should be same. However, this depends on the length of whole data. Therefore, the I/O throughput will be good when the length of data to write at once is long.

When batch processing workloads to write long-length data at once and online processing workloads to write short-length data are executed on the same computer, the execution time of the I/O request of online processing workloads becomes longer as the number of requests increase, and the response time becomes worse.

IV. The Disk I/O Control Mechanism

A. *Execution Control of I/O requests*

When online processing workloads and batch processing workloads are executed on the same computer, the online processing workload, which has I/O requests with short-length data, is often encumbered by the batch processing workload. Thus, we focus on the fact that the length of data of the online processing workloads is short. We propose an execution control mechanism for I/O requests, which executes I/O requests with short-length data with high priority. That is:

1. If the data length is short, it should have high priority (= I/O requests of online processing).
2. If the data length is long, it should have low priority (= I/O request of batch processing)

Figure 6 shows the structure of the execution control mechanism of the I/O requests.

1) I/O queues for I/O requests

We have two types of queues classified according to I/O length. The I/O request is inserted into the queue which corresponds to the data length. To classify each request, the argument of the write() system call will be captured.

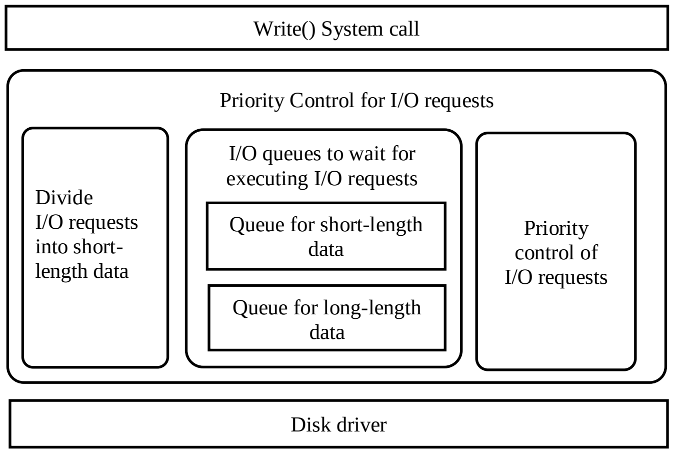

Write() System call
Priority Control for I/O requests
Divide I/O requests into short-length data
I/O queues to wait for executing I/O requests
Queue for short-length data
Queue for long-length data
Priority control of I/O requests
Disk driver

Fig.6. Disk I/O control mechanism for I/O requests to a disk.

2) Divide I/O requests into pieces of short-length data

By dividing the data, we can shorten the execution time of each I/O request. As a result, this mechanism can reduce cases where the long-data disk I/O blocks other I/O requests. In other words, we can shorten the waiting time of the I/O requests of online workloads.

3) Priority control of I/O requests

The priority control takes an I/O request out of a queue and sends it to a disk driver. If there are I/O requests in the queue for short-length data, the priority control takes the I/O request out of the queue for short-length data. If there is no I/O request in the queue for short-length data, then the priority control takes the I/O request out of the queue for long-length data.

V. EVALUATION USING A SIMULATION MODEL

A. *Evaluation points*

Under the situation in which a lot of I/O requests occur, the waiting time for executing the I/O request and disk busy rate increase. Therefore, we evaluate following:

1. Disk busy rate and elapsed time to execute I/O requests.
2. Disk busy rate and elapsed time to wait for starting execution of I/O requests.

We consider three cases of disk busy rate: high rate, low rate, and middle rate. We define the *disk busy rate* as the ratio of the amount of time assigned to drive the I/O bus out of the whole execution time.

We evaluate following three types of controls for above-mentioned 1 and 2 in our simulation model.

1. No priority control
2. Priority control of the CPU (online workload has high priority)
3. Priority control of both CPU and I/O requests (online workload has high priority).

B. *Assigned computer resouce*

We determine the maximum dividing length of I/O request, where the elapsed time to write data is nearly equal to the time to execute the I/O request of 1,000 bytes. Specifically, in our measurement environment, the time to write 1,000 bytes of data is 0.926 ms and the time to write 10,000 bytes of data is 1.26 ms. Thus, we determine that the length of data into which an I/O request is divided is 10,000 bytes of data. The computer using in our simulation model has four CPUs and a single I/O path. We set the time to write 1,000 bytes and 10,000 bytes of data as 1 ms.

We describe below the rules of the CPU and I/O path allocation in our simulation model.

1) Rules of CPU allotment

(A) When multiple tasks are awaiting CPU allotment, the tasks are assigned to the CPU in a first in, first out (FIFO) manner.

(B) The CPU time slice is 10 ms. The next awaiting task is assigned to the CPU in a unit of 10 ms.

(C) Tasks have CPU affinity; thus, when a task is scheduled for the first time, it can be freely assigned to any vacant CPU. But once it is bound to a specific CPU, it should be fixed to that same CPU.

2) Rules of I/O path allotment

(A) If execution of the I/O requests are required from multiple tasks simultaneously, these are handled in the order of FIFO.

(B) The time slice of an I/O path is 1 ms. An I/O request is assigned to the I/O path in a unit of 1 ms if there are other pending I/O requests.

C. *CPUs and I/O path Allotment using the priority control*

We apply the priority control of CPUs and I/O path allotment. We describe the rules of priority control below.

1) Rules for priority control of CPU allotment

(A) When multiple tasks are awaiting CPU allotment simultaneously, the tasks with high CPU priority are assigned. If there are multiple tasks with the same priority, then these are handled in FIFO order.

2) Rules for priority control of I/O path allotment

(A) If execution of the I/O requests are required from multiple tasks awaiting I/O path allotment simultaneously, then the I/O request of the task with high priority is assigned to the I/O path.

(B) The priority of I/O requests is compared. If the priority of I/O requests is same, these are handled in FIFO order. If the I/O request has a high priority, it is assigned to the I/O path.

D. *Simulation model*

Table II shows our model of online processing workloads. The online processing workloads use the same simulation model with the three cases of disk busy rate. Banking IT systems are usually designed to have a margin of CPU utilization because online processing workloads are executed based on the demands of many users, and the workloads for computer resources cannot be assigned preemptively. Therefore, we determine the start timing of the online processing workloads so that CPU utilization of online processing workloads is approximately 50% for the cases of our simulation model. We show models of the batch processing workloads of case 1 (low disk busy rate), case 2 (middle disk busy rate), and case 3 (high disk busy rate) in Table III. We realize that the increase or decrease of the disk busy rate changes the total length of data, which is the I/O request of batch processing workload. The length of data of the I/O request is divided into pieces of data which are assumed to be 10,000 bytes.

E. *Results*

Table IV shows the disk busy rate and total CPU utilization of each case. Table V shows the CPU utilization of the online processing workloads and batch processing workloads.

1) Evaluation 1 (The average execution time)

Figure 7 shows the average execution time of the online processing workloads and batch processing workloads of each case. We make the following observations from Figure 7.

(A) The average execution time of online processing workloads improved for cases 1 to 3 using the priority control for I/O requests. In order of case 1, case 2 and case 3, the average execution time of online processing workloads in case 3 is the longest with the no priority and only CPU priority controls. In other words, this shows that the average execution time of online processing workloads tends to be longer as the disk busy rate rises in cases of the no priority and only CPU priority controls. This is because the execution time of the I/O requests of batch processing workloads increases and the number of online processing workloads that wait a long time for the execution of I/O requests increases. On the other hand, in the case of CPU+I/O priority control, there is a slight increase in the average execution time of online processing workloads in cases 1 to 3.

(B) With regard to the batch workloads, the average execution time using only CPU priority control or CPU+I/O priority control is longer compared with the average execution time using no priority control for all three cases. In addition, the average execution time of the only CPU priority control is slightly different from the CPU+I/O priority control in cases 1 to 3. Table V shows the comparison of CPU utilization of batch processing workloads with only CPU priority control and CPU+I/O priority control. The CPU utilization is lower for the no priority control. In other words, this shows that the CPU allotment for batch processing workloads decreases because the CPU allotment for online processing workloads increases. The average execution time using I/O priority control does not increase. In other words, the I/O priority control does not influence the average execution time of batch processing workloads.

TABLE II. MODEL OF ONLINE PROCESSING WORKLOAD.

Processing time	100 ms
Interval of I/O requests	One time at starting and ending of processing
Number of I/O requests	two times
Length of data to write at a once	1,000 bytes

TABLE III. MODEL OF BATCH PROCESSING WORKLOAD.

	Case 1	**Case 2**	**Case 3**
Disk Busy Rate	Low	Middle	High
Processing time	10 s		
Interval of the I/O requests	Equal distance		
Number of the writing times of the I /O requests	20 times (once in 0.5 s)		
Sum of the processing time of the I/O requests	1,000 ms	2,000 ms	4,000 ms
Total length of data	10,000,000 bytes	20,000,000 bytes	40,000,000 bytes
Length of data to write at a once	500,000 bytes	1,000,000 bytes	2,000,000 Bytes
Divided length of data	10,000 bytes	10,000 byes	10,000 bytes

TABLE IV. DISK BUSY RATE AND TOTAL CPU UTILIZATION.

	Case 1 (%)		**Case 2 (%)**		**Case 3 (%)**	
	DISK	**CPU**	**DISK**	**CPU**	**DISK**	**CPU**
No priority	23.5	97.3	40.0	95.5	72.2	91.1
Only CPU priority	16.5	80.0	28.2	79.5	53.7	80.3
CPU＋I/O priority	16.4	80.0	27.9	79.6	52.8	80.0

TABLE V. CPU UTILIZATION (ONLINE PROCESSING WORKLOAD AND BATCH PROCESSING WORKLOAD).
ON: online processing workload, BT: batch processing workload

	Case 1 (%)		**Case 2 (%)**		**Case 3 (%)**	
	ON	**BT**	**ON**	**BT**	**ON**	**BT**
No priority	49.8	47.5	50.2	45.3	49.9	41.2
Only CPU priority	50.0	30.0	53.5	26.0	49.3	31.0
CPU+I/O priority	50.0	30.0	50.0	29.6	50.0	30.0

2) Evaluation 2 (The average waiting time until start of execution of the I/O request)

Figure 8 shows the average waiting times until the start of execution of the I/O requests of online processing workloads and batch processing workloads. We make the following observations from Figure 8.

(A) The average waiting time until start of execution the I/O request of online processing workloads is 0.4 ms in the case of

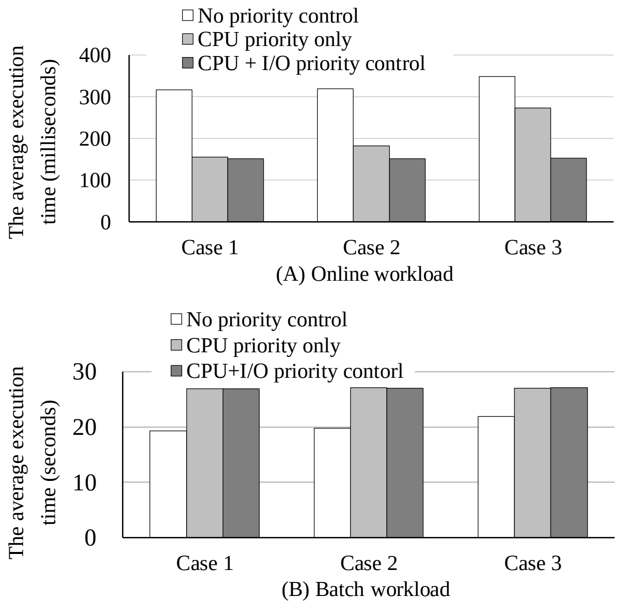
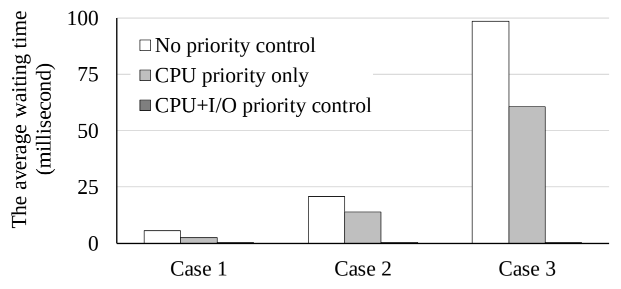
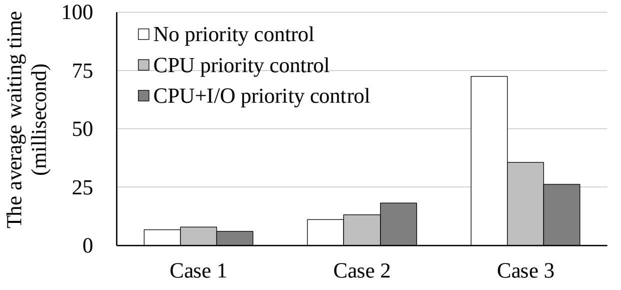


Fig. 7. The average execution time.



(A) Online processing workloads



(B) Batch processing workloads

Fig.8. The average waiting time of the I/O requests.

CPU+I/O priority control. The average waiting time do not increase in cases 1 to 3. However, the waiting times of the no priority control and only CPU priority control becomes worse. In other words, the CPU+I/O priority control has a significant effect on average waiting time until the I/O request of online processing workloads is executed when the disk busy rate is high.

(B) The average waiting time until the I/O request of batch processing workloads is executed do not differ in cases 1, 2, and 3. In addition, the average waiting time of the I/O requests using CPU+I/O priority control is shorter than the only CPU priority control. Therefore, CPU+I/O priority control affects the average waiting time of I/O request in batch processing workload. However, in case 2, the average waiting time of the I/O request using the CPU+I/O priority control becomes slightly longer compared with only CPU priority control. Depending on the timing of CPU allotment, the waiting time of I/O requests might increase.

VI. Related work

Some subjects related to I/O requests in virtual machine environments have been investigated [1][2][3][4][5]. These studies clarified the effectiveness of priority control. However, the priority control of I/O requests in virtual machine environments is intended to bring the utilization rate of I/O paths closer to a utilization value assigned beforehand, and to reduce the influence on other virtual machine environments. Thus, the goal of these studies is different from the priority control of applications. A method was proposed to reduce the waiting time of I/O requests in consideration of the property of application or the execution time of I/O requests [6]. The proposed methods does not solve the problems of I/O requests of online processing workloads. Rather, it aims to improve the execution time of online processing workloads by predicting a processing flow using a characteristic of an application and by controlling interruptions of the time slice assigned to the CPU [7]. However, the priority control of I/O requests is also needed to improve the utilization of computer resources. Therefore, controlling the execution time of the I/O requests was proposed by coordinating their execution time [8]. In this proposal, the execution time of I/O requests was coordinated as the I/O request with high priority was executed with precedence. Then, this proposal showed that the waiting time of I/O requests with high priority becomes shorter. However, because the performance of the response time and processing throughput are important for IT systems, the performance deterioration by coordinating the execution time of I/O requests causes problems.

VII. Conclusion and future work

Online processing workloads are not executed with precedence in systems with online processing workloads and batch processing workloads. We focused on the fact that the length of data of online processing workloads is short. Using our simulation model, we showed that it is possible to reduce the execution time of online processing workloads with a disk I/O control mechanism.

Batch processing workloads should be divided into pieces of short-length data to increase the chances of processing online processing workloads. However, excessive division can decrease performance. Thus, we need to determine the appropriate data length.

The implementation and evaluation of the proposed disk I/O control mechanism will be conducted in future work.

References

[1] Filip Blagojević, Cyril Guyot, Qingbo Wang, Timothy Tsai, Robert Mateescu and Zvonimir Bandić, “Priority IO Scheduling in the Cloud,” 5th USENIX Workshop on Hot Topic in Cloud Computing, June 25–26, 2013.

[2] Mukil Kesavan, Ada Gavrilovska, Karsten Schwan, “On Disk I/O Scheduling in Virtual Machines,” The 2nd Conference on I/O Virtualization (WIOV’ 10), March 13, 2010, USA.

[3] Ziye Yang, Haifeng Fang, Yingjun Wu, Chungi Li, Bin Zhao, H. Howie Huang, “Understanding the Effects of Hypervisor I/O Scheduling for Virtual Machine Performance Interference,” 2012 IEEE 4th International Conference on Cloud Computing Technology and Science, 3-6 Dec. 2012.

[4] Kazuhiko Mizuno Takayuki Imada, “I/O Performance Evaluation in the Virtual Environment and Optimization Method Suggestion,” IPSJ Computer System symposium, pp. 86–93, 2016 (in Japanese).

[5] Ramon Nou, Jacobo Giralt, Toni Cortes, “Automatic I/O scheduler selection through online workload analysis,” The 9th IEEE International Conference on Autonomic and Trusted Computing (ATC 2012), Sep. 4–7, 2012.

[6] Tatsuya Katagami, Toshihiro Tabata (Yamauchi), Hideo Taniguchi, “Proposal of I/O Buffer Cache Mechanism Based on the Frequency of System Call of the File Operation,” IPSJ Transactions on Advanced Computing Systems Vol. 3 No. 1, pp. 50–60 (Mar. 2010) (in Japanese).

[7] Yoshinori AOKI, Sukanya SURANAUWARAT, Hideo TANIGUCHI, “A Load Distribution Scheme for a New Transaction Service Considering the Pre-Loaded Services,” IEICE TRANS. INF. & SYST., Vol. E82-D, No. 22, Nov. 1999.

[8] Takashi Nagao, Hideo Taniguchi, “Inplementation and Evaluation of Mechanism for Regulating the Service Time Based on Controlling the Number of I/O Request,” IEICE Transactions Information and Systems, D Vol. J94-D, No. 7, pp. 1047–1057 (in Japanese).