A Resource Management Method for Improving Recycling Ratio in Recycling Process Elements

Toshihiro TABATA
Faculty of Information Science and Electrical Engineering, Kyushu University
6-10-1 Hakozaki, Higashi-ku, Fukuoka 812-8581, Japan

and

Hideo TANIGUCHI
Faculty of Engineering, Okayama University
3-1-1 Tsushimanaka, Okayama 700-8530, Japan

ABSTRACT

A process is a program in execution. Processes can be executed concurrently in operating Systems (OS) and they may be created and be deleted dynamically. Process creation and termination are required for program execution. They are an important processing, but the cost of them is expensive. The cost of them affects the execution performance of programs. We proposed a fast process creation and termination mechanism by recycling process elements. The management of preserved process elements is an important problem for recycling. We also proposed an efficient resource management for recycling process elements. The proposed method can reduce the amount of memory consumption of preserved resources. It focused on frequency of program execution, but it is insufficient to reduce the cost.

In this paper, we propose an improved resource management method for recycling process elements. In the method, only one process element with program image is preserved for each program with high frequency in program execution. The method can reduce the amount of memory consumption of preserved process elements. We also describe the implementation of proposal method on ***Tender*** operating system and report the contents of experiments and the result of them.

Keywords: Operating system, Resource management, recycle, Process creation, Process termination, ***Tender***

1 Introduction

A process is a program in execution. Processes can be executed concurrently in operating Systems (OS) and they may be created and be deleted dynamically. Process creation and termination are required for program execution. They are an important processing, but the cost of them is expensive. The processing of process creation needs creation of virtual memory space for new process, loading a program and so on. The processing of process termination needs the deletion of process elements.

The cost of process creation and termination affect the execution performance of programs. For example, Apache HTTP Server sometimes creates a process for executing a CGI program, when it receives a request. In this case, the number of created processes is same as the number of the requests. We think that to reduce the cost of process creation and termination is a crucial issue of server programs. Therefore, a new method that can reduce the cost of process creation and termination is required.

We proposed a fast process creation and termination mechanism by recycling process elements. We implemented the mechanism on The ENduring operating system for Distributed EnviRonment (***Tender***)[1]. However, preserved process elements for recycling consume many memory resources. Therefore, the preserved process elements have to be managed efficiently.

We also proposed an efficient resource management method for recycling process elements[2]. We focused on frequency in program execution. There are two types of preserved resources for recycling from the standpoints of frequency in program execution. On programs with high frequency in program execution, process elements are preserved with program image. Thus, they can be recycled when a process is created from the same program. When programs with low frequency in execution terminate, process elements are preserved without program image. They can be recycled for any process that satisfies the condition of recycling.

The proposed method can reduce the amount of memory consumption of preserved resources. It focused on frequency of program execution, but it is insufficient to reduce it. In the method, the number of preserved process elements with program image for each program is more than one. One of them is always recycled when the program is executed.

However, the remains of them (more than 2) are not recycled if plural processes are not executed concurrently. Therefore, the remains of them are useless and only consume memory.

In this paper, we propose an improved resource management method for recycling process elements. In this method, only one process element with program image is preserved for each program with high frequency in program execution. When a preserved process element with process image of same program exists, the process is broken up into process elements without program image. The method can reduce the amount of memory consumption of preserved process elements. It can also improve the ratio of recycling process elements, because preserved process elements without program image can be recycled for process creation for every program. Thus, it can reduce the cost of process creation. We also describe the implementation of proposal method on ***Tender*** operating system and report the contents of experiment and the result of them.

2 Related Work

Many researches that can reduce the cost of process creation have been proposed in the past. Sticky bit function and vfork system call can reduce the cost of process creation in UNIX[3]. Sticky bit is a research of fast process creation. Sticky bit enables OSs to recycle a text region.

On Demand Paging (ODP) and Copy on Write (CoW) are also the researches of memory management. Furthermore, ODP and CoW can reduce the cost.

Existing OSs are not able to recycle all of process elements. They cannot also create process elements in advance. ***Tender*** operating system differs from them with the point that memory resources can be recycled in the processing of a process creation.

Furthermore, there is the research of the page control and the cash memory in the virtual memory, as the research that is similar to manage the recyclable memory resources[4], [5], [6]. These researches manage the memory with an operation unit that the memory management provides. For example, an operation unit is a physical page. On the other hand, page tables and program resources are managed as an operation unit in the ***Tender*** operating system, in addition to the physics pages.

3 *Tender* Operating System

Separation and Individualization of Resources

In ***Tender***, objects to be controlled and managed by an operating system are called “resources”. We realized high modularity among the resources. The resources are given resource name and resource identifier. Figure 1 shows the structure of resource identifier and resource name.

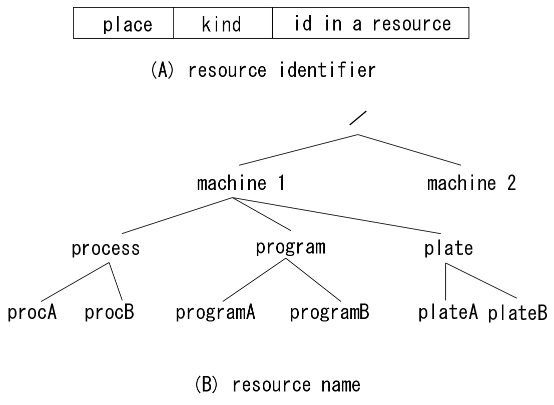

Figure 1. Resource identifier and resource name.

Besides, the interface of the operation of resources is unified. Furthermore, the program parts, which operate each resource, are separated and shared programs are eliminated. The management information of each resource is also separated between each resource. In addition, a pointer between the management tables of each resource is prohibited.

Tender manages resources by resource identifier and resource names. In addition, resource identifier and resource name include the information of location that indicates a machine. Besides, resources are controlled by unified interface. As a result, ***Tender*** can operate local resources and remote resources by the unified interface.

The existence of the process elements of existing OSs depends on a process that owns them, because the management information of each resource is stored in the process management table. If a process terminates, the entry of process management table is cleared. As a result, its process elements and the information of them are cleared, too.

In ***Tender***, the existence of each resource does not depend on other resources including a process, because the table of each resource is separated. The separation and the individualization of the resources enable ***Tender*** to preserve the resources for recycling. We call resources that compose a process ”process elements”. As a result, each resource can exist irrespective of the existence of other resources. Therefore, ***Tender*** can also preserve process elements instead of deletion of process elements. ***Tender*** can recycle preserved process elements in process creation. As a result, the cost of process creation and termination is reduced.

Process elements

A process is an object controlled by OS for ex-

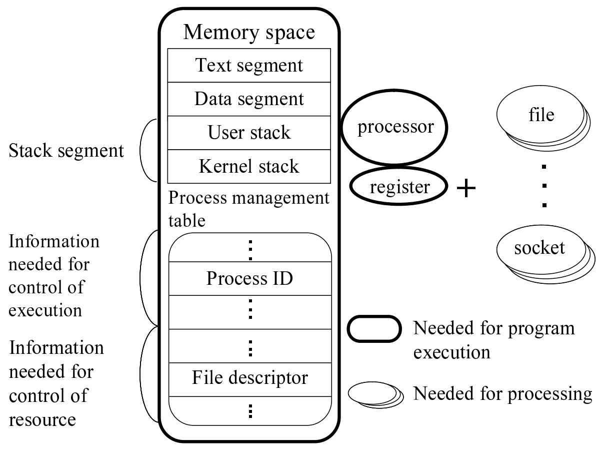

Figure 2. Process components.

ecuting the program. Various resources compose the process. Figure 2 shows the components of process. The components of a process are a program image, a process management table, needed for program execution and needed for processing. A program consists of a text segment, a data segment, a BSS and a stack segment. A text segment has the set of instructions that a processor can execute. A data segment has the set of variables and character strings with initial values, whereas BSS has a set of variables without initial values. A stack segment consists of a user stack and a kernel stack, which are used in user or kernel mode respectively. A process management table has information needed for control of execution and for control of resources. The former includes virtual storage space, processor and a set of register, etc. and the latter includes file and socket, etc.

Memory Resources on *Tender*
Figure 3 shows memory resources on ***Tender***. In this figure, "virtual region" is a resource that virtualizes the data storage region information of the physical memory or the external storage. "Virtual space" is a space of the virtual address and corresponds to the mapping table where the virtual address is mapped into the physical address. "Virtual user space" is a space which is accessible from the processor by the virtual address. It is created by attaching "virtual region" to "virtual space" and deleted by detaching. Here, the attaching means to store the information of data storage region in the mapping table.

Process Creation
Figure 4 shows the flow of process creation on ***Tender***. The processing consists of twelve steps.

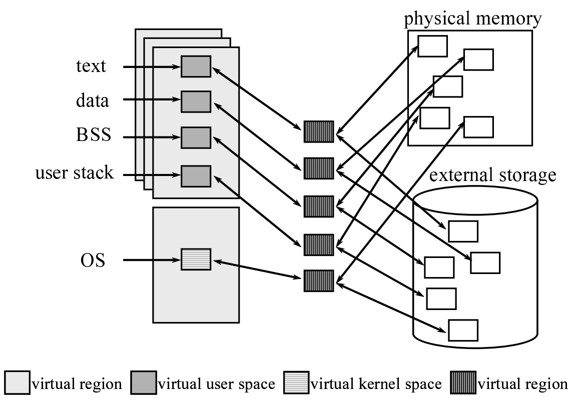

Figure 3. Resource of process and memory.

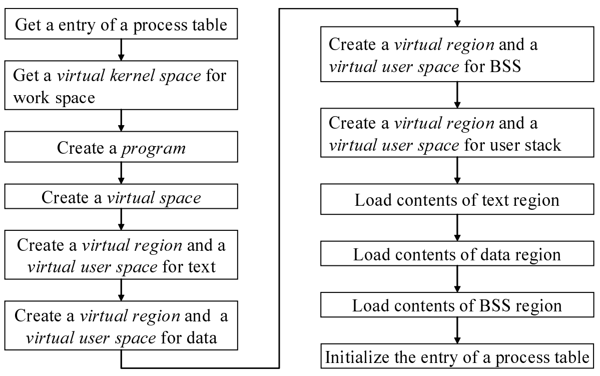
Get a entry of a process table
Get a virtual kernel space for work space
Create a program
Create a virtual space
Create a virtual region and a virtual user space for text
Create a virtual region and a virtual user space for data
Create a virtual region and a virtual user space for BSS
Create a virtual region and a virtual user space for user stack
Load contents of text region
Load contents of data region
Load contents of BSS region
Initialize the entry of a process table

Figure 4. Flow of process creation.

4 Recycle of Process Elements

Overview
Preserving process elements can be realized by the separation and the individualization of resources. Recycling the preserved resources can speed up the processing of process creation. We describe the mechanism of a fast process creation and termination as follows.

Figure 5 and figure 6 show the flow of process creation and termination. Table 1 shows the interface of process creation and termination. The interface of process creation is not changed for recycling process elements, because the process management in kernel searches and recycles process elements automatically. On the other hand, the interface of process termination is modified, because users need to designate which resources should be preserved.

Tender enables resources to be preserved on process termination, which means that in case we want to release a resource we can preserve it without actually deleting it. In addition, it enables resources to be prepared on process creation, which means that in case we want to get a resource we can recycle the resource preserved in advance without creating it. Therefore, the processing is expected to be faster when "resources"

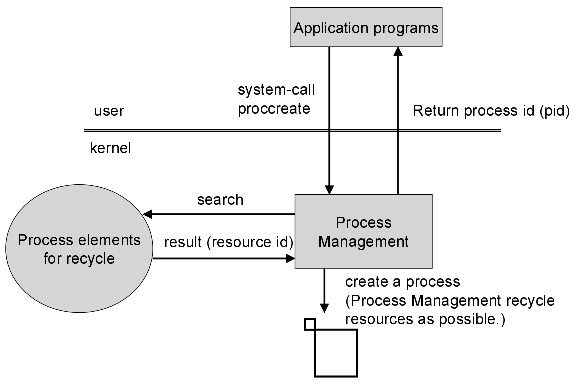

Figure 5. Flow of process creation.

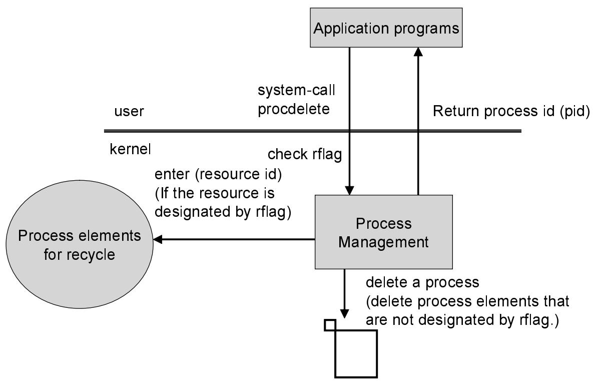

Figure 6. Flow of process termination

are used or released. That is, "resources" which had been used before process termination are not deleted but are preserved. Moreover, preserved "resources" can be recycled at the next time of process creation. As a result, the processing time of "resource" creation or deletion can be faster.

Two Types of Preserved Resources

We proposed efficient resource management for recycling process elements[2]. We focused on frequency in program execution. There are two types of preserved resources for recycling from the standpoints of frequency in program execution. Table 2 shows the types of preserved resources.

On programs with high frequency in program execution, process elements are preserved with program image (called type1). Thus preserved process elements with program image can be recycled when a process is created from the same program. The preserved process elements with program image consist of virtual memory, memory image of the program on virtual memory space. The memory image includes memory space of the text region, the data region, the bss region and the stack region. The contents of each region are loaded on the memory space.

On programs with low frequency in execution, process elements are preserved without program image (called type2). The process elements of programs with low frequency are broken up into virtual memory, unmapped memory objects (virtual regions). There is no memory object on the preserved virtual memory. The unmapped memory object consists of physical memory and disk area on disk and management table of mapping between physical memory and disk. Because virtual memory and unmapped memory objects have no information associated with a particular program image, they are recycled for every process creation if they satisfy following conditions. Virtual memory can be recycled for every process creation. Unmapped memory objects can be recycled, if the size of an unmapped memory object is same as a requested size.

Table 1. Interface of process creation and termination.

open_proc (plateid, arg, vmid)	plateid: an identifier of "plate" arg: a pointer of arguments vmid: an identifier of virtual space
close_proc (pid, rflag)	pid: an identifier of process rflag: a flag of preserved process elements

Table 2. Types of reserved process elements for recyclying.

Type	Condition of recycling
Reserved process elements with program image (type 1)	Type of program
Reserved process elements without program image (type 2)	Virtual space: always Virtual region: size of virtual region

5 Proposal Method

Problem

The proposed method[2] can reduce the amount of memory consumption of preserved resources. However, the method focused on frequency of program execution is insufficient to reduce it. In this method, the number of preserved process elements with program image for particular program is more than one. One of them is always recycled when the program is executed. However, the remains of them (more than 2) are not recycled if plural processes are not executed concurrently. Therefore, the remains of them may be useless and consume much memory.

Improved Resource Management

To solve the problem described above, only one process element with program image is preserved for each program with high frequency in program execution. When a preserved process element with process image of same program exists, the process is preserved as process elements without program image.

Advantages: (1) Process elements of type2 can be recycled for every process. Therefore, the ratio of the success of recycling to process creation can be improved.
(2) Proposed method improves the ratio of recycling to process creation with less amount of memory consumption of preserved process elements.

Disadvantages: The processing time of process creation and termination, increases a little when the processes of a program are executed concurrently. In the case, process elements of type1 can be recycled only once for process creation. On the other case, process elements of type2 can be recycled if they satisfy the condition of recycling.

Evaluation

To evaluate proposed method, we got a log of program execution by using the command of "acct" on a server computer of our laboratory. It consists of 1000 entries of program execution. We made a benchmark program that creates and deletes a process according to it. We can simulate the real sequence of program execution by using it on ***Tender***.

Processing time of process creation and termination and the amount of the memory consumption of preserved process elements are measured in the evaluation. We cannot get the results associated with the disadvantage of the old method [2] when the number of concurrent running process is one. We can get it when the number is more than one. To confirm our proposal's validity through evaluations, the number of concurrent running process is set one and three.

We performed the evaluations on four conditions as follows.

(1) Type1 only

(2) Type1 and type2 (old method)

(3) Type2 only

(4) Type1 and type2 (proposal)

Figure 7 and figure 8 show the result when the number of concurrent running process is one.

(1) The processing time of type1 is the shortest in figure 7, when the number of program execution increases, because in this case process elements of type1 can be recycled. Processing time of process creation is the shortest when process elements with program image are recycled.

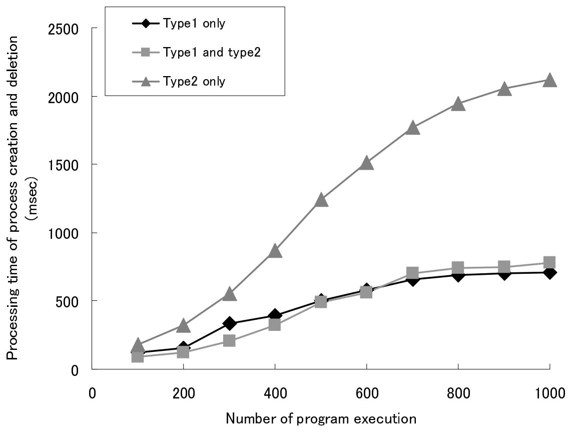

Figure 7. Processing time of process creation and termination. (Number of a process is one)

(2) Preserved process elements of type1 only consume memory more than other two conditions.

(3) Preserved process elements of type2 only consume memory less than other two conditions, but the processing time of type2 only is the longest.

(4) Preserved process elements of type1 and type2 consume less memory. Besides, the processing time of them is relatively short.

That is summarized as follows. Our proposed method (type1 and type2) is reasonable, because it realizes fewer amount of memory consumption and fast process creation and termination.

Figure 9 and figure 10 show the result of the number of concurrent running process is three.

Figure 9 shows the processing time of our proposal is the shortest, because in this case process elements of type1 are preserved for each program. Besides, if there is no process elements of type1 for target program, process elements of type2 can be recycled. This method restricts the number of process elements of type1. Thus, most of processes are broken into process elements of type2. Figure 10 shows them the amount of memory consumption of proposal method is two / three of the old method.

Figure 9 and figure 10 show proposal method consume memory more than type2. However, the difference is small and the processing time of proposal method is about half of type2.

Considering from processing time and the amount of memory consumption, proposal method balances the amount of memory consumption and the processing time of process creation and termination.

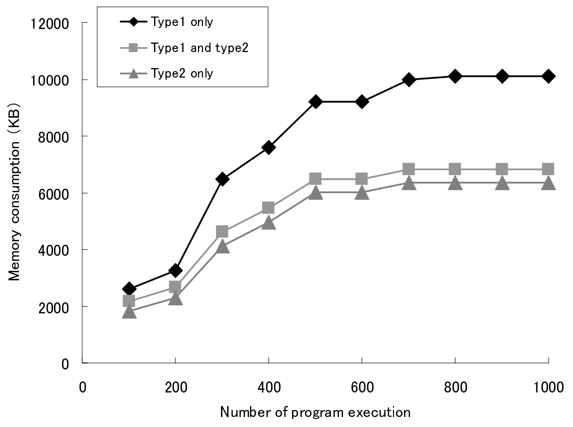

Figure 8. Memory consumption of preserved process elements. (Number of a process is one)

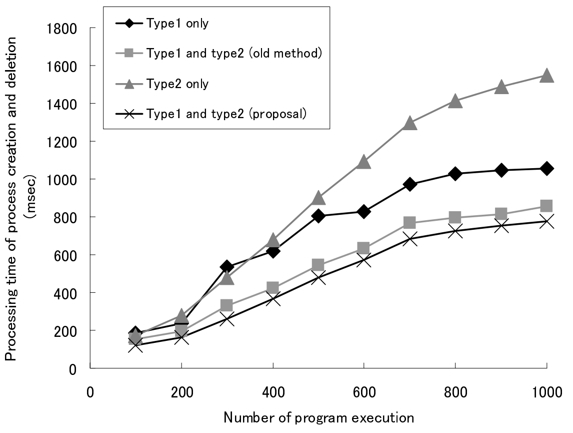

Figure 9. Processing time of process creation and termination. (Number of a process is three)

6 Conclusion

We present an improved resource management method for recycling process elements. In the method, only one process element with program image is preserved for each program with high frequency in program execution. When a preserved process element with process image of same program exists, the process is preserved as process elements without program image. Process elements without program image can be recycled for every process. Therefore, the ratio of the success of recycling to process creation can be improved. Proposed method improves the ratio of recycle to process creation with less amount of memory consumption of recyclable process elements.

We also report the results of the evaluation of proposal method. They show our proposed method (type1 and type2) is reasonable, because it realizes less amount of memory consumption and fast process creation and termination. Considering from

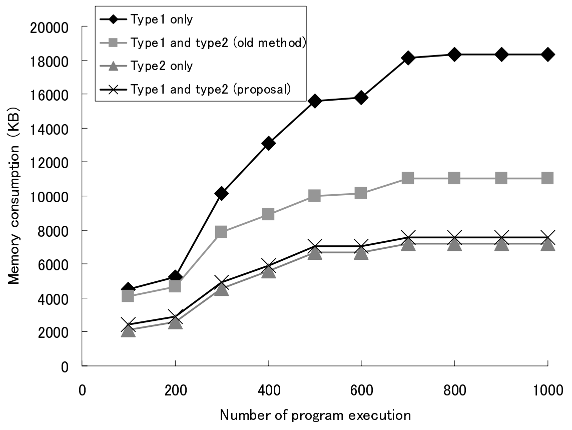

Figure 10. Memory consumption of preserved process elements. (Number of a process is three)

processing time and the amount of memory consumption, proposal method balances the amount of memory consumption and the processing time of process creation and termination.

As a future work, we will evaluate our proposal method by using real application programs.

References

[1] H. Taniguchi, Y. Aoki, M. Goto, D. Murakami, T. Tabata, "***Tender*** Operating System based on Mechanism of Resource Independence," **IPSJ Journal**, Vol. 41, No. 12, 2000, pp.3363–3374.

[2] T. Tabata, H. Taniguchi, "Proposal of Efficient Resource Management for Recycling Process Elements," **IPSJ Transactions on Advanced Computing Systems**, Vol. 44, No. SIG 10(ACS2), 2003, pp.48–61.

[3] J. S. Quarterman and A. Silberschatz and J. L. Peterson, "4.2BSD and 4.3BSD as Examples of the UNIX System," **ACM Computing Surveys**, Vol. 17, No. 4, 1985, 379–418.

[4] A. Braunstein, M. Riley, J. Wilkes, "Improving the Efficiency of UNIX File Buffer Caches," **SOSP: 12th ACM Symposium on Operating Systems Principles**, 1989, pp.71–82.

[5] R. H. Patterson, G. A. Gibson, E. Ginting, D. Stodolsky, J. Zelenka, "Informed Prefetching and Caching," **SOSP'95: 15th ACM Symposium on Operating Systems Principles**, 1995, pp.79–95.

[6] V. S. Pai, P. Druschel, W. Zwaenepoel, "IO-Lite: A Unified I/O Buffering and Caching System," **OSDI'99: USENIX Association 3rd Symposium**, 1999, pp.15–28.