User Process Resource Usage Measurement for Grid Accounting System

HoJeon Hwang, BeobKyun Kim, GilSu Doo*, DongUn An, SeungJong Chung
Dept. of Computer Engineering, Chonbuk National University, Korea
Tel : +82-063-270-2412 Fax : +82-063-270-2394 E-mail: hjhwang@duan.chonbuk.ac.kr
*Dept. of Electrical & Electronic Engineering, Seonam University, Korea
Tel : +82-75-999-8765 Fax : +82-270-2394 E-mail: dgs@seonam.ac.kr

Abstract: Grid computing environment can be used to interconnect a wide variety of geographically distributed heterogeneous computing resources based on high-speed network. To make business service, it is necessary for Grid accounting system to measure the computational cost by consuming computer resources. To collect resource consumption data, and to keep track of process without needing to recompile kernel source, we use system call wrapping. By making use of this technique, we modifies system call table and replace existing system call to new system call that can monitor processes running in CPU kernel currently. Therefore we can measure user process resource usage for Grid accounting system.

Grid Accounting, Resource Usage, System Call Wrapping

1. INTRODUCTION AND MOTIVATION

Grid [1] emerging in the middle 1990's began to solve problems in the field of science and engineering under distributed computing environment, but currently offer opportunities to construct meta-computing environment that enables to user to uniformly access geographically distributed heterogeneous computing resources, secondary storage systems, diverse research devices and other shared resources via advanced optical networks.

Grid offers a way to solve computational grand challenge problems like protein folding, drug discovery, financial modeling, earthquake simulation and weather modeling that are not executed on single machine. Also Grid is able to analyze and organize scalable distributed data, and Grid enables to allocate resource dynamically on demand of user level and user's job requirement.

When the Grid computing environment becomes fully operational, many of Grid users will be actively using Grid resources provided by supplier for their solving problems. Multiple of suppliers will make idle resources to be participated in the Grid that can be utilized efficiently too, and charge accounting fee of resource consumption to each of users accessing these resources. Therefore Grid accounting system is necessary to make Grid service business [2][3].

It is insufficient for Grid accounting system to charge the cost of using only information provided by accounting applications of UNIX-like operating system. Our goal in this paper is to measure diverse resource usage data at process-level. We use system call wrapping technique so that we implement monitoring component to collect resource consumption data and to keep track of all processes.

2. RELATED WORK

Resource Usage Service Working Group(RUS-WG)

The framework of Grid Service Accounting Extensions (GSAX) [8] proposed by RUS-WG within GGF (Global Grid Forum) is composed of a number of core blocks. These blocks are monitoring, metering and accounting [8].

Monitoring block is to collect raw resource and service data associated with a service call; to provide feedback to the service on resource usage, and to control the service based on feedback from higher-level blocks. Metering block is to apply a charging policy to service usage; and to request charging for service usage. Accounting block is to manage a business relationship between the consumer and service provider via an accounting policy. Each of blocks can be combined in various ways to provide different architectures, suitable for various purposes.

Usage Record Working Group

Sites in the Grid must be able to exchange resource usage data and basic accounting information in a common format for resources to be shared. This working group provides information to the Grid community regarding the usage record format requirement. This working group focuses on the representation of resource consumption data by separating between basic field properties and meta field properties. And specially defines necessary common usage record fields. In this paper we have incorporated the latest recommendation of a minimal Usage Record from the UR-WG[6]

3. RESOURCE USAGE MEASUREMENT

3.1 Grid middleware

The Globus toolkit software [6][9] is one of the most popular Grid middleware that offers resource management, data management, and information services independently. Globus is utilizing X.509 based authentication mechanisms to successfully deploy a computational job across a set of supercomputer systems. Grid resource provider sites maintain a file that maps from X.509 DNs(Distinguished names) into a local account identifier that was created for the individual user. Our system was implemented based on Globus toolkit.

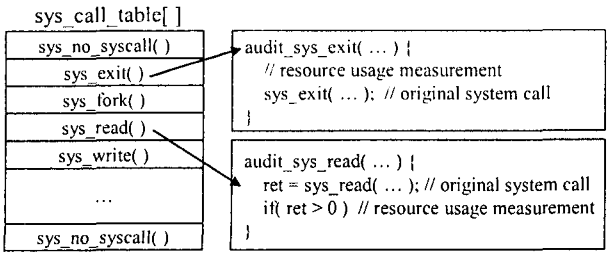
3.2 Local user to run job on a machine

The current situation at Grid sites is that to run jobs on a machine, the user needs to have a local account on that machine[4][5]. Thomas J. Hacker [2] proposed the mechanism that replaces the fundamental systems paradigm of a strong binding between a real user and an account on a system with the paradigm a temporary binding between an account and a real user. UNIX systems don't support temporary account bindings.

When a certain job was submitted from authenticated users, we did have one-to-one mapping between Grid user's DNs to local account for the duration of the job. This approach benefits from the method that can collect resource usage information of mapping user account when a job is completed.

3.3 How to keep track of processes

As shown in figure 1, to keep track of process, we specially try to use system call wrapping method, which modifies system call table and replace existing system call to new system call. A kernel module is a piece of code that adds functionalities to the kernel without needing to recompile or reboot the system. Module can be dynamically loaded or unloaded with the commands insmod and rmmod. You can have to be root to load modules in the kernel so your system can't be compromised because of modules. Modules are not directly related to security but as they are executed in the kernel space, they have full control the running system. Therefore, modules can intercept system calls, access all files and audit any process.



```
audit_sys_exit( ... ) {
    // resource usage measurement
    sys_exit( ... );  // original system call
}
```

```
audit_sys_read( ... ) {
    ret = sys_read( ... ); // original system call
    if( ret > 0 )  // resource usage measurement
}
```

Figure 1. System call wrapping module

So, by wrapping system call using kernel module programming mechanisms, system administrator can audit all processes associated with a particular user as well as measure resource usage data of Grid service process. The resource consumption data can be measured by checking return value from system call function in kernel space. When function is completed successfully, we record argument and return value of this system call function. Also we refer to task structure of process to obtain additional data as shown figure 2.

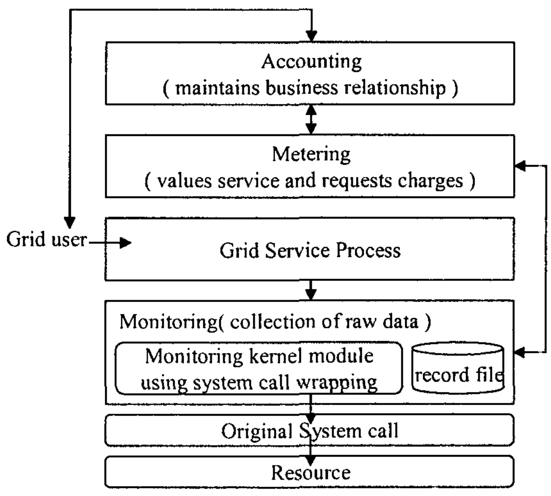

Accounting
(maintains business relationship)
Metering
(values service and requests charges)
Grid user
Grid Service Process
Monitoring(collection of raw data)
Monitoring kernel module using system call wrapping
record file
Original System call
Resource

Figure 2. Monitoring kernel module

3.4 How to collect resource consumption data

Currently, process accounting application allows you to keep track of the accounting information about each process run on machine. Every time a process dies, the exit program collects process resource consumption data and writes it to the log file.

Unfortunately, it is not sufficient to do service accounting information supported by operating system. To provide users with Grid accounting service, accounting information for Grid service consist of not only computational resources usage data, but also diverse hardware and software resources usage data. So our system can collect hardware resource consumption data as well as determine whether there is some applications execution or not as shown as shown table 1.

Table 1. Resource consumption data list

Process accounting information	user and group ID of process owner
	process ID and command name
	process start time
	memory used
	CPU time used(user and system time)
IO Accounting information	standard input, output and error
	amount of disk storage accessed
	amount of transfer data via socket
Etc	special applications

Process Accounting Data

To collect process accounting information on a machine that job runs, resource usage information used by corresponding process was measured referring to task process structure just before the exit of process. We extract information about accounting from process descriptor pointer of the currently executing process in kernel mode when process exit. The pointer to the process descriptor is obtained from current macro.

I/O Accounting Data

As shown in figure 3, the standard UNIX file system allows VFS (Virtual File system) to uniform all file system types, for example, regular file, socket, pipe, and etc. The VFS offer user to access any file system in common interface. Therefore, to determine which file system was accessed, we audit read and write system call that measure the amount of resource usage data.

Note that file descriptor 0 is normally standard input, 1 is standard output and 2 is standard error output. In other case, by tracking the inode structure of file descriptor that is open by process currently running on the system, we can identify the determination whether some process access regular file or socket.

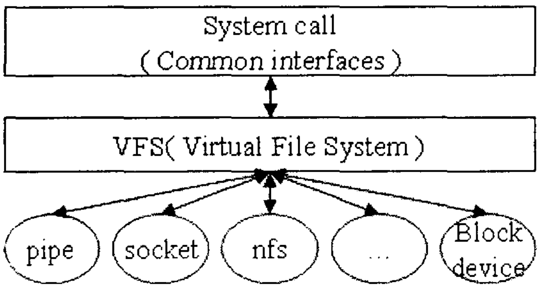


Figure 3. UNIX files system and their interfaces

Special Applications

To determine which application is running which process, we can compare special application name with the file name of open file descriptor that referred to the dentry structure of process file descriptor

4. IMPLEMENTATION AND RESULTS

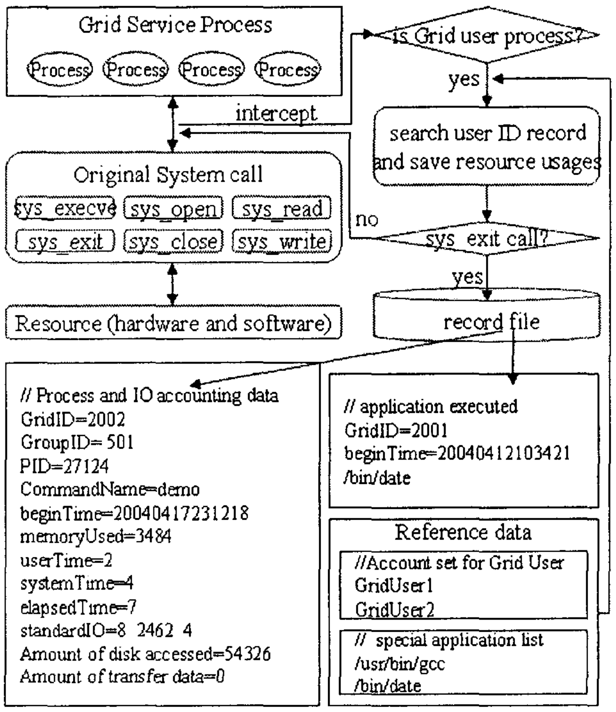
When a Grid job was submitted by authenticated user, to begin with local account and Grid user's DN correctly added to grid-mapfile in the machine where the job was executed. We can remove mapping information in grid-mapfile when a Grid job has been finished.

To collect resource consumption data, system administrator load module in the kernel so make kernel module to write all resource usage information consumed by Grid user account for executing a Grid service, into a specific log file as shown figure 4 Module can be dynamically loaded or unloaded with the commands 'insmod' and 'rmmod'. You can have to be root to load modules in the kernel so your system can't be compromised because of modules. Modules are not directly related to security but as they are executed in the kernel space, they have full control the running system. Therefore, modules can intercept system calls.

This system was implemented with the following restrictions to measure Grid user process resource usage.

1. Local accounts for Grid service can't log in to the machine at all.
2. This mapping is only one-to-one binding to identify multiple Grid user service process

This system was experimented in ANSI C on a Pentium installed Globus Toolkit 2.0 under Redhat Linux server, kernel 2.4.2-2. Monitoring module collect the only the Grid user process resource usage. In the kernel space, our monitoring kernel module check process owner and compare grid user's id with process owner's id running in CPU currently. If user id of process equals grid user id, then this monitoring module save all collected resource usage data used by corresponding process into the kernel memory and write process usage field into the log file just before the exit of user process.



```
// Process and IO accounting data
GridID=2002
GroupID= 501
PID=27124
CommandName=demo
beginTime=20040417231218
memoryUsed=3484
userTime=2
systemTime=4
elapsedTime=7
standardIO=8  2462  4
Amount of disk accessed=54326
Amount of transfer data=0
```

```
// application executed
GridID=2001
beginTime=20040412103421
/bin/date
```

```
//Account set for Grid User
GridUser1
GridUser2
```

```
//  special application list
/usr/bin/gcc
/bin/date
```

Figure 4. System overview and Grid job's resource usage example

Figure 4 shows that resource usage example that Grid job process consumed. Monitoring kernel module decides whether an application was executed or not depending on execve system call of process. And the mount of disk accessed and transferred data through socket hooks read and write system call. And process accounting data was referred to task_struct structure.

5. CONCLUSION AND FUTURE WORK

Resource owner provides the idle resources in the Grid computing environment; resource consumer can obtain the powerful computing service anytime and anywhere using Grid service [1][6]. In order to make Grid business service, we provide a mechanism to measure the account of the amount that the consumer used and the resource owner supported. This method is system call wrapping.

There are many of characteristics associated with Grid accounting information system in our goal. Their characteristics are as follows:

1. Powerful auditing and monitoring capabilities per process.
2. Collecting diverse resource consumption data.
3. Providing collected data to higher-level blocks.

This method of wrapping system call is of benefit to collect resource usage data used by Grid service process, but because of auditing all processes and comparing all file descriptor structure in the kernel space during measurement operation; this will cause system performance decrease. For future work, investigation of not checking all processes but trace the only single process that deal with Grid service should be done. And payment mechanism to meter each of resources value should be attempted.

Reference

[1] I. Foster and C. Kesselman, "The Grid : Blueprint for a New Computing Infrastructure", Morgan Kaufmann Publishers, USA, 1999

[2] Thomas J. Hacker, Brain D. Athey, Q 16 "Accounting Allocations on the Grid", Center for Parallel Computing University of Michigan, 2000.

[3] A. Beardsmore et al, "GSAX(Grid Service Accounting Extensions)", draft, GGF6, 2002.

[4] Rodney Mach, "Accounting Interchange Natural Language Description(Requirements)",Usage Record Working Group, April-04-2004.

[5] Thomas J. Hacker and Brain D. Athey. 'A methodology for account management in grid computing environments", In Craig A. Lee, editor, Proceedings of Grid Computing GRID 2001, Second International Workshop, Denver, CO, USA, November 12, 2001, volume 2242 of Lecture Notes in Computer Science, pages 133-144. Springer, 2001.

[6] I. Foster, et al, "Globus: A Metacomputing Infrastructure Toolkit", Intl. J. Supercomputer Applications, 1997.

[7] Auditing and Accounting on AIX SG24-6020-00 Redbook, published October-24-2000.

[8] Global Grid Forum, http://www.ggf.org

[9] Globus Toolkit, http://www.globus.org