Grid Access Control System for Site Autonomy

Beob Kyun Kim*, Haeng Jin Jang**, Gil Su Doo***,
Ho Jeon Hwang*, Dong Un An*, Seung Jong Chung*

* Dept of Computer Engineering, Chonbuk National University
** Korea Institute of Science and Technology Information
*** Dept of Electrical & Electronic Engineering, Seonam University

Tel : +82-063-270-2412 Fax : +82-063-270-2394 E-mail: kyun@duan.chonbuk.ac.kr
hjjang@kisti.re.kr, hjhwang@duan.chonbuk.ac.kr, doogilsu@naver.com,
{duan, chung}@chonbuk.ac.kr

Abstract: The term "Grid" refers to systems and applications that integrate and manage resources and services distributed across multiple control domains. Resource sharing is, necessarily, highly controlled, with resource providers and consumers defining clearly and carefully just what is shared, who is allowed to share, and the conditions under which sharing occurs. In this paper, we design and implement a grid access control system, called PGAM. This system works on heterogeneous resources, can be applied to the additional service development and its service, provides template account management mechanism, and tries to ensure site autonomy. This system is implemented to increase portability and to be fit to any kind of economic model.

Grid, Globus, Access Control, Site Autonomy

1. INTRODUCTION

Grid computing has emerged as an important new field, distinguished from conventional distributed computing by its focus on large-scale resource sharing, innovative applications, and, in some cases, high-performance orientation [1][2]. The term "Grid" refers to systems and applications that integrate and manage resources and services distributed across multiple control domains [1].

The sharing and coordinated use of distributed resources is fundamental to an increasing range of computer applications, ranging from scientific collaboratories to healthcare. This sharing may involve not only file exchange but also direct access to computers, software, data, and other resources, as is required by a range of collaborative problem-solving and resource-brokering strategies emerging in industry, science, and engineering. This sharing is, necessarily, highly controlled, with resource providers and consumers defining clearly and carefully just what is shared, who is allowed to share, and the conditions under which sharing occurs. So, the grid access control system that covers these concepts and requirements is one of core elements in the grid environments [3][4][5][6].

In this paper, we design and implement a grid access control system, called PGAM (Policy based Account Manager). This system works on heterogeneous resources and can be applied to the additional service development and its service. PGAM uses globus toolkit as its default middleware which is the most widely adopted grid middleware in the world. The requirements, mentioned above, are covered in this system. PGAM tries to support site autonomy, a factor which encourages a site to get into the grid environment, and provides template account mechanism.

2. ACCESS CONTROL IN GLOBUS TOOLKIT

Globus Toolkit is one of the most widely adopted grid middleware in the world. Globus toolkit comprises a set of components that implement basic services for resource management, information service, data management, grid security, etc. GRAM (Grid Resource Allocation Manager) is responsible for access to remote resources, co-allocation of distributed resources, and processing of heterogeneity of resource management.

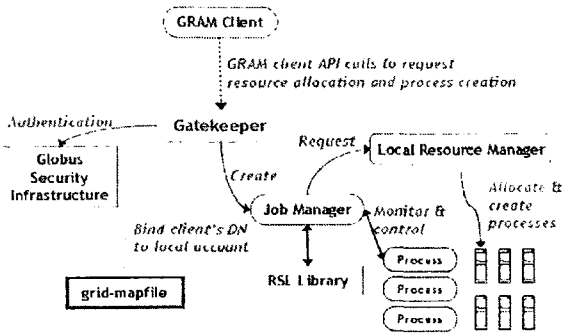

Fig. 1 Major components of the GRAM

The gatekeeper is an extremely simple component that responds to a request by doing three things:

performing mutual authentication of user and resource, determining a local user name for the remote user, and starting a job manager which executes as that local user and actually handles the request. Normally, when a request for access is received, the gatekeeper attempts to find the corresponding local username in the "grid-mapfile." This file lists pairs of certificate subjects (e.g., "/O=Grid/O=Globus/OU=hi.ac.kr/CN= hdg") and usernames (e.g., gw1). If no line is found for the current subject, access request is denied.

```
"/O=Grid/O=Globus/OU=hi.ac.kr/CN=hdg" gw1
"/O=Grid/O=Globus /OU=hi.ac.kr/CN=how" gw2
"/O=Grid/O=Globus /OU=bye.com/CN=say" gw2
"/O=Grid/O=Globus /OU=bye.com/CN=mist" gw3
```

Fig. 2 An example of "grid-mapfile"

In the original syntax of this file, several certificate subjects can be mapped to one local username. But, this mechanism cannot guarantee end-to-end user identity: who is the owner of local process or job, if there are several certificate subjects mapped to one local username. If the site administrator wants to trace the usage of local resource, he must deploy other monitoring or tracing tool which is implemented by kernel programming. Sharing of the same right to local files, directories and mails by multiple grid users can cause security problem, digging into privacy.

3. GRID ACCESS CONTROL SYSTEM

3.1 Architecture of Grid Access Control System

To guarantee end-to-end user identity, 1-to-1 mapping of certificate subject and local username is preferred to n-to-1 mapping. But 1-to-1 mapping can cause a heavy load on the site administrator and local system. So, we implement template account mechanism [9] to reduce the burden of administrator and local system.

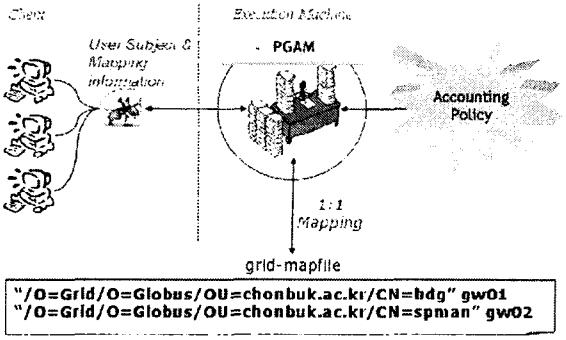

Fig. 3 Function of Grid access control system

When a grid user requests a right for access to local resource with his credential and personal information and job specification, PGAM creates new thread to process it. Each thread has following sub-modules (Figure 4).

- client interface :
  process interaction from client
- logger :
  logging all the record during operation
- configuration file reference :
  reference to configuration files including local resource management policies (e.g., pool of available local usernames, client host policy, stage policy, personal information policy, etc..)
- server status reference :
  reference to status of system (e.g., current available local username list, active username, system memory, disk, etc..)

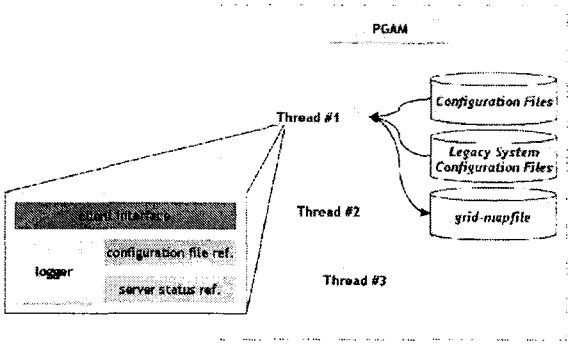

Fig. 4 Architecture of PGAM

3.2 Policy Enforcement for Site Autonomy

When making a request to the resource, the grid user presents his credential, his personal information, and his job specification. Upon receiving these information, PGAM takes several steps to enforce local policy:

- Verify the validity of user's credentials (e.g., signature, time period).
- Enforce the site's policies regarding user's personal information.
- Enforce the site's policies regarding user's job specification and resource's status. For example, if user request 32 processors but system has only 16 processors for the class of user, then user's request is denied.

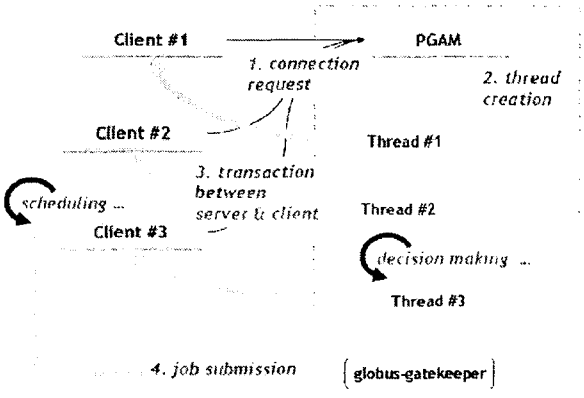

Fig. 5 Steps for job submission

PGAM is intentionally simple, with the minimum intervention to usual resource access in the grid environment. Authorization of user is made by adding a pair of user's certificate subject and local username to "grid-mapfile". To ensure the end-to-end user identity, PGAM selects new username that is not listed in the "grid-mapfile" from the pool of local usernames. User can then submit his job with globus toolkit (Figure 5).

System administrators need to do only following works: making a pool of user accounts for use with different rights, editing configuration files of PGAM. By modifying configuration files and controlling a pool of user accounts, system administrators can enforce local resource management policies. Figure 6 is an example of configuration files of PGAM. System administrators can control grid user's rights with system configuration files. If rights of a local username are changed in "limits.conf" file, a grid user who uses that local username is restricted to changed rights.

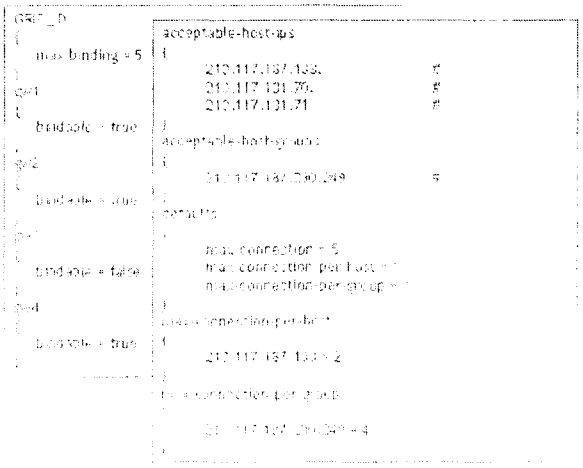

Fig. 6 An example of configuration files for local available usernames and client host policy.

3.3 Management of Authorization Information

Information produced during authorization process may be useful to other additional service. Especially for the accounting information service, authorization information can be used for determination of the owner of each process.

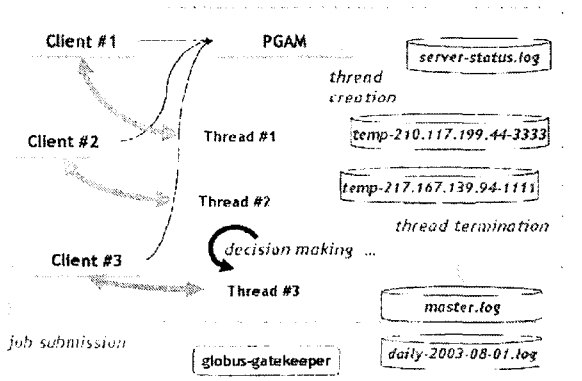

Fig. 7 Management of Authorization Information

So, this system creates authorization information as much as possible: including user's personal information, user's credential information, requested job specification, authorized local username, authorized time, lifetime, etc. Each thread manages records produced from its connection. When the thread is destroyed, its records are moved to the master log. The master log can be processed into daily, weekly, monthly, or yearly log data.

4. EXPERIMENTS AND RESULTS

PGAM was implemented in the following environments (Table 1). To increase portability, we use python and java and their related drivers.

Table 1 Experimental Environments

	Environments
Tested OS	Redhat Linux 7.2, 7.3, 8.0, 9.0 IBM AIX 5.1L
Language	Java 1.4, Python 2.2, wxPython mod_python-4.1.1

Figure 8 shows information when the request for access is accepted. This authorization information is logged and processed daily, weekly, monthly, and yearly information. Other additional services may use this information.

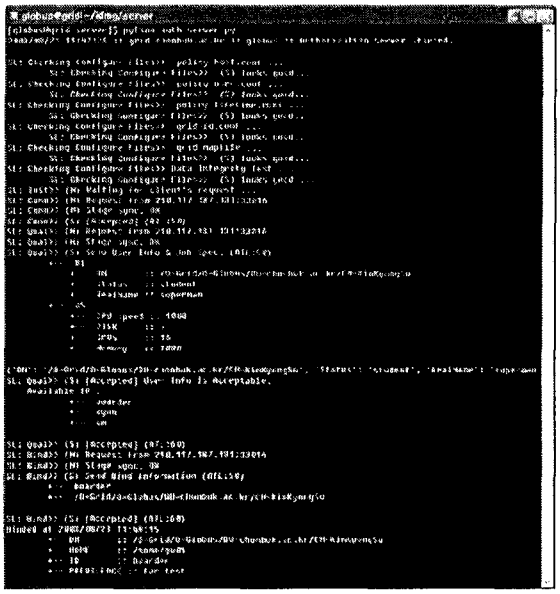

Fig. 8 Authorization information (Success)

Figure 9 shows a denied message for the reason of user's personal information.

```
CL: Critical Error!!>> Qualification of User Info. Failed.
    [Denied] Not Allowable User Information.
```

Fig. 9 An example of a denial message

Figure 10 shows an abstract expression of authorized message. In this message, local usernames which grid user can use is listed. Now, grid user can

get rights of one of the listed usernames.

```
CL: Qual>> (S) [Accepted] (ATL:60)
( gw02 : ( DN : , STAT : 0, PSTAT : 1, LIFETIME : 0, HOST : , PORT : ) )
( gw03 : ( DN : , STAT : 0, PSTAT : 0, LIFETIME : 0, HOST : , PORT : ) )
```

Fig. 10 An abstract expression of authorized message

Figure 11 shows a client program, coded with python language.

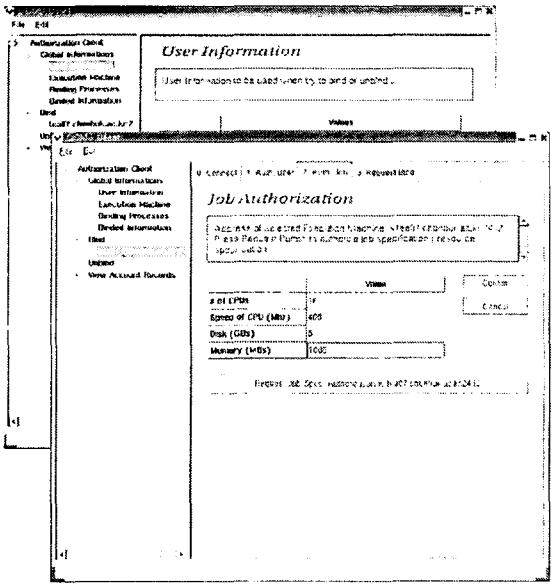

Fig. 11 A GUI version of client interface

5. CONCLUSION

In this paper, we design and implemented a grid access control system, called PGAM. PGAM uses globus toolkit as its default middleware which is the most widely adopted grid middleware in the world. We provide various configuration options to support site autonomy, a factor which encourages a site to get into the grid environment. PGAM is intentionally simple, with the minimum intervention to usual resource access in the grid environment. Python language is selected to increase portability. PGAM is independent of any kind of platform and of any economic model. PGAM is a good and simple tool to get access rights when a user tries to schedule his job in the grid environment.

The current PGAM does not support VO (Virtual Organization) related concepts and policies. We plan to add VO related services.

References

[1] Foster, C. Kesselman(eds), *The Grid : Blueprint for a New Computing Infrastructure, Mogan Kaufmann Publishers*, 1998.

[2] Foster, C. Kesselman(eds), S. Tuecke "The Anatomy of the Grid: Enabling Scable Virtual Organizations", Intl. J. Supercomputer Applications, 2001.

[3] S. Mullen et al, "Grid Authentication, Authorization and Accounting Requirements Research Document", (draft), GGF8, 2003

[4] Sebastian Ho, "GridX System Design Documentation", (draft), Bioinformatics Institute, 2002

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

[6] R. Baker et al, "Conceptual Grid Authorization Framework and Classification", (draft), GGF8, 2003

[7] BeobKyun Kim et al, "Design of Distributed Grid Accounting Platform for Site Autonomy", *Proc. of the 20th KIPS Fall Conference*, 2003

[8] K. Czajkowski, I. Foster, et al, "A Resource Managment Architecture for Metacomputing Systems", *Proc. of the 4th Workchop on Job Scheduling Strategies for Parallel Processing*, 1998

[9] Thomas J. Haker, Brian D. Athey, "Account Allocations on the Grid", Center for Parallel Computing University of Michigan. 2000.

[10] http://www.girdforum.org

[11] http://www.globus.org

[12] http://www.gridforumkorea.org