SUPERNODE II: A GRID NODE IN EDAGRID

Thoai Nam, Tran Dang Khoa
University of Technology, VNU-HCM

1.INTRODUCTION

Nowadays, the computing demands increase rapidly because many applications in fields like physics, aerospace, life science, economics, etc. are using modeling, simulation and analysis. To fulfill these computing demands, parallel and distributed computing is often the only solution. With the decreasing price of PCs, and the increase in bandwidth and the speed of communication networks, the cluster model becomes more and more popular. Moreover, clusters are easy to build, flexible and scalable for budget and application demands. The performance/cost ratio of a cluster is often far less than that of a comparable multiprocessor computer. Supernode II is a PC-based cluster with 128 processors developed at Ho Chi Minh City (HCMC) University of Technology. It provides cost-effective computing power for scientific applications [7] by supporting both sequential and parallel applications. Supernode II runs its own management software in order to use its resources efficiently.

In addition, Grid technology allows connecting computing resources together, and sharing them with many users. For example, our campus grid will help us to share the computing power of clusters, servers and computer labs in a university like HCMC University of Technology. Such a campus grid is developed in the EDAGrid project, where a campus Grid middleware based on GT4 to provide location-transparent computing/executing services with high level resource requirements.

Several clusters using different management tools like PBS [4], LSF [1] and SGE [2] have already been integrated into the campus Grid successfully because of the already support modules in GT4. However, clusters using other cluster management softwares such as Supernode II are not supported.

In this paper, a technique used to integrate Supernode II into GT4 is proposed. The solution is based on an extension for GRAM (Grid Resource Allocation and Management), which is an adapter between the grid framework and the underlying scheduler used in Supernode II. While originally intended for Supernode II, we believe that the same solution supports other cluster management softwares to be integrated into GT4 easily.

This paper is divided into five sections. Section 2 provides the basis of our approach by giving an overview on GRAM in GT4. A short introduction on Supernode II and the campus grid in EDAGrid project are shown in Section 3. Then the procedure to integrate Supernode II into Grid by GRAM is provided in section 4. The final section concludes the paper with comments on this work.

2.GRAM - THE GRID RESOURCE ALLOCATION AND MANAGEMENT COMPONENT

Grid computing resources are typically operated under the control of a scheduler which implements allocation and prioritization policies in order to optimize the execution of all submitted jobs. Grid Resource Allocation and Management (GRAM) is not a resource scheduler, but rather a protocol engine for communicating with many different local schedulers using a standard message format. The GRAM service supports submission, monitoring and control of jobs on the computers. It is meant to be used in situations where the ability to run arbitrary programs, achieve reliable operation, perform stateful monitoring, stage files, manage credentials, and interact with schedulers are required [5].

As illustrated in Figure 1, the primary elements of a GRAM deployment are:

- A set of services running in a GT4 Java container:
- GRAM-specific services for creating, monitoring, and managing jobs.
- A general-purpose delegation service, to manage delegated credentials.
- A general-purpose reliable file transfer (RFT) service, to manage data staging operations.
- A scheduler-specific GRAM adapter, to map GRAM requests into appropriate requests to a local scheduler.
- A GridFTP server used to execute data staging commands.

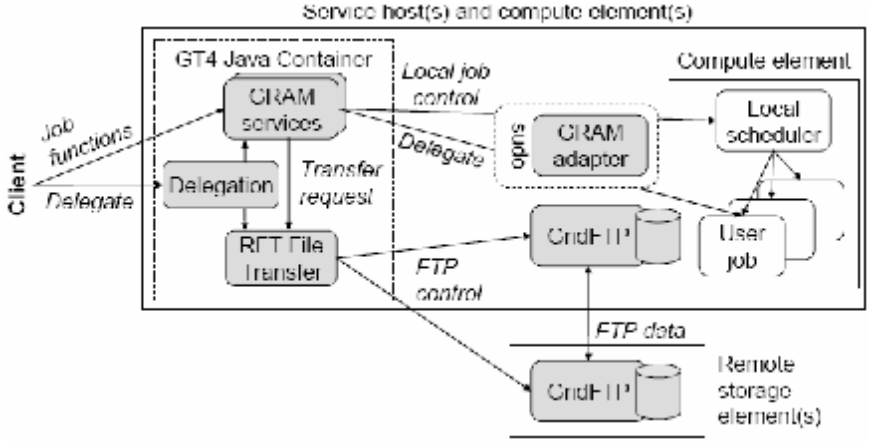

Figure 1. GRAM Implementation Structure

GRAM provides operations that enable Resource Manager to communicate with local schedulers. For instance, the staging operation may be used before and after the execution process to distribute the execution files and get the results. The delegation services help Resource Manager acquires the credentials of the user to allocate resources.

GRAM controls the life cycle of the job by using provided operations. When a client requests the submission of a job, it may be needed to request delegation (delegation when staging files or that job wants to perform some further GRAM job submission or other web services calls). After that, GRAM will create job and send stage-in request to File Transfer server. RFT server uses the credentials from Delegation to interact with GridFTP to transfer data. Then GRAM calls adapter to start execution. The adapter will call exactly the local scheduler depending on client's request. When execution finishes, process reverses from clean up, staging out to notifying completion [5].

Some modules in GRAM have to be modified to integrate a new local scheduler into the Grid:

The Scheduler Event Generator (SEG) module uses scheduler-specific monitors to report job state changes to the Managed Job Service (MJS). The SEG module implements a simple interface (by C language) to generate events from a scheduler. It relies on the scheduler job log file to determine when events should be issued. When the web service container fails and is restarted, the SEG module must be able to resume generating events from the last state that has been successfully completed.

The Scheduler Interface module contains the adapter script. This script is written in the Perl programming language following the proprietary GRAM adapter plug-in API. It implements the system-specific submission, job exit detection, job cancellation and (optional) job exit status determination processes.

These modules need to be adapted with Supernode II as describe below.

3.SUPERNODE II IN THE CAMPUS GRID

3.1.Supernode II

Supernode II is a PC-based cluster with 128 processors [7]. It is divided into two parts: the front-end and the computing nodes. The front-end is used to compile, test and debug programs, while the computing nodes are used to run programs. Supernode II is managed by its own management software, called Sysmanson [6]. Most functions seen in other cluster management softwares, such as submit, delete and monitor are supported in Sysmanson. Both sequential programs and parallel programs can be run on Supernode II.

3.2. EDAGrid — a campus Grid

The EDAGrid project is supported by HCMC National University. It aims at building the campus Grid middleware to provide location-transparent computing/executing services with high level resource requirements. Based on GT4 as the local resource portal, EDAGrid develops an information warehouse for service-centric applications. The EDAGrid is considered as one of the first Grid middleware projects in Vietnam.

The job execution of EDAGrid is shown in Figure 2. Users first describe their job requirements in an XML (Extensible Markup Language) format and submit this description to the Execution component. The Execution service performs resource discovery by accessing the WSRF (Web Service Resource Framework) Discovery service. In turn, the Discovery service needs to query qualified resources from the Information service. A list of resources will be matched against the user's requirements described in the job description and ranked on the levels of satisfaction. Upon receiving available resources, the Execution service makes reservation on the qualified grid node(s) through the Reservation service and finally launches the job with the given reservation on the corresponding Globus's WS-GRAM (Web Service GRAM).

Computing resources, called grid nodes, are allowed connecting and sharing in EDAGrid. It can be vary from small workstation to a robust cluster. Supernode II may be the most powerful grid node in this system. Because it is using Sysmanson as a management software and EDAGrid is using GT4 as grid middleware, so Supernode II is not integrated into EDAGrid.

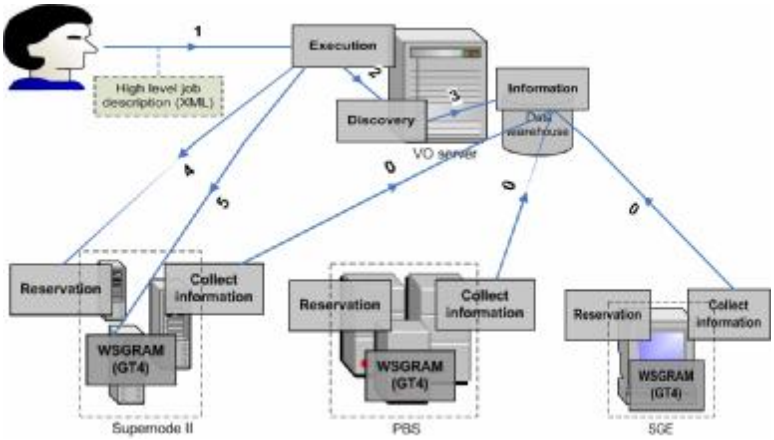

Figure 2. Supernode II in the campus Grid

4.THE INTEGRATION PROCESS

4.1.System Overview

When users submit jobs to the scheduler, the flow of request travels as indicated in Figure 3.

Firstly, clients intending to run a job contact GRAM. This module creates the environment for executing jobs, staging files to/from the environment, actual execution of jobs' processes, monitoring the execution, signaling important state changes to clients, and enabling clients to access the jobs' output files [5]. GRAM is an adapter between the grid framework and the

underlying scheduler, such as PBS, LSF and SGE. It converts the request from the grid to the corresponding form used by the individual resource managers. When running a job, GRAM needs to know current status of the job, which requires that the resource manager outputs the job state's changes to a log file. This log file is read by GRAM to determine the jobs state.

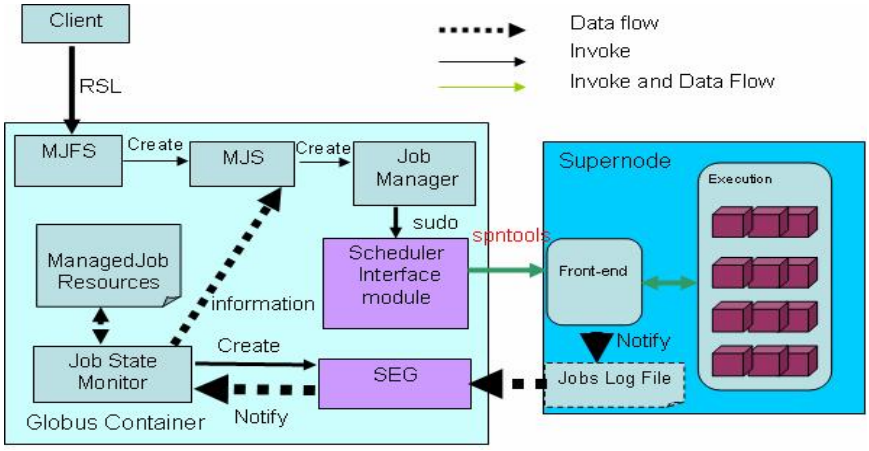

Figure 3. The data flow from grid to Supernode II scheduler

4.2.The Supernode II's Interface

To satisfy the requirements from the grid, each local scheduler must provide two basic functions: submission and cancellation. In addition, it may provide additional functions such as staging, polling job, sending signal, etc. Our own system, Sysmanson currently provides only three functions: submission, cancellation and job state logging.

4.2.1.Submission

Submission is the process to receive jobs from clients and to execute them. Of course, these jobs have to be scheduled on previously allocated resources in the most efficient way. The submission function in Supernode II requires a job description file, which differs from the RSL file used in GT4. It requires the job name, execution file, argument, and output location [6]. Therefore, it is necessary to convert the RSL file to the job description format used in Supernode II while integrating Supernode II into GT4.

4.2.2.Cancellation

The cancellation function allows canceling jobs which have been submitted. If a job has not been executed so far, it is simply removed from the scheduling queue. On the other hand, the system will send a signal to terminate the running job. The cancellation function in Supernode II requires a job description file, which indicates the identifier of the job that users want to stop.

4.2.3.Job state logging

While a job is running, the resource management software monitors and logs the job state changes to the log file. Each resource management system could use different formats to store these data, and consequently the SEG (Scheduler Event Generator) in GT4 may not understand this format. To integrate a cluster into the grid environment based on GT4, the format of the file must be understood by SEG module. The format used in Supernode II describes below, and the legends are shown in Table 1: `001;TIMESTAMP;JOBID;STATE;EXITCODE`

Table 1. Description of Supernode II's job log format

TIMESTAMP	The period of time when the event has been issued

JOBID	The local scheduler-specific job id
STATE	The requested new job state (integer as per the GRAM protocol constants)
EXITCODE	The exit code of the job, if STATE is done or failed

4.3.Integrating Supernode II into Grid through GRAM

In order to integrate Supernode II into a GT4 grid, the SEG module and Scheduler Interface module need to be modified as described below.

4.3.1.Scheduler Event Generator module

The SEG module provides information about job's status. It parses the job log file to determine job states and reports to the Managed Job Service (the module is implemented in C language). It is required to define the signature of this module, and to implement three main functions: activate, deactivate and callback.

The activate function is needed to start a module up. It sets up the variables, reads the configuration, and locates the log file. The deactivate function is invoked when the module is stopped. It cleans up memory, and removes the mutexes. The callback function is called periodically. It parses the log file, and reports to the Job State Monitor module whenever it observes that a job state is changed.

4.3.2.Scheduler Interface module

The Scheduler Interface module provides the interface for Managed Job Service to send requests to the underlying scheduler. It is also an adapter to convert requests from Job Manager into the form used by the local scheduler.

This module is implemented as a Perl module which is a subclass of the Globus::GRAM::JobManager module [3]. Its name must match the scheduler type string used when the service is installed. In the Supernode II scheduler, the module name is Globus::GRAM::JobManager::spn and it is stored in the file spn.pm. Though there are several methods in the Globus::GRAM::JobManager interface, only submit and cancel methods are necessary to be implemented in a scheduler module. These methods create the job description file and invoke the local scheduler to fulfill the request. Then they collect the execute result of the request to local scheduler.

5.CONCLUSIONS AND FUTURE WORK

Connecting computing resources into a campus Grid allows sharing and using the resources more easily and efficiently. This is a main goal in the EDAGrid project, where the Grid middleware is developed based on GT4. Many workstations, servers and clusters are connected to the campus Grid; and Supernode II with 128 processors is such a powerful grid node. Therefore, it has been of major importance to develop a solution for the integration of Supernode II into EDAGrid, which is necessary due to Supernode II's management software and that has not been supported by GT4. Our technique is based on using GRAM and modifying the software management of Supernode II. This useful piece of work makes the campus Grid developed in EDAGrid project being more powerful due to the participation of Supernode II. In the future, the reservation service will be supported on Supernode II. Moreover, additional useful information for monitoring will be provided by Supernode II.

REFERENCES

[1]. *Platform LSF (Load Sharing Facility*), http://www.platform.com.

[2]. *Sun N1 Grid Engine*, http://www.sun.com/gridware.

[3]. *WS-GRAM scheduler interface tutorial*, http://www.globus.org/toolkit/docs/4.0/execution/-wsgram/developer/scheduler-tutorial-seg.html.

[4]. Altair Grid Technologies, *PBS Pro Administrator Guide 5.4,* (2004).

[5]. Ian Foster, *A Globus Toolkit Primer*, (2005).

[6]. Tran Dang Khoa and Truong Nghia An, *Supernode II Technical Report*, (2006).

[7]. Nam Thoai, Tran Van Hoai, Thieu Quang Trung, Dang-Khoa Tran, and Truong Nghia An, *A High Performance Computing System: Supernode II*, In Proceedings of BK21 GSNU.IJU-HCMUT International Symposium on Transport Vehicle Engineering, Ho Chi Minh city, Vietnam, Dec (2005).