Design and Implementation of the Web-Based PSE *GridGate*

KyungWoo Kang, YunHee Kang, and KwangMoon Cho

Department of Computer and Communication Engineering,
Cheonan University, 115, Anseo-dong,
Cheonan 330-704, Choongnam, Republic of Korea
{kwkang,yhkang,ckmoon}@cheonan.ac.kr

Abstract. Grid-computing on networked computers is increasingly applied to solve a variety of large-scale computation problems. Several PSEs(Problem Solving Environment) are developed to provide the application programmers with computers which are available in wide environment. However, these systems do not supplies web-based interface or the real-time visualization facility. Web technology is becoming the general technology on the development of network applications, in particular, because its interface can be made platform independent. In this paper, we propose a web-based PSE for executing the parallel SPMD application written in MPI. Also, a web-based collaborative environment is developed with a real-time visualization technology.

1 Introduction

With the advance of network and software infrastructure, grid-computing technology on a cluster of heterogeneous computing resources becomes pervasive [1, 2,3,4]. Grid-computing describes a coordinated use of an assembly of distributed computers, which are linked by networks and are deployed by many kinds of softwares. The potential benefit of the grid-computing is that users can exploit a powerful monolithic virtual machine. However, construction of a grid-computing system has been a challenging task which involves broad spectra of technical issues. The major hurdles of grid-computing environments are due to the lack of tools to facilitate the development of parallel and distributed applications [5, 4]. Consider the basic operations that arise in the development cycle of such applications [6]:

1. Generation: source files
2. Transfer: source files and data files between computing resources.
3. Compile: each source file on each computing resource
4. Execution: execution file
5. Visualization: the result to be collected from computing resources after the finish of the execution

The steps including two, three and five can become quite time consuming as the number of resources increases because users must transfer many files between

M. Bubak et al. (Eds.): ICCS 2004, LNCS 3038, pp. 1117–1123, 2004.

computers. Moreover, step five should wait for the completion of the previous step, the fourth step, which might require long time. In order to use the visualization service, the user needs to have an account on the machine on which the visualization system is installed. In this research, a web-based PSE(hereinafter referred to as *GridGate*) is developed for doing those five steps on one interface. For this purpose, this research will focus on more effective use of computing resources by making the systems more easily available and collaborative to the researchers.

This paper is organized as follows: Section 2 describes the system architecture of *GridGate*. Section 3 shows the experimental results on our testdbed. Section 4 concludes this paper and sketches some future works.

2 System Aarchitecture of *GridGate*

The simulation procedure in *GridGate* is schematically described in Fig. 1. The structure of *GridGate* consists of two major functional parts; grid portal and realtime visualization. The basic components for a computational run in *GridGate* are the user's source code for simulation (referred to as solver) and data files. It is assumed that the user prepares these files. The realtime visualization provides monitoring of the intermediate result of the assigned application by real-time graphical processing to a three-dimensional graph.

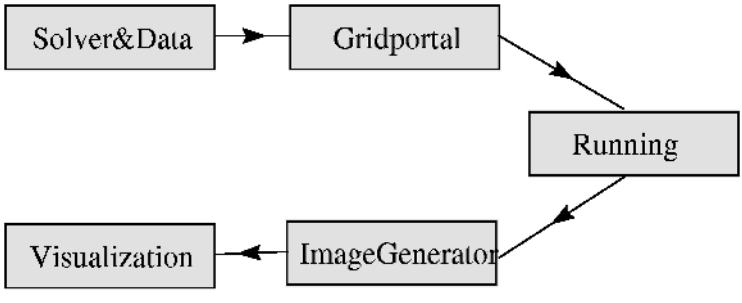

Fig. 1. System Flow of *GridGate*

2.1 Grid Portal

A grid portal is developed for easy and rapid development of SPMD applications on the grid environment. Our grid portal supplies the web-based functions of authentication, visualization of resource's state, distribution&compilation of user's job and spawning the job. These functions are shown in the following Fig. 2.

The basic components for a computational run in *GridGate* are the user's source code and data file for simulation. It is assumed that user prepares these files and the makefile to compile the source in each machine. The authentication step needs user's id and password one time. In this research, authentication step

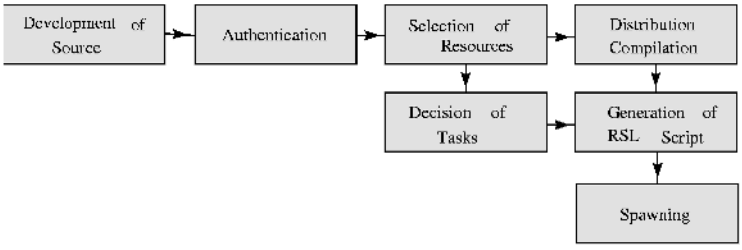

Fig. 2. Structure of grid portal

generates an user proxy using SWING[7]. In the next step, user could choose resources and the number of tasks according to the current information of the computing resources. The hardware information includes the name, architecture, CPU load and network traffic.

2.2 Realtime Visualization

Visualization should show the time dependence of a data set, the result of simulations, using several techniques. Simulations of fluid dynamics are examples of the applications that generate data sets in time. Static visualizations of dynamic information are sometimes ineffective and incorrect because they do not convey motion. Animations can help the user understand, particularly if the user is interested in the process of simulation. A traditional approach to the visualization is for the user to use the visualization system or a library to implement application specific visualizations. Many such visualization systems require that the user has an account on the machine on which the software is installed. For these reasons scientists often transfer the results of these visualizations by videotape or by converting a series of animation frames to a movie file(MPEG, GIF etc). The process of translating a visualization to the movie file eliminates user interaction with the visualization. Recent technological developments have created opportunities for new approaches for representing time-sensitive problems using web-based computer animation. More importantly, Java and the World Wide Web have provided a universal platform for building animations and visualizations. This web-based execution model solves many dissemination problems. Users can interact with animations without having an account on a particular machine. Fig. 3 shows the process of realtime visualization based on the web. Each task generates a series of intermediate files that could be merged into one file. Image generator gets the merged files as its input and generates the animation that represents its simulation. In this research, we use gnuplot as the image generator.

2.3 User Interface

A dedicated graphic user interface is designed for *GridGate*. This GUI is responsible for controlling the system and manipulation of a RSL(Resource Speci-

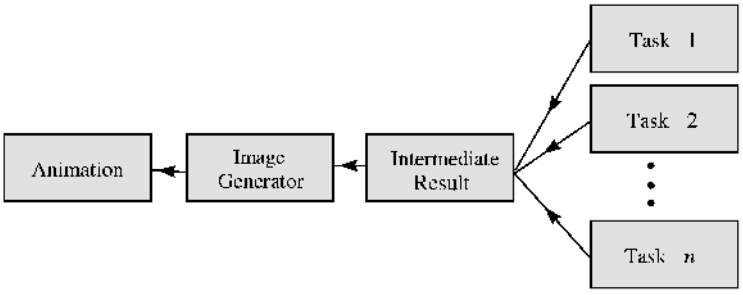

Fig. 3. The process of realtime visualization

fication Language) file. The RSL file is the important input for Globus as a text file which contains the information both of the resources and the solver required for running a grid-computing application. The resource information includes the host name, local scheduler, and working directory. The solver information describes data I/O as well as its name.

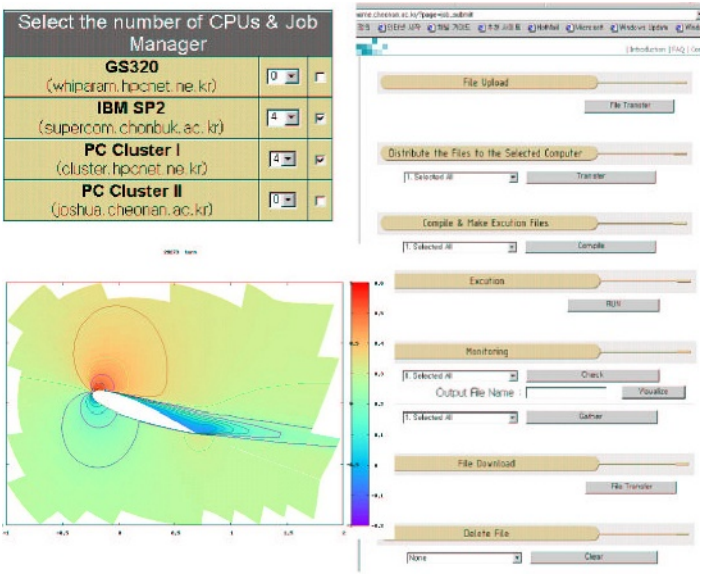

Fig. 4. User interfaces in *GridGate*

GridGate provides three major interfaces: list of available resources, panel for job-submit and visualization. The control panel supplies many buttons. A button "Run" as shown in Fig. 4 invokes the user's job according to the information prescribed in the RSL file that is automatically created by selecting the computing resources. A button "Visualize" opens the visualization interface, on which the status of execution of the solver is monitored by a real-time animation.

The other buttons, "Transfer" and "Compile" transfers the user's job to each resource and compiles the solver, respectively.

3 Experiment

3.1 Testbed for Grid-Computing

In this research, we established a testbed that consists of five parallel systems; Compaq HPC320, Compaq GS320, Linux-Cluster, two IBM SP2 machines. These systems achieve the different utilization from each other depending on the time. Eventually, our *GridGate* allows people to reach out and get the computational resources they need from their own desktop workstations and to monitor their intermediate results from anywhere. In this research, we developed *GridGate* System and used Globus Toolkits[3,4] and several job schedulers depending on the supercomputing resources as shown in Fig. 5. The *GridGate* System is the grid-computing tool that supplies GUI, job-distribution and remote compilation.

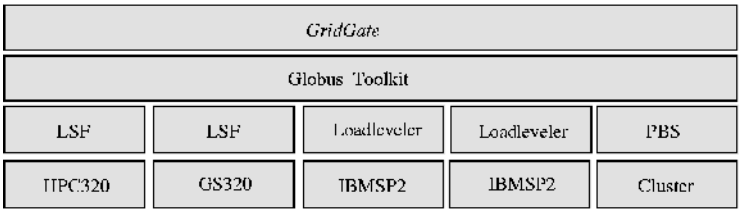

Fig. 5. Hierarchy of middlewares in our testbed

Fig. 6 shows the network configuration of our testbed. The supercomputers of KISTI are connected with 800 MBps HIPPI(High-Performance Parallel Interface) and are linked to the Korean R&D network; Kreonet and HPCnet. The HIPPI makes our grid faster because the latency time of the network is short. Two IBM SP2s are linked to 45 Mbps Kreonet/HPCnet

3.2 Experimental Results

Table 1 shows the number of processors on the experiments using a turbo machine fluid analysis program.

These experiments are conducted according to the number of processors when the size of computation is 28 blocks. The elapsed time is measured when the iteration reaches 100. We are not fully able to use the processors of the supercomputers because of the restriction of the management policy. Although we could obtain the speed-up according to the increase of processors, the communication time commanded the absolute majority.

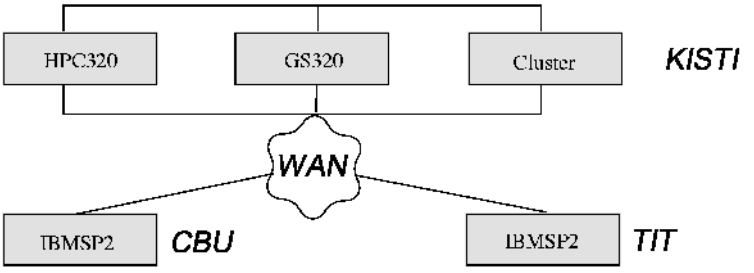

Fig. 6. Network configuration in our testbed

Table 1. The number of processors on the experiment

CPU #	CASE I		CASE II		CASE III			
	HPC320	GS320	IBM SP2	GS320	Cluster	IBM SP2	HPC320	GS320
1	1		1			1		
2	1	1	1	1				
4	2	2	2	2	1	1	1	1
7	3	4	2	5	1	1	2	3
14	4	10	2	12	1	2	4	7

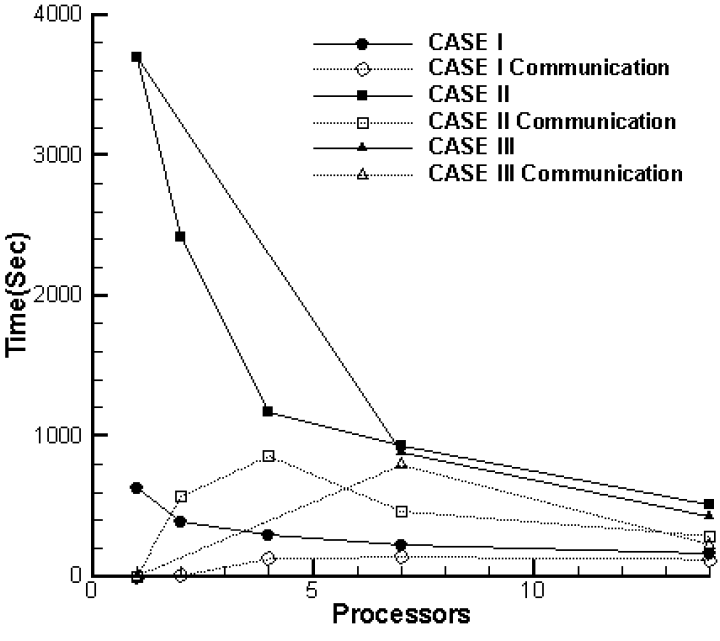

Fig. 7. Performance Analysis

1. CASE I (computation using HPC320, GS320) This experiment is conducted using two supercomputers connected with LAN. The result shows the speed-up in proportion to the number of processors in spite of the communication overhead.
2. CASE II (computation using IBM SP2, GS320) CASE II simulates the same code on the WAN environment between CNU and KISTI. In spite of WAN environment, the increase of processors makes the elapsed time reduced. In fact, we distribute the job on two supercomputers unequally in order to reduce communication data.
3. CASE III (computation using cluster, GS320, IBM SP2, HPC320) This case is similar to CASE II on the WAN environment and the unequal distribution of job.

4 Conclusion and Future Works

In this research, we implemented a grid toolkit named *GridGate* on several supercomputers connected with LAN or WAN. The users from any systems could submit jobs and have them transparently run on the grid environment. This would provide many benefits to the supercomputing centers, including utilization and to the users a convenient simulation environment. A major challenge for this research was providing a uniform software environment across the geographically distributed and diverse computational resources. To meet this challenge, we developed *GridGate* and used Globus Toolkit, which provides a variety of services including co-allocation, security, and parallel programming support.

References

1. V. Sunderam, "pvm: A framework for parallel distributed computing," *Concurrency: Practice and Experience*, vol. 2, December 1990.
2. C. K. I. Foster, "globus: A metacomputing infrastructure toolkit," *Intl. J. Supercomputer Applications*, vol. 11, no. 2, 1997.
3. I. Foster and C. Kesselman, *The Grid: Blueprint for a new Computing Infrastructure*. Morgan Kaufmann Publishers, Inc., 1998.
4. I. F. K. Czajkowski, S. Fitzgerald and C. Kesselman, "Grid information services for distributed resource sharing," in *Proceedings of the Tenth IEEE International Symposium on High-Performance Distributed Computing (HPDC-10)*, August 2001.
5. Bhatia and Burzevski, "webflow-a visual programming paradigm for web/java based corse grain distributed computing," *Concurrency: Practice and Experience*, vol. Java Special Issue, March 1997.
6. K. A. Hoffmann, *Computational Fluid Dynamics for Engineers*. Morgan Kaufmann Publishers, Inc., 1993.
7. "Creating new information providers," *MDS 2.1 GRIS Specification Document*, May 2002.