Design and Implementation of GPDS[*]

Tae-Dong Lee, Seung-Hun Yoo, and Chang-Sung Jeong[**]

Department of Electronics Engineering Graduate School,Korea University
{lyadlove,friendyu}@snoopy.korea.ac.kr, csjeong@charlie.korea.ac.kr

Abstract. In this paper, we describes the design and implementation of Grid-based Parallel and Distributed Simulation environment(GPDS). GPDS not only addresses the problems that it is difficult for parallel and distributed application to achieve its expected performance, because of some obstacles such as deficient computing powers, weakness in fault and security problem, but also supports scalability using Grid technologies. GPDS supports a 3-tier architecture which consists of clients at front end, interaction servers at the middle, and a network of computing resources at back-end including DataBase, which provides three services: Automatic Distribution Service, Dynamic Migration Service and Security Service, designed by UML-based diagrams such like class diagram and interaction diagram. The GPDS has been implemented as Grid Agent(GA) and Simulation Agent(SA) using C++. The object-oriented design and implementation of GA and SA in GPDS provides users with modification, extensibility, flexibility through abstraction, encapsulation and inheritance.

1 Introduction

Parallel and distributed simulation (PADS) is concerned with issues introduced by distributing the execution of a discrete event simulation program over multiple computers. In paper [1], we described the problems with PADS of performance and deficient computing power, weak in fault and security problem. for solving these problems, the paper suggested the three services: Automatic Distribution Service, Dynamic Migration Service, and Security Service. The three services provide both the supply of computing resources and robustness of the system which PADS does not provide. The GPDS is composed of two agents, Grid Agent (GA) and Simulation Agent (SA). The GA has fundamental functions of resource broker using Globus toolkit[5]. It accomplishes three major services, which include automatic distribution, dynamic migration, and security. Automatic distribution service makes parallel and distributed system have the strong extensibility to utilize abundant resources. Also, dynamic migration enables the

[*] This work has been supported by KOSEF and KIPA-Information Technology Research Center, University research program by Ministry of Information & Communication, and Brain Korea 21 projects in 2004.

[**] Corresponding author.

fault tolerance of whole system as well as the improvement of overall performance. The SA provides several modules which assist the PADS to achieve its objective. It is responsible for communication, monitoring, and dynamic configuration for parallel and distributed architecture, and manages available servers via the communication with the GA. GA and SA enable clients to transparently perform a large-scale object-oriented simulation by automatically distributing the relevant simulation objects among the computing resources while supporting scalability and fault tolerance by load balancing and dynamic migration schemes

In this paper, we describe design, implementation and performance evalution of GPDS. It shows how grid-based parallel and distributed simulation improves the existing simulation environment. In Sect.2, we illustrate the design of GPDS including architecture and UML-based diagrams such like class diagram and interaction diagrams. In the next section, we depict the implementation of GPDS and experimental results. At last, we conclude in Sect. 4.

2 Design of GPDS

2.1 System Architecture

The system architecture of GPDS is 3-tier architecture based on client-server model shown in Fig.1(a). Here, the server indicates the broad range which is enclosed by virtual organization (VO). The hosts within the VO can be servers of the GPDS. All processes of the server side are transparent to the client. The client only sees the results from the server side. The client in client tier delivers an executable and standard input files of required simulation to the GPDS Manager in Server tier. The GA creates a process to allocate simulations to remote hosts and the SA. Note that there is a simulation process in the SA. This simulation process is used to accomplish dynamic migration service. The SA manages the DBs in Database tier while cooperating with the GA. Each simulation process within remote hosts and the SA joins to simulation-specific middleware to execute the PADS. The result of simulation is returned to the client by the SA.

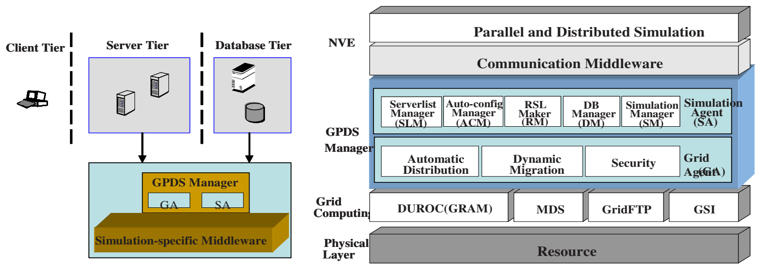

Fig. 1. (a) 3-tier architecture of GPDS (b) Layered structure of GPDS

Moreover, the characteristics in the architecture of GPDS can be grasped by observing its layered structure. The layer structure of GPDS is shown in Fig.1(b). The GCE(Grid Computing Environment) accomplishes the management of resources. In the GPDS, powerful modules of the Globus toolkit are used on the GCE. The GCE comprises of four modules: GRAM which allocate and manage the job in the remote hosts, MDS which provides information services, GridFTP which is used to access and transfer files, and GSI which enables authentication via single sign-on using a proxy. The NVE(Networked Virtual Environment) consists of application and middleware to communicate efficiently. In the application layer, the logic of a simulation is performed through the interaction of entities. Each application joins to a corresponding middleware. The middleware layer provides the communication of entities, interest management, data filtering, and time management required to achieve stable and efficient simulation. In brief, the GPDS means the NVE over the GCE. The GPDS Manager is an intermediate layer between NVE and GCE. It is in the charge of a bridge between both layers. As mentioned earlier, GPDS Manager is composed of Grid Agent and Simulation Agent. Agent is an identifiable computational entity that automates some aspect of task performance or decision making to benefit a human entity.

2.2 Class Diagram

Figure 2 shows a class diagram which uses a facade pattern which provides a unified interface to a set of interfaces in a subsystem. The facade pattern offers the benefits to shield clients from subsystem components, promote weak coupling between the subsystem and its clients. In Fig. 2, CGPDS is a facade for GPDS system, CGA for Grid-based classes and CSA for simulation-based classes. Each applications approach to agent classes through CGPDS. Also, the agents can be extended through aggregation to CGPDS. This object-oriented architecture provides extensibility of agents. CGA manages and controls the Grid-based classes, and CSA does simulation-specific classes. The SA includes five classes: ServerlistManager(CSLM), RSLMaker(CRSLMaker), Auto-configuration Manager(CACM), Simulation Manager(CSM) and DB Manager(CDBM). The CSM makes the list of resources available in corresponding virtual organization(VO). The number and performance of available hosts have great effect on the configuration of the PADS. This severlist of available resources is periodically updated

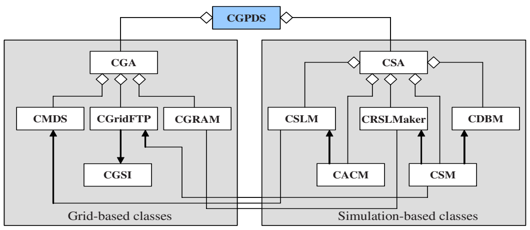

Fig. 2. Class diagram of GPDS

and referenced. The CRSLMaker dynamically creates a RSL code to meet the status of simulation and the requirements of the GA. The CACM automatically makes configuration files to provide information needed to initiate the PADS, according to the serverlist. The CSM has three missions. First, it establishes a connection to the client. Second, it periodically received and monitored simulation data from one simulation process within the SA, and delivers them to the CDBM. Third, the simulation data is returned to the client as simulation results by CSM. Lastly, the CDBM stores the simulation data of each host periodically. The stored data is applied to the recovery of the fault.

2.3 Interaction Diagram

Automatic distribution service means that GPDS Manager can automatically create the PADS by transferring and executing solicitated executable and standard input files on new host. Dynamic migration strategy means that the computing which has been proceeding on one host can be transferred to and continuously performed on another host. This service is used to achieve two purposes in GPDS. First goal is the fault tolerance of the whole system, and second is the improvement of performance. Each service composes of 3 steps.

Interaction Diagram for Automatic Distribution Service. Figure 3(a) shows the interaction diagram for Automatic Distribution Service. In (1) step, client connects to GPDS. The CSM of SA establishes a connection with the client. The client submits the job and sends indispensable files to GPDS. (2) step accomplishes the preparation for creating remote servers. It consists of four stages, that is, server list production, configuration, storage, and transmission. In server list production stage, the CSLM in SA makes out server list which includes available resources, using the metadata of hosts registered to GIIS through CMDS. In next stage, the CACM automatically modifies an initial configuration into several configuration files corresponding to parallel and distributed architecture by considering the number of available hosts and reflecting all information which is required to initiate remote servers. The CRSLMaker automatically generates the required RSL codes to use Globus. In the storage stage, the initial configuration data of each remote server is saved in DB by CDBManager. At last, the program files and configuration file is sent to remote hosts through CGridFTP service using CGSI by GA in the transmission stage. In (3) step, The GA forks one process to execute remote hosts. Through CGRAM service of Globus, the GA simultaneously activates the simulation process of remote servers and the SA. At this time, the RSL code which CRSLMaker created is used to submit simultaneous jobs. The CGRAM provides the barrier that guarantees all remote servers start successfully. The GA establishes the flag so that the process of SA can identify its mission. This process plays a role in delivering periodic simulation data to SA. These data is stored by CDBManager, transmitted to the client by CSM. Remote servers are initiated by reading each assigned configuration file, and periodically deliver own computation to the simulation process of the SA through simulation-specific middleware.

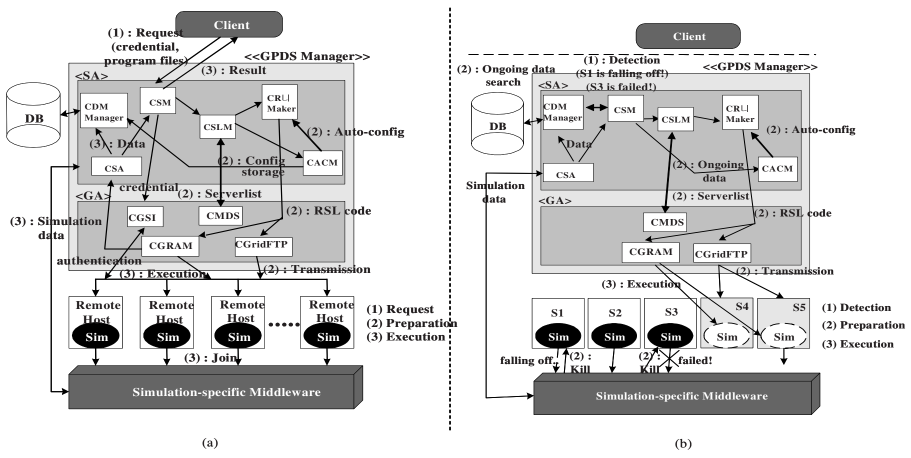

Fig. 3. Interaction diagram for (a)Automatic Distribution Service (b) Dynamic Migration Service

Interaction Diagram for Dynamic Migration Service. Figure 3(b) shows the interaction diagram for Dynamic Migration Service. In first step, The CSM of SA employs the timeout mechanism to detect the fault of remote servers. In Fig. 2(b), S1, S2, and S3 are remote hosts on the execution step of automatic distribution service. The CSM perceives the fault of S3 because GA regularly retrieves the current status of servers who are the members of GIIS through the metadata of CMDS. By analyzing this information, the GA can look up the better resource or recognize the degradation of current slave servers. We assume that the performance of S1 is falling off and S5 is founded as better server in Fig. 2(b). In second step, the CGPDS needs to prepare the creation of new remote server. The CSM retrieves ongoing data of the target server in DB through CDBManager, which manages the simulation data per remote host. Following mechanisms are the same as the storage and transmission stages of the preparation step in automatic distribution strategy. In Fig. 2(b), after the CSM searches for the ongoing data of S1 or S3, configuration file is made out using this data by the Auto-configuration Manager and transmitted to S5 or S4 through the GridFTP service by GA. The third step is same as the third step of Automatic Distribution Service.

3 Implementation of GPDS

For an experiment of the GPDS system, we implemented parallel and distributed war game simulation where distributed forces are operated by movement, detection, and close combat. The HLA/RTI is used as simulation-specific middleware to provide stable communication and interoperability. High Level Architecture (HLA) was developed by the Defense Modeling and Simulation Office (DMSO) to provide a common architecture that facilitates simulation interoperability and reusability across all classes of simulations [8]. HLA has been adopted as IEEE standard 1516 in September 2000. The Runtime Infrastructure (RTI) is a collection of software that provides commonly required services to simulation systems using HLA. The information about forces is supplied in the configuration file, and distributed by the GPDS Manager. The GPDS uses the RTI NGv1.3 [8] and Globus v2.4. Client is constructed on windows based system, while servers are based on Linux. Our implementation is accomplished on 4 PCs as clients, and 10 clusters(5 : Pentium IV 1.7GHz, 5 : Pentium III 1.0GHz Dual) and one 486 computer as servers on a VO. Our experiments are accomplished to confirm key services of the GPDS.

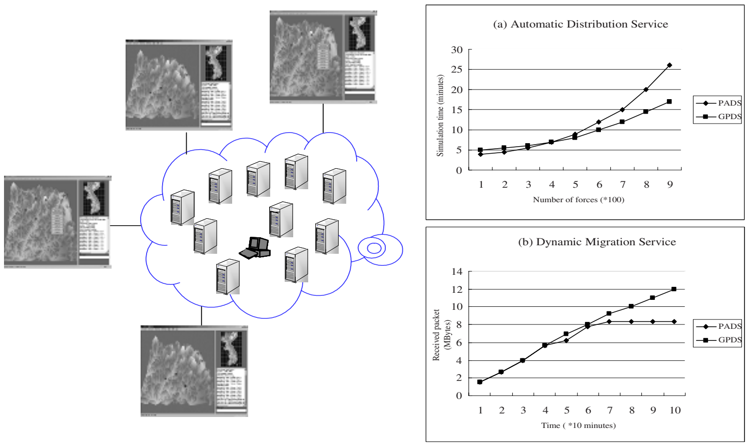

Fig. 4. Experimental results : (a) Simulation time according to the Increase of forces, (b)Accumulated received packets updated by 600 forces per 30 second

First experiment is for the automatic distribution service. We organizes a PADS which has five servers(we assume that 486 computer is included as server, because of the limitation in local condition), and the GPDS which has a VO of 11 servers(10 clusters and 486 PC). Then, we estimated the complete time of simulation as the number of forces increases. As we expect, the resource selection of the GPDS did not choose the 486 computer. In Fig.4(a),the GPDS is superior to the PADS as the scale of simulation is increasing,although the time consumption of initialization have an effect on the state of small forces. The GPDS can utilize abundant computing power and adapt for various environment,as well as provide convenient user interface.

To verify the dynamic migration service, we comprised second experiment. In this test,we measured accumulated received packets updated by 600 forces per 30 second. One packet has the size of 100 bytes. In 70 minutes, we intentionally made a failure on one server. DB Manager stores the information related to federation

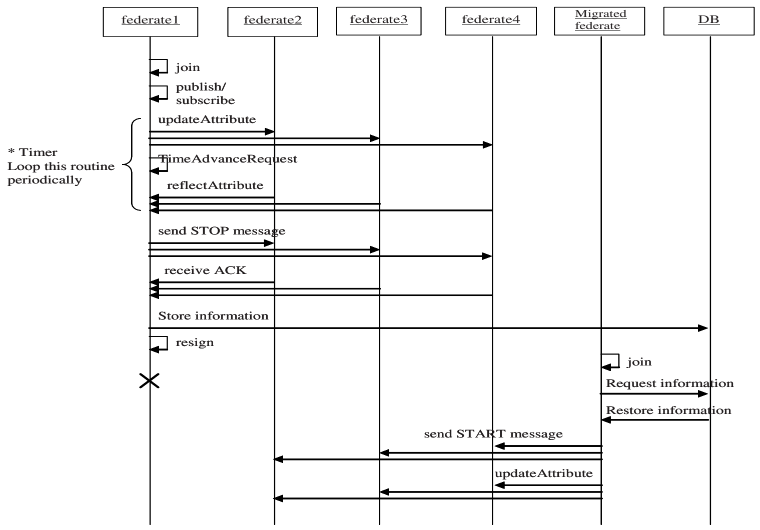

Fig. 5. Sequence diagram for second experiment

like LBTS and object information periodically, and then the application resigns from the federation. The application sends the stop message to federates before resignation. GA gathers the information of MDS and SA sends the application to a selected host. The application sends the restart message to all federates and receives the data stored before failure. As shown Fig.4(b), the GPDS can fulfill its mission after the failure, while the PADS is halted. Fig.5 shows the sequence diagram for second experiment.

4 Conclusion and Future Work

The paper has described the design and implementation of GPDS. GPDS not only addresses the problems that it is difficult for parallel and distributed application to achieve its expected performance, because of some obstacles such as deficient computing powers, weakness in fault and security problem, but also supports scalability using Grid technologies. GPDS supports a 3-tier architecture which consists of clients at front end, interaction servers at the middle, and a network of computing resources at back-end including DataBase, which provides three services: Automatic Distribution Service, Dynamic Migration Service and Security Service, describing the design by UML-based class diagram and interaction diagrams. Grid and simulation agents in the interaction server enable client to transparently perform a large-scale object-oriented simulation by automatically distributing the relevant simulation objects among the computing resources while supporting scalability and fault tolerance by load balancing and dynamic migration schemes.

As for future work, GT2.4 are being replaced by GT3 which is implemented by Java, and GPDS must be changed based on GT3 using Java. We will develop web-based GPDS in future using Java. Also, in the paper we did not describe HLA-specific issues concerning the design of migration service for HLA components. We have developed RTI [9] according to HLA interface specification and are developing RTI implementation (RTI-G) using Grid components. We will submit the RTI-G related works, and then we will describe the HLA-specific issues in detail.

References

1. C.H. Kim, T.D. Lee, C.S. Jeong, "Grid-based Parallel and Distributed Simulation Environment" 7th international conference PaCT2003 Nizhni Novogorod Russia, Proceedings LNCS pp. 503–508, September 2003
2. www.globus.org
3. I. Foster, C. Kesselman, G. Tsudik, S. Tuecke, "A Security Architecture for Computational Grids," Proc. 5th ACM Conference on Computer and Communications Security Conference, pp. 83–92, 1998.
4. I. Foster, C. Kesselman, S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," International J. Supercomputer Applications, 15(3), 2001.
5. I. Foster, C. Kesselman, "Globus: A Metacomputing Infrastructure Toolkit," Intl J. Supercomputer Applications, 11(2):115–128, 1997.
6. K. Czajkowski, I. Foster, "Grid Information Services for Distributed Resource Sharing," Proceedings of the Tenth IEEE International Symposium on High-Performance Distributed Computing (HPDC-10), IEEE Press, August 2001.
7. J. Dahmann, R.M. Fujimoto, R.M. Weatherly, "The DoD high level architecture: an update," Winter Simulation Conference Proceedings of the 30th conference on Winter simulation Washington, D.C., United States Pages: 797–804, 1998
8. U.S. Department of Defense(DMSO), "High Level Architecture Run-Time Infrastructure (RTI) Programmer's Guide Version 1.3," http://hla.dmso.mil, 1998.
9. T.D. Lee, C.S. Jeong, "Object-oriented Design of RTI using Design Patterns," 9th international conference OOIS2003 Geneva Switzerland Proceedings LNCS 2817 pp. 329–333, 2003