Support for Effective and Fault Tolerant Execution of HLA-Based Applications in the OGSA Framework*

Katarzyna Rycerz¹, Marian Bubak^{1,2}, Maciej Malawski¹, and Peter Sloot³

¹ Institute of Computer Science, AGH, al. Mickiewicza 30, 30-059 Krakow, Poland

² Academic Computer Centre – CYFRONET, Nawojki 11, 30-950 Kraków, Poland

³ Faculty of Sciences, Section of Computational Science, University of Amsterdam
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands

phone: (+48 12) 617 39 64, fax: (+48 12) 633 80 54

{kzajac,bubak,malawski}@uci.agh.edu.pl, sloot@science.uva.nl

Abstract. The execution of High Level Architecture (HLA) distributed interactive simulations in an unreliable Grid environment requires efficient brokering of resources, which is an important part of the Grid Services framework supporting execution of such simulations. This paper presents the overall architecture of the framework with emphasis on services supplying the Broker Service with information about application performance on a Wide Area Network (WAN). For that purpose, a benchmark interaction-simulation-visualization schema is designed basing on CrossGrid medical application architecture [1,10].

Keywords: Interactive simulation, Grid computing, HLA, OGSA

1 Introduction

Distributed interactive applications often require extensive computing resources. The Grid is a promising concept for solving this problem as it offers a possibility to access resources which are not centrally controlled and are under different administrative policies.

Interactive applications built with an implementation of the HLA standard [9] allow for merging geographically-distributed parts of simulations into a coherent whole. However, applying HLA to the Grid environment on an "as is" basis is not enough. HLA-based simulations require a certain quality of service in the underlying infrastructure. First, HLA is based on the assumption that the environment it executes on is faultless, while the Grid environment is inherently unreliable and requires fault tolerance mechanisms. Secondly, the Grid facilitates access to computing resources and makes them more transparent to the user.

* This research is partly funded by the European Commission IST-2001-32243 Project "CrossGrid" and the Polish State Committee for Scientific Research, SPUB-M 112/E-356/SPB/5.PR UE/DZ224/2002-2004.

Currently, creating a distributed application based on HLA requires manual setup and configuration. HLA limitations are discussed in [15,16].

[15] identifies different levels on which it is possible to integrate HLA applications with Grid solutions. On the uppermost layer, we can consider HLA components as available resources that can be used to run HLA-based application on the Grid. In the case of the Department of Defense Runtime Infrastructure (DoD RTI) implementation of HLA, we can treat RTIExec and RTI libraries as resources that are needed to execute applications dynamically linked with the RTI library. The lower layers of porting HLA concepts to the Grid would depend on implementing HLA communication using Grid technologies (i.e. using GridFTP optimizations[8] for data transfer[15]) or redesigning HLA service (such as time or data management) implementations [14].

We have decided to focus on the uppermost level, which involves considering HLA as a resource and propose a system to support transparent and faultless execution of HLA-based applications in a Grid environment. The aim of this research is to design and build a system supporting interactive applications in the Grid environment and at the same time preserve backward compatibility with the HLA standard. As an example we use the medical application developed in the CrossGrid project [2]. In this application, connections between blood flow simulation, visualization and interaction are provided by means of Interactive Simulation System Studio (ISS-Studio) agents middleware [17] that, in turn, uses HLA in its communication layer.

In our previous paper [16] we have presented the overall architecture of the Grid Service framework for supporting execution of HLA-based applications. In this paper we present a more advanced concept of system architecture. We focus on the part of the framework responsible for benchmarks, which supports decisions regarding application setup and migration. In Section 2 we describe related work in the area of integrating simulation standards with Web services and the Grid. In Section 3 we present the overall system design. In Section 4 we focus on the part of the system responsible for supporting the broker in selecting the appropriate HLA Services for distributed interactive applications. We propose Grid Services that manage HLA benchmarks according to the model of application distribution. We base our measurements on CrossGrid [2] medical application requirements. Benchmark results are presented in Section 5. Section 6 concludes the paper.

2 Related Work

The modeling and simulation community has come to realize that currently, there does not exist a widely-accepted standard which could handle both modeling and simulation mechanisms, such as the HLA [9] and Web/IT standards [13]. Such was the motivation behind the Extensible Modelling and Simulation Framework (XMSF) project [14], whose aim is to develop a common standard for Web and DOD modeling and simulation.

A parallel research effort is going on within the Commercial Off-The-Shelf Simulation (COTS) Package Interoperability Forum (HLA-CSPIF) [3]. As there is no real standard use pattern for the High Level Architecture within the context of this area, the goal of the Forum is to create standards through Simulation Interoperability Standards Organization (SISO) [11] that will facilitate the inter-operation of COTS simulation packages and thus present users of such packages with the benefits of distributed simulations enjoyed by other modeling and simulation communities.

Then there is research related to supporting management of HLA-distributed simulations. Cai et al. [12] implemented a load management system for running large-scale HLA simulations in the Grid environment basing on GT2. They also present a framework where the modeler can design parallel and distributed simulations with no knowledge of HLA/RTI [12]. The authors hope that in the future the framework design should facilitate easy checkpointing and federate migration for dynamic load balancing.

Creating a scalable and fault-tolerant framework for HLA-based applications is also the goal of the Swedish Defence Research Agency (FOI), where the Decentralized Resource Management System (DRMS) is being developed[6]. DRMS is a JXTA-based peer-to-peer system for execution of HLA (High Level Architecture) federations in a decentralized environment.

3 Overview of Framework Design

In our previous paper [16] we presented the overall architecture of the Grid Service framework for supporting execution of HLA-based applications. In particular, we discussed an experimental HLA-speaking service that manages execution of a federate as described below. In this work we focus on the part of the framework that supports the *Broker Service* in making decisions on which HLA-speaking service to choose. Our proposed architecture involves a framework of Open Grid Services Architecture (OGSA) Grid services [7] that cooperate to set up distributed interactive applications. Within our framework, presented in Fig.1, each site providing HLA to the Grid community sets up the following *HLA Management Services*:

The *HLA-Speaking Service*, responsible for setting up, running and directly managing user federate code on its site and also acting as an interface between other services and the federate. The service starts execution of federate code on its site, saves the application state and restores it from the checkpoint file on *Migration Service* requests. A prototype of the *HLA-Speaking Service* was described in [16].

Benchmark Services that perform HLA benchmarks according to various scenarios. In this paper we concentrate on the simple interaction-simulation-visualization scenario described in Section 4.

Benchmark Analyzer Services analyze HLA benchmarks results according to *Broker Service* needs. These services analyze data produced by the *Benchmark*

Fig. 1. Grid Services Framework for managing HLA-based application

Service online (by asking it to perform measurements) or offline (by analyzing data from benchmarks stored in a database).

The *Monitoring Service* monitors performance of current federation.

The *Migration Service* orchestrates migration of HLA federates from its site to other sites basing on *Broker Service* information (see below).

The *RTIExec Service* (optional)– manages execution of the RTIExec process.

We also assume the presence of the *Registry Service* that stores locations of possible *HLA Management Services* described above and can be used as a database for finding appropriate services.

Additionally, the user has to have access to the following services:

The *Broker Service* that finds appropriate HLA-speaking Services for a given execution scenario basing on *Benchmark Analyzer Services*.

The *Flow Setup* service that sets up the application for the user and supplies appropriate HLA-speaking services with application code that is already compiled and dynamically linked with the HLA RTI library.

The goal of our system is to facilitate the usage of these distributed resources (such as HLA RTI library) in robust manner regardless of the Grid's unpredictable behavior.

4 Proposal of Support for Broker Decisions

Broker Service algorithms are crucial for performance of distributed applications. The broker has to make decisions not only during application setup, but also during migration of its parts. One important problem is performance of the distributed application on WANs. It is obvious that components, between which communication is intensive, should be placed within the same high performance network. A problem arises if there are not enough resources on one site. In that

Fig. 2. Performing HLA Interaction-Simulation-Visualization Benchmarks in the Grid Services Framework

case some intelligent decisions must be made on how to effectively distribute parts of applications between Grid nodes.

In this paper we focus on the part of the framework that supplies the *Broker Service* with information about communication performance of the application. In order to make brokerage decisions, the system needs to monitor hosts and the network infrastructure. Although various monitoring tools are currently being developed in the CrossGrid project[2], it is extremely difficult to monitor WAN connections. One possible solution is to perform on-line HLA-based benchmarks and make decisions about migration basing on their results.

In general, benchmarks should be suited to the application model of distribution. In this work, we focus on the scenario present in the CrossGrid biomedical application [1]. This application consists of three components : interaction, simulation and visualization. While simulation is being conducted, the user can change its parameters using the interaction module (sending small messages from interaction to simulation) and see results through visualization (simulation sends large updates to visualization). According to this scenario, our benchmark consists of two components: an integrated Interaction-Visualization Component (IVC) and a Simulation Component (SC). IVC sends small messages that are human interactive requests for change of SC parameters. SC responses are large messages that present the altered state of the simulation to the user. Certainly, this scenario is suitable only for some class of applications. In the future we are going to extend the set of benchmarks (Section 6).

The HLA benchmark federation consists of two federates, acting as IVC and SC respectively. IVC sends SC requests in which it asks for certain amounts of data and measures the time between sending the request and receiving a response. Messages are sent through updates of HLA data objects [9]. The benchmark is managed within the Grid Services framework by the *Benchmark_IVC Service* and the *Benchmark_SC Service*, that start, control and measure performance of IVC and SC respectively.

The task of the *Benchmark_SC Service* is to start and control SC federates. This is done by invoking their **Start** operation. The *Benchmark_IVC Service* takes from the *Registry Service* a list of available Grid Service Handlers (GSH) (step1 in Fig.2) of *Benchmark_SC Services* and *RTIexec* services. Then, it can perform operations within the *Benchmark_IVC PortType* listed in Tab.1.

Table 1. Operations of the *Benchmark_IVC PortType*.

check – makes requests to a specific <i>Benchmark_SC Service</i> indicated by Grid Service Handler (GSH)(step 2 in Fig.2). To perform this operation, <i>Benchmark_IVC Service</i> chooses the <i>RTIExec Service</i> and creates a federation. Then, it invokes the start operation of the <i>Benchmark_SC PortType</i> of the <i>Benchmark_IVC Service</i> (step 3) that sets up the SC part of the benchmark by connecting it to the same federation(step 4). Following that, the <i>Benchmark_IVC Service</i> starts its IVC federate (step 5) and performs measurements as described above
scan – performs check operations on the selected list of sites. If there is no response until a specific timeout, the <i>Benchmark_IVC Service</i> proceeds to the next site.
forscan – performs the scan operation from start time to stop time for each time interval.

5 Implementation and Results

The structure of our benchmarks was based on the CrossGrid medical application requirements [10]. In the original application, for a medium resolution of the computational mesh used in the simulation, the uncompressed update message from visualization to simulation is about 600 MB in size. For larger resolutions, the uncompressed update can reach several gigabytes. Obviously, compression will be used for reducing the volume of update messages.

Our measures confirm what can be expected of the Grid environment in conditions similar to those described above. Our current implementation consists of the *Benchmark_IVC Service* – **check** and **scan** operations, as well as the *Benchmark_SC Service* – **start** operation. Measurements were performed on the DutchGrid DAS2 [5] coupled with the Krakow ACC Cyfronet [4] infrastructure. The internal bandwidth between DAS2 nodes was 10Gbps and the bandwidth between DAS2 nodes and Cyfronet was 100Mbps. We set up the *Benchmark_IVC Service* at Vrije University in Amsterdam and performed the **scan** operation for connections between all other sites of DAS2 as indicated in Tab.2. The RTIExec process was run in Krakow, at ACC Cyfronet. The first column presents the measured time of the **check** operation, which consisted of setting up the federation in the OGSA framework as described above and sending actual data (10 requests (small message) and 10 responses (400MB each)). Each response was in fact di-

Table 2. Benchmark results (in seconds).

	check operation (incl. 10 x send) time		actual 10 x send time	
	avr	δ	avr	δ
A'dam(Vrije)–Leiden	695	10.5	656	10.7
A'dam(Vrije)–A'dam (UvA)	450	13.5	413	15.1
A'dam(Vrije)–Delft	289	9.9	252	8.2
A'dam(Vrije)–Utrecht	377	8.9	340	9.1
Total scan operation	1811.2	30.9		

vided into 1MB chunks of data due to memory allocation issues (RTI requires four times more memory then the size of the actual sending/receiving buffer). In the next column we show time of ten request-response operations made by pure HLA object updates. In the last row of the table we show the total duration of the **scan** operation. The measurements were taken outside working hours form 9-10 p.m., to avoid additional traffic. They give us an idea of what to expect from application performance in a simple interaction-simulation scenario and helps decide if it is good enough for running interactive applications (for some applications this can be satisfactory, for others – less so). The results also show that the benchmarks can provide support for decisions on where to put application components (the location of the interaction and visualization component is fixed due to human presence constraints, but the simulation component can be placed – for instance – in Delft according to benchmark results). Some tuning of benchmark parameters is necessary depending on whether they are performed on- or off-line.

6 Conclusions and Future Work

In this paper we describe the framework for management of HLA-based interactive simulations in the Grid environment. While in [16] we describe the migration support part of the framework, this paper focuses on support for the *Broker Service* that will help in choosing between HLA Services on the Grid not only during setup of the HLA-based application, but also while performing migration decisions under two different scenarios - failure or bad performance. We are currently working on the following issues:

Broker Decisions and Benchmark Analyzer. While designing *Broker Service* architecture and algorithms, it is also necessary to have a service that will analyze benchmark results according to specific needs of the broker.

Various Benchmark scenarios. The benchmarking case described here is based on a simple Interaction-Simulation-Visualization scenario, however we are working on extending it to other scenarios.

Migration Service - periodic checkpointing. Periodic checkpointing will allow saving application state before a failure and restoring it afterwards. Also, a decision should be made where to back up checkpoint files in order to use them in the case of a failure.

Single HLA-speaking service – many federates. In general, an HLA-speaking service should be able to manage many federates on its site. We are working on extending our experimental implementation to cover that case.

Monitoring Service As described in Section 3, there is a need for a service that will monitor performance of the current application to supply *Migration Service* with information on whether and when migration is needed.

Acknowledgments. The authors wish to thank Dick van Albada, Alfredo Tirado-Ramos and Zhiming Zhao for valuable discussions about OGSA and HLA, and Piotr Nowakowski for his remarks.

References

1. Bubak, M., Malawski, M., Zajac, K.: Architecture of the Grid for Interactive Applications, in: Sloot, P. M. A., et al. (Eds.), Proceedings of Computational Science - ICCS 2003, International Conference Melbourne, Australia and St. Petersburg, Russia, June 2003, vol. I, no. 2657, Lecture Notes in Computer Science, Springer, 2003, pp. 207-213.
2. Crossgrid project web page. <http://www.crossgrid.org/>
3. HLA CSPIF home page: <http://www.cspif.com>
4. ACK Cyfronet <http://www.cyfronet.krakow.pl>
5. Distributed Ascii Supercomputer <http://www.cs.vu.nl/das2/>
6. Eklof, M., Sparf, M., Moradi, F: HLA Federation Management in a Peer-to-Peer Environment in In Proceedings of the 03 European Simulation Interoperability Workshop, organized by the Simulation Interoperability Standardization Organization. Stockholm, 2003. <http://www.sisostds.org/>
7. Foster I, Kesselman C., Nick J., Tuecke S.: The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. Open Grid Service Infrastructure WG, Global Grid Forum, June 22, 2002.
8. Globus project homepage <http://www.globus.org>
9. HLA specification, <http://www.sisostds.org/stdsdev/hla/>
10. Private communication with Robert Bellman (available upon request).
11. Simulation Interoperability Standards Organization's (SISO)
<http://www.sisostds.org/>
12. Yuan, Z., Cai, W. and Low, M.Y.H.: A Framework for Executing Parallel Simulation using RTI published in: in: S.J. Turner and S.J.E. Taylor, editors, Proceedings Seventh IEEE International Symposium on Distributed Simulation and Real Time Applications (DS-RT 2003), pp. 20-28 IEEE Computer Society, Delft, The Netherlands, October 2003.
13. Web services, <http://www.w3.org/2002/ws/>
14. XMSF project home site. <http://www.movesinstitute.org/xmsf/xmsf.html>
15. Zajac, K., Tirado-Ramos A., Bubak, M., Malawski, M., Sloot, P.: Grid Services for HLA-based Distributed Simulation Frameworks, to appear in Proceedings of First European Across Grids Conference, Santiago de Compostela, Spain, Springer-Verlag, Heidelberg, February 2003.
16. Zajac, K., Bubak, M., Malawski, M., Sloot, P.: Towards a Grid Management System for HLA-based Interactive Simulations, in: S.J. Turner and S.J.E. Taylor, editors, Proceedings Seventh IEEE International Symposium on Distributed Simulation and Real Time Applications (DS-RT 2003), pp. 4-11. IEEE Computer Society, Delft, The Netherlands, October 2003.
17. Zhao, Z., van Albada, G. D., Tirado-Ramos, A., Zajac, K., Sloot, P. M. A., ISS-Studio: A Prototype for a User-Friendly Tool for Designing Interactive Experiments in Problem Solving Environments, in: Sloot, P. M. A., et al. (Eds.), Proceedings of Computational Science - ICCS 2003, International Conference Melbourne, Australia and St. Petersburg, Russia, June 2003, vol. I, no. 2657, Lecture Notes in Computer Science, Springer, 2003, pp. 679-688.