Experimental Grid Access for Dynamic Discovery and Data Transfer in Distributed Interactive Simulation Systems

Alfredo Tirado-Ramos[1], Katarzyna Zając[2], Zhiming Zhao[1], Peter M.A. Sloot[1], Dick van Albada[1], and Marian Bubak[2,3]

[1] Faculty of Sciences, Section Computational Science,
University of Amsterdam
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands
{alfredo|zhiming|sloot}@science.uva.nl,
[2] Institute of Computer Science, AGH, al.Mickiewicza 30,
30-059 Kraków, Poland
{kzajac | bubak}@uci.agh.edu.pl
[3] Academic Computer Centre -CYFRONET, Nawojki 11,
30-950 Kraków, Poland

Abstract. Interactive Problem Solving Environments (PSEs) offer an integrated approach for constructing and running complex systems, such as distributed simulation systems. New distributed infrastructures, like the Grid, support the access to a large variety of core services and resources that can be used by interactive PSEs in a secure environment. We are experimenting with Grid access for interactive PSEs built on top of the High Level Architecture (HLA), a middleware for interactive simulations. Our current approach is such that once a PSE simulation has been executed in the framework, mechanisms from both HLA and Grid middleware are used to broker resources, for job submission services, performance monitoring services, and security services for efficient and transparent execution.
We are experimenting with the Web-based Open Grid Services Architecture (OGSA) for HLA RTI Federate registration and discovery, as well as for data transmission. We have found that Grid Services give the possibility to allow for dynamic modification capabilities of HLA Federates, though the dynamic discovery of Federates and the use of Service Data (metadata) for service introspection is not trivial. Also, we have found that opening many data transmission channels between HLA Federates to one destination affects the number of connections you can make with other destinations.

Keywords: PSE, interactive, Grid, OGSA, HLA

1 Introduction

Problem Solving Environments (PSEs) are integrated computational systems that allow scientists to define complex problems, find the required nearest components and resources available, and utilize them efficiently. PSEs offer an integrated approach for constructing and running complex systems, such as distributed simulation and decision support systems. New distributed infrastructures, like the Grid [11], support the access to a large variety of core services and resources in a secure environment.

In this paper we report on some preliminary findings from our approach to Grid access for interactive PSEs built on top of a mature middleware for interactive simulations, the High Level Architecture (HLA) [14]. We investigate mechanisms and issues in existing implementations of HLA Runtime Infrastructure (RTI) [16] and the new Open Grid Service Architecture (OGSA) [19] to allow interactive applications and distributed components to communicate in near real-time.

OGSA [19] extends the Web Services [23] terminology to include Grid concepts, and to manage the creation and termination of resources as services. Its main focus is on the definition of abstract interfaces that allow services to cooperate without too much concern about the actual protocols being used. OGSA has currently a very general approach; it does not distinguish interactive services from batch services, and does not provide special support for near real-time interactive systems. This is left to higher-level services built on top of it. OGSA is also a specification which is still under development, changing continuously in response to feedback from the user community.

On the other hand, HLA is a stable solution for lower-level services for distributed and interactive simulation systems. Implementations of the HLA RTI are based on the Common Object Request Broker Architecture (CORBA)[22] middleware. Although HLA and CORBA offer features for developers of interactive and distributed applications, existing implementations lack the flexibility we believe essential for Grid-based interactive applications.

In [26] we proposed a three-level approach to this problem. In the first step we focus on automatic discovery of HLA RTI processes that coordinate distributed components of interactive applications. Next, we investigate efficient Grid-based data transfer protocols as a promising alternative for current RTI communication. Finally, we completely replace RTI by Grid technology mechanisms. As a proof-of-concept example, we use the European CrossGrid [5] biomedical simulation application, which requires near real-time steering of simulation parameters during runtime of the simulation executing on the Grid. For underlying HLA-based services, we use an agent-oriented software architecture, the Interactive Simulation System Conductor (ISS-Conductor), which is based on HLA for implementing and interconnecting distributed interactive simulation components.

This paper is organized as follows: Section 2 briefly introduces the Globus open Grid services infrastructure and the HLA-based ISS Conductor Framework, Section 3 describes our approach for Grid Service migration and our preliminary findings. We conclude in Section 4 with a brief discussion of current issues and future work.

2 Grid Services and the HLA-Based Interactive Simulation Infrastructure

2.1 Open Grid Services

Grid Services, as defined by OGSA [19], integrate Grid technologies [11] from the Globus Toolkit [12] with Web Services mechanisms [23] to construct a Grid-based distributed framework. A Grid Service instance is a potentially transient service that conforms to a set of conventions, expressed as Web Service Description Language (WSDL) [24] interfaces, extensions, and behaviors.

Grid Services provide controlled management of the distributed and often long-lived state that is commonly required in sophisticated distributed applications. OGSA also introduces standard factory and registration interfaces for creating and discovering Grid services. The Globus Open Grid Service Infrastructure (OGSI) [21] implementation of OGSA is intended to expose status information as well as probed, measured, or discovered platform information according to well-defined Service Data Descriptions in their Service Type WSDL. Also, the Extensible Markup Language (XML) format [25] is used for data storage in different kinds of repositories, like XML databases, simple file system cache, and in-memory cache.

2.2 The HLA-Based ISS Conductor

We are using the HLA-based ISS-Conductor as our prototypical interactive simulation infrastructure for Grid-access experimentation. The ISS Conductor is an agent-oriented software architecture based on HLA for implementing and interconnecting distributed interactive simulation components [27,28]. In ISS-Conductor, we use a layered interconnection mechanism: at the lower-level, messages between modules are carried by Communication Agents (ComAs), and at the higher-level, application logic is controlled by Module Agents (MAs). The software bus is normally the run-time infrastructure of the communication middleware adopted by the ComAs. The interaction scenarios between modules are represented as knowledge bases, which can be bound to MAs at run-time. In the current implementation of ISS-Conductor, the RTI 1.3NGV5 of HLA [15] is the communication interface between ComAs, and Amzi Prolog [3] is used to implement the reasoning engines in the MAs. ISS-Conductor is part of Polder [18], a distributed computing environment built by the University of Amsterdam.

3 A Grid-Enhanced HLA RTI Infrastructure

3.1 Migration Approach

We believe that Grid Services have the potential to bring remote and decentralized Federate service discovery and invocation to HLA-based simulation systems. We have targeted the issue of the discovery of control Federate components initially. We call this the RTI Layer Migration [26].

Federate discovery mechanisms in HLA use either a multicast discovery protocol, which does not scale well on large Wide Area Network Grid environments, or performs discovery via the specification of the RTI Executive Process (RTIexec) endpoint using a naming service. We consider this mechanism not very convenient for our purposes, because, for instance, having to know the endpoint in advance is not an acceptable restriction in Grid environments. As a first step, we are investigating the use of Grid services to address this issue.

As a finer grain approach, we investigate the use of Grid core services for the transport of actual Federate data, once the RTIexec information has been made available via a Grid Service. We call this the Federation Layer Migration. We experiment with Globus GridFTP and Globus I/O implementation extensions for interfacing with the RTI logical bus. It is important to note that Grid communications are basically peer-to-peer (P2P), which means that the destination information, well encapsulated within HLA, has to be found on the application layer. As a third step and future work, we plan to reimplement most of HLA communications using Grid technologies.

3.2 Federate Information Description

For our first migration approach, we publish the RTIexec as a Grid service. Grid Services basically inherit from a common base class, ServiceSkeleton, which includes the basic functionality to be provided by all Grid Services. All Grid Services contain, this way, a ServiceDataContainer holding Service Data elements exposed by the service for introspection. Therefore, to provide our service we inherit from ServiceSkeleton and implement the WSDL PortType named set of abstract operations and messages interface to expose. The Grid service information is initially used for Federate and Federation initialization, in a simple pull mode. The information provided by the Grid service includes endpoint information of the RTIexec execution (i.e., machine address and port number), name of the executing Virtual Organization (VO) for selecting the appropriate RTIexec to join, and RTIexec version number.

We have identified two possible approaches for exposing RTIexec information description. In the first approach we wrap the RTIExec as a service, and provide the information description via a set of self-describing PortType operations (e.g., GetRtiExecMachineName, GetRtiExecPortNumber, GetRtiExecVoName, GetRtiExecVersionNumber) in our WSDL definition. PortTypes are therefore used for defining the service interface, and Service Data is used just for providing service metadata and state data. In the second approach, we also take the RTIexec process as a whole, and wrap it as a service (OGSARTIExecService) with a basic Grid Service PortType, thus we provide the RTIexec information description as Service Data, offering a more extensible mechanism.

3.3 Federate Discovery

We believe that Grid Services have the potential to address the problem of remote and decentralized Federate discovery and invocation. Indeed, the OGSA community is currently working on High Level Index Services [13] that will provide mechanisms for publication and query Service Data. Among others, a notification mechanism is mentioned as one of the ways of obtaining such information.

We identify different levels of granularity for Federate discovery mechanisms using Grid technologies. So far we have identified two possibilities for service registry; one is to use a high-level Community Registry service for VO and RTIExec Service registration and inspection (based on OGSA Registry PortType), and the other is to build a kind of UserInteractiveService functionality that orchestrates subscriptions and publications.

In the first approach, Federate information is accessed via a Community Registry Service based on OGSA higher level service implementations. Here a VORegistryService implements OGSA Registry PortType to allow remote services to publish their Grid Service Handles (GSHs) into a public repository, and a ContainerRegistryService implements a service that exposes currently available services.

For the second approach, we take advantage of RTI algorithms to reuse and extend them. RTI mechanisms are restricted to exchanging data and event objects which structure and attributes are described explicitly in the configuration file. Nevertheless, RTI implementations do provide users with advanced mechanisms for matching subscriptions to publications. For instance, the translation of the HLA object description format into OGSA service data and representation of regions in the OGSA model is an approach that may provide RTI users with more flexible solution. This issue is strongly dependent on the final version of the OGSA specification [19], though, which is still evolving.

3.4 HLA Communication Infrastructure for Large Data Transfers

Within HLA two kinds of data can be exchanged between Federates, namely data objects and events. However, as mentioned before, current HLA implementations are generally restricted to the CORBA communication protocol [22]. In order to investigate interoperability issues related to our approach, we experiment with different data exchange protocols within the RTI framework.

We believe that Grid Services should give the ability to choose from a number of protocols, according to needs of particular distributed applications. We decided to experiment with GridFTP concepts [2] for our proof-of-concept medical interactive application. Such application requires large data transfers between simulation and visualization modules, currently carried out through a RTI logical bus.

Our ideas for building interoperable runtime communication infrastructure for interactive services is based on a bottom-up approach. At the bottom layer we are investigating P2P techniques. We start with adapting GridFTP solutions for our purposes, since there is a strong need for interactive applications to allow transfer of large volumes of simulation data in near real-time.

The RTI software bus is designed to simplify communications. It uses data distribution management algorithms [17] to efficiently route publications to subscriptions. Since the RTI software bus is going to be build on the top of P2P communications, it is useful to distinguish between the actual data transfer and the control channel that will be used by higher RTI levels responsible for Data Distribution Management (DDM) matching [17]. The basic communications stack is presented in figure 1.

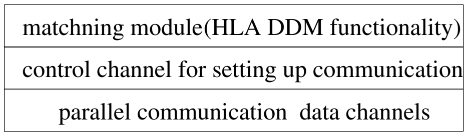
matchning module(HLA DDM functionality)
control channel for setting up communication
parallel communication data channels

Fig. 1. Basic communication stack in RTI design

The actual communication is presented as the bottom layer (Data channel Layer), and is performed by data channels between publisher and subscriber. On this level we apply TCP optimization techniques used also in GridFTP (parallel data channels, tuning of TCP buffer). For gathering data from multiple channels, we reuse the extended block mode concept [2].

The next control layer (Control Channel Layer) allows matching of the modules to actually setup the communication. In this architecture, each Federate in the distributed federation has to provide a HLA-like PortType with functionality similar to the behavior of the passive side of GridFTP [2]. This PortType will be responsible for opening a listening socket, and returning the port it is listening to. A separate listening socket is needed for each "federate to federate" communication. This listening socket is then used by the active federate (see figure 2) for creating multiple parallel channels between itself and the passive Federate. The channels can then be used for constantly sending the simulation data, and will remain open until an explicit operation is invoked. The use of basic FTP and GridFTP concepts for separation of control and data channels allows the basic architecture shown in figure 2 to be very easily redesigned, to satisfy also the the upper layer of the communication stack.

The upper layer (DDM Matching Layer) is responsible for matching subscriptions and publications. The HLA-like PortType is used by the RTIExec Service that performs actual matching, rather then directly by another federation.

3.5 Federate Layer Migration and Related Issues

As a the third step, we redesign some of the functionality of RTI components to build an event-based fully Grid functional service. We support discovery and data transfer for interactive applications, and fulfill the requirements of high level Grid services [20]. The interface of the service is defined by translating existing HLA's RTIAmbasador Application Programming Interface into WSDL. The functionality of the RTI library will be encapsulated within the Grid Service,

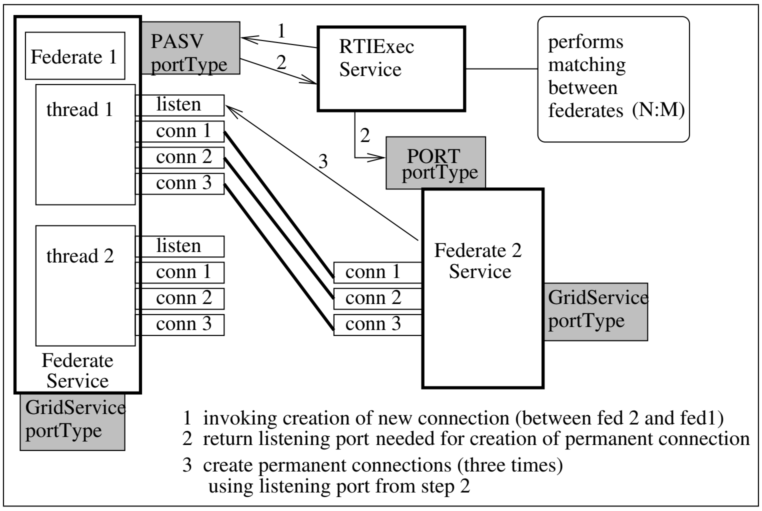
Federate 1
thread 1
thread 2
Federate Service
GridService portType
PASV portType
listen
conn 1
conn 2
conn 3
RTIExec Service
performs matching between federates (N:M)
PORT portType
Federate 2 Service
GridService portType
1
2
3
1 invoking creation of new connection (between fed 2 and fed1)
2 return listening port needed for creation of permanent connection
3 create permanent connections (three times) using listening port from step 2

Fig. 2. The HLA communication architecture for large data transfer

which can be itself a distributed high level service. The internal design requires further investigation of DDM algorithms, like effective matching subscriptions to publications as well as ownership and time management issues [16].

4 Discussion and Future Work

Currently, HLA requires explicit description of data and event objects that will be exchanged before the actual federation starts execution. This cannot be changed during runtime, and is specified in the Federate configuration file, which is quite a big drawback. We believe that Grid Services give the possibility to allow for dynamic modification capabilities.

One of our interests is to address this problem by providing a specialized discovery service for supporting interactive applications. For instance, one of the main questions for us is to what extend will the OGSI-compliant Index Service [13] satisfy the requirements of near real-time response (such as caching and aggregation of information). Initially, we work with one well-known registry to bind the local Federates with the services and modify the HLA RTI configuration file, since we do not expect to migrate RTI execution at runtime at the moment. We have found that the current OGSI implementation of the OGSA Registry PortType, even though unstable, works well for simple ServiceData queries. In the near future, we plan to investigate the use of OGSA discovery topologies and intelligent searching agents for more complex dynamic discovery.

Federate interface application components that want to join the federation have to provide a FederationAmbassador interface. The actual data and event sending is performed directly. We have found that opening many channels to one destination affects the number of connections you can make with other destinations, so we are also working on ways to allow the system to adjust to such constraints depending on number and location of remote processes. We are investigating mechanisms within RTI that will also provide information coherency and interoperability with the OGSI reference implementation.

In the near future we plan to investigate time and ownership management. This will allow services to exchange responsibility for published data as well as synchronize time between them.

Acknowledgments. The authors wish to thank Maciek Malawski for helpful discussions about OGSA. This research is partly funded by the European Commission the IST-2001-32243 Project "CrossGrid".

References

1. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. I. Foster, C. Kesselman, S. Tuecke. International J. Supercomputer Applications, 15(3), 2001.
2. GridFTP Protocol Specification. W. Allcock, J. Bester, J. Bresnahan, A. Chervenak, L. Liming, S. Meder, S. Tuecke. GGF GridFTP Working Group Document, September 2002.
3. Amzi Inc., Amzi Prolog Homepage, http://www.amzi.com, 2002
4. Belleman R.G., and Shulakov R.: High Performance Distributed Simulation for Interactive Simulated Vascular Reconstruction, International Conference on Computational Science (ICCS), Lecture Notes in Computer Science (LNCS) volume 2331, p. 265–274, 2002, Springer-Verlag, Berlin.
5. Bubak M., Malawski M., and Zajac K:: Towards the CrossGrid Architecture. In: D. Kranzlmeller, P. Kacsuk, J. Dongarra, J. Volker (Eds.) Recent Advances in Parallel Virtual Machine and Message Passing Interface, Proc. 9th Eurpean PVM/MPI Users' Group Meeting, Linz, Austria, September/October 2002, LNCS 2474, pp. 16–24.
6. CrossGrid – Development of Grid Environment for interactive Applications, EU Project, IST-2001-32243, www.eu-crossgrid.org
7. The DataGrid Project, http://www.eu-datagrid.org/
8. Frey J., Tannenbaum T., Foster I., Livny M., and Tuecke S.: "Condor-G: A Computation Management Agent for Multi-Institutional Grids", Journal of Cluster Computing volume 5, pages 237–246, 2002.
9. Foster I: "What is the Grid? A three checkpoints list". Grid Today Daily News And Information For The Global Grid Community July 22, 2002: VOL. 1 NO. 6
10. Foster I., Kesselman C.: Introduction to Grid Computing http://www.globus.org
11. Foster I., Kesselman C., Tuecke S.: The Anatomy of the Grid: Enabling Scalable Virtual Organizations. in:International J. Supercomputer Applications, 15(3), 2001.
12. The Globus Toolkit, http://www.globus.org/toolkit
13. Status and Plans for Globus ToolkitTM 3.0, http://www.globus.org/toolkit/gt3-factsheet.html
14. The HLA Working Group, http://www.sisostds.org/stdsdev/hla/
15. Defence Modelling and Simulation Office (DMSO), High Level Architecture (HLA) homepage, http://hla.dmso.mil/

16. High Level Architecture Run-Time Infrastructure RTI 1.3-Next Generation Programmer's Guide. https://www.dmso.mil/public/transition/hla/
17. Van Hook, D. J., Calvin, J.O.: "Data Distribution Management in RTI 1.3," 98S-SIW-206, 1998 Spring Simulation Interoperability Workshop, March 9–13, 1998.
18. Iskra K.A., Belleman R.G., van Albada G.D., Santoso J., Sloot P.M.A., Bal H.E., Spoelder H.J.W. and Bubak M.: The Polder Computing Environment, a system for interactive distributed simulation, Concurrency and Computation: Practice and Experience((Special Issue on Grid Computing Environments) in press), 2002
19. Grid Service Specification. S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, C. Kesselman; Open Grid Service Infrastructure WG, Global Grid Forum, Draft 2, 7/17/2002.
20. OOpen Grid Services Architecture Working Group (OGSA WG), http://www.ggf.org/ogsa-wg/
21. OGSI Technology Preview Release, http://www.globus.org/ogsa/releases/TechPreview/
22. Ryan C.O., and Levine D.L.: Applying a Scalable CORBA Events Service to Large-scale Distributed Interactive Simulations In: Proceedings of the 5 th Workshop on Object-oriented Real-time Dependable Systems. Monterey, CA.
23. Web Services, http://www.w3.org/2002/ws/
24. Web Services Description Language (WSDL) 1.1, http://www.w3.org/TR/wsdl
25. Extensible Markup Language (XML), http://www.w3.org/XML/
26. Zajac K., Tirado-Ramos A., Zhao Z., Sloot P., Bubak M: A Grid Service Approach to HLA-based Distributed Simulation Frameworks for Interactive Problem Solving Environments, accepted for First European Across Grids Conference, Santiago de Compostela, Spain, 13–14 Feb, 2003.
27. Zhao Z., Belleman R.G., van Albada G.D., Sloot P.M.A.: State Update and Scenario Switch in an Agent Based Solution to Constructing Interactive Simulation Systems, Proceedings of the Communication Networks and Distributed Systems Modeling and Simulation Conference, 2002
28. Zhao Z., Belleman R.G., van Albada G.D., Sloot P.M.A.: AG-IVE an agent based solution to constructing Interactive Simulation Systems, Proceedings of the second inter-action conference of computational science (ICCS02), Amsterdam, NL, 2002
29. Zhao Z., Belleman R.G.,van Albada G.D. and Sloot P.M.A.: AG-IVE: an Agent based solution to constructing Interactive Simulation Systems, in P.M.A. Sloot; C.J.K. Tan; J.J. Dongarra and A.G. Hoekstra, editors, Computational Science – ICCS 2002, Proceedings Part I, in series Lecture Notes in Computer Science, vol. 2329, pp. 693–703. Springer Verlag, April 2002. ISBN 3-54043591-3.