The edutain@grid Portals – Providing User Interfaces for Different Kinds of Actors

Roland Landertshamer[1], Christoph Anthes[1], Jens Volkert[1], Bassem I. Nasser[2], and Mike Surridge[2]

[1] GUP, Institute of Graphics and Parallel Processing
Johannes Kepler University, Altenbergerstraße 69, A-4040 Linz, Austria
rlander@gup.uni-linz.ac.at
[2] IT Innovation Centre, University of Southampton, UK
{bmn,ms}@it-innovation.soton.ac.uk

Abstract. In recent years grid computing has evolved from simple batch processing systems to highly interactive applications. One approach heading in that direction is the edutain@grid project. This project aims to enable the computing power of current grid systems to the area of online computer games and e-learning applications. The shift from traditional scientific computing applications into the domain of business oriented applications has led to the involvement of a variety of actors. All of these organisations and persons have different needs considering the access to the system.

To communicate with the edutain@grid system a novel combination of portals had to be developed. Three types of portals allow the stakeholders to get access to this system. These portals allow a unified and consistent interface approach hiding the complexity of traditional grid applications in order to support users from all different fields to work with edutain@grid in an intuitive way.

1 Introduction

Grid computing [11] has come far in the recent years. Traditional approaches from the area of batch processing have evolved to support interactive applications even in the area of real-time interactivity. An example for the change of Grid applications to support more interactive applications was the CrossGrid project [8], where the computation results of a flooding simulation could be displayed in real-time.

One of the novel approaches to bring more interactivity to the Grid is the edutain@grid project [9] which supports real-time interactive online applications (ROIAs) mostly from the field of e-learning and computer games. The two application domains have commonalities: they support a large amount of concurrent users and they both have to provide instantaneous feedback to the user input. Computational power is needed in these areas mainly for interaction processing of the connected participants of multi-user sessions.

E. César et al. (Eds.): Euro-Par 2008 Workshops, LNCS 5415, pp. 382–389, 2009.

To create support for such applications the portals in edutain@grid follow a different approach than traditional portals. They are considered as a user interface to the different layers and components of edutain@grid, rather than combining and exposing information from different resources in a single view.

Becaues of the real-time constraints of the edutain@grid applications it is not sufficient anymore to restrict portals to web interface technology. Other low-level technology in the form of a C++ API has to be used to interconnect to the interfaces of ROIAs to edutain@grid.

The following Section of the paper will introduce the related work focusing on available portal frameworks. Section three to six describe the different portals for different groups of actors in edutain@grid. The last sections provide an outlook into future work by showing up enhancements of the edutain@grid portals and conclude the paper.

2 Related Work

Portals in the context of Grid computing are typically considered to be web interfaces which aggregate information gathered from different web resources. This composition of information is often enhanced with the possibility of user interaction.

In order to ease the design and implementation of such portals many frameworks have been developed in the recent years. The most prominent examples are Jetspeed [3] and Gridsphere [2]. Other commonly used solutions are jPortlet, uPortal, LifeRay, and PGrade [4] which is built on top of Gridsphere.

A good overview on the mentioned examples can be found in [6], where the previously listed frameworks are described in detail and their functionality and architectures are compared.

The edutain@grid project enhances this portal concept from traditional web portals by providing for example an additional C++ portal API in oder to create a full portal framework which supports the needs of the different actors. A similar approach by offering a portal framework is chosen by Gannon et al. [12].

In the area of computer games Steam [5] could be considered as a community portal where users are able to create accounts and communicate via forums for a large set of games.

3 The edutain@grid Portals

The main approach of the edutain@grid middleware lies in the Grid support for ROIAs. It does not only consider newly developed ROIAs using parts of the middleware but it also includes the support of legacy applications. The main differences of edutain@grid to traditional Grid computing do not only lie in the real-time constraints of the executed jobs, but also in the nature of the life-time of a job.

The portals of the edutain@grid project support the three layers of edutain@grid, namely the business layer, the management layer and the real-time

layer by offering different approaches. They provide a traditional interface approach in the form of web portals for the management layer and the business layer as well as a low-level C++ API based approach as implemented in the client portal API which interconnects with ROIAs executed on the real-time layer.

An early draft on the portal architecture has been previously published by Anthes et al. [7] while more detail on the edutain@grid overall architecture can be found in [9]. This work provides a more precise insight into the developed portals and the interconnections within the edutain@grid architecture.

4 The Business Portal

The business portal is developed for coordinators who manage the distribution of applications on the different hosters. It provides functionality to manage contracts between hosters and the coordinator about provided hosts, SLAs and pricing. The functionality is provided via a standard GRIA web portal which follows the classic web portal approach so that the portal is accessible via an arbitrary web browser. It is directly connected to the business layer of the edutain@grid system which is implemented through GRIA web services [1].

5 The Management Portal

The management portal was developed for hosters which participate in this system by providing computing resources. It is implemented as a web portal which can be accessed via any web browser. The portal provides resource management and monitoring functionality to allow the hosters to keep an overview about their participating machines.

The management portal acts as administration interface for the participating hosters in edutain@grid. It is implemented as web portal using the Gridsphere portal framework. The main functionality of the portal are the configuration of participating machines in the edutain@grid middleware, the display of the status of these machines and controlling functions.

The portal itself is configured to provide three different views to the hosters: the configuration view, the monitoring view and the controlling view.

In the configuration view the hoster is able to manage and configure the machines which should participate in the edutain@grid system. The view consists of three separate portlets, the HostConfigurationPortlet, the ROIATypeConfigurationPortlet and the ROIADescriptorConfigurationPortlet. Each of these portlets allows to modify the configuration for the hoster's local site. The configuration itself is stored and managed by the management layer. The portlets provide a graphical interface in order to modify this configuration via the portal. Figure 1 shows the different portlets of the configuration view.

The first portlet in the configuration view is the HostConfigurationPortlet. This portlet is used to define which machines are available at the hoster's site. It displays a list of all configured machines together with the port settings used for the

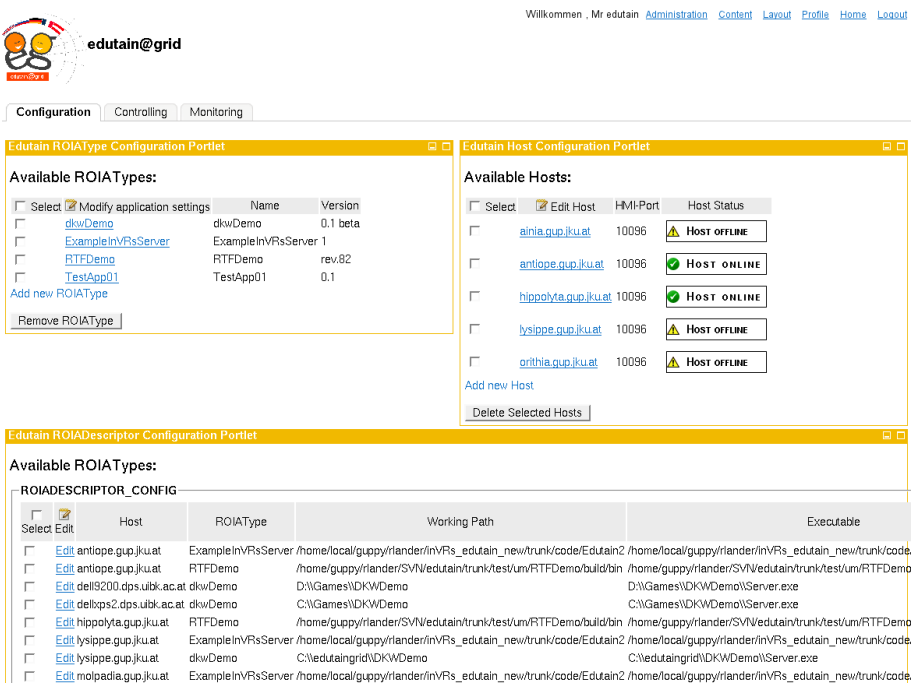

Fig. 1. Management Portal: Configuration view

connection of the edutain@grid management layer to the host. In order to connect the configured machines to the edutain@grid system a separate program has to be started on the single hosts, the ROIAServerStarter. This light-weight application listens on the defined port for commands from the management layer in order to start server applications on the host. To provide an overview if all configured hosts are able to communicate with the edutain@grid middleware the portlet displays a status box for each host which shows if the ROIAServerStarter application is running. Via this portlet it is possible to add new hosts, modify the port settings of participating hosts or remove hosts from the edutain@grid middleware.

The second portlet is the ROIATypeConfigurationPortlet. This portlet allows to define the different application types which can be hosted on the server machines. A configuration for a ROIAType currently consists of the name and the version of the application.

The last configuration portlet is the ROIADescriptorConfigurationPortlet. It allows for the configuration of the deployed applications on the participating hosts. This configuration is used by the management layer in order to start up the applications on the server machines. Every configuration entry currently contains the ROIAType, the host where it is deployed, the working path where the application is installed, the executable which can be started by the ROIAServerStarter, command line arguments which are passed to the executable at startup time and the number of threads.

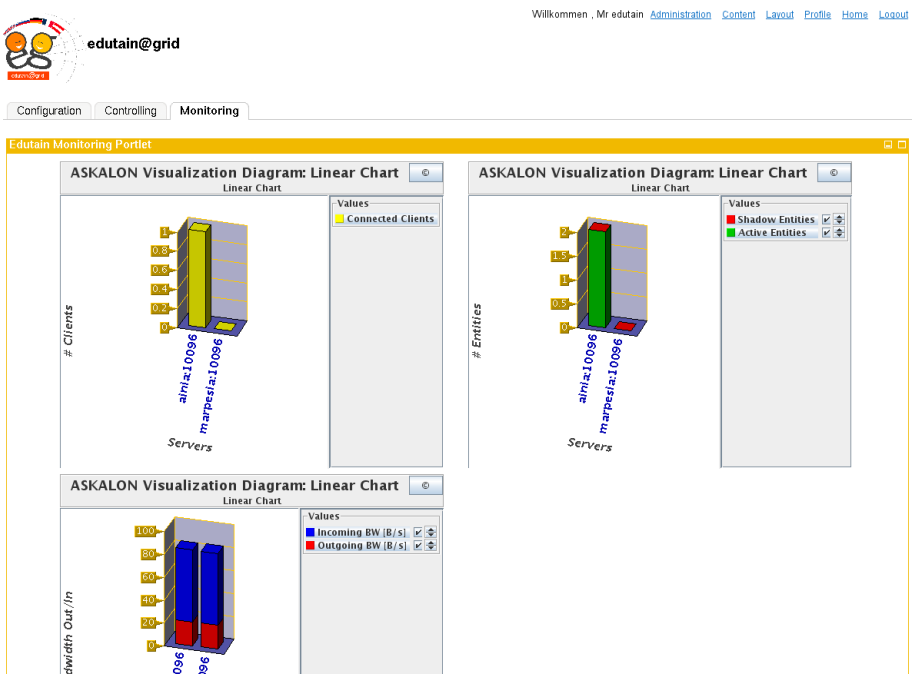

Fig. 2. Management Portal: Monitoring view

The monitoring view of the management portal consist of multiple instances of the MonitoringPortlet. This portlet allows for displaying the visualisation of monitoring data gathered by the edutain@grid middleware. Several monitoring targets like the incoming and outgoing bandwidth, the number of connected users but also application internal data like the saturation of the application loop are provided by the management layer which can be displayed per portlet instance. The configured monitoring data is displayed by a java applet which uses the visualisation library of the ASKALON tool-set for cluster and grid computing [10]. To update the monitoring data in the java applet a monitoring service is running in the management portal. At startup of the java applet a socket connection is established between the applet and the monitoring service. The service streams the monitoring data via this connection to the visualisation applet which displays the data in form of diagrams. Figure 2 shows an example of the monitoring view with three active MonitoringPortlets.

The controlling view of the portal consists of a single portlet, the ControllingPortlet. The portlet allows the user to view which hosts are involved in the different ROIA sessions. Via this portlet the hoster can manually start and stop ROIAProcesses on his local machines. This functionality is intended to be used for testing purposes whenever a new application is deployed on different hosts. The controlling view is shown in Figure 3.

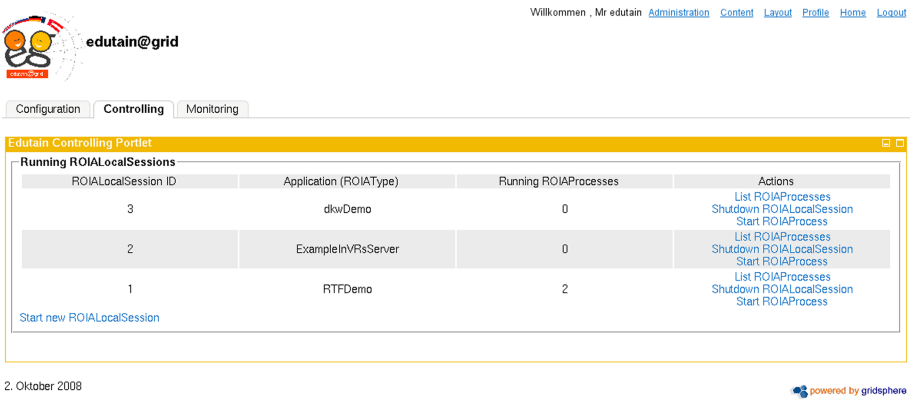

Fig. 3. Management Portal: Controlling view

6 The Client Portal API

The client portal is developed for the end users of the edutain@grid system which are either computer game players or participants in e-learning application. It provides functions to get information about hosted applications, to communicate with other users and to connect to running applications. The portal is implemented as a C++ API which will be included in the application clients to provide a common way of access for the different types of client applications. The communication is achieved via web service calls which allows a secure and standardized way of data transmission.

In order to connect a client application to an application server which is hosted by edutain@grid a communication from the client to the edutain@grid middleware is necessary. This communication is achieved via the client portal. The client portal provides an interface for the client applications to log in to the system, find running application servers, connect to a running server application, etc..

In current computer game clients or e-learning client applications such functionality is provided by an integrated portal. This portal is usually designed to match to the appearance of the application itself. To achieve this property also within edutain@grid clients the client portal functionality is needed to be integrated into these applications. Therefore the portal does not follow the traditional portal definition but is implemented as a C++ library. The API of this library is kept at a high abstraction level so that this portal hides the complexity of the underlying GRIA grid middleware from the application developers and the end users.

To be able to connect to a server application the client portal API provides a function to login to the system. The login function takes a username and password as credential. This information is forwarded via a web service call to the edutain@grid business layer. The business layer then checks the credential for validity and returns a security token if the login was successful. This security

token is later on used in every other communication between the client and the edutain@grid middleware to authenticate the user.

For connecting the client to a running application server a list of available servers has to be provided to the client. Therefore the client portal API provides a method which allows to request all running server instances of a defined ROIAType. The ROIAType is usually preset by the type of the client application in order to find matching server applications. The request is forwarded to the business layer via a web service call. The business layer then checks if the user is allowed to request this information. If so the list of available server applications is replied to the client portal.

After an appropriate server application is selected the client has to get the connection details of the server where the application is hosted. Therefore the client portal API sends an request to the business layer for the IP address and the port of the host for the specified server application. Figure 4 shows the communication chain of this request. At first the client portal API sends a request to the business layer via web service calls containing information about the desired application server. In order to obtain this information the business layer has to communicate with the management layer of the edutain@grid system. This layer is responsible for managing the resources at the different hoster sites. The management layer itself has a connection to all available server hosts. Via this connection the layer can obtain the IP and port information of the server host. This information is returned to the business layer and forwarded to the client portal API as web service call result. The application then gathers the connection details from the client portal API and uses this information to establish the real-time connection to the application server. This example shows that the complexity of the underlying grid communication is successfully hidden from the client application.

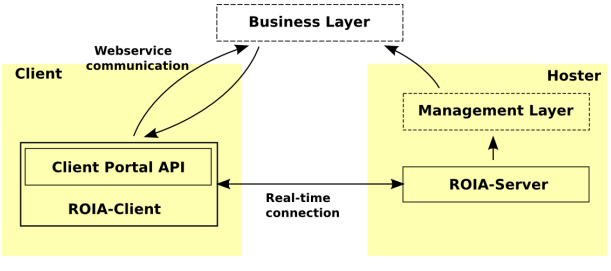

Fig. 4. Client Portal example: Communication path for establishing connection between ROIA-Client and ROIA-Server

7 Conclusions and Future Work

This paper has shown approaches how to support ROIAs in the edutain@grid project by providing a novel portal concept. To offer interfaces for the different

kind of actors the definition of portals has been extended beyond the use of web interfaces.

To communicate efficiently with ROIAs a C++ portal API was provided and described. It is obvious that web portals are not sufficient to support modern computer games, thus Login mechanisms and look-up functionality has been offered in order to be seamlessly integrated into the applications.

To control such applications and to allow for enhanced debugging possibilities on the management layer real-time monitoring, monitoring visualisation and control functionality is offered by the management portal.

In the client portal API community functionality like friend lists or chat should be added in order to support standard game functionality. Packaging the approach of the different portals would lead to a portal framework, which could be used as a generic solution for a class of of interfaces for interactive grid applications.

Acknowledgments

The work described in this paper is supported in part by the European Union through the IST-034601 project "edutain@grid".

References

1. Gria (last visited, October 2008), http://www.gria.org/
2. Gridsphere (last visited, October 2008), http://www.gridsphere.org/
3. Jetspeed-2 (last visited, October 2008), http://portals.apache.org/jetspeed-2/
4. Pgrade (last visited, October 2008), http://www.lpds.sztaki.hu/pgrade/
5. Steam (last visited, October 2008), http://www.steampowered.com/
6. Allan, R., Awre, C., Baker, M., Fish, A.: Portals and portlets 2003. Technical Report UKeS-2004-06, CCLRC e-Science Centre (June 2003)
7. Anthes, C., Landertshamer, R., Hopferwieser, R., Volkert, J.: A novel portal architecture for real-time online interactive applications on the grid. In: 7th Cracow Grid Workshop (CGW 2007), Cracow, Poland, October 2007, pp. 180–187 (2007)
8. Bubak, M., Holger Marten, J.M., Meyer, N., Noga, M., Sloot, P.A.M., Turala, M.: Crossgrid - development of grid environment for interactive applications. In: PIONEER, Poznan, Poland, April 2002, pp. 97–112 (2002)
9. Fahringer, T., Anthes, C., Arragon, A., Lipaj, A., Müller-Iden, J., Rawlings, C.M., Prodan, R., Surridge, M.: The edutain@grid project. In: Veit, D.J., Altmann, J. (eds.) GECON 2007. LNCS, vol. 4685, pp. 182–187. Springer, Heidelberg (2007)
10. Fahringer, T., Jugravu, A., Pllana, S., Prodan, R., Seragiotto Jr., C., Hong-Linh, T.: Askalon: a tool set for cluster and grid computing: Research articles. Concurr. Comput.: Pract. Exper. 17(2-4), 143–169 (2005)
11. Foster, I., Kesselman, C., Tuecke, S.: The anatomy of the grid enabling scalable virtual organizations. International Journal of High Performance Computing Applications 15(3), 200–222 (2001)
12. Gannon, D., Alameda, J., Chipara, O., Christie, M., Dukle, V., Fang, L., Farellee, M., Fox, G., Hampton, S., Kandaswamy, G., Kodeboyina, D., Moad, C., Pierce, M., Plale, B., Rossi, A., Simmhan, Y., Sarangi, A., Slominski, A., Shirasauna, S., Thomas, T.: Building grid portal applications from a web service component architecture. In: IEEE, March 2005, pp. 551–563 (2005)