Scalability of Real-Time Online Applications in Edutain@Grid

S. GORLATCH¹, F. GLINKA¹, A. PLOSS¹, T. FAHRINGER², R. PRODAN², V. NAE², M. SURREIDGE³, S. MIDDLETON³, C. ANTHES⁴, A. ARRAGON⁵, A. LIPAJ⁶ and C. RAWLINGS⁷

¹University of Muenster, Germany
²University of Innsbruck, Austria
³IT Innovation Centre, University of Southampton, UK
⁴Institute of Graphics and Parallel Processing, University of Linz, Austria
⁵Darkworks S.A., France
⁶Amis d.o.o., Slovenia
⁷BMT Cordah Ltd., UK

Abstract: We study a class of Internet-based virtual environments with interactivity and real-time requirements. A relevant representative of this class are Real-Time Online Interactive Applications (ROIA), examples being multi-player online games and high-performance e-learning and training systems. We present how the European project edutain@grid addresses the development of scalable ROIA. After describing the four-layer architecture of edutain@grid, we focus on the Real-Time Framework (RTF) – our novel middleware solution – and report experimental results demonstrating performance and scalability of our approach.

Key-Words: Grid Infrastructure, Real-Time, Interactive Applications, Resource Management, Business Models

1 Introduction

This paper addresses a challenging class of Internet applications – so called Real-Time Online Interactive Applications (ROIA). Current examples of such environments are massively multi-player online computer games and high-performance systems for simulation-based e-learning and training. Because of the very high interactivity and real-time requirements of ROIA, the main difficulty in their development is how to make them scalable, i.e. to maintain the real-time constraints and responsiveness when the number of users increases. This can be achieved by, first, distributing and parallelizing computation in the application design and, then, efficiently supporting computations and communication over multiple servers.

We describe in this paper a comprehensive middleware solution built in the European project edutain@grid that provides: a) a high-level development API for ROIA, and b) a complete Grid runtime support for their scalable multi-server execution. The paper is organized as follows. We describe the challenges of ROIA and then outline the hierarchical edutain@grid architecture and comment on its innovative features. We focus in detail on the Real-Time Framework (RTF) that enables the high-level development and execution of scalable ROIA. Finally, we present the experimental results on scalability and performance and conclude the paper.

2 Real-Time Challenges

Real-time virtual environments comprise the important sector of the entertainment industry, including computer gaming, with high growth rates worldwide. The market of Massively Multiplayer Online Games (MMOG) grew from 10 thousand subscribers in 1997 to 6.7 million in 2003 and the rate is accelerating estimated to 60 million people by 2011.

Regarding the design and implementation, the most widely employed architectural model for online games is the multi-client and multi-server version of the client/server architecture [1, 2, 3]: it consists of a set of servers that are concurrently accessed by a number of users that dynamically interact with each other within a game session. Clients connect directly to the servers and send their play actions such as movements, shootings, collection of items, or chat. Based on the actions submitted by the players, the servers compute the global state of the game world represented by the position and interactions of the entities, store it into a persistent database, and send appropriate real-time responses to the players containing the new relevant state information, typically in a high-speed data stream. In most game genres, the responses must be delivered promptly within a given time interval to ensure a good game experience for all participants, which is crucial for keeping the players engaged, and has an immediate consequence on the income of the game operators.
Typically, the load of one server increases dramatically with the number and density of the hosted players and their interaction within the simulated world. Today, a single computer is limited at around 500 simultaneous and persistent network connections, and databases can manage the update of around 500 objects per second [4]. To support potentially millions of active concurrent players and many more other game entities, Hosters need to install and operate a large dedicated multi-server infrastructure [5], with hundreds to thousands of computers hosting the distributed load of each game [1, 6]. However, due to the dynamic character of ROIAs, both short- and long term, the resource providers often have to over-provision their infrastructure, which leads to a low and inefficient resource utilisation and new providers finding it difficult to join the market. For example, the operating infrastructure of World of Warcraft employs over 10,000 compute servers. To cope with an increasing demand in the number of players connected to extremely popular ROIA sessions, techniques for parallelising and distributing the ROIA load across multiple computational resources are currently being studied in the community [2, 7].

The next generation of distributed online games, for which no massively multiplayer support exists yet, are the fast-paced First Person Shooter (FPS) action games. The FPS type of online games has very specific QoS demands in terms of client state computations and updates. For example, a typical FPS game client has to receive state updates from the server (showing the movements and actions of teammates and opponents) with a frequency of around 35 Hz in order to offer a satisfactory gaming experience.

3 Modeling ROIA

In this section we present a model for the ROIA ecosystem from the resource management perspective, in which a global network of data centres host services that execute many ROIAs at the same time.

ROIAs are large-scale simulations of persistent worlds comprising various objects or entities that we classify in four categories: (1) avatars are representation of the participants; (2) bots or non-player characters (NPC) are mobile entities that have the ability to act independently; (3) movable objects (such as boxes or guns) are passive entities which can be manipulated but do not initiate interactions; and (4) immutable entities or decor.

The architectural model employed for ROIAs is client/server [1], with hosting companies maintaining the servers that simulate a distributed ROIA world. The clients dynamically connect to a joint ROIA session and they execute a so-called real-time loop; in each iteration they interact with each other by sending play actions such as movements, shootings, operations on objects, or chat.

In each loop iteration (also called tick), there are certain steps to be performed: (1) processing events coming from the clients, (2) processing the states of the active entities, and (3) processing state updates from other servers.

Depending on the game, typical response times to ensure fluent play must be below 100 ms in online FPS action games and 1-2 sec for MMORPGs [8]. To ensure scalability and real-time response, a ROIA session is distributed on multiple servers using three different techniques (see Fig. 2), and each participant is mapped to one of the servers, usually to one in its closest proximity to minimise latencies.

Spatial scaling of a game session is achieved through a conventional parallelization technique called zoning [2, 9], based on similar data locality concepts as in scientific parallel processing. Zoning partitions the game world into geographical areas to be
handled independently by separate machines. Zones are not necessarily of same shape and size, but should have an even load distribution.

The second technique called replication targets parallelization of ROIA sessions with a large density of players located and interacting within each other’s area of interest. Such situations are typical to fast-paced FPS action games in which players typically gather in certain hot-spot action areas that overload the ROIA servers which are no longer capable of delivering state updates at the required rate. To address this problem, replication defines a novel method of distributing the load by replicating the same game zone on several CPUs. Each replicated server computes the state for a subset of entities called active entities, while the remaining ones, called shadow entities (which are active in the other participating servers), synchronised across servers. It was proven in previous research that the overhead of synchronised shadow entities is much lower than the overhead of computing the load produced by active entities [6].

The third technique called instancing is a simplification of replication which distributes the session load by starting multiple parallel instances of the highly populated zones. The instances are completely independent of each other, which means that two avatars from different instances will not see each other.

4 Architecture of Edutain@Grid

The edutain@grid architecture comprises four distinct layers as shown in Fig. 3.

The client layer is where primary access from the customers to the edutain@grid platform is performed. Customers ask a coordinator for connection, get information, launch or join, display, and globally interact with the different available ROIAs. The coordinator is an organisation that makes a ROIA instance accessible to its consumers, and coordinates actors in the lower layers to deliver the required ROIA. The underlying Grid and its technical functionalities are completely hidden from this layer.

The business layer is where the different business interactions between the actors are supported. These include the registration of customers with a Coordinator, the establishment of SLA with Hosters to provide the required resources, the licensing of application components and multimedia content, and accounting and billing processes. This layer is implemented using Web Services technologies, and provides services for each actor to support its business interactions with other actors. The advanced SLA technology ensures realistic and viable Quality of Service (QoS) metrics.

The management layer is where Grid-style resource management methods are used to map the requirements of users (expressed through their business-level demands on the coordinator) to the available resources, all subject to the terms of SLA negotiated in the business layer. The management layer should handle the deployment of ROIA software and content to individual hosts, the management of security policy to enable users and hosters to access each other, and the scheduling, monitoring, and steering of ROIA onto the hoster resources to ensure that they meet the terms of their SLA. This layer is implemented using Web Services technologies, though not all services are exposed to the other actors.

5 Real-Time Framework (RTF) and Performance

The real-time layer in Fig. 3 is where the ROIA actually execute, using resources (CPU, file storage and network bandwidth), and security credentials assigned to them by the management layer. Customers that contacted a coordinator within the client-layer for a connection to a ROIA instance can use the information now to connect to one of the executing ROIA processes at a participating hoster. The real-time layer provides scalability to a large number of online customers, as well as fast communication links between application components and users.

5.1 Real-Time Framework

The Real-Time Framework constitutes a grid-aware computation and communication middleware for Real-Time Online Interactive Applications (ROIA). It
offers dedicated functionality for entity-based virtual environment processing which is needed by online games and virtual environment training and e-learning applications. Furthermore, it enables in-application monitoring and controlling, streaming, and integrated persistency for application states. The RTF establishes communication channels between participating clients and servers and communicates events and state updates over these channels in a bandwidth- and latency-optimized manner. The RTF functionality is offered to application developers in form of a C++-based modular library.

The high-level development approach of RTF provides a single interface to the developer which also supports a smooth transition from single-server to multi-server development [10]. RTF’s interface virtualizes the application data distribution along the familiar zoning, instancing and replication concepts for ROIA developers. As the developer no longer communicates with dedicated resources, RTF and edutain@grid can manage the distribution of data and the related computations transparently for the developer, while communication channels are redirected underneath if necessary and data is migrated accordingly to resource management decisions triggered in the remaining edutain@grid layers.

RTF is integrated into the edutain@grid security architecture and allows authenticating clients (single-sign on), encrypted communication connections and authorization support without bothering the application developer or user with details.

5.2 Performance Experiments for RTF

This section reports on preliminary performance experiments that were conducted using the test application suite RTFDemo developed in Muenster following the RTF-development methodology described above. In this test application, users are allowed to participate with their client in an online game scenario controlling a robot avatar in an abstract virtual world. The virtual world allows to be operated in different types, like being run with a single RTFDemo-ROIAProcess or being zoned or replicated on several ROIAProcesses for parallelized and distributed operation.

Using the RTFDemo application suite, the following preliminary performance experiments have been carried on in a local area network:

1. Maximum Number of Clients on a Single RTFDemo ROIAProcess: Test the maximum number of players that is supported by one single server for this test application. This test allows the comparison of performance improvements between different RTF versions.

2. Scalability Test: Maximum Number of Clients in the RTFDemo Zoning Setup: Test the scalability of the zoning approach within RTF for multiple servers. This test checks the maximum number of users possible while still maintaining the application’s real-time requirements for a setup with an increasing number of zones.

The screenshot depicted in Fig. 4 shows a scene from the performance test. The avatar of the client application that was used for the screenshot is marked by the green ellipse and all the other shown robots are computer-controlled clients.

Four different scalability tests were run and Fig. 5 shows the CPU load for the 1, 2, 4 and 8 ROIAProcess setup and various client numbers. The clients were distributed equally between the servers and could migrate between the available zones. The results show that the usage of additional servers significantly increases the number of possible users. For the fast-paced RTFDemo ROIA, the scalability is nearly linear, allowing up to 1200 clients when using eight zones.

Our experimental results show that:

1. The overhead of the current RTF implementation is quite low, allowing client numbers even on a single ROIAProcess (170 clients for the RTFDemo) to favourably compare to those of sophisticated, fast-paced commercial action games (usually up to 64 players, e.g., in Battlefield 2).

2. The multi-server processing using the zoning approach enables a significant increase in the number of participating clients. Zoning with RTF supports a smooth and seamless migration of clients between adjacent zones, thus providing a single, seamless virtual world to the users.
Figure 5: Average CPU load for different ROIA processes setups and client numbers.

6 Related Work and Conclusion

Edtuain@Grid is based on well established software technologies and experiences from the business- [11], resource management- [12] and scalability area [13] that were integrated and extended for the unique requirements of ROIA.

Compared to Grid middleware systems such as Globus [14], gLite [15] and UNICORE [16], which enable high-throughput applications by sharing computational resources for processing and data storage to meet the needs of individual and institutional users, edutain@grid delivers a platform capable to operate secured, multi-hosted ROIA applications, allowing scaling beyond the limits of any one hoster, as well as to consider important in-application characteristics of ROIA like zones, instances and replications during service provision and resource management.

The RTF provides for the ROIA development, where real-time communication and consistent distribution of data and computational load impose a high complexity, a new balance between complexity and flexibility:

- Compared to existing approaches in the field of basic communication middleware like NetZ [17], RakNet [18] or HawkNL [19] a much higher level of abstraction is provided including automatic entity serialization and hiding the details of the network communication.
- Compared to reusable game engines like the Quake [20] or Unreal [21] engines, RTF is much more flexible because it is not bound to a specific graphics engine and leaves the real-time loop implementation to the developer, who is now supported by the high-level mechanisms of RTF for entity and event handling.
- The multi-server capability of RTF allows the application developer to easily incorporate three different parallelization and distribution approaches and is open to be extended in future game designs. This flexible support of different parallelization concepts and its combinations allows RTF to be usable for a broader range of multi-player game concepts than existing multi-server middleware like BigWorld [22], HeroEngine [23] or Virtiverse [24] which are limited to zoning and instancing.
- RTF is integrated with the resource management and business aspects of the ROIA service provisioning within the edutain@grid system. It provides automatic ROIA monitoring and controlling facilities which enable the management layer to use new prediction techniques for the QoS-aware resource management and load balancing. Furthermore, edutain@grid provides a comprehensive security concept with authorization support across all layers [25].

7 Conclusion

In this paper, we described the novel approach of the European edutain@grid project to developing and executing real-time environment in a scalable manner, as the number of players is increasing and must be supported by employing additional parallel servers.

The findings of the project demonstrate that the four-layer architecture of edutain@grid facilitates both the high-level development and the runtime execution of Real-Time Interactive Applications (ROIA). We have described the results of the performance experiments that demonstrated good scalability characteristics under hard conditions for a multi-player gaming environment.

These tests are first performance experiments of the stable implementation. Performance evaluation of RTF will continue and include also the replication parallelization approach and distributed setups. Not only the RTFDemo application will be used for testing, but also the pilot applications developed in the consortium and other evaluation ROIAs developed by partners.

Acknowledgements: This work was supported by the European Union through IST-004265 CoreGRID and IST-034601 edutain@grid projects.
References:


