Developing Infrastructures for Online Games and Distance Learning using RTF

Sergei Gorlatch, Frank Glinka, Alexander Ploss, Chris Rawlings, and Mike Surridge

***Abstract*— The Real-Time Framework (RTF) is a novel development and execution platform for emerging Internet infrastructures and applications with real-time requirements, such as distance learning and multi-player online computer games. In this paper, we describe RTF as part of the edutain@grid service architecture and explain the role distribution between the application developer and the framework. We study in detail the use of RTF for two application use cases: 1) multi-player online games running on multiple servers, and 2) distance learning with frequent interactions over a wide-area network. Then, we report experimental results on the performance and scalability of RTF-based infrastructures and applications. Finally, we formulate the advantages of RTF and the edutain@grid architecture that go beyond the state of the art in the area.**

***Index Terms*— Infrastructures, Distance Learning, Multi-player online games, RTF (Real-Time Framework), Scalability.**

I. Introduction

This paper addresses a challenging class of emerging applications for Internet infrastructures – so called Real-Time Online Interactive Applications (ROIA). Popular examples of ROIA are Massively Multiplayer Online Games (MMOG) and high-performance systems for simulation-based e-learning and training. Because of the very high interactivity and the 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 by adding resources (server machines) when the number of users increases. This can be achieved by, firstly, distributing and parallelizing computations in the application design and, secondly, by efficiently supporting computations and communication across the infrastructure.

Internet infrastructures are increasingly used for the socially important subclass of ROIA applications called *edutainment* (*edu*cation + enter*tainment*) including computer gaming and distance learning, with high growth rates worldwide. For example, the market of MMOG grew from only 10 thousand subscribers in 1997 to 6.7 million in 2003 and to an estimated 60 million people by 2011.

In the paper, we describe the Real-Time Framework (RTF) that enables the high-level development and efficient execution of ROIA for Internet infrastructures. The paper is organized as follows. We describe the challenges of ROIA and then outline the hierarchical infrastructure developed in the European edutain@grid project and comment on its innovative features. We focus on two use cases for RTF: 1) multi-player online computer games, and 2) infrastructures for online distance learning. Finally, we present our experimental results on scalability and performance of infrastructures and applications developed using RTF.

Sergei Gorlatch (corresponding author), Frank Glinka, and Alexander Ploss are with the University of Muenster, Einsteinstr. 62, 48149 Germany (phone: +49 251 8332740; fax: +49 251 8332742; e-mail: gorlatch@uni-muenster.de).

Chris Rawlings is with the BMT Cordah Ltd., UK

Mike Surridge is with the IT Innovation Centre, Southampton, UK

II. RTF as part of Edutain@Grid

The execution model employed for ROIA is client/server [1], with hosting companies providing an infrastructure with multiple servers that simulate a distributed ROIA world. The clients dynamically connect to a joint application session and execute a so-called *real-time loop* (Figure 1); they interact with each other by issuing application-specific actions such as movements, shootings, operations on objects, or chat commands.

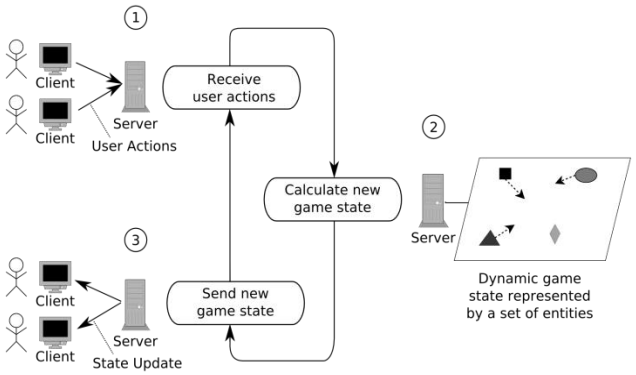

Figure 1: The real-time loop model.

At each loop iteration (also called *tick*), the three steps shown in Figure 1 are performed: (1) processing the events coming from the clients, (2) calculating the state of the active entities, and (3) exchanging state updates between servers. This process must be highly responsive for the user: it takes usually only a few milliseconds between sending the inputs and receiving a new state. Depending on the game, typical response times to ensure fluent play must be below 100 ms in online FPS (First-Person Shooters) and 1-2 sec for role-playing MMOGs which are called MMORPG [3].

A typical ROIA application consists of a *client application* and a *ROIA process*. Users employ the client application to connect to the ROIA process and send their inputs which directly influence the application state. The application state is updated and then received by each client connected to the ROIA process.

The Real-Time Framework (RTF) is part of the European edutain@grid architecture for the development, business management and hosting of interactive distributed infrastructures and applications. This comprehensive service architecture built within the European project edutain@grid [1] provides: a) a high-level development API for ROIA, and b) a complete runtime support for their scalable multi-server execution. With this architecture, Internet

ISBN: 978-988-19251-1-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

infrastructures and applications can use on-demand hosting: there is no need to possess and manage an own infrastructure. The maximum possible number of users using a ROIA can be negotiated with the hoster (resource provider), which will be responsible for the needed computing performance and communication bandwidth at the time contracted. These services are enabled by the Business and Management layers of edutain@grid. The ROIA themselves are executed on the assigned resources in the Real-Time layer implemented by RTF [2].

The high-level development approach of RTF provides a developer's interface that also supports a smooth transition from single-server to multi-server infrastructures [8]. RTF is integrated into the edutain@grid security architecture and allows clients to benefit from authentification, encrypted communication and authorization support, without bothering the application developer or user with details.

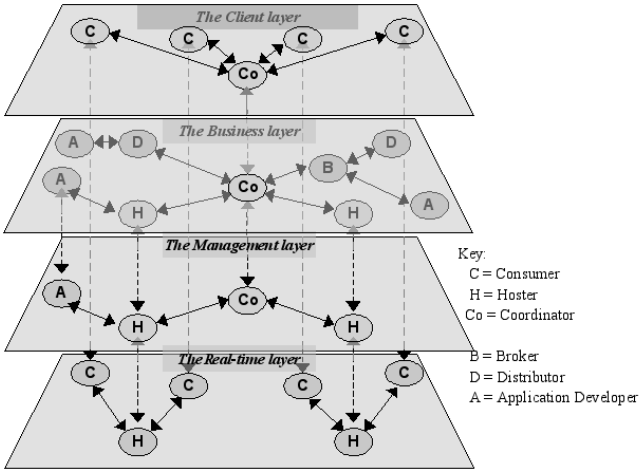

Figure 2: The four layers of the edutain@grid architecture.

Figure 2 shows that the edutain@grid service-oriented architecture comprises four layers: *client*, *business*, *management*, and *real-time* layers, explained in the following.

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

The business layer is where the different business interactions between the actors of the infrastructure are supported. These include the registration of customers with the Coordinator, the establishment of Service-Level Agreements (SLA) with hosters who provide the required resources, the licensing of application components and multimedia content, and accounting and billing. This layer is implemented using Web Service 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 resource management methods are used to map the requirements of users (expressed through their business-level demands to the coordinator) onto available infrastructure resources, all subject to the terms of SLA negotiated in the business layer. The management layer handles the deployment of ROIA software and content to individual hosts, the management of a security policy to enable users and hosters to access each other's data, as well as 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.

The real-time layer is where applications are actually executed using infrastructure resources (CPU, file storage, and network bandwidth) and security credentials assigned to them by the management layer. Customers contact a coordinator within the client-layer and use the obtained information to connect to one of the running ROIA processes at a participating hoster. The real-time layer provides scalability w.r.t. the potentially increasing numbers of online customers, as well as fast communication links between application components and users.

The Real-Time Framework (RTF) provides a high-level communication and computation middleware for ROIA. RTF supports both the server-side and client-side processing on the infrastructure. The role distribution between the framework and the application developer is as follows. RTF deals with the entity and event handling in the real-time client loop and with the continuous state processing in the real-time server loop, as well as with the distribution of the state processing across multiple servers. The developer implements the application-specific real-time loop on client and server, as well as the application logic, using RTF to automatically exchange information between the processes.

RTF is implemented as a C++ library, because C++ is nowadays the language of choice in performance-critical applications. RTF supports in-application monitoring and controlling, streaming, and integrated persistency for the application state. The communication of events and state updates is performed in a bandwidth- and latency-optimized manner. The high-level development approach of RTF provides a developer's interface that also supports a smooth transition from single-server to multi-server infrastructures [8]. RTF is integrated into the edutain@grid security architecture and allows clients to benefit from authentification, encrypted communication and authorization support, without bothering the application developer or user with details.

III. Use Case: Online Games

The majority of today's online games simulate a spatial virtual world which is conceptually separated into a static part and a dynamic part. The static part covers, e.g., environmental properties like the landscape, buildings and other non-changeable objects. The dynamic part covers *entities* like avatars, non-playing characters (NPC) controlled by the computer, items that can be collected by players or, generally, objects that can change their state. Both parts, together, build the game state which represents the game world at a certain point of time.

When designing the real-time loop (Figure 1) for a particular game application, the developer has to deal with several tasks. In steps 1 and 3 of the loop, the developer organizes the network transfer of the data structures that

ISBN: 978-988-19251-1-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

realize user actions and entities. If the game is distributed on the infrastructure between multiple servers, then step 2 also requires the developer to organize the distributed state computation and necessary communications for its update across different servers.

There are three major parts comprising an online game:

- *Game logic*, including entities, events (implemented as data structures), and processing rules for the virtual environment;
- *Game engine*, which continuously processes user actions and events in the real-time loop to compute a new game state, according to the game logic;
- *Game distribution*, including partitioning the game world on multiple servers and dynamic management of distributed computation and communication in the infrastructure.

These three aspects are treated differently, depending on the requirements and properties of a particular game genre. For example, fast-paced action games strongly rely on an efficient communication and high-performance engine implementation while using a relatively simple game logic.

The most widely employed infrastructure model for online games is the multi-client and multi-server version of the client/server architecture [4,5]: it consists of a set of servers that are concurrently accessed by a number of users that dynamically and actively interact with each other within a game session. Clients connect directly to the servers and issue their play actions such as movements, shootings, collection of items, or chat commands. 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 to the players appropriate real-time responses containing the new state information, typically in a high-speed data stream.

Typically, the load of one server increases with the rising number and density of the involved players and their interaction within the simulated world. Today, a single computer is limited to support around 500 simultaneous and persistent network connections, and databases can manage the update of around 500 objects per second [6]. To support potentially millions of active concurrent players and even many more other game entities, hosters need to install and operate a large dedicated multi-server infrastructure [7], with hundreds to thousands of computers hosting the load of each game [4,8]. However, due to the dynamic character of games, both in short and long term, the resource providers often have to over-provision their infrastructure, which leads to an inefficient resource utilisation, such that new providers find 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 in popular games, techniques for parallelizing and distributing the load across multiple resources have been extensively studied [5,9].

To ensure scalability and real-time response, a game session is distributed in RTF over multiple servers using three different techniques (see Figure 3): zoning, replication, and instancing.

Spatial scaling of a game session is achieved through a parallelization technique called *zoning* [2] that partitions the game world into disjoint areas to be handled by separate machines. Zones are not necessarily of the same shape and size, but preferably have an even load distribution.

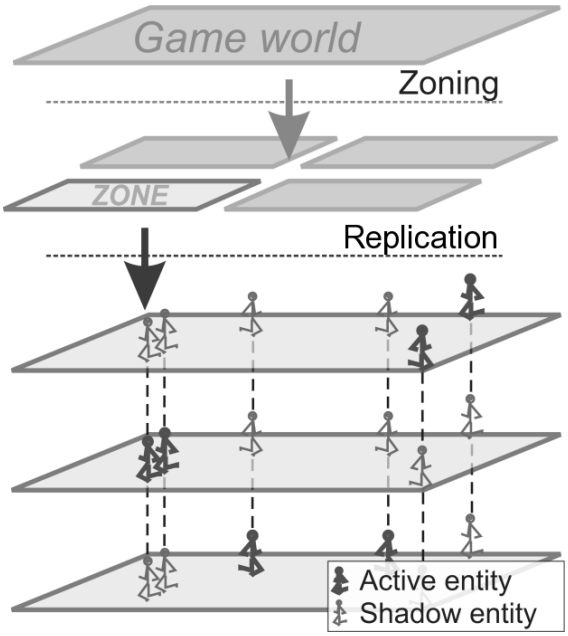

Figure 3: Zoning and Replication.

Our novel distribution technique called *replication* targets games with a large density of players as, e.g., fast-paced FPS action games in which players typically gather in certain hot-spot areas; thus, the servers become overloaded and are no longer able to deliver state updates at the required rate. To address this problem, we replicate the same zone on several servers. Then, each server computes the state for a subset of entities called its *active entities*, while for the remaining *shadow entities*, the state is obtained from the servers where these entities are active, i.e. a synchronization between servers takes place. We demonstrated in previous research that the overhead of synchronizing shadow entities is usually significantly lower than the overhead of computing the load produced by active entities [8].

The third distribution technique called *instancing* is a simplification of replication: it distributes the session load by starting multiple parallel instances of the highly populated zones. The instances are completely independent of each other, i.e. two avatars from different instances will not see each other, which is a restriction for some games.

IV. Use Case: Distance Learning

The work described in this section was conducted in cooperation with BMT Cordah Ltd. (BMT) [10] - a commercial company offering consultancy and software.

We consider a particular use case of distance learning for the Search and Rescue Information System (SARIS) [11] of BMT. Figure 4 shows a screenshot of SARIS as used by coastguards, navies and port authorities. It allows these organizations to manage incidents, for example, when a sailor has fallen overboard. To react to such an incident, it is important to estimate where the target (e.g., the sailor) is located, which is done by SARIS based on environmental data. The operation of SARIS has to be taught in courses in order to use it properly.

Distance learning using the Internet infrastructure reduces

ISBN: 978-988-19251-1-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

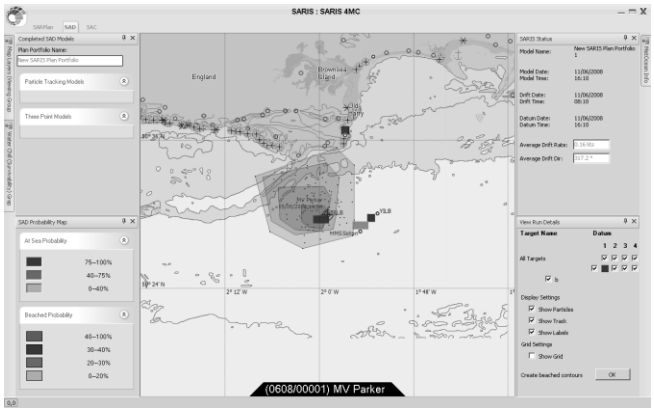

Figure 4: SARIS screenshot (provided as a courtesy of BMT Cordah Ltd).

the overall costs of training: Instructors and course attendees will use the software at their workplace. It enables users to interact with other users in the same way as they would do in an ordinary course. It is possible to interact with the instructor or other attendees, watch the simulation of incidents, and hold and mark examinations. The data that has to be exchanged between attendees and instructors is transmitted via the Internet.

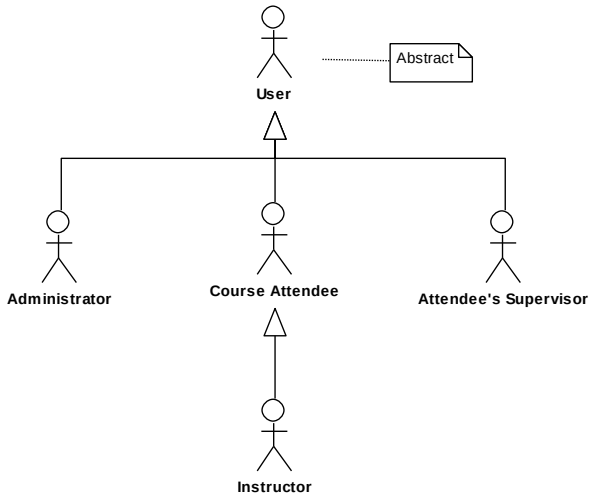

Figure 6: Hierarchy of actors in a distance learning infrastructure.

The people interacting with the system are depicted in an actor hierarchy (see Figure 6). Administrators create customer accounts and grant access to courses or tests. An instructor creates lessons and starts them. During a lesson, he is also able to communicate with customers over several channels such as text communication and to modify the lesson scenario as needed (e.g. trigger additional incidents). Employees of the customers can join ordered lessons and communicate with other students or the instructor. Moreover, they can take a test. Managers of the customer's company can fetch test results to rate the abilities of the employees. All users are subtypes of the actor User, i.e., are able to interact with the system. The subactors of User are Administrator, Course Attendee and Attendee's Supervisor. The Administrator is responsible for management tasks and is able to enforce business contracts, signed with customers, in the system. A Course Attendee is the user who mainly interacts with the system. Basically, he is an employee of the customer's organization who takes part in e-learning sessions. A special subtype of the Course Attendee is the Instructor who is responsible for administrative intervention in e-learning sessions. Furthermore, he has the same opportunities as a customer's employee and is therefore an extended subactor of the Course Attendee.

Most applications in the field of distance learning rely on a client-server infrastructure. Usually, a learning ‘session’ is being conducted at an instructor organisation, with session attendees all connecting to a single point source. It is evident that no matter the speed of connection and bandwidth available to the client attendee, the participants are throttled back to the connection available to the instructor organization (typically about 2 Mbit in the figure).

It is preferable to have a situation where the learning session ‘process’ can be independently deployed to a partner organisation that is competent in providing and hosting high-performance resources (for both networking and computation). This partner is called *hoster*; a significantly improved networking performance is expected in an infrastructure with hoster as demonstrated in Figure 5: the lesson session is no longer hosted at the instructor location. The relative connection bandwidth for instructor and clients remains unchanged, but the hoster in the figure has a 100Mbit connection which clearly meets the requirements of learning sessions.

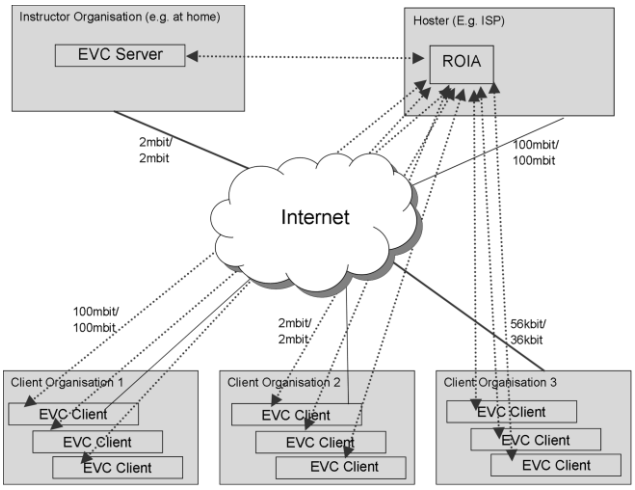

Figure 5: Client-Server infrastructure with hoster for distance learning.

The instructor organisation keeps control over the business-relevant topics of access control, billing and service level agreements (SLA) while the service instances (lessons) are hosted on foreign resources. Connecting clients are authenticated by RTF against the business identities (single-sign-on), credentials for participation are checked and encrypted communication channels are established as illustrated in Figure 5. RTF furthermore allows using more than one hoster in the infrastructure for a single lesson session, e.g., one in Europe and one in US in order to minimise global distribution bandwidth by replicating/sharing load between cross-Atlantic processes.

The Edutain Virtual Classroom (EVC) application is utilising the functionality offered by RTF. The communication subsystem of the infrastructure is based on RTF, and ROIA applications are created on both client and server sides. The applications provide entry points, via the integration of RTF and the GRIA business collaboration infrastructure, where the student can search for lessons using a directory service and student numbers and associated

ISBN: 978-988-19251-1-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

Figure 7: EVC View as seen by the instructor.

pricing can be controlled by SLA respectively. Control of permissions for students (entering a lesson and activities during lesson time) is provided by the RTF security plugin.

The different aspects of the instructor's application screen are as follows, see Figure 7:

1 – is a Webcam view of the instructor, streamed to all of the connected students.

2 – is the current client list, showing IP address and port number used for communication. The instructor allocates and removes permissions through context menu selection as to whether the student can take control of the demonstrated application seen in area 3.

3 - is the main demonstration window showing an application being explained, taught or demonstrated. The application itself is reparented into the EVC workspace; the students see a facsimile view that the instructor sees. In the example above, a VMWare server running Ubuntu 9.10 is being demonstrated via the EVC. If the instructor permits the student to take control, the mouse movements and key presses within area 3 will be transmitted to the instructor.

4 - is the text chat area: messages are sent between all participants – i.e. from the instructor to everybody or from a single student to every other student.

V. Performance Experiments for RTF

This section reports our performance experiments that were conducted using the multi-player online game *RTFDemo* which was designed and implemented following the RTF development methodology. In this test application, users (clients) control their robot avatars which can move, shoot at each other, etc. The virtual world can be operated in different ways, e.g., being zoned or replicated on several ROIA processes for parallel and distributed operation.

Using the RTFDemo application suite, several performance experiments have been conducted in a local area network infrastructure. We tested the maximum number of players that is supported by one single server for this test application. We also tested the scalability of the zoning approach within RTF for an infrastructure with multiple servers. This test checks the maximum number of users still able to play while maintaining the application's real-time requirements for an increasing number of zones.

The screenshot in Figure 8 shows a scene from the performance test. The avatar of the client application that was used for making the screenshot is marked by the green ellipse and all the other robots in the figure are computer-controlled clients. Four different scalability tests were run, and Figure 9 shows the measured CPU load for the 1, 2, 4 and 8 ROIA process setup and various client numbers. The clients were distributed equally between the servers of the infrastructure 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, the scalability is nearly linear, allowing up to 1200 clients when using eight zones.

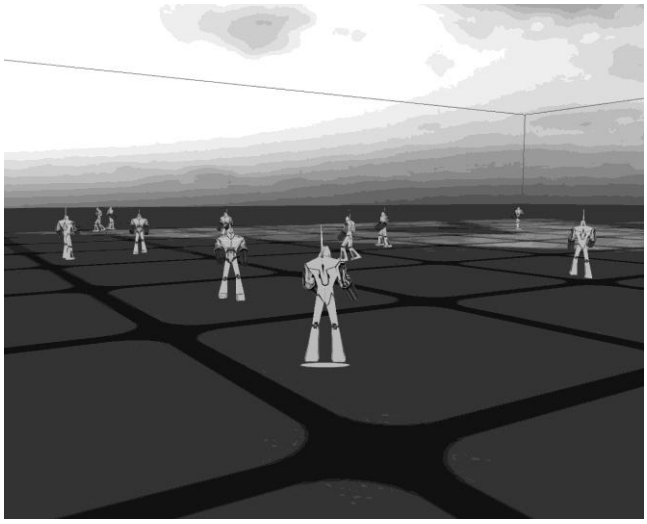

Figure 8: RTFDemo screen.

Our experimental results show that:

1. The overhead of the current RTF implementation is quite low, allowing the client numbers even on a single ROIA process (170 clients for RTFDemo) to favorably compare to those of fast-paced commercial action games (usually up to 64 players, e.g., in *Battlefield 2*).
2. The multi-server version using the zoning approach enables a significant increase in the number of participating clients. RTF supports a smooth and seamless migration of clients between adjacent zones, i.e. yielding a single seamless virtual world to users.

VI. Related Work and Conclusion

In this paper, we describe the novel approach of the European edutain@grid project to developing and executing real-time virtual environments using Internet infrastructures in a scalable manner, as the number of users is increasing and can be supported by multiple compute servers.

The four-layer architecture of edutain@grid facilitates

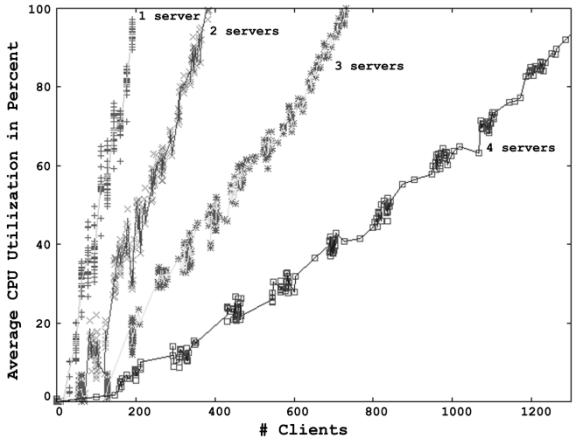

Figure 9: Average CPU load for different setups and client.

ISBN: 978-988-19251-1-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

both the high-level development and the efficient runtime execution of Real-Time Interactive Applications (ROIA). Our performance experiments demonstrated good scalability characteristics under hard conditions for a multi-player gaming environment.

The Edutain@Grid architecture is based on well-established software technologies and experience from the business [12], resource management [13] and scalability area [14] that were extended, optimized and integrated for the specific requirements of ROIA.

Compared to middleware systems for Grid infrastructures, such as Globus [15], gLite [16] and UNICORE [17], which enable high throughput by sharing computational and storage resources among individual and institutional users, edutain@grid delivers a novel platform for secured, multi-hosted ROIA applications, allowing them to scale beyond the limits of one arbitrary hoster, as well as to take into account such important in-application characteristics of ROIA like zones, instances and replications during service provision and resource management.

RTF frees the application developer from complicated low-level programming, e.g., network transmission via sockets. The RTF is implemented as an object-oriented C++ library with a comfortable user API.

The main contributions of RTF beyond the state of the art in developing performance-critical Internet applications in the areas of gaming and distance learning are as follows:

- Compared to existing approaches in the field of basic communication middleware like Net-Z, RakNet or HawkNL, a higher level of abstraction is provided including automatic serialization and hiding the details of the network communication.
- Compared to reusable game engines like *Quake* or *Unreal*, RTF is significantly 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 their combinations and is open for extension in future game designs. This flexible support of different parallelization concepts and their combinations allows RTF to be usable for a broader range of multi-player game concepts than existing multi-server middleware products like *BigWorld*, *HeroEngine* or *Virtiverse*, 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 [12].

In this paper, we show how a novel e-learning system – the Edutain Virtual Classrom (EVC) - is developed using the RTF. We demonstrate how RTF facilitates a comfortable and efficient development of new e-learning products acting in a distributed Internet-based infrastructure.

ACKNOWLEDGEMENT

This work was partially supported by the European Commission (the research project edutain@grid and the Network of Excellence S-Cube).

Bibliography

[1] *edutain@grid project*. Available: http://www.edutain.eu

[2] F. Glinka, *et al.*, "RTF: A Real-Time Framework for Developing Scalable Multiplayer Online Games," in *NetGames 2007: Proceedings of 6th Annual Workshop on Network and System Support for Games*, Melbourne, 2007, pp. 81-86.

[3] M. Claypool and K. Claypool, "Latency and player actions in online games," *Communications of the ACM*, vol. 49, pp. 40-45, 2006.

[4] R. Bartle, *Designing Virtual Worlds*: New Riders Games, 2003.

[5] W. Cai, *et al.*, "A Scalable Architecture for Supporting Interactive Games on the Internet," in *Proceedings of the 16th Workshop on Parallel and Distributed Simulation*, ed. Washington, D.C., 2002, pp. 60-67.

[6] W. White, *et al.*, "Database research opportunities in computer games," *SIGMOD Rec.*, vol. 36, pp. 7-13, 2007.

[7] A. Shaikh, *et al.*, "On demand platform for online games," *IBM Syst. J.*, vol. 45, pp. 7-19, 2006.

[8] J. Müller and S. Gorlatch, "Rokkatan: scaling an RTS game design to the massively multiplayer realm," *ACM Computers in Entertainment*, vol. 4, p. 11, 2006.

[9] A. Bharambe, *et al.*, "Colyseus: a distributed architecture for online multiplayer games," in *NSDI'06: Proceedings of the 3rd conference on 3rd Symposium on Networked Systems Design & Implementation*, ed. San Jose, CA: USENIX Association, 2006, pp. 155-168.

[10] *BMT Cordah Limited*. Available: http://www.bmtcordah.com

[11] *Search and Rescue Information System.* Available: http://media.bmt.org

[12] M. Surridge, *et al.*, "Experiences with GRIA - industrial applications on a web services grid.," *e-Science 2005*, pp. 98-105, 2008.

[13] T. Fahringer, *et al.*, "ASKALON: A Development and Grid Computing Environment for Scientific Workflows, Workflows for e-Science," I. J. Taylor, *et al.*, Eds., ed: Springer London, 2007, pp. 450-471.

[14] J. Müller, *et al.*, "Rokkatan: Scaling an RTS Game Design to the Massively Multiplayer Realm," in *ACM SIGHCHI International Conference on Advances in Computer Entertainment Technology (ACE 05)*, ed. Valencia, Spain: ACM, 2005, pp. 125-132.

[15] I. Foster and C. Kesselmann, "Globus: A metacomputing infrastructure toolkit.," *Int. J. of Supercomputer Applications*, vol. 11, pp. 115-128, 1996.

[16] Karl Czajkowski. (2004). *The WS-resource framework.* . Available: http://www.globus.org/wsrf/specs/ws-wsrf.pdf

[17] D. Breuer, *et al.*, Eds., *Scientific computing with UNICORE* (NIC Symposium. 2004, pp. 429-440.