A Dynamic Resource Management System for Real-Time Online Applications on Clouds

Dominik Meiländer, Alexander Ploss, Frank Glinka, and Sergei Gorlatch

University of Muenster, Germany
{d.meil,a.ploss,glinkaf,gorlatch}@uni-muenster.de

Abstract. We consider a challenging class of highly interactive virtual environments, also known as Real-Time Online Interactive Applications (ROIA). Popular examples of ROIA include multi-player online computer games, e-learning and training applications based on real-time simulations, etc. ROIA combine high demands on the scalability and real-time user interactivity with the problem of efficient and economic utilization of resources, which is difficult to achieve due to the changing number of users. We address these challenges by developing the dynamic resource management system RTF-RMS which implements load balancing for ROIA on Clouds. We illustrate how RTF-RMS chooses between three different load-balancing actions and implements Cloud resource allocation. We report experimental results on the load balancing of a multi-player online game in a Cloud environment using RTF-RMS.

1 Introduction and Related Work

This paper is motivated by the challenges of the emerging class of *Real-Time Online Interactive Applications (ROIA)*. Popular and market-relevant representatives of these applications are multi-player online computer games, as well as real-time training and e-learning based on high-performance simulation. ROIA are characterized by: short response times to user actions (about 0.1-1.5 s); frequent state computation (up to 50 Hz); large and frequently changing number of users in a single application instance (up to 10^4 simultaneously).

The high demands on ROIA performance usually cannot be satisfied by a single server. Therefore, distributed, multi-server application processing with suitable scalability concepts is required. A major problem in this context is the efficient utilization of multiple server resources for application provision, which is difficult to achieve for ROIA due to a variable number of users who are continuously connecting to the application and disconnecting from it.

In this paper, we study how Cloud Computing with its *Infrastructure-as-a-Service (IaaS)* approach can be utilized for ROIA applications. We are developing a novel resource management system for ROIA, called *RTF-RMS*, that is implemented on top of the *Real-Time Framework (RTF)* [1]. RTF is our middleware platform for a high-level development of scalable ROIA that provides a rich set of distribution and monitoring mechanisms. RTF-RMS utilizes a

M. Alexander et al. (Eds.): Euro-Par 2011 Workshops, Part I, LNCS 7155, pp. 149–158, 2012.

Cloud infrastructure to lease virtualized hardware resources on demand and provides dynamic load balancing for ROIA, taking into account the monitoring data provided by RTF.

A number of research projects studied the potential of Cloud Computing for cost-efficient provision of particular application classes, e.g., batch execution systems [2], scientific astronomy workflows [3], and market-oriented resource management systems [4]. However, none of these approaches targets applications with requirements similar to ROIA. Rather than general challenges for efficient Cloud resource management, e.g., different performance characteristics of identical resource types [5] or data inconsistency in Cloud storage solutions [6], this work addresses the particular challenges of cost-effectively leasing Cloud resources and reducing their startup times for ROIA.

We showed the impact of virtualized Cloud resources on concrete ROIA implementations in our previous work [7] which proved that the performance requirements of ROIA can be fulfilled on commercial Cloud systems like the Amazon EC2 [8]. In [9], the influence of Cloud resources allocation on the hosting process of multi-player online games was analyzed, showing that load balancing is a critical challenge for ROIA provision on Clouds, as targeted in this paper.

Cloud platforms offer services that provide monitoring and load balancing for their Cloud resources, e.g., Amazon Cloud Watch or Amazon Elastic Load Balancing [8]. However, these services only provide generic system information about resource utilization (CPU, memory, bandwidth, etc.). This information is not sufficient for up- and down-scaling of ROIA sessions since ROIA have a specific runtime behaviour: e.g., regardless of the current number of users, an online game may run with a constant CPU load of 100 % in order to deliver the highest state update rate possible. In this paper, we aim at a dynamic load-balancing strategy based on application-specific monitoring information (e.g., update rate) that is more suitable for ROIA.

The paper is organized as follows. Section 2 describes our target class of Real-Time Online Interactive Applications (ROIA) and the Real-Time Framework (RTF) for their development and execution. Section 3 presents our new dynamic resource management system RTF-RMS and illustrates its strategy for load balancing and Cloud resource allocation. Section 4 presents our experimental results in a Cloud environment, and Section 5 concludes the paper.

2 Scalable ROIA Development with RTF

Typically, there are multiple users in a Real-Time Online Interactive Application (ROIA) who access a common application state and interact with each other concurrently within one virtual environment. The users connect to the application from different client machines and control their avatars that interact with other users' avatars or computer-controlled characters (*entities*). Since ROIA usually have very high performance requirements, the application state processing should be performed on multiple servers. Hence, ROIA are highly distributed applications with new challenges for application development and provision, such

as: short response times to user actions, high concurrency, frequent state computation, ad-hoc user connections, and variable numbers of users.

We use the *real-time loop* model [10] for describing ROIA execution on both physical and Cloud resources. One iteration of the real-time loop is called a *tick*. The time required for one iteration of the real-time loop (*tick duration*) is directly related to the application's response time to user actions, and, hence, is a suitable criterion for dynamic up- and down-scaling of ROIA sessions. A loop iteration consists of three major steps (on the left-hand side of Fig. 1):

1. Servers receive user actions from the clients connected to them.
2. Servers compute a new application state according to the application logic.
3. Servers send the new state to their clients and to other servers.

Steps 1 and 3 involve communication to transmit the user inputs and state updates between processes. The computation of a new application state (step 2) involves quite compute-intensive calculations which apply the application logic to the current state, taking into account the newly received users' actions.

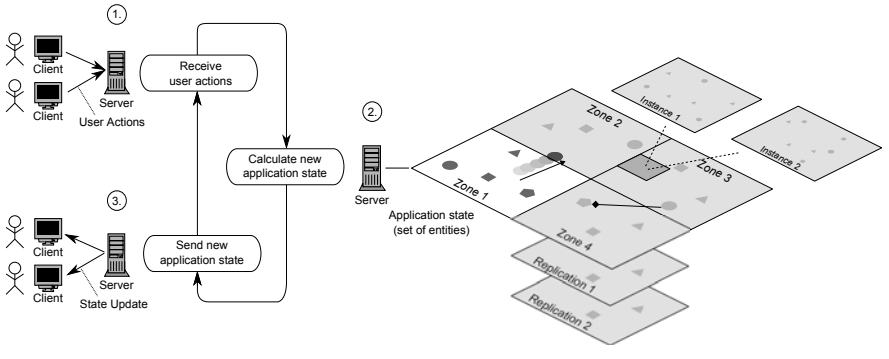

Fig. 1. One iteration of the real-time loop (left); RTF distribution methods (right)

The *Real-Time Framework (RTF)* [11] is our high-level development platform for ROIA which supports the application developer in three essential tasks:

1. *Application State Distribution:* RTF supports three major methods of partitioning the virtual environment among servers (on the right-hand side of Fig. 1): zoning, replication and instancing, and combinations of them. Zoning assigns the processing of the entities in disjoint areas (*zones*) to distinct servers. Instancing creates separate independent copies of a particular zone; each copy is processed by a different server. In the replication approach, each server keeps a complete copy of the application state, but each server is responsible for computing a disjoint subset of entities. A more detailed description of the distribution methods in RTF can be found in [11].

2. *Communication Handling:* RTF provides automatic serialization for entities to be transferred, implements marshalling and unmarshalling of data types, and optimizes the bandwidth consumption of communication.
3. *Monitoring and Distribution Handling:* RTF offers a high-level Hoster Management Interface (HMI) which allows developers to send controlling commands directly into a running RTF-based server application (*ROIA process*), as well as to receive monitoring values from RTF inside a ROIA process. Controlling commands include, e.g., adding physical resources, changing the game world in an online game during runtime, etc.

3 Resource Management for ROIA on Clouds

A major problem for an efficient ROIA execution is the economical utilization of server resources, which is difficult to achieve due to a continuously changing number of users. Cloud Computing promises the potential to distribute application processing on an arbitrary number of resources and to add/remove resources on demand. We develop a Resource Management System (RTF-RMS) as a software component on top of RTF for dynamic up- and down-scaling of ROIA sessions on Cloud resources during runtime.

3.1 The RTF-RMS Architecture

RTF-RMS consists of three software components which are shown in Fig. 2 (left-hand side) and described in the following.

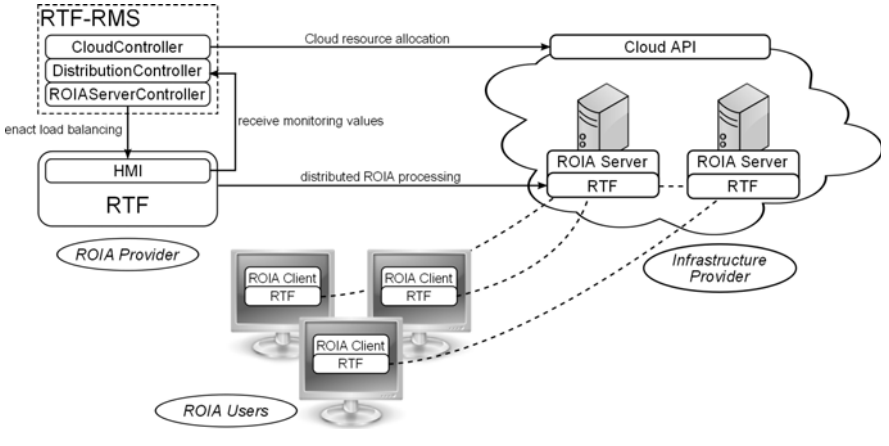

Fig. 2. Composition of RTF-RMS and RTF

The `CloudController` implements the communication with the interface (API) of the Cloud system. RTF-RMS currently implements Cloud resource allocation for the Amazon Compute Cloud (EC2). A particular challenge in the implementation of the `CloudController` is the compensation of a comparatively long startup time of Cloud resources that can be up to several minutes. Furthermore, cost optimization is an important issue since Cloud resources are typically leased for a certain usage period. We address these challenges in Section 3.3.

The `DistributionController` implements our new load-balancing strategy that dynamically changes the workload distribution between application servers. For this purpose, it receives monitoring values from HMI which are used to determine the load of application servers. If the `DistributionController` identifies an overloaded server, it chooses between three different load-balancing actions, as described in Section 3.2.

The `ROIAServerController` enacts load-balancing actions on request from the `DistributionController`, e.g., adding a new resource to the application processing. The `ROIAServerController` uses HMI's distribution handling functionalities (Section 2) to integrate new servers in the application session.

3.2 A Load-Balancing Strategy for ROIA on Clouds

In order to identify the demand for load-balancing actions, a suitable workload analysis is required. RTF-RMS is configured for the workload analysis of a particular application by specifying an *application profile* that defines the monitoring values of interest, e.g., tick duration in ms, update rate in Hz, etc., see Listing 1 for an example. RTF-RMS allows the application developer to specify upper and lower thresholds for each monitoring value in the profile. If a monitoring value exceeds the upper threshold, a new resource is added to the application processing; if monitoring values drop below the lower threshold then the dispensable resources are removed. The application profile also allows application developers to define the maximum number of active replicas for each zone, in order to limit the overhead for inter-server communication caused by the replication. The developer can find suitable thresholds for all parameters in the application profile by considering the runtime behaviour of the application on physical resources and calculating the virtualization overhead, or by conducting benchmark experiments in the Cloud environment.

Listing 1. Example application profile for a fast-paced action game

```
<appProfileData>
  <maxNumReplications>2</maxNumReplications>
  <metric>
    <name>TickDuration</name>
    <monitoringInterval>1</monitoringInterval>
    <upperThreshold>40</upperThreshold>
    <lowerThreshold>10</lowerThreshold>
  </metric>
</appProfileData>
```

RTF-RMS implements workload analysis as follows. The `DistributionController` creates for each application session a `TimerTask` object, responsible for collecting the monitoring values specified in the profile and received from RTF. Each `TimerTask` registers an event listener at RTF which updates the monitoring values with the frequency specified in the application profile.

In RTF-RMS, we implement a load-balancing strategy for Cloud resources that chooses between the following three load-balancing actions:

1. *User Migration*: users are migrated from an overloaded server to an underutilized server which is replicating the same zone. For this purpose, RTF-RMS uses HMI's `migrate` method that switches user connections from one server to another. The `migrate` method expects a so-called `MigrationPolicy` which specifies the amount of migrated users; RTF-RMS provides by default a `MigrationPolicy` that distributes an equal amount of users onto each application server for a particular zone.
   User migration is the preferred action if the load of an overloaded server can be compensated by running resources.
2. *Replication Enactment*: new game servers are added in order to provide more computation power to the highly frequented zone. For this purpose, RTF-RMS uses HMI's `activateReplication` method that assigns a new server to the processing of a new replica; a number of users are migrated to the new replica in order to balance the load. Replication implies an additional inter-server communication and thus, its scalability is limited.
   If the number of active replicas for a particular zone is below the maximum number of replicas specified by the profile, then replication is used; otherwise the resource substitution action (described next) is preferred.
3. *Resource Substitution*: an existing resource in the application processing is substituted by a more powerful resource in order to increase the computation power for highly frequented zones. RTF-RMS replicates the targeted zone on the new resource (using `activateReplication`) and migrates all clients to the new server (using `migrate`). The substituted server is shut down.

In order to choose the optimal load-balancing action, the `DistributionController` creates *reports* for each application server (Fig. 3). A report describes the load of a particular server, e.g., current tick duration in ms. A *zone report* is created for each zone by collecting all reports from servers involved in the zone processing and calculating the average load of these servers. The zone report is the basis for choosing the load-balancing action as described above.

Fig. 3 illustrates an example of how zone reports are used to choose a load-balancing action. We assigned Server A to the processing of Zone 1, and Server B and C were assigned to Zone 2. The zone report of Zone 1 identifies an average tick duration of 45 ms. Given an upper threshold of 40 ms, RTF-RMS decides to enact a new replication for Zone 1. The zone report of Zone 2 identifies an average tick duration of 30 ms which is below the threshold, but Server B has a tick duration of 45 ms. Hence, RTF-RMS migrates users from Server B to Server C in order to distribute the workload equally on both servers.

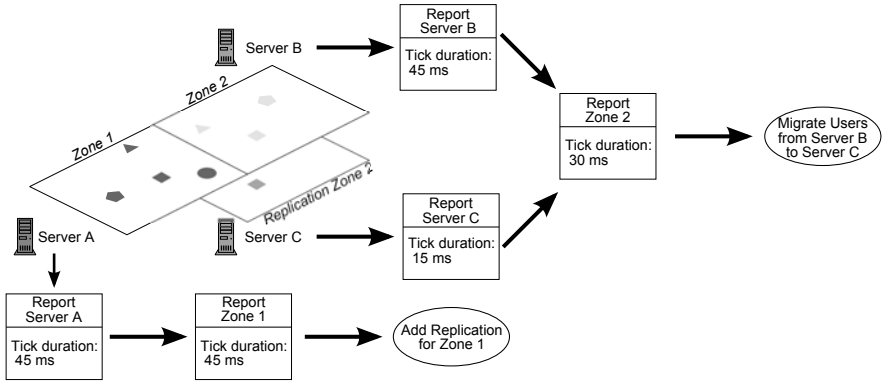

Fig. 3. Finding a suitable load-balancing action using zone reports

3.3 Cloud Resource Allocation

An important factor that may limit the performance of applications on Clouds is the long startup time of Cloud resources which may take up to several minutes. Since ROIA are highly responsive applications, finding a compensation of long startup times is an important task.

To start multiple resources as quickly as possible, the `CloudController` of RTF-RMS sends multiple requests in parallel to the Cloud API. Moreover, the `DistributionController` introduces a *resource buffer* to which a predefined number of Cloud resources are moved in advance, i.e. before they are demanded by the application. The `CloudController` tags resources that are currently running the application as *running* in contrast to *buffered* resources that are currently not used. If any resource changes its state from buffered to running, the `DistributionController` checks whether new Cloud resources must be started in order to keep a certain number of resources in the resource buffer. The number of buffered resources is configured in the application profile.

By leasing and buffering Cloud resources in advance, additional overhead is generated. Therefore, the size s of the resource buffer should be chosen carefully. Given n clients and m servers for a particular zone, the time $t_{\text{integr}}(r, n, m)$ for integrating a single resource r from the resource buffer R is split up into time $t_{\text{intro}}(r, m)$ for the introduction of r to m other servers and time $t_{\text{migr}}(r, n)$ for migrating users to the new server, i.e., $t_{\text{integr}}(r, n, m) = t_{\text{intro}}(r, m) + t_{\text{migr}}(r, n)$. The overall time required for integrating all resources $r_1, ..., r_s$ from the resource buffer R in the application processing should be less than the time $t_{\text{leas}}(p)$ required for leasing a new single Cloud resource from the infrastructure provider p. Therefore, the following condition describes a reasonable limit for the size s of the resource buffer: $1 \leq s \leq \max\{i \in \mathbb{N} \mid \sum_{j=1}^{i} t_{\text{integr}}(r_j, n, m) < t_{\text{leas}}(p)\}$.

Another important issue for the cost-efficient ROIA provision on Clouds is the consideration of leasing periods. In commercial Cloud systems, resources are typically leased and paid per hour or some longer leasing period. Since ROIA have dynamically changing user numbers, the time after which Cloud resources

become dispensable is very variable. However, resources will not be used cost-efficiently if they are shut down before the end of their leasing period. Hence, RTF-RMS removes the resources that have become dispensable from the application processing and moves them to the resource buffer. Cloud resources in the buffer are shut down at the end of their leasing period or they are integrated in the application processing again if required.

4 Experiments

While the scalability of the proposed load-balancing actions was already demonstrated in [7], we evaluate the suitability of the proposed load-balancing actions to overcome performance bottlenecks in an example multi-player action game. This game, RTFDemo, is developed using RTF and is a representative of the first-person shooter online game category. In RTFDemo, zoning and replication are used for the distribution of the game state. We use the tick duration as the monitoring value for up- and down-scaling of the RTFDemo session. In order to provide a seamless gaming experience, users should not receive less than 25 updates per second over a longer time period. Hence, we configured RTF-RMS by specifying the upper threshold of 40 ms for the tick duration.

In our experiments, we use a private Cloud environment with the Eucalyptus framework (version 2.0.2) [12]; servers are Intel Core Duo PCs with 2.66 GHz and 2 GB of RAM running CentOS 5.3. Since Eucalyptus and Amazon EC2 use the same API, no adaptation of the `CloudController` was required.

For our first experiment, we have started two servers, with all clients initially connected to server 1, which implies an imbalanced load on application servers. We apply load balancing by user migration without leasing new Cloud resources. Fig. 4 shows that the tick duration of server 1 initially grows with the number of connected clients. When the tick duration reaches the threshold of 40 ms, RTF-RMS migrates half of the users (i.e., 120 users) from server 1 to server 2. This action reduces the tick duration of server 1 from 40 ms to 10-15 ms.

Our second experiment demonstrates load balancing by replication enactment. We start a single RTFDemo server and connect multiple clients to it. Fig. 5 shows that as soon as the tick duration of server 1 exceeds the threshold, RTF-RMS requests a new Cloud resource from the resource buffer (for this experiment, the size of the buffer was 1). Although the requested resource is already started up (since it is in the buffer), we observe that a certain time period is still required to add the new server to the application processing and start user migration. This delay of approximately 10 sec is caused by the initialization and inter-server communication required to integrate the new server in the application processing. After the migration is accomplished, the tick duration of server 1 is significantly reduced. Note that if the resource were not in the buffer and would have been started up from scratch, the delay would be much longer, in the order of 130 sec.

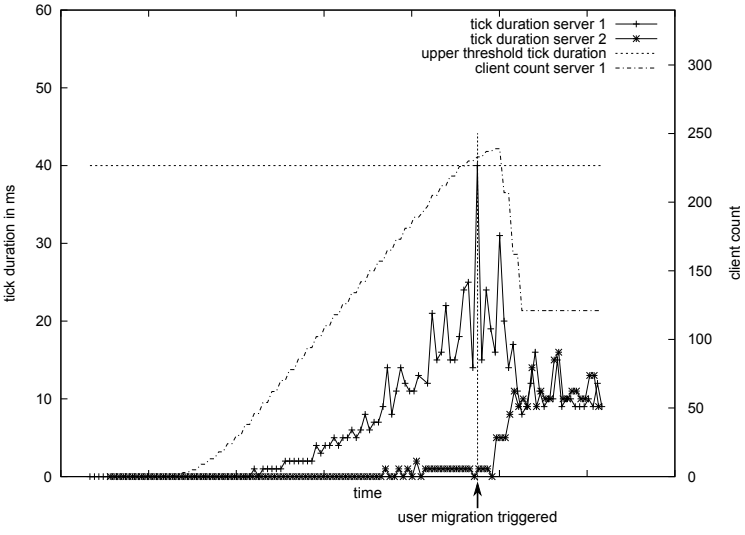

Fig. 4. Load balancing by user migration

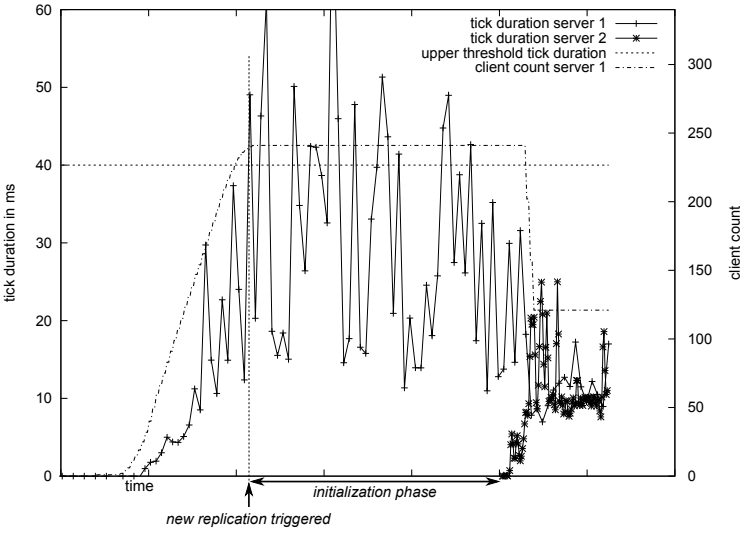

Fig. 5. Load balancing by replication enactment

5 Conclusion

This paper presents our new resource management system RTF-RMS for executing Real-Time Online Interactive Applications (ROIA) on Cloud systems. We have described our load-balancing strategy for ROIA on Clouds which chooses

between three possible load-balancing actions: user migration, replication enactment, and resource substitution. The strategy is based on the application-specific monitoring values provided by the Real-Time Framework (RTF), and goes beyond the state-of-the-art load balancing on Clouds which is based on generic system information. RTF-RMS allows for a cost-effective leasing of Cloud resources on demand by buffering unused resources; thereby the startup times of Cloud resources are reduced. Our experiments demonstrate the use of RTF-RMS load balancing by user migration and replication enactment for improving the performance of a multi-player, real-time online game on a Cloud.

Acknowledgment. Our research has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 under grant agreement 215483 (S-Cube).

References

1. Real-Time Framework, RTF (2011), `http://www.real-time-framework.com`
2. Freeman, T., Keahey, K.: Flying Low: Simple Leases with Workspace Pilot. In: Luque, E., Margalef, T., Benítez, D. (eds.) Euro-Par 2008. LNCS, vol. 5168, pp. 499–509. Springer, Heidelberg (2008)
3. Deelman, E., Singh, G., Livny, M., Berriman, B., Good, J.: The Cost of Doing Science on the Cloud: The Montage Example. In: Supercomputing Conference, pp. 1–12 (2008)
4. Buyya, R., Yeo, C.S., Venugopal, S.: Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities. In: Int. Conf. on High Performance Computing and Communications, pp. 5–13 (2008)
5. Dejun, J., Pierre, G., Chi, C.-H.: Resource provisioning of Web applications in heterogeneous clouds. In: Proceedings of the 2nd USENIX Conference on Web Application Development (2011)
6. Wei, Z., Pierre, G., Chi, C.-H.: CloudTPS: Scalable transactions for Web applications in the cloud. IEEE Transactions on Services Computing (2011) (to appear)
7. Ploss, A., Meiländer, D., Glinka, F., Gorlatch, S.: Towards the Scalability of Real-Time Online Interactive Applications on Multiple Servers and Clouds. Advances in Parallel Computing, vol. 20, pp. 267–287. IOS Press (2011)
8. Amazon Web Services, AWS (2011), `http://aws.amazon.com`
9. Nae, V., Iosup, A., Prodan, R., Fahringer, T.: The Impact of Virtualization on the Performance of Massively Multiplayer Online Games. In: 8th Annual Workshop on Network and Systems Support for Games (NetGames), pp. 1–6 (2009)
10. Valente, L., Conci, A., Feijó, B.: Real Time Game Loop Models for Single-Player Computer Games. In: SBGames 2005 – IV Brazilian Symposium on Computer Games and Digital Entertertainment (2005)
11. Glinka, F., Ploss, A., Gorlatch, S., Müller-Iden, J.: High-Level Development of Multiserver Online Games. International Journal of Computer Games Technology 2008, 1–16 (2008)
12. Nurmi, D., Wolski, R., Grzegorczyk, C., et al.: The Eucalyptus Open-Source Cloud-Computing System. In: 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, pp. 124–131. IEEE Computer Society (2009)