Peer-to-Peer Netw. Appl. (2016) 9:1145–1161
DOI 10.1007/s12083-015-0383-6

Operation analysis of massively multiplayer online games on unreliable resources

Radu Prodan[1] · **Alexandru Iosup**[2]

Received: 30 November 2014 / Accepted: 14 June 2015 / Published online: 4 July 2015

Abstract We investigate the use of a new Massively Multiplayer Online Gaming (MMOG) ecosystem consisting of end-users, game providers, game operators, and Cloud resource providers, for autonomous, self-adaptive hosting and operation of MMOGs on unreliable resources. For this purpose, we developed an MMOG simulator compliant with our ecosystem, supported by a dynamic resource provisioning and load balancing algorithm. Using our simulator in which we inject traces collected from a real-world MMOG and resource characteristics from 16 Cloud providers, we study the impact on the involved actors by considering different resource availability levels. We highlight the advantages of dynamic resource allocation over the static overprovisioning with respect to two types of metrics: QoS offered to the clients and financial profit of game providers and operators.

Keywords Massively multiplayer online games · Fault tolerance · Dynamic resource provisioning · Load balancing · Cloud computing · Quality of service

✉ Radu Prodan
radu@dps.uibk.ac.at

Alexandru Iosup
A.Iosup@tudelft.nl

[1] Institute of Computer Science, University of Innsbruck, Innsbruck, Austria

[2] Parallel and Distributed Systems, Delft University of Technology, Delft, Netherlands

1 Introduction

Massively Multiplayer Online Games (MMOGs) are a new type of large-scale distributed applications characterised by a real-time virtual world entertaining millions of players spread across the globe. To concurrently support millions of active players and many other server-driven entities (non-playing characters and other game objects) generating variable computational and latency-sensitive resource demands, the MMOG operators purchase and overprovision a multi-server infrastructure with sufficient computational, network and storage capabilities for guaranteeing the Quality of Service (QoS) requirements and a smooth gameplay at all times. This statically provisioned infrastructure has two major drawbacks: it has high operational costs and is vulnerable to capacity shortages in case of unexpected increases in demand. For example, the infrastructure of the World of Warcraft MMOG has over 10,000 computers [6]. However, similar to fashion goods, the demand of a MMOG is highly dynamic and thus, even for the large MMOG operators that manage several titles in parallel, a large portion of the resources are unnecessary which leads to a very inefficient and low resource utilisation. RuneScape's (http://www.runescape.com/) infrastructure also comprises thousands of computers in hundreds of physical locations, and resource ownership can take up to 40 % of the total game revenue (see http://www.dfcint.com/).

To address these gaps, we investigated in previous work [9, 19, 20] the use of Cloud technologies for on-demand provisioning of virtualised resources to MMOGs based on their actual variable load. For this purpose, we designed a new ecosystem for MMOG operation and provisioning [20] consisting of four actors with smaller and better focused roles that facilitate their market penetration and chances of business success: clients, game providers,

game operators, and resource (Cloud) providers. The interaction between them is negotiated and regulated through bipartite Service Level Agreements (SLA), representing wrappers around QoS parameters they agree to deliver. In this paper, we extend our framework and ecosystem for dealing with autonomous, self-adaptive MMOG operation, with a focus on self-healing in case of unexpected resource failures. We present an analysis of the impact of employing real Cloud resources on the QoS offered to the clients based on a dynamic resource provisioning and load balancing algorithm. We model the resources offered by 16 Cloud providers for which we study the resulting quality of game-play considering different resource availability levels, and show that our system can automatically mitigate the negative effect resource failures have on MMOGs. For this purpose, we developed an MMOG simulator compliant with our ecosystem capable of emulating the behaviour of the four actors using traces collected from a real-life MMOG and resource characteristics from real-world commercial Cloud providers. Using our simulator, we study the impact of MMOG operation and provisioning on the involved actors by considering different resource availability levels, and highlight the advantages of dynamic resource allocation over the static overprovisioning with respect to two metrics types: QoS offered to the clients and financial profit of the game provider and operator.

The paper is organised as follows. Section 2 presents our client-server MMOG model followed by our new ecosystem in Section 3. Section 4 introduces the failures that can occur during the MMOG operation considered in our model, followed by the dynamic resource allocation approach and load balancing algorithm for dealing with them in Section 5. Section 6 presents our MMOG simulation tool used for evaluation in Section 7 using traces collected from a real-life MMOG on commercial Cloud resources. Section 8 presents our simulation results, Section 9 reviews the related work and Section 10 concludes the paper.

2 MMOG model

Today's online games operate as client-server architectures in which the server simulates the game world via computing and database operations, receives and processes commands from the clients, and interoperates with a billing and accounting system (depending on the game type) [1, 32]. The players dynamically connect through the *game client* program to a joint game session and interact with each other by sending play actions to the *game servers*. The vast majority of game servers follow a similar computational model implementing an infinite loop, each loop iteration consisting of three main steps:

1. processing events coming from the connected clients (e.g. avatar movements);
2. computing the new state of the entities as a result of the clients' commands and avatar interactions (e.g. trading, collection of items, battles);
3. broadcasting updated entity states to the clients.

The main characteristic of online games is the fact that they are soft real-time applications, having to deliver timely responses to the clients in order to create the needed immersive game-play experience. The resource load generated by a game server is dependent on the number of connected clients and, more significantly, on the number of interactions between their respective avatars and between their avatars and other game entities. A high number of connected clients and interactions can determine an overload of the state computation step of the main game server loop (step 2 of the server loop) resulting in a degradation of the game-play experience for the clients which makes the game unplayable and unappealing to players who eventually quit.

MMOGs emerged as a natural evolution of classic online games, responding to the trend of continuously increasing number of players within a joint game session. To accommodate the tens of thousands of concurrent players into one single MMOG session, the current practice is to parallelise the game server code and distribute the load across multiple resources employing parallelisation techniques. Currently there exist three such techniques: (1) *zoning*, which geographically partitions the game world into disjoint zones, that can be assigned to different computing resources; (2) *instancing*, which consists of creating multiple autonomous copies of the game world (or zone) and assigning them to different machines, and (3) *replication*, which enables load distribution by maintaining copies of the same game world (or zone) on multiple resources but distributes the clients between the resources. These techniques also allow transparent load balancing actions, such as seamless migration of clients between servers and transparent inclusion/exclusion of servers into/from a game session, enabling fine grained load distribution control with virtually no impact on the QoS. Software libraries such as the Real Time Framework [7] implement these techniques for distributed game operation under QoS constraints and with low overhead, for both the client and the server. Our system relies on such libraries, as the low-level software layer.

In this work, we consider running MMOG sessions on heterogeneous resources offered by multiple Cloud providers distributed around the world, as depicted in Fig. 1. Resources are virtual machines interconnected through local networks as well as through Internet. The resource representation comprises parameters for all the relevant characteristics for MMOG operation, namely computational power,

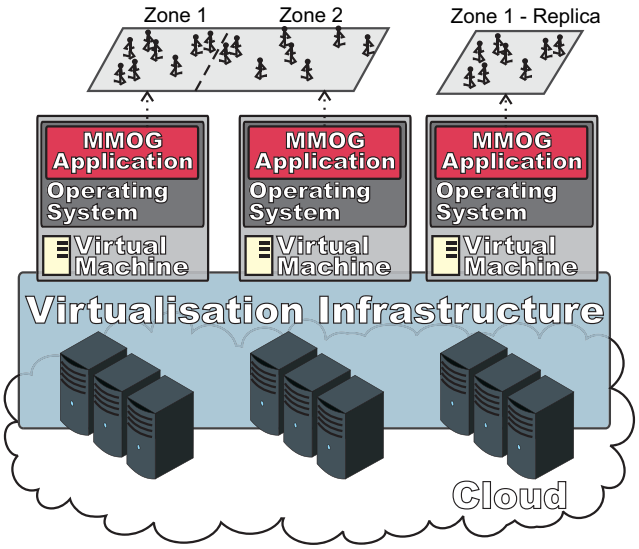


Fig. 1 Players connected to an MMOG session running on distributed virtualised Cloud resources

amount of installed memory, internal network bandwidth and Internet connection bandwidth.

3 MMOG ecosystem

We propose in our work a new ecosystem consisting of four actors, each fulfilling distinct roles (see Fig. 2):

1. the *client* who connects to the game operator's MMOG sessions;
2. the *game provider* who offer a selection of MMOGs by contracting new games from development companies (we do not cover this offline interaction);
3. the *game operator* who provisions resources from the resource provider and ensures the autonomous execution of the MMOG sessions;
4. the *resource provider* who offers the physical or virtualised Cloud machines on which the game servers run.

Resource providers are scattered around the world and aggregate Cloud resources that may serve multiple game operators simultaneously. Similarly, there are several geographically distributed game operators offering MMOG titles to clients and ensuring proper game operation by allocating the correct amount of resources from the providers.

3.1 Clients

Clients can join MMOG sessions offered by game operators on the basis of MMOG subscriptions. An MMOG subscription represents a contract between a client and a game operator based on which the client is allowed, under certain terms and with certain QoS guaranties, to join a MMOG session managed by the game operator.

3.2 Game providers

Game providers offer a selection of MMOGs by contracting new games from development companies. Based on clients' requests, game providers assign clients to game zones delegated to game operators for QoS-based execution. The quality of gameplay is monitored and in case of *SLA faults* (e.g. state update rate below minimum threshold), the client is compensated.

3.3 Game operators

The game operators interact with the clients and offer them a selection of MMOGs, usually by contracting new games from game development companies (interaction not modelled in this work). The operators autonomously execute distributed MMOG sessions with guaranteed QoS comprising interconnected MMOG servers. The game operator runs four main services.

3.3.1 Capacity management service

This service is responsible for the autonomous MMOG steering. Using a game state prediction mechanism, the capacity management service estimates the resource requirements for short-medium time intervals (order of minutes). Based on these load estimates, it instructs the resource management service to provision the correct amount of resources in the next allocation cycles. It also decides if the MMOG server start, stop, and migrate actions are necessary to accommodate and balance the generated load. For example, by timely foreseeing critical hot-spots in the game world (i.e. excessively populated areas of the game world with a large number of interactions), one can dynamically provision additional MMOG servers on newly leased resources and take proactive load balancing actions that redistribute the game load before the existing game servers become overloaded. We investigated viable approaches to capacity requirement estimation in [27] and no longer consider it in this paper.

3.3.2 Fault tolerance service

This service's responsibility is to reduce, or ideally eliminate the negative effects of unforeseen failures. We designed this service to ensure a high level of tolerance to resource faults which can lead to low QoS or even MMOG session unavailability. Although the commercial Cloud providers promise relatively high levels of resource availability, they are nevertheless subject to a multitude of unexpected events

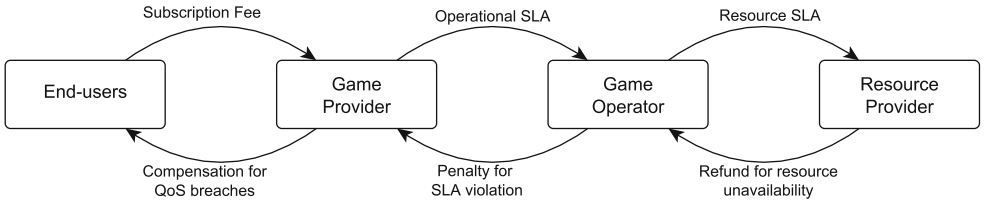


Fig. 2 Overview of MMOG ecosystem and SLA relationships

which can lead to failures. The fault tolerance service is responsible for maintaining the MMOG sessions' integrity in case of faults appearing in resources which host MMOG servers, such as independent or correlated machine failures, and local or Cloud site network connectivity loss. These events result in disruptions in game play which the fault tolerance service can minimise, or even completely hide to the clients by timely taking the appropriate counter measures such as redistributing the clients connected to a failing server to others within the same MMOG session (transparent measure), or starting other MMOG servers on new resources with the help of the resource management service (minimally disrupting measure).

3.3.3 QoS monitoring service

The QoS monitoring service collects and analyses information about the state of the MMOG sessions and the QoS delivered by the running game servers, such as like client update frequencies, utilised memory and network bandwidth, and average client-server connection latency. The QoS monitor analyses the collected information and aggregates it into monitoring reports which are then utilised by the other services. In particular, this service represents a feedback loop for the capacity management service, supplying the necessary sensor information to enable its autonomous and self-adaptive operation.

3.3.4 Resource management service

The resource management service interacts with the resource providers, negotiating for the resources that best fit the requirements provided by the capacity management service. The negotiation process takes into account multiple resource parameters such as computational power, amount of memory, network bandwidth, geographical location in relation to the requests, overheads introduced by the virtualisation technology, and price. It also provides the necessary low-level mechanisms for game session management such as MMOG server start, stop and migrate actions, as well as high-level paralellisation mechanisms such as zoning, replication, and instancing which distribute the client-generated load among computing resources (see Section 2). In previous work, we thoroughly analysed some of the challenges of the operator-provider interaction including the underlying resource negotiation process [22] and the effects of dynamic resource allocation on MMOG hosting [19]. As a consequence, we will not focus on these aspects of MMOG operation in this paper.

The game operator is also responsible for running a persistence service which ensures the continuity of the MMOG session throughout the lifetime of the game, but given that all MMOG operators already utilise advanced fault tolerance mechanisms for this particular service, we leave this aspect outside the scope of our investigation.

3.4 Resource providers

We consider Cloud providers employing the Infrastructure-as-a-Service (IaaS) paradigm and offering resources for fine grained time intervals (i.e. hours) though a virtualisation platform that allows automated software deployment and maintenance. The resource providers lease virtual machines with fuzzy definitions of their characteristics, but with much more precise guaranties in terms of resource availability. For example, Amazon (http://aws.amazon.com/ec2/) employs for the processor performance the "EC2 Compute Units" defined as "the equivalent CPU capacity of a 1.0-1.2 Gigahertz 2007 Opteron or Xeon processor", while FlexiScale (http://flexiscale.com/) uses the "vCPU unit" representing a computational unit of unknown power associated to 0.5 Gigabytes of memory. Conversely, the availability of an Amazon resource is defined as 99.95 % over a period of one year, while FlexiScale promises 100 % availability over one month along with an additional promise that, should a resource fail, the recovery time will be limited to 15 minutes.

Although security considerations are known to be critical in public clouds [33], we consider them as outside the scope of our work and defer them to the related game-specific middleware technologies [4].

4 Failure model

We identify in our MMOG operational model two principal types of failures that disturb the users' gameplay with a negative impact on the offered QoS, but from which the

system is capable of recovering with no human intervention: *resource failures* and *management failures*.

Resource failure In our scenario, the game operator runs the MMOG sessions on distributed heterogeneous resources provisioned from Cloud providers which are subject to unexpected events that can lead to failures. If a machine crashes, hangs or becomes unreachable through the network, the running MMOG server is compromised disrupting the normal operation of the session. In existing commercial deployments, such a severe unexpected event typically requires human intervention and can lead to hours of partial service unavailability, or even total unavailability in case of correlated failures. In our proposed architecture, the detection of this type of failure at the game operator level triggers a self-healing process consisting of two actions:

1. provisioning of a new resource or set of resources with the same (or better) characteristics as the failing one;
2. starting a new MMOG server part of the session and to which the clients are instructed to reconnect.

Thus, the MMOG session is salvaged but, regardless of these actions, the clients connected to the failing MMOG server will experience a *total interruption* in gameplay for a certain amount of time.

Management failure The game operator automatically adapts the amount of provisioned resources in a dynamic fashion to achieve a proper MMOG session operation with reduced allocation costs while reaching the targeted QoS. In case of erroneous load estimations or sudden increases in the number of accessing users, the provisioned resources are not sufficient to handle the generated load leading to the degradation of the QoS. The players connected to the overloaded servers will experience in this case a fragmented, unrealistic gameplay. In this situation, the game operator compensates either by redistributing the users to other MMOG servers within the same session aiming a better load distribution and a more efficient utilisation of the already provisioned resources, or by provisioning more resources for the affected session. We call this type of disturbance, where the users are not disconnected from the MMOG session but their gameplay experience is degraded, as *partial interruption*.

Evaluation metrics In previous work we covered several interesting QoS aspects of MMOGs, such as the performance impact of resources [19] and the Cloud virtualisation overheads [9]. In this work, we broaden our MMOG QoS investigation by studying the effects of the two identified types of disruptions on the QoS of MMOG sessions through three special metrics:

1. the *number* of interruptions in a certain time interval (e.g. the simulation period);
2. the *duration* of the interruptions, measured from the start of the event (resource/management failure) to the moment when all affected clients recover (i.e. when they are re-connected to the MMOG session in case of total interruptions, or when the QoS is above the promised level in case of partial interruptions);
3. the *severity* of the interruptions, representing the percentage of affected players of the MMOG session.

5 Dynamic resource allocation

Based on the predicted session and resource load in the next time interval (i.e. typically two minutes [19]), the game operator service arranges for the provisioning of the correct amount of resources required for a proper execution that guarantees a good experience to all players. A typical action is to extend game sessions with new zones or replication servers to accommodate an increased number of players during peak hours. Conversely, it may deallocate and merge multiple under-utilised game servers to improve the resource utilisation. The resources that need to be provisioned are of four types, as considered by the capacity planning service: CPU, memory, incoming network, and outgoing network bandwidth.

After allocating the required resources for a game session, the game operator instructs the game servers what parallelization strategy to apply and which entities to migrate to new servers (see Section 2). We designed an event-driven load balancing solution that receives *capacity events* from the capacity planning service, analyses them in the context of the global state of the game session and of the resources in question, and directs the resource allocation service in taking the appropriate measures. A capacity event consists of:

- the MMOG session and the resource this runs on;
- the event's status which can be of four types:
  - `SESSION_OVERLOADED` meaning that a server in the game session is becoming overloaded and requires a load balancing action, possibly by adding a new server to the game session;
  - `SESSION_UNDERUSED` meaning that a game server of a session carries too little load and can receive more load or even be withdrawn;
  - `HOST_OVERLOADED` meaning that a resource, such as a multi-core processor hosting multiple sessions, is becoming overloaded; typical actions to be performed are to load balance the sessions or to replicate servers;

  - HOST_UNDERUSED meaning that a resource carries too little load and can receive more through load balancing actions.

- a monitoring "snapshot" with a vector $[L_{CPU}, L_{mem}, L_{int\,net}, L_{ext\,net}]$ of four load metrics: processor, memory, incoming network, and outgoing network;
- a prediction "snapshot" that estimates the value of these metrics for the next (two minute) time interval [27].

We evaluate the aggregated load of resources as follows:

$$Load = \sum_{\forall l \in \overline{L} \times \overline{W}^T} l,$$

where $\overline{L}$ is the load vector:

$$\overline{L} = \left[L_{cpu} \;\; L_{mem} \;\; L_{int\,net} \;\; L_{ext\,net} \right]$$

and $\overline{W}$ represents the weight vector:

$$\overline{W} = \left[W_{cpu} \;\; W_{mem} \;\; W_{int\,net} \;\; W_{ext\,net} \right],$$

with:

$$W_{cpu} + W_{mem} + W_{int\,net} + W_{ext\,net} = 1.$$

The weight vector is defined either by the game developer, or determined in real-time while hosting the MMOG sessions and should be proportional to the resource demand of the game servers.

The load balancing algorithm displayed in Algorithm 1 requires a complete capacity event as input, analyses its components, and eventually generates the game session load balancing actions to be executed. The game session overload event (lines $5-16$), indicating that the session does not have enough resources, is addressed differently depending on whether the session is replicated or not. If the session is replicated across multiple resources and the load is unevenly distributed, a load balancing action is triggered (lines $6-8$). Otherwise, if the load is correctly distributed and all game servers participating in the replication are currently, or the prediction estimates they will be, overloaded, new resources are allocated to the game session by creating a replication action (lines $10-11$). If the overloaded game session is not distributed, a new replication action is generated (line 15).

The second case handles the game session under-utilisation events indicating that these sessions are allocated to many resources (lines $17-26$). If the game session is distributed and all involved game servers are currently, or the prediction estimates they will be, under-utilised, the resources are deallocated by creating a de-replication action (lines $18-20$). Otherwise, if not all game servers are under-utilised, a new load balancing action is generated (lines $22-23$).

The next branch (lines $27-41$) handles a resource overload event. The algorithm checks whether the respective machine has previous similar events to ensure that this event is not the result of an isolated load spike only. Next, in case the resource in question is hosting game servers involved in replications, load balancing actions are created for all unbalanced replications (lines $30-35$). Otherwise, when the overloaded resource is not hosting any replicated session, a replication action is generated for the game session responsible for the most load (lines $36-40$).

Finally, when the received event signals an under-utilised resource (lines $42-50$), the algorithm generates load balancing actions for all game session replications showing uneven load distributions, if any are locally hosted.

6 MMOG simulator

To ensure a flexible and complete validation of our MMOG research, we developed a trace-based MMOG simulation tool displayed in Fig. 3 that implements our MMOG architecture and failure model.

Coordinator is the central part in charge of initialising the simulation according to the input configuration and managing the whole execution of the simulation through the following tasks: (1) calculation of the game load and required resources; (2) checking of the current simulation state; (3) negotiation of new SLAs; (4) generation of resource requests and offers; (5) dynamic resource allocation; (6) creation output files for experimental evaluation.

Game traces are the central element of every MMOG simulation, consisting of a data file for every active game zone, and containing the number of currently active clients and the corresponding timestamps. Based on these data, we simulate the same amount of game zones distributed over the world together with their related number of players using a configurable simulation time step. The total simulation duration can be reduced by truncating the trace files.

User information is extracted from the trace files by a special service of the simulator based on the connected user accounts, necessary for the calculation of the subscription fees. To create a correct geographic mapping of the captured game, the simulator also uses the location of each individual client.

Game providers are in charge of managing the MMOG load models, connections to the user accounts, and provisioning of offers from the game operator to the customers. Based on the number of user access requests and their geographical origin, the game providers estimate the demand for each geographical area and construct operational SLA templates negotiated with the game operators. The SLA

Algorithm 1 The load balancing algorithm

```
Input: Capacity event event
Output: Load balancing actions actions
1  index ← 0;
2  host ← event.getHost(); session ← event.getGameSession();
3  L ← event.getMonitoringSnapshot(); P ← event.getPredictionSnapshot();
4  switch event.type do
5      case SESSION_OVERLOADED:
6          if isDistributedGameSession(session) then
7              if isReplicationUnbalanced(session,L) then
8                  actions[i++] ← createLoadBalancingAction(session,L);
9              else
10                 if isReplicationOverloaded(session,L) or
                   isOverloadPredicted(session,P) then
11                     actions[i++] ← createReplicationAction(session);
12                 end
13             end
14         else
15             actions[i++] ← createReplicationAction(session);
16         end
17     case SESSION_UNDERUSED:
18         if isDistributedGameSession(session) then
19             if isReplicationUnderused(session,L) or isUnderusePredicted(session,P)
               then
20                 actions[i++] ← createDereplicationAction(session);
21             else
22                 if isReplicationUnbalanced(session,L) then
23                     actions[i++] ← createLoadBalancingAction(session,L);
24                 end
25             end
26         end
27     case HOST_OVERLOADED:
28         if hasPreviousOverloadEvents(host) then
29             replications ← getDistributedGameSessionsOn(host);
30             if length(replications)> 0 then
31                 for r:replications do
32                     if isReplicationUnbalanced(r,L) then
33                         actions[i++] ← createLoadBalancingAction(r,L);
34                     end
35                 end
36             else
37                 sessions ← getGameSessionsOn(host);
38                 session ← getGameSessionWithMostLoad(sessions,L);
39                 actions[i++] ← createReplicationAction(session);
40             end
41         end
42     case HOST_UNDERUSED:
43         replications ← getDistributedGameSessionsOn(host);
44         for r:replications do
45             if isReplicationUnbalanced(r,L) then
46                 actions[i++] ← createLoadBalancingAction(r,L);
47             end
48         end
49
50 endsw
```

negotiation strategy and other provisioning variables can be altered in special configuration files. Additionally, the game providers manage the simulated game sessions by mapping the input traces to game zones.

Game operators are in charge of predicting and balancing of the load of the game sessions and zones by changing the properties for the respective services and algorithms. An essential part of the setup is the configuration of the *SLA templates* that manage the business transactions between the game operators and the game providers with variables such as base price, duration, and number of serviced players. Based on historical data and the currently assigned users, the operator predicts the load for predetermined time intervals (shorter for dynamic MMOGs and longer otherwise) using the method presented in [19]. The prediction consists of the user distribution in the MMOG world and a load model for translating it into resource requirements. At each prediction interval, the operator evaluates the provisioned resources and makes adjustments by allocating or releasing resources and redistributing the load. Other configuration files allow controlling the penalties for QoS violations, resource unavailability, as well as sorting and ranking resource offers or the desired allocation time.

Data centre emulates a configurable network of Cloud providers with resources and services spread across the

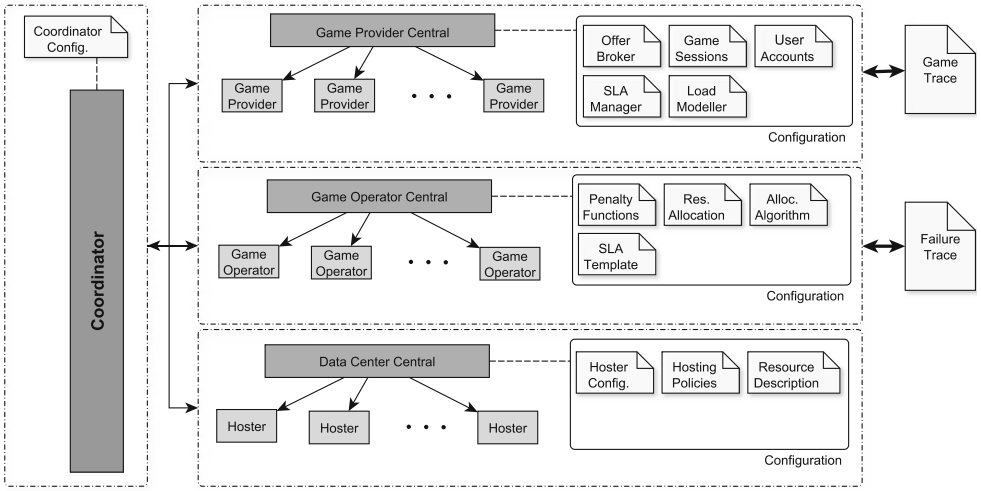


Fig. 3 Schematic MMOG simulation tool architecture

globe. To configure the resource providers, each setup configuration has a folder for every simulated data center containing information such as location, network bandwidth, and machines defined by their number of cores, processing speed and memory or disk size.

Failure traces are files containing the resource failures occurring during the simulated MMOG sessions. The format of the traces is inspired from the Failure Trace Archive, with a simplified format consisting of only three columns: the host name of the failing resource and the associated start and end (UNIX epoch) timestamps. Our resource availability model is based on three important parameters: (1) *inter-arrival time (IAT)* between two consecutive failures, (2) *duration* of each individual failure, and (3) *size* or number of the affected resources. Other parameters required for the failure generation are the number of available resources in the MMOG environment, the total duration of one simulation period, and the percentage of resource availability.

7 Simulation setup

For evaluating the impact of the resource and of the management failures, we use our MMOG simulator and six months of trace data collected from a real-world MMOG called RuneScape (http://www.runescape.com/).

We collected traces from 150 RuneScape servers with a sampling rate of two minutes, containing the number of players over time for each server group, aggregating approximately 40 million metric samples per simulation. Based on the number of concurrently active accounts from the RuneScape traces, we approximate the actual number of connected clients using the concurrent active account ratio metric approximated at around 15 % for RuneScape. We use a prediction interval and a simulation step of two minutes, which proves to be adequate for this type of games and the collected traces. Since we also study the financial impact of resource unavailability on the MMOG actors, we modelled the client accounts with different monthly, bimonthly, trimestrial and semestrial subscription types based on the real subscription and payment methods of Jagex Ltd. (the developer and publisher of RuneScape) as of August 2010 (see Table 1). We employ a setup with one game operator and one game provider since their number does not impact the QoS metrics for the clients targeted by our experiments.

The traces contain the maximum load of approximately 150 concurrent game zones spread all across the globe, where every zone needs one virtual machine (VM) instance to be properly executed. For this purpose, we model a distributed heterogeneous environment with 16 commercial Cloud providers aggregating 70 different VM types (see Table 2) and providing a large enough resource pool for the game operator of approximately 3500 concurrent VM instances for running all game zone instances. The parameters that model each VM type include the number of cores (1 to 12), CPU speed, RAM size (1 to 48 GB), and in- and outbound network bandwidth. While most of the parameters are clearly defined in the specifications of the commercial Cloud providers, the processing power of the VMs is not concretely quantified. Thus, we express it using an appropriate metric called *RS unit* representing the equivalent computational requirements of one RuneScape server servicing 2000 concurrent clients. We compute this metric based on existing benchmarks[1],[2] and our previous performance investigations [10]. We evaluated the quality of our simulator with respect to user load in [27], resource provisioning in [19], virtualization overheads in [9] and business

[1] http://blog.cloudharmony.com/2010/05/what-is-ecu-cpu-benchmarking-in-cloud.html

[2] http://www.chadkeck.com/2010/05/cloud-hosting-provider-hardware-benchmarks/

Table 1 RuneScape subscription plans for monthly, bimonthly, trimestrial, and semestrial payments

Plan	*Payment*	*Subscription price [USD]*			
		Monthly	*Bimonthly*	*Trimestrial*	*Semestrial*
SCC	Credit card	5.95	11.9	17.85	35.7
SPP	PayPal	7.5	12.99	18.15	36.1
SBT	Bank transfer	7.99	14.59	19.99	37.99

relationships in [21] which gives us high confidence on the accuracy of the obtained results.

Since one of our goals is to also show the impact of resource unavailability in a sparse resource environment, we derive six additional synthetic setups by gradually removing machines from the original VM instance pool to increase the resource utilisation, as illustrated in Table 3. In this table, the utilisation describes the percentage of machines to be allocated for a proper execution of all game zones at maximum load. In addition to the sparse resource environment, we introduce a controlled resource unavailability to our environment by associating to Cloud providers failure traces with tunable availability. We employ the resource availability model proposed in [8] and generate failure traces with average availabilities ranging from 99.5 % to 99.9 %. The traces are characterised through their failures' duration, size and IAT, each modelled through a statistical distribution. The distributions' parameters are presented in Table 4, along with the statistical properties of the resulting traces, where $Q1$, $Q2$ and $Q3$ represent the lower, the median, and the upper quartiles. For all traces, we employ the same distribution for the failure duration and size, but we vary the resource availability by adjusting the failure IAT. We generate both independent and correlated failures with a the ratio of 3 : 2.

Table 2 Summary of Cloud providers

Provider	*VM types*	*Location*	*Allocation time [hours]*
Amazon	6	4 (Asia, UK, USA (E, W))	1
CloudCentral	5	1 (AUS)	1
ElasticHosts	4	1 (UK)	1
FlexiScale	4	1 (UK)	1
GoGrid	4	1 (USA (E))	1
Linode	5	1 (USA (E))	24
NewServers	5	1 (USA (E))	1
OpSource	6	1 (USA (E))	1
RackSpace	4	2 (USA (E, C))	1
ReliaCloud	3	1 (USA (C))	1
SoftLayer	4	3 (USA (E, C, W))	1
SpeedyRails	3	1 (CAN W)	24
Storm	6	2 (USA (E, W))	1
Terremark	5	1 (USA E)	1
Voxel	4	3 (USA (E), NED, AUS)	1
Zerigo	2	1 (USA (C))	1

8 Simulation results

We evaluate the advantage of dynamic resource allocation over the static one with respect to two metrics: QoS and financial impacts. In case of static allocation, resource provisioning is performed once the zone is registered for operation, and the session does not get additional resources upon a failure but waits for the failure to end when the machine becomes operational again.

8.1 Impact of resource availability on MMOGs' QoS

In the first experiment, we investigate the impact of resource failures on the quality of game play under different resource availability conditions. We run one experiment for each average resource availability and evaluate the experienced QoS through the total interruptions metric.

Figure 4 depicts the number, the average duration and the severity of the total interruptions registered in the MMOG sessions over the six months simulation, as a function of resource availability. We observe stable, constant values for the severity and duration of total interruptions (the two bottom graphs) across all resource availability values, which indicates a proper recovery from resource

Table 3 Number of machines for a given utilisation

Utilisation [%]	*Number of machines*
5	2120
20	520
40	280
60	190
80	150
95	140

Table 4 Resource availability parameters and statistical characterisation

Failure metric	*Distribution*	*Scale*	*Shape*	*Statistical properties*						*Availability*
				Min	*Max*	*Average*	*Q1*	*Q2*	*Q3*	
Duration [min.]	Log-Normal	2.12	0.306	1	37	8	6	8	10	–
Size [machines]	Weibull	4	5	0	6	3	3	3	4	–
IAT [sec.]	Weibull	13600	7	2073	19668	12722	11381	12906	14254	99.5 %
IAT [sec.]	Weibull	7000	7	888	10123	6548	5860	6645	7336	99.6 %
IAT [sec.]	Weibull	4750	7	535	6988	4443	3976	4509	4977	99.7 %
IAT [sec.]	Weibull	3550	7	476	5146	3320	2971	3370	3720	99.8 %
IAT [sec.]	Weibull	2830	7	341	4119	2647	2369	2686	2965	99.9 %

failures. The median duration of a total interruption is two minutes and just below four minutes (i.e. approximately two resource allocation cycles) for more than 75 % of the events. The median percentage of affected players is below 2 %, as shown by the bottom interruption severity plot. The trend in the number of total interruptions (top graph) is inversely proportional to the average resource availability. This could be alleviated by a fault prediction method [19] that migrates MMOG servers away from failing resources.

Running MMOGs on real, limited availability cloud resources can potentially have a strongly negative impact the QoS for the clients due to prolonged recovery times and the need for human intervention for restoring the game session. However, we conclude based on this first experiment that this performance degradation can be mitigated by employing our proposed recovery techniques, which effectively limits the duration of the resulting interruptions to a constant value (lower than four minutes for 75 % of the events in our concrete scenario) independent of the duration of the underlying resource failure.

8.2 MMOG QoS in competition-based environments

The goal of the second experiment is to analyse the impact of resource failures on MMOGs in resource scarcity scenarios. We add this new dimension to our study by generating an increasing resource contention through gradually reducing the amount of resources in our setup. Thus, we run a set of simulations employing the same six month long RuneScape traces, but varying the amount of resources so that the peak load requires between 5 % and 95 % of the available resources. Concretely, for example, the setup in which the RuneScape peak load requires 60 % of the total amount of resources has a resource contention value of 60 %. As in the previous experiment, we shape the availability of the resources from 99.5 % to 100 % by employing the traces presented in Table 4. We cover these two dimensions with six values each, for a total of 36 simulations.

Figure 5 shows the number of interruptions experienced by the clients during the six-months simulation in the different resource availability and contention scenarios. We observe a slanting in the number of total interruptions

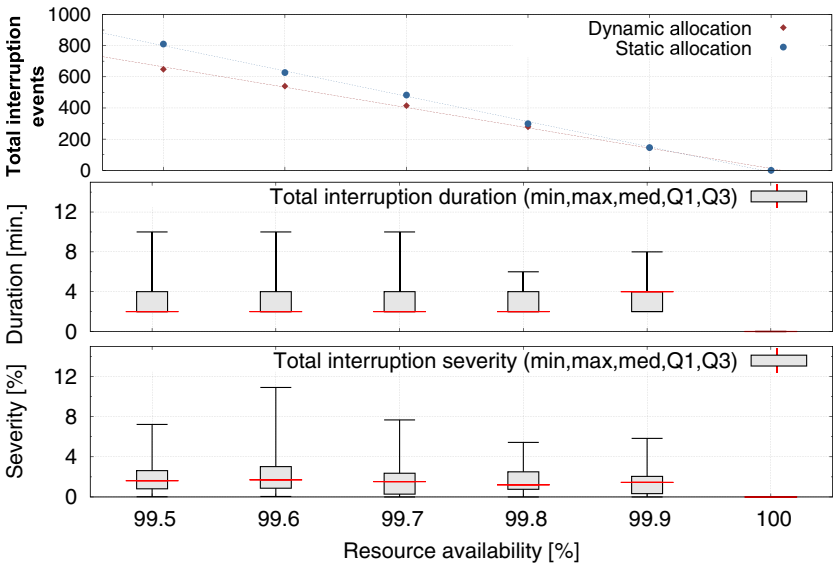
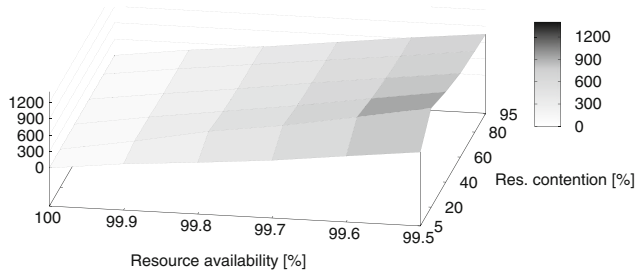
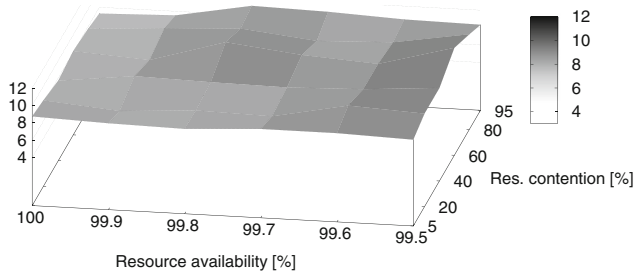


Fig. 4 Total interruptions under different resource availability conditions (all graphs show resource availability on horizontal axis with the bottom values)



(a) Number of total interruptions.



(b) Number of partial interruptions (thousands).

Fig. 5 Number of interruption events for varying resource availability and resource contention

(seen in Fig. 5a), which is consistent with the decrease in availability for all resource contention values, confirming the previous experiment's conclusions. The central positive finding of this investigation is the fact that the number of total interruptions remains constant with increasing resource contention, even in the limit case of 95 % resource contention. The only observed particularity is the lower number of total interruptions for the other limit case with an extreme resource abundance (5 % resource contention) over all availability values. The number of partial interruptions (Fig. 5b) appears to not be impacted either by the competition for resources, or by the resource availability.

For a complete assessment of the effectiveness of our MMOG operational model, the number of interruptions in isolation is not sufficient, as it does not concretely present the extent to which these failures impact the game session. We therefore present in Fig. 6a statistical analysis of all interruptions for two metrics: their duration and severity. Regarding the duration of the total interruptions (Fig. 6a, top), we notice a step-wise increase proportional to the resource contention, but a very stable behaviour with changing resource availability. Overall, the median time needed for recovery from a total resource failure is of two minutes (i.e. one resource allocation step) for a resource contention of up to 40 %, and four minutes (i.e. two allocation steps) for higher resource contention. The step-wise variation is due to the cyclic nature of the recovery evaluation. More precisely, an MMOG session might recover, for example, in 90 seconds, but the evaluation of the resource allocation state is only done every 120 seconds. Thus, the reported duration of recovery will be 120 seconds. In a real implementation, this issue would be easily circumvented by employing an event-driven monitoring system. Regarding the severity of the total interruptions (Fig. 6a, bottom), we notice a gradual increase proportional to the resource contention, from a median of approximately 0.7 % to 1.4 % correlated with an increase of resource contention from 5 % to 95 %. This variation is similar across different resource availability levels and, regardless of the individual configuration, at least 75 % for the events affect less than 2 % of the clients. In contrast to the total interruptions, the behaviour of the partial interruptions' duration and severity, shown in Fig. 6b, is clearly not dependent on either of the studied metrics (resource availability and contention). The recovery time is for the vast majority of events one allocation cycle long (i.e. two minutes), with only some outliers (less than 5 % of events) reaching 12 minutes. The severity of the partial interruptions is also very steady at 0.01 % of the number of clients for 95 % of the events, regardless of resource availability and contention. However, a general growing trend of the outliers coherent with the increase of resource contention can be observed.

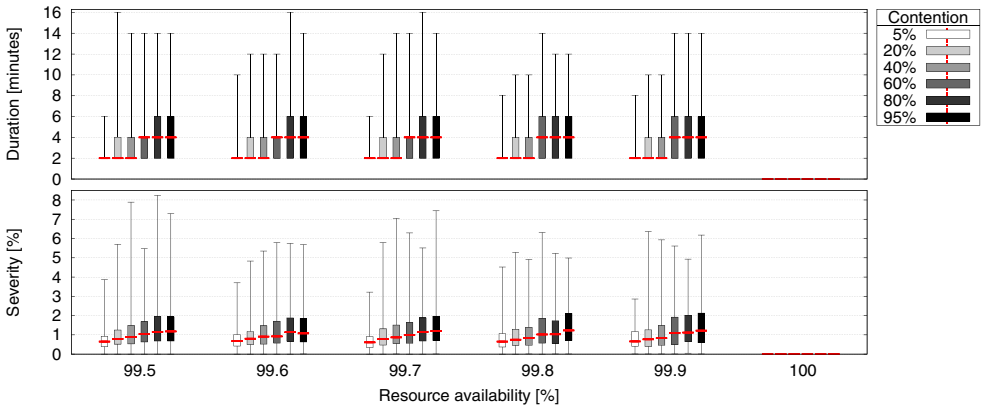
We conclude that clouds can be used for MMOG hosting with high QoS, even in cases of resource contention, as summarised in Table 5. Concretely:

1. Resource availability strongly impacts the number of total interruptions, but does not influence their duration and severity;
2. Contention for resources has a low negative impact on the number of total interruptions, but strongly affects their duration and severity;
3. Resource availability has little or no visible impact on partial interruptions, while an increased resource contention might lead to a slight increase in the severity of the partial interruptions.

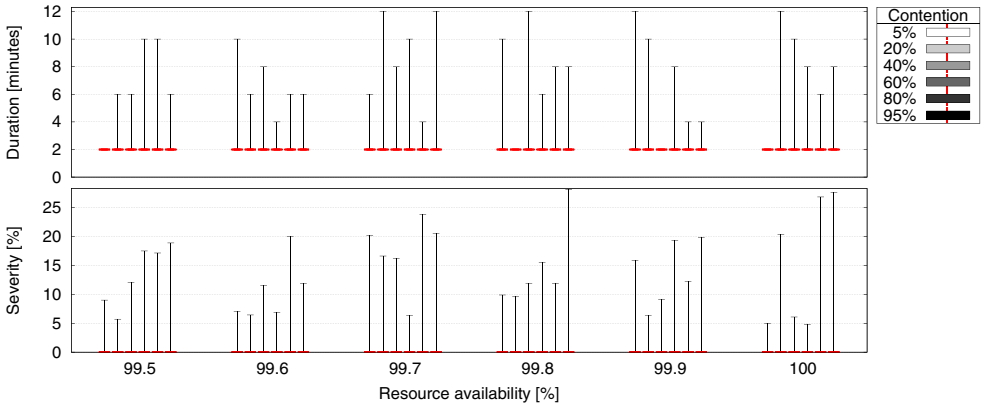
8.3 Status versus dynamic allocation

Comparing static with the dynamic allocation strategies, it it was already visible in Fig. 4 that the number of total interruptions is increasing with higher resource unavailability for static allocation, leading to an average of 30 % increase for a resource availability of 99.5 %. Despite the higher count of events and their prolonged duration, the median percentage of affected players (i.e. the severity) remains almost constant below 2 % across all availability values, and is only slightly higher in case of the static allocations (see Fig. 7).

The impact of the different allocation strategies can be best noticed in the analysis of the partial interruption events



(a) Total interruptions.



(b) Partial interruptions.

Fig. 6 Event duration and severity QoS analysis in competition-based conditions with increasing resource availability and resource contention

in Fig. 8. While the number of partial interruptions increases with the decreasing resource availability for static allocations, they remain relatively constant for the dynamic allocation strategy for all availabilities and utilisations. On average, the number of partial interruptions for an availability of 99.5 % is 4.5 times lower using dynamic allocations. With the dynamic allocation of resources, the partial interruption events have little to no impact on the QoS offered to the end-users (i.e. the severity) since our system can mitigates the effects by allocating additional resources or balancing the load. With static resource allocations, the partial interruption events affect on average about 1 % of the connected players.

8.4 Financial impact

Following the hierarchical architecture presented in Section 1, the first actors to analyse are the end-users who pay a subscription fee that allows them to enter gaming sessions provided by the game providers. Employing the static resource allocation approach not only leads to a degradation of the offered QoS, but also introduces a linear increase in the amount of QoS compensations paid to the users for lower availability and higher utilisation, as displayed in Fig. 9a. While the compensation payments lower the monthly costs for most of the clients, their overall play experience will suffer.

Table 5 Impact of resource availability and contention on MMOG QoS

Metric	*Impact on QoS*					
	Total interruptions			*Partial interruptions*		
	Number	*Duration*	*Severity*	*Number*	*Duration*	*Severity*
Resource availability	**Strong**	None	None	None	None	None
Resource contention	Light	**Strong**	**Strong**	None	None	Light

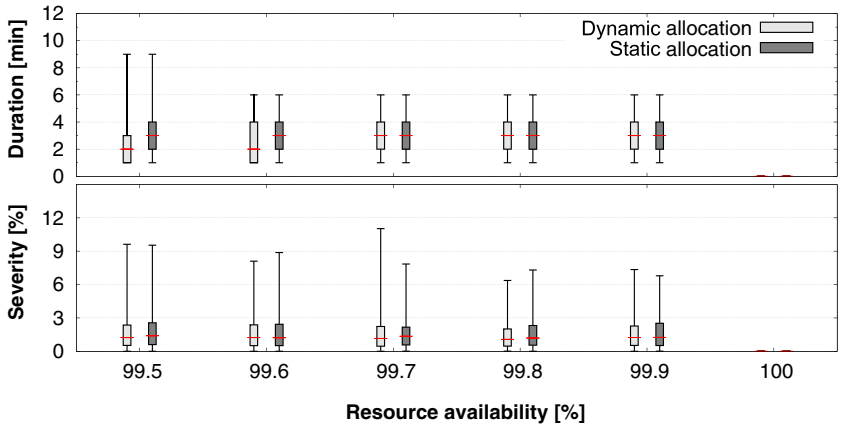


Fig. 7 Total interruption analysis for static and dynamic resource allocations for different resource availabilities

The impact of the static allocation on the QoS can also be noticed in the financial situation of the game provider who earns money from penalties paid by the game operator for SLA violations (part of which it transfers to the client as QoS compensations). Figure 9b shows the drop of the game providers' profit for using the static allocation approach in the different resource availabilities, averaged across all resource utilisation scenarios.

In the lowest tier, the game operator is directly affected by machine failures that occur in the data centres of the resource provider, where static allocation has a strong negative impact on the QoS. In case of a resource failure, the dynamic allocation mechanism provisions additional machines to ensure that no SLAs with the game provider are violated. Since it is not possible to allocate additional resources with the static allocation, almost every resource failure leads to a violation of an SLA, accompanied penalty fees. Figure 9c shows increasing compensation fees to be paid by the game operator to the provider with increasing unavailability and utilisation of resources.

The financial situation of the resource providers is generally not affected by the static allocation since the negative impacts of this approach are mostly restricted to the three higher tiers of the architecture.

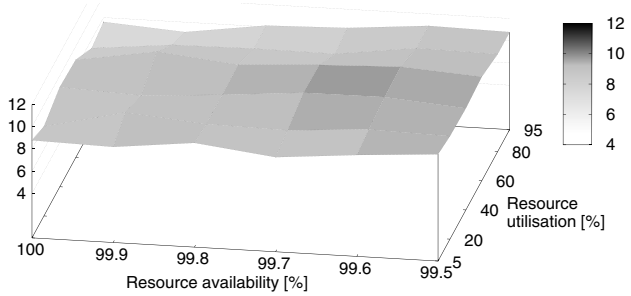
9 Related work

9.1 Dynamic resource provisioning

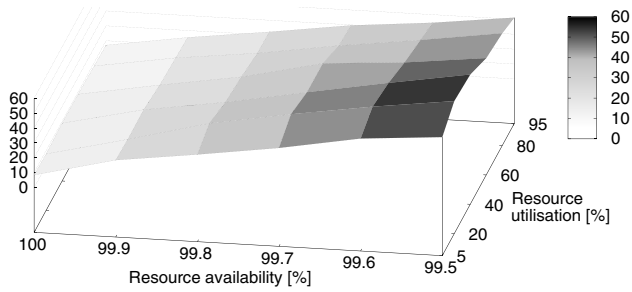
Much recent work focuses on (soft) QoS guarantees for MMOG operation [12, 16, 34]. Wong [34] proposes a resource provisioning algorithm with QoS guarantees, but considers only networking aspects, whereas we focus on maintaining QoS even during total resource failures. Complementary to our study, Lee and Chen [12] investigate MMOG server consolidation techniques, focusing on energy consumption. There have been a number of research activities in assessing the performance of virtualised resources in Cloud computing environments [25] and in general [28], some also considering the availability of Cloud resources [23]. In contrast to these studies, ours targets realistic Cloud resources with limited availability for a new application class (MMOG). Regarding the resource and MMOG deployment models, one study [29] comes close to our approach by proposing virtual machines for multi-player game operation. However, our work focuses on MMOGs which, in contrast to classic multi-player games, are distributed applications (multiple MMOG servers interconnected in a single session) serving a several orders of magnitude higher number of clients. Additionally, we also consider the virtualised resources as part of commercial Cloud computing platforms. Complementary to our work, [15] proposes a dynamic provisioning algorithm for a MMOG that maintains the response time below a given threshold based on a Queueing Network performance model used within a greedy algorithm. The work in [5] proposes an integrated approach which combines resource provisioning algorithms (such as static ones based on analytical queuing models, and dynamic on-demand ones) with scheduling disciplines (both QoS-aware and simple first come-first-serve ones) for revenue-cost optimisation of latency-sensitive applications, such as online games, in IaaS Clouds. The work is evaluated on a smaller-scale battle game called BZflag.

In the area of reliability, there are studies which investigate the characteristics of resource and workload failures, but do not assess their effects on the underlying systems' performance [24, 31]. Other efforts consider uncorrelated failures in distributed systems [2] and evaluate the resulting performance of the affected systems [8], but only for high-performance computing applications. In contrast, we employ the failure model introduced by [8], but apply it to Cloud resources and evaluate

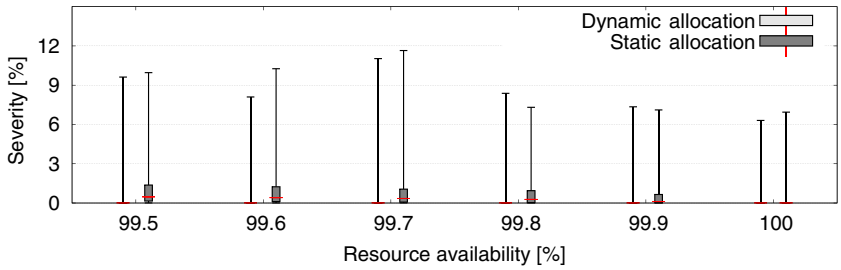
Fig. 8 Partial interruption analysis for dynamic and static resource allocations for different resource availabilities



(a) Number of partial interruptions (thousands) for dynamic resource allocation.



(b) Number of partial interruptions (thousands) for static resource allocation.

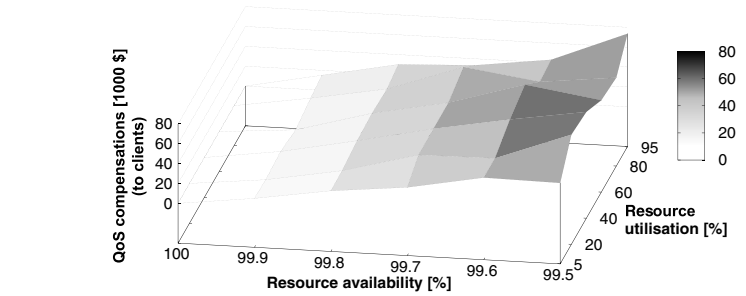


(c) Partial interruption severity.

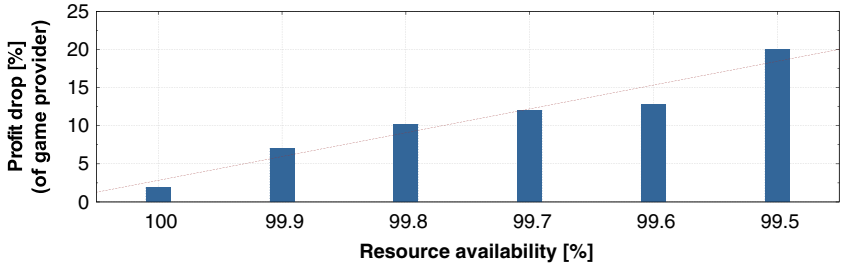
the consequences of utilising such resources on the QoS of MMOGs.

9.2 Load balancing

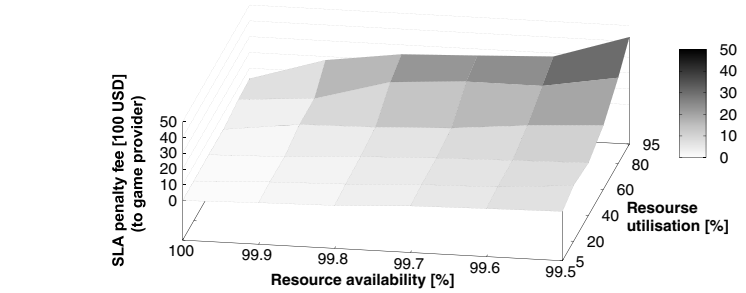
A few alternative load balancing techniques for game servers and for MMOG servers in particular exist with no regard to fault tolerance. Several provide dynamic control over their load distribution algorithms to a certain extent, others do not, but can be extended to support dynamic adjustments to the number of utilised servers. [13] presents a load distribution approach for MMOG servers which involves fine grained management of the game entities, each server being capable of managing an arbitrary subset of the clients. The load balancing is thus realised by adjusting the number of clients assigned to each server. However, the complexity of such fine grained client management makes this approach unsuited for fast-paced FPS games. Another load balancing method introduced in [18] relies on a dynamic adjustment of zone borders. In this approach the load is distributed by dividing the two-dimensional game world in vertical zones with mobile boundaries, the load balance being reached by moving these borders. Because of the resulting zone deformation, this method has the disadvantage that it does not scale to a high number of servers. Similar to our solution, the approach in [3] is based on a static partitioning of the game world and tries to obtain a sufficiently fine load granularity by using a high number of small zones. Additionally, it maps adjacent zones on topologically nearby nodes in order to reduce the communication overhead. However, this method assumes that the network load represents the bottleneck when provisioning games, while for the fast-paced games considered by our solution, the processor requirements are most critical. The Real-Time Framework [7] has been developed as a portable library to support the development of game servers using the zoning and replication parallelization techniques, used in [17] for load balancing of a multi-player online game in a



(a) Game provider compensations to clients.



(b) Game provider profit drop.



(c) SLA penalty fee from game operator to game providers.

Fig. 9 Financial impact on involved actors for static resource allocations

Cloud environment using a dynamic resource management system.

9.3 Peer-to-peer computing

Peer-to-peer computing has emerged as a scalable and low-cost technology, and as a potential alternative to traditional on-demand resource provisioning. When employing peer-to-peer technology, the game operators make use of the resources of their clients instead of renting them from hosters. The NPSNET project [14] uses a peer-to-peer approach in which all the game computation is performed on client resources. The SimMud [11] project uses a similar approach to NPSNET, but also balances and optimises the use of resources. However, three problems have prevented so far the adoption of peer-to-peer technology for MMOGs: the lack of appropriate business models, the wide-spread attempts of cheating, and the low availability of peers observed for other peer-to-peer systems (such as the Gnutella and the BitTorrent file sharing networks [26, 30].

10 Conclusions

We presented a simulator that implements a new ecosystem for operating MMOGs on Cloud infrastructures which effectively splits the traditional monolithic MMOG companies into three smaller and better focused actors whose interaction is regulated through bipartite SLAs: game providers, game operators, and resource providers. In our model, game operators dynamically provision resources from Cloud resource providers based on the MMOG load so that the QoS to the end-users is guaranteed at all times. Game providers lease operation SLAs from the game operators to satisfy all client requests and manage multiple distributed MMOG sessions. These three self-standing, smaller, more agile service providers enable access to the MMOG market for the small and medium enterprises, and to the current commercial Cloud providers. We evaluated in this paper using traces collected from a real-life MMOG the impact of resource availability and utilisation on the QoS and financial situation of the involved actors by comparing our novel dynamic resource allocation method with

the traditional static allocation strategy. We found out that our MMOG ecosystem successfully mitigates the performance degradation of running MMOGs on real commercial Cloud resources with limited availability to gameplay disruptions of less than four minutes, independently of the duration of the underlying resource failure. The majority of resource failures affect less than 2 % of the users participating in autonomously operated MMOG sessions. A low resource availability increases the number of gameplay disruptions, while a high resource contention results in longer disruptions affecting more clients. Finally, static resource allocation has a negative impact on the financial situation of the involved actors due to high compensation and penalty fees for QoS and SLA breaches upon low resource availability and high utilisation.

References

1. Bartle R (2003) Designing Virtual Worlds. New Riders
2. Bhagwan R, Savage S, Voelker G (2003) Understanding availability. In: Peer-to-Peer Systems II, LNCS, vol. 2735, pp 256–267. Springer
3. Chen J, Wu B, Delap M, Knutsson B, Lu H, Amza C (2005) Locality aware dynamic load management for massively multiplayer games. In: 10th SIGPLAN Symposium on Principles and Practice of Parallel Programming. ACM, pp 289–300
4. Diao Z, Schallehn E (2013) Towards cloud data management for mmorpgs. In: 3rd International Conference on Cloud Computing and Services Science. CLOSER 2013. Springer
5. Duong TNB, Li X, Goh RSM, Tang X, Cai W (2012) Qos-aware revenue-cost optimization for latency-sensitive services in iaas clouds. In: 16th International Symposium on Distributed Simulation and Real Time Applications. IEEE Computer Society, pp 11–18
6. Gamasutra: GDC Austin: An inside look at the universe of Warcraft., http://www.gamasutra.com/php-bin/news_index.php?story=25307
7. Glinka F, Ploss A, Müller-Iden J, Gorlatch S (2007) RTF: A real-time framework for developing scalable multiplayer online games. In: 6th ACM SIGCOMM Workshop on Network and System Support for Games. ACM, pp 81–86
8. Iosup A, Mathieu J, Sonmez O, Epema DHJ (2007) On the dynamic resource availability in grids. In: 8th IEEE/ACM International Conference on Grid Computing. IEEE Computer Society, pp 26–33
9. Iosup A, Nae V, Prodan R (2011) The impact of virtualization on the performance and operational costs of massively multiplayer online games. Int J Adv Media Commun 4(4):364–386
10. Iosup A, Ostermann S, Yigitbasi N, Prodan R, Fahringer T, Epema D (June 2011) Performance analysis of Cloud computing services for many-tasks scientific computing. IEEE Trans Parallel Distrib Syst 22(6):931–945
11. Knutsson B, Lu H, Xu W, Hopkins B (2004) Peer-to-peer support for massively multiplayer games. In: INFOCOM. IEEE, pp 96–107
12. Lee YT, Chen KT (2010) Is server consolidation beneficial to MMORPG? A case study of World of Warcraft. In: 3rd International Conference on Cloud Computing. IEEE Computer Society, pp 435–442
13. Lu F, Simon P, Graham M (2006) Load balancing for massively multiplayer online games. In: 5th ACM SIGCOMM Workshop on Network and System Support for Games. ACM, p 1
14. Macedonia MR, Brutzman DP, Zyda MJ, Pratt DR, Barham PT, Falby J, Locke J (1995) NSPNET: A multiplayer 3D virtual environment over the internet. In: Symposium on Interactive 3D Graphics, pp 93–94
15. Marzolla M, Ferretti S, D'Angelo G (2012) Dynamic resource provisioning for cloud-based gaming infrastructures. Comput Entertain 10(1):4:1–4:20
16. Briceño LD et al (2009) Robust resource allocation in a massive multiplayer online gaming environment. In: 4th International Conference on Foundations of Digital Games. ACM, pp 232–239
17. Meilnder D, Ploss A, Glinka F, Gorlatch S (2012) A dynamic resource management system for real-time online applications on clouds. In: Euro-Par 2011: Parallel Processing Workshops, Lecture Notes in Computer Science Volume, vol 7155. Springer, pp 149–158
18. Min D, Lee D, Park B, Choi E (1999) A load balancing algorithm for a distributed multimedia game server architecture. In: International Conference on Multimedia Computing and Systems, vol 882. IEEE Computer Society
19. Nae V, Iosup A, Prodan R (2011) Dynamic resource provisioning in massively multiplayer online games. IEEE Trans Parallel Distrib Syst 22(3):380–395
20. Nae V, Prodan R, Iosup A (2013) SLA-based operation of massively multiplayer online games in competition-based environments. In: Proceedings of the International C* Conference on Computer Science & Software Engineering. ACM, pp 104–112
21. Nae V, Prodan R, Iosup A (2014) SLA-based operations of massively multiplayer online games in Clouds. In: Multimedia Systems, vol 20, pp 521–544. https://www.springer.com/pay+per+view?SGWID=0-1740713-3131-0-0
22. Nae V, Prodan R, Iosup A, Fahringer T (2011) A new business model for Massively Multiplayer Online Games. In: Proceeding of the second joint WOSP/SIPEW international conference on Performance engineering. ACM, New York, USA, pp 271–282
23. Nagarajan AB, Mueller F, Engelmann C, Scott SL (2007) Proactive fault tolerance for hpc with xen virtualization. In: 21st Annual International Conference on Supercomputing. ACM, pp 23–32
24. Nurmi D, Brevik J, Wolski R (2005) Modeling machine availability in enterprise and wide-area distributed computing environments. In: Euro-Par 2005 – Parallel Processing, LNCS, vol 3648. Springer, pp 612–612
25. Palankar MR, Iamnitchi A, Ripeanu M, Garfinkel S (2008) Amazon S3 for science grids: a viable solution? In: International Workshop on Data-aware Distributed Computing. ACM, pp 55–64
26. Pouwelse J, Garbacki P, Epema D, Sips H (2005) The bittorrent p2p file-sharing system: Measurements and analysis. In: Castro M, Van Renesse R (eds) Peer-to-Peer Systems IV, Lecture Notes in Computer Science, vol 3640. Springer Berlin Heidelberg, pp 205–216. doi:10.1007/11558989_19
27. Prodan R, Nae V (2009) Prediction-based real-time resource provisioning for massively multiplayer online games. Futur Gener Comput Syst (FGCS) 25(7):785–793. http://www.sciencedirect.com/science/article/B6V06-4V0MJ6H-1/2/45b9cd3f8de8d176aebbeb9594231a1d
28. Quétier B, Neri V, Cappello F (2007) Scalability comparison of four host virtualization tools. J Grid Comput 5:83–98
29. Reed D, Pratt I, Menage P, Early S, Stratford N (1999) Xenoservers: Accountable execution of untrusted programs. In: Seventh Workshop on Hot Topics in Operating Systems, pp 136–141
30. Ripeanu M, Iamnitchi A, Foster IT (2002) Mapping the gnutella network. IEEE Internet Comput 6(1):50–57

31. Schroeder B, Gibson G (2010) A large-scale study of failures in high-performance computing systems. IEEE Trans Dependable Secure Comput 7(4):337–351
32. Shaikh A, Sahu S, Rosu MC, Shea M, Saha D (2006) On demand platform for online games. IBM Syst J 45(1):7–20
33. Winkler VJ (2011) Securing the Cloud. Cloud Computer Security Techniques and Tactics. Elsevier
34. Wong KW (2008) Resource allocation for massively multi-player online games using fuzzy linear assignment technique. In: Consumer Communications and Networking Conference. IEEE, pp 1035–1039

Radu Prodan is Associate Professor at the Institute of Computer Science, University of Innsbruck, Austria. He has over 15 years of research experience in the parallel and distributed computing areas. He earned his Ph.D. in 2004 from the Vienna University of Technology and his Habilitation in 2009 from the University of Innsbruck. Prodan coordinated and participated in numerous national and European projects and is currently scientific coordinator of the H2020 project ENTICE. He authored over 100 conference and journal publications (including one IEEE best paper award) and one book in the areas of parallel and distributed computing. The IEEE Transactions on Cloud Computing journal ranked him recently as the 11th most productive researcher worldwide in the area of Cloud computing.

Alexandru Iosup received his Ph.D. in Computer Science in 2009 from the Delft University of Technology (TU Delft), the Netherlands. He is currently an Assistant Professor with the Parallel and Distributed Systems Group at TU Delft. He is the cofounder of the Grid Workloads, the Peer-to-Peer Trace, and the Failure Trace Archives, which provide open access to workload and resource operation traces from large-scale distributed computing environments. He is the author of over 50 scientific publications and received several awards and distinctions, including best paper awards at IEEE CCGrid 2010, Euro-Par 2009, and IEEE P2P 2006.