Autonomous Massively Multiplayer Online Game Operation on Unreliable Resources[*]

Vlad Nae, Lukas Köpfle, Radu Prodan
Institute of Computer Science
University of Innsbruck
Technikerstr. 21a, A-6020 Innsbruck, Austria
radu@dps.uibk.ac.at

Alexandru Iosup
Dept. of Software and Computer Technology
Delft University of Technology
Mekelweg 4, 2628 CD, Delft, Netherlands
A.Iosup@tudelft.nl

ABSTRACT

Massively Multiplayer Online Games (MMOGs) are a new type of large-scale distributed applications characterised by seamless virtual worlds in which millions of world-wide players act and interact in real-time. Although for the past decade the number of MMOG players has grown exponentially, to the current tens of millions, this very growth may now hamper the progress of this important branch of the entertainment business. To guarantee Quality of Service (QoS) to a highly variable number of concurrent users, operators statically over-provision a large infrastructure capable of sustaining the game peak load, even though a large portion of the resources is unused most of the time. To address this problem, we propose a Cloud middleware-based system for autonomous operation of MMOGs. Our system provisions resources on-demand from multiple Cloud providers, automatically distributes the MMOG load between these resources, and self-heals when confronted with unforeseen resource failures. This new operational model allows small and medium enterprises to join the competitive MMOG market through near-zero initial infrastructure investment and operate MMOGs at given levels of QoS with small human intervention. We evaluate through simulations based on real-life MMOG traces the impact of resource availability on the QoS offered to the MMOG clients. We find that: (1) our proposed MMOG operation system can mitigate the negative effects of resource failures in under four minutes, (2) MMOG server consolidation in a resource-scarce environment can accentuate the negative effects of resource failures, and (3) the competition for resources can indirectly affect the QoS of the MMOG sessions.

Categories and Subject Descriptors

C.4 [**Computer Systems Organization**]: Performance of Systems—*Fault tolerance, Reliability, availability, and serviceability*; I.6.8 [**Simulation and Modeling**]: Types of Simulation—*Gaming*

General Terms

Management, Measurement, Reliability

Keywords

Massively Multiplayer Online Games (MMOG), reliability, Quality of Service (QoS), autonomic Cloud computing

1. INTRODUCTION

Online entertainment including gaming is a strongly growing sector worldwide. Massively Multiplayer Online Games (MMOG) grew from ten thousand subscribers in 1997 to eight million in 2005 and the rate is accelerating, estimated to 60 million people by 2015. The market size is estimated by the Entertainment Software Association (ESA) to 24 billion US Dollars (USD) with an avid growth over 120% in the last six years. In comparison, the Motion Picture Association of America (MPAA) reports a size of 10.6 billion USD with a 30% growth and the Recording Industry Association of America (RIAA) a size of 7 billion USD which has decreased by 30% in the last six years. The game industry is therefore among the fastest growing entertainment markets.

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 comply with the variable computational and latency-aware resource demands of MMOGs, the MMOG operators over-provision an own multi-server infrastructure with sufficient capabilities for guaranteeing the *Quality of Service (QoS)* requirements and a smooth game play 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 [5]. 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/`).

In contrast to static provisioning, the new Cloud computing technology based on resource virtualisation has the po-

[*] Austrian Science Fund project TRP 72-N23 funded this research.

C^3S^2E-13 2013, July 10-12, Porto [Portugal]
Copyright 2013 ACM 978-1-4503-1976-8/12/06 ...$15.00.

tential to provide an on-demand infrastructure for MMOGs, where resources are provisioned and paid for only when they are actually needed. The virtualisation technology also can alleviate the problem of porting and deploying MMOGs on on-demand resources by providing homogeneous resources for on-demand MMOG hosting. Conversely, this technology can introduce virtualisation overheads which may cancel the benefits. In previous work, we studied the consequence of using virtualised resources with respect to the incurred QoS overheads [9] and the economic benefits [12], while considering ideal resources in terms of availability.

In this work, we propose a Cloud-based middleware for autonomous, self-adaptive MMOG operation, complementary to the one introduced in [11], 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. 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.

The paper is organised as follows. The next section presents the general MMOG model underneath our approach, followed by the proposed multi-tier Cloud-based middleware architecture in Section 3. Section 4 presents the failures considered by our MMOG operational model, evaluated in Section 5 using simulation traces collected from a real-life MMOG on commercial Cloud resources. Section 6 reviews the related work and Section 6 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, 20]. 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.

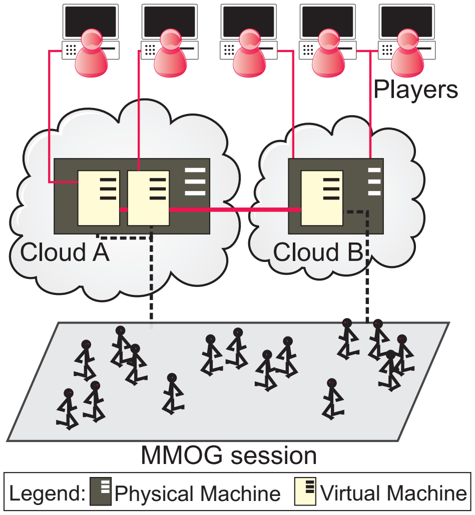


Figure 1: Players connected to an MMOG session running on distributed virtualised Cloud resources.

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 [6] 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 Figure 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, amount of installed memory, internal network bandwidth and Internet connection bandwidth.

3. ARCHITECTURE

We propose a three tiered architecture composed of game session and resource management services supporting and steering the execution of MMOGs. Our MMOG platform

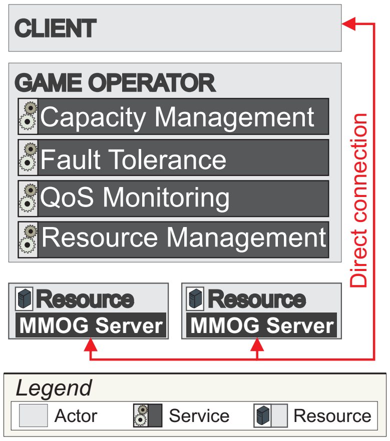


Figure 2: MMOG ecosystem architecture.

depicted in Figure 2 consists of three actors, each fulfilling distinct roles:

1. the *client* who connects to the game operator's MMOG sessions;
2. the *game operator* who provisions resources from the resource provider and ensures the autonomous execution of the MMOG sessions;
3. 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 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.

3.2.1 *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 [16] and no longer consider it in this paper.

3.2.2 *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 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).

3.2.3 *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.

3.2.4 *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 [12] and the effects of dynamic resource allocation on MMOG hosting [11]. 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.

3.3 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 [21], 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 the proposed MMOG operational model two principal types of failures that result in two types of disturbances in the clients' game play 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*.

4.1 Resource failure

The game operator runs the MMOG sessions on distributed heterogeneous resources provisioned from Cloud providers which are subject to a multitude of 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 distributed MMOG 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 game-play for a certain amount of time. Although the added value of such an autonomous system is inherently clear, we present in Section 5 an evaluation of this self-healing process and its low impact on QoS.

4.2 Management failure

The proposed middleware system automatically adapts the amount of provisioned resources to achieve a proper MMOG session operation (i.e. reaching the targeted QoS) through the game operator's capacity management service (see Section 3.2). In case of erroneous estimations or sudden surges in the number of clients, the provisioned resources are not sufficient to handle the generated load, which leads to the degradation of the QoS (i.e. fragmented, unrealistic game-play) for the clients connected to the overloaded servers. In this situation, the game operator compensates by either redistributing the clients to other MMOG servers within the same session aiming a better load distribution and a more efficient use of the already provisioned resources, or by provisioning more resources for the affected session. We call this type of disturbance, where the clients are not disconnected from the MMOG session but their game-play experience is degraded, as *partial interruption*.

4.3 Evaluation metrics

In previous work we covered several interesting QoS aspects of MMOGs, such as the performance impact of resources [11] 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. EXPERIMENTS

In this section we evaluate certain QoS aspects of our proposed self-adaptive, self-healing MMOG operational system in real-world scenarios involving realistic, failing resources. We perform experiments using traces collected from RuneScape (`http://www.runescape.com/`), a real MMOG ranked second after World of Warcraft by number of active paying customers in the US and Europe. The input workload consists of six months worth of monitoring data collected from 150 RuneScape servers, sampled every two minutes and consisting of the number of players over time for each server group. Overall, the data sums up to approximately 40 million metric samples per simulation, ensuring statistical soundness.

Table 1: Summary of the modelled Cloud providers.

Cloud provider	*VM types*	*Data centres (locations)*	*Allocation time [hours]*
Amazon	6	4 (Asia, U.K., U.S. East, U.S. West)	1
CloudCentral	5	1 (Australia)	1
ElasticHosts	4	1 (U.K.)	1
FlexiScale	4	1 (U.K.)	1
GoGrid	4	1 (U.S. East)	1
Linode	5	1 (U.S. East)	24
NewServers	5	1 (U.S. East)	1
OpSource	6	1 (U.S. East)	1
RackSpace	4	2 (U.S. East, U.S. Centre)	1
ReliaCloud	3	1 (U.S. Centre)	1
SoftLayer	4	3 (U.S. East, U.S. Centre, U.S. West)	1
SpeedyRails	3	1 (Canada West)	24
Storm	6	2 (U.S. East, U.S. West)	1
Terremark	5	1 (U.S. East)	1
Voxel	4	3 (U.S. East, Netherlands, Australia)	1
Zerigo	2	1 (U.S. Centre)	1

Table 2: Resource availability parameters and statistical characterisation.

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

We model a distributed heterogeneous resource environment consisting of a set of 16 commercial Cloud providers aggregating 70 different virtual machine (VM) types summarised in Table 1, and enough physical resources to support approximately $3,500$ concurrent VM instances. Each of the modelled Cloud providers' data centres is associated with generated 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 inter-arrival time (IAT), each modelled through a statistical distribution. The distributions' parameters are presented in Table 2, 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. Both independent and correlated failures are generated, the ratio between the two being 3:2.

As defined in our architecture (see Section 3), the simulated environment comprises multiple game operators, each running their four management services. The resource provisioning has a two-minute cycle and is based on multiple parameters, detailed in Section 3.2. The evaluation targets the QoS the clients experience in the proposed environment where MMOGs are hosted on unreliable Cloud resources and is realised through the three metrics defined in Section 4, collected from the QoS monitoring services.

5.1 Resource availability impact on QoS

In this 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 QoS experienced by the clients through the total interruptions metric, defined in Section 4.

Figure 3 depicts the number, the average duration and the severity of the total interruptions registered within the MMOG sessions over the six months simulation period, as a function of resource availability. We observe stable, constant values for the severity and the duration of total interruptions (the two bottom graphs) across all resource availability values, which validates the proposed automated process for recovery from resource failures of our MMOG architecture. 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, while the median percentage of affected players is below 2%, as shown by the failure severity graph. The trend in the number of total interruptions (the top graph) is inversely proportional with the average resource availability, which could be further improved by employing a fault prediction method [11] that enables the resource management service to preemptively migrate the MMOG servers away from the failing resources.

We conclude that running MMOGs on real Cloud resources with limited availability 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, based on this first experiment

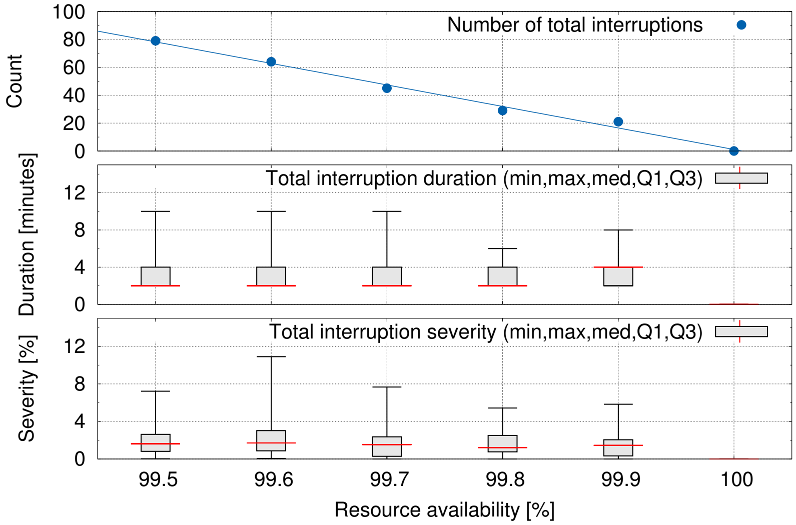


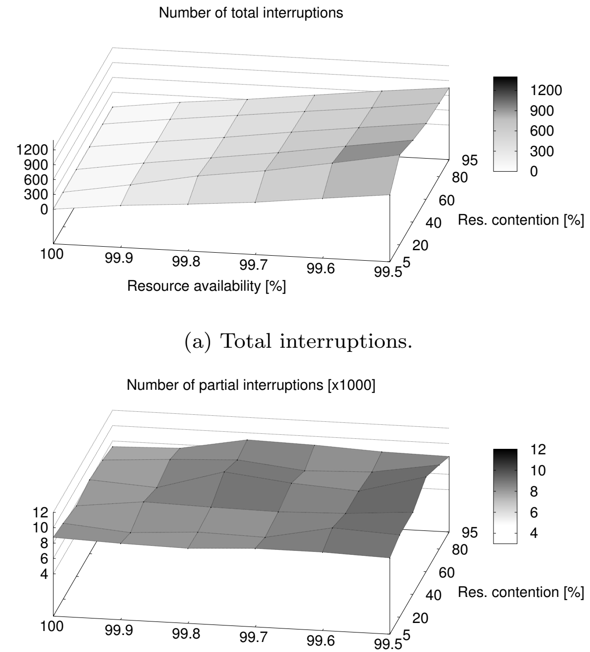
Figure 3: Total interruption analysis under different resource availability conditions (all graphs show resource availability on their horizontal axes, with the values from the bottom of the figure).

we conclude that this performance degradation can be mitigated by employing our proposed autonomous MMOG operation system, 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.

5.2 Impact of resource contention on QoS

The goal of this second investigation is to analyse the autonomous resource failure recovery of the proposed MMOG operational system in resource scarcity scenarios. We add this new dimension to our study by generating an increasing resource contention by 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. 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 vary resource availability from 99.5% to 100% by employing the traces presented in Table 2. We cover these two dimensions with six values each, for a total of 36 simulations.

Figure 4 shows the number of interruptions experienced by the clients during the six-months simulation in the different resource availability and contention scenarios. We observe in Figure 4a a slanting in the number of total interruptions, 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 extreme 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 (Figure 4b) appears to not be impacted either by the competition for resources, or by the resource availability.

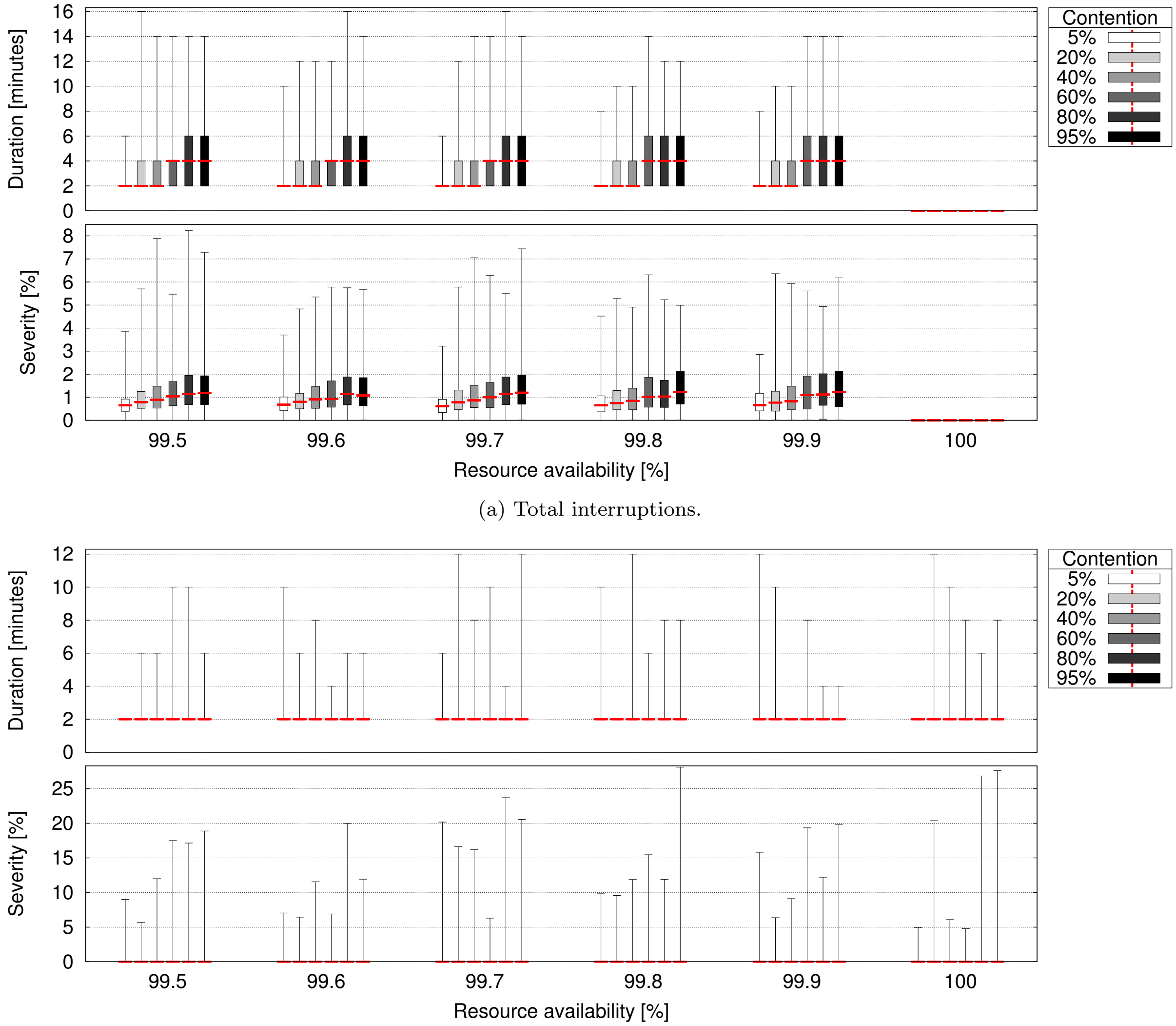


(a) Total interruptions.

(b) Partial interruptions.

Figure 4: Number of interruption events in varying resource availability and contention conditions.

The number of interruptions in isolation is not sufficient for a complete assessment of the effectiveness of our MMOG operational model, as it does not concretely present the extent to which these failures impact the game session. We therefore present in Figure 5 a statistical analysis of all interruptions for two metrics: their duration and severity. Regarding the duration of the total interruptions (Figure 5a, top), we notice a step-wise increase proportional to the resource contention, but a very stable behaviour with chang-



(a) Total interruptions.

(b) Partial interruptions.

Figure 5: Analysis of the interruption duration and severity with increasing resource availability (99.5% to 100%) and resource contention (5% to 95%).

ing resource availability. Overall, the median time needed for automatic 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 periodic nature of the recovery evaluation. For example, an MMOG session might recover 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 (see Figure 5a, 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 partial interruptions' duration and severity shown in Figure 5b are 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.

Based on these findings, we conclude that real Cloud resources can be used for autonomous MMOG hosting with high QoS, even in conditions of extreme resource contention, as summarised in Table 3. Concretely:

1. the resource availability strongly impacts the number of total interruptions, but does not influence their du-

Table 3: Summary of observed 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

ration and severity;

2. the contention for resources has a low negative impact on the number of total interruptions, but strongly affects their duration and severity;

3. the resource availability has no visible impact on partial interruptions, while an increased resource contention might lead to a slight increase in the severity of the partial interruptions.

6. RELATED WORK

Much recent work focuses on (soft) QoS guarantees for MMOG operation [10, 3, 22]. Wong [22] 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 [10] 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 [15] and in general [17], some also considering the availability of Cloud resources [13]. In contrast to these studies, ours targets realistic Dloud resources with limited availability for a new application class (MMOG).

Regarding the resource and MMOG deployment models, one study [18] 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.

The entertainment industry has already started to migrate from the in-house to Cloud-based infrastructure. Zynga operated in 2011 online gaming services for over 250 million users using Amazon EC2 resources up to several months after their launch. Nevertheless, the games supported by Zynga require much less computational and network resources than MMOGs. On-demand gaming, which offloads gaming computation to the Cloud and streams back to remote clients the video output of the game, is provided by companies such as Geelix [7], OnLive, Gaikai, and OTOY. We do not consider this game operational model because major MMOG operators have yet to switch to it, in part because of the high network requirements imposed on the players. Since late 2011, Amazon Web Services has been used for video-streaming by Netflix and for offloading web browsing for mobile devices with Android operating system. In contrast zo these approaches, our work adapts this model to the specifics of MMOGs and proposes a study of the impact of using Cloud resources on QoS.

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 [14, 19]. 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 the consequences of utilising such resources on the QoS of MMOGs.

7. CONCLUSION

We proposed a Cloud-based middleware system for autonomous MMOG operation with the end-goal of allowing small and medium enterprises to join the competitive MMOG market through nearly-zero initial investment. Our middleware architecture is based on the separation between the resource providers represented by today's IaaS Cloud providers, and game operators specialised in offering higher and more tunable QoS to the clients. We evaluated the viability of autonomous MMOG operation on real-world, failing Cloud resources analysing a set of QoS metrics in different simulation scenarios. We have also concretely evaluated the impact of resource availability and resource contention on the provided QoS. We found that:

1. our MMOG ecosystem successfully mitigates the performance degradation of running MMOGs on Cloud resources with limited availability to game play disruptions of less than four minutes, independently of the duration of the underlying resource failure;

2. the majority of resource failures affect less than 2% of the clients participating in autonomously operated MMOG sessions;

3. a low resource availability increases the number of game play disruptions, while a high resource contention results in longer disruptions affecting more clients.

8. REFERENCES

[1] R. Bartle. *Designing Virtual Worlds*. New Riders, 2003.

[2] R. Bhagwan, S. Savage, and G. Voelker. Understanding availability. In *Peer-to-Peer Systems II*, volume 2735 of *LNCS*, pages 256–267. Springer, 2003.

[3] L. D. Briceño, H. J. Siegel, A. A. Maciejewski, Y. Hong, B. Lock, M. N. Teli, F. Wedyan, C. Panaccione, C. Klumph, K. Willman, and C. Zhang. Robust resource allocation in a massive multiplayer online gaming environment. In *4th International Conference on Foundations of Digital Games*, FDG '09, pages 232–239. ACM, 2009.

[4] Z. Diao and E. Schallehn. Towards cloud data management for mmorpgs. In *3rd International Conference on Cloud Computing and Services Science*, CLOSER 2013. Springer, 2013.

[5] Gamasutra. GDC Austin: An inside look at the universe of Warcraft. http://www.gamasutra.com/php-bin/news_index.php?story=25307.

[6] F. Glinka, A. Ploss, J. Müller-Iden, and S. Gorlatch. RTF: A real-time framework for developing scalable multiplayer online games. In *6th ACM SIGCOMM Workshop on Network and System Support for Games*, NetGames '07, pages 81–86. ACM, 2007.

[7] O.-I. Holthe, O. Mogstad, and L. A. Ronningen. Geelix livegames: Remote playing of video games. In *6th IEEE Consumer Communications and Networking Conference*, CCNC, 2009. ISBN 978-1-4244-2308-8.

[8] A. Iosup, M. Jan, O. Sonmez, and D. H. J. Epema. On the dynamic resource availability in grids. In *8th IEEE/ACM International Conference on Grid Computing*, Grid 2007, pages 26–33. IEEE Computer Society, 2007.

[9] A. Iosup, V. Nae, and R. Prodan. The impact of virtualization on the performance and operational costs of massively multiplayer online games. *International Journal of Advanced Media and Communication*, 4(4):364–386, September 2011.

[10] Y.-T. Lee and K.-T. Chen. Is server consolidation beneficial to MMORPG? A case study of World of Warcraft. In *IEEE International Conference on Cloud Computing*, CLOUD, pages 435–442. IEEE Computer Society, 2010.

[11] V. Nae, A. Iosup, and R. Prodan. Dynamic resource provisioning in massively multiplayer online games. *IEEE Transactions on Parallel and Distributed Systems*, 22(3):380–395, 2011.

[12] V. Nae, R. Prodan, A. Iosup, and T. Fahringer. A new business model for massively multiplayer online games. In *Second Joint WOSP/SIPEW International Conference on Performance Engineering*, ICPE 2012, pages 271–282. ACM, March 2011.

[13] A. B. Nagarajan, F. Mueller, C. Engelmann, and S. L. Scott. Proactive fault tolerance for HPC with Xen virtualization. In *21st Annual International Conference on Supercomputing*, ICS '07, pages 23–32. ACM, 2007.

[14] D. Nurmi, J. Brevik, and R. Wolski. Modeling machine availability in enterprise and wide-area distributed computing environments. In *Euro-Par 2005 Parallel Processing*, volume 3648 of *LNCS*, pages 612–612. Springer, 2005.

[15] M. R. Palankar, A. Iamnitchi, M. Ripeanu, and S. Garfinkel. Amazon S3 for science grids: a viable solution? In *2008 International Workshop on Data-aware Distributed Computing*, DADC '08, pages 55–64. ACM, 2008.

[16] R. Prodan and V. Nae. Prediction-based real-time resource provisioning for massively multiplayer online games. *Future Generation Computer Systems*, 25(7):785–793, 2009.

[17] B. Qu'etier, V. Neri, and F. Cappello. Scalability comparison of four host virtualization tools. *Journal of Grid Computing*, 5:83–98, 2007.

[18] D. Reed, I. Pratt, P. Menage, S. Early, and N. Stratford. Xenoservers: accountable execution of untrusted programs. In *Seventh Workshop on Hot Topics in Operating Systems*, HOTOS '99, pages 136–141. IEEE Computer Society, 1999.

[19] B. Schroeder and G. A. Gibson. A large-scale study of failures in high-performance computing systems. *IEEE Transactions on Dependable and Secure Computing*, 7(4):337–351, 2010.

[20] A. Shaikh, S. Sahu, M.-C. Rosu, M. Shea, and D. Saha. On demand platform for online games. *IBM Systems Journal*, 45(1):7–20, 2006.

[21] V. J. Winkler. *Securing the Cloud. Cloud Computer Security Techniques and Tactics*. Elsevier, April 2011.

[22] K. W. Wong. Resource allocation for massively multiplayer online games using fuzzy linear assignment technique. In *5th IEEE Consumer Communications and Networking Conference*, CCNC 2008, pages 1035–1039. IEEE, 2008.