Efficient Management of Data Center Resources for Massively Multiplayer Online Games

Vlad Nae*, Alexandru Iosup†, Stefan Podlipnig*, Radu Prodan*, Dick Epema†, and Thomas Fahringer*
*Institute for Computer Science, University of Innsbruck,
Technikerstrasse 21a, A-6020 Innsbruck, Austria.
Email: {Vlad,Stefan.Podlipnig,Radu,TF}@dps.uibk.ac.at
†Parallel and Distributed Systems Group, Delft University of Technology,
Mekelweg 4, 2628CD Delft, The Netherlands.
Email: A.Iosup@gmail.com, D.H.J.Epema@tudelft.nl

***Abstract*—Today's Massively Multiplayer Online Games (MMOGs) can include millions of concurrent players spread across the world. To keep these highly-interactive virtual environments online, a MMOG operator may need to provision tens of thousands of computing resources from various data centers. Faced with large resource demand variability, and with misfit resource renting policies, the current industry practice is to maintain for each game tens of self-owned data centers. In this work we investigate the dynamic resource provisioning from external data centers for MMOG operation. We introduce a novel MMOG workload model that represents the dynamics of both the player population and the player interactions. We evaluate several algorithms, including a novel neural network predictor, for predicting the resource demand. Using trace-based simulation, we evaluate the impact of the data center policies on the resource provisioning efficiency; we show that dynamic provisioning can be much more efficient than its static alternative.**

I. Introduction

Massively Multiplayer Online Games (MMOGs) have emerged in the past decade as a new type of large-scale distributed application: real-time virtual world simulations entertaining players spread across the world. To maintain the level of responsiveness demanded by players, MMOG providers install and maintain for each game a dedicated multi-server infrastructure [1]. However, due to the dynamic character of MMOGs, both on the short and on the long term, the game operators have to over-provision their infrastructure, and new providers find it difficult to join the market. In this work we propose a dynamic resource provisioning method that addresses these two problems. Our solution uses a generic game model and in-game monitoring to dynamically predict the user load. Based on accurate load prediction, we obtain resources from data centers, and run game servers on them.

MMOGs have already started to attract the interest of the research community. Challenges such as scalability, trust, and data consistency have been identified by the distributed systems [2] and the database [3] communities. In this work we draw the attention to a new direction of research: resource provisioning for MMOGs.

Today's MMOG operate as client/server architectures, in which the game server simulates a world via computing and database operations, receives and processes commands from the clients, and inter-operates with a billing and accounting system [1], [4]. Failing to ensure timely simulation updates leads to a degraded game experience, and to players canceling their accounts. As a result, the game providers need to install and operate a large infrastructure, with hundreds to thousands of computers for each game (e.g., the operating infrastructure of the MMOG World of Warcraft has over 10,000 computers [5]). However, similar to fashion goods, the demand of a MMOG is highly dynamic. Thus, even for the large providers that operate several titles in parallel, a large portion of the resources are unnecessary. Furthermore, determining the resource requirements dynamically is non-trivial for MMOGs: unlike many other client-server applications [6]–[10], the interaction between the concurrent system users has an important role in the load of the system. In this work we investigate the operation of MMOGs on resources obtained dynamically from data centers spread across the world. Towards this end, we first present in Section II a model for the MMOG ecosystem that describes the application, the resource providers, and their interaction. In contrast to previous models of client/server applications over the Internet [6]–[10], which focus on mostly non-interacting user requests, our model is designed around the notion of player interactions. We show through the analysis of RuneScape, a large-scale MMOG, that MMOGs are more dynamic than previously believed [11] and that the interaction with the players is a major component of the MMOG workload (Section III). Thus, we show that the amount of unnecessary resources due to the static resource provisioning is much higher than previously believed.

Then, motivated by the dynamic MMOG workloads, we propose a dynamic resource provisioning model in which the amount of resources is dynamically predicted, and the necessary resources are obtained dynamically from data centers. To minimize the resource over-provisioning, we propose in Section IV a novel load prediction algorithm based on neural networks, which has reasonable speed and better accuracy than six other alternatives for a variety of realistic MMOG workloads. Based on this algorithm and on the MMOG application model, we are able to convert the load prediction into a resource demand estimation. In contrast to previous work, where resources are provided by a single data center [6],

[8], in this work we consider a new approach that focuses on the requirements of MMOGs: the resources are provided by many data centers, and the geographical location of these centers with respect to the location of the users generating the demand is taken into account. We further extend this model to consider the lease policies of the data centers: we favor centers that allow leasing fewer resources at a time, and for shorter periods of time. In Section V we present a thorough investigation of the dynamic resource provisioning for MMOGs. Using the traces analyzed in Section III, and the prediction algorithm proposed in Section IV, we show through simulations that the dynamic resource provisioning reduces considerably the MMOG operation costs with a reasonable loss of performance. Last but not least, we argue in Section VI that, in comparison with the related work, ours is the first to investigate the resource provisioning problem for a multi-MMOG, multi-data center ecosystem.

II. A Model for the MMOG Ecosystem

In this section we present a model for the MMOG ecosystem, in which a global network of data centers services many MMOGs at the same time. First, we introduce a model for a generic MMOG application. Then, we present the hosting platform: data centers scattered around the world. Our multi-MMOG, multi data center model extends the previous work, which focuses on either a single application, usually web service, and/or a single data center [7], [9], [12]–[15].

A. The MMOG Application Model

MMOGs are large-scale simulations of persistent *game worlds* comprising various game objects (*entities*): in-game representation of the players (*avatars*), mobile entities that have the ability to act independently (*bots* or non-player characters (NPCs)), other entities that can be interacted with (*mobiles*), and immutable entities (*decor*). The most used architectural model for implementing MMOGs is client-server [4], with *game operators* maintaining the servers. The clients dynamically connect to game servers and interact with each other within a game session by sending their play actions (e.g., movement, operations on game objects, chat). Based on the actions submitted by the players, the game servers compute the global state of a game world represented by the position and interactions of the entities, and send appropriate responses as a high frequency data stream that guarantees a fluent and seamless game experience.

A good game experience is critical in keeping the players engaged, and has an immediate consequence on the income of the MMOG operators. A lack of bandwidth or of computational power manifests immediately as a lack of responsiveness in the game world, and as a result players may quit the game (and join other games). Unlike other Internet applications, the quitting of one player may trigger a mass-effect (in Section III-D we show that an unpopular decision can lead to a 25% drop in the number of players in a matter of days). Today, a single computer is limited at around 500 simultaneous and persistent network connections, and databases can manage the update of around 500 objects per second [3]. To support at the same time millions of concurrent players (*active concurrent players*) and many more other game entities, MMOG operators are distributing the load of a game world across multiple computational resources [4], [16].

The load of a MMOG application depends not only on the number of active concurrent players, but also on the number and type of their interactions (see Section III-D for empirical evidence). The interaction type and count span a wide range, depending on the game design. The number of interactions between the entities, and especially between the human players, may be very low (e.g., for puzzle games where a player interacts with the system after long periods of thinking), to low (e.g., for a role-playing game (RPG) where small groups of people interact with a sparse environment), to very high (e.g., for a first-person shooter game (FPS) where many players test their reaction time against the others in a confined area). Assuming the number of entities is n, the update model for the various interaction types may range from $O(n)$ for games in which players are mostly solitary and do not require many state changes or compute complex environment reactions, to $O(n^2)$ for games in which many players acting individually are interacting, or to $O(n^3)$ for games in which groups of many players each are interacting. To reduce the computational load, most MMOGs only simulate the areas that are populated with human players, and send updates only for things that directly affect their avatars, that is, they only update the *area of interest* of each avatar. When using such techniques, the update model may become $O(n \times \log n)$ from $O(n^2)$, and $O(n^2 \times \log n)$ from $O(n^3)$.

The lack of responsiveness may also be caused by high latency, independently from the game operator's server and bandwidth capacity. However, depending on the game design MMOGs have different *latency tolerance*; the exact values for several games have been investigated by previous research [17], [18].

B. The Data Center Model

The hosting platform considered in our work consists of data centers scattered around the world. Each data center pools together resources that may serve several users (game operators) at the same time. For simplicity, we assume that each data center consists of a single cluster of computing resources, and that a resource owner (*hoster*) possesses only one data center; we use from here on interchangeably the terms data center, cluster, and hoster. The game operators submit resource requests to the data center, specifying the type and number of resources desired, and the duration for which the resources are needed. The resources considered in this work can be of one of the following four types: CPU time from data center machines (**CPU**), memory from data center machines (**memory**), input from the external network of a data center (**ExtNet[in]**), and output to the external network of a data center (**ExtNet[out]**).

Depending on the data center's service model, either best-effort or based on advance reservations, resource requests are

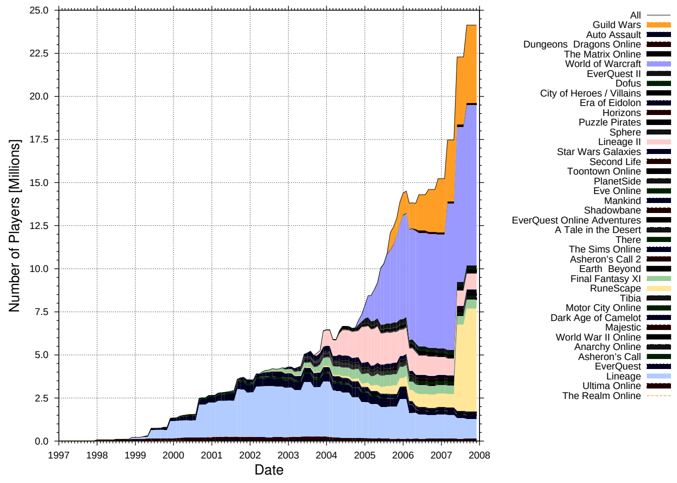

Fig. 1. The number of MMORPG players over time. The six highlighted games, e.g., World of Warcraft and RuneScape, have currently over 500k players each. The data sources are the survey of Woodcock [20] for dates until June 2006, and our own research, afterwards.

queued or immediately fitted in the schedule, respectively. Once the available resources are matched against the requests, these resources are *allocated* to the game operators. From the game operator's point of view, we say that the resources have been *provisioned*; we use from here on interchangeably the terms resource allocation and resource provisioning. The allocated resources are reserved for MMOG execution for the whole duration of the game operator's request, i.e., task preemption or migration are not supported by the system.

Our data center resource allocation model considers the size and the duration of the minimal resource allocation. In practice, a resource allocation, e.g., for CPU or for network bandwidth, may be for a fraction of that resource (i.e., a virtual machine running on a physical node [10] or a channel of an optical network), or for the resource as a whole (i.e., a server in web data centers [12], or a multi-processor node in a grid system [19]). Similarly, the minimal duration for a which a resource may be allocated may be between a few seconds (servicing one user request by a web service) to several months (a typical value for web server hosting). We define the *resource bulk* as the minimum number of resources that can be allocated for one request, expressed as the multiple of a minimal resource size. Similarly, we define the *time bulk* as the minimum duration for which a resource allocation can be made, expressed as the multiple of a minimal time period. A space-time policy expresses the sizes for the resource and of the time bulks. A data center may choose to allocate resources for MMOGs only in bulk, under the space-time policy (*hosting policy*) set by the data center's owners. We discuss in the next section how this definition is applied in the MMOG ecosystem.

C. The Ecosystem: Multiple Game Operators, Multiple Hosters

The MMOG ecosystem comprises multiple game operators that rent resources from multiple hosters for running the MMOG servers. Each hoster may set a different space-time policy for its resources. The game operators handle simultaneously MMOGs of different genres and designs with different interactivity types and counts and with different latency tolerance. Each MMOG may run on multiple data centres geographically distributed across different countries.

The relative success of these games is characterized by the number of players of each game. Figure 1 shows the number of MMORPG players over time for the USA and for the European markets. Notably, there are six games which currently have more than 500k players each. Assuming the same rate of growth, there will be over 60 million players by 2011 in the US and EU markets alone. The number of players both for the largest MMOG and for the MMOG ecosystem as a whole are a motivation motivate for this work.

The game operators make requests based on the load of the games they operate (either statically or dynamically computed), and the data centers respond with offers based on their local time-space renting policy. The resource allocation is realized by a request-offer matching mechanism based on multiple criteria that favor the game operator. We focus in this work on the following three such criteria. First, the number and the type of resources requested must match with the offer; when they do not match, the matching mechanism ensures that the offer includes at least the requested amounts. Second, depending on the game latency tolerance, the matching mechanism locates the resources closest to the request. Third, to deal with data center policies, the matching mechanism selects first the finer grained resources with the shorter period of reservation time.

III. MMOG Workload Analysis

In this section we present an analysis of MMOG workloads. Previous MMOG workload characterizations [11], [21], [22] have either focused on highly interactive games with few players [21], traced small to mid-sized MMOGs [11], [22], or collected data from only one server from a large MMOG [23]. In contrast, our analysis focuses on all server groups of one of the largest commercial MMOGs, RuneScape [24], for which we analyze the workload at server and network level based on server location and on user interactions.

A. The RuneScape Traces

RuneScape is ranked second by number of players in the US and European markets (see Figure 1). RuneScape is not a traditional MMORPG: through specific parts of the world where player interaction follows different rules (in RuneScape terminology *minigames*), the game combines elements of RPG with FPS (and other genres). Thus, various levels of player interactivity coexist in the same game, and the game load cannot be trivially computed (e.g., not with the linear models employed in [15]).

We have collected RuneScape traces from the official RuneScape web page [24] starting with August 2007. Our traces contain the number of players over time for each server group used by the RuneScape game operators. The traces are sampled every two minutes. For this work we have analyzed the traces up to July 2008 (over ten months of data).

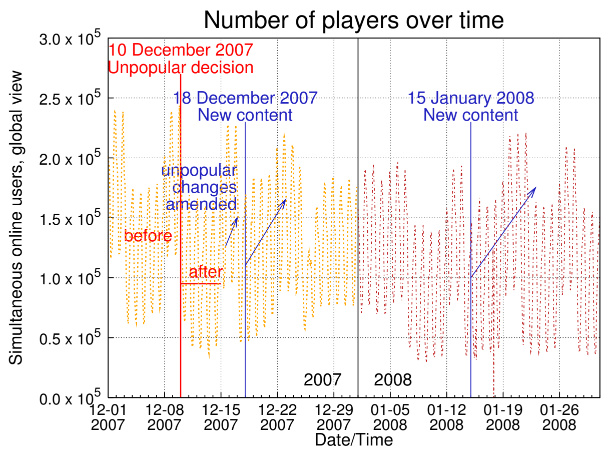

Fig. 2. The number of globally active concurrent players for RuneScape. Each point represents a two-hours average.

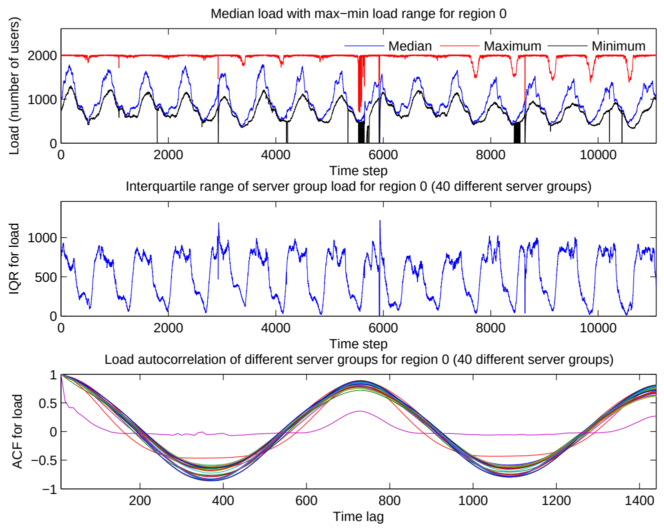

Fig. 3. RuneScape workload for region 0 (Europe).

B. The Global Number of Players

The number of RuneScape players has surged over the past two years, starting with the introduction of the minigames. From 180,000 active players [1] at the beginning of 2005, the game is estimated to have now over 5,000,000 active players from the over 8,000,000 open accounts in 2007 [25], [26]. Our own study of the official list of top RuneScape players counted over 3,000,000 unique active players in September 2007. Given that a player needs to be active (and efficient) for about a month to become a top player, we conclude that RuneScape converts into dedicated players between 30% and 60% of the starting players.

Our study shows that the maximum global number of active concurrent players for RuneScape is around 250,000. However, this number is strongly driven by the mood of the player base. Figure 2 depicts the number of active concurrent users for RuneScape over a period of two months. A highly unpopular decision issued on 10 December 2007 results in massive account cancelations; the number of active concurrent players drops by over 30,000 units (a quarter of its value) in less than one day. Under intense pressure, the game operators agree to amend the changes; the number of active concurrent players raises again, but to only 95% of the previous value. On 18 December 2007 and on 15 January 2008, the game operators release new content; a period of about one week after each release sees an over 50% surge of the number of active concurrent players.

The variability and the dependency on the MMOG being ”fashionable” of both the number of active players and of active concurrent players mean that static resource provisioning would lead to significant over-provisioning. We show in Section V how dynamic resource provisioning can greatly reduce the over-provisioning.

C. Patterns in the Regional Number of Players

Our RuneScape traces characterize a global population spread across five different geographical regions: Europe, US East Coast, US West Coast, etc. We present below the analysis of the number of players that use the server groups of region 0 (Europe). The input for this analysis is a sub-set of two weeks (mid to end of August 2007) from the original RuneScape traces. The analysis of the other four regions, and of the rest of the traces, yielded very similar results. The analysis uses over 11,000 data samples taken at intervals of two minutes, i.e., the whole plot includes data for two full weeks and the two days immediately preceding or following them. Each sample contains a set of values, one value (the load) for each server in the region. Figure 3 depicts the three main topics of the analysis of the regional number of players in three sub-plots. The subplot at the top shows the minimum, the median, and the maximum load measured in number of users online of any server group in the region at each time step. The median load shows a diurnal pattern, that is, sampling at intervals of one day the load would yield very similar values. There is a strong load variation during the peak hours: the median is about 50% higher than the minimum[2].

To characterize the load variability between server groups, the middle subplot depicts the load interquartile range (IQR[3]) over time. Similarly to the median load, the load variability has a diurnal cycle. Unlike the load of e-business and web servers [27], the median load for this 2 week period shows no weekend effects, e.g., the load does not differ significantly between weekend and normal workdays. This behavior is typical for one third of our traces, while the other parts show typical weekend behavior (e.g., the left part of Figure 2).

To establish the duration of the cycles observed in the top and middle sub-plots, the bottom sub-plot displays for each of the server groups in region 0 its autocorrelation function. We can see a very significant peak around 720 (720 samples * 2 minutes / sample = 1440 minutes, i.e., 24 hours) and a strong

[1]An active player is a player that has played at least once in the last month.

[2]Some heavy fluctuations are caused by server group outages. These outages are few and short-lived and thus fall outside the scope of this analysis.

[3]The IQR is the difference between 75th and 25th percentiles of a data set.

negative peak around 360 (12 hours). This shows again the diurnal pattern of the load of most of the servers. However, the same sub-plot shows evidence that there are several servers in region 0 whose load does not follow a diurnal pattern: the load of 2-5% of the servers is always 95%, except for outages.

D. The Influence of Player Interaction on Single Server Load

A fundamental premise of this work is that the workload of MMOGs depends on the interactions between players. We show in this section that this dependence exists.

Using the network tracing tool `tcpdump`, we collect from RuneScape eight game session traces, and analyze the distribution of the packet sizes and of the inter-arrival time (IAT) between consecutive packets. Each trace is collected from a game session of at least five minutes and at most one hour. For validation, two traces, Trace 5a and Trace 5b, were collected from the same environment at consecutive periods of time. To ensure the independence of the measurements, the traces were collected at different dates over a period of six months.

Figure 4 gives evidence that the (network) load depends on the number and type of player interactions. For traces that involve a fast paced game (i.e., T1 and T6), the level of interaction (i.e., crowded or non-crowded) does not increase the server load, as the players are very sensitive to delays. Thus, for fast paced games the server needs to send packets as often as possible, and including as much information as possible. For traces that involve direct player-to-player interaction (i.e., traces T2 (market) and T7 (new content leading to some player-to-player interaction)), the packet sizes are similar, but their IAT is very different; the statistical moments of the IAT of T7 are lower statistically than those of T2 (for T2 the need for updates is conditioned on players starting and agreeing to trades, with more thinking time than for the player actions in T7). For traces that involve group player-to-player interaction (i.e., trace T4), the packets need to arrive more often (lower IAT than for other traces) and to include information about more objects (higher packet size).

IV. Load Prediction for MMOGs

Fast and accurate load prediction with respect to the number of players and interactions per game zone is needed to dynamically allocate resources for MMOGs. In this section we investigate the suitability of several prediction algorithms for MMOGs and find that, from the tried alternatives, a novel algorithm based on neural networks delivers the best accuracy while offering prediction results at an appropriate speed.

A. Predictor Families

Two options are available for quantitative predictions in MMOGs: explanatory models and time series prediction. While the explanatory models can deliver good accuracy with little computation, they are difficult to obtain for such complex applications such as MMOGs, and are tightly-coupled with the application instance and sometimes with the platform for which they have been constructed. With MMOGs relying on frequent and large updates to maintain interest among the players (in Figure 2 we see a rate of one update per month), the explanatory models quickly become unmaintainable. Thus, our work is based on prediction algorithms that use past values to discover patterns in the historical data series, and extrapolate these patterns into the future. Many such prediction algorithms have already been proposed [28].

The simple prediction algorithms (like exponential smoothing and variants thereof) are computationally inexpensive and can be applied in parallel on several data sets, but their predictive power is limited. More elaborated prediction algorithms, e.g., the autoregressive (AR) models, the integrated (I) models, the moving average (MA) models, and combinations thereof like ARMA or ARIMA, try to find the best model for the given data set and base their predictions on this model. Although their predictive power is higher, such methods are also more time consuming and resource intensive, thus being ill suited for MMOGs. As an alternative to the simple and to the complex prediction algorithms, we also investigate in this work the use of neural networks [29]. Neural networks provide a robust approach to approximating real-valued or discrete valued target functions. Low complexity neural networks are capable of approximating complex, noisy functions, provided that a good input preprocessing is performed. This approach has the downside of requiring a training phase which can become computationally intensive once the network structure complexity is increased; however, we show in the next section that accurate predictions can be achieved with our neural network-based predictor at an adequate rate.

B. Prediction in MMOGs

We have shown in Section III that the load of MMOGs is more dynamic than believed, mostly because of the player interactions. Thus, an accurate game load prediction should be based on, besides the entity count, the entity interaction inside the game world. However, it is difficult to obtain detailed interaction information from MMOGs. Instead, the entity interaction can be inferred in practice from the entity distribution in the simulated environment. The game world is partitioned into sub-zones; when the size of the sub-zones is small, the load imposed by the sub-zone can be characterized by using only their entity count. The overall entity distribution in the entire game world consists of a map of entity counts for each sub-zone. The predictor uses as input the entity count for each sub-zone at equidistant past time intervals (*steps*), and delivers as output the entity counts at the next time step. The predicted entity count for the entire game world is the sum of all the sub-zone predictions.

C. The Neural Network-based Predictor for MMOGs

We have developed a neural network-based predictor which uses historical information collected by tracing the execution of MMOGs which we presented in detail in [30]. Our predictor is based on a low complexity multi-layer perceptron (MLP) together with several available signal preprocessors. It is a three layered MLP with a (6,3,1) structure (input, hidden and output neuron layers). The signal preprocessors are based

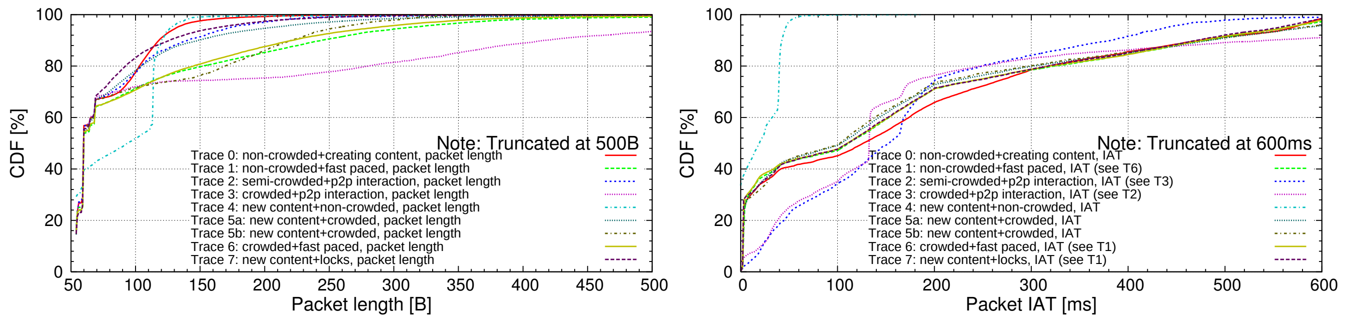

Fig. 4. Evidence of the influence of player interaction on the load of MMOG servers. (*left*) The CDF of packet length; (*right*) The CDF of packet inter-arrival time. Eight traces are displayed.

on several polynomial functions which have the purpose of removing the unwanted noise from the processed signal.

Two off-line phases are required before deploying the neural network-based prediction. The *data set collection* phase is a relatively long process in which the game is observed by gathering entity count samples for all sub-zones at equidistant time steps. The *training* phase uses most of the previously collected samples as training sets, and the remaining samples as test sets. The training phase runs for a number of *training eras*, until a convergence criterion is fulfilled. A training era consists of three steps: (1) presenting all the training sets in sequence to the network; (2) adjusting the network's weights to better fit the expected output (the real entity count for the next time step); and (3) testing the network's prediction capability with the different test sets.

D. Prediction Results

We now evaluate seven time series prediction algorithms for MMOG data: six simple algorithms and our novel neural network-based algorithm. We find that the latter performs best from these alternatives.

1) The MMOG Emulator: For these experiments we have developed a distributed game emulator which supports the concept of sub-zones and realistically emulates the behavior of the game players. The motivation for using an emulator is twofold: we do not have access to the code or the documentation of the RuneScape server, and by developing this emulator we are able to give further evidence that the player interaction determines the server load (see also Section III-D). We use this emulator for generating realistic load patterns with typical MMOG behavior such as entity interaction hotspots for a large number of entities. The emulated players are driven by several Artificial Intelligence (AI) profiles which determine their behavior during a simulation: the *aggressive* profile determines the player to seek and interact with opponents; the *team player* profile causes the player to act in a group together with its teammates; the *scout* profile leads the entity for discovering uncharted zones of the game world (not guaranteeing any interaction); and the *camper* player simulates a well-known tactic in FPS games to hide and wait for the opponent, thus gaining a tactical advantage through the use of the element of surprise. The four profiles have been selected to match the four behavioral profiles most encountered in MMOGs [4], the achiever, the explorer, the socializer, and the killer, respectively. To also account for the mixed behavior encountered in deployed MMOGs [4], each entity has its own preferred profile, but can change the profiles dynamically during the emulation.

TABLE I
CONFIGURATION PARAMETERS FOR OBTAINING THE EIGHT TRACE DATA SETS THROUGH EMULATION.

Data	Player Behaviour [%]				Peak	Peak	Dynamics	
Set	Aggr.	Scout	Team	Camp	Hours	Load	Overall	Inst.
Set 1	80	10	0	10	No	+++	+++++	+++
Set 2	60	10	0	20	No	+	+++	++++
Set 3	70	20	0	10	No	++++	++++	++++
Set 4	70	30	0	0	No	++	++++	+++++
Set 5	30	40	30	0	Yes	++++	++	+++
Set 6	10	80	10	0	Yes	+++++	+	+
Set 7	20	40	40	0	Yes	++++	+++++	++
Set 8	20	80	0	0	Yes	++++	+	+

We model four aspects besides the AI profiles use: the peak hours, the peak load, the overall dynamics, and the instantaneous dynamics. The peak hours correspond to the periods with high player count in online gaming such as late afternoon (see Section III). The peak load represents the highest load that can be observed for a MMOG, and can be used to represent the relative popularity of MMOGs. The overall dynamics represent the variability of the entity interaction over a period of one day. The instantaneous dynamics represent the variability of the entity interaction over a period of two minutes.

Using the game emulator, we generated eight different trace data sets with different characteristics. Each set was obtained by running one simulated day for each set and sampling the game state every two minutes. We varied the entities speed and inter-zone migration patterns to mimic two different dynamic profiles: (1) instantaneous dynamics (typical to fast-paced FPS games) meaning a large difference in the entity interaction over a short period of time; and (2) overall dynamics (typical to MMORPGs) has a much more stable entity interaction load with large variations over a longer period of time. Table I lists the configuration parameters. The eight data sets comprise three major types of signals: *Type I*, signals with high instantaneous dynamics and medium overall dynamics (sets 2, 3, and 4); *Type II*, signals with low instantaneous dynamics (sets 6, 7 and 8); and *Type III*, signals with medium instantaneous dynamics (sets 1 and 5).

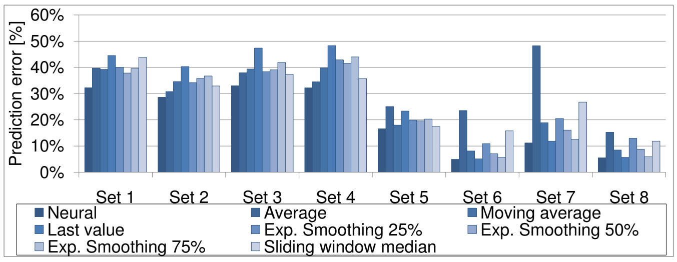

Fig. 5. The accuracy of seven prediction algorithms applied to MMOG data.

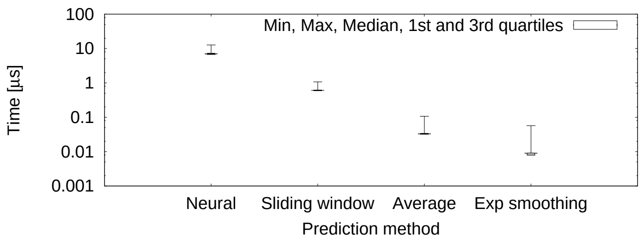

Fig. 6. The statistical properties of the time took to make one prediction for four prediction algorithms applied to MMOG data.

2) The Neural Network-based Predictor Performs Best: Each prediction algorithm receives as input each trace data set, and outputs for each input set sample a prediction for the next two minutes. For a prediction algorithm, we define the un-normalized sample prediction error as the absolute value of the difference between the sample and the prediction made by algorithm for that sample. For the same algorithm, we further define the *prediction error* for an input trace data set as the ratio between the sum of un-normalized sample prediction errors for all samples and the sum of all samples in the trace data set, expressed as a percentage.

The results in Figure 5 show that, apart from having lower prediction errors, the important quality of our neural network predictor is its ability to adapt to various types of input signals. In contrast, other algorithms exhibit poor performance for some types of signals, e.g., the ”Average” predictor is the 2^{nd} most accurate for Type I signals, but performs poorly for all Type II and for some Type III signals. Notably, our neural network predictor was significantly better for the sets with high instantaneous dynamics (Types I and III signals).

Figure 6 depicts the time required to make one prediction on a common off-the-shelf desktop with an Intel Core Duo E6700 ($2.66GHz$) processor for the discussed prediction methods with the exception of the last value method which has no computational requirements. Although the neural predictor is the slowest of the presented methods, with an average prediction duration of approximately $7\mu s$, it nevertheless fits into the *fast prediction methods* category, making it suited to its intended usage.

V. Resource Provisioning and Management for MMOGs

In this section we present an evaluation of our model for resource provisioning and management for MMOGs described in Section II-C. In our simulation, the game operators perform a prediction of the game load (i.e., number of players and interactions per zone) every two minutes and, based on the results, request an appropriate amount of resources to the data centres. The protocol how resources are matched based on their number, type, location, and time was presented in Section II-C. We assume zero overhead in resource allocation, provisioning, and setup from data centres to game operators.

TABLE II
The evaluation space coverage of the experiments presented in this section. The characteristics in bold indicate the main focus of each section.

Section	Resource Allocation	Prediction Algorithm	Update Models	Hosting Policies	Latency Tolerance	No. MMOGs
V-B	**static,dyn.**	**all**	$O(n^2)$	optimal	none	one
V-C	dynamic	Neural	**all**	optimal	none	one
V-D	dynamic	Neural	$O(n^2)$	**all**	none	one
V-E	dynamic	Neural	$O(n^2)$	optimal	**all**	one
V-F	dynamic	Neural	$O(n^2)$	optimal	none	**several**

We cover an evaluation space with six axes: the allocation mechanisms, the prediction algorithms, the player interaction, the hosting policies, the latency tolerance, and the multi-MMOG workloads. To explore this space efficiently, we assess in the following subsections in turn with a set of experiments for each parameter the impact on performance of varying the parameter along one of the axes of the evaluation space; Table II shows an overview of what each experiment covers.

In each experiment we characterize the performance with three metrics: the resource over-allocation, the resource under-allocation, and the number of significant under-allocation events. The *resource over-allocation* characterizes for a resource type, e.g., CPU, the percentage of that resource that has been allocated, from the amount of that resource necessary for the seamless execution of the MMOG. Equation 1 defines $\Omega(t)$, the resource over-allocation at time t, as the cumulated over-allocation for all the machines participating in the game session, where M is the number of machines in the session, $\alpha_m(t)$ represents the allocated resource on machine m and $\lambda_m(t)$ represents the resource usage (the generated load) on machine m. The *resource under-allocation* characterizes the percentage of resources that have not been allocated, from the amount of that resource necessary for the seamless execution of the MMOG, but taking into consideration that missing resources on one machine can be hidden by over-allocating the resource on other machines. Equation 2 defines $\Upsilon(t)$, the resource under-allocation at time t. The $min(\cdot)$ function limits the maximum value of under-allocation to at most 0; thus, an over-allocation at one moment of time does not reduce impact of an under-allocation at another, and the two metrics $\Omega(t)$ and $\Upsilon(t)$ are not correlated.

$$\Omega(t) = \frac{\sum_{m=1}^{M} \alpha_m(t)}{\sum_{m=1}^{M} \lambda_m(t)} \cdot 100[\%] \quad (1)$$

$$\Upsilon(t) = \frac{\sum_{m=1}^{M} min(\alpha_m(t) - \lambda_m(t), 0)}{M} \cdot 100[\%] \quad (2)$$

The *number of significant under-allocation events* character-

TABLE III
THE PHYSICAL CHARACTERISTICS OF THE DATA CENTERS IN THE EXPERIMENTAL ENVIRONMENT.

Location		Data	Machines (total)
Continent	Country	Centers	
Europe	Finland	2	8 machines
	Sweden	2	8 machines
	U.K.	2	20 machines
	Netherlands	2	15 machines
North America	U.S. (West)	2	35 machines
	Canada (West)	1	15 machines
	U.S. (Central)	1	15 machines
	U.S. (East)	2	32 machines
	Canada (East)	1	10 machines
Australia	Australia	2	8 machines

TABLE IV
THE HOSTING POLICIES USED IN SECTION V. SEE TEXT FOR A DESCRIPTION OF THE ABSTRACT RESOURCE UNITS.

Hosting	CPU	Memory	External net.		Time
policy			in	out	[min]
HP-1	0.25	n/a	6	0.33	360
HP-2	0.25	n/a	4	0.5	360
HP-3	0.22	2	n/a	n/a	180
HP-4	0.28	2	n/a	n/a	180
HP-5	0.37	2	n/a	n/a	180
HP-6	0.56	2	n/a	n/a	180
HP-7	1.11	2	n/a	n/a	180
HP-8	0.37	2	n/a	n/a	360
HP-9	0.37	2	n/a	n/a	720
HP-10	0.37	2	n/a	n/a	1440
HP-11	0.37	2	n/a	n/a	2880

izes the number of times the under-allocation causes game play disruption over a long period of time. In this work we consider an under-allocation as being disruptive if its absolute value is over 1%, and a period of time of 2 minutes as being long, that is, if the game is slowed down for more than 2 minutes, players become frustrated and may quit the game; due to the mass quitting effect that may follow (see Section III-B), it is desirable to keep the value of this metric as low as possible.

A. Experimental Setup

The experiments are performed in a simulated RuneScape-like environment. The input workload consists of the first two weeks from the RuneScape trace data analyzed in Section III; with the metrics being evaluated every 2 minutes, this gives over 10,000 metric samples for each simulation, ensuring statistical soundness. The data centers are located on four continents and in seven countries; the location and the number of machines of each data center are depicted in Table III. Each machine in the specified setup is capable of handling at least one game server at full load, e.g., 2000 simultaneous clients for RuneScape. The data centers can be set to use different hosting policies; each policy describes the time bulk and one resource bulk for each type of resource, e.g., one resource bulk for the CPU, one for memory, one for the network bandwidth. The measurement unit for the policy resources is a generic "unit" which represents the requirement for the respective resource of a fully loaded RuneScape game server (e.g. one external outward network unit is equivalent to a real bandwith value of 3 MB/s, see also Figure 4).

B. The Impact of the Prediction Performance

TABLE V
THE AVERAGE PERFORMANCE OF THE DYNAMIC RESOURCE ALLOCATION BASED ON SIX DIFFERENT PREDICTION ALGORITHMS. THE TWO BEST-PERFORMING PREDICTION ALGORITHMS ARE DEPICTED IN BOLD.

Predictor type	Avg. Over-allocation [%]			Avg. Under-allocation [%]		
	CPU	ExtNet [in]	ExtNet [out]	CPU	ExtNet [out]	$\lvert\Upsilon\rvert > 1\%$ events
Neural	**25.90**	**995.27**	**66.04**	**-0.09**	**0**	**317**
Average	32.41	1023.43	69.29	-12.84	-2.46	8123
Last value	**25.11**	**989.10**	**65.36**	**-0.16**	**0**	**608**
Moving average	24.92	992.06	65.69	-0.33	-0.03	1142
Sliding window	24.97	992.73	65.76	-0.41	-0.03	1423
Exp. smoothing	24.76	977.85	64.11	-0.42	-0.03	1429

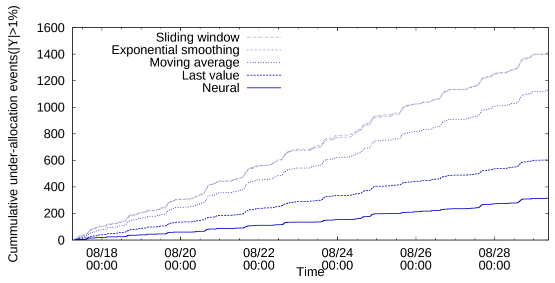

Fig. 7. Cumulative number of significant under-allocation events for five predictors.

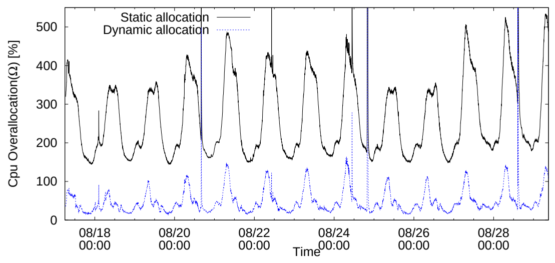

Fig. 8. The resource over-allocation when using static or dynamic resource allocation for the same workload.

In this experiment we evaluate the impact of the prediction algorithm on the performance of the provisioning process. We assign round-robin the *HP-1* and *HP-2* hosting policies described in Table IV to the data centers described in Table III. When two data centers have the same location (column "Country" in Table III), their hosting policies are set one as *HP-1* and one as *HP-2*, and their number of machines is set to half the number of resources at that location (column "Machines" in Table III).

We first compare the performance obtained by dynamic resource allocation based on one of six prediction algorithms (see also Section IV). Table V presents the average performance of the six prediction algorithms. For over-allocation we observe two performance classes: the poor performance class with one member (the Average predictor), and the normal performance class with as members the other five predictors. The reason for the big over-allocation values for the external network bandwidth resource is the fact that the two utilized policies were not well fitted to the input workload, that is, the policies depicted in Table IV included too much external network bandwidth relative to the CPU. We show in Section V-E that the resources from the data centers with such policies are not used when suitable alternatives exist. We further rank the five predictors from the normal performance class using the under-allocation metrics. First, only the Neural

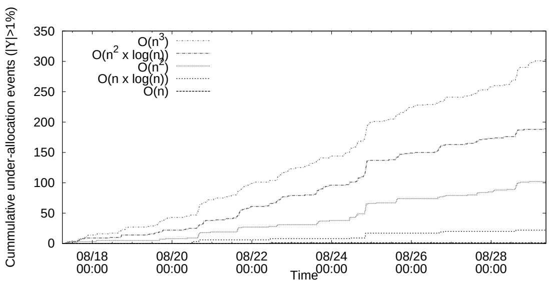

Fig. 10. Cumulative number of significant under-allocation events for five update models.

TABLE VI
AVERAGE PERFORMANCE OF THE STATIC AND OF THE DYNAMIC ALLOCATION MECHANISMS FOR VARIOUS INTERACTION TYPES.

Interaction type	Static allocation method	Dynamic allocation method		
	Avg. allocation	Avg. allocation		$\lvert\Upsilon\rvert > 1\%$
	Over	Over	Under	events
	[%]	[%]	[%]	
$O\left(n\right)$	55.71	8.47	0	1
$O\left(n \cdot log\left(n\right)\right)$	71.86	16.07	-0.024	22
$O\left(n^2\right)$	146.03	27.77	-0.066	103
$O\left(n^2 \cdot log\left(n\right)\right)$	180.60	36.26	-0.096	191
$O\left(n^3\right)$	242.04	54.62	-0.130	304

and the Last value predictors lead to no under-allocation of external network bandwidth; they also lead to the lowest CPU under-allocation. In addition, our Neural predictor yields the lowest number of significant under-allocation events: almost half the value of the Last value predictor. Figure 7 depicts the cumulative number of significant under-allocation events over time for the five predictors with normal over-allocation performance. The cumulative number of significant under-allocation events over time shows a more stable evolution for our Neural predictor than for the other predictors. We conclude that our novel neural predictor has the best resource allocation performance, followed by the last value predictor, thus corroborating the results in Section IV-D.

We now assess the efficiency of the dynamic resource allocation in comparison with its static alternative. The dynamic resource allocation is based for these experiments on Neural, the best-performing predictor. Figure 8 shows the resource over-allocation resulted from the use of static or dynamic resource allocation for the same workload. Expectedly, the dynamic allocation of resources achieves the best resource over-allocation; its average over-allocation is around 25%, compared to 250% for static allocation. The over-allocation of the dynamic allocation of resources can be even lower when the data centers policies are more favorable: the deallocation of resources was allowed only at least six hours after the start of the allocation (column ”Time” from Table IV).

C. The Impact of Player Interaction

In this experiment we assess the impact of the player interaction on the performance of the resource allocation mechanism. The dynamic resource allocation uses the Neural predictor, which gives the best performance from the prediction algorithms considered in this work. Following the MMOG model described in Section II-A, we consider in turn one of the following five update models used for player interaction: $O(n)$, $O(n \times \log n)$, $O(n^2)$, $O(n^2 \times \log n)$, and $O(n^3)$.

Figure 9 shows the the resource over- and under-allocation over time for the $O(n)$, $O(n^2)$, and the $O(n^3)$ MMOG update models, when using dynamic resource allocation. The higher the complexity of the update model, the greater the fluctuations in resource over-allocation. At the same time, the significant under-allocation events become more frequent as the complexity of the update model increases. This is further confirmed by Figure 10, which depicts the cumulative number of significant under-allocation events over time: at the end of the two simulated weeks, this number is significantly higher for $O(n^3)$ than for $O(n)$.

We now compare the results obtained for the dynamic resource allocation to those obtained for the static resource allocation. Table VI shows the average performance of both resource allocation mechanisms for various interaction types. The static resource allocation has 5-7 times higher resource over-allocation than the dynamic resource allocation, but no under-allocation events. However, the number of significant under-allocation events over the whole simulated period when using dynamic resource allocation remains below 3% (at most 304 samples from the over 10,000 samples in the simulation). When even this low occurrence cannot be tolerated, a mechanism that allocates more than the predicted volume of required resourced can be used.

D. The Impact of the Data Centres Hosting Policy

In this section we evaluate the influence of the hosting policies on the performance of the dynamic resource provisioning. Each hosting policy expresses the sizes of the resource and of the time bulks (see also Section II); when more resource types are involved in the matching process, there is a separate bulk size for each resource type. Because the resource and the time bulks have a combined influence the provisioning of resources, we conduct three separate experiments that use each a different resource hosting policies setup. We assess with two experiments the individual impact of the resource bulk and of the time bulk parameters by in turn varying one and keeping the other constant.

We first estimate the impact of the resource bulk variation on the resource allocation performance. We vary the CPU resource bulk by employing five different policies: the policies *HP-3* to *HP-7* from Table IV; the resource bulks for other resource types and the time bulk are kept constant. The values selected for the CPU resource bulk, i.e., from 0.22 to 1.11, are not evenly distributed in the selected interval. This reflects the real-life policies of the data centers, which try to maximize the data center's resource usage and do not willingly adapt to a specific MMOG's resource requirements. We show in Section V-E that in the context of our MMOG ecosystem the game operators are not always forced to accept such conditions, and can penalize the data centers with unsuitable hosting policies by not using their resources.

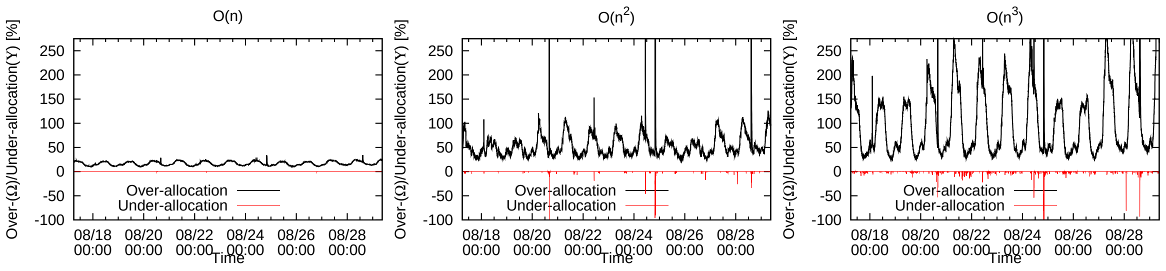
Fig. 9. Over- and under-allocation for three update models.

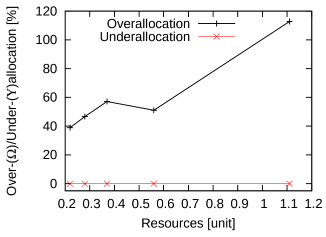
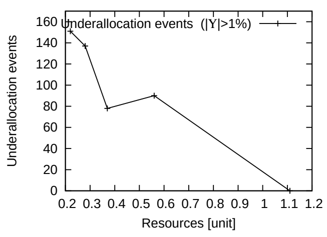
Fig. 11. The impact of the CPU resource bulk on the performance of the dynamic resource allocation.

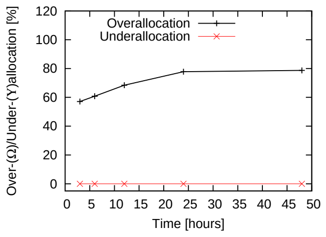
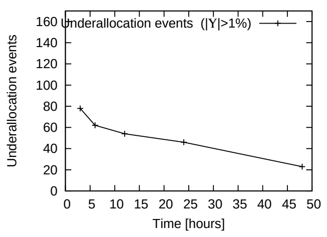
Fig. 12. The impact of the time bulk on the performance of the dynamic resource allocation.

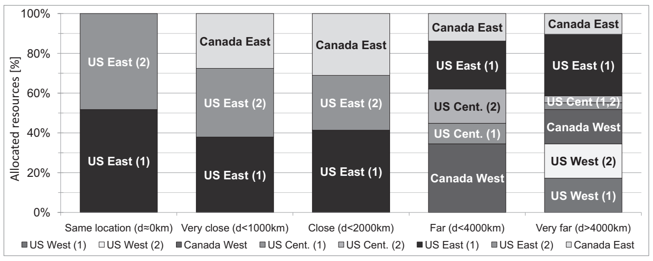
Fig. 13. Distribution of the allocated resources for various latency tolerance values. The workload contains all the North American resource requests.

The influence of the resource bulk on the dynamic resource allocation performance is depicted in Figure 11. Although the trends are not monotonous, there is a visible tendency of higher over-allocation values for bigger resource bulks. Conversely, we can observe an increase in significant under-allocation events as the resources are offered with finer grained quantities. In conclusion, the finer grained policies have the potential to increase the resource allocation efficiency, but also the risk of increasing the number of significant under-allocation events. An optimal value for the resource granularity can be determined with respect to the type of game serviced and its tolerance to resource shortages.

To assess the impact of the time bulk, we vary it from 0.1 to 2.0 days while keeping the resource bulks constant; to this end, we use the policies *HP-5*, and *HP-8* to *HP-11* described in Table IV. The results depicted in Figure 12 show that the efficiency of the resource allocation can be much improved by using resources from the data centers whose policies specify the shortest time bulks. The increase of the average under-allocation is low if the time bulks are set to realistic values, e.g., above one hour.

E. The Impact of the MMOG Latency Tolerance

In this experiment we investigate the impact of the MMOG latency tolerance on the performance of the dynamic resource allocation. We consider an ideal network behavior, thus the latency between the players and the data centers is exclusively determined by their physical distance. The higher the latency tolerance of a MMOG, the further away can the servers be located from the users, and the longer the list of data centers that from which resources can be dynamically provisioned. We show in this experiment that higher latency tolerance leads to the resources of the data centers with unsuitable hosting policies being unused when suitable alternatives exist.

For this experiment we define five classes of maximal physical distances between the location of the players and the location of the servers: *same location* - users must be handled by resources at the same location, *very close* - resources can be allocated from providers within a radius of 1,000 km from their users, *close* - a radius of 2,000 km, *far* - a radius of 4,000 km, and *very far* - any server can serve any user. Note that in practice these distances, while course, would depend on the design of each MMOG. We consider from the setup described in Table III, only the data centers located in the North American region, and select from the workload only the requests that arrive at these data centers. The hosting policies are coarse grained (i.e., with large resource and time bulks) for the data centers located on the East Coast and become gradually finer grained for the data centers located at the Central and West Coast locations.

We assess the system behavior in a realistic situation: under the combined workload of all North American game operators. Figure 13 shows the distribution of the allocated resources for various latency tolerance values. Due to resource contention, the resource allocation follows different patterns than in the optimal case. To assess whether the data centers with unsuitable hosting policies are penalized by having more

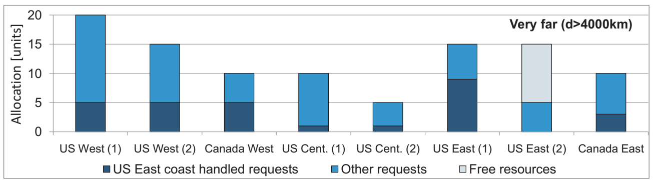

Fig. 14. Resource allocation for all North American data centers for the Very far maximal allocation distance. The workload contains all the North American resource requests.

TABLE VII
OVER- AND UNDER-ALLOCATION AVERAGES WHILE CONCURRENTLY HANDLING DIFFERENT MMOGS.

Workload structure			Avg. Allocation		$\lvert\Upsilon\rvert > 1\%$
MMOG A [%]	MMOG B [%]	MMOG C [%]	Over [%]	Under [%]	events
0	0	100	35.98	-0.12	240
5	5	90	36.89	-0.12	231
10	10	80	36.50	-0.12	220
25	25	50	35.80	-0.10	213
33	33	33	33.84	-0.09	200
0	100	0	33.24	-0.09	216
100	0	0	19.79	-0.03	75

unused resources, we show in Figure 14 the resource allocation for all North American data centers. The unsuitable policies are indeed penalized: the US East Coast data centers are the only ones to have free resources. Moreover, the US East Coast requests are served under the best policies even in a busy system: Figure 14 shows that they use resources from US Central, Canada West, and US West Coasts when the latency tolerance admits Far and Very far maximal service distances.

F. The Impact of Servicing Multiple MMOGs

In this last experiment we evaluate the system behavior when servicing multiple types of MMOGs. To this end, we select three MMOG types with different update models defined in Section V-C: *MMOG A* - using $O\left(n \cdot log\left(n\right)\right)$ interaction pattern, *MMOG B* - using $O\left(n^2\right)$ and *MMOG C* - using $O\left(n^2 \cdot log\left(n\right)\right)$. We run seven scenarios in which the system has to handle resource requests for the three selected MMOG types in different proportions. The workload structure for each scenario is depicted in the ”Workload structure” group of columns from Table VII.

The average performance is summarized in Table VII. When the workload is dominated by the more computing-intensive *B* or *C* MMOGs, i.e., the first six rows in Table VII, the performance of the system is stable: the over-allocation under a workload of only *C* MMOGs is less than 3% higher than under a workload of only *B* MMOGs. In the seventh scenario, when the workload comprises only the (less compute-intensive) *A* MMOGs, the performance of the dynamic resource allocation system is significantly better than in the other scenarios. We conclude that the efficiency of the provisioning system is determined by its biggest consumer. Under these circumstances, game operators of a different MMOG type, e.g., type *A*, may find it better to install their own infrastructure. As an alternative, we plan to investigate in future work the impact of prioritizing the resource requests according to the interaction type of the MMOG.

VI. RELATED WORK

We have pointed throughout this article to work related to the MMOG ecosystem model, the MMOG workload model, and the MMOG load prediction introduced in this work. We turn now our attention to related work in the area of dynamic resource provisioning.

The case when resources from one data center are shared between multiple applications with statistical performance guarantees has received much attention [7], [10], [12], [14], [31]. The MUSE system [12] allocates periodically a percentage of each resource's capacity for each service class such that a target utilization for each service class is achieved at data center level. An approach based on virtual machines as opposed to physical reasources has also been explored [31]. Resource demand profiles for business applications are constructed for various durations (e.g., hour, day, week) and resources are allocated in advance to provide statistical performance guarantees [7]. A similar approach based on application profiles is proposed in [10]. These approaches do not consider the interaction between users and may perform poorly for the MMOG workloads.

The problem of dynamically allocating geographically spread resources to applications has been a popular topic of grid computing research. Recent work investigates mechanisms for resource allocation across single- and multi-cluster grids [19], [32]. They assess the performance of various resource allocation mechanisms for typical grid workloads, comprising batches of scientific and engineering jobs [33]. Unlike MMOGs, these grid applications do not change their resource requirements at runtime. Also, the grid resource allocation policies only allow for whole resources to be allocated at a time; our work also considers the sub-unitary allocation sizes specific to business data centers.

Closest to our work, the benefit of provisioning resources from single data centers has been evaluated for databases and web services [8], [13]. Our work differs from these approaches in two significant ways. First, MMOGs have a different load model, and in particular their load also depends on the interaction between users. Second, we consider multiple data centers to handle the different load patterns in different geographical locations specific to MMOGs.

VII. CONCLUSION AND FUTURE WORK

In this work we focus on MMOGs, a new type of large-scale distributed simulations with a user base of tens and millions and growing. To ensure that the user demand is satisfied at all times, game operators resort to static resource provisioning: they build and maintain computing platforms of up to 10,000 machines located on several continents for a single MMOG. In this work we propose a more efficient alternative to the static resource provisioning: the dynamic resource provisioning and management of data center resources.

Ours is the first thorough investigation of a MMOG ecosystem, that is, of a multi-MMOG, multi-data center environment. We show that the number and the type of interactions between players, and between the players and the environment, are an important contributor to the game load, and we propose a new model for MMOGs that focuses on the interaction count and type. We also show that these interactions lead to much more dynamic resource demands than previously believed, and propose a novel prediction algorithm based on neural-networks that is fast yet accurate; our algorithm performs significantly better than the six time predictors also investigated in this work. We investigate the performance of the resource provisioning and management of data center resources with a large variety of scenarios that focus both on the MMOG-specific properties and on the data center hosting policies. Most importantly, we show that the static resource provisioning can be on average from five up to ten times more inefficient than dynamic allocation under the same conditions, and that in our model the game operators can penalize the data centers with hosting policies that do not suit MMOGs by not using their resources.

We are currently implementing the results of this work within the joint research-industry project EDUTAIN@GRID [34], which aims at providing the middleware between grids and MMOG developers and operators. For the future, we plan to extend this research with more data center sizes and hosting policies. We would also like to study more interaction types from existing MMOGs.

ACKNOWLEDGEMENTS

Part of this work was carried out in the context of the Virtual Laboratory for e-Science project (`www.vl-e.nl`), which is supported by a BSIK grant from the Dutch Ministry of Education, Culture and Science (OC&W), and which is part of the ICT innovation program of the Dutch Ministry of Economic Affairs. Part of this work was funded by EU through IST-034601 edutain@grid and IST-004265 CoreGRID projects.

REFERENCES

[1] A. Shaikh, S. Sahu, M.-C. Rosu, M. Shea, and D. Saha, "On demand platform for online games," *IBM Systems Journal*, vol. 45, no. 1, pp. 7–20, 2006.

[2] C. Neumann, N. Prigent, M. Varvello, and K. Suh, "Challenges in peer-to-peer gaming," *Computer Communication Review*, vol. 37, no. 1, pp. 79–82, 2007.

[3] W. M. White, C. Koch, N. Gupta, J. Gehrke, and A. J. Demers, "Database research opportunities in computer games," *SIGMOD Record*, vol. 36, no. 3, pp. 7–13, 2007.

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

[5] B. Hack, M. Morhaime, J.-F. Grollemund, and N. Bradford, "Introduction to vivendi games," Presentation. [Online] Available: http://www.vivendi.com/, Jun 2006.

[6] R. P. Doyle, J. S. Chase, O. M. Asad, W. Jin, and A. Vahdat, "Model-based resource provisioning in a web service utility," in *USENIX Symposium on Internet Technologies and Systems*, 2003.

[7] J. Rolia, X. Zhu, M. F. Arlitt, and A. Andrzejak, "Statistical service assurances for applications in utility grid environments," *Perform. Eval.*, vol. 58, no. 2-3, pp. 319–339, 2004.

[8] B. Urgaonkar, P. J. Shenoy, A. Chandra, and P. Goyal, "Dynamic provisioning of multi-tier internet applications," in *ICAC*. IEEE CS, 2005, pp. 217–228.

[9] B. Urgaonkar, G. Pacifici, P. J. Shenoy, M. Spreitzer, and A. N. Tantawi, "An analytical model for multi-tier internet services and its applications," in *SIGMETRICS*, D. L. Eager, C. L. Williamson, S. C. Borst, and J. C. S. Lui, Eds. ACM Press, 2005, pp. 291–302.

[10] A. Karve, T. Kimbrel, G. Pacifici, M. Spreitzer, M. Steinder, M. Sviridenko, and A. N. Tantawi, "Dynamic placement for clustered web applications," in *WWW*, L. Carr, D. D. Roure, A. Iyengar, C. A. Goble, and M. Dahlin, Eds. ACM Press, 2006, pp. 595–604.

[11] K.-T. Chen, P. Huang, and C.-L. Lei, "Game traffic analysis: An mmorpg perspective," *Computer Networks*, vol. 50, no. 16, pp. 3002–3023, 2006.

[12] M. Aron, P. Druschel, and W. Zwaenepoel, "Cluster reserves: a mechanism for resource management in cluster-based network servers," in *SIGMETRICS*, 2000, pp. 90–101.

[13] J. S. Chase, D. C. Anderson, P. N. Thakar, A. Vahdat, and R. P. Doyle, "Managing energy and server resources in hosting centres," in *SOSP*, 2001, pp. 103–116.

[14] B. Urgaonkar, P. J. Shenoy, and T. Roscoe, "Resource overbooking and application profiling in shared hosting platforms," in *OSDI*, 2002.

[15] M. Ye and L. Cheng, "System-performance modeling for massively multiplayer online role-playing games," *IBM Systems Journal*, vol. 45, no. 1, pp. 45–58, 2006.

[16] J. Müller and S. Gorlatch, "Rokkatan: scaling an rts game design to the massively multiplayer realm," *Computers in Entertainment*, vol. 4, no. 3, 2006.

[17] M. Claypool, "The effect of latency on user performance in real-time strategy games," *Computer Networks*, vol. 49, no. 1, pp. 52–70, 2005.

[18] M. Claypool and K. T. Claypool, "Latency and player actions in online games," *Commun. ACM*, vol. 49, no. 11, pp. 40–45, 2006.

[19] A. Iosup, D. Epema, T. Tannenbaum, M. Farrellee, and M. Livny, "Inter-operating grids through delegated matchmaking," in *SC*. ACM Press, 2007.

[20] B. S. Woodcock, "An analysis of mmog subscription growth," Report, 21 Edition. [Online] Available: http://www.mmogchart.com, Jun 2006. [Online]. Available: http://www.mmogchart.com

[21] W.-C. Feng, F. Chang, W.-C. Feng, and J. Walpole, "A traffic characterization of popular on-line games," *IEEE/ACM Trans. Netw.*, vol. 13, no. 3, pp. 488–500, 2005.

[22] W.-C. Feng, D. Brandt, and D. Saha, "A long-term study of a popular MMORPG," in *NETGAMES*. ACM Press, 2007, pp. 6–11.

[23] D. Pittman and C. Gauthier, "A measurement study of virtual populations in massively multiplayer online games," in *NETGAMES*. ACM Press, 2007, pp. 25–30.

[24] Jagex, Ltd., "Runescape," MMOG. [Online] Available: http://www.runescape.com/, Nov 2007.

[25] BBC News, Technology, "British gaming firm takes on the world," News Item. [Online] Available: http://news.bbc.co.uk/2/hi/technology/7090490.stm, Nov 2007.

[26] The Screen Digest Group, "Western world MMOG market: 2006 review and forecasts to 2011," Research Report. [Online] Available: http://www.screendigest.com/Reports, Mar 2007.

[27] D. A. Menascé, V. Almeida, R. H. Riedi, F. Ribeiro, R. C. Fonseca, and W. M. Jr., "A hierarchical and multiscale approach to analyze e-business workloads," *Perform. Eval.*, vol. 54, no. 1, pp. 33–57, 2003.

[28] G. E. P. Box, G. M. Jenkins, and G. C. Reinsel, *Time Series Analysis, Forecasting and Control*. Prentice Hall, 1994.

[29] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford University Press, 1996.

[30] V. Nae, R. Prodan, and T. Fahringer, "Neural network-based load prediction for highly dynamic distributed online games," in *Euro-Par*. Springer Verlag, August 2008.

[31] P. Padala, K. G. Shin, X. Zhu, M. Uysal, Z. Wang, S. Singhal, A. Merchant, and K. Salem, "Adaptive control of virtualized resources in utility computing environments," in *EuroSys*. ACM Press, 2007, pp. 289–302.

[32] M. Siddiqui, A. Villazón, and T. Fahringer, "Grid allocation and reservation - grid capacity planning with negotiation-based advance reservation for optimized qos." in *SC*, 2006, p. 103.

[33] A. Iosup, C. Dumitrescu, D. H. Epema, H. Li, and L. Wolters, "How are real grids used? The analysis of four grid traces and its implications." in *GRID*. IEEE CS, 2006, pp. 262–270.

[34] T. Fahringer, C. Anthes, A. Arragon, A. Lipaj, J. Müller-Iden, C. J. Rawlings, R. Prodan, and M. Surridge, "The edutain@grid project," in *GECON*, ser. LNCS, vol. 4685. Springer-Verlag, 2007, pp. 182–187.