Bringing Online Gaming to the Cloud: a Case Study

Michele Pirovano
Dipartimento di Elettronica e Informazione
Politecnico di Milano, Italy
Email: michele.pirovano@elet.polimi.it

Abstract—**Online gaming has been a main trend in the field of video games for more than ten years and it is still rapidly evolving, with new game archetypes and tailored software architectures being created every year. These games are typically harder to create and maintain than offline games due to the limitations of the current web infrastructures that conflict with the need for fast, fluid and fun games. However, the boom of Cloud Computing services in the recent years has created new opportunities and removed many of the limitations that are typically tied to software deployed on the Internet. It has been shown that the development of online games can benefit from the integration of Cloud services into their design in order to solve many of the problems that plague their development and maintenance. In this paper, we first discuss the current designs of online gaming architectures and present their limits. The solutions offered by using Cloud services in respect to these video games are then explored and the new problems that arise from this combination are addressed. We also present a case study by deploying an online First Person Shooter multiplayer video game on currently available Cloud infrastructures (*using* Amazon Elastic Compute Cloud) and detailing our architecture specifically designed for this purpose. We show that, where once entering the market of online games was solely affordable by big companies due to the high infrastructure costs, a single person could now create and deploy an online game.**

I. Introduction

Online video gaming has seen its market grow relentlessly in the last years. More and more triple-A videogames now sport some kind of Internet functionality, from the simple download of additional content to full-fledged persistent online worlds.

In addition, totally new game archetypes were born during the recent years, such as ~~the~~ Massive Multiplayer Online games (MMO), where thousands of people can play together at the same time (see Figure 1), Browser games, simple and fast games designed to work in most browsers, and Social games, which focus on social play and communication. These are all forms of videogames designed to take advantage of Internet access. It has been reported that the Online game market has reached in 2010 a value of more than \$23 millions[1].

Online games require ad-hoc hardware and software architectures, as well as game design efforts around the technology limitations of Internet play. Most online videogames require a persistent connection as well as real-time responses to guarantee good interaction times. In addition, most competitive online games are in need of a strict anti-cheating and anti-hacking policy. Additional requirements of many online games are the massive storage of the user data of up to millions of people and the stable hosting of even thousands of concurrent users.

Online game architectures can benefit from the introduction of Cloud-based services, resulting in reduced costs for the developer and a more enjoyable and novel experience for the player. **Most of the specific requirements of online games, as will be further discussed throughout this paper, can be fulfilled by employing services offered by Cloud providers.** Among **those services**, we can mention scalable computational power, persistent and fault-free storage and elasticity to be beneficial to the current online gaming solutions.

In this paper, we aim to assess the feasibility of deploying an online game on a Cloud infrastructure and analyze the major design and technology issues **and opportunities** of this novel integration.

Chapter III presents the current online games architectures and explores their design peculiarities. In chapter IV, Cloud services useful to the online gaming market are detailed, pointing out what limitations can be surpassed. A case study is then presented in section V, where the details of deploying an online game on the Cloud are presented. Section VI finally presents conclusions and directions for future work.

II. Related Work

The peculiar needs of online games, especially persistent world online games, have been discussed throughly in the years due to the special server architecture requisites they impose (see [1] for an introduction). Ad-hoc server architectures, tailored to online video games, have been already explored from as long as ten years ago [2]. Architectures that take advantage Cloud services for online video games have also been suggested in more recent years (see [3] for an example).

Cloud services are indeed already being used effectively by a few online game producers, especially for smaller-scale MMOs (such as Dragon Nest[2] or the games of PixelPandemic[3]) and Social games (such as **the games of PlayFish**[4]). However, these cases are still rare and there is still much room for improvement, such as bringing Cloud services to

[1]http://www.mcvuk.com/news/read/global-games-market-worth-over-100bn/07021

[2]http://dn.cherrycredits.com/

[3]http://pixelpandemic.net/

[4]http://www.playfish.com/

different genres of games. The OnLive[5] service must also be mentioned, as it provides games to be played using any **capable device** and broadband connections through the use of a Cloud infrastructure. A few Cloud solutions have been already put to use for online game deployment, such as Terracotta, that alongside SmartFoxServer can be used to create a clustered online game deployed on the Cloud [4].

Figure 1. A typical crowded scene when playing a MMOG. Copyright Blizzard Entertainment.

III. Online Game Design

In the current state-of-the-art, the design of online games and their architectures still presents open issues. The design space of videogames is often constrained by the technology required to run them [5], [6]. This is true for classic single player videogames, and even more so for online multiplayer videogames, that require their developers to cope with ad-hoc architectures and, in particular, with the limited bandwidth, storage space and connection speed.

Apart from turn-based strategy games and similar games, who are now a minority, online games, much like their offline counterparts, require *real-time interaction*. This is usually obtained through careful design of the game, making sure to communicate rapidly and with small packets of data, only sending necessary data. For this reason, online games usually use protocols such as TCP (for persistent connections and absolutely needed data) and UDP (for fast, but unreliable, communication). Multiplayer online videogames thus require, for the most part, *persistent connections* to achieve high speed and reliable transmission. This fact often differentiates games from other web-based applications.

Cheating and hacking are another problem that online game developers must cope with. *Anti-cheating* is needed for competitive online games, while *anti-hacking* is a requirement for those multiplayer video games that provide real-valued virtual items. Many on-line video games now provide virtual items that can be earned in-game or even bought with real money. This is often the case with recent massive online games, that work with micro-transactions economies. Such games also require secure and reliable databases since hackers may access user accounts and steal or delete their items, or players may even duplicate their own items, therefore ruining the game economy (for instance, see [7]). In addition, those same massive multiplayer games can reach huge amounts of users, with some games reaching millions of subscribers[6]. Since those games require the servers to hold all information on players and their avatars, the *storage space* needed can become as massive as the game itself, which leads to greatly increased server costs. The game simulation servers also need to host hundreds if not thousands of users at the same time and thus require *high computational power*. Again, for massive multiplayer games, this can become quite costly. Architectures thus need to be somewhat scalable and this is not easy to do with typical server infrastructures, which **may** require the developer to hold and maintain a big cluster of datacenters. **Different servers can and are used for the two purposes of storage and simulation.**

Taking into account such requirements, online videogames typically present one of three basic client-server architectures. The choice is usually based on the game genre and its requirements.

The most common architecture uses a *dedicated server*, that is an authoritative server that runs the game (or a model of the game) by itself, enforcing its rules. Clients can connect to the server and interact with it. This architecture protects from cheating and does not favor any client, but it is costly due to the maintenance and running costs of the server machines. The architecture is used by multiple-user multiplayer games such as competitive action games or massive multiplayer games and is currently the preferred one by **large** companies [8].

A second architecture uses a *listen server*. In this case, the server is hosted by one of the users, who also plays on the same machine through a client. The other clients connect to the server and play with him. This removes the server costs from the game developer, but it is much easier to cheat in such servers (especially for the host) and the host will usually get a delay advantage, since he will have a faster channel of communication with the server. In addition, should the host user disconnect, the server also shuts down.

A last architecture is called *peer-to-peer*. With this architecture, there is no server and the peers just communicate their actions to each other. Each peer then reconstructs the results of all actions. This architecture is usually used for two-player strategy games, where the sheer number of units would require too much information to be sent from and to a server.

The architectures focus on the game servers but usually also support a data server that stores all the user data. This is especially common for dedicated server architectures and in particular with MMOs.

IV. Enter the Cloud

The term *Cloud Computing* has been used in the recent years with some confusion on its meaning. For additional

[5]http://www.onlive.com/

[6]http://wow.joystiq.com/2010/10/07/world-of-warcraft-reaches-12-million-players/

clarity, we will refer to the convention **defined by** [9] and summarize with the term Cloud the following definition: *Cloud Computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.* Such a definition encompasses very diverse products and, as such, for the purpose of this paper we will instead focus on what elements of Cloud Computing can be useful for online gaming.

Online games architectures can greatly benefit from cloud solutions through different services, which can be used to address the **requirements** that have been presented in section III. **From the previous analysis, we summarize the main requirements of online games in the order that they are addressed in the rest of this section: large and reliable storage space for holding user data, high computational power for game servers, anti-cheating and anti-hacking safeguards, real-time interaction and thus persistent connections.**

First of all, the storage requirements for the **data** servers, especially critical for massive online games, can be easily fulfilled using those Cloud services that provide a reliable and secure source of storage space. Cloud Computing in its more etymological definition can also be useful to online gaming, as most online games require servers to be always up and ready, often even providing persistent worlds. Video game users are even likely to abandon the game altogether if the reliability of the servers is not high enough. Among the services of the Cloud providers, reliable and scalable computation power is offered, often with a pay per hour billing contract.

The online gaming market is characterized by great and periodic fluctuations in user load in daily, weekly and monthly periods. **In Figure 2, the daily fluctuations in user load on the Steam platform can be seen**[7]**. From the plot, we can see that during the course of the day the number of concurrent users logged on the platform oscillates from a minimum of around 2.5 millions to a maximum of almost 5 millions of users. This means that at certain times half of the Steam servers are likely to be idling.** This periodic activity thus often requires online videogame developers to provide servers even when there is a small number of users logged in. It is not rare for game servers to get full **and clogged** on peak hours and, in this case, user discontent quickly kicks in. In addition, due to the ever-changing flickering tastes of gamers, an online game developer may be confronted with unusual drops or rises in the concurrent logged user base. The elasticity provided by Cloud Computing services both for computation and storage can therefore be a great ally to the game developers. Most online games are already designed to be easily scalable and thus the transfer from a typical architecture to a Cloud-based architecture is painless. Another important advantage of switching to Cloud providers for the server infrastructure is the fact that the game developer does not have to deal with **possibly** faulty hardware **and its lifecycle issues**, **time-consuming** operating system **set-ups** or physical server maintenance. By shifting this burden from the game developer to the Cloud provider, the developer can allocate more resources to the good design and implementation of the actual game and thus create a better product.

[7]http://store.steampowered.com/stats/

It is clear that one main advantage of the Cloud services lies in the reduced infrastructure cost in terms of money and time. As a consequence another advantage can be found as the Cloud opens the market of online games, typically reserved for big and affirmed companies, to smaller developers, to the point that a single person could create and run an online multiplayer game (as is shown in this article).

In our case study (see Section V) we take advantage of the scalable computing power and the elasticity offered by current Cloud providers. We focus on dedicated server architectures since their only downside **is** the infrastructure cost, **a problem taht can be relaxed** thanks to these Cloud services.

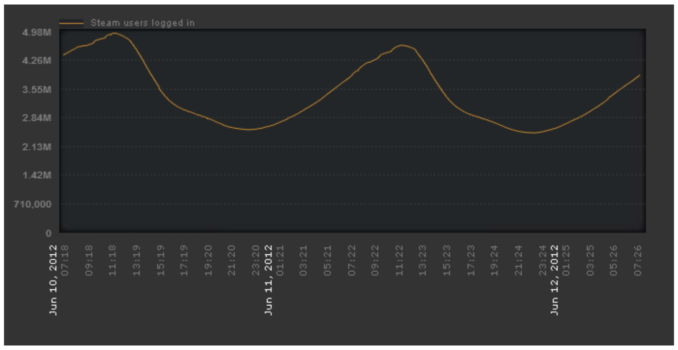

Figure 2. The periodic user activity on the Steam platform.

Regarding anti-hacking and anti-cheating policies, we must address the doubts that many people have regarding the security of Cloud services. The main issue that is tied to Cloud services in respect to security is the destiny of user's sensible data. Typically, when a user registers to an online service, his data is held by the service provider that is in charge of providing a secure and private storage space for it. When using cloud services, however, such data is held on unknown servers somewhere in the world and the service provider usually never even knowns whether this data is close or near to his location, nor who is in charge of handling it. This fact can become a serious concern and is one of the main open issues of Cloud services. It has been reported that most information technology professionals trust the Cloud providers, who claim that the security they offer is high due to the decentralized data centers [10]. In the case of online games, there are two types of security concerns: cheating and hacking. Cheating is easily handled by the authoritative game server and thus by the game developer's application. For this reason, the presence of the Cloud doesn't make a change. Hacking, on the other hand, is stopped by highly secure data servers and secure connections. This is especially needed for sensible user data such as their credit card information, but is also important

for in-game virtual items. The burden of anti-hacking thus shifts from the game developer to the Cloud provider and can become a problem if the Cloud provider is not trusted enough. This is still an open issue, but security on the Cloud is listed as a top priority and as such this problem could be resolved in the near future.

A particular discussion must be made about persistent connections. As previously mentioned, having to guarantee a persistent connection to the server is a peculiarity of online games in respect to other online applications. This fact becomes problematic when using Cloud Computing services. In most other online applications, the user usually connects to the server to send a message and get the response. The connection is usually closed afterwards. Due to this, *load balancing* infrastructures have been devised for Cloud services which allow the user to be transparently redirected to a new server when the original one is clogged. The load balancing also takes care of scaling the application capacity by creating or terminating server instances. With the persistent connections of online games, this cannot be done. The user must stay connected to the same server even if it is full of other people and the server cannot be shut down even if there is just one person on the server, since kicking the user out is not an option.

This problem could be attacked by changing the initial design of the video game. The typical massive online game world is divided into *virtual zones* (see [11]), where one server holds one or more virtual zones. The players will move around the game world to explore it and play in it and thus change virtual zones. This way, the persistent connection to one server may be closed from one server and opened in another one, introducing however a connection delay and thus interrupting the real-time game flow. Many massive multiplayer online games already use this solution. The same solution can be beneficial for load balancing, since when the user switches from one server to another, the load balancing logic can redirect the user to the proper server, without the user even noticing. **In addition**, the user may be **directly** encouraged to leave servers which are too crowded or empty by the design of the game. By designing the game around this idea, a natural balance of user load would arise. To our knowledge, no game currently pursues this idea, even if this can arise naturally in some massive multiplayer games when a server becomes too crowded (and thus the player may have too much delay) or empty (since the player will not have anybody to play with). The latter case is beneficial, but the former presents a bad side effect (the delay) which is not desirable, hence the need for specific design.

This virtual world division, however, is not done for some genres of games such as First Person Shooters (FPS), since the users will not change servers, they will keep playing on the same server (often with the same people) until they willingly disconnect. A different solution altogether **to virtual world division** may be found by modifying the load balancing logic and scope, as presented in our case study, **as explained in the next section**.

V. Bringing Games to the Cloud: a Practical Case

As a case study to assess the feasibility of bringing online gaming to the Cloud, this chapter discusses the design and deployment of an online multiplayer game on the Amazon Elastic Compute Cloud (EC2) infrastructure. The EC2 service allows the utilization of virtual computer instances for computation, using Amazon's Cloud resources. Software used in the accomplishment of this study comprises Unity3D, SmartFoxServer, MySQL and the Amazon Web Service Java SDK.

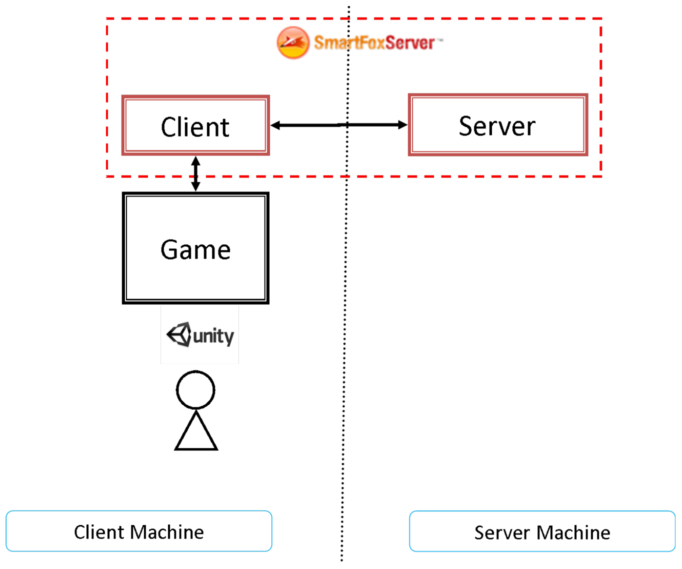

Figure 3. The classic multiplayer FPS architecture

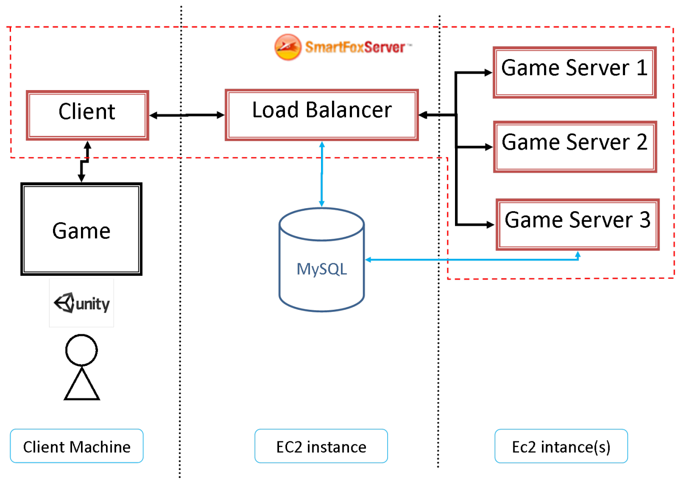

Figure 4. The extended multiplayer FPS Cloud architecture

A. An Online First Person Shooter

The game brought to the Cloud is a simplified version of a First Person Shooter (FPS), similar to many current successful multiplayer games, built with the Unity3D videogame development framework[8]. The Unity3D framework allows game developers to easily create prototypes of videogames for early testing, as well as providing all the features needed to create a complete videogame and deploy it on several platforms, removing this burden from the game developer, who can then concentrate on designing and creating a successful game. Our game is a modification of the FPS test example provided by the SmartFoxServer documentation[9]. The game consists of a 3D arena that players can enter, playing as an armed soldier against online opponents. The goal of the game is to collect points by shooting and thus killing the virtual avatars of opponents. Health packs and ammo packs are scattered around the map, adding complexity to the individual strategies. The application in which the game runs is composed of two communicating parts: the client and the server.

The client is deployed as a browser game and can thus be played in any browser, provided the Unity Web Plugin is installed. This is the only software that the player needs in order to play.

The server is hosted on a specific machine (which is not required to be the same as the client's) and it is built using the SmartFoxServer2X framework (Sfs2x). This framework provides a complete solution for deploying videogames on the Internet. Sfs2x provides C# APIs for Unity3D, used to communicate with the server, and Java APIs for the server machine.

Users can log into the game server through the client, create virtual game rooms and play with other people in the same room. While users play, the server holds a model of the game world. The server receives messages from the clients, it simulates and updates the whole game state, then returns the updated information to the clients. The architecture is thus a *dedicated server* and can be summarized by Figure 3. As mentioned, it can reduce cheating and does not grant unfair advantages to a specific player, but these gains are usually offset by the costs of running the server. In particular, for a game with a large user base, hundreds of different servers may be needed to meet the users' requests. Taking into account this problem, the inner workings of the game server have been here extended in order to take advantage of Cloud services. Our aim is to decrease the usual costs of dedicated servers by dynamically instantiating and removing virtual instances (and thus server machines). The new architecture designed and implemented for this study can be seen in Figure 4.

B. Our Architecture

In our architecture, the user first logs into a special server here named *Load Balancer*. The Load Balancer server is responsible of balancing the number of Game Servers that must be running simultaneously to meet the user load demand. The Load Balancer receives user connections and redirects them to the proper Game Server, It also decides when to shut down or open new Game Server instances using the Amazon Web Service Java SDK. Load balancing features are often provided by the Cloud provisioner, as is the case with the Amazon Elastic Load Balancing service, but, as previously mentioned, online games require persistent connections. When the user logs into a server, the session persists until the user himself logs out willingly (or, in extreme cases, he is kicked out by the authoritative server). Thus, the Load Balancer has been here implemented anew.

Both the Load Balancer server and the Game Servers are hosted on Amazon EC2 micro instances. All the instances are based off an Ubuntu 10.f standard Amazon instance, with a SmartFoxServer instance running on them. In particular, all instances have the 9933 (Sfs2x admin) and 8080 (HTTP) ports open to the public. The Load Balancer also opens port 3600 for MySQL database access, but only to the rest of the instances (as such, all instances can read the database). This configuration is saved as a Security Group through the Amazon EC2 Web Server Interface. Each Game Server runs with a Java extension which takes care of inserting information about the server into a MySQL database which runs on the Load Balancer. The MySQL database maintains a table of servers. Each row of this table holds, for each server, its public IP, the EC2 instance identifier and the number of currently logged users, which is initialized at zero. The Game Server extension periodically updates the load of the server in the same database once every five minutes.

C. Usage Case

The flow of events following the connection of an user to the game is here detailed and is summarized in Figure 5. The user initially connects to the Load Balancer server as the client is started. The IP of the Load Balancer server is already known by the client application. The user is then logged through a minimal user interface into the Load Balancer, that reads the current user load on each Game Server instance from the shared MySQL database.The Load Balancer then selects a server according to the following logic. From the list of running Game Server instances, the servers with more than M users are not considered, with M being the maximum number of concurrent users sustainable by a server while ensuring a maximum delay of dt milliseconds (a value which is defined according to the game type, usually below 100 milliseconds for a FPS). During this phase, the servers with zero users online are marked for removal. The remaining n servers are then given a candidate value w_i for each server $i = 1 \cdots n$. This candidate value is partly based on the number of user currently logged into the server (the more the people, the lower the value) and it is then offset by a random amount, inserted here to avoid sending all connections to the same server if it has always the minimum number of users. At last, all the servers marked for removal but one have their instance terminated in order to avoid paying for unused power. One server, even if it has zero users logged in, is not destroyed in order to always have at least an empty server fully available for user peaks. In the case that there are no servers with less than M users logged in, the Load Balancer will create a new

[8]http://unity3d.com/
[9]http://docs2x.smartfoxserver.com/ExamplesUnity/fps

Game Server instance to be used by the next client, while redirecting the user to the current least full game server (since the instance will take a few minutes to start). After the proper Game Server is found, its IP address is sent to the client, which automatically connects to it, logs in and starts the game. This flow of events may seem over-complicated, but since the SmartFoxServer requires both a connection and login request for security purposes, we cannot remove any steps. Notice, however, that all of the steps are automatic and transparent to the user, as if he was connecting directly to a game server.

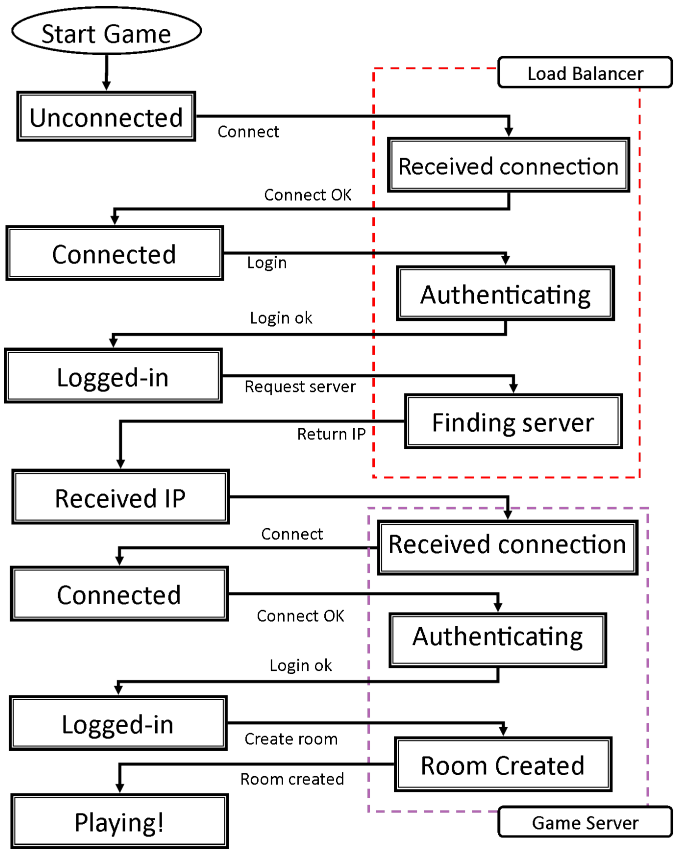

Figure 5. The flow of connections when the user starts the game

VI. Conclusion and Future Work

We present here the results of our study. With our architecture, we are able to provide a scalable and elastic server infrastructure for a multiplayer game. Thanks to this fact, on the one hand, we can scale up the total capacity of the game servers and thus provide a challenging and satisfying experience for any number of users, avoiding the delay which is typical of crowded online game servers. On the other hand, we avoid spending money on idle servers during shortages of users. Using an integrated architecture composed of softwares with free licenses, we were in fact able to create the whole architecture with no cost, apart from the running costs of the Amazon EC2 instances, which is quite small and completely scales with the number of logged users. All of this was the work of one person, showing that the days when only big companies with even bigger budgets were the only ones that could deploy a multiplayer online videogame are over.

In conclusion, Cloud Computing can be an useful asset in the hands of online video game developers, easing the tedious and difficult work of setting up and maintaining the server infrastructures by themselves and thus dedicating effort and even specific people to the task. Indeed, due to the scalable costs and no setup issues, we can safely say that the costs **both in terms of money and effort** of running an online game on the Cloud are ~~way~~ lower than doing so with dedicated machines. **The main advantages of bringing online gaming to the Cloud are thus (i) the reduced costs of deployment and maintenance and subsequently (ii) the expansion of the market to smaller companies.**

Much work has yet to be done **as** the appearance of Cloud Computing in the online videogame field is still recent. We can point to future work in two areas, **for game design. Regarding game design,** in this paper we present our results for a typical multiplayer game, but the same results could be applied to less explored designs. The design of online games, typically limited by the many constraints imposed by the classic server-client architecture and the high costs of running the servers, can be expanded thanks to the use of Cloud services. Games which were only possible offline could now be ported to an online environment in an extremely easier way and with minimal costs compared to a few years ago.**Possibly, new archetypes could be even created**. **Regarding server infrastructures, current MMOs are still using the virtual world division approach in the handling of multiple servers. We can point to future work in the creation of dynamic and elastic clusters of servers hosted on Cloud infrastructures. These architectures could allow an even greater number of users to play the same game in the same virtual world than is possible as of today.**

Inside the scope of this project, possible work can be testing the implemented architecture with a high number of users, retrieving actual data on system usage and costs, in order to better compare these costs to classic architectures.

Acknowledgment

The author wishes to thank professor Dario Maggiorini for his introduction to online game design and directions for further research.

References

[1] M. Lapi, "Smartfoxserver 2x - server architecture white paper," http://www.smartfoxserver.com/downloads/sfs2x/documents/SFS2X_WP_ServerArchitecture.pdf, 2012.
[2] W. Cai, P. Xavier, S. J. Turner, and B.-S. Lee, "A scalable architecture for supporting interactive games on the internet," in *Proceedings of the sixteenth workshop on Parallel and distributed simulation*. IEEE Computer Society, 2002, pp. 60–67.
[3] M. Marzolla, S. Ferretti, and G. D'Angelo, "Dynamic resource provisioning for cloud-based gaming infrastructures," *to appear in ACM Computers in Entertainment*, 2011.
[4] A. Integrated, "Clustering smartfoxserver using terracotta smartfoxserver," http://www.smartfoxserver.com/downloads/sfs1/clustering/SFS_clustering_A51.pdf, 2012.

[5] J. Schell, *The Art of Game Design: Book of Lenses*. Elsevier, 2008.

[6] K. Salen and E. Zimmerman, *Rules of Play - Game Design Fundamentals*. MIT Press, 2003.

[7] T. Phillips, “Diablo 3 duplication exploit,” http://www.eurogamer.net/articles/2012-06-11-diablo-3-asian-servers-offline-after-item-duplication-exploit, 2012.

[8] M. Weilbacher, “Dedicated servers in gears of war 3 - scaling to millions of players,” http://www.gdcvault.com/play/1015337/Dedicated-Servers-In-Gears-of, 2012.

[9] P. Mell and T. Grance, “The nist definition of cloud computing,” *Nist Special Publication*, vol. 145, no. 6, p. 7, 2011.

[10] T. Olavsrud and D. Muse, “How secure is the cloud? it pros speak up,” http://www.cio.com/article/703064/How_Secure_Is_the_Cloud_IT_Pros_Speak_Up, 2012.

[11] M. Lapi, “Smartfoxserver 2x - performance and scalability white paper,” http://www.smartfoxserver.com/downloads/sfs2x/documents/SFS2X_WP_PerformanceAndScalability.pdf, 2012.