A Simple Architecture for Digital Games On Demand using low Performance Resources under a Cloud Computing Paradigm

Diego Cordeiro Barboza Hamilton Lima Junior* Esteban Walter Gonzalez Clua
Vinod E. F. Rebello

Universidade Federal Fluminense, Instituto de Computação - Media Lab, Brasil

Abstract

Cloud computing is becoming an increasingly viable source of low cost computing power for developers and users in diverse areas. Services from the creation of presentations, spreadsheets and text processing, to picture and video editing, and more recently high performance scientific computing are some examples of systems currently available in the cloud. While these applications are typically executed in the form of batch jobs, responsiveness or timeliness is not usually an issue. Executing interactive applications, i.e. applications that require real time responsiveness in the cloud, however, is more challenging and still not so common in this environment. This work proposes and evaluates the feasibility of building a simple off-the-shelf architecture for an on demand gaming service. Our proposal consists on running an appropriate remote server in the cloud so that the client need only perform a few basic tasks, such as reading user input and displaying the resulting game screens. Consequently, even low-power computing systems, such as mobile devices or digital TV setup boxes, can have access to sophisticated graphic rich or complex games, since most of the processing is performed remotely.

Keywords: cloud computing, games, computer networks

Authors' contact:
*hlima@acm.org
{dbarboza, esteban, vinod}@ic.uff.br

1. Introduction

Today, digital games reach further than just entertainment, with applications in diverse areas of society including health and education, and as such are reaching a broader public each day [ESA, 2010]. Its popularization is being further driven through the availability of applications on computational devices of all kinds, but in particular cheap low power equipment such as cell phones or digital TV receivers. Still, the development of games that achieve a good balance between technical quality and required processing power of devices is a challenge for developers. The higher the level of detail in graphics, artificial intelligence and physical modeling, for example, the higher the computational requirements and smaller is the group of devices able to execute them. Thus, a balance must be found to avoid launching games that only a small fraction of the target audience can play.

The splitting of the processing of a game in two parts: the client - responsible for reading the user's commands and display the game's graphical output - and a server - responsible for most of the processing such as generation of graphics, modeling of physical processes and artificial intelligence - might be expected to minimize the problems relating from the correlation between processing power and technical qualities. The games are being processed on servers controlled by the service provider and client devices need only basic computer capacity and a network access to play. Of course, now an additional component - communication across the network - will also affect performance. Today, almost all computational devices come with wireless connectivity, e.g. WiFi, GSM, and telecommunication companies have been steadily increasing their backbone bandwidth capacity to support new applications such as digital TV.

The concepts of cloud computing [Armbrust et al, 2009] can be applied to build a client/server game infrastructure, where a player consumes a service available in the cloud that is able to manage all system components, performing access control, establishing communication between the client and server, managing the hardware resources, software available and so on. This work aims to explore the concept of cloud computing applied to the use of digital games as a service. It is intended to study the feasibility of running games remotely, through a cloud computing architecture, and display the result to the user in real time. Also, within the scope of this work is the manipulation of virtual machines that are initiated and finalized in accordance with the requirements of the system to provide on demand services to new customers.

The following aspects were studied in the context of this work: the design of a service management system for the execution of multiple types of digital games, including the management and manipulation of virtual machines on demand; the construction of a service capable of behaving like a game server in the cloud; and building a game client capable of soliciting and receiving the outcome of the game rendered in the cloud, displaying it to the user, and receiving and transmitting interactions with the user.

This paper is organized as follows: Section 2 presents an brief overview of the little related work encountered; in Section 3, we describe some concepts of cloud computing, while Section 4 details the development of the various system components proposed in this work. In Section 5, we comment on some of the initial results obtained and present our final considerations and future work in Section 6.

2. Related work

As far as we can tell, relatively little has been explored to date in the context of taking advantage of combining the concept of cloud computing with digital games.

OnLive is a game on demand service developed primarily for desktop operating systems, like *Microsoft Windows* and *Mac OS X*, though some demonstrations have already been made with smart phones. The system is designed for Internet access, instead of a local network, so its large bandwidth requirements is a great disadvantage. It is estimated that standard definition games will require a 1,5 Mbps connection, while for high definition games this value rises to around 4 or 5 Mbps. Another drawback is its high computational requirements for desktop platforms, such as a dual core processor [OnLive, 2009b]. These characteristics limit accessibility due to costs not only to acquire the appropriate equipment but also rent bandwidth for the transmission of data.

3. Cloud computing

Cloud computing is a paradigm that is still under development and its definitions and terms will likely change over time. According to [Mell and Grance, 2009], the cloud is defined as a type of on demand network access to a set of computational resources that can be delivered quickly, with minimal managerial effort. [Mell and Grance, 2009] also define a set of essential features, service models and ways of providing the computing model promoted by the clouds. Sections 3.1 to 3.3 describes these features in more detail.

3.1. Service models

This section describes the service models in the cloud computing paradigm, namely, how this paradigm can be applied in practice.

- **Software as service (*SaaS*):** In this model, the client uses the provider's applications running on a cloud infrastructure. Applications can be accessed by different devices, such as a web browser but the user has no access or control to the infrastructure of the cloud (network, servers, operating system, among others). An example of software as a service customers are webmail clients.

- **Platform as a service (*PaaS*):** this model allows the customer to use the cloud infrastructure to run their own applications. These applications should be developed using tools and programming languages supported by the provider. As in the *SaaS* model, the consumers does not have control of the infrastructure of the cloud, but has control over their installed applications and can also configure the hosting environment as required.

- **Infrastructure as a service (*IaaS*):** In the model, the client uses the infrastructure of the cloud according to their particular needs. The client is able to install and run any software, including operating systems and other diverse applications. The control and management of the physical structure of the cloud is not available to the client, but he can control his own private virtual environment including operating systems, data storage, and installed applications, among others. It is also possible to have limited control of certain networking components, such as choosing a firewall system, for example.

3.2. Essential characteristics

This section describes the elements considered essential in the paradigm of cloud computing.

- **Self-service model for use of resources:** The resources of the cloud should be available to consumers automatically, according to their needs, without the need for human interaction by the service provider.

- **Provision of services via the network:** The services are offered through a network and should be open to heterogeneous devices such as mobile phones, laptops and PDAs

- **Resources pooling:** The resources (such as disk storage, processing time, memory, virtual machines and bandwidth) of the provider are arranged to serve multiple customers simultaneously and their physical and virtual resources are allocated dynamically according with consumer demand. The location of these resources cannot be controlled by the user, except for a higher level of abstraction, where he can choose to use resources from a particular state or country, for instance.

- **Elasticity of the system:** The system capabilities can be quickly increased or decreased to meet the requests of the client. For consumers, the cloud computing resources often seem limitless and can be purchased in any quantity.

- **Monitoring of resource use:** The computer systems in the clouds can control and optimize their resources by measuring the consumption of specific services such as storage or processing time. Monitoring the use of resources is fundamental to provide transparency to the provider and consumer.

3.3. Deployment models

A cloud computing structure may be available to consumers in different ways. This section describes how this provision can be made.

- **Private:** The cloud's infrastructure is operated for a single organization and can be managed by the organization itself or a third party. In this case, the infrastructure can be physically on the premises or in an external location.

- **Community:** The cloud's infrastructure is shared among various organizations and is designated for a community with common interests and standards, such as technical requirements and security policies, for example.

- **Public:** A public cloud structure is accessible to the general public or a large industrial group. This type of cloud belongs to an organization that sells cloud services to consumers.

- **Hybrid:** A combination of cloud infrastructures of other kinds, private, community or public. While each entity remains with its own identity, they are bound by standards to promote data portability between applications and interoperability of clouds infrastructures.

4. The proposed architecture

This section presents an overview of the proposed architecture, divided in three components that interact to enable the implementation of the game. First, we present the system architecture and then details for setting up sessions and games running through a cloud computing infrastructure are described.

4.1. System architecture

The system was organized in an architecture composed of three basic components: the client, the cloud manager and the host manager. These components are described with more details in the following sections and Figure 1 displays an overview of this architecture and how the components communicate.

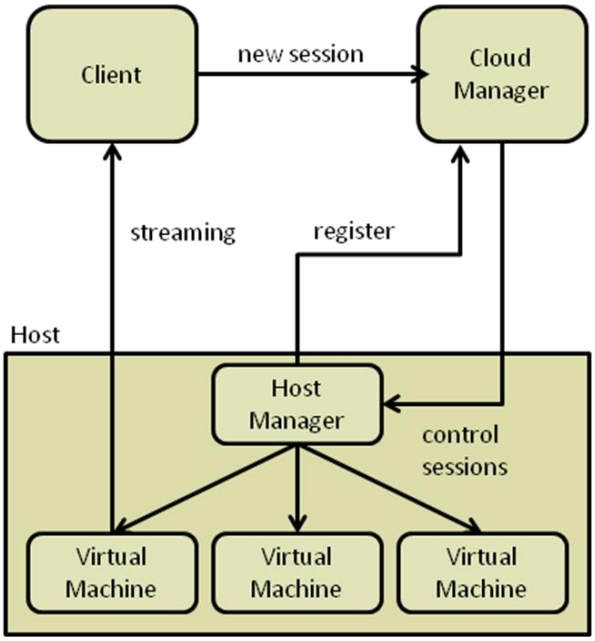

Figure 1 - Architecture overview.

4.1.1. Client

The client is a software system running on a device hosted by the user to access the game. This device can be a mobile phone, a personal computer, or any other device capable of running the client software and connect to a wired or wireless network.

The request to initiate a new game is made by the client to the cloud manager, a component that must be available and visible within the network. This request is performed through an HTTP call made to the cloud manager. The cloud manager will find from its list of available and managed hosts, one node that has resources and capacity required to run the game and provide the client with identification data of the virtual machine that will process its requests.

Since the communication between the client and the game server has been established, they exchange messages directly without going through other system components. The client handles user input and sends the data over the network. The game is executed remotely on a virtual machine and acts as a normal desktop or mobile game, carrying out physics processing, artificial intelligence and so on. The differences appear only in user's input and graphical output.

Reading data from user's input, such as keystrokes and mouse coordinates and clicks, is done locally on the client. These data are collected on the client device and transmitted to the server through the network, then are processed according to the behavior programmed to run the game at the moment. This allows an interaction between the client devices and server for implementing the game.

During the execution of a game, the graphical output is used to show the player the current state of the game. This output is processed by the game server without the usage of the server graphics device. The output image, a game frame, is compressed and sent over the network to be displayed to the user. The client's device must be able to receive several images per second over the network and display them in sequence to the player, giving the impression of smooth movement.

In the experiments carried out so far, we tried to work with complex backgrounds images, thus creating worst case scenario in order to simulate the behavior of more elaborate games. Figure 2 shows an example of a game running on the *Android* operating system, with a resolution of 320x240 pixels. The background photo was used to increase the complexity of the image to be sent over the network, causing tests to occur always at the worst case scenario or near it. Figure 3 shows the desktop client of the same game running under Windows Vista with Java 6.

Figure 2 - Game screen on a mobile phone with the *Android* operating system.

Figure 3 - Game screen on a desktop client running on Windows Vista.

The possibility of performing all processing of the game on a remote server, including physics and graphics processing is one of the most important points of this system. This allows games to be developed considering only the server's system requirements, i.e., the customer's equipment is no longer a factor to be considered within the limits of resource use in the game and now more cost effective use can be made of state of art graphics hardware if installed and shared amongst multiple clients. Under the present situation, the games should be developed to match the target audience equipment, where graphics realism and performance must be balanced to generate the best outcome for the player. With games running on a server, the client devices need only meet the basic requirements, like being able to connect to a high speed network and display streaming video. This allows both the implementation of simpler games as well as games with fairly advanced graphics effects, since this additional processing load will never be passed along to the game client.

The game client is a simple piece of software that's only function is to display the frames received over the network, as streaming video player, and send user's input to be handled on the server. This facilitates the creation of clients for several platforms since the game code remains unchanged. Different clients can be created for personal computers and mobile devices with processing power and varied APIs, for example. We developed two versions of this test game, both using the *Java* platform: a desktop and one for mobile devices using version 1.5 of *Android SDK* [Lecheta, 2010].

4.1.2. Cloud manager

The cloud manager is a component responsible for managing the infrastructure of the cloud. It is this component that performs the first communication between the client and a virtual machine controlled by the host manager, allowing a new game to be initiated.

Initially, available host managers in the network register themselves with the cloud manager, indicating that they are ready to run new games. From this moment, when a new client appears, the communication between him/her and the virtual machine that runs the game is mediated by the cloud manager.

When a client wants to play, the cloud manager searches among its registered host managers the one who has availability to provide the new game. When a physical resource is found, a command is sent to boot up an appropriate virtual machine and begin a session, while the address of the virtual machine is returned to the client, thus permitting communication between the two extremes of the system. All future communication will be done directly between the game server and the client, without going through the cloud manager. At the end of the gaming session, the cloud manager is signaled by the client and the corresponding session is closed and virtual machine shut down.

The cloud manager is also responsible for registering and providing information about the sections in progress, including data on which virtual machines are running and customer's usage time. Periodically, the manager of the cloud sends requests for state information from the host managers and they respond using a data structure in JSON *(JavaScript Object Notation)* format.

This component was developed as a *Java Servlet* with a *MySQL* database [Oracle 2010b] for storage of settings and system status. The cloud manager runs on an *Apache Tomcat* server.

4.1.3. Host manager

The host manager is the system component that manages virtual machines responsible for running the games available to the clients.

This component is registered in the cloud manager, indicating that it is available to run games requested by a client. In a heterogeneous environment, several different game types may be available and each host manager may be able to provide only a subset of these games. Thus, when registering the component in the system must identify which games a host is capable of providing.

The registration of a host managers is carried out via a HTTP call to the cloud manager. At this point, it reports its IP address to the cloud manager, so it can be accessed later. New host managers can be added to the system at any time according to system requirements.

Once registered, the host manager becomes available to provide gaming to new customers. Upon receiving a new customer to be serviced, a new virtual machine to run a particular game is initiated. This virtual machine will function as the game server, communicating directly with the customer without relying further on the host manager. This is done through the exchange of IP addresses between the client and server mediated by the host manager. Once a component knows the IP of the other, they begin to communicate directly.

A host manager can be classified as one of two types: homogeneous or heterogeneous. It is said homogeneous if all the virtual machines managed by it can serve only a single game type. In this case, the host manager is able to meet only the demands of this single game. Heterogeneous host managers are the ones able to meet requests from two or more game types, that is, it manages two or more distinct groups of virtual machines and each group is able to perform a different type of game.

Virtual machines managed by this component are responsible for running the game servers. These virtual machines run the *Linux* operating system and are managed through Oracle's general purpose full virtualizer for x86 processors, *Virtual Box*. Just as the cloud manager, the host manager was also developed using the *Java servlets* technology.

4.2. Stages for running a game

The game execution can be considered as the elapsed time since the player makes the request to start a new game session until the time when this session ends. During this time, various actions are performed so that the game can be started and run. The following is an overview of steps for running a game:

1. Client asks the cloud manager the initialization of a new game;
2. Cloud manager identifies the hosts that can satisfy the request;
3. Cloud manager requests a new session from the chosen host manager*;*
4. Host manager starts a virtual machine capable of running the requested game;
5. The number of customers can be serviced by the host manager is decreased by one;
6. Host manager obtains the IP address of the virtual machine and returns to the cloud manager;
7. Cloud manager passes the IP address for the client;
8. Client establishes communication with the game server in the virtual machine;
9. Client sends keyboard commands to control the game for the server;
10. Server processes the user input and runs the game logic;
11. Server processes the graphical output of the game and sends it via the network to the client;
12. Client displays the graphical output of the game for the player;
13. Repeat steps 9-12 until the client requests the closure of the game;
14. Host manager shall notify the cloud manager of the termination of the game;
15. Host manager closes the virtual machine;
16. The amount of clients that can be serviced by the host manager is incremented by one.

The following sections describe in more detail some of the above steps.

4.2.1. New game request

When a client wants to start a new game, it makes this request to the cloud manager. Despite being a virtual machine that will actually run the game, the customer needs not to have any knowledge of internal system components to play. Similarly, a user's webmail client does not need to know where his messages are stored or which machine will serve you. He needs only know how to access the infrastructure of your mail server, in this case, an IP address of the service.

The game client needs to know a way to access the cloud manager. In the proposed system, the

infrastructure of the cloud is contained in a local network and cloud manager has a fixed IP address in this network location. Thus, the client software can access it directly, without additional configuration. Despite the ease that the use of a local network brings, the transfer of this infrastructure for an open environment like the internet is quite simple. Instead of using the feature of a fixed IP address, the cloud manager could be accessible via a URL that would be translated by a DNS service.

Once the customer has access to the cloud manager's services, he can order the start of a new game session. It is important to note that many different games may be available to the client. By its choice, the cloud manager will detect which host manager is best able to process the request of the new game.

4.2.2. Establishing communication between a client and a server through the cloud

The client accesses the cloud manager to request the initiation of a new game, but this is not who will run the game. The cloud manager first searches among host managers that are able to run the game and returns to the client the IP address of the virtual machine responsible for processing the game. After these steps, the client and server are able to communicate directly without going through other components of the system, starting the game play.

4.2.3. Execution of the game through the exchange of data packets

All game logic is handled remotely by a server running on a virtual machine. The customer is only responsible for capturing the user's keyboard input and display screens of the game. Thus, much of the processing is performed on the server and only a small fraction is left to the client. To work together, we need these two components to exchange data constantly.

With the IP addresses known after a session has started, the client and server initiate a communication channel via sockets [Donahoo and Calvert, 2001]. Currently, TCP is used, but the authors propose further tests using the UDP protocol to verify a possible performance gain.

The sequence of steps executed during the game is listed below. These steps are repeated continuously until the game session is closed.

1. Client sends user input through the network;
2. The network server receives the user input;
3. Server processes the game's logic;
4. Server generates game's graphical output in memory and sends it across the network;
5. Customer receives graphical output from the network;
6. Client displays the graphical output to the user.

When communication between the client and server is broken or there is a request to logoff from the client, the game ends. Thus, the virtual machine is disconnected from the server that then returns to be available to host new customers.

4.3. Control game sessions' state

The control of game sessions' state is done to ensure that system resources remain allocated only as needed, being released when they no longer be used.

Game clients communicates regularly with the cloud manager, in a configurable time interval, reporting their status. While playing, the resources allocated for this client remain active. When a customer terminates the game or is disconnected, a message with the request for disconnection is sent to the cloud manager (the first case, a request to disconnect) or status reports are no longer received (in the second case, inadvertent disconnection).

The cloud manager maintains a list of active sessions and updates its state whenever it receives the report from the client. Upon receiving a request to close the session or when stipulated time limit has been exceeded, the component sends a message to the host manager stating that the game session has ended and the resources of the virtual machine can be released and become available for provide a new session.

5. Results

This section describes the results of this implementation work. The results were evaluated in relation to the use of network bandwidth required to run an interactive game, the response time to commands sent by the client and the amount of frames by the client per second achieved.

The bandwidth usage was analyzed in relation to the number of bytes sent from server to the client per second, for displaying game screens. The server sends each ready game screen for the customer divided into packets of 512 *bytes.* Using the screen resolution of 320x240 *pixels*, each image sent is divided into 28 packets of this size, resulting in 14 *kilobytes* per image.

Achieving an optimum performance of 30 frames per second, requires send across the network medium 420 *kilobytes* per second (or 3.3 Mb/s). This is a quite acceptable value is taken into account the capacity of a local network, as shown in Figure 4.

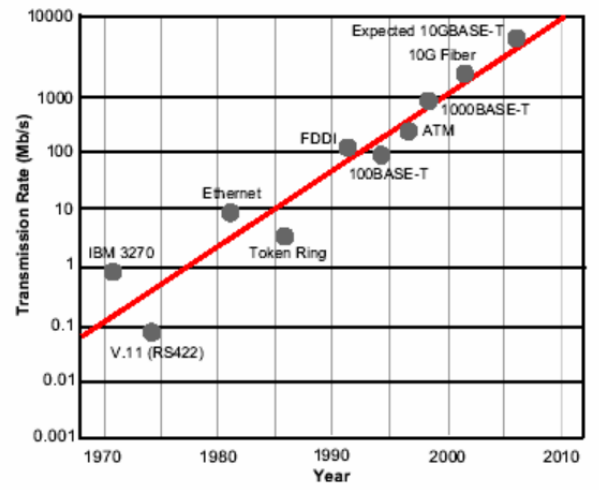

Figure 4 - Growth rates of investments in transmission networks [Siemon, 2010].

In prototypes developed, using Wi-Fi network, the desktop version averaged 20 frames per second received over the network. In empirical tests, the results were satisfactory, giving a feeling of fluidity in the animations in the game. In the mobile client, the transmission of frames from the network reached lower values, being close to 10 frames per second, so some improvements are still required to be done.

Besides the number of frames per second that the game is capable of displaying, system response time to user actions is of great importance for a good result during the sessions. The response time is the time that a user input command is sent over the network, this command is processed by the server according to the game's logic and the result is displayed to the customer. The desktop client received an average of 30 milliseconds of delay in response time, while the mobile version showed results in between 800 and 1000 milliseconds.

The value presented here can be improved in future by exploring further the specific features of the *Android* platform and seeking more efficient ways to transmit data. All tests were performed based on packet switching network using the reliable TCP protocol. It is believed that the use of UDP protocol, which does not guarantee reliability, will lead better results regarding the number of frames per second transmitted on the network and the response time.

6. Conclusion and future work

This work has shown that a cloud computing infrastructure can be used in real-time interactive applications, including digital games. The system allows the execution of digital games in a format that separates the reading keyboard and display graphical output of processing the heavier parts, such as generation of graphical output and processing of physics and artificial intelligence.

The focus of this study was restricted to the processing of user input, the execution of the game logic and display graphics output. The treatment of sounds and their transmission across the network has not been addressed, remaining as an avenue for future work.

Some issues relating to the network are currently under investigation and alternatives are being studied. Currently, virtual machines have a static IP address, obtained through a table configured on the router. The inclusion of a server IP address leases can make this task automatic.

With respect to communication, it is necessary to define security mechanisms for communication between the manager of the cloud and the host manager and also between the client and server. This security is necessary to prevent, for example, a client sending messages to the game server with another client.

Finally, it is suggested to create tests using the UDP network protocol, which may not provide reliability benefits with respect to high amounts of data traffic on the network but may reduce the delay of communication.

Also, lower data traffic over the network should be achieved with the usage of light-weight data compression algorithms before sending the frames to the client. The trade-off between required CPU power to uncompress these images and the bandwidth usage should be analyzed in order to find out the best approach.

References

Armbrust, M., Fox, A., Griffith, R., Joseph, A., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., e Zaharia, M., 2009. Above the Clouds: A Berkeley View of Cloud Computing. Disponível em: <http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.pdf> [Accessed 03 jul 2010].

Buyya, R., Yeo, C., Venugopal, S., 2008. Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities. Disponível em: <http://www.buyya.com/papers/hpcc2008_keynote_cloudcomputing.pdf> [Accessed 04 jul 2010].

Delic, K., Walker, M., 2008. Emergence of The Academic Computing Clouds. Disponível em: <http://www.acm.org/ubiquity/volume_9/v9i31_delic.html> [Accessed 04 jul 2010].

Donahoo, M., Calvert, K., 2001. TCP/IP Sockets in C: Practical Guide for Programmers, 1º edição.

ESA – Entertainment Software Association, 2010. Video Games in the 21st Century – The 2010 Report. Available

from: < http://theesa.com/facts/pdfs/VideoGames21stCentury_2010.pdf > [Accessed 15 sep 2010].

Lecheta, R., 2010. Google Android, 2º edição.

Manfe, T., 2009. VirtualBox Factory: HowTo Automate VBox Provisioning in a Cloud. Disponível em <http://blogs.sun.com/partnertech/entry/automating_virtualbox_provisioning_for_a> [Accessed 04 jul 2010].

Mell, P., Grance, T., 2009. The NIST Definition of Cloud Computing, National Institute of Standards and Technology, Information Technology Laboratory. Disponível em: < http://csrc.nist.gov/groups/SNS/cloud-computing/cloud-def-v15.doc> [Accessed 16 jun 2010].

Nurmi, D., Wolski, R., Grzegorczyk, C., Obertelli, G., Soman, S., Youseff, L., Zagorodnov, D., 2008. The Eucalyptus Open-source Cloud-computing System. Disponível em <http://www.cca08.org/papers/Paper32-Daniel-Nurmi.pdf> [Accessed 04 jul 2010].

OnLive, 2009a. OnLive. Disponível em <http://www.onlive.com/> [Accessed 03 jul 2010].

OnLive, 2009b. OnLive Techinical FAQ. Disponível em <http://www.onlive.com/support/performance#performance_1> [Accessed 03 jul 2010].

Taurion, C., 2009. Cloud Computing - Computação em Nuvem, 1º edição.

Oracle, 2010a. Java Servlet Technology. Disponível em: <http://java.sun.com/products/servlet/> [Accessed 03 jul 2010].

Oracle, 2010b. MySQL. Disponível em: <http://www.mysql.com/?bydis_dis_index=1/> [Accessed 03 jul 2010].

Siemon, 2010. Os Ciclos de Vida do Cabeamento e as Leis das Comunicações em Redes. Disponível em <http://www.siemon.com/br/whitepapers/10G-Assurance.asp> [Accessed 04 jul 2010].

The Apache Software Foundation, 2010. Apache Tomcat. Disponível em: <http://tomcat.apache.org/> [Accessed 03 jul 2010].

Zhen, J.,2008. Five Key Challenges of Enterprise Cloud Computing. Disponível em; <http://cloudcomputing.sys-con.com/node/659288> [Accessed 04 jul 2010].