

Analyzing Google Earth application in a heterogeneous commodity cluster display wall

Ismael Arroyo¹  · Francesc Giné¹ · Concepció Roig¹ · Toni Granollers¹

Received: 10 January 2015 / Revised: 9 June 2015 / Accepted: 3 August 2015 /
Published online: 29 August 2015
© Springer Science+Business Media New York 2015

Abstract Nowadays, clusters of interconnected workstations have become a common solution for powering large composite displays, or “cluster display walls”, to visualize high resolution images. Our paper is focused on analyzing a specific cluster display wall developed by Google, named Liquid Galaxy, made up of heterogeneous commodity hardware with different degrees of heterogeneity, running master-slave (Google Earth) and client-server (Quake III Arena) multimedia applications. With this in mind, we define and test different scenarios, representing the behavior of many kinds of users. Our results show that the CPU, memory and network are good enough to execute the client-server application, while, depending on the user behavior, the external network constitutes the bottleneck of the system in Google Earth. So, the master-slave application has focused our attention. Likewise, in order to analyze the users’ point of view when interacting with Google Earth in the Liquid Galaxy, we define a new metric, named Visualization Rate (VR), which enables a relationship to be established between the user experience and the platform performance. In order to set the minimum acceptable value of the VR parameter according to users perception, we carried out different tests with real users. Then, this minimum threshold was compared with the VR value obtained from the automated benchmarking performed afterwards on clusters with different heterogeneity degrees. Finally, we analyzed the VR trend when the Liquid Galaxy is scaled from 3 up to 8 nodes in both the homogeneous and heterogeneous architectures to study the scalability of the system.

✉ Ismael Arroyo
ismael.arroyo@udl.cat

Francesc Giné
sisco@diei.udl.cat

Concepció Roig
roig@diei.udl.cat

Toni Granollers
tonig@diei.udl.cat

¹ University of Lleida, Lleida, Spain

Keywords Cluster display wall · Liquid galaxy · Master-slave · Client-server · Google Earth · Benchmarking

1 Introduction

Nowadays, new trends in information and communication technologies, storage density and increasingly sophisticated data acquisition technologies have led to an explosion in data size. In order to find patterns of knowledge to this huge data sets, new visualization methods are arising [4].

A current method for visualizing large-scale data sets consists of using a display-wall based infrastructure, in which various screens are distributed in tiles connected to a single computer using multiple video outputs. The main problem that arises when using display walls is that the images are stretched, as resolution of a single screen is resized to fit all the screens together. Furthermore, specialized video processors may be needed to manage a large video wall, raising the price of the system. To bypass this problem, a cluster-based display wall is proposed, in which each screen is served by a computer. Some examples are CAVE [5] and GeoWall [18].

Cluster-based solutions to create large displays have recently generated a lot of interest [24]. Such displays have the potential to put high-performance visualization within more users' reach. These systems consist of a number of commodity PCs that are interconnected over a LAN or via low-latency networks like the Myrinet. Thus, visualization of the images across screens in cluster display walls is carried out by synchronizing the computers to pass the data between them. Nowadays, besides the fact that conventional PCs can be equipped with powerful consumer graphics cards with multiple outputs, the availability of software packages for clusters makes setting up cluster display walls affordable. Upgrading a cluster based display wall is as easy as upgrading the individual nodes in the cluster. So, cluster-based displays are affordable, scalable in resolution and easy to maintain. This opens a wide range of new possibilities for using them in day-to-day scenarios in such areas as entertainment, finance or education.

The majority of applications executed in a cluster-based display follow two approaches [3]: *master-slave* and *client-server*. In the master-slave applications, the dataset is mirrored across all the nodes and multiple instances of a program run in parallel, one on each node, being the master who synchronizes all the nodes; whereas in the client-server approach, the server runs a different instance of the program executed by the clients, distributing appropriate data to each client node and performing the synchronization among the client nodes.

This paper analyzes the use of a specific cluster display wall hemisphere infrastructure developed by Google, named Liquid Galaxy [8], running both kinds of applications, a video-game as a client-server and the well-known Google Earth application [9], which is an example of a master-slave application. The Liquid Galaxy system is made up of eight displays, each connected to a computer node, and that provide an immersive geographic visualization. Figure 1 shows the Google Liquid Galaxy infrastructure installed in the Technological Park in the city of Lleida (Spain) [27].

Taking into account the architecture of Liquid Galaxy and its possibilities, this paper analyzes the viability of building this kind of infrastructure by adding some already-available PCs, thus decreasing its price drastically and extending its use to new groups of potential



Fig. 1 Lleida Liquid Galaxy

users. To do so, we built a new cluster display wall infrastructure with different heterogeneous configurations to identify the relationship between performance and user behavior according to the degree of heterogeneity of the system.

With this aim, the paper analyzes the main performance metrics of CPU, memory and networking by running a client-server and a master-slave application. We observe that client-server applications have an homogeneous consumption of these computing resources, independently of the user's behavior. However, master-slave applications present significant variability in resource consumption during execution depending on the user's activity. Due to that, the paper is focused in master-slave by carrying out a detailed performance analysis of Google Earth running in the Liquid Galaxy. The results show that a cluster display wall composed of conventional desktop computers provides enough CPU and memory to execute Google Earth. However, networking constitutes an important issue, as we identify user behavior as the determining factor in the bandwidth requirements. We also define a new parameter, named Visualization Rate (VR), which enables the system performance to be related to the level of satisfaction of a user of the system. In order to set the minimum threshold of the VR parameter acceptable to users, we performed tests with real users to see what their behavior was. Taking these tests into account, the minimum VR value accepted by real users was set at 30 %. This allows the performance of the system to be evaluated in terms of user satisfaction in relation to the degree of heterogeneity. Our results show that VR is directly related to the average CPU power of the system and is negatively affected by the slowest CPU. Taking the VR parameter into account, we also analyzed the user perception when the cluster display wall infrastructure is scaled from 3 up to 8 nodes, which is the standard size set by Google to create an immersive experience. The results obtained reveal that scalability is affected by bandwidth; so a maximum number of nodes to maintain the VR above an acceptable level of 30 % is established.

The paper is structured as following. Section 2 presents the background of cluster visualization systems. Section 3 describes the case study of Liquid Galaxy. In Section 4, the performance parameters of the Liquid Galaxy infrastructure built with commodity hardware are evaluated and performance issues are also identified. Section 5 analyzes user behavior in the Google Earth application in Liquid Galaxy using both homogeneous and heterogeneous systems. This section also includes scalability studies, increasing the number of nodes

in the clusters from 3 up to 8. Finally, Section 6 concludes the paper and discusses future directions.

2 Background of cluster visualization systems

Cluster-based display walls can provide cost-effective and scalable displays with high resolution and large display area, making them suitable for a wide range of applications with a requirement for high resolution. As a consequence, this has arisen the interest of the scientific community, proposing a wide range of new cluster-based display wall platforms together with software frameworks oriented towards handling the details of synchronizing and distributing the rendering tasks across these nodes [4, 21]. Current cluster display wall infrastructures include: CAVE [5], an immersive environment which simulates a cave, a room where all the walls are being projected; GeoWall [18], a set of tiled 3D displays on a solid structure making a wall; ViewDock TDW [20], a computing cluster connected to a tiled display wall used in drug discovery and Liquid Galaxy [8], a set of tiled displays forming a hemisphere to provide immersivity when running Google Earth. This fast evolution of cluster display walls, together with the commercial explosion of the thin-bezel LCD panels with built-in support for stereoscopic 3D, has presented the opportunity to build *hybrid-reality* environments, which combine the benefits of immersive virtual reality and large cluster display wall environments [28]. There are also many works focused on enabling collaboration between users through screen sharing in the same room using display walls. These include SAGE [29], a number of rendering resources connected over the network and Dynamo [17], an interactive surface that allows sharing, displaying and exchanging media with other people in the room.

As listed, there are many types of visualization systems, but depending on the configuration of the system and the communication protocol to be applied, we can differentiate the following two cluster display wall architectures [33].

- **Client-server:** In the client-server model (Fig. 2a), there are two sides of the application running on the system. The server is running the server-side application (APP-server),

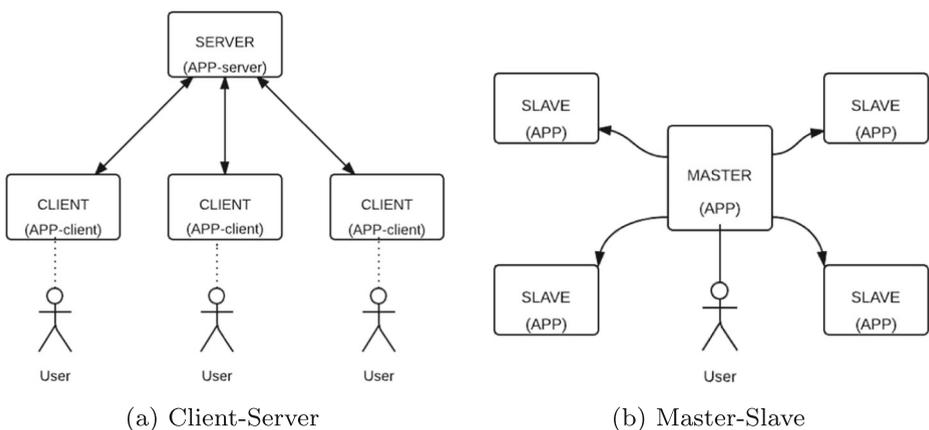


Fig. 2 Cluster display wall architectures

which is the main application, and which stores and refreshes the clients' status. Each client runs an instance of the client-side application (APP-client), which is connected to the server and modifies their status and/or some of the content of the APP-server. The user can choose one of the client nodes to interact with.

This architecture can be used to execute both interactive and non-interactive applications. Interactive applications are those that allow a user to access a client node and modify its status. When a client's status is changed, consequently, the client node will send the changes to the server making the others clients fetch these new data modified by any client. OptiPlanet [31], Equalizer [7], Chromium [14] and CGLX [6] are examples of interactive applications to be executed in a cluster display wall environment. Non-interactive applications do not usually allow any changes by the user, although, in the case that the user could make some, these would not affect either the server or the other clients. An example of a non-interactive application is Video Streaming.

- **Master-slave:** In the master-slave architecture (Fig. 2b), each node runs the same application (APP) and the master must manage the synchronization among slaves to ensure consistency. The master is the only node that accepts user input. Thus, the user only interacts with this node, and every time the master modifies its status, it sends the changes to the slaves. Applications for this kind of architecture are usually interactive, such as Google Earth.

This paper is focused on the performance analysis of both kinds of architectures built by means of a Liquid Galaxy platform with different degrees of heterogeneity and implemented with commodity PCs. To the best of our knowledge, only Nirmimesh et al. [24] have tackled the development and analysis of a cluster-display wall, named Garuda, made up of commodity PCs. In contrast to Liquid Galaxy, Garuda is a homogeneous client-server based display wall oriented towards rendering any application built using the Open Scene Graph (OSG) [26].

In addition, we are interested in finding a quantitative measure to establish a relationship between the performance issues and the real perception of a user of this kind of platform. In general, in the literature, this relationship is settled with qualitative parameters. This is the case of Ball and North [1], where they address the study of user behavior on high resolution tiled displays to be used as personal desktop computing. They focus in measuring qualitatively the interaction of users with this kind of platform, whereas we are interested in defining a performance metric to relate it to the real user's perception of a master-slave application, such as Google Earth, running on the Liquid Galaxy platform

Only few works evaluate performance based on quantitative approaches. They are mainly based on measuring the FPS (Frames per second) [24, 30, 33]. However, this measure is not suitable in our context given that the FPS can only be obtained from specific graphics cards; while our purpose is building a system with off-the-shelf hardware under a Linux environment. Thus, we propose to evaluate the hardware performance focusing in CPU consumption, which is a measure that can be easily obtained from any kind of platform.

In addition, Liu et al. [22] compare physical navigation in a wall in contrast with a virtual navigation in a desktop for data manipulation tasks by means of quantitative parameters, such as task completion or physical movements of the participants in front of the wall. Unlike this work, we are interested in obtaining a metric which gives a relation between hardware performance and user satisfaction.

In the next section, we present the characteristics of the Liquid Galaxy system and how it works under the two described architectures.

3 Liquid Galaxy system

Liquid Galaxy [8] is a cluster display wall, that started as a Google project, made up by a custom amount of computers, where every node has a single monitor. The Liquid Galaxy system is originally built to run Google Earth [9] in order to create an immersive experience for the user. Liquid Galaxy lets you navigate around the globe with its 6-axis controller, allowing you to instantly zoom in, zoom out, and turn in a completely fluid motion. You can also search and navigate to specific locations automatically using an optional touch-screen interface. However, the immersive visualization environment of Liquid Galaxy opens up this kind of system to be used in a wide range of applications that can benefit from this feature. Some examples of applications that can be run in this system are WebGL applications like Aquarium [12] or Peruse-a-rue [10], video streaming [34] and video-games like Quake III Arena [25].

Every node is connected to the same network and nodes share a distributed cache named Squid [32], included in the Liquid Galaxy repository. This is used to cache http objects for repetitive use, thus, improving throughput, as Internet data requests would be reduced.

Whenever cache is needed, the nodes must be built with Solid-State Disks (SSDs). This is imperative when the internet traffic is heavy, as it reduces significantly the time of visualization of the multimedia applications that require this technique. This requirement is very important when a lot of data is passed between nodes and low access time to the disks is required.

To provide an easier use of the cluster, configurations and scripts, which are not listed here but can be found on the Liquid Galaxy official page from Google [11], are installed and enable the Liquid Galaxy to be used as a single system, instead of different computers, in a feasible way.

The interaction with the system can be carried out with different devices depending on the application: Mouse & keyboard, 3D SpaceNavigator, Leapmotion, Myo, etc. Although this can be achieved, some technical configuration may be required in order to install these peripherals correctly. Additionally, a web administration tool is present on the Liquid Galaxy allowing external access from another computer, tablet, phone or device that has a browser.

Next subsections describe the workflow of two kinds of representative applications, where the Liquid Galaxy System is used as a client-server or master-slave architecture.

3.1 Client-server work-flow

One kind of application that run in client-server mode is video-games, not mattering if the game is programmed specifically for a Liquid Galaxy system or not, as they all rely on a server for synchronization. As a representative case, we will explain how a First Person Shooter game works on the Liquid Galaxy.

These applications have two types of nodes: server and clients. The server can be a computer located in the same or different network or via Internet, but also the same node can handle server and client applications. There is only one server, which runs a server-side application of the game on which it is stored information about the clients connected to it. This information may contain data about the players as position, camera position, name, weapon, team and also events happening in the game like team scores, time limit, etc.

The clients run a client-side application that has all the assets to be visualized, updating the different elements of the game based on events received by the server. Clients can change

the server data, which will provoke the server to broadcast that information to suitable clients (e.g. position of the players in the team to the radar).

The workflow of this kind of application running under the Liquid Galaxy system, depicted in Fig. 3, has the following steps:

- (1) Whenever there is a change in the clients (such as movement, team change or item pickup), an event is prepared to be sent to the server.
- (2) The server node receives the events and updates its status.
- (3) The server sends only the suitable information that has been updated to the other clients.
- (4) Every client receives the information and updates their game status to the new changes to be synchronized.

Most of the video-games that run under this architecture follow the same procedure. In this paper, we will analyze the Quake III Arena as a specific and representative case of a First Person Shooter game.

3.2 Master-slave work-flow

Figure 4 depicts the workflow of the Liquid Galaxy while running Google Earth. We can see that each node runs the Google Earth application (GE) and the user only interacts with the master node by means of a 3D mouse. Each movement of the mouse makes the master node launch a synchronization protocol that consists of the following steps, indicated in Fig. 4:

- (1) The master node captures the coordinates (Coords) of the position in Google Earth indicated by the user. These coordinates are codified in an UDP packet, named ViewSync, which contains the following information from the view in the application: counter, latitude, longitude, altitude, heading, tilt, roll, time start, time end and planet name.

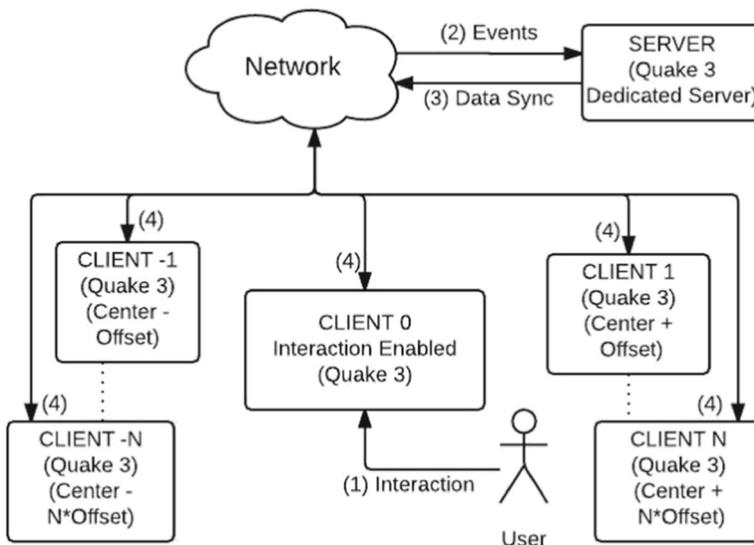


Fig. 3 Liquid Galaxy architecture running Quake III Arena

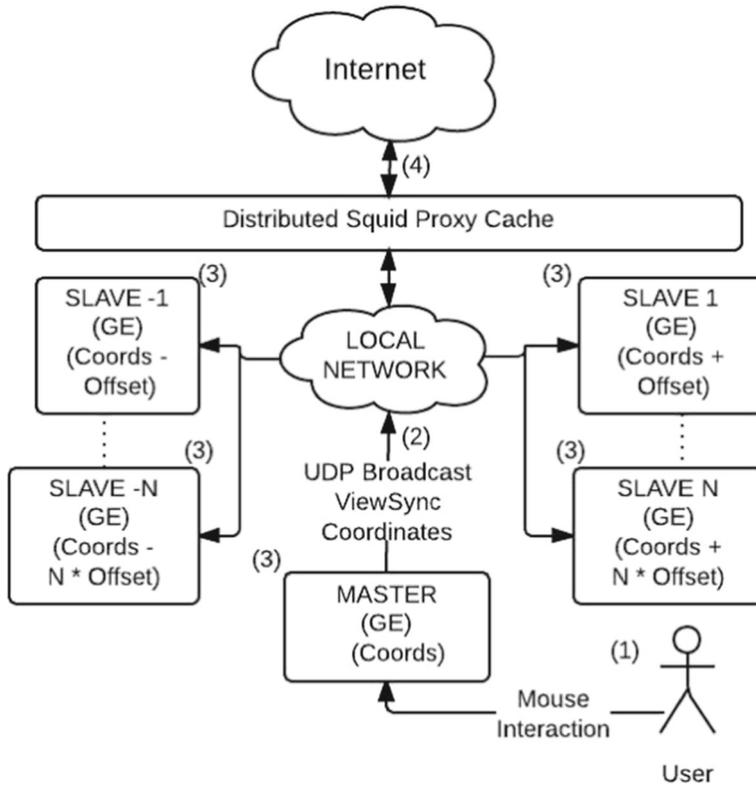


Fig. 4 Liquid Galaxy architecture running Google Earth

- (2) When the master's view is moved, it sends ViewSync packets to broadcast with the objective of sending it to all the nodes in the network.
- (3) Every slave node has a configuration file, which holds an offset from the original master's view. When the slaves receive ViewSync packets from the master, they calculate and adjust their relative view by adding the local offset.
- (4) Every node accesses Internet with its own coordinates to download the required data (maps, imagery, 3D layer, etc.) independently from the other nodes.

4 Performance evaluation

Performance of the Liquid Galaxy system running a client-server and a master-slave application was evaluated under the following parameters: CPU, RAM and network traffic. The experimentation platform is a homogeneous cluster display wall made up of 3 nodes. Each node is composed of an Intel Core i5 3GHz, with 2x4GB RAM 1600 MHz, SSD 128GB, NVidia GT620 and a 32" screen. Note that we used a size of 3 nodes because the width of three 32" displays, the size used in this experimentation, covers the human 60 horizontal viewing angle from a distance of 1m [15].

This platform was built with off-the-shelf hardware to evaluate the viability of using general-purpose platforms to visualize high-resolution images in a cluster-based way.

4.1 Quake III arena video-game

Quake III Arena was studied as a representative case of a client-server application in order to analyze the performance issues while running it on Liquid Galaxy. In this case, the server is always transmitting the position of every player to the others, no matter how far they are. Thus, all the synchronization is performed only between the server and the clients, nothing is transmitted between clients.

The procedure to provide an immersive experience with Quake III in the Liquid Galaxy was to connect all the nodes to a local server and then set the middle node to be the actual player, while the other nodes are just spectators of that player with a view offset depending on the physical position inside the cluster. The server was set to local in order to avoid the high latency of Internet network.

The initial test had a duration of 60s and was performed in the map *Q3DM6* with 5 players being controlled by the game, also called bots, and a Liquid Galaxy system of 3 nodes, which is translated into a real player with 2 spectators.

4.1.1 Experimental results

The analysis of CPU, memory and network traffic consumption obtained for the Quake III Arena game shown the same results for all the client nodes. Thus, we are going to present the obtained results for the server node and only one of the clients.

Regarding to the percentage of CPU usage, it can be appreciated that in Fig. 5 there is very low CPU usage on the server and the clients. We can see as the server has slightly higher CPU consumption than the clients, given that it has to broadcast all the events to the clients. Although the CPU load will be proportional to the number of players, such as the studies of Barri et al. [2] have stated, it will never represent a significant amount of CPU load. On the other hand, the client CPU load has low consumption independently of the number of players.

For the Memory usage, Fig. 6 shows that the application loads all of its assets when joining the game and, from then, the memory remains constant throughout the game.

As for the Network traffic, the test was performed placing the server inside the same LAN, so no external traffic is involved. This also means that Squid cache is unused, given that any data is downloaded from the Internet. As can be seen in Fig. 7, the traffic is purely

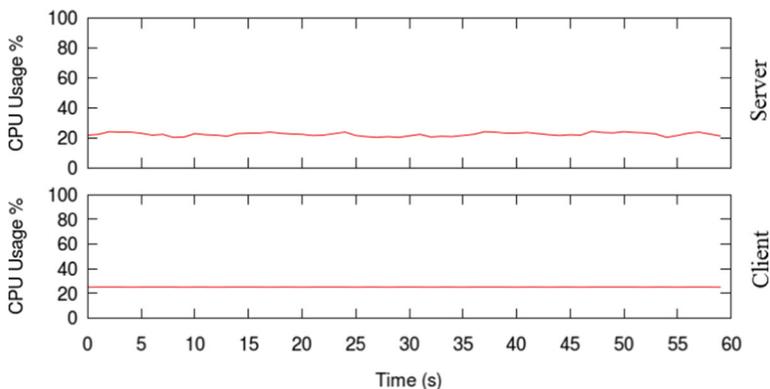


Fig. 5 Percentage of CPU usage - Quake

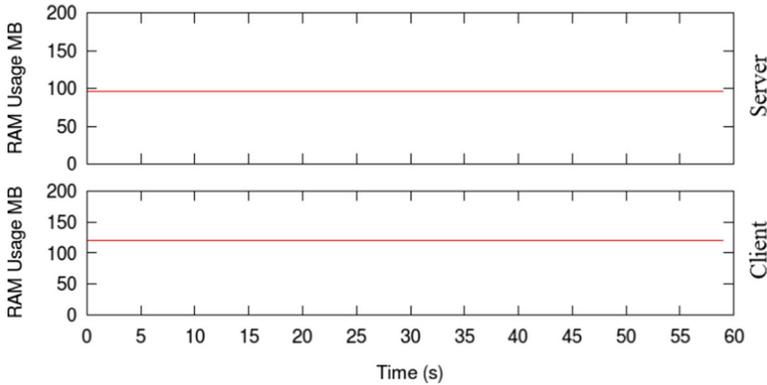


Fig. 6 Memory usage - Quake

synchronization without high-weight packets being sent. This means that the bandwidth of the connection is not as important as the latency or better known as ping between server and clients.

The obtained results show that a medium-quality hardware system is enough for this kind of application, as they have an homogeneous and low consumption of CPU, memory and network traffic and, so, there is no sign of any problem with the visualization system. Note, that other client-server applications tested in this experimentation, such as Peruse-arue (a modified version of Google Street View for Liquid Galaxy), behave the same way. According to this, the remaining of the paper is focused on the study of Google Earth as the representative case of master-slave applications.

4.2 Google Earth application

Experimentation was carried out to measure and categorize the overhead produced by the use of Google Earth, a master-slave application, in Liquid Galaxy in order to show the resource requirements in normal displacements around the globe.

For the application under study, the requirements for computing resources vary with the environment to be visualized. Thus, in order to have different environments to test, we

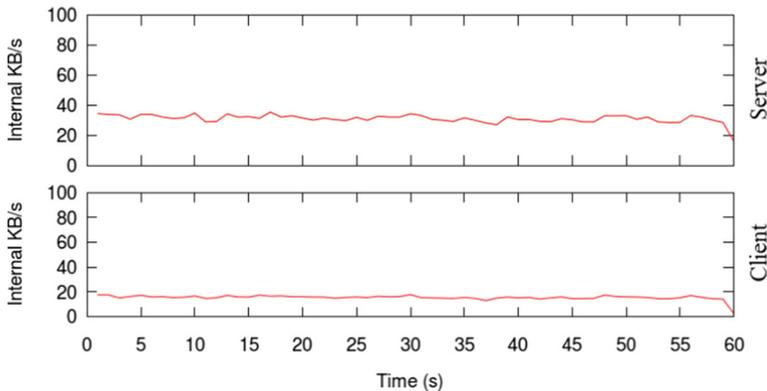


Fig. 7 Internal networking - Quake

measured the following three environments, illustrated in Fig. 8, with significant differences in the number of high-defined 3D buildings to be visualized:

- *City (Barcelona, Spain)*: A high-density place made up of 3D buildings.
- *Town (Horsens, Denmark)*: A low-to medium-density place that combines zones with a few 3D buildings and areas without them.
- *Desert (Sahara, NW Sudan)*: A low-resolution area without any 3D buildings to be loaded.

In each environment, or tour, the user moves throughout eight specific points of interest with a previously-established time intervals between consecutive jumps. We distinguish the following two intervals:

- Short time jumps: the time interval was set at 30 seconds.
- Long time jumps: the time interval was set at 60 seconds.

A wide range of time intervals was tested and the minimum interval chosen was 30 seconds as this is the minimum amount that allows us to differentiate the performance across all the tours in our current system configuration. Also, a lower interval does not allow all the requested data to download fully from the Internet. The interval of 60 seconds allows us to see when the system is in a full idle position.

Additionally, we carried out the experimentation with the Squid cache turned on and also off to evaluate its effect on the performance of the system.

Under the presented experimentation framework, the key computing resources, CPU, memory and networking, were monitored using the Top and Tshark (Command-line based Wireshark) tools.

4.2.1 Experimental results

Both the master and slave nodes were monitored in our experimentation. The results obtained showed that the master and the slave nodes behave the same way. Thus, only the results for the master node are presented.

Figure 9 shows the percentage of CPU usage for the three described environments and both timing jumps having the Squid on. In both cases, a clear difference between the three environments can be seen. In the case of the desert tour, there were many idle intervals because it lacked 3D buildings and had a low-resolution imagery. This made the time interval between jumps to be long enough to allow the images to load fully. In the other environments, the city and town, similar behavior can be seen in the CPU usage, because both contained 3D buildings. However, due to the higher density of 3D buildings in the city, the system needed more CPU to render all the visual data, giving higher peaks in the CPU use. According to the CPU results, it can be seen that there is no need for the nodes to have



(a) City tour



(b) Town tour



(c) Desert tour

Fig. 8 Images of the three tested environments

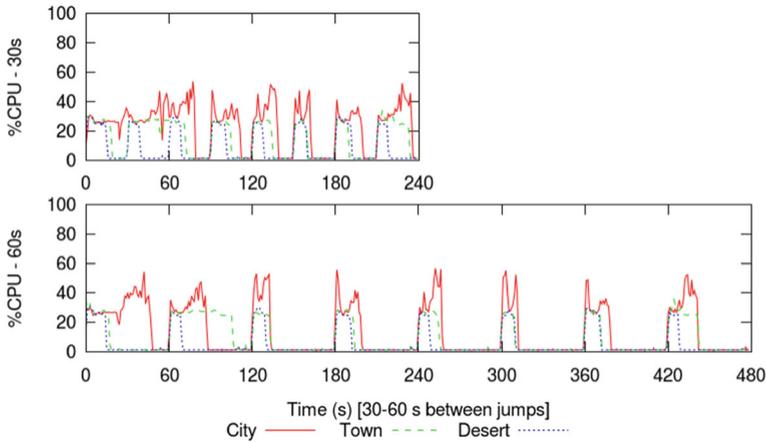


Fig. 9 Percentage of CPU usage - Squid ON

a very high-performance CPU, given that the average usage was below 40 % in all cases. Nevertheless, the time between jumps is determinant for fully visualizing the images of the environment. From now on, the results for the 60-seconds test are presented because they permit better visualization of the differences between the tested environments.

Another parameter studied was the memory usage throughout the test. This is shown in Fig. 10. All tests proved that the RAM always increases moderately regardless of the environment and the timing jump. Memory will always be reserved at the same rate until the limit established by the application is reached (in this case, 1500 MB), at which point the OS will free memory pages. This limit is always less than the total memory available.

A key performance parameter is the network usage because Google Earth has a heavy data request from the Internet and many internal synchronization packets per second. Figure 11 presents the network graph measured in kilobytes/s. It shows the external traffic (the data downloaded from the Internet) and the Squid internal traffic (that shows how data is passed between nodes). From the results of both graphs in Fig. 11, it can be observed that, despite having the Squid cache activated, we can still perceive that data was still downloaded externally whenever a request was made to change position. This happened because Squid only has an average of 20 % cache hits, leading the nodes to download new data when the next place to visit had not been cached. We can also note that the desert tour had a lower data request because it had no 3D buildings and only low-resolution maps to download. There was also a common starting peak in both graphs and all the tours. This initial peak

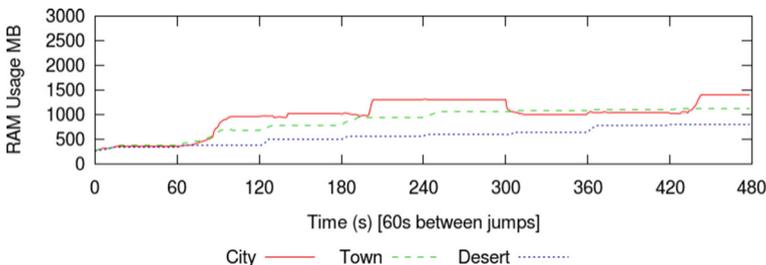


Fig. 10 Memory usage - Squid ON

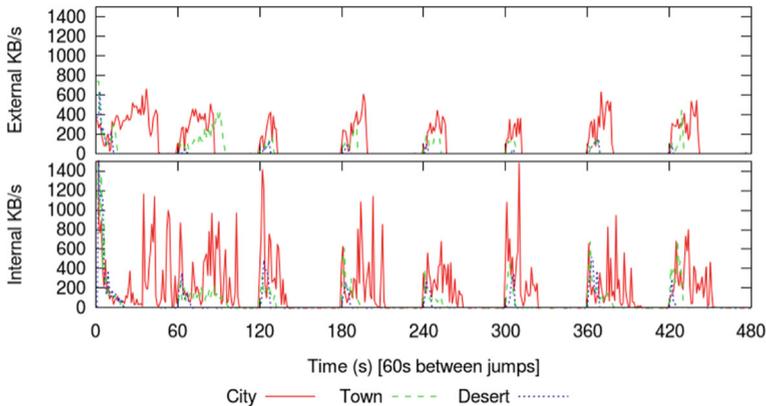


Fig. 11 External and internal networking - Squid ON

appeared because the tour started with a general view of the Earth before moving to the first coordinate and, thus all the data had to be loaded until the first position was reached.

To help us to understand how much the proxy cache influences the system, we performed the same tests with Squid disabled. Figure 12 shows the external networking when Squid was disabled. If we compare it with the external traffic when Squid was enabled (Fig. 11 top), at first glance there seems to be no difference, although the download time is seen to be somewhat longer. This is very important, because it tells us that when Squid was enabled, it shortens the download time from the Internet, which is where we detected a bottleneck.

Figure 13 shows the CPU usage when Squid was disabled. If we compare this with the same case with Squid enabled (Fig. 9 bottom), we can see that there was longer activity of the CPU than when Squid was enabled, but with a lower load. This was because the required data had to be downloaded from the Internet without the help of Squid, as we saw in the networking graph. This means that the data downloaded slower and the CPU was rendering all the data until the download had been completed, meaning lower but longer CPU usage.

The behavior of memory performance was the same when Squid was enabled as when it was off, so there is no need to present that graph.

The performance results obtained in this section reveal that the CPU, memory and local network were powerful enough to visualize images with different densities of 3D buildings; while the external network bandwidth was the real bottleneck of the system. Therefore, user

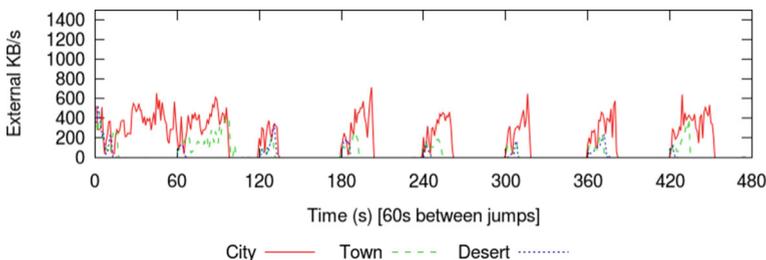


Fig. 12 External networking - Squid Off

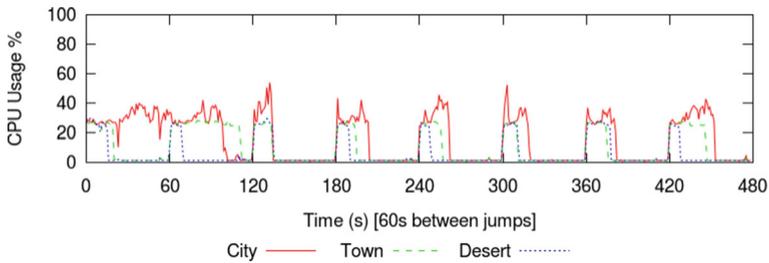


Fig. 13 Percentage of CPU usage - Squid Off

navigation behavior constitutes the determinant factor for being able to load all the required data. So, the next section analyzes the behavior of real users in this kind of system.

5 User experience related to performance

With the evaluation of the performance based on the elements of system configuration, we were able to verify that the cluster, built with commodity hardware, had enough computing capacity to run and visualize Google Earth. However, the user is at the other side of the system, so, an additional aspect that is important to consider is what the user experience is, related to performance, when using this kind of application in a cluster visualization system.

User eXperience (UX) is a widely-used term that deals with the person's perceptions of, and responses to, use and/or anticipated use of a product, system or service [16]. UX is a complex term mainly studied in the Human-Computer Interaction (HCI) and Human Factors fields as a consequence of a user's internal state (predispositions, expectations, needs, motivation, mood, etc.), the characteristics of the designed system (e.g. complexity, purpose, usability, functionality, etc.) and the context (or the environment) within which the interaction occurs (e.g. organizational/social setting, meaningfulness of the activity, willingness of use, etc. [13]). Therefore, as related researchers assert, evaluating UX is no easy task and frequently "traditional" usability (the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specific context of use [16]) evaluation techniques are used to measure UX. The focus of the present paper is not the evaluation of the whole UX, but is centered on knowing some aspects of the UX usability attributes of our system. For this reason, we propose a new quantitative metric to assess effectiveness and efficiency usability attributes and, for the satisfaction attribute, we will carry out a non-formal user test evaluation (also known as "guerrilla HCI" [23] or "Discount usability testing" [19]). According to this, we analyze the minimum satisfaction threshold of real users in relation to this new metric and discuss its evolution according to different configurations of the Liquid Galaxy system.

Other works of the literature that measure performance quantitatively in multimedia applications are mainly based on the FPS (Frames per Second). However, as Nvidia graphics drivers do not allow monitoring this parameter under average means, FPS is not a suitable measure in our environment. In order to allow FPS to be monitored in Linux, the system had to have a Nvidia Quadro graphics card installed (asked directly to Nvidia), in which it was not the case of our system as we built up a commodity hardware cluster. So, in order to relate the hardware performance metrics analyzed with the user experience through this paper, we should be able to know when the current view of Google Earth has been loaded

completely. The two main parameters that could give this information were CPU load and network traffic. We found a relation between these two parameters: the data being downloaded is processed by the CPU, thus, when the CPU stops working, it means that the images have been loaded. Because of this, we defined the new quantitative measure, named Visualization Rate (VR), based on the CPU usage.

5.1 Visualization rate (VR)

Taking into account that the CPU usage tells us when the visualization items have been loaded, we defined the new parameter, called Visualization Rate (VR), as a CPU usage percentage to determine indirectly how satisfied a user is with the visualization and waiting times of the application running on the system being used. To calculate the VR, we looked at when the CPU usage dropped below 5 %, (the CPU is considered to be idle below 5 %) in relation to the total time of the test.

According to this, VR was calculated as the average CPU idle time for a cluster of n nodes by means of the following equation:

$$VR = \frac{100}{n} \sum_{i=1}^n \frac{T_idle_i}{T_total}, \quad (1)$$

where T_idle_i is the amount of time when the CPU is idle in node i and T_total is the total time of the test. Note that when VR is near 100 %, it denotes a high visualization rate and so, good user perception as images are fully visualized, while having VR equal to 0 % means that the data has not been fully loaded.

5.2 Discount usability tests

The visualization time is a subjective parameter, which means that the minimum acceptable VR value depends on each user. So, in order to obtain this minimum acceptable VR value, we carried out a discount usability test with 86 volunteers. They were visitors to the *fair Euskal Encounter fair* (<https://www.euskal.org/>), where a homogeneous Liquid Galaxy was exhibited. Note that while using discount usability testing [19], we cannot completely answer such questions as, "How usable is this system?" or "What is the UX level of the system?". Nevertheless, it provides a quick and non-formal way of observing people using our system. According to our aims, their first usages and impressions were enough for our purposes.

Table 1 shows the Multimedia eXperience Level (MXL) of people involved in the test, grouped by age. Users were asked about their MXL with an orientative value from 1 to 5 (1 being the lowest value and 5 the highest) that indicates the familiarity of the users using multimedia applications.

As could be expected, we can observe in the Table 1 that the lower the age of the participants in the test, the higher their familiarity with using multimedia applications. In general, we can see that the majority of the participants had a MXL in a range between 3 and 4, which is logical if we take into account that the encounter was oriented towards computer enthusiasts.

The test consisted on presenting six locations in a guided tour of the city of New York, where the task for each user was to press a key when he/she wanted to go on to the next point on the tour. The reason for choosing New York is that we wanted to draw the user's attention

Table 1 MXL in relation to age ranges

Age Range	MXL				
	1	2	3	4	5
12-16	–	–	2	5	3
17-24	–	–	4	7	7
25-35	–	2	7	8	4
36-50	1	7	10	5	2
51-75	5	3	3	1	–

towards the city as a famous place with very important points of interest (POI) that any user is willing to see. We asked the volunteers to do the tour with no time-limit, spending as much time as they would like looking at the buildings. In addition, we told them to only go to the next location when they were bored with the current one. This method would tell us how long a person usually spent on each point of the tour in a real environment, giving us real VR values to match our performance metrics.

Figure 14 shows the results for the test with users, grouped by their MXL category, showing the mean and deviation of the time spent by every MXL group at each of the six points on the New York tour. As a guideline, it also shows a horizontal line indicating the time needed to load each of the points on the tour. Given that the city of New York has a higher density of 3D buildings than the three benchmarking tours described in Section 4 (Barcelona, Horsens and Sahara), the loading time of New York was much longer.

According to the values obtained, users can be grouped into two sets:

- Group 1 (MXL1-2): This includes people with low technological skills but with a high attraction to this. The technology is more impressive to them or they are more interested in the tour. So, they spent longer time looking at each location.

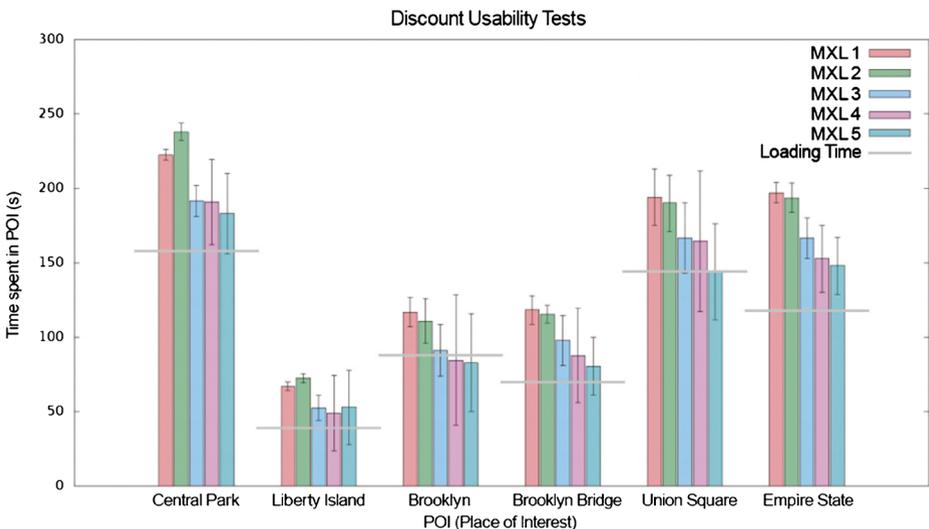


Fig. 14 Discount usability test

- Group 2 (MXL3-5): This includes young people but also those with high multimedia experience or knowledge of multimedia applications. In general, they do not wait long enough to visualize the data completely loaded because they are too impatient or non-motivated. It is worth pointing out that people with higher MXL do not usually expect to be amazed by this kind of application. Therefore, their interest towards the multimedia environment is low and they give lower visualization times. This statement also affects young people as their MXL tends to be very high or they are too impatient.

Both groups follow a homogeneous pattern of visualization in a way that if one group tended to spend less time than another at a given point in the tour, this trend would be followed throughout all the points. In general, we noticed that the users were more interested in the points of “Central Park”, “Liberty Island”, “Brooklyn Bridge” and “Empire State”, as they spent more time to visualizing them compared with their loading time. This confirms that if someone is interested in a place, he/she will spend more time visualizing this. Likewise, the graph also shows that some people did not need all the data to be downloaded in order to visualize it comfortably, which means that lower VR values would be acceptable.

Regarding the deviations, there were greater deviations in the MXL4 and MXL5 groups. This was because many of the users from those groups were younger and behaved unpredictably, such as skipping the tour fast.

From the visualization times shown in the Fig. 14, we calculated the related values of the VR parameter. Table 2 shows the VR parameter for every MXL and the average (AVG) for each point of the tour. From Table 2, we can see that high VR values were achieved in the points with interest on the tour (Central Park, Liberty Island, Brooklyn Bridge and Empire State) and for the groups of people with low MXL. Likewise, it is worth pointing out a particular behavior for this specific tour. On one hand, given the high density of 3D buildings in New York, the points chosen need a long time to fully load. On the other hand, they let the users imagine their content while they were still loading. Therefore, the VR values obtained for this specific tour were lower than the ones obtained from other tours with a lower density of 3D buildings.

In order to obtain the minimum VR value accepted by real users, we took the values obtained for the points with most interest specified before. This is because we assume that a habitual user of a cluster display wall is a person who is expecting to see a specific image in high resolution and, as a consequence, those points with less interest are not representative for our purposes. According to this, we chose a $VR = 30\%$ as the minimum acceptable VR value for users. This value corresponds to the average value for the *Central Park* point and is the minimum VR average among the four points of most interest (Central Park, Liberty Island, Brooklyn Bridge and Empire State). We use this value as a reference in further studies shown in this paper.

Table 2 VR for every MXL category

	MXL1	MXL2	MXL3	MXL4	MXL5	AVG
CentralPark	41 %	51 %	21 %	21 %	16 %	30 %
LibertyIsland	72 %	86 %	35 %	26 %	36 %	51 %
Brooklyn	33 %	26 %	4 %	0 %	0 %	13 %
BrooklynBridge	69 %	65 %	40 %	25 %	15 %	43 %
UnionSquare	35 %	32 %	16 %	14 %	0 %	19 %
EmpireState	67 %	64 %	41 %	29 %	25 %	45 %

5.3 Benchmarking tests

Having defined our new parameter and set the VR minimum threshold, we performed tests for both homogeneous and heterogeneous clusters using the automated scripts described in Section 4 to simulate the user behavior. This gives an understanding of the performance of the Liquid Galaxy system in relation to several scenarios with different densities of 3D buildings and patterns of users.

5.3.1 Homogeneous system

The homogeneous cluster used in this experimentation was made up of a variable number of nodes, from 3 to 8, where each node had the configuration described in Section 4.

Table 3 shows the impact that the type of environment, the timing jump and the proxy cache have on the VR metric, when we used our benchmarking tours in a cluster of 3 nodes. It is worth remarking that there was a noticeable difference between each tour, which tells us that the type of environment is very important because the more 3D buildings that have to be rendered, the lower the VR is.

Additionally, it can be observed that the short-timing jump gave a lower VR value, so, the user had little time to visualize the images fully. In fact, there were lower values than the minimum threshold of 30% that we defined. This means that the user may not be very satisfied with the visualization of these tours. For this reason, we can state that the timing jump set as 30 seconds is too short, as the table shows values lower than 30% for this in the case of the city and town tours.

Another important parameter to be considered is whether Squid is enabled or disabled, because the Squid ON reduces the external downloaded traffic and images can be visualized faster, thus increasing the VR value. This improvement due to the Squid ON is perfectly visible in Table 3.

In order to analyze how performance parameters behave for bigger platforms, we analyzed scalability, in terms of VR, when the number of nodes increases from 3 to 8 nodes.

Table 4 shows the results obtained in the scalability study with Squid ON. In general, we can see that the overall VR decreases when the number of nodes increases. This happens because the broadband connection is a fixed resource and the bandwidth must be divided between the nodes. Thus, the incoming input of data to be processed by each node per unit of time is diminished and, as a consequence, the rendering time is lengthened. So, the acceptable number of nodes is determined by the type of Internet connection.

As was expected, we can see how Short Timing Jumps obtained an unacceptable VR for the City and Town Tours. On the other hand, a Long Timing Jump of 60 seconds is enough to scale the cluster up to 8 nodes whenever the rendering images are not extremely heavy, such as in the Town and Desert cases are; while it is not sufficient for the City tour case.

Table 3 VR obtained from benchmarking tests

	Short timing jump (30s)			Long timing jump (60s)		
	City	Town	Desert	City	Town	Desert
Squid ON	15 %	27 %	57 %	48 %	56 %	78 %
Squid OFF	11 %	24 %	45 %	39 %	50 %	72 %

Table 4 VR related to scalability

Node	Short timing jump (30s)			Long timing jump (60s)		
	City	Town	Desert	City	Town	Desert
3	15 %	27 %	57 %	48 %	56 %	78 %
5	5 %	22 %	50 %	31 %	52 %	77 %
8	2 %	14 %	46 %	14 %	36 %	68 %

Therefore, whenever the images to be rendered are heavy, a Timing Jump longer than 60 seconds would be needed.

It is also worth pointing out that when the number of nodes is scaled, not all the nodes load the Internet data at the same time, and there is a slight delay of 3 seconds between the central and extreme nodes of the cluster. This delay can influence the VR value but not the real user perception, given that the user focuses on the central nodes and does not give much importance to the extreme nodes. As explained previously, the VR is averaged from all the nodes in the cluster without taking their position into account. With this fact in mind, we can say that the user can have a better perception of the multimedia application using more displays with the same VR value.

After studying the scalability of this specific cluster display wall, we can say that the bandwidth of the Internet connection is a bottleneck, as it limits the speed of renderization.

5.4 Heterogeneous system

In order to extend the usage of a cluster display wall to a wider set of people, we are interested in building a low-cost system. In this context, a real usage cluster would be built with different kinds of nodes, assembling a heterogeneous cluster. According to this, we specified different configurations of Liquid Galaxy setups on which the different tests were carried out. Each configuration was made up of n commodity PCs chosen from the list of nodes given in Table 5. For each kind of node used in this experimentation, Table 5 shows its identifier N_i , hardware specification, benchmarking time T_i in seconds and relative power W_i . It is worth pointing out that the hardware specification of each node was above the minimum required by Google to run Google Earth.

The benchmarking time T_i was obtained by running two different Benchmarks in the node N_i , T_{FPU} and $T_{Integer}$, these being the *FPU Raytracing* and *CPU N-Queens* from the *hardinfo tools*, respectively. Each T_i was calculated as follows:

$$T_i = 0.80 * T_{FPU} + 0.20 * T_{Integer} \quad (2)$$

where the weight assigned to each benchmark was obtained by monitoring the Floating Point and Integer instructions when a standard user was navigating with Google Earth running in the Liquid Galaxy cluster.

The relative power W_i shows the power of the node N_i in relation to the fastest node in Table 5, which was N_4 . This was calculated by means of the following relation:

$$W_i = \frac{T_{min}}{T_i}, \quad (3)$$

where T_i was the benchmarking time of node N_i and T_{min} , the lowest benchmarking time from all the nodes shown in Table 5 (11.5 seconds in this case).

Table 5 Kinds of nodes

Node (N_i)	Characteristics	T_i (s)	W_i
N_1	i5 3330 @ 3GHz (4 cores), nVidia GeForce GT620, 8GB RAM 1600MHz, SSD	20.15	0.57
N_2	i5 3330 @ 2.4GHz (4 cores), nVidia GeForce GT620, 8GB RAM 1600MHz, SSD	25.24	0.46
N_3	i5 3330 @ 2GHz (4 cores), nVidia GeForce GT620, 8GB RAM 1600MHz, SSD	30.75	0.37
N_4	Intel Xeon @ 3.4GHz (4 cores), nVidia GeForce GT420, 6GB RAM 1333MHz, SSD	11.5	1
N_5	i5 3470 @ 3.2GHz (4 cores), MESA DRI Intel IvyBridge, 4GB RAM 1600MHz, HDD	17.58	0.65
N_6	AMD Athlon 64 3500+ (1 core), ATI RS480, 1GB RAM 400MHz, HDD	36.22	0.32
N_7	Core2 Duo @ 3.00GHz (2 cores), Intel 4 Series Graphics, 4GB RAM 1300MHz, HDD	21.13	0.54

Table 6 shows the different Liquid Galaxy configurations used throughout this experimentation sorted by their Heterogeneous Degree. The following information is shown for each configuration:

- n : number of nodes in the cluster
- *Het_Degree*: Heterogeneous Degree parameter that shows how different the nodes are in any specific configuration. This is calculated as follows:

$$Het_Degree = \frac{\sum_{i=1}^{n-1} \frac{W_{max} - W_i}{W_{max}}}{n - 1}, \quad (4)$$

where n is the number of nodes in the cluster, W_i is the relative power of node N_i and W_{max} is the relative power of the fastest node in the cluster. Note that a *Het_Degree* = 0 means that the cluster is homogeneous and a value of 1 means that the cluster is completely heterogeneous. In practice, the *Het_Degree* of the compositions that we built only ranged between 0 and 0.43 because the upper bound was limited by two different aspects: the slowest nodes, which were limited by the minimum specifications given by Google, and the fastest nodes, which were limited by the fact that we only wanted to use commodity hardware.

- Node composition: the number of nodes of each type, shown in Table 5, that make up the platform.

All the configurations had a local network of 100 Mbps and a standard 10-Mbps broadband Internet connection. It is worth pointing out that we always maintained a type N_1 node as the master node in all the configurations.

Given our purpose of analyzing the performance of the new metric in relation to different *Het_Degree*, we first evaluated the VR over the six specific Liquid Galaxy compositions given in Table 6 that had three nodes. Each composition is identified by the corresponding *Het_Degree*.

Table 6 Liquid Galaxy composition

n	Het_Degree	Node composition
3	0.43	2xN ₁ + 1xN ₄
3	0.22	2xN ₁ + 1xN ₆
8	0.19	2xN ₁ + 1xN ₂ + 1xN ₃ + 2xN ₅ + 2xN ₇
3	0.18	2xN ₁ + 1xN ₃
5	0.17	2xN ₁ + 1xN ₃ + 1xN ₅ + 1xN ₇
3	0.12	2xN ₁ + 1xN ₅
3	0.1	2xN ₁ + 1xN ₂
3	0	3xN ₁

Table 7 shows the VR obtained for each composition in the three tours tested and the Short and Long Timing Jumps. Together with the *Het_Degree* value, the W_{avg} , which is the average power (W_i) of the three nodes forming the cluster, and the $W_{slowest}$, which corresponds to the minimum W_i power among them, are shown.

In general, the results in the Table 7 show a significant difference between Short and Long Timing Jumps, as already seen in the homogeneous test. Likewise, there are more differences between clusters in the Short Timing Jumps test. The reason of this behavior is that all the configurations for the Long Timing Jumps, apart from the *Het_Degree* = 0.22 case, had enough time to download the data from the Internet and, as a consequence, the CPU advantage of some clusters was not highlighted. Thus, we can see that the Timing Jump of 60s was too long to differentiate the performance between similar clusters. The exception to this behavior was the case of the *Het_Degree* = 0.22, which obtained very low values for the City and Town tours. This poor performance reveals that a low $W_{slowest}$, such as in this case, can lead the cluster’s VR to very low levels. This shows that clusters

Table 7 VR related to heterogeneous clusters

<i>Het_Degree</i>	W_{avg}	$W_{slowest}$	City	Town	Desert
Short timing jump (30s)					
0.43	0.71	0.57	14 %	36 %	59 %
0.22	0.49	0.32	4 %	13 %	49 %
0.18	0.50	0.37	8 %	26 %	55 %
0.12	0.60	0.57	14 %	25 %	58 %
0.1	0.53	0.46	11 %	25 %	54 %
0	0.57	0.57	15 %	27 %	57 %
Long Timing Jump (60s)					
0.43	0.71	0.57	51 %	57 %	78 %
0.22	0.49	0.32	14 %	24 %	63 %
0.18	0.50	0.37	43 %	54 %	77 %
0.12	0.60	0.57	50 %	56 %	78 %
0.1	0.53	0.46	46 %	53 %	77 %
0	0.57	0.57	48 %	56 %	78 %

with similar W_{avg} , such as the case of $Het_Degree = 0.22$ and 0.18 , can have significantly different performance because one of the nodes is very slow.

Focusing on the Short Timing Jumps tests, we can see that the differences between clusters were minimal in the Desert tour. In the Town tour, there were greater differences given the increase in the number of rendering operations and the consequent parallel increase in the influence of W_{avg} . Likewise, we can point out that the fastest cluster ($W_{avg} = 0.71$) obtained the best VR for the Town tour, but not in the City tour. This was due to the broadband connection for the tests, this being a common resource of $10Mbps$, that did not allow the CPU speed advantage of the best clusters to be exploited when the data to download was extremely high. Thus, the VR for the City tour was always below 14 %, which can be considered unacceptable for a standard user. Note that for the City tour, the homogeneous configuration ($Het_Degree = 0$) gave slightly better performance, although this difference was not significant. Apart from the homogeneous case, it is worth emphasizing that the cluster with the highest heterogeneous degree ($Het_Degree = 0.43$) obtained the best performance, this being the configuration with the highest $W_{slowest}$ and W_{avg} parameters.

In general, we can state that W_{avg} and $W_{slowest}$ are the parameters with the greatest impact on the VR value, whereas the Het_Degree does not have as much significance as we could imagine. Thus, and taking our results into account, it could be established that heterogeneity is not a problem when building this kind of infrastructure while the Timing Jump is adjusted to the broadband connection and hardware characteristics of the available nodes.

In order to analyze the scalability of a heterogeneous Liquid Galaxy platform, the number of nodes was increased from 3 up to 8 nodes. Likewise, with the aim of isolating the influence of the size in relation to the others parameters, Het_Degree , W_{avg} and the $W_{slowest}$, these were maintained close to constant throughout this experimentation.

Table 8 shows the results obtained. In general, we can see that, as expected, the overall VR decreases when the number of nodes increases. We can see very similar behavior as in Section 5.3.1 with little differences. The results of this scalability study again show that heterogeneity is not a problem when building a cluster with different types of node. We can point out that the Internet connection still limits the speed of renderization, thus becoming a bottleneck.

Table 8 VR related to the scalability

Node	Het_Degree	W_{avg}	$W_{slowest}$	City	Town	Desert
Short Timing Jump (30s)						
3	0.18	0.56	0.37	8 %	26 %	55 %
5	0.17	0.52	0.37	2 %	16 %	56 %
8	0.19	0.53	0.37	2 %	8 %	46 %
Long Timing Jump (60s)						
3	0.18	0.56	0.37	43 %	54 %	77 %
5	0.17	0.52	0.37	29 %	49 %	78 %
8	0.19	0.53	0.37	16 %	39 %	69 %

6 Conclusions and future work

In this paper, we present a cluster display wall infrastructure built with commodity hardware with both homogeneous and heterogeneous configuration of the nodes that make up the cluster. We analyzed the performance for both kinds of applications, client-server and master-slave. Given that client-server applications do not represent a significant consumption of resources, we focus on the execution of the Google Earth as a representative case of master-slave applications that run in this kind of system, so that the results may be indicative of the viability of using this kind of infrastructure in a wide range of everyday applications that can benefit from an immersive visualization environment.

The performance analysis of Google Earth showed that CPU, memory and local network are good enough to visualize the images. However, the bandwidth of the external network constitutes a bottleneck. This means that the user navigation behavior in the environments tested is a determining factor for being able to load all the needed data. Thus, the time interval between consecutive displacements has to be adjusted according to the density of data of the environment to be visualized.

In order to establish a relationship between performance analysis and the user satisfaction when using our cluster display platform, we defined the VR (Visualization Rate) parameter. This measures the visualization time in relation to the total time used in each test. A discount usability test was performed with real users with different multimedia skills. It was observed that for both homogeneous and heterogeneous systems, the VR value is mainly influenced by the environment (amount of 3D buildings to be visualized) and user behavior (interval of time between consecutive displacements). According to our experience and experimentation, VR values above 30 % can be considered acceptable because this is the minimum value where the user feels that the time spent loading the images is short in relation to the visualization time.

Additionally, our results show that for heterogeneous systems, the slowest CPU can have a negative impact on the VR, whereas the degree of heterogeneity has less influence. On the other hand, the heterogeneity of the cluster is not a problem when building this kind of system whenever the power average of the cluster and the power of the slowest node are above a certain threshold.

Likewise, we tackled the scalability problem for both types of system, the homogeneous and the heterogeneous, scaling up to 8 nodes. From this analysis, we can say that the bottleneck is located in the Internet connection, as it greatly limits the rendering speed. In general, we saw that the overall VR decreases when the number of nodes is increased.

In future works, we plan to develop an analytic model to predict the performance of any given heterogeneous platform. Additionally, we are interested in extending our analysis of the user experience with the Liquid Galaxy platform by means of eye tracking glasses combined with other traditional techniques for HCI environments.

Acknowledgments We would like to thank to the company Ponent 2002 for facilitating the use of the Lleida Liquid Galaxy and specifically to Andreu Ibañez for his collaboration.

This work was supported by MEyC-Spain under contract TIN2011-28689-C02-02 and TIN2014-53234-C2-2-R.

References

1. Ball R, North C (2005) Analysis of user behavior on high-resolution tiled displays. *Lect Notes Comput Sci* 3585:350–363
2. Barri I, Roig C, Giné F, Solsona F (2011) Mapping mmofps over heterogeneous distributed systems. *J Supercomput* 58(3):341–348
3. Chen Y, Chen H, Clark D, Liu Z, Wallace G, Li K (2001) Software environments for cluster-based display systems. In: *Proceedings - 1st IEEE/ACM international symposium on cluster computing and the grid*, vol 2001. IEEE, CCGrid, pp 202–210
4. Chung H, Andrews C, North C (2014) A survey of software frameworks for cluster-based large high-resolution displays. *IEEE Trans Vis Comput Graph* 20(8):1158–1177
5. DeFanti T, Acevedo D, Ainsworth RA, Brown MD, Cutchin S, Dawe G, Doerr KU, Johnson A, Knox C, Kooima R et al (2011) The future of the CAVE. *Central Eur J Eng* 1(1):16–37
6. Doerr K, Kuester F (2011) CGLX: a scalable, high-performance visualization framework for networked display environments. *IEEE Trans Vis Comput Graph* 17(3):320–332
7. Eilemann S, Makhinya M, Pajarola R (2009) Equalizer: a scalable parallel rendering framework. *IEEE Trans Vis Comput Graph* 15(3):436–452
8. Google (2010) Liquid galaxy live demo at TED. *Lecture notes in computer science*. Springer
9. Google (2013a) Google Earth. <http://www.google.com/earth/index.html>, retrieved at 2015-06-07
10. Google (2013b) Liquid galaxy website. <http://blog.endpoint.com/2013/11/liquid-galaxy-and-its-very-own-street.html>, retrieved at 2015-06-07
11. Google (2015) Liquid galaxy website. <http://www.google.com/earth/explore/showcase/liquidgalaxy.html>, retrieved at 2015-06-07
12. Greggman EH (2009) Aquarium WebGL. <http://webgl.samples.org/aquarium/aquarium.html>, retrieved at 2015-06-07
13. Hassenzahl M, Tractinsky N (2006) User experience - a research agenda. *Behav Inf Technol* 25(2):91–97
14. Humphreys G, Houston M, Ng R, Frank R, Ahern S, Kirchner PD, Klosowski JT (2002) Chromium: a stream-processing framework for interactive rendering on clusters. In: *ACM transactions on graphics (TOG)*, vol 21. ACM, pp 693–702
15. Ishiguro Y, Rekimoto J (2011) Peripheral vision annotation: noninterference information presentation method for mobile augmented reality. In: *Proceedings of the 2Nd augmented human international conference*. ACM, AH '11, pp 8:1–8:5
16. ISO (2010) Ergonomics of human system interaction - part 210: human-centred design for interactive systems (formerly known as 13407). ISO 9241-210
17. Izadi S, Brignull H, Rodden T, Rogers Y, Underwood M (2003) Dynamo: a public interactive surface supporting the cooperative sharing and exchange of media. In: *Proceedings of the 16th annual ACM symposium on User interface software and technology*. ACM, pp 159–168
18. Johnson A, Leigh J, Morin P, Van Keken P (2006) Geowall: Stereoscopic visualization for geoscience research and education. *Comput Graph Appl IEEE* 26(6):10–14
19. Krug S (2009) Rocket surgery made easy: the do-it-yourself guide to finding and fixing usability problems. New Riders
20. Lau CD, Levesque MJ, Chien S, Date S, Haga JH (2010) ViewDock TDW: high-throughput visualization of virtual screening results. *Bioinformatics* 26(15):1915–1917
21. Leigh J, Johnson A, Renambot L, Peterka T et al (2013) Scalable resolution display walls. *Proc IEEE* 101(1):115–129
22. Liu C, Chapuis O, Beaudouin-Lafon M, Lecolinet E, Mackay WE (2014) Effects of display size and navigation type on a classification task. In: *Proceedings of the SIGCHI conference on human factors in computing systems*. ACM, pp 4147–4156
23. Nielsen J (1993) Usability engineering. Morgan Kaufmann Publishers Inc
24. Nirmimesh HarishP, Naraayanan P (2007) Garuda: a scalable tiled display wall using commodity pcs. *IEEE Trans Vis Comput Graph* 13(5):864–877
25. OpenArena Team (2005) OpenArena (Quake III Arena). www.openarena.ws, retrieved at 2014-12-01
26. OSG (2014) Openscene graph. <http://www.openscenegraph.com>, retrieved at 2015-06-07
27. Ponent 2002 (2012) G-liquid galaxy. <http://www.g-liquidgalaxy.com>, retrieved at 2015-06-07
28. Reda k, Febretti A, Knoll A, Aurisano J, Leigh J, Johnson A, Papka M, Hereld M (2013) Visualizing large, heterogeneous data in hybrid-reality environments. *IEEE Comput Graph Appl* 33(4):38–48
29. Renambot L, Rao A, Singh R, Jeong B, Krishnaprasad N, Vishwanath V, Chandrasekhar V, Schwarz N, Spale A, Zhang C et al (2004) SAGE: the scalable adaptive graphics environment. In: *Proceedings of WACE, Citeseer*, vol 9, pp 2004–09

30. Schikore D, Fischer R, Frank R, Gaunt R, Hobson J, Whitlock B (2000) High-resolution multiprojector display walls. *IEEE Comput Graph Appl* 20(4):38–44. doi:[10.1109/38.851748](https://doi.org/10.1109/38.851748)
31. Smarr L, Brown M, de Laat C (2009) Special section: OptIPlanet the OptIPuter global collaboratory. *Future Gen Comp Syst* 25(2):109–113
32. Squid Team (2002) Squid web proxy cache. <http://www.squid-cache.org>, retrieved at 2015-06-07
33. Staadt OG, Walker J, Nuber C, Haman B (2003) A survey and performance analysis of software platforms for interactive cluster-based multi-screen rendering. *Eurograph Workshop Virt Env*:261–270
34. (2012) Panoramic video streaming. <https://code.google.com/p/liquid-galaxy/wiki/PanoramicVideo>, retrieved at 2015-06-07



Ismael Arroyo received BSc and Master in Computer Science degrees from the Universitat de Lleida (UDL), Spain, in 2011 and 2013 respectively. He has an UdL grant of the group of distributed computing (GCD). The project, that he is developing with GCD, is a study of the Liquid Galaxy system, an immersive visualization cluster, implementation of monitoring and benchmarking tools and modeling some application behaviors in this system.



Francesc Giné received the B.S. in telecommunication engineering from the Universitat Politècnica de Catalunya (UPC), Spain, in 1993 and the M.S. and Ph.D degrees in computer science from the Universitat Autònoma de Barcelona (UAB), Spain, in 1999 and 2004, respectively. He is currently an Associate Professor of Computer Architecture at University of Lleida (UdL), Spain. His research interest includes multicluster and peer-to-peer computing and scheduling-mapping for parallel processing.



Concepció Roig received the MS and PhD degrees in Computer Engineering from the Universitat Autònoma de Barcelona, Spain, in 1996 and 2002 respectively. She has been an associate professor in the Department of Computer Science at the University of Lleida, Spain, since 1992. Her current research interests include parallel and distributed systems, modelling of parallel applications, task graph models, static mapping and optimization of parallel applications.



Toni Granollers is a full Professor at Department of Computer Science and Industrial Engineering at Politechnical School (University of Lleida). Studies: PhD in Computer Science, Human-Computer specialization. University of Lleida, 2004. Master in “Architectural Structures”. Politechnical of Catalonia University, 1996. Degree in Sciences (Informatics). Autonomous University of Barcelona, 1998. Doctoral Thesis in MPIu+a. A methodology that integrates Software Engineering, Human-Computer Interaction and accessibility in a multidisciplinary teams context.