Towards Virtualization of Rich Applications for Distribution under a SaaS Model

D. A. Rodríguez-Silva, L. Adkinson-Orellana, V. Fernández-Díaz and F. J. González-Castaño

Gradiant, Network and applications Area, Ed. Citexvi, 36310 Campus Vigo, Spain
{darguez,ladkinson,vfernandez,javier}@gradiant.org

Keywords: Virtualization, SaaS, Desktop Applications, Cloud Storage, Remote Desktop, Streaming.

Abstract: Current mobile devices (smartphones, tablets, netbooks...), widely used nowadays, can run potent native applications, but they cannot support typical desktop applications from any operating system. Since modern devices support recent web standards as HTML5, it is possible to develop a solution based on a thin web client to grant remote access to desktop applications offered under a SaaS model. This paper proposes the development of an innovative remote desktop system able to detect application content and encode it efficiently in real-time, to support an optimal visualization on clients, combining both remote desktop and streaming protocols. The system is hosted by a cloud infrastructure that ensures scalability, and it follows a pay-per-use model. Application providers can include their software in a dynamic cloud repository, from where it is launched remotely to meet final user demands.

1 INTRODUCTION

Nowadays, the use of mobile devices is increasing to the point that, according to some forecasts (Barton, Zhai and Cousins, 2006), smartphones and tablets will displace PCs in the near future. Besides, set-top-boxes and, later, intelligent TVs, with enhanced TV capabilities and functionalities, are also widespread (Want, 2009). Although some mobile devices do not seem constrained in terms of computing performance or even display size (since they can be attached to a large screen if available), the relationship between computational power and battery capacity, the input interface or the maximum number of concurrent applications are also limiting factors (Simoens, De Turck, Dhoedt and Demeester, 2011).

Consequently, for some reason or another, heavy, highly demanding applications are not appropriate for simple user devices, and in most cases applications functionalities need to be adapted or reduced to show up in these devices. Cloud-oriented remote visualization technologies which allow running an application on a remote high-performance server and exporting its graphic user interface to a low-end terminal can be of help. They also free end-users from the burden of software maintenance and protection against malicious attacks.

Remote desktops are well-known visualization technologies (RealVNC, n.d.). They usually perform well with static graphic outputs, as they just transmit screens differences. However, current remote desktop technologies present drawbacks, as the lack of efficient audio support or the poor visualization of dynamic contents (i.e. video). On the other hand, even though streaming may offer an appropriate solution in that case, it results inefficient when only a small part of the transmitted content is changing.

As a compromise, to minimize bandwidth, the quality of the stream is usually constrained, so that texts are not clearly displayed.

In this paper we describe a virtualization platform that provides optimal remote access to rich interactive applications through a thin web client, supported by several recent mobile devices. The platform relies on a cloud infrastructure that will also host an application market, following a pay-per-use business model. Section 2 explains the proposed architecture and section 3 discusses related work. Finally, section 4 concludes the paper.
2 SYSTEM ARCHITECTURE

Keeping in mind the scenario described in the previous section, the proposed work aims at developing intelligent virtualization technologies to provide rich interactive multimedia applications in the cloud. The platform we pursue will allow the execution of any application in a powerful virtual machine, located in a remote location. Thus, a user will be able to visualize the application output and interact with it from any device without the need to install any local software. This will be completely transparent to the user, regardless of the device capabilities.

![Diagram of the system architecture](image)

Figure 1: General architecture of the proposed system.

The client side will be developed entirely using standard web technologies, to achieve complete platform independence. The server side will deploy ready-to-use applications and will detect motion in screen content, in order to code any necessary information intelligently to send it to the client in real-time using the most appropriate protocol (RFB or streaming) in each case. To enhance scalability, a cloud infrastructure will be used. The general architecture of the system is shown in Figure 1.

2.1 Cloud Infrastructure

The proposed architecture is composed of four different layers: IaaS, hypervisors, a virtual resource layer and the SaaS platform. The infrastructure level provides the upper layers with the physical resources necessary to run the applications. As the server side will rely on a virtual machine, a cloud broker is necessary to manage the cloud infrastructure, in order to create virtual machines on demand. Thus, resources will be managed in an efficient manner, assuring that the requested applications will be executed smoothly in the SaaS layer.

Each virtual machine will be able to support several applications from different users. Hence, virtualization is not only applied at OS-level (when the virtual machines are deployed), but also at the application level (applications that run over the OS). This scheme improves the efficiency of resource management, as a single virtual machine can be exploited by several users working with their own different applications, instead of allocating a dedicated virtual machine to individual users or applications.

As different users share the same virtual environment, additional security measures are required to guarantee data isolation. This is achieved by keeping the user data that the applications access in a cloud storage instead of in the virtual machine. How this storage is implemented is open, as it can be an internal component of the solution or it can be provided by an external cloud storage service (e.g. Dropbox). In both cases, the virtualized applications will manage user files remotely and encryption techniques will be applied to enhance the cloud storage mechanism, so user data will remain safe during the whole process.

![Diagram of the cloud architecture](image)

Figure 2: Cloud architecture.
application following the OCCI standard (OCCI, 2009). This template is stored in the application repository, enabling the corresponding application for user access.

### 2.2 Remote Visualization

Our virtual machines currently support either Windows or Linux OS templates, although the proposed scheme could be extended to other operating systems. Each virtual machine will offer support to the virtualized applications that run in its particular OS. Furthermore, each application will be managed through a different user session in the selected OS.

In addition to the virtualized applications, each virtual machine includes a RFB server to allow users to access their applications remotely. The protocol chosen for RFB communication is VNC (VNC, 1999), one of the most popular ones. VNC servers can support different user sessions simultaneously through displays. Thus, each user working with a virtualized application will belong to a different OS session, tied to a different VNC display (Figure 3).

![Application virtualization architecture](image)

**Figure 3:** Application virtualization architecture.

Nevertheless, Windows and Linux applications are delivered following different schemes:

- The virtualization of Linux applications relies on Xvfb (Xvfb, 2012), an X window virtual frame buffer, which implements a virtual X server as a frame buffer in which different graphical outputs can be dumped, using several virtual displays, without overlaps. Consequently, several applications can be served through several VNC servers in the same virtual machine.
- In Windows virtual machines there are no virtual graphic devices such as Xvfb, so each application needs to be virtualized in a separated virtual machine that contains its own VNC server.

### 2.3 Motion Content Detection

The VNC server has three functions:
- Screen delivery of low-motion content through the VNC protocol.
- HTTP video streaming of high-motion content
- Intelligent switching between VNC and HTTP streaming protocols depending on the content.

In order to detect whether the screen content is static or dynamic, the raw images available at the VNC server are analyzed as follows:

```c
while(true)
    fork=0 to N-1 begin
        fb_k = getFramebuffer(k);
        for x = 0 to blocks(fb_k)-1
            w = 32;
            h = 32;
            b_{x,y} = getBlock(fb_{x,y},w,h);
            changed = compare(b_{x,y},b_{x-1,y});
            if (changed == true)
                block_changes_{x,y} += 1;
        end for
    end for

    for i=0 to size(block_changes)-1
        if (block_changes_{i} > threshold)
            block_changed_{i} = 1;
        else
            block_changed_{i} = 0;
        end

    end if
    detectMotionArea(block_changed);
end for

sleep(t);
end while
```

The frame buffer content is gathered ‘N’ times in a loop, grouping the raw pixels in 32x32 blocks. The blocks are compared with their instances in the previous iteration and, if at least a pixel differs, the entire block is marked as a change; on the other hand, if all the pixels are the same, the block remains unchanged.

When the comparison loop ends, the number of changes occurred in each block is checked. If it exceeds a predefined threshold, the pixels of the block are set to ‘1’; otherwise, the block is set to ‘0’.

Finally, an array of binary values is obtained, showing the position of the dynamic blocks in the screen. Figure 4 shows an example of a VLC player playing a video inside an Ubuntu desktop and its...
corresponding representation as an array with the motion detected areas marked with ‘1’.

Figure 4: Motion detection process.

Due to the high computational load required to compare all the pixels of the frame buffer, only a representative subset is checked. We decided to look only at those pixels in block diagonals, simplifying the complexity of the problem from order $n^2$ to $2n$, where $n$ is the block width in pixels (assuming a square block). Nevertheless, the accuracy of the detection is preserved due to the proximity of the pixels in a $32\times32$ block, compared with a typical 1024x720 desktop size. In addition, a hysteresis criterion is followed to prevent false positives, improving reliability.

Once the motion area is detected, the VNC server stops sending the RFB images corresponding to that area. Then the stream is generated and the client receives the URL to play the video in the corresponding rectangle over the VNC content.

2.4 Web Client

A major concern regarding the visualization of applications is the necessity of a specific client for each device and platform. A possible solution is to develop a thin client based on common technologies supported by all the devices (typically Web over HTTP) following the latest standards as HTML5. Thus, the client may be accessible everywhere through compatible web browsers.

The proposed solution is able to offer several applications, regardless of their profile (office applications, image editor, video editor, etc.) or their operating system (Windows, Linux...) by means of an HTML5 web client, as shown in Figure 3. This client can combine and display contents received by VNC and HTTP streaming protocols seamlessly, by using canvas and video tags, as explained in the previous section.

As a final result, users will be able to access the platform by means of a web portal containing the client and accessible from any device. This portal will show a catalogue with existing applications for the users to run them remotely through standard Internet browsers supporting HTML5. The platform has an open-community orientation, so that users will be able to demand and evaluate applications and software providers will distribute their applications easily.

3 RELATED WORK

The fact that thin clients improve power efficiency has been demonstrated previously (Vereecken, Deboosere, Simoens, Vermeulen and Colle, 2010). However, a comprehensive study over a wide range of well-known thin client protocols (Deboosere, De Wachter, Simoens, De Turck and Dhoedt, 2007) has proved that additional functionality is required for a satisfying multimedia experience. In this case streaming is a feasible option, because it is a mature technology, optimized to transmit multimedia data through the Internet. Nevertheless, it may consume lots of unnecessary bandwidth in case the user interface is quite static. Several studies have combined remote desktop and streaming technologies. Simoens, Praet, Vankeirsbilck, De Watcher and Deboosere(2008) proposed a hybrid remote display protocol approach in which the images are relayed to the client either through the RFB protocol or through video streaming depending on the level of motion detected. In (Tan, Gong, Wu, Chang and Li, 2010), another solution is presented.
based on a hybrid RTP protocol to transmit motion content. However, these solutions are not scalable at all, and they severely limit the number of users that can access the system. To solve this, some authors have proposed the use of Cloud Computing to support scalable remote visualization through optimized protocols for mobile devices, although there are still many open issues (Simoens et al., 2011). In (Zhong, Wu, Li and Li, 2010), an approach named vSaas for providing software as a service from the cloud was described. The system allows end users to access different software programs transparently, without limitations on the client operating system or device capabilities. vSaas employs OS-level virtualization and remote display technologies. Nevertheless, this platform does not offer a smoothly remote visualization of applications, since it relies in traditional remote desktop technologies. Shi, Lu, Li and Engelsma (2010) have present the SHARC solution for enabling scalable support of real-time 3D applications in a Cloud Computing environment, delivering content to clients through a streaming server. However, this proposal requires installing a specific client or a Flash browser to visualize the 3D contents.

4 CONCLUSIONS

Recent advances in technologies as Cloud Computing and virtualization have made possible to run desktop applications in remote servers and transfer their screen content to a client. This allows sharing the advantages of local desktop applications, such as rich user interfaces, and those of on-line applications, such as scalability and centralized maintenance. This paper focuses on the development of a virtualization platform that will provide optimal remote access to rich interactive applications through a thin HTML5 web client supported by many recent mobile devices. For this to become possible, it was necessary to apply mechanisms and algorithms at the server side to automatically detect changes in the screen content of the applications and code the dynamic areas in real-time when necessary, combining optimal protocols to display the screens at the client side in a efficient way. In addition, the solution, which is supported by a cloud infrastructure, includes a repository with applications that will be delivered as SaaS. The creation of a community around the platform will guarantee that the latest popular applications will be always available, encouraging new end-users and application providers to participate.

ACKNOWLEDGEMENTS

This research has been supported by projects VIMAIN (grant IPT-2011-1224-430000), funded by the Spanish Ministry of Economy and Competitiveness, SECANI (grant 10SEC001CT), funded by Xunta de Galicia (Spain), and partially by CloudMeUp (grant IDI-20101357), funded by CDTI (Spain).

REFERENCES


