ISSN : 2347 - 8446 (Online)
ISSN : 2347 - 9817 (Print)

Virtualized Screen: A Third Element for Cloud-Mobile Convergence

[I]Vishal M. Toshniwal, [II]Vishal P. Kawale, [III]Tushar L. Bhanage, [IV]Rahul S. Sonawane
[I,II,III,IV]I.T Dept., Pune University, Maharashtra, India
Guided By: Prof. Abhale B.A

Abstract

Mobile and cloud computing have emerged as the new computing platforms and are converging into a powerful cloud-mobile computing platform. This article envisions a virtualized screen as a new dimension in such a platform to further optimize the overall computing experience for users. In a virtualized screen, screen rendering is done in the cloud, and delivered as images to the client for interactive display. This enables thin-client mobile devices to enjoy many computationally intensive and graphically rich services. Technical challenges are discussed and addressed. Two novel cloud-mobile applications, Cloud Browser and Cloud Phone, are presented to demonstrate the advantages of such a virtualized screen.[1]

Keywords

Virtualized Screen, Screen Rendering, Remote Computing, Screen Compression, Screen Transmission.

I. Introduction

A virtualized screen as a new dimension in such a platform to further optimize the overall computing experience for users. In a virtualized screen, screen rendering is done in the cloud, and delivered as images to the client for interactive display. This enables thin-client mobile devices to enjoy many computationally intensive and graphically rich services.

Screen virtualization or screen rendering in the cloud doesn‘t always mean putting the entire screen-rendering task in the cloud. Depending on the actual situations-such as local processing power, bandwidth and delay of the network, data dependency and data traffic, and display resolution-screen rendering can be partially done in the cloud and partially done at the clients (that is, scalable screen virtualization); collaboratively, a complete rendered screen is presented to the user. This is very similar to traditional cloud computing in which we have to decide where we should put the program execution and data storage, either remotely in the cloud or locally on the device, to obtain an optimized computing experience[2].

This flexibility added by screen virtualization can enable us to provide an even further optimized computing experience by balancing between local client devices and remote clouds in terms of data storage, program execution, and now, screen rendering. However, rendering a screen in the cloud also introduces obstacles for the client devices to access the virtual screen, if it needs to maintain high-fidelity display images and responsive user interactions. Fortunately, we have already developed a number of advanced multimedia and networking technologies to address these issues. Ultimately, we would like to define a common cloud API for cloud computing with scalable screen virtualization, with which the developers never have to care where the data storage, program execution, and screen rendering actually occur because the cloud services for the API will adaptively and optimally distribute the storage, execution, and rending among the cloud and the clients. Similar to the GUI, which drove the wide grassroots adoption of personal computers, the cloud API, which bridges the cloud and a wide variety of clients, will introduce new computing experiences both locally and remotely. It will drive the evolution of cloud-mobile computing into a revolution [1].

Screen images might include, but are not limited to, webpages, slides, posters, images, videos, and anything showing up on your computer screen. For natural pictures, many existing image- and video-coding standards (such as, JPEG2000 and H.264/Advanced Video Coding) have demonstrated excellent coding performances. However, they are inefficient for compressing screen images that in most cases contain rich text.

II. System Architecture

Thin-client, remote-computing systems are expected to provide high-fidelity displays and responsive interactions to end users as if they were using local machines. However, the complicated graphical interfaces and multimedia applications usually present technical challenges to thin-client developers for achieving efficient transmissions with relatively low bandwidth links. Figure 1 depicts the proposed thin-client, remote-computing system, which decouples the application logic (remote) and the user interface (local) for clients to use remote servers deployed as virtual machines in the cloud. The servers and the clients communicate with each other over a network through an interactive screen-remoting mechanism. The clients send user inputs to the remote servers, and the servers return screen updates to the clients as a response.

The screen-update model determines whether the screen can be efficiently compressed and transmitted to the client. Most of the existing thin-client systems, such as Virtual Network Computing (VNC) [5] and Remote Desktop Protocol (RDP), [6] represent screen updates with graphics primitives of arbitrarily-sized regions. Such a mechanism allows the server to simply forward the graphics primitives to be updated into the compressors and discard other stable regions directly. At the client side, the screen presenter renders the received graphics primitives and overlays rectangular areas of pixels in destined regions. However, the regions to be updated are usually small and at arbitrary positions, such as menus or edit boxes. Encoding these small regions with arbitrary locations will result in the system suffering from compression efficiency degradation.

In contrast to the architecture based on arbitrarily-sized regions, our thin-client system employs a frame-based screen-representation model. This model reads all the pixels on the screen from the frame buffer at once and feeds the whole screen image into the compressors and transmitters. At the client side, the screen presenter replaces the whole screen with the newly decoded one. As shown in Figure 1, both the server and the client store the same reference frame, which is used to remove the redundancies between consecutive frames. Besides the advantage in compression, the frame-based representation model also simplifies the system architecture without the overhead in scheduling the updates of screen regions. Furthermore, this frame-based representation model could recover from errors due to packet loss very quickly without retransmissions through key frame refreshing. [4]

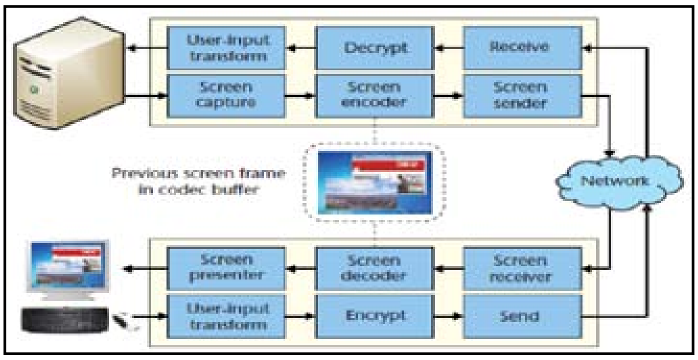


Fig.1: The Interactive Screen Remoting System

Compared with existing remote-computing systems, such as VNC and RDP, the proposed solution could significantly improve user experience in terms of both the smoothness of the screen update and the response in interaction, which are enabled through the advanced screen-compression and transmission technologies introduced next.

Screen in the Cloud:

Rapid development of the Internet has provided opportunities for using remote computing and storage resources hosted by powerful, parallel, distributed machines in a public or private data centre. In a typical cloud-client computing architecture, the data and the program can be stored, loaded, and run remotely and/or locally. To take advantage of the cloud, computationally intensive tasks are usually undertaken in the cloud to generate some intermediate results (for example, HTML data), which are then delivered to the clients for the creation of a locally processed display screen. In other words, local screen rendering is separated from the data storage and program execution, and is connected to them through the Internet.

Up until now, the virtualization of the screen in the cloud has been an underdeveloped area in cloud computing. An even less addressed area is how to leverage the virtual screen in the cloud and combine it with local rendering capabilities to give the same or even better user experiences across different devices, regardless of their computing capability, rendering capability, bandwidth, and screen resolution. [3]

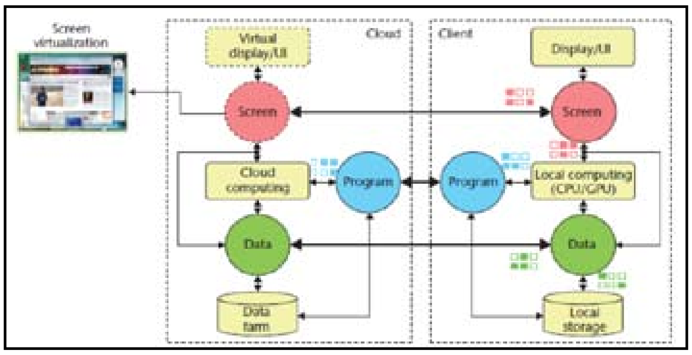


Fig. 2: The Conceptual Diagram of The Cloud-Client Computing Architecture

Figure 2 depicts the proposed conceptual architecture for cloud-client computing, in which the virtual screen is rendered in the cloud. Behaving like the data and the program, the screen can be processed adaptively and collaboratively between the cloud and the clients, determined by the client and cloud capabilities. As we mentioned earlier, the cloud-client architecture with screen adaptation could provide the following benefits:

- An optimized interactive experience to end users across different screens, such as laptop, television, and phone.
- A simplified programing model for developers, as if they were building local applications without struggling on various types of real-time data communications and
- A quick model for software providers to deploy software services in the cloud, through a mechanism of virtual machines plus screen remoting- for example, this model could help to quickly deploy trial software without piracy concerns.

III. Cloud Mobile Services

The rapid evolution of mobile computing offers a wide variety of freedom to mobile users. Besides communication functions, a mobile device could be a computing, sensor, control, gaming, and natural-interaction platform. However, it‘s difficult for a mobile device to serve as a dominant device for all the user‘s computing needs, mainly because of its limited capabilities in terms of computing, storage, display, and interaction. On the other hand, cloud computing offers unlimited computing and storage capabilities through centralized data centres. The capacity of mobile devices.

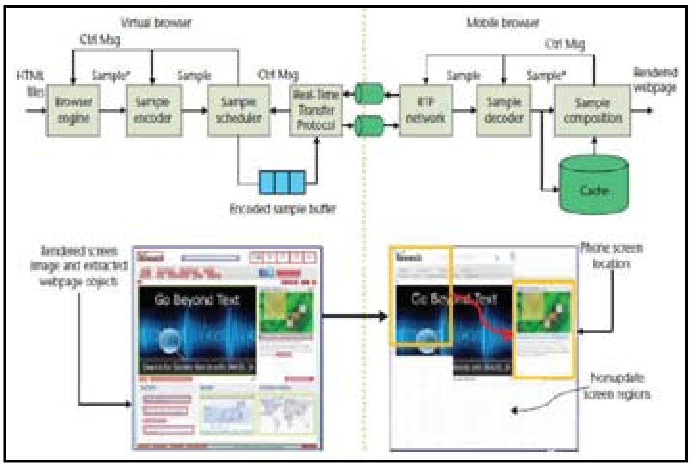


Fig. 3: The System Architecture of the Cloud Browser

ISSN : 2347 - 8446 (Online)
ISSN : 2347 - 9817 (Print)

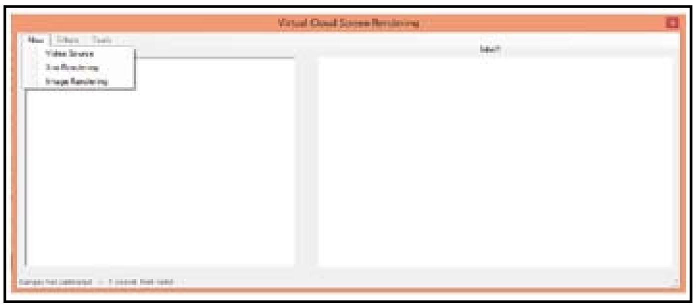
IV. Algorithm

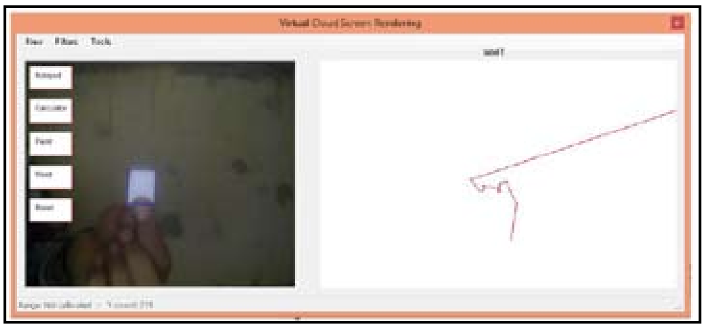
Blowfish Algorithm:

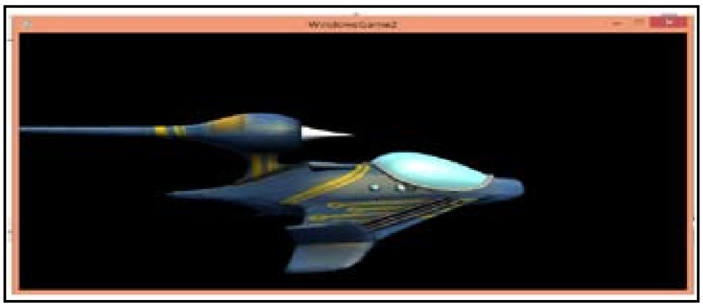
Blowfish was designed by Bruce Schneider as a fast, free alternative to existing encryption algorithms. It is easiest and fastest algorithm for encryption.

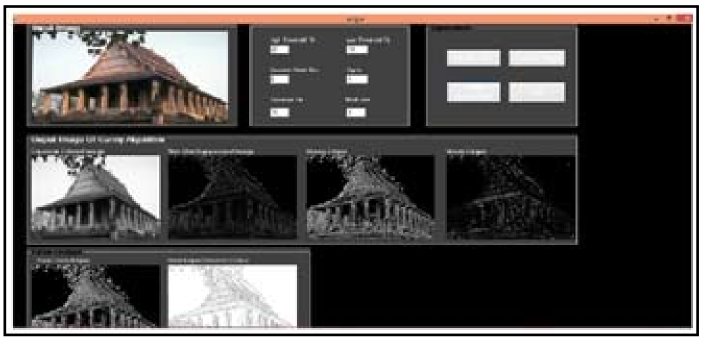
Blowfish is a variable-length key block cipher. It is significantly faster than DES when implemented on 32-bit microprocessors with large data caches, such as the Pentium and the PowerPC. For bulk encryption this algorithm is efficient in encrypting data files or a continuous data stream. Blowfish is a symmetric block cipher that can be used as a drop-in replacement for DES or IDEA. It takes a variable-length key, from 32 bits to 448 bits, making it ideal for both domestic and exportable use. Blowfish uses a large number of sub keys. These keys must be pre computed before any data encryption or decryption.

V. Result









VI. Conclusion

Cloud computing is not a simple extension of Web services. Breakthroughs may come with the introduction of a new application interface model between the cloud and the clients. User interaction with the cloud through client-cloud communications bounds the user experience and presents many technical challenges. We have shown that advanced multimedia compression and networking technologies can bridge the devices and the cloud effectively and efficiently, and potentially could help the evolution of cloud computing become a revolution.

Refrences

[1]. *M. Armbrust et al., View of Cloud Computing," Comm. ACM, vol. 53, no. 4, 2010, pp. 50-58.*

[2]. *T. Richardson, Teleporting—Mobile X Sessions,‖ Proc. 9th Ann. X Technical Conf., Jan. 1995. Also in The X Resource, Issue 13, O'Reilly & Associates, Jan. 1995. Also available as ORL Technical Report 95.5, ORL, Cambridge CB2 1QA, England.*

[3]. *M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. A view of cloud computing. Commun. ACM, 53:50–58, April 2010.*

[4]. *P. Wang, W. Huang, and C. A. Varela, of virtual machine granularity on cloud computing workloads performance,‖ in Workshop on Autonomic Computational Science (ACS'2010), Brussels, Belgium, October 2010, pp. 393–400.*

[5]. *T. Richardson et al., Network Computing, "IEEE Internet Computing, vol. 2, no. 1, 1998.*

[6]. *Microsoft, Desktop Protocol, " 2009; http://msdn.microsoft.com/en us/library/aa383015.aspx.*