Cross-Platform Service for Nomadic Devices in Biodiversity Research

M. Aslam, M. Ali, Syed Ahsan, M. Junaid Arshad, Amjad Farooq, M. Shahbaz

Department of Computer Science and Engineering, UET, Lahore, Pakistan. maslam@uet.edu.pk

Abstract: The synergy between cloud and virtualization has become popular in the recent years. Different handheld devices come with versatile and heterogeneous hardware and operating systems, resulting in incompatibility issues for application users. We extend the concept of virtualization to provide a set of virtual machines capable of emulating major operating systems. These virtual machines run on a Virtual Box Web Services in a SOA and are designed to be tailored to meet the specific requirements of a user through which they can run any software on any handheld device. Our service provides the user with the ability to make and run major operating systems using cloud of virtual machines. This allows unlimited portability between different hardware and software architectures if some minor requirements are met. We present the result of initial testing of running Nokia Symbian application on an android device.

[M. Aslam, M. Ali, Syed Ahsan, M. Junaid Arshad, Amjad Farooq, M. Shahbaz. **Cross-Platform Service for Nomadic Devices,** Life Science Journal. 2012;9(1):603-609] (ISSN:1097-8135). http://www.lifesciencesite.com. 89

Keywords: Handheld Device, Cloud Computing and Virtual Machine

1. Introduction

Cloud computing has somewhat succeeded in its ability to provide versatile services to users irrespective of the device hardware (e.g., storage, computation, and architecture constraints) and software resources like operating system. The system running inside the cloud is responsible for all computing intensive work [1]. Since, cloud computing is not processor intensive and yet provides all services of a business or personal needs, this has encouraged hardware manufactures to make lightweight, inexpensive, and low cost devices with just enough capability to run a web browser, the best examples are Netbooks and Tablets. In the future, we expect devices to become even smaller and there would be an increasing trend towards using internet on handheld devices like Tablets [2]. Cloud usually follows the economic model in which a user can add and remove resources dynamically on the cloud and therefore pay only as much as he needs [3].

Scope of problems that can be solved by cloud computing is immense for e.g. with cloud, we can have unlimited storage capacity by adding clouds of storage services as necessary and as our business grows. Similarly, we can have the illusion of super computer on our device by using cloud as a utility computing. Amazon EC2 [4] provides dynamic scalability of the cloud within minutes. Besides numerous advantages, cloud computing also provides mobility and collaboration to end users [5]

Traditional software like Word and Paint have now online cloud counter parts that can be used freely using just a web browser like Google Docs, adobe Photoshop express, online conversation services like doc to PDF and back, Online tools to create virtual machines, etc. When software is used in this way it is called "software as a service (SAS)", an integral part of the cloud architecture.

Virtualization is an art of dividing the computer into many virtualized systems each with its own hardware and software architecture. Hardware virtualization tools such as Intel Virtualization Technology [6] assist virtualization on Intel machines. Virtualization was once confined to large machines and now it is being available on a micro scale. Handheld users having some hardware or software constraints can now get benefit from a cloud virtual machine in order to perform any resource extensive task.

Till now virtualization for handheld devices has not acquired widespread due to low computation power and limited capability of a typical handheld device. But, gradually we have seen an increasing number of handheld devices with huge computation power for e.g. smart devices.

The problem that we tackle in this paper is of using cloud as a virtual platform for running heterogeneous mobile platforms.

There are many existing cloud services like Amazon EC2 [4], Google App Engine [6], Facebook Platform [7]. Amazon E2 provides a business cloud platform while Google App Engine provides its users with a free limited space to run their application written using python or java language on the Google engine. Facebook Platform allows users to run applications on the Facebook platform and taking advantage of the Facebook API. Facebook platform is not suitable for developing enterprise and utility applications.

Currently in most of the popular handheld devices like Android [8] and iOS [9] a user could only run the application for which the device has been

programmed for. This introduces several problems for users and also makes it difficult for them to switch to other device with a different architecture and platform. Legacy software like critical business and enterprise applications can't be ported to other handheld device with a different architecture. These problems have motivated us to solve it using a cloud architecture which is same for any handheld device.

Different mobile devices like Blackberry, Android and IPhone are in intense competition with each other. Though, the present trend is towards Android [10], other platforms will remain significant. There are millions of people using those platforms and in the future, we expect some other mobile platform. All these platforms are not at all compatible with each other as they are strong rivals of each other.

Millions of applications exist for each of these platforms and they are not compatible with each other. The ultimate burden is on the user community to live with this limitation. For example, a user of android device has no way to run IPhone application except by hacking the android platform or by creating a virtual IPhone device machine. We follow the second approach to implement our solution. Unfortunately it's not possible to create a virtual machine on a cellular device since it has its own memory and process limitation. But we can implement a virtual machine on a cloud that we called an emulator since its purpose is to emulate an operating system for running some handheld device file. Instead of creating a virtual machine for IPhone or Android Phone, we have created a generic virtual machine which can host any major operating system depending on the needs and requirement of a user.

Each of the virtual machine in our cloud acts as an emulator for a particular handheld device. The service theoretically allows users to run major mobile operating system files like Android, Blackberry, and IPhone IOS. Our virtual machine has been enhanced specifically to work with mobile platforms.

By using our solution an android user would be able to run IPhone applications on their device without even the web browser. Our cloud service would consist of two parts, a client and a server. A client run on the device and server reside on the cloud. Whenever, an attempt is made to run the IPhone application in an android phone, our client application detects this and asks the user about whether they want to run this application by using the online cloud. By approval from the user, application would be compiled and run on the VM residing in cloud and its GUI would be transferred at run time to the mobile device. It is important to note that only the computation part runs on the virtual machine since an application which requires camera can't be simulated on the virtual machine. Therefore, a client application must act as a bridge to provide transparent camera interface to the virtual machine and to the user. In other words, all the hardware of the phone like camera, microphone, and accelerometer must be simulated on the client side. This adds extra complexity in implementing such a cloud which we shall see in the proposed solution section later on.

The motivation for RunIT Cloud comes after searching the Google for a service which can run cross-platform mobile code e.g. sis file, so when we Goggled an online service to run sis files, we couldn't find any online service which can assist users in running mobile files. This is how we developed the idea in the first place.

The rest of this paper is organized as follows. In section 2 we highlight and compare some existing cloud services similar to ours. In section 3 we present our proposed solution and subsequent details in subsections. In section 4 we present some of the results of using our service to run Symbian Operating system sis file on an Android device. Finally in section 5 conclusions are presented followed by references.

2. Related Work

GoPC [11] is an online service that provides its users the facility to access their systems using clouds. They can use their machines as well as use the software that is available on the cloud. The users can access the service using their handheld devices as well. Together with online storage, scheduled backups, and the same 128 Bit encryption used by most banks ensures the highest level of security. You can build your own cloud platform from GoPC's products and services. However it provides access to the complete desktop and may cost more than services which provide access to only one service like the online spread sheet. Additionally GOPC consumes more bandwidth than the service which provides limited functionality. RunITCloud service provides the same interface as available on the handheld device and is accessed only when a non-native program is executed on the device. This gives handheld device users the illusion that they are able to run any program on their device. The RunITCloud is an attempt to run non native programs on the handheld devices with the comfort of the native GUI whereas GOPC provides a desktop environment to access common services

Amazon Web Services (AWS) [12] also provide the Elastic Compute Cloud (EC2), in which customers pay for compute resources by the hour, and Simple Storage Service (S3), for which customers pay based on storage capacity.

Other utility services include IBM Blue Cloud [13] which provides services very similar to

the Amazon E2, EMC's recently launched storage cloud service, and those offered by startups such as Joyent and Mosso.

Layered Tech's virtual machines (VM) [14] provide high-performance, high-availability computing resources that are not confined to a single, vulnerable server. With a virtual machine (VM) your site or application is implemented across many nodes. SnowFlock [15] is another system that uses the fork command for creating multiple virtual machines. A user can create machines according to their needs. We have used SnowFlock for implementing virtual machines in our system.

Intel virtualized computer [16] is another technology that provides the functionalities of creating VM's. It provides facility of creating servers with security and flexibility.

As we can see cloud services and platforms are on the rise. Many services and platform exist which cover the common services from document editors to graphics application to pc desktop sharing but none of these services and platform provide an explicit ability to provide a cloud emulator to run non-native mobile files.

2.1 Google Trends

If we look at Google trends for the popular mobile operating systems like Android, Symbian, IOS, BlackBerry and Java Me we get the results shown in Fig 1.

Fig. 1: Google Trends for Popular Mobile Operating Systems

It is obvious from the Google Trend that Android is the most popular mobile OS in 2011 with a sharp steep curve. Very close to Android is Blackberry with a curve steadily going up from 2004 to 2010 but decreasing abruptly in 2011 and just below the Android. It should also be noted that the popular IOS operating system has not seen any worthy growth over the years and lags behind Android and Blackberry by a significant factors. Finally Java Me seems to in its last breath as the interest of user community in Java ME has become negligible.

3. RunItCloud

Instead of running the program on the handheld device, we propose that the program should run on a cloud service and only its interface should be accessible from the handheld device. This can theoretically give illusion to a mobile user that the software is actually running on the embedded device. However this solution comes with its various technical and feasibility challenges for e.g. how different events should be propagated from the mobile device to the cloud service and vice versa. Can the user interact with the software interface being streamed from a cloud service? If so what would be the screen refresh rate for a smooth interface playback. Can we stream video quality refresh rated on a limited handheld limited bandwidth, memory and computation power. How would the system compensate for the different handheld display sizes and resolutions? .Before we attempt to answer these important questions, we want the reader to have some reality check. Our solution is not the ideal one, it's in no way alternative of the real device, and it attempts to provide a solution so that software written for platform A can run successfully on platform B provided there exist a Virtual Box service for it.

We propose a service oriented architecture to provide a set of services each running an instance of VirtualBox Web Services as shown in Fig 2.

VirtualBox is a popular open source virtual machine solution through which we can run many different operating systems on a single host computer. Solutions like VirtualBox abstract the hardware and operating system of the host computer into several executing environments. Very similar to VirtualBox is a commercial software solution called VMware. However VMware is not an open source solution and is therefore not suitable for customizing it to our needs. VirtualBox on the other hand provides comprehensive support and access to its internal Virtual Box Main API. Using VirtualBox Main API an application user could develop a program which can create, run and delete virtual machines on the fly and also interact with the virtual machine guest operating system by providing a full set of API. Besides that Virtual Box provides Web Services which provide full access to the Virtual Box Main API.

Fig 2 shows the architecture of the RunItCloud, a handheld device user downloads a portion of the RunItCloud client program which

provides mobile specific implementation to access the RunItCloud service. A mobile specific program is responsible for discovering the services of the mobile like GPS, Accelerometer, and Cam etc and determines whether they are compatible with the service. In other words the client portion of the RunItCloud aims to provide a bridge between the mobile and the RunIt Service. In Fig 2 three layers of services are shown and only the top layer "Service 3" is visible. A mobile user can't directly access any of these services. These layers of services are added for flexibility and load sharing purpose. Client program can only access the main server which contains the core implementation of the RunItCloud Service. The main server is responsible for dynamically allocating available web service to the mobile user and keeping track of which mobile web service is being used by a specific mobile device. This provides the additional architecture flexibility of the SOA as new VirtualBox web services could easily be added without any significant implementation and architecture changes. Each VirtualBox web service container can hold one or more virtual machines each running a specific operating system.

VirtualBox web service provides access to the major desktop operating systems like Windows, Mac, and Linux and mobile operating systems like Android, iOS. Most of the mobile versions running inside VirtualBox are only a stripped up version containing only the major features with core kernel, applications and GUI.

3.1 System Architecture

Our cloud architecture (see Fig. 2) is based on Service oriented Architecture. There are number of services each with its own VirtualBox web services instance. A client such as an Android device or a PC can only access the service through a RunItCloud main server.TCP/IP indicates that connections could be either wired or wireless. Initially a client connects to the Main Server which is running the main instance of the RunIt system. Main Server is hosted on a public IP server and can be accessible by using the host name and port number. This system is responsible for selecting the appropriate service and routing the client connection to the appropriate service. A client is unaware of the service it is using. This architecture is flexible; more services can be added on demand to satisfy a large number of users. Each service can only contain a limited number of virtual machine instances and each virtual machine instance can only run a single operating system such as Android OS or windows. Fig 2 shows detailed cloud architecture in which some consumer device (mobile phones, laptops, etc.) are consuming one of the cloud services each running with its own virtual machine and running on the cloud container.

Each service has its own instance of the Virtual machine which is shown by rectangular layers of screens and works independently.

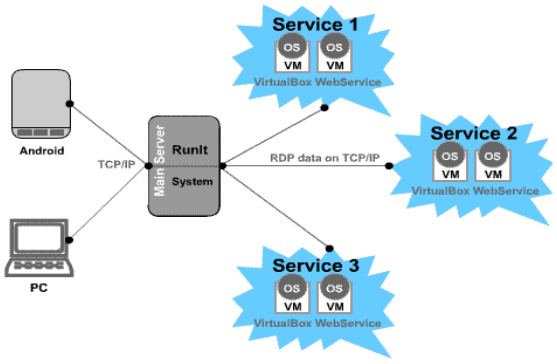

Fig. 2: RunItCloud Architecture Diagram

3.2 System Components

RunItCloud is divided into components as shown in Fig 3.The top layer is the Main API interface and provides access to core features of the system. Clients and users could only access this top layer to use the RunIt services.

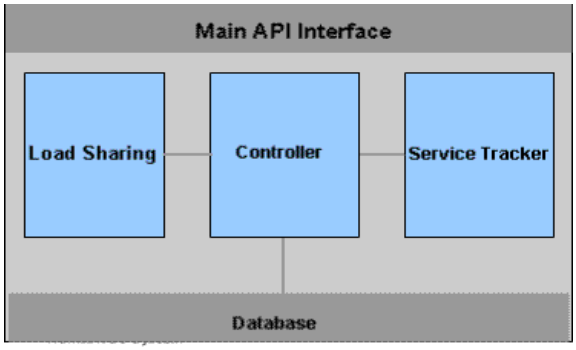

Fig. 3: RunItCloud Components

Main system consists of Load Sharing, Controller and Service Tracker. Load Sharing is responsible for dynamically sharing the clients load over the services so that the system appears to run smoothly when the traffic increases. Running a virtual machine on hardware is an extremely expensive process and it's not possible to run many virtual machines on the same physical hardware. Load sharing provides the necessary flexibility so that users can enjoy a continuous reliable service by dynamically consuming the next free service to provide the maximum throughput.

The Controller is the center hub and performs the controlling functions on the service tracker and load sharing. It is responsible for

managing other components. Finally Service Tracker is responsible for profiling and keeping information about which service has been assigned to which client. Service Trackers also provide cache of the programs that the client has used previously and Controller first checks with the Service Tracker to determine whether the program should be transferred from the client to the web service. Database is used to provide storage, service related and configuration information and is directly accessible from the controller.

3.3. The Display

The RunIt service can be used to emulate different kinds of operating systems and versatile type of software such as real time systems and games can be requested by the client devices. Since the program that user is trying to run is incompatible with the user native device, therefore it must run on one of the RunIt services. If the RunIt service recognizes the software and runs it successfully, it must send the result back to the client. Naturally, the result is an image containing the GUI of the program. However it's not just an image, it's a series of images that are sent at regular intervals based on the following algorithm.

i). Determine the type of the program game, simulation or a static application.

ii). Intelligently calculate the refresh rate of the virtual machine screen by noting the movements at a video rate. This is done by continually scanning the program after every 50 milliseconds and noting down the difference from the earlier image sequences. If no difference is found the program is probably an application and only has static content waiting for the user to interact with it.

iii). For a static software, RunIt service only send a single image and waits for some action from the client side such as click of a button. For video, it continually transmits 25 frames per second or less depending on the buffering rate of the client internet connection. This rate can be manually adjusted by the mobile user or determined by benchmarking the user mobile internet bandwidth.

The display is therefore based on the illusion that the application is running in real time on the client device when actually it's just a streaming. This approach may seem quite complicated, but in reality it is quite feasible due to the faster internet mobile internet connections.

3.4. The Client Part

The client part is small software written specifically for each of the mobile platform and contains the platform specific logic. The client part acts as a bridge between the cloud service and the user mobile device. It's a connector to the advance features of the RunIT service as it provides interface to the user mobile storage access, I/O device access like camera, microphone, GPS and accelerometer. Device hardware for e.g. GPS is associated with a java interface and a platform specific client program must implement the interface to allow the main server to access the hardware features. Standard interfaces for hardware devices have been provided to on the service as well as on client side. To illustrate it let us consider a user tries to run a non-native program which needs the GPS to work properly. For example program may need a GPS location to pinpoint the location of the nearest restaurant. Obviously GPS can't be emulated on service side. Therefore, such information must come directly from the consumer hardware device in this case a GPS. The server ends a GPS request to the client, the client either denies or accept the request if GPS is available and has been implemented by the client program. The serve uses the GPS specific interface to request any feature of the GPS hardware on the client side. The result of the GPS hardware is sent back to the server periodically. The client actually starts a separate thread as a callback function and also uses the timer to periodically sent the device specific data. A separate client/server connection is established for each hardware device to be used. The client part handles all the low level details and provides the GPS data to the service whenever the service requests it to do so. Currently client part is available for only two mobile platforms that is Android and BlackBerry.

3.5 Remote Desktop Streaming

Once a client program requests a non-native program to run on the cloud service, the main server runs the program on a free service and starts the virtual machine. The interface of the virtual machine then must be streamed over a TCP/IP connection back to the client. Fortunately Virtual Box web control provides a component called VRDB Server whose purpose is to provide a remote desktop access to the RDB client. Virtual Box VRDB also supports the keyboard and mouse events interaction from the RDB client. We implemented the RDB client on the client part of the RunIt cloud to stream the remote desktop.

3.6 Program Run Information

Program run information is a set of fields extracted by RunIt service and contain general information about the device and Platform. The device information includes the type of device, the

manufactures of the device, the SIM information for the device, etc. Platform information consists of operating system, supported file types, version, and supported hardware features like GPS, CAM. It also includes information about the SDK of the platform for which the program is written. This information is used to prepare the system to determine which OS emulator should be called upon.

3.7 Mechanism

What exactly happens when a user tries to run a non-native application? Simply starting a virtual machine with a specific operating system running is not very useful as the operating system can't access the user mobile program and therefore can't run it. Ideally a user should be able to click on the non-native application and somehow the application should start running automatically on the user device. This is accompanied using the following steps.

i). User clicks on the non-native application or browse it using the RunIt client software.
ii). RunIt client tool determines the device, program run information etc.
iii). Client tool connects with the RunIt Main server and checks with the Service Tracker component if the program code is available on the service. If the program code is available move to point 5.
iv). Client tool establishes a file transfer session with the main server and transmits the program code on the main server. Main server saves the program name and version information in its internal database.
v). Main Server then shares the program code with the service Virtual machine so that it is visible to the guest operating system. This uses the VirtualBox shared folders feature which are provided exclusively to shared data between guest and host operating systems.
vi). Using VirtualBox SDK Events Management, the main server simulates the click event on the program file in the shared folder to start the program on the virtual machine.
vii). Main Server configures the VirtualBox VRDB server to transmit the remote desktop of the virtual machine to the client device.

If all of the above steps are successful a user of the client device gets the illusion that the program is actually running on the client device when it's in fact running on the cloud VM.

4. Use-Case: Run Android On Iphone

We have tested our system for mobile platform interoperability. The device which was used as the host platform was HTC Android G2 cell phone as shown in Fig 4.

Fig 4: RunIT Cloud Client Application

The goal was to run the Symbian mobile application with extension "SIS" on this phone. The phone had a 400-600Kbit/s EDGE based internet access enough to stream a 3 inch video at low detail. As shown in the Fig 4, user clicks on the browse button and selects the sis file. By default any user who tries to run this file gets an error message that device does not support this file.

The client part of our service can be downloaded for each different mobile operating system and is optimized to run on that platform. After installation of the client part and assigning it necessary internet download permission, we again try to run the sis file. This time we get a prompt asking the user that running this file may cost you air time. With permission from the user, the client part prepares HTTP connection and connected to our RunIT service passing it the device information and program code. The RunIT service determines that the device operating system is Android and program code can be run in Symbian emulator, it emulates the suitable version of the Symbian emulator and calculates the refresh rate of the screen from which it determines that it doesn't contain much screen refreshes and runs it in application mode by transmitting one file after every 2 seconds threshold which can be adjusted. It important to note that the screen refresh calculation which is simply a standard motion detector between two or more consecutive screen frames is done entirely on the server side and is not visible to the client. However client part must be able to receive data at any time as in order to handle varying chunks of data from the server. This is due to when most of the application starts, this first screen is mostly a static interface waiting for user to

interact with the .Due to this reason, and our system fails to classify the application as an interactive process and transmits a single screen after every 2 seconds.

However there is a catch that we have incorporated in the system. After 50ms the system again checks the refresh rate of the screen using a difference between previous screens and the new screen for a specified time to calculate the screen repaint rate. This time it classifies the application as an interactive process based on the varying screen repaint rate. As a result the system sends a stream of screens after 33 ms to meet the 30 frames/sec requirement of the smooth video. Then, after every 50ms the system again evaluates the program to see if it has changed its behavior to a static or dynamic program. This is important because we want to make the system efficient and send the screen only when it is required based on principle of least privilege resulting in less bandwidth usage.

So based on the application the system intelligently determines the repaint rate and accordingly classifies the running file. However there is no formal classification of the application other than interactive or background processes. The goal is to determine the FPS of the application so that only necessary frames should be sent back to the handheld device saving bandwidth and computation.

5. Conclusion

RunIT Cloud service allows users of different platforms to run each other software using a set of virtual machines running on the cloud supporting major operating systems. For instance, an android user can run Symbian applications on his device without a dedicated Nokia set. Our cloud service consists of two parts, a client and a server. The client runs on the device and server reside on the cloud. This is a new approach as it allows the user to run non-native application from the comfort of the native GUI of the device. However there are obvious challenges to this approach, the most notable is to find a suitable version of the operating system image which can run the file user is trying to execute. Also the Operating system image should be compatible with the Virtual box system. The use case demonstrated our solution can be used to solve real world problems. Additionally the experience of people who were deliberately not told about the emulator was noted and necessary steps were taken to make the system feel more real. In future, the support of more operating, statistics and usage feedback of the service would be added.

Acknowledgments

This research was supported by the Directorate of Research Extension and Advisory Services U.E.T., Lahore-Pakistan.

References

[1] Michael A., Fox, A., Griffith, R., Anthony D., Randy, J. K., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., Zaharia, M., "Above the Clouds: A Berkeley View of Cloud Computing"., Technical Report EECS-2009-28, EECS Department, University of California, USA., Feb. 10, 2009.

[2] Dikaiakos, M. D., Katsaros, D., Mehra, P., Pallis, G., Vakali, A., "Cloud computing: Distributed internet computing for it and scientific research, Int. J. of IEEE Internet Computing, Vol. 13, No. 5, pp. 10–13, published by IEEE Computer Society, USA., 2009.

[3] Dean, D., Saleh, T., "Capturing the value of Cloud Computing", white paper available at http://www.bcg.com, November, 2009.

[4] http://aws.amazon.com/ec2/

[5] Hayes, B., "Cloud computing", Communications of the ACM, vol. 51, no. 7, pp. 9–11, July 2008.

[6] Ciurana, E., "Developing with Google App Engine" published by Apress, Berkely, CA,USA, 2009.

[7] Hickey, M., "Facebook launches Facebook Platform; They are the Anti-My Space", available at http://www.techcrunch.com/2007/05/24/ facebook-launches-facebook-platform-they-are-the-anti-myspace/ consulted on September 07, 2011.

[8] Rogers, R., Lombardo, J., Mednieks, Z., Meike, B., "Android Application Development: Programming with Google SDK", edition 1, published by O'Reilly Media, UK., May 20, 2009.

[9] Patel, N., " iPhone OS 4 renamed iOS", available at www.engadget.com retrieved on June 09, 2010.

[10] Chakraaborty, P., "Android to Become No. 2 Worldwide Mobile Operating System in 2010 and Challenge Symbian for No. 1 Position by 2014", blog posted at http://www.pctelecoms.blogspot.com on on Sep. 12, 2010.

[11] http://www.gopc.net accessed on April 2011.

[12] http://aws.amazon.com accessed on Feb. 19 2009.

[13] http://www.ibm.com/cloud-computing/us/en/ access on April 2011.

[14] http://www.layeredtech.com/cloud-computing/virtual-machine-hosting/ retrieved on April 25, 2011.

[15] Andres, H. L. C., Whitney, J. A., Scannell, A. M., Patchin, P., Rumble, S. M., Lara, E., Brundo, M., Satyanarana, M., "SnowFlock: Rapid Virtual Machine Cloning for Cloud Computing. In proceeding EuroSys' 09, 4th ACM European Conference on Computer System, held in Nuremberg, Germany, March 2009.

[16] Uhlig, R., Neiger, G., Rodgers, D., Santoni, A. L., Martins, F. C. M., Anderson, A. V., Bennett, S.M.., Kagi, A.., Leung F.H., Smith,L., "IntelVirtualization Technology", IEEE Computer, vol. 38, no. 5, pp. 48-56, May 2005.

2/9/2012