Ceaseless Virtual Appliance Streaming

By
Shahid Nawaz
NUST-2012-60759-M-SEECS-60012-F

Supervisor
Dr. Asad Waqar Malik
NUST-SEECS

A thesis submitted in partial fulfillment of the requirements for the degree of Masters of Science in Information Technology (MS IT)

In
School of Electrical Engineering and Computer Science,
National University of Sciences and Technology (NUST),
Islamabad, Pakistan.

(July 2016)

Approval

It is certified that the contents and form of the thesis entitled "Ceaseless Virtual Appliance Streaming" submitted by Shahid Nawaz have been found satisfactory for the requirement of the degree.

Advisor: Dr. Asad Waqar Malik

Signature: _____
Date: _____

Committee Member 1: Dr. Muneeb Ullah

Signature: _____
Date: _____

Committee Member 2: Dr. Anees-Ur-Rehman

Signature: _____
Date: _____

Committee Member 3: Dr. Arslan Ahmed

Signature: _____
Date: _____

Abstract

Cloud computing is a sort of Internet computing which offers the shared processing of applications and data to computer devices on demand. The procurement, maintenance, and up-gradation of the resources is the responsibility of cloud provider. The client can utilize the cloud services on pay-*as-you-go* basis. It offers flexibility, efficiency, competitiveness, and swift disaster recovery along with the reduction in capital and maintenance for the client organizations. The cloud-based virtual appliances can be exploited to solve the issues of software maintenance, mobility, hardware compatibility, and security. The virtual appliance is an image file of virtual machine which is created by integrating single software application with intended components of operating system to run it optimally on diverse hardware infrastructures. Clouds generally stores virtual appliances over network attached storage. It generates scalability bottleneck because when user tries to access virtual appliances from cloud, it triggers hundreds of megabytes of data reads and subsequent network congestion problem.

The fundamental question arises, how to initiate a virtual appliance and to load its application in a minimum time taken into consideration the transmission capacity of the shared networks. Streaming is probable solution to this critical issue. As streaming techniques associated with the video-on-demand, where data is integrated into frames and assures in-order delivery are available, no streaming scheme offers proper solution to overcome the complexity of streaming virtual machines over shared networks. The novel Ceaseless Virtual Appliance Streaming system offers virtual machine's streaming over Internet in par with video-on-demand streaming. It helps to reduces burden over computing resources and internal networking nodes to enhance network resource utilization. When applied over real networks it exhibits better efficiency over traditional legacy systems where it was mandatory to download virtual appliance before its execution over a local machine.

Certificate of Originality

I hereby declare that the research paper titled "Ceaseless Virtual Appliance Streaming" my own work and to the best of my knowledge. It contains no materials previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any degree or diploma at NIIT or any other education institute, except where due acknowledgment, is made in the thesis. Any contribution made to the research by others, with whom I have worked at NIIT or elsewhere, is explicitly acknowledged in the thesis.

I also declare that the intellectual content of this thesis is the product of my own work, except to the extent that assistance from others in the project's design and conception or in style, presentation and linguistic is acknowledged. I also verified the originality of contents through plagiarism software.

Author Name: Shahid Nawaz

Signature: _____

Acknowledgement

My labours bore fruit with successful completion of this project by the grace of Allah Almighty. I would like to extend my sincere and heartfelt obligation towards all the personages who have helped me in this endeavor. Without their active guidance, help, cooperation & encouragement, I would not have made headway in the project.

My special thankfulness to the Supervisor Dr. Asad Waqar Malik for sparing his valuable time in putting together all the bits and pieces I brought in as raw data to him, his contribution towards the success of this project is unmatched. I have to appreciate the guidance given by Dr. Muneeb Ullah, Dr. Anees-Ur-Rehman, Dr. Arslan Ahmed, Dr. Junaid Qadir, Dr. Zahid Anwar, Dr. Hamid Mukhtar, and Dr. Raihan-Ur-Rasool, to their comments and advices.

My special thanks to my parents and family for their support and understanding and of course for bearing with me when I was all too busy with the project.

Shahid Nawaz

Contents

List of Abbreviations:

Abbreviations	Descriptions
Arpanet	**Advanced Research Project Agency Network**
CVAS	**Ceaseless Virtual Appliance Streaming**
DISA	**Defense Information Systems Agency**
DMTF	**Distribution Management Task Force**
DSA	**Defense Communication Agency**
DSL	**Digital Subscriber Line**
FPS	**Frame Per Second**
ISP	**Internet Service Provider**
ISS	**Intelligent Streaming Server**
JeOS	**Just enough Operating System**
MIT	**Massachusetts Institute of Technology**
NAP	**Network Access Points**
NORSAR	**Norwegian Seismic Array**
OVF	**Open Virtualization Format**
PoP	**Point of Presence**
RTCP	**Real-Time Transport Control Protocol**
RTP	**Real-Time Transfer Protocol**
RTSP	**Real-Time Streaming Protocol**
SaaS	**Software as a Service**
SRI	**Stanford Research Institute**
TCP/IP	**Transmission Control Protocol/Internet Protocol**
UCLA	**University of California, Los Angeles**
USDoD	**US Department of Defence**

List of Figures

List of Tables

1 INTRODUCTION

1.1 Introduction

Virtual machines (VMs) are executed by virtual machine monitors (VMs) or hypervisors from onboard VM images. With the combination of host-level virtualization and storage area network facilities, the cloud computing offered extensive variety of new applications for the VM technology [1]. The virtualized public and private clouds facilitates the organizations to transfer their operation from their dedicated hardware to the shared infrastructure having virtual servers. It resulted into proficient hardware utilization, simple management, and faster disaster and failure recovery [2]. But the system must download the complete image of the virtual disk the execution of the virtual machine. But trying to execute larger numbers of VM instance may resulted into network congestion. Foremost impediment in the popularity of virtual appliances is that they could not be directly streamed from the server to the client. Therefore, it is necessary to device a mechanism for the management and transmission of the virtual appliance in a similar fashion as the video file. In this way, with the transmission of a smaller chunk of a virtual appliance from server to the client, the user may be able to begin the execution of the virtual appliance. According to principle of factor scarcity, eighty percents of the users utilize only twenty percent functionality of the software [3]. Therefore, it is waste of the expensive network resources if large virtual appliance files would be transmitted from the server to the client. It is important that the streaming server must predict which subsequently function or chunk of a virtual appliance will be required by the user and to transmit the desired chunk instead of the complete virtual appliance file [4].

Virtualization means the creation of a virtual version of operating systems, hardware platforms, storage devices, and network resources. Virtual machine, as the vertebral column of virtualization, is a software emulation to facilitate the installation and operation of diverse types of operating systems as guest over the host operating system [20]. It relies upon a middleware virtualization layer called the hypervisor to allocate

hardware resources to the guest operating systems. Host system is transparent to the guest OS and it functions in a similar fashion as it is directly installed on the hardware. It aids the execution of those applications on computing devices which are inherently not compatible to the host operating system. Numerous virtual machines can be installed on a single host operating system provided there is enough storage capacity available. The prime advantage of the virtual machine is its security and platform independence. It can be executed on diverse types of computer devices regardless of the disparity in hardware architecture.

Virtual Appliance (VA) is an image file of the virtual machine consisting of pre-configured operating system and a single application [21]. It is not a complete virtual machine but an image file which is intended to run on a virtual machine having **type 1 or type 2 hypervisor platform**. It usually hosts single application. It functions like an application but is managed as a virtual machine. Idea behind the concept of virtual appliance is to create and deliver a readymade package of an application and operating system for its easy installation and operation over non compatible devices on network. Only those components of the operating system which are essential for the support and execution of a single application is included in the image file. As the necessary operating system components are embedded in the virtual appliance, it can be operated on diverse types of architectures. For instance the game developed for Windows operating system can be played on Linux with the same ease. Virtual appliances can be categorized as the closed and open virtual appliance. The closed virtual appliances are configured and distributed as indestructible units. Whereas, users have option to modify the open virtual appliances according to their requirements through web page user interface.

1.1.1 Virtual Machine

The virtual machine is software which allows the different operating systems to run as a guest over the host operating system by using a virtualization layer called the hypervisor. It is a very helpful technique when a user wants to run an application on his computer which is not compatible to his host operating system. Virtual machine allows a user to install guest operating system of his liking over which he can install the applications. For

example, a user can install the Microsoft Windows 7 as a guest operating system over the Ubuntu 14.04 LTS host operating system by using virtual machine software such as the 'virt-manager'. With the installation of the Microsoft Windows 7, the user can install the Microsoft Windows specific software such as the Microsoft Office 2010. Now he can use the Microsoft Excel and Word data processing software and at the same time retain the Ubuntu 14.04 LTS as the host operating system. The virtual machine creates a virtualization layer between the host and the guest operating systems and allocates virtual hardware resources to the guest operating system. The guest operating system does not know about the presence of the host operating system between itself and the hardware layer. It works in similar fashion as it is directly installed on the hardware. The virtual machine provides the virtual hardware to the guest operating system like virtual CPU, virtual memory, and virtual network interfaces. The virtual hardware is mapped to the real hardware. For example the virtual hard disk is stored in a file in the real hard disk. The virtual machine can be directly access remotely using its virtual network interface.

1.1.2 Virtual Appliance

The virtual appliance is defined as the image file of a virtual machine which contains pre-configured operating system and a single application. The basic idea behind the creation of a virtual appliance is that application will be delivered and operated in an easy and simplified manner [9]. To achieve this purpose, only necessary components of operating system which are required in the execution an application are included. They can be installed as a virtual machine which is running on some virtualization programme such as virt-manager, VMWare, Citrix,and so on. The deployment of virtual appliance also take care of the issues related to the installation and configuration of drivers. A single file is delivered to the user, who can download it and run its intended application. In this way resources which are required for maintenance of virtual machine also reduced. The virtual appliances have proven record of excellence in the deployment and maintenance of network applications. It is also helpful in Software as a Service (SaaS) model due to its simplicity of deployment and execution.

The virtual appliances are divided into two categories, i.e., closed virtual appliances and open virtual appliances. In first case the virtual appliance is configured and distributed as indestructible unit. In second case clients and customers have option to modify the virtual appliance. The developer can deliver the updated packages to the customers or can provide the interface on web server from where clients can directly configure the virtual appliance according to its needs and requirements. They are either directly provided to the customer in the form of a file or they can download it from the source website. The common file format which is vogue is Open Virtualization Format (OVF). Many popular virtualization vendors like Oracle, Microsoft, Ubuntu and so on supports it. The OVF documentations are published by the Distribution Management Task Force (DMTF). The file extension is ".ovf" The virtual appliances can be packaged and distributed as an OVF which is an open standard format. It allows the virtual appliances to be run on virtual machines. The OVF is not vendor specific. Large variety of processors and hypervisors this format as mentioned earlier. They are created and distributed as a ready-to-use system. It is just like a out of box solution. User only needs to download the single file and run its own virtual environment. All necessary operating system to run that application is already configured in it. This approach is really helpful in the cloud environment, where ready-made solutions are required.

1.1.3 Streaming

Streaming is modus operandi of unremitting presentation and deliverance of data from source to receiver. Media player launches audio or video file before complete video file has been downloaded from source server. Streaming as a delivery process is separate entity from media itself. If a client first downloads a same video file and then run it on the media player at his leisure time, it is not called streaming of the video. Delivery mechanisms are spontaneously fall into the category of either streaming or non streaming media. For example the television and the radio are inherently falls into the category of streaming media whereas the books and the CDs are inherently non-streaming media.

1.2 Motivation

Virtual appliances include variety of software and hardware emulations like routers, switches, firewalls, WAN optimization controllers, and software solutions. They are network based software appliance running on virtual machines. The virtual appliance developer could produce a package of application with the intended operating system and publish it across multiple platforms for use on for its use on *pay-as-go-basis*. Main advantage of VA is that they offer consolidation strategies by employing single software device to perform the functions which are generally performed by physical devices [22]. Moreover the cost of acquiring software-based virtual appliance could be lesser than the cost of procuring hardware-based physical devise having similar functionality and in various cases the cost of virtual appliance would be one-third of the cost of hardware appliance. The user can leverage the functionality and flexibility offered by the hypervisor to acquire and execute a system without any need to have second physical appliance. Therefore organizations could employ cost saving strategies by deploying more virtual appliances with the same cost with which they would install hardware appliances. The deployment of virtual appliances particularly benefits those organizations which have already disseminated serve virtualization technology at its remote offices and data centers. The large applications could be packaged into virtual appliance and then streamed them to the user through network. For instance a professional spreadsheet rental service could stream the spreadsheet application to the user whenever user needed to create, edit, or print the spreadsheet workbook. A state of the art virtual appliance streaming service would allow the user to swiftly access the application without any need to first download its codec and libraries.

1.3 Problem Statement

Existing virtual appliance streaming services are not suitable to be used in a current network environment facing high latency, jitter, and low bandwidth challenges. There is need to develop a new procedure to facilitate ceaseless streaming of virtual appliances over networks

1.4 Thesis Organization

The remaining portion of thesis is composed as follows: Chapter 2 is related to previous research oriented work and supportive study concerned with the streaming of virtual appliances over network. The Chapter 3 encompasses proposed architecture and research methodology for virtual appliances streaming. The Chapter 4 is related to the assessment of performance evaluation results. The Chapter 5 concludes the research and discusses about future work.

2 LITERATURE REVIEW

In this chapter we have presented the overview of the researches and techniques for the streaming of virtual appliance over network.

2.1 Profiling of Pieces:

The profile base execution pre-fetch approach focused on the creation of profiles of virtual machine image and the workload pair [6]. The streaming of virtual appliance depends upon the prefetching of pieces based on those profiles. Important phase in the creation of profile is the identification of those pieces of software which are not likely to be accessed by the user and to blacklist them for prefetching at the client end. Only those pieces will be entered into the profile that has been identified in at least one execution of the software. These profiles are specific to the virtual machine image and its workload. The workload was executed once or more times from the booting to the shutdown phase to identify the pieces which were accessed during the running of software. The time and order of appearance of were also recorded. Then the pieces were arranged in a list from earliest to the latest containing piece index, its average appearance time, and the number of executions in which it was identified. The profile is thus the ordered list of those pieces. The drawback of this approach is that important information may be lost during the averaging procedure. Moreover these profiles are associated with specific virtual machine image/workload pair and are not general purpose profiles. Therefore these

cannot be universally used on different types of workloads to figure out common access sequence. Prefetching the wrong pieces may not only stall the execution of software and also resulted in the clogging of the network.

2.2 Prefetching State Access:

According to prefetching approach, the key to successful streaming of virtual appliance depends up accurate prefetching hints [15]. These hints can be obtained by analyzing the access traces from the previous executions of the software. It proposed the creation of task-specific virtual appliance with focused work flow. The execution of virtual appliance commenced with the arrival of subset of the initial working set. As the execution progressed the missing state has to be demand fetched from the cloud. It resulted into stalls in system. A large number of stall may be perceived by the user as an irresponsive system. It tries to avoid such stalls by prefetching the VA state that will be expected to be accessed in the future course of action. Such predictions are made on the information collected through the prior execution of the virtual appliance. The insufficient prefetching knowledge may results into interruption during system execution. Sometimes correct prefetching knowledge may not be available in time to curtail the interruptions effectively.

3 DESIGN GOALS

Before the explanation of the working and architecture of CVAS, the design goals are taken into consideration:

3.1 Modular Construction of VA:

Our objective is to convert large-sized virtual appliance into small sized virtual packs for their swift transmission over real network. With current bandwidth capacity of real

networks, the large-sized virtual appliance would take from few minutes to several hours to download to client end.

3.2 Large-sized Virtual Appliances:

The size of virtual appliance is growing with the passage of time. Currently the size of virtual appliance varies from hundreds of megabytes to gigabytes and this trend is expected to continue in future. We therefore targeted the large-sized virtual appliance in our study.

3.3 Single-task oriented Virtual Packs:

We proposed the construction of single-task oriented virtual pack for ease of development and streaming. We concentrated on the uniqueness of virtual packs instead of imposing rigid constraints. The producer of virtual appliance would provide single task functionality in each virtual pack. According to principle of factor sparsity the 80% of the users only utilize 20% functionality of the software. Therefore single-task oriented virtual pack could cater the needs of large number of users.

3.4 Equal-sized Virtual Packs:

We targeted the creation of equal-sized instead of variable-sized virtual packs. The equal sized virtual packs are easy to handle and transmit over networks. It consumes less processing power of the streaming server and networking nodes. The creation of variable sized virtual packs would complicate the virtual pack construction process and therefore we left that question for future research.

3.5 The Swift Transmission of Virtual Appliances over Network:

We focused on the issue of swift transmission of virtual appliance over network. Our objective is to allow the hypervisor to access the virtual appliance as it is stored on the local host. The creation of scalable service oriented system is beyond the scope of this topic.

3.6 Streaming of Complete Virtual Pack:

Video streaming could tolerate the delivery failure of few frames to the users end. It does not seriously affect the running of the video as a whole. But to run the software it is necessary that complete image file will be delivered to the client end. Therefore we focused on the streaming of complete virtual pack before the execution begins.

4 CVAS STREAMING MODEL

Motivating force behind the concept of Ceaseless Virtual Appliance Streaming (CVAS) is to facilitate the interactive user friendly streaming of virtual appliances as comparable to video streaming. Therefore we proposed and developed a new virtual appliance streaming model that is based on the fundamental principle of video streaming.

4.1 Video Streaming:

Video file is consisted of sequence of still frames to be displayed one after the other to produce a moving image effect [7]. The video frames are independent entities and

damage or lost of some frames does not affect the running of video as a whole and only lost portion of the video may simply be unavailable to the viewer. As frames are independent entities, media player start running the video with the receiving of first few frames whereas rest of the frames may be in the network pipeline. If frames are received at higher rate than playback window, the excessive number of frames may be stored at buffer for future run. [8]. Media player does not require downloading entire file before running the video.

4.2 Virtual Appliance Streaming:

The streaming of virtual appliance over network cannot be attainable in the similar fashion as the video streaming. The basic difference is that a video is consisted of autonomous frames whereas virtual appliance is an image file of a virtual machine (VM). If a single chunk of image file is not received at client end, it could not function [9]. CVAS makes the streaming of the virtual appliance's image file feasible by first fragmenting it into small sized modules or chunks called virtual packs (VP) and then forwarding the required virtual packs to the client end. The virtual appliance which we have taken to illustrate the CVAS concept is the image file of the virtual machine of a single application of computer game and supporting operating system components. The game has composed of four phases but the user cannot play all of them at a time [10]. He must run the phases in a set of sequence from phase 1 (startup stage) to phase 4 (finishing stage) as shown in Fig. 1.

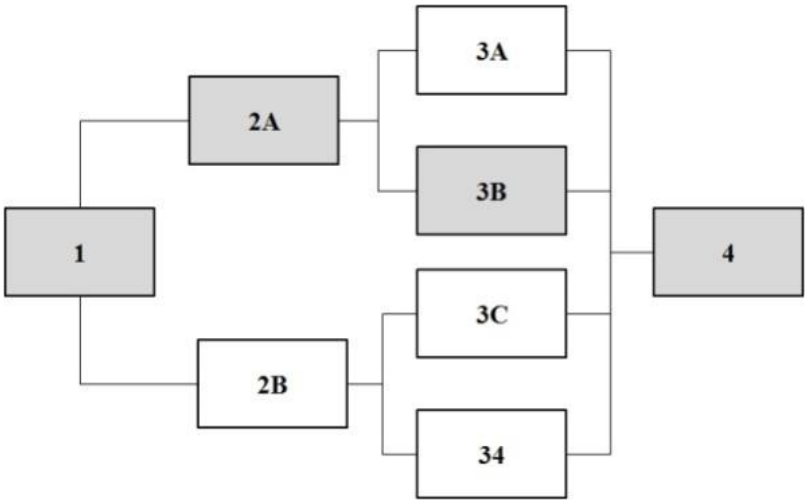

Fig. 1 Phases of gaming virtual appliance played shown by filled boxes

The parallel set of phases is also available to the user from which only one can be chosen. For instance after phase 1 user can select either phase 2A or 2B but not both. The phases of game played by the user are shown with filled rectangles in Fig. 1. It shows that user played only four phases of games out of total eight stages. According to the "law of vital few" it is wastage of network resources to transmit all eight stages to the client end. The CVAS project would convert the larger VA into eight independent smaller modules or VPs. Each virtual pack contains only one phase of the game. Each VP is a complete set of VM image fie independent of other VPs. To achieve this objective, a fundamental modification is required in the basic programming code of the computer game. As each VP contains only a portion of the game, it reduces the size of file that is to be transmitted at a single time. Therefore, the transmission of a VP consumes lesser bandwidth and processing power of internal nodes of network as compared to the entire VA file [11]. At the completion of a particular phase, user has been given option to advance to the next phase or to logout from the system. Accordingly, the subsequent phase may also consist of parallel phases. This phase selection process would continue till the game has been completed or the user would consumes all of his life lines. This process has similarity with the video files containing specific number of frames arranged in sequential order.

By employing traditional method of playing online games, user has to download the entire game before running it on the system. Depending upon network speed it takes from several minutes to few hours in order to download the entire game from server side. But by using CVAS project the user does not require to download the entire game at a time. He requires only the first phase of the game to start with. It is similar to the streaming of video file where only initial set of frames are required by the media player to play the video. Now instead of transferring the larger VA only the VP consisting of the first phase of the game has been delivered to the client end. Therefore user can start the execution of the virtual appliance by just receiving the smaller portion of the larger VA. When user reaches at the end of the Phase-1 and before moving on to the next phase there comes a decision point (DP) about the selection of the next phase as shown in Fig. 2. It depends upon whether the next phase is consisted of two or more parallel stages or is a single entity. If the coming phase is a single entity then streaming server would simply delivers the VP of the next stage. The matters become complicated when the next phase is consisted of parallel stages and user have to decide in the previous phase about the particular phase he wants to execute in future.

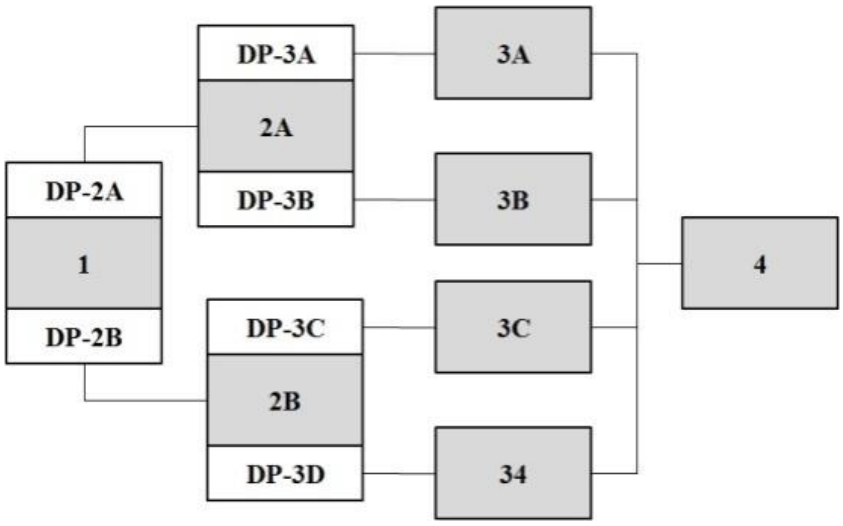

Fig. 2 Decision points (DPs) used to predict future user action.

To illustrate the concept, the DPs are placed at the end of each phase as shown in Fig. 2. For instance, if user have chooses decision point “DP-2A” at the end of Phase 1 then streaming server will send the VP containing phase 2A and so on. User will receive only that phase of the game which he opted for. It reduces the size of the file for its efficient delivery over the network.

4.3 CVAS Hypothesis:

We assume that producers of VA encapsulate a task specific workflow in each VP. Therefore each VP would be unique in nature. This single task hypothesis restrains to some extent the variety of interactions that a user will have with the VA. Nevertheless, VAs can offer variability in its functionality. For example, as described in our computer game VA example, there are different scenarios or phases, each having its own audio, video, and graphic data. Moreover, some VAs may use multiple processes to execute a task, and a large number of these processes could be multi-threaded. Therefore VA would be simply constructed on the principle of uniqueness instead of rigid constraints. For the swift booting of the system the RAM image may also be included along with the disk image by the producer of the VA. It is observed that less state is being accessed when RAM image is included instead of non inclusion of RAM image. We focused on larger VA instead of smaller ones as we expected that the size of the VAs will grow in size in future. To make the construction and streaming of VAs simple we do not focus on agility requirements like power consumption or scalability.

As observed earlier VA is an image of VM [17]. But instead of one large VA, several smaller VPs have to be created each containing portion of the VA. The VPs can be sub-divided into packets or wads for their rapid transmission over networks [18].

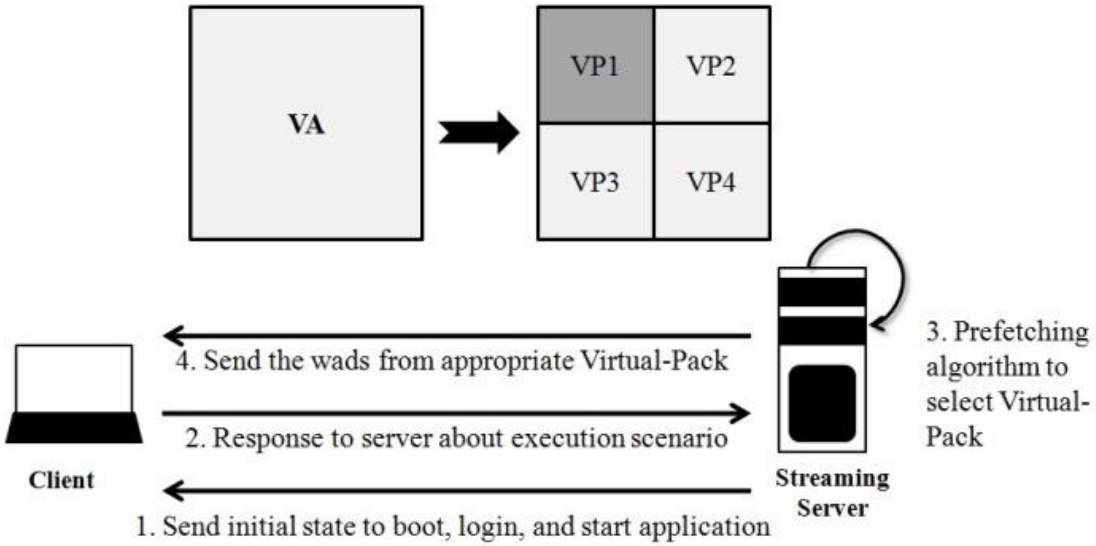

Fig. 3 Selection of virtual packs on basis of decision points

After the establishment of secure communication link between server and client using TCP/IP connection, the streaming server would forward the initial state towards the client end encapsulating the login and booting threads. On the successful completion of the booting stage, the client would send to the streaming server information related to its current state of action. On the basis of this information, the streaming server selects the appropriate VP. As shown in Fig. 3, a larger VP has been divided into four smaller VPs. Each VP is a module of specific functionality. Once that process is completed, all wads or chunks of that virtual pack are transmitted to the client side for execution. In this way user does not need to download larger VA and he can execute the smaller VPs as per requirement. As the VA is sent in the form of virtual packs containing an image file of virtual machine, there would be no compatibility issues arises on the user end. It substantially reduces the size of virtual appliance file because the streaming server transmits only the intended small sized VP to the user instead of larger VA file. After applying compression the size of file already containing only one phase of the game is further reduced. It needs lesser bandwidth to transmit smaller files. Smaller the size of packet, the lesser is the time it requires to arrive at the destination. If the data size is larger than the maximum transmission unit (MTU) of the particular network then the data

will be fragmented into smaller packets which consume time. In case of lost of larger amount of data there will be more to resend. The sending of small sized VP increased the process of VA streaming. The same is the case of video streaming where small amounts of frames are quickly required to start playing the video. The size of wads would be in proportion to the window size of the TCP to lower the overhead of acknowledgement.

4.4 Design Assumptions:

It is assumed that the developer or producer will encapsulate a task-specific and focused work flow in every virtual appliance [14]. But this assumption restrains the type and limit of interaction between the virtual appliance software and the user. Furthermore, many virtual appliances may use numerous processes for the implementation of a task, and a number of these processes possibly are multi-threaded. It increases the complexity in the prefetching of the required virtual appliance state.

The decision points would be interpreted correctly by the streaming server. Incorrect decision point interpretation would result into transfer of undesired VP to the user [15]. There should be some mechanism to transfer the desired portion of the software either automatically or through some interaction on part of the user [16]. In either case it may result in clogging of the network with junk material. The network bandwidth may be consumed for transferring undesired and unrequested data.

The streaming server could perfectly weigh against the current user actions with the VP that would be probably required in the future. Before user would finish the execution of current VP, the packaged file of next VP must be transmitted to client end for smooth user-system interaction. Otherwise there will be delay between the executions of adjoining VPs. It adversely affects the user-software interaction.

5 ARCHITECTURE

CVAS architecture has leveraged the advantages of client-server model as shown in Fig. 4. Both the client and server exploit the Linux-based QEMU-KVM (quick-emulator kernel-based virtual machine) virtualization module to stream VA. Virtualization is supported by QEMU when executing under Xen hypervisor or employing KVM module in Linux. QEMU is selected as it is open source and generic virtualizer and machine emulator. As an emulator, QEMU can execute operating systems on heterogeneous machines by employing dynamic translation. On the other hand, when utilized as virtualizer, it realizes near native performance by directly running the guest code on host machine. When employing KVM, the QEMU can virtualizes x86 and ESA/390 guests. The KVM offers complete virtualization solution for Linux operating system on x86 having virtualization extensions such as AMD-V or Intel VT. It is made up of loadable kernel module called kvm.ko which offers virtualization and processor-specific module, kvm-amd.ko or kvm-intel.ko. Through KVM we can operate multiple VM containing unmodified Windows or Linux images. The mainline QEMU encapsulates the userspace component of KVM to facilitate the hardware virtualization of numerous processors. Kernel component of KVM is embedded in mainline Linux to run VM. The user session can be established between the client and server through open source libvirt application programming interface (API). It is preferred because it supports multiple different hypervisors such as qemu-kvm, Xen, VirtualBox, VMware ESX/GSX, Microsoft Hyper-V, IBM PowerVM, Bhyve, and Virtuozzo hypervisors. Through libvirt we managed and control remote VM on streaming server using virsh tool. Another reason for using libvirt in CVAS project is that it can also be accessible from Microsoft Windows clients along with Linux OS. On the server side qemu-kvm run multiple VAs simultaneously as they are images of VMs. A database of VAs is maintained at server side from where the user can select the VA of his choice through libvirt on web. But user would get the VP instead of VA as all the VAs are fragmented into VPs. The VPs are transparent to user. For ease of management and make the system simple, we fragmented VAs into equal sized VPs. It

is easy to manage and transmit VPs of equal sizes as compared to variable sizes VPs. The variable-sized VPs may need memory management for queues and results in delays.

Therefore streaming server stored the equal-sized VPs as their construction and transmission is less complicated and unproblematic as compared to variable-sized VPs. The construction and management of variable-sized VPs are left for future research. Simultaneously the VPs are further fragmented into equal sized wads or packets to further reduce the size of file during transmission phase.

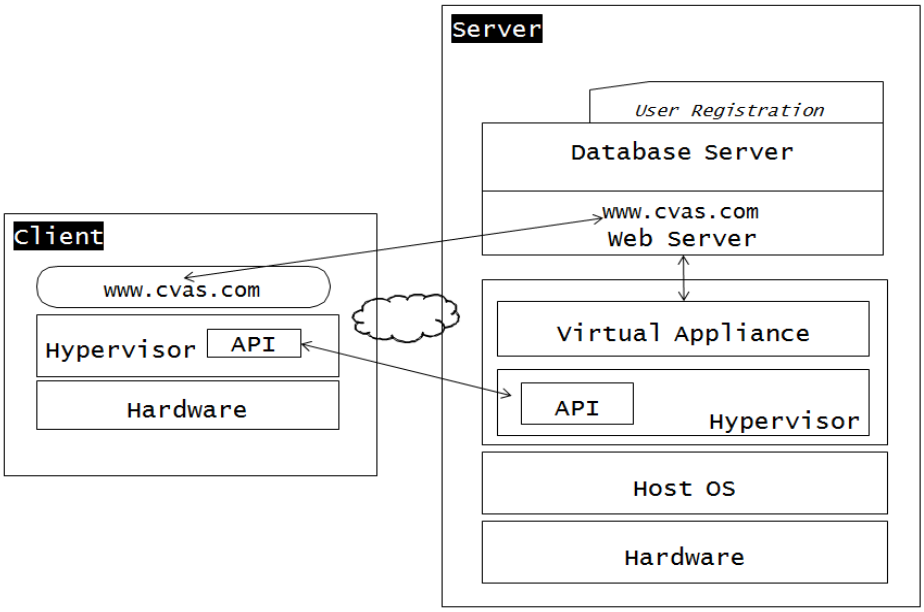

Fig. 4 CVAS Architecture

5.1 Client Architecture:

A small modification at qemu-kvm on client side enables us to access the virtual appliance from our cloud based streaming server. The client could demand the appropriate virtual packs instead of the complete virtual appliance file. The VP image

would be accessed through libvirt API and for this purpose the virsh command line and GUI virtual machine manager (VMM) utilities would be available at client side. When the client is paused due to any reason, the user would be notified through client GUI. During delays at buffering, user could utilizes his time in some other productive manner.

5.2 Server Architecture:

The process of selection, transmission, and control of appropriate VPs is embedded in our streaming server. The VA is fragmented into appropriate number of fixed-sized VPs through recoding and modular programming by the producer. The limitation is that non-destructive type VAs cannot be handle in the same manner. Therefore, in case of non-destructive VAs, the complete VA image file would be transmitted from server to client for its execution. Each VP could be accessed through its interface by exploiting open source libvirt API for the establishment of TCP session between client and intended VP. Opendedup data deduplication compression technique is applied on our Linux-based streaming server to avoid the storage of redundant and duplicate copies of data. On the selection of appropriate VP trough libvirt API, the server would spontaneously send the whole image file of the VP without any delay. The selection of the VP through DP relies upon client instead of server. It simplified the decision making process.

6 EXPERIMENTAL EVALUATION

To conduct the experimental evaluation we installed the prototype version of CVAS on Linux server and offer quick streaming service to the Linux hosts. Our primary assessment standard is the buffer time and the number of interruptions during the execution of software.

6.1 Experimental Setup:

The central idea behind the CVAS is to create a user friendly streaming system that would provide good performance on low bandwidth networks and general purpose computing devices like PC. Therefore the CVAS has been evaluated and tested over the 4-mbps shared bandwidth network. The Internet connection speed varies from country to country and lies between less than 1 mbps to over 16 mbps. The network speed of 4 mbps could represent the large user base especially of developing countries. We conducted our experiments on client side hardware system with minimal specifications.

The computing system used for the CVAS experiment is general purpose system is in vogue and possesses by general user instead of high profile users as illustrated in Fig. 5. Out of total 2 GB main memory available at client end, we set aside 1 GB as virtual memory to accommodate Linux operating system. We take the leverage of Linux-based qemu-kvm hypervisor for the execution of virtual packs over host machines. The remaining 1 GB of main memory has been utilized by host operating system. The host operating system may be Windows or Linux.

Description	*System Specifications*
Clock Speed	3.60 GHz
Cache	3 MB
Cores / Threads	2 / 4
TDP Power	54
Primary Memory	2 GB
Virtual Memory	1 GB
Secondary Storage	250 GB
Virtual Storage	60 GB
Physical Bus Extension	32 bits
Network Download Speed	4 mbps shared

Table 1: CVAS system and network specifications

The working of CVAS was tested on minimal 4-mbps shared bandwidth networks but with the increment in bandwidth capacity, the performance of the system increases. The CVAS is assumed to be used by a general purpose users, therefore the system specifications and network settings were laid down keeping that objective in view.

6.2 Experimental Virtual Appliance:

We have chosen the Microsoft Office suit to build an experimental virtual appliance. Over 1 billion people used Microsoft Office worldwide therefore it is a commonly run software. Its desktop components include Word, Excel, Access, PowerPoint, etc. The selection of Microsoft Office served the dual purpose. It is large-sized and non-compatible to the Linux operating system. The installation of Microsoft Office requires 3 GB of secondary memory along with 1 GB of primary memory on 32 bits system and 2 GB of primary memory on 64 bits system. In order to execute the Microsoft Office over Linux, we converted it into a virtual appliance. When converted into virtual appliance, it could be run over virtual machine taking the leverage of Linux based qemu-kvm hypervisor.

Instead of creating a large-sized virtual appliance which is consisted of complete set of Microsoft Office suit, we fragmented it into small sized virtual packs as shown in Fig. 6. For this purpose we made “Not Available” all the desktop applications of Microsoft Office saving the one which we need to create the virtual pack. The virtual packs thus created included Excel, Word, PowerPoint, and Access. Each of them is task oriented and offers explicit functionality which is not possibly provided by its sibling virtual pack. For instance the virtual pack consisting of Excel spread sheet facilitates the mathematical calculations, graphical tools, tables, and Visual Basics. On the other hand virtual pack consisted of Access application is utilized by users for the management of database system in order to store, update, retrieve, and manage data. For the installation and running of virtual packs on diverse architectures, the application software and customized

operating system were tied together to create a packaged image file of virtual machine. That image file was stored by the streaming server as a virtual pack. The virtual packs were interconnected through content coupling.

CVAS Experimental Virtual Appliance			
Description	*Type*	*Buffer Time Hr:Min:Sec*	*Interruptions Hr:Min:Sec*
Word	VP	00:01:49	00:00:00
PowerPoint	VP	00:01:53	00:00:00
Excel	VP	00:01:55	00:00:00
Access	VP	00:01:58	00:00:00
Office Suit	VA	00:03:25	00:00:00

Table 2 Buffer Time and Interruptions Time during execution of VA and VPs

Figure 7 shows the list of virtual appliance and virtual packs that we tested on the CVAS. We used GNS3 emulator on Linux system to simulate the network consisted of qemu-based VMs. Another advantage of employing GNS3 is that it offers large number of ready-made virtual appliances and allows creating new VAs according to the requirements. We then streamed the VA and VPs to the client end and recorded the data related to the buffer time and number of interruptions during the process. When we streamed the complete virtual appliance, there was buffering delay of 3 minutes and 25 seconds before the execution started. But once the virtual appliance was entirely streamed to the client end; there were zero interruptions during its executions. The reason was that the virtual machine's image file contained complete functionality along with the customized operating system that had been received at the client end. After the completion of task we streamed the four small-sized virtual packs one after the other. When we streamed the virtual packs instead of large-sized virtual appliance the buffer time reduced to between 1 minute and 49 second to 1 minute and 58 seconds. There were approximately 52 percent reduction in buffer time. The major advantage of CVAS system

is that once the execution has been started, there would be no interruptions as we faced during the execution of profiling and prefetching streaming approaches.

6.3 Comparison of the CVAS with the Profiling and the Prefetching of Access State Approaches

To facilitate the streaming of VAs over network, the VMTorrent put forward the idea of profile-based prefetching of the VM images from the cloud to the host machine. It advocated the creation of a packaged VM image and specific workflow profiles. To attain that objective they run workflow at least once on VM image from boot to shutdown to sort out those pieces which were appeared in at least one profile run and made a profile of ordered list of those pieces. They blacklisted those pieces which were never appeared in any run. The profile was taken as criteria to predict the way user will execute the VM image. But this approach has limitations. Important information would be lost during the profiling process. Users execute the VM image differently for each workflow. As the profiles are specific to workflow and VM image, these are not general profiles to be applied for all types of workflows.

vTube advocated the idea of prefetching VA state to the host machine that is predicted to be used in future course execution. Instead of historical traces of the VM execution in past, it relies on the current state of VM execution and the amount of available bandwidth. It is in contrast to the VMTorrent profiling approach where static profiles were build which rely on VM image and workflow combinations. But the vTube approach is not devoid of limitations. The inaccurate prefetching hints would result in system stalls and network clogging with undesired data chunks. It seriously affects the overall performance of the streaming process.

6.3.1 Lesser Buffer Time:

We created the big-sized virtual appliance equivalent to 3 GB. It was then modulated into small-sized and task-oriented virtual packs. These virtual packs were streamed to the client on-demand instead of the entire virtual appliance. It is observed that CVAS consumed less buffer time before the start of VA execution as compared to above-mentioned prior streaming approaches. As shown in Fig. 7, the CVAS has utilized 1 minute and 53 seconds before the execution of the virtual pack started. VMTorrent had consumed 4 minutes and 25 seconds to start the execution of virtual appliance, while vTube delayed for 16 minutes and 33 seconds for the same purpose.

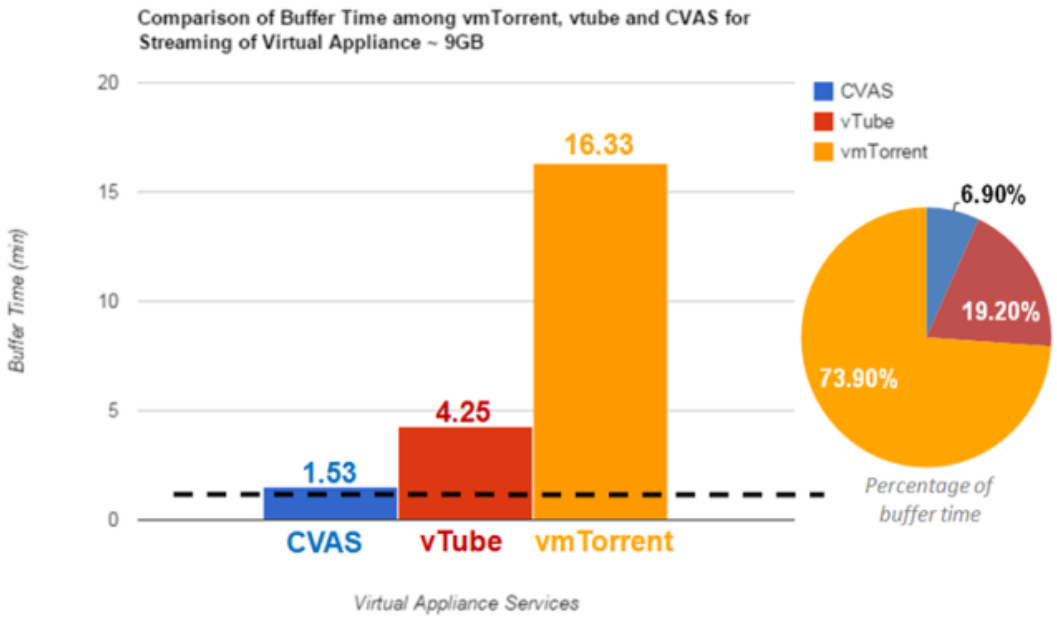

Fig. 5 Comparison CVAS, Profiling, and Pre-fetching streaming approaches

Less buffer time resulted in timeliness execution of VA over network. The CVAS consumed 6.9% of time when compared to 19.2% by VMTorrent and 73.9% by vTube. The applications having less buffer time are quick to start and considered as user friendly.

6.3.2 Zero Interruptions during Execution:

Once the execution of the virtual appliance stared at the client end, zero interruption was experienced from the boot-up to the shutdown phase. The zero interruption experience was the result of the streaming of virtual pack (i.e. packaged virtual machine image file) containing task oriented functionality and supported operating system. As all the necessary functionality is prefetched, the software function smoothly without hinderance.

6.4 Comparison of CVAS with Onboard Native Application:

It is noted that the CVAS system performs almost in equivalence to the native application installed on the host machine. The native application is more efficient because it has shorter span of delay time. But once the buffering phase has been completed the CVAS provides performance on the same level as native application as shown in Fig. 8.

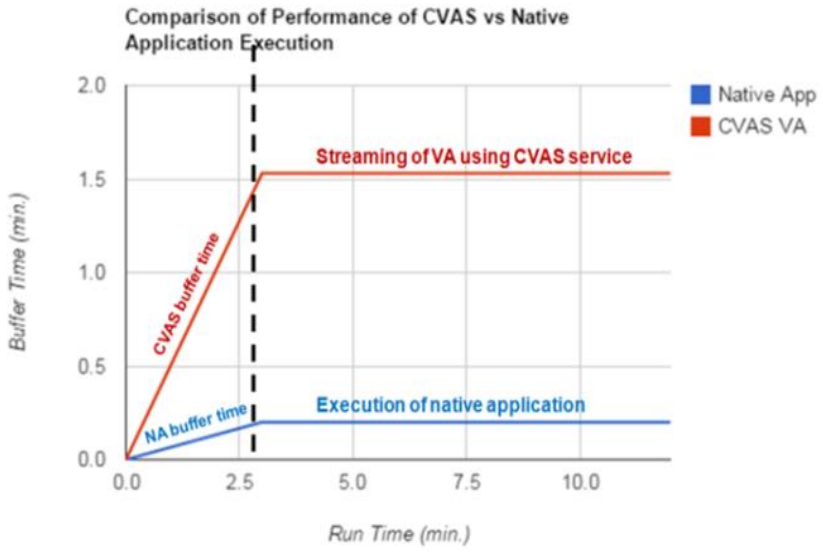

Fig. 6 Similarity between CVAS and native application execution.

The user has to wait for 0.20 minutes before the native application is started. Whereas in case of CVAS, the delay during buffer time is 1.5 minutes which is 1.30 minutes higher than native application. Though the CVAS system has taken more time to buffer than the native application but it is cost effective solution for proprietary software. To install and execute application on a native system, the user has to buy the license which bears cost. Sometimes software is required for a shorter span of time and to purchase the license for little period of time is not feasible solution. Whereas virtual appliance is available on a cloud computer owned by a third party. It is more appropriate to streaming that VA from the cloud server on the basis of pay as you go contract. The procurement, maintenance, and up-gradation of VA would be the responsibility of the cloud provider. It greatly reduces the cost of utilizing the software. VA stored on the cloud can be streamed to any host machine installed at remote location provided the availability of Internet connection. Whereas native machine would be executed on the host machine at which it is installed. Therefore CVAS offers flexibility in the execution of different types of VA over diverse hardware architectures.

7 CHALLENGES OF THE CVAS SYSTEM

Major challenge before CVAS system is the fragmentation or modulation of big-sized virtual appliance into small-sized task-specific virtual packs. It requires modification in the source code of the software application. For proprietary applications the source code is not publically available for modifications. Even in case of open source software, high technical expertise would be required to fragment the software code into well structured and functional modules or virtual packs. Moreover some software applications are indestructible and cannot be fragmented further. Linking of modules also requires special knowledge of programming.

8 THE FUTURE RESEARCH & DIRECTIONS

The CVAS system is applicable where virtual appliance can be readily disjointed into independent task-specific virtual packs. The linking and message passing between virtual packs is an important phenomenon which needs cautious implementation. But this is not true for every type of virtual appliance. For example virtual appliance does not work well if after broken down into task-specific virtual packs, there is no mechanism to get results from one virtual pack into sister virtual pack. To make it easy it is assumed there is automobile engineering virtual appliance having two interrelated components, i.e., engine design unit and body design unit. Both components are complementary to each other. The engine's design must match with the body design otherwise automobile design could not be completed. For example a 2000 cc engine design cannot be matched with the body of 800 cc automobile. According to engineering practices engine is designed first and then body is manufactured to accommodate the engine. The vehicle body's weight, its dimensions, and its seating capacity all depends upon the engine design. If two independent virtual packs are created, i.e., one for the engine design module and another for the body design module, then there must be some linkage mechanism to get information about the engine design from the engine virtual pack into the body virtual pack. The research to find out solution to the problem is open for future studies.

9 CONCLUSION

The significance of cloud based virtual appliance is growing with new innovation in virtualization technology. Streaming of large-sized virtual appliances would address the issue of proprietary rights of the software and cost and maintenance issues on client end. But it is not feasible to stream large sized virtual appliances in a single go. Even with

better network connections it would be cumbersome to download large virtual appliances and sometimes the downloading would not be successful at all. It would be frustrating if that happens time and again. The essence of the CVAS system is modulation of virtual appliance into task-oriented virtual packs. The on-demand prefetching of virtual packs greatly reduces the transmitting time, and bandwidth consumption of the network. The burden on the processing power of internal nodes also reduces with the reduction in size of data. The CVAS has taken into consideration of this principle of sparsity to devise a mechanism where user will get only those components of the software which is needed by him. To achieve this objective some modifications in the source code of the software are required for modularization. The user can begin the execution of the virtual appliance with the reception of initial virtual pack. The CVAS system thus works like the streaming of video where media player could start playing video when it gets frames equivalent to its playback window size. The users do not wait for lengthy download times. The CVAS system works without interruptions like the native host application once the initial buffer time elapsed.

Bibliography

[1] W.E. Dong, W. Nan, L.Xu, "QoS-Oriented Monitoring Model of Cloud Computing Resources Availability," Computational and Information Sciences (ICCIS), 2013 Fifth International Conference on, Pages: 1537 - 1540, DOI: 10.1109 / ICCIS.2013.404, 2013

[2] G. Kecskemeti; G. Terstyanszky; P. Kacsuk, "Virtual Appliance Size Optimization with Active Fault Injection ", IEEE Transactions on Parallel and Distributed Systems, Pages: 1983 - 1995, DOI: 10.1109/TPDS.2011.309, Year: 2012, Volume: 23, Issue: 10

[3] C. Y. Huang; C. S. Kuo; S. P. Luan, "Evaluation and Application of Bounded Generalized Pareto Analysis to Fault Distributions in Open Source Software", IEEE Transactions on Reliability, Year: 2014, Volume: 63, Issue: 1 Pages: 309 - 319, DOI: 10.1109 / TR.2013.2285056

[4] R. Nossenson; S. Polacheck, “On-Line Flows Classification of Video Streaming Applications”, Network Computing and Applications (NCA), 2015 IEEE 14th International Symposium on, Pages: 251 - 258, DOI: 10.1109/NCA.2015.51

[5] G. Kecskemeti; G. Terstyanszky; P. Kacsuk; Z. Nemeth, “Towards Efficient Virtual Appliance Delivery with Minimal ManageableVirtual Appliances”, IEEE Transactions on Services Computing, Year: 2014, Volume: 7, Issue: 2, Pages: 279 - 292, DOI: 10.1109/TSC.2013.12

[6] J. Reich, O. Laadan, E. Brosh, A. Sherman, V. Misra, J. Nieh, and D. Rubenstein. VMTorrent: Scalable P2P Virtual Machine Streaming. In Proceedings of CoNEXT12, Nice, France, December 2012.

[7] M. I. Islam; J. I. Khan, “Video Splicing Techniques for P2P Video Streaming”, Distributed Computing Systems Workshops (ICDCSW), 2015 IEEE 35th International Conference on, Year: 2015, Pages: 72 - 76, DOI: 10.1109 / ICDCSW.2015.23

[8] J. Wu; C. Yuen; N. M. Cheung; J. Chen; C. W. Chen, “Enabling Adaptive High-Frame-Rate Video Streaming in Mobile Cloud Gaming Applications”, IEEE Transactions on Circuits and Systems for Video Technology, Year: 2015, Volume: 25, Issue: 12, Pages: 1988 - 2001, DOI: 10.1109 / TCSVT .2015. 2441412

[9] P. Orosz; T. Skopkó; P. Varga, “Towards estimating video QoE based on frame loss statistics of the video streams”, Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on, Year: 2015, Pages: 1282 - 1285, DOI: 10.1109 / INM.2015.7140482

[10] M. Banerjee; S. R. Roy; C. Kumar, “Feature Oriented Programming: A step towards flexible composition of modular programming” Recent Advances in Information Technology (RAIT), 2012 1st International Conference on, Year: 2012, Pages: 369 - 373, DOI: 10.1109 / RAIT.2012.6194448

[11] Kaijun Fan; Bingyin Xu; Guofang Zhu; Jie Gao, “Fast peer-to-peer real-time data transmission for distributed control of distribution network”, Electricity Distribution (CICED), 2014 China International Conference on, Year: 2014, Pages: 1041 - 1045, DOI: 10.1109 / CICED.2014.6991864

[12] X. Fu; X. Li; Y. Zhu; L. Wang; R. S. M. Goh, "An intelligent analysis and prediction model for on-demand cloud computing systems", Neural Networks (IJCNN), 2014 International Joint Conference on, Year: 2014, Pages: 1036 - 1041, DOI: 10.1109 / IJCNN.2014.6889875

[13] N. P. Cardo, "Logjam: A scalable unified log file archiver", High Performance Computing, Networking, Storage and Analysis (SC), 2011 International Conference for, Year: 2011, Pages: 1 - 9, DOI: 10.1145 / 2063348.2063379

[14] M. Schots; C. Werner, "Using a Task-Oriented Framework to Characterize Visualization Approaches", Software Visualization (VISSOFT), 2014 Second IEEE Working Conference on, Year: 2014, Pages: 70 - 74, DOI: 10.1109/VISSOFT.2014.20

[15] Y. Abe, R. Geambasu, K. Joshi, H. A. Lagar-Cavilla, and M. Satyanarayanan, "vTube: efficient streaming of virtual appliances over last-mile networks," in Proc. of the 4th annual Symposium on Cloud Computing. ACM, 2013.

[16] S. K. Soni; R. Arora; A. S. Rodge, "Referer' based Predictive Caching and session prefetching for Browser", India Conference (INDICON), 2014 Annual IEEE, Year: 2014, Pages: 1 - 6, DOI: 10.1109/INDICON.2014.7030529

[17] N. Sabahat; A. A. Malik; F. Azam, "Size estimation of open source board-based software games", 2015 International Conference on Open Source Systems & Technologies (ICOSST), Year: 2015, Pages: 126 - 131, DOI: 10.1109 / ICOSST. 2015. 7396414

[18] Zhaojuan Yue; Xiaodan Zhang; Yongmao Ren; Jun Li; Qianli Zhong, "The performance evaluation and comparison of TCP-based high-speed transport protocols", Software Engineering and Service Science (ICSESS), 2012 IEEE 3rd International Conference on, Year: 2012, Pages: 509 - 512, DOI: 10.1109 / ICSESS.2012.6269516

[19] Jen-Ho Yang; Ya-Fen Chang; Chih-Cheng Huang, "A user authentication scheme on multi-server environments for cloud computing", Information, Communications and Signal Processing (ICICS) 2013 9th International Conference on, Year: 2013, Pages: 1 - 4, DOI: 10.1109/ICICS.2013.6782791

[20] Anish Babu S.; Hareesh M. J.; John Paul Martin; Sijo Cherian; Yedhu Sastri. "System Performance Evaluation of Para Virtualization, ContainerVirtualization, and Full Virtualization Using Xen, OpenVZ, and XenServer". Advances in Computing and Communications (ICACC), 2014 Fourth International Conference on.

[21] Gabor Terstyanszky; Gabor Kecskemeti; Peter Kacsuk; Zsolt Nemeth. "Towards Efficient Virtual Appliance Delivery with Minimal Manageable Virtual Appliances". IEEE Transactions on Services Computing. Year: 2014, Volume: 7, Issue: 2. Pages: 279 - 292, DOI: 10.1109/TSC.2013.12

[22] Changhua Sun; Le He; Qingbo Wang; Ruth Willenborg. "Simplifying Service Deployment with Virtual Appliances". Services Computing, 2008. SCC '08. IEEE International Conference on. Year: 2008, Volume: 2, Pages: 265 - 272, DOI: 10.1109/SCC.2008.53