Literature Survey and Review: Virtualization Technologies

Himalayan University
Itanagar, Arunachal Pradesh, INDIA
Dr.Syed Umar
Parmanand Prabhat.

Abstract - The implementation and use of virtualization offers big advantages, when used in an existing IT infrastructure. For example better utilization of server resources, reduction of power consumption and easy management of virtual infrastructures. The virtualization technology has been proven by the use of actual products. Due to this technology and the security features, the design and the implementation of a sensitive part of the IT infrastructure, a DMZ, is displayed. This thesis focuses on the security aspects of virtualization and its use for the virtualization of a DMZ. The current situation is that the DMZ always runs within a physical environment. With the installation of a virtual DMZ it is possible to use all the advantages of a virtual infrastructure. The aim of this thesis is to develop a system to implement a DMZ within a virtual infrastructure, which is at least as secure as a physical one. In order to understand the virtualization technology the theoretical basics are described. Their history and types of virtualization are evoked. Then comes the description of the system architecture of server virtualization and the available virtualization products on the market. Based on the features and functions of the virtualization products the VMWare Virtual Infrastructure 3 was chosen to implement the virtual DMZ. All the features and advantages of this product fully comply with the needs of a modern datacenter. The security measurements, and the comparison of the virtualized environment and a physical environment are discussed.

Keywords--- Aggregation, Hardware Independence, Isolation, Encapsulation, Performance Evaluation.

I.INTRODUCTION

Virtualization is a framework that aggregates or partitions the physical computing resources to offer dissimilar operating environments by various methodologies and techniques. Virtualization hides the complexity and heterogeneity of the core hardware. On the top of core hardware, many operating systems were running all at once by this technology. The concept of virtualization that divides a physical machine into one or more isolated machines is called as a virtual machine [1] (VM). At the time of VM creation, the attributes of VM like hardware independence, isolation, encapsulation, compatibility, control, equivalence and performance are set automatically.

The isolation of VM can be made by the virtualizable processor, which allows any instruction altering or examining machine state to be trapped when executed in any, but the highest privileged mode. Normally, a machine operates in two modes: user and supervisor. In supervisor mode, the privileged/non-privileged instructions can be executed, where in the operating system (OS) runs where as only non-privileged instructions can be executed in user mode. Before handing over CPU control to the user, the OS sets the mode bit to user. In user mode, if privileged instruction is executed in OS then interrupt is invoked and it is to be serviced by interrupt service routine (ISR).

The duplication process of the physical machine by using a software, mapping is the process of trapping software routines and passing instructions to the physical machine is called emulation. The imitation process is to accept the pre-defined inputs and providing the pre-defined responses is called simulation. An emulated hardware, VM, managed by the software is called a virtual machine monitor (VMM) or hypervisor.

In this paper, the virtualization technologies are classified in different abstraction levels. Using these technologies and techniques, and how they are used for implementing the researches, is surveyed and summarized based on the classifications and levels discussed by the research papers. The creation of abstraction by the virtualization software using virtualization layer at the different levels of a computer is discussed in the next section. The Virtualization layer consists of instruction set architecture (ISA) level, hardware abstraction layer (HAL) level, operating system level, library support level, and application/programming language level. An overview of the classifications and products of the hypervisor is discussed in the later section.

II.ABSTRACTION LEVELS OF VIRTUALIZATION

2.1. **Instruction Set Architecture (ISA) Level**:

Is an abstract model of a computer. It is also referred to as **architecture** or **computer architecture**. A realization of an ISA is called an *implementation*. An ISA permits multiple implementations that may vary in performance, physical size, and monetary cost (among other things); because the ISA serves as the interface between software and hardware. Software that has been written for an ISA can run on different implementations of the same ISA. This has enabled binary compatibility between different generations of computers to be easily achieved, and the development of computer families. Both of these developments have helped to lower the cost of computers and to increase their applicability. For these reasons, the ISA is one of the most important abstractions in computing today.

2.2 Hardware Abstraction Layer (HAL) Level:

Virtualization at this level exploits the resemblance in guest architectures and host platforms to cut down the interpretation latency. Virtualization technique helps map the virtual resources to physical resources and use the native hardware for VM computations. When an emulated machine communicates the physical resources, the simulator takes over and multiplexes appropriately. It is performed on top of the bare hardware. This approach generates a VM in virtual hardware environment. The computer resources are virtualized, such as its processors, memory, and I/O devices. The objective of this virtualization is, to increase the hardware utilization by multiple users concurrently. Examples: Xen, VMware, VirtualBox, VirtualPC, Denali, User-Mode- Linux (UML), and Plex86.

2.3 Operating System Level:

Operating-system-level virtualization, also known as containerization, refers to an operating system feature in which the kernel allows the existence of multiple isolated user-space instances. Such instances, called containers,[1] partitions, virtualization engines (VEs) or jails (FreeBSD jail or chroot jail), may look like real computers from the point of view of programs running in them. A computer program running on an ordinary operating system can see all resources (connected

devices, files and folders, network shares, CPU power, quantifiable hardware capabilities) of that computer. However, programs running inside a container can only see the container's contents and devices assigned to the container.

2.4 Library Support Level:

The application binary interface (ABI) gives a program access to the hardware resources and services available in a system through the user ISA and the system call interface. The ABI does not include system instructions; rather, the hardware resources communicated indirectly by the application programs by calling the operating system's services via the system call interface. System calls provide to perform operations on behalf of a user program after validating their authenticity and safety. The application programming interface (API) gives a program that accesses the system services and resources via the user ISA supplemented with high-level language (HLL) library calls. Any system calls are performed through libraries. Virtualization techniques are used to implement a different ABI and/or a different API using the underlying system. Such techniques are done by the ABI/API emulation. Virtualization with library interfaces is controlling the communication link between the applications and the rest of a system through API hooks. Using this approach, WINE software tool supports Windows applications on top of UNIX hosts and vCUDA software tool allows to execute applications within VMs to leverage GPU hardware acceleration [2]. Other examples: WABI, LxRun, OpenGL, Mocking and Visual MainWin.

2.5 User-Application Level:

Virtualization at this level virtualizes an application as a VM, which executes as a process in traditional OS. It is also called as process-level virtualization and deploys in high level language (HLL) VMs. The process involves packaging of isolated application VM from the host OS and other applications. At this level, the created VM behaves like a machine that supports a set of applications and a new self-defined set of instructions. Application VM resides on top of the OS and deploys self-contained executable files as application software in an isolated
environment exclusive of installation, system modifications, or elevated security privileges. Examples: Microsoft .NET CLR, Java Virtual Machine (JVM) and Parrot.

III. VIRTUALIZATION IMPLEMENTATION TECHNIQUES

3.1 Guest Operating System Virtualization

A guest OS is the software installed on either a virtual machine (VM) or partitioned disk that describes an operating system that is different than the host operating system. Virtualization technology allows a computer to run more than a single OS at the same time. A guest OS on a virtual machine can be different from the host OS while a guest OS on a partitioned disk must be the same be the same as the host OS. For example, if the host OS is running Windows, then any guest OSes on a portioned disk must also run windows. In a virtualized environment, the guest OS can be different than the host OS. A guest OS is required before a virtual machine can be deployed.

3.2 Shared Kernel Virtualization :

Shared kernel virtualization, also called operating system virtualization or system level virtualization, takes advantage of the unique ability of UNIX and Linux to share their kernels with other processes on the system. This shared kernel virtualization is achieved by using a feature called change root (chroot). The chroot feature changes the root file system of a process to isolate it in such a way as to provide some security. It (chroot) is often called a chroot jail or container-based virtualization. A chrooted program, set of programs, or entire system in the case of shared kernel virtualization is protected by setting up the chrooted system to *believe* that it is a standalone machine with its own root file system.

3.3 Kernel Level Virtualization :
Kernel-level virtualization is kind of an oddball in the virtualization world in that each VM uses its own unique kernel to boot the guest VM (called a root file system) regardless of the host's running kernel.

Linux KVM (Kernel Virtual Machine) is a modified QEMU, but unlike QEMU, KVM uses virtualization processor extensions (Intel-VT and AMD-V). KVM supports a large number of x86 and x86_64 architecture guest operating systems, including Windows, Linux, and FreeBSD. It uses the Linux kernel as a hypervisor and runs as a kernel loadable module.

User-mode Linux (UML) uses an executable kernel and a root file system to create a VM. To create a VM, you need a user-space executable kernel (guest kernel) and a UML-created root file system. These two components together make up a UML VM. The command-line terminal session you use to connect to the remote host system becomes your VM console. UML is included with all 2.6.x kernels.

3.4 Hypervisor Virtualization :
The x86 CPU families offer various protection levels, like system space, kernel mode or supervisor mode called as rings in which the code can execute. Under hypervisor virtualization, hypervisor or type 1 VMM is a software program that runs directly on the host machine in ring 0, i.e., highest level privilege. The task of this hypervisor is to handle resource allocation and memory allocation for the VM in addition, to providing interfaces for higher level administration and monitoring tools. Any guest OS kernels running on the system must run in ring less
privilege.

A hypervisor can be a firmware, hardware and piece of software that provides an impression to the guest machines, as if they were operating on a real or physical hardware. It allows many operating systems to share a single host and its hardware. It controls demands by virtual machines to access to the hardware resources, like CPU, RAM, NIC, etc, acting as an independent machine. It is also known as Virtual Machine Monitor (VMM). The hypervisor can be categorized into two types, such as Type I Hypervisor and Type II Hypervisor

Type I Hypervisor: It is also known as native or embedded or bare metal, as it directly runs on top of the underlying hardware. It does not have an OS running below it. In this case, VMM is a small code which is responsible for scheduling and allocating of the system's resources between virtual machines. It is more secure than Type II Hypervisor. Example: VMware ESX and Xen.

Type II Hypervisor: The host OS does not have any knowledge about Type II VMM; it treats it as any other process. The OS run inside of the Type II VMM is referred to as the Guest OS. It typically performs I/O on behalf of guest OS. The guest OS issues the I/O request that is trapped by host OS that in turn send to device driver that perform I/O. The completed I/O request is again routed back to the guest OS via host OS. It is
less secure than Type I, because any security Vulnerabilities that lead to the compromise of the host OS will also give full control of the guest OS. The Host OS are heavyweight than Type I. Example: VMware GSX (Workstation) and UML (User-Mode Linux)

3.4.1. Full Virtualization or Native Virtualization
The unmodified OS preserves guest OS kernel and consequently executes instructions in privileged level as ring . The full virtualization offers support for unmodified guest OS. The hypervisor provides CPU emulation to handle and modified privileged and protected CPU operations made by unmodified guest OS kernels.

Unfortunately, this emulation process needs mutually computing time and system resources to operate resulting in lower performance levels when compared to paravirtualization.

3.4.2. Para Virtualization :
The guest OS kernel is modified exclusively to run on the hypervisor. The hypervisor performs the task for the guest kernel. This limits support to open source Linux OS, which may be freely modified and proprietary operating systems, where the owners have decided to build the essential code modifications to target a specific hypervisor. In spite of these issues, the ability of the guest kernel to directly communicate with the hypervisor results in greater performance levels compared to other virtualization.

3.4.3 Hardware Assisted Virtualization:

Hardware-assisted virtualization first appeared on the IBM System/370 in 1972, for use with VM/370, the first virtual machine operating system. With the increasing demand for high-definition computer graphics (e.g. CAD), virtualization of mainframes lost some attention in the late 1970s, when the upcoming minicomputers fostered resource allocation through distributed computing, encompassing the commoditization of microcomputers.

IBM offers hardware virtualization for its POWER CPUs under AIX (e.g. System p) and for its IBM-Mainframes System z. IBM refers to its specific form of hardware virtualization as "logical partition", or more commonly as LPAR.

The increase in compute capacity per x86 server (and in particular the substantial increase in modern networks' bandwidths) rekindled interest in data-center based computing which is based on virtualization techniques. The primary driver was the potential for server consolidation: virtualization allowed a single server to cost-efficiently consolidate compute power on multiple underutilized dedicated servers. The most visible hallmark of a return to the roots of computing is cloud computing, which is a synonym for data center based computing (or mainframe-like computing) through high bandwidth networks. It is closely connected to virtualization.

IV. OTHER TYPES OF VIRTUALIZATION

4.1. Application Virtualization
VMware ThinApp is an application virtualization and portable application creator suite by VMware that can package conventional applications so that they become portable applications. According to VMware, the product has a success rate of about 90–95 % in packaging.
this virtualization, a server application is accessed and executed by the user locally using the local resources without this application required on user machine. Applications are virtualized and designed to run in a small virtual environment containing the only resources required for the application to execute. Accordingly, each user has an isolated application environment virtually in this virtualization. This small isolated virtual environment acts as a layer between the application and the host operating system.

4.2.Network Virtualization :

In computing, network virtualization or network virtualisation is the process of combining hardware and software network resources and network functionality into a single, software-based administrative entity, a virtual network. Network virtualization involves platform virtualization, often combined with resource virtualization.

Network virtualization is categorized as either external virtualization, combining many networks or parts of networks into a virtual unit, or internal virtualization, providing network-like functionality to software containers on a single network server.

In software testing, software developers use network virtualization to test software which are under development in a simulation of the network environments in which the software is intended to operate. As a component of application performance engineering, network virtualization enables developers to emulate connections between applications, services, dependencies, and end users in a test environment without having to physically test the software on all possible hardware or system software. The validity of the test depends on the accuracy of the network virtualization in emulating real hardware and operating systems.

4.3.Storage Virtualization

Storage virtualization is the pooling of physical storage from multiple storage devices into what appears to be a single storage device -- or pool of available storage capacity -- that is managed from a central console. The technology relies on software to identify available storage capacity from physical devices and to then aggregate that capacity as a pool of storage that can be used in a virtual environment by virtual machines (VMs).

The virtual storage software intercepts I/O requests from physical or virtual machines and sends those requests to the appropriate physical location of the storage devices that are part of the overall pool of storage in the virtualized environment. To the user, virtual storage appears like a standard read or write to a physical drive.

Even a RAID array can sometimes be considered a type of storage virtualization. Multiple physical disks in the array are presented to the user as a single storage device that, in the background, replicates data to multiple disks in case of a single disk failure.

4.4. Automotive Virtualization

It is a well-accepted fact that in the future of the automotive industry — autonomous driving, connectivity, and mobility — the role of software is going to increase exponentially. We are also witnessing an increase in the software complexity and the number of components in automotive.

At present, the approach of vehicle system designing is towards modularized vehicle control system where the software subsystems are distributed across multiple control units but it is widely believed that the future of the automotive industry will be more towards centralized vehicle architecture.

The centralized architecture has advantages like it can save development cost as well as enable faster communication and computation power but it requires integration of hardware and software components of different automotive safety integrity levels.

There are standards like AUTOSAR available that provide flexibility in integrating different software components but they also have disadvantages as they compromise on isolation from software interference.

In this line, Virtualization has emerged as a technique which can handle the upcoming innovations, increasing complexity and the requirement of computation power in automotive.

Virtualization is not a new concept and it is being used for several decades in server domains, ensuring integration of heterogeneous subsystems without interference. With the increased usage of faster multicore chips in the automotive industry, the scope of implementation of virtualization has increased.

Virtualization is achieved through an introduction of a software layer termed as Hypervisor or Virtual Machine Monitor into the system that creates and runs virtual machines that can host operating systems like Linux and Android, and also real-time operating systems such as AUTOSAR etc. It makes it possible to run functions with different requirements regarding real-time behavior and functional safety simultaneously on a single Soc.

4.5. Desktop Virtualization

Desktop virtualization is software technology that separates the desktop environment and associated application software from the physical client device that is used to access it. Desktop virtualization can be used in conjunction with application virtualization and user profile management systems, now termed "user virtualization", to provide a comprehensive desktop environment management system. In this mode, all the components of the desktop are virtualized, which allows for a highly flexible and much more secure desktop delivery model. In addition, this approach supports a more complete desktop disaster recovery strategy as all components are essentially saved in the data center and backed up through traditional redundant maintenance systems. If a user's device or hardware is lost, the restore is straightforward and simple, because the components will be present at login from another device. In addition, because no data are saved to the user's device, if that device is lost, there is much less chance that any critical data can be retrieved and compromised.

4.6 Parallel Desktop Virtualization

Parallels Desktop for Mac is a hardware emulation virtualization software, using hypervisor technology that works by mapping the host computer's hardware resources directly to the virtual machine's resources. Each virtual machine thus operates identically to a standalone computer, with virtually all the resources of a physical computer. Because all guest virtual machines use the same hardware drivers irrespective of the actual hardware on the host computer, virtual machine instances are highly portable between computers. For example, a running virtual machine can be stopped, copied to another physical computer, and restarted. Parallels Desktop for Mac is able to virtualized a full set of standard PC hardware, including·

V. LITERATURE SURVEY DESIGN

This literature survey was motivated by the emerging technology of virtualization to diverse in techniques and its applications whichresulted in numerous research publications in the recent years. The next section shows the survey finding. In total, 186 research papers published in scholarly journals in the last two decades are investigated and grouped under the abstraction levels as per the methodology used, architectures, complexity, security, performance and applications. The papers under these groupings are reviewed from Section.

5.1. Survey Findings

Once you have finished conducting a survey, all that is left to do is write the survey report. A survey report describes a survey, its results, and any patterns or trends found in the survey. Most survey reports follow a standard organization, broken up under certain headings. Each section has a specific purpose. Fill out each section correctly and proofread the paper to create a polished and professional report.

5.2.Survey Findings in Instruction Set Architecture (ISA) Level

Instruction Set Architecture (ISA) separates the hardware and software components of a computer system. An ISA is considered similar to human language, it is the language used by processors. An ISA lists all the instructions user can give. Registers, addressing modes, data-types, memory management, devices and exception handling, power management, and multi-threading support serve as components of ISA. Implementation of an ISA comes under micro-architecture. Multiple implementations can be made for a single ISA, and hence it can have various micro-architectures. ISAs are classified into RISC (Reduced Instruction Set Computer) and CISC (Complex Instruction Set Computer) [1]. A basic set of operations is specified by the instruction set. Any instruction from the instruction set commands an operation that paves way for altering the machine state, data, or performing an input-output operation. Arithmetic and logical operations are performed by the instructions from the first category. Operations valid can be string, logical, arithmetic, or floating-point. Appropriate functional units perform the operations of the particular implementation of the architecture. Branches, calls and returns, and loop control instructions affect the flow of the program and machine state. Instructions performing data movement across different functional units of the machine belong to the third category. Examples are LOAD instruction that helps to load the content of a memory location to a register in particular or STORE instruction that functions exactly the opposite. Then there's an instruction that moves a block of data from a memory location to another, it is called.

Reduced Instruction Set Computer (RISC) - Nowadays, modern CPUs are mostly of the GPR (General Purpose Register) type. For example IBM 360, DEC VAX, Intel 80x86 and Motorola 68xxx. But while these CPUs were clearly better than previous stack and accumulator-based CPUs they were still lacking in several areas, one out of many examples is: Instruction length varies from 1 byte to 6-8 bytes. This caused problems with the management of instructions such as pipelining and pre-fetching of instructions. Thus, the idea of RISC was introduced [4]. A reduced instruction set computer architecture (RISC) termed as MIPS is considered as the original RISC Instruction Set Architecture.

Characteristics of RISC - (a) RISC processor per instruction's (CPI) of a single sequence is clocked which is available for optimization of each instruction on the microprocessor. (b) To avoid dealings with memory, a large number of registers usually join. (c) Instructions which change the movement of controls such as branch's instruction which is executed well since they compress about 20 to 30% of distinctive programs. (d) infrequently used instructions are not attempted to implement by RISC.

INSTRUCTION EXECUTION - Every computer performs a very basic function i.e., the execution of a program, where every executable program is basically a vast collection of instructions which are present in the memory. In order to complete a specific task, the instructions of the program are executed by the CPU (Central Processing Unit). The CPU has the prime responsibility of executing every instruction and every execution takes place in the CPU registers [11]. It executes the programs that are hoarded in the main memory by fetching their instructions, thereby investigating them, and executing them in a sequential manner.

ADDRESSING MODES - Addressing modes can be seen as the way in which in an instruction operand are quantified. The location of the operand is an important parameter on which the way of formation of the operand address depends along with the choices present in the instruction architecture. In the case of accumulator architecture or stack, there is no need of citing the operand address as it is already entailed. If it belongs to the General-Purpose Register (GPR) then the address of that particular register is already intact in the operand field in the instruction. So, this mode is known as register direct addressing and is among the easiest ways of citing the location of the operand. When the operand is being comprised within the instruction then the addressing of the operand becomes even simpler and is known as immediate addressing mode. The pointed location by the addressed formed comprises in the operand field of the instruction that is capable of containing the address of the operand or the operand itself.

5.3.Survey Findings in Hardware Abstraction Layer (HAL) Level

This Whitepaper describes the concept of Hardware Adaptation Layer (HAL) for applying the OpenFlow protocol to the non-OpenFlow hardware. In this document, first, the motivation of designing the HAL is explained as well as the high level goals and supported network platforms. Then, the logical architecture of HAL is presented including the architectural and functional requirements. As next chapter the network hardware integration models are provided and the supported hardware platforms are classified. Finally, the proposed implementation of HAL is presented including the software interfaces for platform integrators such as the Abstract Forwarding API and the Pipeline Interface.

Logical architecture of HAL - Following the SDN concept and according to OpenFlow architecture, the data path or forwarding engine inside of an OpenFlow device must be controlled and managed by a controller who resides outside of the device and communicates to the device via a secure channel. The data path is represented to the controller in an abstracted fashion as a table or tables with flows holding packets information and actions associated to them which could be manipulated (programmed) by the controller software.

Logical architecture of HAL - Following the SDN concept and according to OpenFlow architecture, the data path or forwarding engine inside of an OpenFlow device must be controlled and managed by a controller who resides outside of the device and communicates to the device via a secure channel. The data path is represented to the controller in an abstracted fashion as a table or tables with flows holding packets information and actions associated to them which could be manipulated (programmed) by the controller software.

Requirements for HAL - Optimum HAL for network devices to support OpenFlow protocol needs to have certain features so that it can support the current and future architecture of networking devices.

Network hardware integration models - Various types of network hardware give different possibilities for HAL integration with devices. ALIEN HAL supports three main hardware integration models for devices with different requirements, constraints and hardware capabilities.

Proposed HAL architecture - Following the functional and architectural requirements for desired HAL a unified architecture is proposed. Hardware platform categorization helps to build organized and modular components which then yields to create a flexible and extensible HAL. In the following, HAL components and their functionalities are described.
HAL implementation - The modular architecture of xDPD [XDP] that could use ROFL [ROF] pipeline library for its datapath implementation and freedom that ROFL library provides for implementation OpenFlow protocol on various platform regardless of their underlying hardware and software, leads to conclusion that the combination of xDPD and ROFL could build ALIENs HAL architecture. In the following, it will be explained that how the building blocks of xDPD and ROFL could be used in the ALIEN project.

Software interfaces available for platform integrators - Considering the architecture of ROFL , since the library is platform agnostic, it could be used on any platform for different purposes. In a basic conceptual view, a developer could implement an OpenFlow forwarding module by calling ROFL library. Also, ROFL provides an Abstracted Forwarding API (AFA) for controlling and management of the logical datapath. Part of AFA covers OpenFlow API but it also supports management tasks API. The library provides.

Pipeline interface - As it was mentioned FM employs ROFL pipeline library for implementing the OpenFlow pipeline. The ROFL pipeline library supports logical switch concept. It also supports for multi version of OpenFlow which is convenient for developers to implement different version of OpenFlow protocol based on their device capabilities. It is worth to mention that ROFL pipeline library is not a fully ready to use OpenFlow data path with support of platforms subsystems such as I/O. Although the subsystems are part of FM, they are platform specific components which are different on each platform. The I/O subsystems could be seen as device or interface drivers which have different architecture and API on every platform and systems.

5.4. Survey Findings in OS-Level

Improve mobile experience of users, recent mobile devices have adopted powerful processing units (CPUs and GPUs). Unfortunately, the processing units often consume a considerable amount of energy, which in turn shortens battery life of mobile devices. For energy reduction of the processing units, mobile devices adopt energy management techniques based on software, especially OS (Operating Systems), as well as hardware. In this survey paper, we summarize recent OS-level energy management techniques for mobile processing units. We categorize the energy management techniques into three parts, according to main operations of the summarized techniques: 1) techniques adjusting power states of processing units, 2) techniques exploiting other computing resources, and 3) techniques considering interactions between displays and processing units. We believe this comprehensive survey paper will be a useful guideline for understanding recent OS-level energy management techniques and developing more advanced OS-level techniques for energy-efficient mobile processing units.

5.5.Survey Findings in Application Level

In the context of computer networking, an application-level gateway[1] (also known as ALG, application layer gateway, application gateway, application proxy, or application-level proxy[2]) consists of a security component that augments a firewall or NAT employed in a computer network. It allows customized NAT traversal filters to be plugged into the gateway to support address and port translation for certain application layer "control/data" protocols such as FTP, BitTorrent, SIP, RTSP, file transfer in IM applications, etc. In order for these protocols to work through NAT or a firewall, either the application has to know about an address/port number combination that allows incoming packets, or the NAT has to monitor the control traffic and open up port mappings (*firewall pinhole*) dynamically as required. Legitimate application data can thus be passed through the security checks of the firewall or NAT that would have otherwise restricted the traffic for not meeting its limited filter criteria.

5.6. Survey Findings in Kernel-based Virtual Machine

Kernel-based Virtual Machine (**KVM**) is a virtualization module in the Linux kernel that allows the kernel to function as a hypervisor. It was merged into the Linux kernel mainline in kernel version 2.6.20, which was released on February 5, 2007.[1] KVM requires a processor with hardware virtualization extensions, such as Intel VT or AMD-V.[2] KVM has also been ported to other operating systems such as FreeBSD[3] and illumos[4] in the form of loadable kernel modules.

KVM was originally designed for x86 processors and has since been ported to S/390,[5] PowerPC,[6] IA-64, and ARM.[7]

KVM provides hardware-assisted virtualization for a wide variety of guest operating systems including Linux, BSD, Solaris, Windows, Haiku, ReactOS, Plan 9, AROS Research Operating System[8] and macOS.[9] In addition, Android 2.2, GNU/Hurd[10] (Debian K16), Minix 3.1.2a, Solaris 10 U3 and Darwin 8.0.1, together with other operating systems and some newer versions of these listed, are known to work with certain limitations.[11]

Additionally, KVM provides paravirtualization support for Linux, OpenBSD,[12] FreeBSD,[13] NetBSD,[14] Plan 9[15] and Windows guests using the VirtIO[16] API. This includes a paravirtual Ethernet card, disk I/O controller,[17] balloon device, and a VGA graphics interface using SPICE or VMware drivers.

5.7. Survey Findings in Data Center Network Virtualization

Data centers have recently received significant attention as a cost-effective infrastructure for storing large volumes of data and hosting large-scale service applications. Today, large companies like Amazon, Google, Face book, and Yahoo! routinely use data centers for storage, Web search, and largescale computations [1]–[3]. With the rise of cloud computing, service hosting in data centers has become a multi-billion dollar business that plays a crucial role in the future Information Technology (IT) industry. Despite their importance, the architectures of todays data centers are still far from being ideal. Traditionally, data centers use dedicated servers to run applications, resulting in poor server utilization and high operational cost. The situation improved with the emergence of server virtualization technologies (e.g., VMware [4], Xen [5]), which allow multiple virtual machines (VMs) to be co-located on a single physical machine. These technologies can provide performance isolation between collocated VMs to improve application performance and prevent interference attacks. However, server virtualization alone is

insufficient to address all limitations of todays data center architectures. In particular, data center networks are still largely relying on traditional TCP/IP protocol stack, resulting in a number of limitations.

The virtualization of data center networks is still in its infancy, and recent research has mainly focused on how to provide basic functionalities and features including the partitioning of data center network resources, packet forwarding schemes and network performance isolation. Accordingly, we focus our attention in this survey on: • Packet forwarding schemes which specify the rules used to forward packets between virtual nodes. • Bandwidth guarantees and relative bandwidth sharing mechanisms that provide network performance isolation and more efficient network resource sharing, respectively. • Multipathing techniques used to spread the traffic among different paths in order to improve load-balancing and fault-tolerance. Nevertheless, there are other features worth considering when virtualizing data centers such as security, programmability, manageability, energy conservation, and fault-tolerance. So far, however, little attention has been paid to these features in the context of data center virtualization. We provide more details about the challenges related to these features in the future research directions section (Section V). In the following, we briefly describe the features we focus on in the paper, and then survey the proposals. A forwarding scheme specifies rules for sending packets by switching elements from an incoming port to an outgoing port. A FIB allows mapping MAC address to a switch port when making a decision about packet forwarding. To support relative bandwidth sharing, congestion controlled tunnels [6] may be used, typically implemented within a shim layer that intercepts all packets entering and leaving the server. Each tunnel is associated with an allowed sending rate on that tunnel implemented as a rate-limiter. The other alternatives [29] are group allocation for handling TCP traffic, rate throttling for controlling UDP traffic, and centralized allocation for supporting more general policies, e.g., handling specific flows. The first alternative uses fair queueing; the second one relies on a shim layer below UDP. One of the techniques to achieve bandwidth guarantee is the use of rate-limiters [7], [15], [16], [28]. In particular, a rate-limiter module is incorporated into a hypervisor of each.

Net Lord - To maximize revenue, providers of Infrastructure-as-aService (IaaS) are interested in a full utilization of their resources. One of the most effective ways to achieve that is by maximizing the number of tenants using the shared.

VICTOR - Cloud tenants have a need to migrate services across data centers, to balance load within and across data centers, or to optimize performance of their services. On the other hand, cloud users want to have fast and efficient delivery of services and data. One approach that allows to achieve the above objectives of tenants and users is migration of VMs. To avoid service interruption, a VM should keep the same IP address during migration. Although that is not a challenge for migration within the same IP network, providing migration over different networks is not straightforward. VICTOR (Virtually Clustered Open Router) [25] is a network architecture for supporting migration of VMs across multiple networks that enables migrating VMs to keep their original IP addresses. The main idea of VICTOR shown in Figure 6 is to create a cluster of Forwarding Elements (FE) (L3 devices) that serve as virtual line cards with multiple virtual ports of a single virtualized router. Thus, the aggregation of FEs performs data forwarding for traffic in a network. FEs are distributed over several networks, which helps to support migration of VMs across multiple networks. The control plane is supported by one or several Centralized Controllers (CC). A VM is deployed on a server connected to only one edge FE. A CC maintains a topology table that specifies the connectivity between FEs, and an address table that determines the connectivity.

PortLand - VM population scalability, efficient VM migration, and easy management are important characteristics of current and nextgeneration data centers. PortLand addresses all these issues for a multi-rooted fat-tree topology (see Figure 3). In particular, the architecture proposes an L2 routing mechanism employing the properties of that topology. It supports plugand-pay functionality for L2, which significantly simplifies administration of a data center network.

SPAIN - The current spanning tree protocol (STP) used for large Ethernet LANs is inefficient for supporting modern data center networks, since it does not exploit path diversity offered by data center networks, resulting in limited bi-section bandwidth and poor reliability [37]. Smart Path Assignment In Networks (SPAIN) [22] uses the VLAN support in existing commodity Ethernet switches to provide multipathing over arbitrary topologies. SPAIN computes disjoint paths between pairs of edge switches, and pre-configures VLANs to identify these paths. An end-host agent installed on every host spreads flows across different paths/VLANs. To improve load balancing and avoid failures, the agent can change paths for some flows. For instance, if the traffic is not evenly spread across the paths, the agent can change the VLANs used by some flows. The agent also detects failed paths and re-routes packets around the failures by using a different path. Whereas SPAIN provides multipathing, and improves load balancing and fault-tolerance, the proposal has some scalability issues. In particular, although path computation algorithm proposed by SPAIN is executed only when the network topology is designed or significantly changed, the scheme is computationally expensive for complicated topologies. In addition, SPAIN requires that switches store multiple entries for every destination and VLAN; it creates more pressure on switch forwarding tables than the standard Ethernet does. Furthermore, the number of paths is limited to the number of VLANs allowed by the 802.1q standard (4096) [21]. Finally, maintaining a mapping table between flows and VLANs leads to an additional overhead in each end-host.

Oktopus - Although infrastructure providers offer to tenants ondemand computing resources through allocating VMs in data centers, they do not support performance guarantees on network resources to tenants. The mismatch between the desired and achieved performance by tenants leads to the following problems. First, variability of network performance induces unpredictable application performance in data centers making application performance management a challenge. Second, unpredictable network performance can decrease application productivity and customer satisfaction, leading to revenue losses. Oktopus is the implementation of two virtual network abstractions (virtual cluster and virtual oversubscribed cluster) for controlling the tradeoff between the performance guarantees offered to tenants, their costs, and the provider revenue. Oktopus not only increases application performance, but offers better flexibility to infrastructure providers, and allows tenants to find a balance between higher application performance and lower cost.

Gatekeeper - Gatekeeper focuses on providing guaranteed bandwidth among VMs in a multiple-tenant data center, and achieving a high bandwidth utilization. In general, achieving a strict bandwidth guarantee often implies non effective utilization of a link bandwidth when free capacity becomes available. Gatekeeper addresses this issue by defining both the minimum guaranteed rate and maximum allowed rate for each VM pair. These parameters can be configured to achieve minimum bandwidth guarantee, while ensuring that link capacities are effectively utilized by tenants. Gatekeeper creates one or more logical switches that interconnect VMs belonging to the same tenant. The virtual NIC (vNIC) of each receiving VM monitors the incoming traffic rate using a set of counters, and reports.

Scalability - Achieving high scalability in virtualized data centers requires address spaces that support large number of tenants and their VMs. Furthermore, since todays commodity switches often have limited memory size, it is necessary to keep the number of forwarding states in each switch at minimum for achieving high scalability. Table VII shows the maximum number of tenants, VMs per tenant, and the size of the forwarding table. The maximum numbers of tenants and VMs depend mainly on the number of bits used to identify tenants and VMs. The number of VMs per tenant depends on the address space supported by IPv4, which can be extended when using IPv6. Depending on the forwarding scheme, the size of the forwarding table depends on the number of VMs, physical machines, switches or pods. In practise, the number of VMs is higher than the number of physical machines, which is in turn higher than the number of switches. We also notice that VICTOR and Portland do not support multi-tenancy.

VI. CONCLUSION :-

This paper offers a discussion about the virtualization technologies and its implementation techniques in various applications. Also, this paper discusses the levels of abstraction and their software products, to understand how virtualization is attained in various abstraction levels. This paper enables researchers to wisely prefer proper hypervisor based on the abstraction levels. How to improve the performance, entire hardware resource utilization, scalability and flexibility using concurrent virtualization techniques on desktop are revealed in the survey. Finally, this paper evinces the fact that the virtual machine monitor techniques are useful to support developing moreprotective hypervisor.

REFERENCES

[1]Yaozu Dong, et al, "HYVI: A HYbrid VIrtualization Solution Balancing Performance and Manageability", IEEE Transactions on Parallel and Distributed Systems, Vol. 25, Pp. 2322-2341, 2014.
[2]Durelli, "A Systematic Mapping Study on High-level Language Virtual Machines", VMIL'10, ACM, Reno, USA, 2010.
[3]Yaohui Hu, et al, "An Application-Level Approach for Privacy-preserving Virtual Machine Check pointing", ACM, Chicago, Illinois, USA , 2012.
[4]Tudor-Ioan Salomie, et al , "Application Level Ballooning for Efficient Server Consolidation",Eurosys'13, ACM, Prague, Czech Republic, 2013.

[5]Carlos Antunesa and Ricardo Vardascab, "Performance of Jails versus Virtualization for Cloud Computing Solutions", Conference on ENTERprise Information Systems-International Conference on Project MANagement / HCIST 2014 - International Conference on Health and Social Care Information Systems and Technologies, Procedia Technology, Elsevier, Pp. 649-658, 2014.
[6]Shu-Sheng Guo, et al, "A New ETL Approach Based on Data Virtualization", Journal Of Computer Science And Technology, Springer Science+Business Media, LLC & Science Press, China, pp. 311-323, 2015.
[7]Francois Armand and Michel Gien, "A Practical Look at Micro-Kernels and Virtual Machine Monitors", IEEE Computer Society, 2008.
[8]Tzi-cker Chiueh et al, "Stealthy Deployment and Execution of In-Guest Kernel Agents", Technical Report, Symantec Research Labs, 2009.
[9] R. Goldberg, "Survey of Virtual Machine Research", IEEE Computer Society, Pp. 34-45, 1974.
[10]L. Shi, H. Chen and J. Sun, "vCUDA: GPU accelerated high performance computing in virtual machines", Proceedings of the IEEE International Symposium on Parallel and Distributed Processing, 2009.
[11]Milind Ramesh Kolte, "Survey Paper On Different Virtualization TechnologyWith Its Merits And Demerits On Organization", International Journal of Engineering Research and Applications, Vol. 2, No. 5, Pp. 764-770, 2012.
[12]B. Quetier, V. Neri and F. Cappello, "Scalability Comparison of Four Host Virtualization Tools" Springer Science + Business Media B.V., J Grid Computing, Vol. 5, Pp. 83-98, 2006.
[13]John R. Lange, "Automatic Dynamic Run-time Optical Network Reservations", Department of Computer Science, Northwestern University, 2005.
[14]Ashish Gupta and Peter A.Dinda, "Inferring the Topology and Traffic Load of Parallel Programs Running In a Virtual Machine Environment", Department of Computer Science, NorthWestern University, 2005.
[15]Ashish Gupta, "Measuring and Understanding User ComfortWith Resource Borrowing", Department of Computer Science, Northwestern University, 2005.
[16]Fabio Marchio, Boris Vittorelli and Roberto Colombo, "Automotive Electronics: Application & Technology Megatrends", IEEE Computer Society, 2014.
[17]Bin Lin, et al, "User-driven Scheduling Of Interactive Virtual Machines", Department of Computer Science, Northwestern University, 2005.
[18]R.L. Bocchino Jr and V.S. Adve, "Vector LLVA: a virtual vector instruction set for media processing", Proceedings of the 2nd international conference on Virtual execution environments, Pp. 46-56, 2006.
[19]Stephen Wright, "Using EventB to Create a Virtual Machine Instruction Set Architecture", Department of Computer Science, University of Bristol, UK, 2009.
[20]P. Martinez-Julia, A.F. Skarmeta and A. Galis, "Towards a secure network virtualization architecture for The future internet", The Future Internet Assembly, Springer Berlin Heidelberg, Pp. 141-152, 2013.
[21]Bryan D. Payne, et al, "A Layered Approach to Simplified Access Control in Virtualized Systems," ACM, Chicago, Illinois, USA , 2007.
[22]Geoffroy Vallee, et al, "System-Level Virtualization for High Performance Computing", Technical Report, Laboratory Directed Research and Development Program of Oak Ridge National Laboratory (ORNL), USA, 2008.
[23]Monirul Sharif, et al, "Secure In-VM Monitoring Using Hardware Virtualization", CCS'09, ACM, Chicago, Illinois, USA, 2009.
[24]Edison Tsui Ka Wing and Zakaria Lamliki, "Survey on Virtual Machine", International Federation for Information Processing, 2009.
[25]Muhammad Wannous and Hiroshi Nakano, "NVLab, a Networking Virtual Web-Based Laboratory that Implements Virtualization and Virtual Network Computing Technologies", IEEE Transactions on Learning Technologies, Vol. 3, 2010.
[26]Peter A. Dinda, et al, "The User In Experimental Computer Systems Research", ACM, San Diego, CA, 2007.
[27]Chang-Hao Tsai, et al, "Virtualization-Based Techniques for Enabling Multi-tenant Management Tools", International Federation for Information Processing, Pp. 171-182, 2007.
[28]N.M. Mosharaf Kabir Chowdhury and Raouf Boutaba, "A Survey of Network Virtualization", Technical Report, David R. Cheriton School of Computer Science, University of Waterloo, 2008.
[29]Yunfa Li, et al, "A Survey of Virtual Machine System: Current Technology and Future Trends", Third International Symposium on Electronic Commerce and Security, IEEE Computer Society, 2010.
[30]Rabi Prasad Padhy, et al, "Virtualization Techniques & Technologies: State-Of-The-Art", Journal of Global Research in Computer Science, Vol. 2, Pp. 30-43, 2011.
[31]Shivani Sud, et al, "Dynamic Migration of Computation Through Virtualization of the Mobile Platform", Mobile Netw Appl, Springer Science+Business Media, Pp. 206–215, 2012.
[32]Pontus Skoldstrom, et al, "Network Virtualization and Resource Allocation in OpenFlow-based Wide Area Networks", IEEE Computer Society, Pp. 6622-6626, 2012.
[33]Md. Motaharul Islam, et al, "An Efficient Model for Smart Home by the Virtualization of Wireless Sensor Network", International Journal of Distributed Sensor Networks, 2013.
[34]Weizhe Zhang, et al, "Multiple Virtual Machines Resource Scheduling for Cloud Computing", An International Journal, Applied Mathematics & Information Sciences, Vol. 7, Pp. 2089-2096, 2013.
[35]Lin Wang, et al, "Joint Virtual Machine Assignment and Traffic Engineering for Green Data Center Networks", Greenmetrics '13, ACM, Pittsburgh, Pennsylvania USA, 2013.
[36]En-Hao Chang, et al, "Virtualization Technology for TCP/IP Offload Engine", IEEE Transaction on Cloud Computing, Vol. 2, Pp. 117-129, 2014.