Effective Usage of Resources and Optimizing the Power Consumptions using Virtualization

Mr. Kumaraswamy K S
R. V. College of Engineerin
kumaraswamy.ks@rvce.edu.

Mrs. Amudha T
R. V. College of Engineering
amudha2004@gmail.com

***Abstract*—Virtualization is ability to reducing power consumptions and hardware costs by server consolidation. Virtualization can optimize resource sharing among applications hosted in different virtual machines to meet their resource needs. To safely transition an application running natively on real hardware in a virtualized environment, IT organizations are being challenged to improve resource utilization, reduce power consumption, security, improve reliability, and become more responsive to rapid changes. The virtualization deployed in conjunction with consolidation can help to meet all the objectives, and additional benefits of effective usage of resources and optimizing the power consumptions. Virtualization is the creation of a virtual version of something, such as a hardware, software, operating system, a storage device, desktop or network resources.The usual goal of virtualization is to centralize administrative tasks while improving efficiency, scalability and overall effective resource utilization.**

***Keywords*—Virtualization,Virtmanager,KVM(kernel virtualization module), libvirtd, QEMU, QEUM-KVM, brigdeutlis**

I. INTRODUCTION

Currently, virtualization is ability to virtualize servers including more efficient processing and resource utilization, reduced power/cooling costs, and improved management capabilities. Virtualization optimizes the use of existing resources; simplifies infrastructure and software administration, maintenance, and deployment; and reduces hardware needs, resulting in less power consumption, less space required, and lower cooling costs.

In the current IT (Information Technology) infrastructure operates in a radically different way than it was done few years ago. Although the data-center may have a similar look in some ways, still servers and storage are major part of data-center, beneath the layers, the way that the datacenter is operated has evolved in a significant way. As for energy conservation, earlier the amount of energy wasted is a function of the number of physical servers which is clearly much lower in a virtualized environment. The advantage here is the reduced cost of maintenance and reduced energy wastage. As you have fewer physical servers, you need only maintain them and therefore maintenance becomes much easier and cheaper.

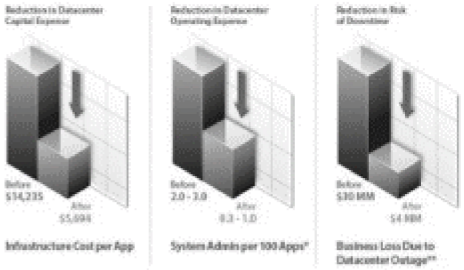


Fig-1: Energy Consumption and Optimizing resources

Before the concept of virtualization was extended to the X86 server markets, when organizations needed a new service, the deployment of that service started with the purchase, installation and configuration of expensive pieces of hardware. In those days, individual servers were sized to accommodate peak loads. After all, you didn't want the end user or customer experience to suffer because a server, for example, had too little RAM or too few disks. However, although servers were sized for peak loads, typical/average utilization for most resource components – processors, RAM and storage – felt far short of the maximum provided by the hardware. In other words, organizations sized each individual workload at peak and did so across all services. Reduce IT capital costs (CapEx) and operating expenses (OpEx) by as much as 60% while simultaneously lowering power, cooling and real estate spend. Virtualization shows how to cut energy consumption by up to 80% and optimize organization's financial resources as shown in Fig-1.

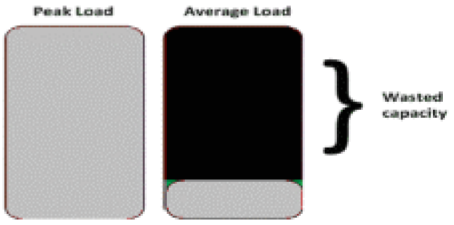


Fig 2: Each server's resource usage follows a similar pattern

At the same time, the technology landscape was booming and new services being brought on at a fast and furious pace. In order to accommodate new workloads, individual servers were purchased for each of those workloads in order to avoid the potential for conflict between software and resources. Thus, the phrase "server sprawl" was born [1].

As things added up, each of these servers carries a price tag, which, at the time, was not insignificant. On top of that, as each server was added to the data-center, the organization had to assume new power and cooling costs. Further, given the pace of growth, data-center space was a rare commodity. Racks were being added at a faster pace while companies struggled to keep up with demand.The companies were deploying servers that would not run at peak capacity and with every new service, adding new costs in two way: First, each hardware device carried with it ongoing capital costs due to the need to ultimately replace that device; Second, the company's ongoing operating budget had to be adjusted for account for new power and cooling costs.

KVM (for Kernel-based Virtual Machine) is a full virtualization solution for Linux on x86 hardware containing virtualization extensions (Intel VT or AMD-V)[2]. It consists of a loadable kernel module (kvm.ko) that provides the core virtualization infrastructure and a processor specific module, kvm-intel.ko or kvm-amd.ko. KVM also requires a modified QEMU although work is underway to get the required changes upstream. Using KVM, one can run multiple virtual machines running unmodified OS images. Each virtual machine has private virtualized hardware: a network card, disk, graphics adapter, etc. The kernel component of KVM is included in mainline Linux, as of 2.6.20. KVM is open source software.

The scalability of x86 platforms has been restricted by inherent kernel memory limitations; thus, organizations have typically been unable to fully utilize the computing power delivered by higher end servers. As a result, consolidation in the x86 environment has been impractical for most scenarios. Test results indicate that a virtualized x86 platform may now be able to support as many or more users compared to an x64 platform. These results demonstrate that virtualization may be the key to unleashing the power of today's high Performance servers in an x86 environment.The project was started in mid-2006, and has been part of the Linux kernel since the 2.6.20 release in February of 2007. kvm is a full virtualization system for x86-based Linux hosts, allowing users to run isolated x86 guest operating systems in virtual machines.

II. TYPES OF VIRTUALIZATION

Although these are different kinds of virtualization, these virtualization types are generally included in x86 server virtualization plans [1].

A. Operating System Virtualization:

Operating system virtualization is the use of software to allow a piece of hardware to run multiple operating system images at the same time. The technology got its start on mainframes decades ago, allowing administrators to avoid wasting expensive processing power.

B. Network Virtualization

It is a method of combining the available resources in a network by splitting up the available bandwidth into channels, each of which is independent from the others, and each of which can be assigned (or reassigned) to a particular server or device in real time. The idea is that virtualization disguises the true complexity of the network by separating it into manageable parts; much like your partitioned hard drive makes it easier to manage your files.

VLANs have been around for a long time. A VLAN is a group of systems that communicate in the same broadcast domain, regardless of the physical location of each node. By creating and configuring VLANs on physical networking hardware, a network administrator can place two hosts – one in New York City and one in Shanghai – on what appears to these hosts to be the same physical network. The hosts will communicate with one another under this scenario. This abstraction had made it easy for companies to move away from simply using physical connections to define networks and be able to create less expensive networks that are flexible and meet ongoing business needs.

C. Storage Virtualization

It is the pooling of physical storage from multiple network storage devices into what appears to be a single storage device that is managed from a central console. Storage virtualization is commonly used in storage area networks (SANs).

D. Server Virtualization

It is the masking of server resources (including the number and identity of individual physical servers, processors, and operating systems) from server users. The intention is to spare the user from having to understand and manage complicated details of server resources while increasing resource sharing and utilization and maintaining the capacity to expand later.

It[4] allows a physical server to be partitioned to run multiple secure virtual servers. This creates an opportunity to consolidate physical servers, thus helping to reduce hardware acquisition and management costs by eliminating "infrastructure sprawl" at the server level. If the resource requirements of one of the server-based products running in a virtual server grows, because of increased usage for example, moving that virtual server to a different physical server with more available resources is as simple as copying a file. This ease of replication of virtual servers also means that it's simple to maintain snapshots of virtual servers as file back-ups and quickly restore complex systems to operation in the event of physical server hardware failures. This increased resilience results in more efficient use of existing physical server resources,

lower operating costs, reduced power consumption and greater overall reliability.

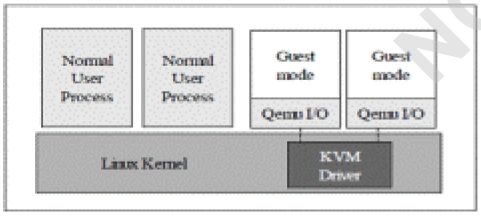
E. *Application Virtualization:* Virtualization is all about abstraction. When it comes to application virtualization, traditional applications are wrapped up inside a container that allows the application to believe that it is running on an original supported platform. The application believes that it has access to the resources that it needs to operate. Although virtualization applications are not really "installed" in the traditional sense, they are still executed on systems as if they were.

F. *Desktop Virtualization:* Virtualizes the traditional desktop and moves the execution of that client workload to the data center. Those workloads are then accessed via a number of different methods, such as thin clients or other means.

III. METHODOLOGY

A normal Linux process has two modes of execution: kernel and user. Kvm adds a third mode: guest mode (which has its own kernel and user modes, but these do not interest the hypervisor at all).The division of execution process among the different modes is:

A. Guest mode: Execute non-I/O guest code
B. Kernel mode: Switch into guest mode, and handle any exits from guest mode due to I/O or special instructions.
C. User mode: Perform I/O on behalf of the guest.
By integrating into the kernel, the kvm 'hypervisor' automatically tracks the latest hardware and scalability features without additional effort [5].



Fi 3: Diagramatic representation of kvm/qemu based virtualization solution on GNU/Linux Operation System.

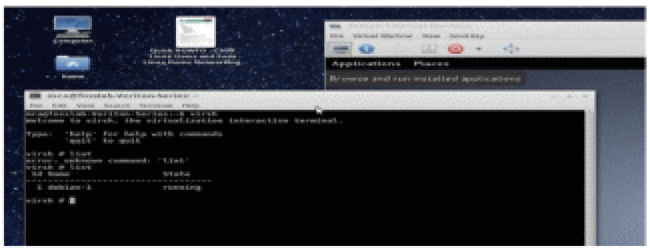


Fig 4: Administration of virtual machine guest OS through virsh tool Provided with libvirtb in package

First there is a need for a processor that supports virtualization. That can be checked via /proc/cpuinfo and if vmx or smx in the cpu flags field is present, then that system supports KVM. It supports full virtualization type of hardware virtualization

A typical KVM installation consists of a device driver for managing the virtualization hardware, this driver exposes its capabilities via a character device /dev/kvm[3]. A user-space component for emulating PC hardware; currently, this is handled in the user space and is a lightly modified QEMU process.

QEMU is a generic and open source machine emulator and virtualizer. The I/O model is directly derived from QEMU's, with support for copy-on-write disk images and other QEMU features. QEMU is made of several subsystems:[6] CPU emulator (currently x861, PowerPC, ARM and Sparc). Emulated devices (e.g. VGA display, 16450 serial port, PS/2 mouse and keyboard, IDE hard disk, NE2000 network card). Generic devices (e.g. block devices, character devices, network devices) used to connect the emulated devices to the corresponding host devices. Machine descriptions (e.g. PC, PowerMac, Sun4m) instantiating the emulated devices. Debugger and User interface.

The libvirt project develops such a virtualization abstraction layer, which is able to manage a set of virtual machines across different hypervisors. The goals of libvirt are to provide a library that offers all necessary operations for hypervisor management without implementing functionalities, which are tailored to specific virtualization solutions and which might not be of general interest [7].

IV. OVERVIEW OF TYPES OF HARDWARE VIRTUALIZATION

Hardware virtualization or platform virtualization refers to the creation of a virtual machine that acts like a real computer with an operating system. Software executed on these virtual machines is separated from the underlying hardware resources [3].

Different types of hardware virtualization include:

1. ***Full virtualization*:** Almost complete simulation of the actual hardware to allow software, which typically consists of a guest operating system, to run unmodified
2. ***Partial virtualization*:** Some but not all of them target environment is simulated. Some guest programs, therefore, may need modifications to run in this virtual environment.
3. ***Para virtualization***: A hardware environment is not simulated; however, the guest programs are executed in their own isolated domains, as if they are running on a separate system.

Hardware-assisted virtualization is a way of improving the efficiency of hardware virtualization. It involves

employing specially-designed CPUs and hardware components that help improve the performance of a guest environment.

Hardware virtualization is not the same as hardware emulation. In hardware emulation, a piece of hardware imitates another, while in hardware virtualization, a "hypervisor"(software) imitates a particular piece of computer hardware or the whole computer altogether. Furthermore, a hypervisor is not the same as an emulator; both are computer programs that imitate hardware, but their domain of use in language differs.

Binary translation: Instead of emulating the processor, the virtual machine runs directly on the CPU. To trap and manage privileged instructions, the hypervisor scans the virtual machine memory and intercepts privileged calls before they are executed and then dynamically rewrites the code in memory. The operating system kernel is unaware of the change and operates normally. This combination of trap-and-execute and binary translation allows any x86 operating system to run on the hypervisor [8].

V. CONCLUSION

By applying virtualization on server the resources were shared equally between each virtual machine image. The KVM model has an approach to virtualization that is fully aligned with Linux architecture; it integrates the hypervisor capabilities into a host Linux kernel as a loadable module which has simplified management and improved performance in virtualized server, and effective usage of resources and optimizing the power consumptions, and also resources like storage, network access and power were centralized for better management. Efficiency was achieved in the process of access to the resources on virtualized server by client over the network.

REFERENCES

[1] 1 .General Virtualization Articles by Scott.D. Lowe, article published on Sep 08, 2011.
[2] 2 Kernel Based Virtual Machine: http://www.linux-kvm.org/page/Main_Page
[3] Virtualization with KVM jounal published by Irfan Habib on Feb 01, 2008
[4] Virtualization in Education, IBM Global EducationWhite Paper published on Oct 2007.
[5] KVM: Kernel-based Virtualization Driver White Paper, 2006 Qumranet Inc.
[6] QEMU, a Fast and Portable Dynamic Translator, by Fabrice Bellard, FREENIX Track: 2005 USENIX Annual Technical Conference.
[7] Non-intrusive Virtualization Management using libvirtMatthias Bolte, Michael Sievers, Georg Birkenheuer, Oliver Niehörster and André Brinkmann Paderborn Center for Parallel Computing PC2, University of Paderborn, ürstenallee 11, 33102 Paderborn, Germany
[8] Harmonizing the Twin Trends of Open Source and Virtualization: How Kernel Based Virtual Machine (KVM) Drives Enterprise Business Value Sponsored by IBM Srini Chari, Ph.D., MBA, November, 2010