Server Virtualization and Virtual Machine Operating Systems

By

Karl Ray

Colorado Springs, Colorado

Overview

Virtual machine (VM) technology, software applications that simulate multiple instances of the underlying hardware, has been around since the 1960s. Originally, VMs were used for testing operating system upgrades for mainframe computers. As data centers transitioned from mostly using mainframes to mostly using large "farms" of commodity-priced servers, interest in VM waned. Recently, however, interest in VM technology has resurged. Underutilization of servers, the need to dynamically reallocate server resources for load balancing and availability, and the need to host legacy applications on new servers have created a compelling business case for server virtualization. This paper discusses the history of, technology, business case for, and a practical experiment with virtualization technology.

Server Virtualization and Virtual Machine Operating Systems

This paper discusses the historical, technical, business, and practical aspects of server virtualization. “Virtualization” is sharing the physical resources of a host computer amongst guest clients in such a way that each client “thinks” that it is operating natively on the host hardware. Virtualization evolved from a need to test fixes and new versions of mainframe operating systems. Despite beginning as a mainframe technology, there are now compelling business reasons for deploying virtualization technology in data centers using the Intel 32-bit and 64-bit architectures. The history, technology, benefits, and a practical experiment are further described in the remainder of this paper.

History

IBM began experimenting with “Virtual Machine” operating systems in the early 1960s. These early experimental versions are detailed in Creasy’s 1981 paper. IBM had a compelling business interest in creating a successful virtual machine operating system. During the period 1960-1990, IBM’s 360 and 370 architectures dominated the computing market. These were high-end machines typically costing hundreds of thousands to millions of dollars. Dedicating one of these machines to an operating system maintainer for testing was extremely costly. Virtualization of the machine allowed maintainers to simultaneously test different operating system images, saving IBM millions of dollars. The evolving virtual machine operating system became very robust and capable. Seeing potential market demand, IBM commercialized their internal product as the VM/370 operating system in 1981.

During the 1990s, minicomputers running proprietary operating systems or variants of UNIX began to undercut IBM’s 370 architecture in the lower end of the data center market. This trend toward commoditization of data center servers accelerated in the late 1990s and the 2000s

as low cost servers based on the Intel 32-bit and 64-bit architectures evolved. With commodity priced servers available, providing dedicated systems to developers of operating systems became feasible. Interest in virtual machine operating systems declined. However, another market trend drove a reemergence of interest in VM. This was the trend towards large, collocated, "server farms". Operators of these large "server farms" needed a technology that would allow them to dynamically move operating services from one machine to another and would also allow dynamically increasing the resources allocated to different services to achieve load balancing. VM reemerged to fill this need. Rosenblum's 2004 paper follows the decline of VM use as commodity-priced servers emerged in the market and then the reemergence of VM as the business case for server virtualization again became compelling.

The Business Case for Server Virtualization

Server virtualization can allow significant cost savings. Although server costs have dropped significantly between the 1980s and present, the cost of managing large server farms is significant. Many of these servers are added to support a single application and are lightly utilized. Consolidating these lightly used servers into virtual machines can significantly reduce management costs. According to Intel:

> Business growth invariably demands IT infrastructure growth. Servers are often added to support new applications, which in turn can lead to many under-utilized servers, higher network management costs, and decreased agility and reliability.
>
> Virtualization reduces server proliferation, simplifies server management, and significantly improves server utilization, network agility, and network reliability. It does this by consolidating multiple applications onto fewer enterprise-level servers.
>
> Hundreds of servers can be reduced to tens of servers with consolidation and virtualization. Server utilizations of 10% or less can be increased by as much as 60% or more. IT infrastructure agility, reliability, and efficiency are improved.

Another challenge arising from server proliferation is the need to support legacy applications on legacy servers. A large data center may have dozens of uniquely configured servers running legacy applications. Migration of these applications to newer hardware can be problematic. Virtualization in many cases provides an easier upgrade path for these applications. VMWare, in fact, provides a simulated IBM 370 environment running as a virtual machine as an option for rehosting a legacy IBM 370 application. According to Rosenblum:

> Hardware-level virtual machines encapsulate all software that runs on the hardware, thus giving the VMM the unique ability to manage the hardware resources, as well as manipulate and control the entire software stack. The monitor allows the software running in the virtual machine to be effectively decoupled from the hardware. This decoupling provides for many of the unique features of hardware-level virtual machines.
>
> Because all of the virtual machine software is encapsulated, the monitor can transparently manage the software and hardware in the virtual machine. The monitor can use this capability to run multiple virtual machines simultaneously on the same physical machines—or migrate running virtual machines between different hardware platforms. The monitor effectively controls all of the hardware. It also maps it to whatever virtual machines need the resource.
>
> The virtualization layer can also smooth out minor differences between hardware platforms to allow the same virtual machine to run on them. The VMM provides a conversion layer than can map the virtual devices of a virtual machine onto different physical devices.
>
> Finally, the encapsulation of all the software in the virtual machine allows the monitor to manage the entire software stack. The virtual machine abstraction can be used to provision software on a machine, as well as manage and load-balance it. The monitor can save or checkpoint the execution state of a virtual machine and restore it some other time. This capability allows it to effectively undo the execution of a virtual machine.

In summary, virtual servers provide an abstracted hardware interface for running virtualized clients. Since the interface is abstracted, these clients can be dynamically moved between servers. This gives the data center manager the ability to consolidate servers, to balance loads, and to maintain service when hardware failures occur, and to host legacy applications on

newer servers. The total advantages can far outweigh the small performance overhead incurred by server virtualization.

How Server Virtualization Works

There are different kinds of virtualization:

- Hardware Virtualization
  - Emulation or full system simulation: the virtual machine simulates the complete hardware, allowing an unmodified OS (perhaps even for a completely different CPU) to be run
  - Paravirtualization: the virtual machine does not simulate the hardware but instead offers a special API that requires OS modifications (this is the technique used by Xen)
  - Native Virtualization: the virtual machine simulates enough hardware to allow an unmodified OS to be run in isolation, but the guest OS must be designed for the same type of CPU. "Native virtualization" is also sometimes used to designate that hardware assistance through Virtualization Technology is being used. Virtualization Technology is offered on Intel 3.6 and 3.8 GHz Prescott 2M 32-bit CPUs and many of Intel's 64-bit CPUs.
- Application Virtual Machine: a piece of software that isolates the application from the computer allowing the application to be run on any machine. The most common example of this type of virtual machine is Sun's Java Virtual Machine.

- Virtual Environment or Virtual Private Server: a virtualized environment for running user-level programs. Examples are FreeBSD Jails, Solaris Containers, and OpenVZ.

The remainder of this paper discusses only hardware virtual machines.

In order for virtualization to work, the host software must maintain the mapping between virtual and real resources, enforce separation between the clients, and handle interrupts issued by the clients. The Virtual Machine or Virtual Machine Monitor, depending on the implementation, controls the server hardware and provides an abstracted image of the hardware to the guest virtual machines. This is depicted in Figure 1.

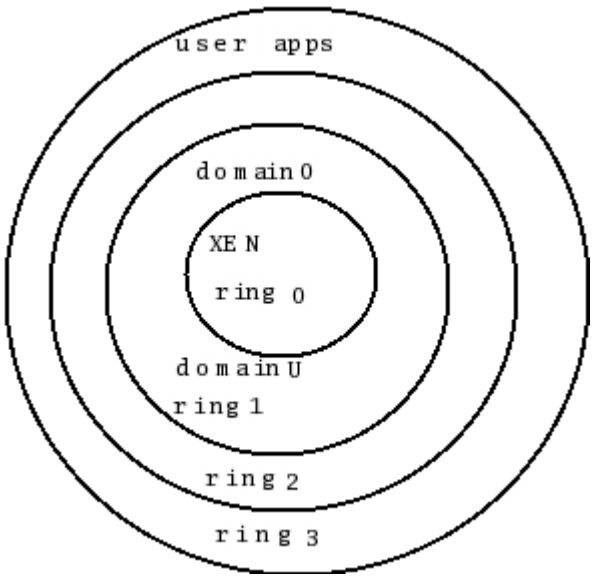

Figure 1 Abstraction and Virtualization. From Rosen, R. (2006). *Virtualization in Xen 3.0,* Linux Journal, March 2006. Retrieved June 21 from http://www.linuxjournal.com/article/8909.

Note that the virtualized server, Xen in this instance, is running in ring 0. It is interfacing directly with the hardware. The virtual client has been moved to ring 1. It “thinks” it is interfacing with the hardware but, in reality, it is interfacing with the abstracted hardware provided by Xen. There are two technical problems to overcome in order to run an application that “thinks” it is a privileged operating system as a unprivileged user application. Rosen notes:

> About 250 instructions are contained in the IA-32 instruction set, of which 17 are problematic in terms of running them in ring 1. These instructions can be problematic in two senses. First, running the instruction in ring 1 can cause a general protection exception (GPE), which also may be called a general protection fault (GPF). For example, running HLT immediately causes a GPF. Some instructions, such as CLI and STI, may can cause a GPF if a certain condition is met. That is, a GPF occurs if the CPL is greater than the IOPL of the current program or procedure and, as a result, has less privilege.
>
> The second problem occurs with instructions that do not cause a GPF but still fail. Many Xen articles use the term "fail silently" to describe these cases. For example, the POPF at the restored EFLAGS has a different interrupt flag (IF) value than the current EFLAGS.

Xen uses two different techniques to overcome these problems. For the first class of problem, the GPF, Xen “traps” the GPF. When a GPF occurs, control is transferred to Xen. Xen, provides, in essence, an interrupt handler. It determines the cause of the GPF and issues the appropriate directives to the hardware to provide the necessary services to the client application.

Xen handles the second class of problem, the “silent failures”, by searching for problematic instructions. It replaces these instructions with what it calls “hypervisor calls”. These are calls to the Xen API that will transparently implement the desired functionality at run-time.

While Xen uses the paravirtualization technique, VMWare uses a modified version of the emulation technique. From Wikipedia:

Conventional emulators like Bochs emulate the microprocessor, executing each guest CPU instruction by calling a software subroutine on the host machine that simulates the function of that CPU instruction. This abstraction allows the guest machine to run on host machines with a different type of microprocessor, but also operates very slowly.

Dynamic recompilation offers an improvement on this approach: it dynamically compiles blocks of machine instructions the first time they are executed, and later uses the translated code directly when the code runs a second time. This approach is taken by Microsoft's Virtual PC for Mac OS X.

VMware Workstation takes an even more optimized approach and uses the CPU to run code directly whenever possible. This is the case for user mode and virtual 8086 mode code on x86. When direct execution is not possible, code is rewritten dynamically. This is the case for kernel-level and real mode code. In VMware's case, the translated code is put into a spare area of memory, typically at the end of the address space, which can then be protected and made invisible using the segmentation mechanisms. For these reasons, VMware is dramatically faster than emulators, running at more than 80% of the speed that the virtual guest OS would run on hardware. VMware boasts an overhead as small as 3%-6% for computationally intensive applications.

Although VMware virtual machines run in user mode, VMware Workstation itself requires installing various drivers in the host operating system, notably in order to dynamically switch the GDT and the IDT tables.

One final note: many people erroneously believe that virtualization products like VMware or Virtual PC replace offending instructions or simply run kernel code in user mode. Neither of these approaches can work on x86. Replacing instructions means that if the code reads itself it will be surprised not to find the expected content; it is not possible to protect code against reading and at the same time allow normal execution; replacing in place is complicated. Running the code unmodified in user mode is not possible either, as most instructions which just read the machine state do not cause an exception and will betray the real state of the program, and certain instructions silently change behavior in user mode. A rewrite is always necessary; a simulation of the current program counter in the original location is performed when necessary and notably hardware code breakpoints are remapped.

In summary, virtualization can be achieved using different approaches. The market leading virtualization vendors, VMWare and Xen, use two different approaches. Xen uses the paravirtualization technique. This provides high performance but requires recompilation of the operating system kernel. Market leader, VMWare, uses a modified version of the full hardware

emulation technique. This technique provides lesser performance but does not require recompilation of the operating system kernel.

Performance Implications of Server Virtualization

This section provides a comparison of performance of native program execution and execution using the two virtualization approaches previously discussed. Figure 2 shows a comparison of an application running native, under Microsoft's Virtual PC, and under VMWare. Throughput under VMWare ranges from 88 percent to 98 percent of native. Astoundingly, "File System" throughput is actually faster under VM! This certainly defies common sense.

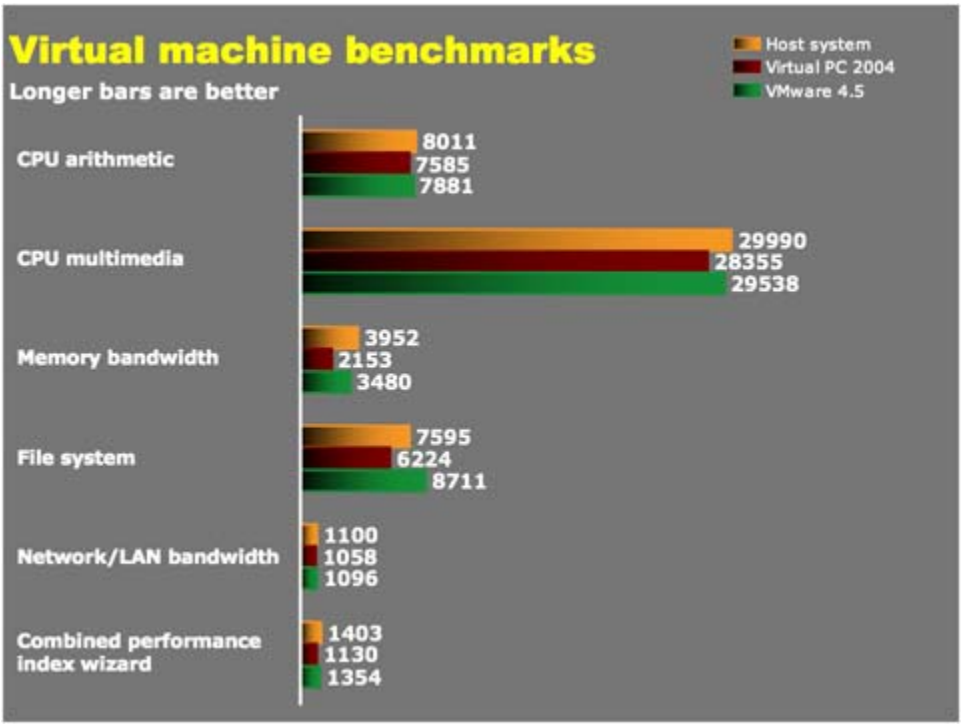

Figure 2 Comparison of Native Performance and Two Virtual Machine Implementations on the SiSoft Sandra VM benchmarks. From Baratz, A. (2004), *Virtual machine shootout: VMware vs. Virtual PC,* Ars Technica, Retrieved June 23, 2006 from http://arstechnica.com/reviews/apps/vm.ars/1

The second comparison is with the SPEC INT benchmark running native and under Xen, Figure 3. Xen uses the paravirtualization technique, which, in theory, should provide better performance than VMWare's full emulation. Although these results cannot be directly compared to the previous, we see that the application performance under Xen is 98 percent of native. This supports the theory that Xen's paravirtualization technique adds very little overhead.

Table 1: SPEC CPU INT2000 suite running on a 733Mhz dual Pentium-III system. XenoLinux *vs.* Linux

	XenoLinux 2.4		Linux 2.4	
benchmark	Run Time	Ratio	Run Time	Ratio
164.gzip	487	287	482	291
175.vpr	640	219	623	225
176.gcc	475	232	454	242
181.mcf	930	194	881	204
186.crafty	316	316	315	317
197.parser	762	236	748	241
252.eon	1824	71.3	1833	70.9
253.perlbmk	530	340	530	340
254.gap	415	265	403	273
255.vortex	640	297	630	302
256.bzip2	692	217	676	222
300.twolf	1236	243	1212	248
SPECint_base2000		229.4		234.0

Figure 3 Performance Under Xen Versus Native. From Harris, T. (2003), *Xen 2002*, University of Cambridge Computer Laboratory Technical Report #553. Retrieved June 23, 1006 from http://research.microsoft.com/~tharris/papers/2002-xen-tech-report.pdf

Shelden's white paper provides extensive benchmarks and recommendations for sizing servers when virtualizing data centers. His studies indicate that we might anticipate 2 to 7 percent CPU overhead while running applications under VMWare, Figure 4.

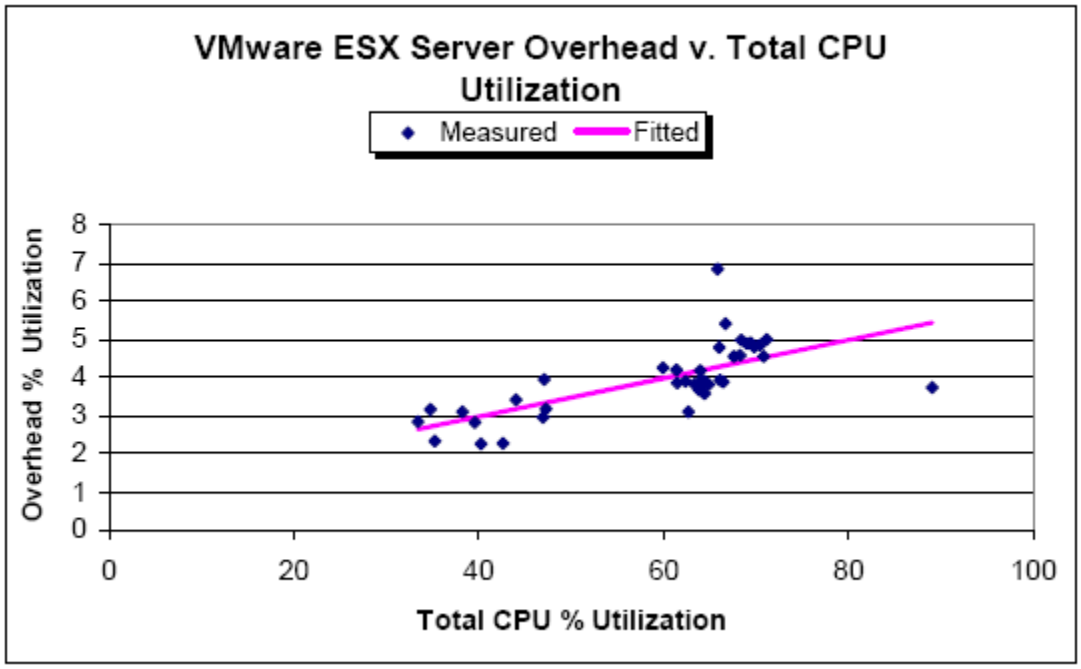

Figure 1 ESX Server Overhead

Figure 4 CPU Overhead Under VMWare. From Shelden, W. (2005). Modeling VMware ESX Performance. Retrieved May 23, 2006, from http://www.perfman.com/resource_center/whitePaperRequest.asp?paperName=7.

In summary, we can estimate that applications running in a virtualized environment will achieve throughputs of, at minimum, eighty percent of native and, perhaps, as much as ninety five percent. These figures demonstrate that virtualization technology is robust and mature. The small performance penalty required for virtualization is insignificant when compared to the large business benefits available to a typical, large, collocated server data center.

A Practical Experiment

In this section, I describe the results of my attempt to build a virtual server running separate Windows and Linux images. The proposed project was to install the Xen or VMWare

virtual server software on a spare computer and then install Linux in one VM and Windows 98SE in another. The final goal was to tailor a Samba server on the Linux VM and install CyberGenie, an orphaned, obsolete, Windows 98-based voice mail application on the Windows VM. Questions to be answered were:

- How mature are the installation packages and documentation?
- What problems are encountered?
- Can both printers be installed on the Linux side?
- Can the Linux side be given the NIC while denying CyberGenie access?
- Are performance issues noted?

I first downloaded a standalone Xen demonstration iso image. This is an operating system that can be booted entirely from CD. It uses part of the computer's RAM as a virtual hard drive and thus does not affect the system on which it is booted. The standalone demo version of Xen booted on an older laptop, an IBM Thinkpad with a 1 GHz Pentium 3 processor and ¼ gigabyte of RAM. However, the demo hung when attempting to start a virtual client. While waiting for class to start, I successfully ran the demo with two virtual guests, one running Debian Linux and one running CentOS Linux on a Colorado Technical University lab computer. The conclusion is that the older laptop did not have sufficient RAM to run the host OS, two guests, and a virtual hard drive.

The next step was to build a server and attempt to run a native host with two guests. I had a server that was used for various experiments on my home LAN. I had previously pulled it from the rack and fitted it with three network interface cards and a wireless network card intending to configure it as an intrusion detection system. That project was put on hold during

the semester. In addition to the network cards, the server had a 1.5 GHz Pentium processor, ½ gigabyte of DDR RAM, and 160 GB and 200 GB hard drives. It would serve as a good representation of a low-end server that might be found in a real-world data center. Therefore, it was the "guinea pig" for this experiment.

Almost immediately, "Murphy" appeared. The server, which had previously been working, was dead. Troubleshooting quickly revealed a dead power supply. Fortunately, a new 500W unit was on hand, see Figure 5.

Figure 5 LED Off Indicates a Dead Power Supply. Fortunately a Spare Is On Hand.

With the experience with the laptop in mind and as the server's case was open for the power supply replacement, adding extra memory at the same time seemed like a good idea. Extra memory is cheap, Figure 6. A quarter gigabyte "stick" of DDR was on hand and was installed.

Figure 6 Memory Is Cheap (After Rebate). From TigerDirect (2006), *48 Hour Hot Deals*, Retrieved June 23, 2006 from http://www.tigerdirect.com/email/wem1141.asp?SRCCODE=WEM1141TT.

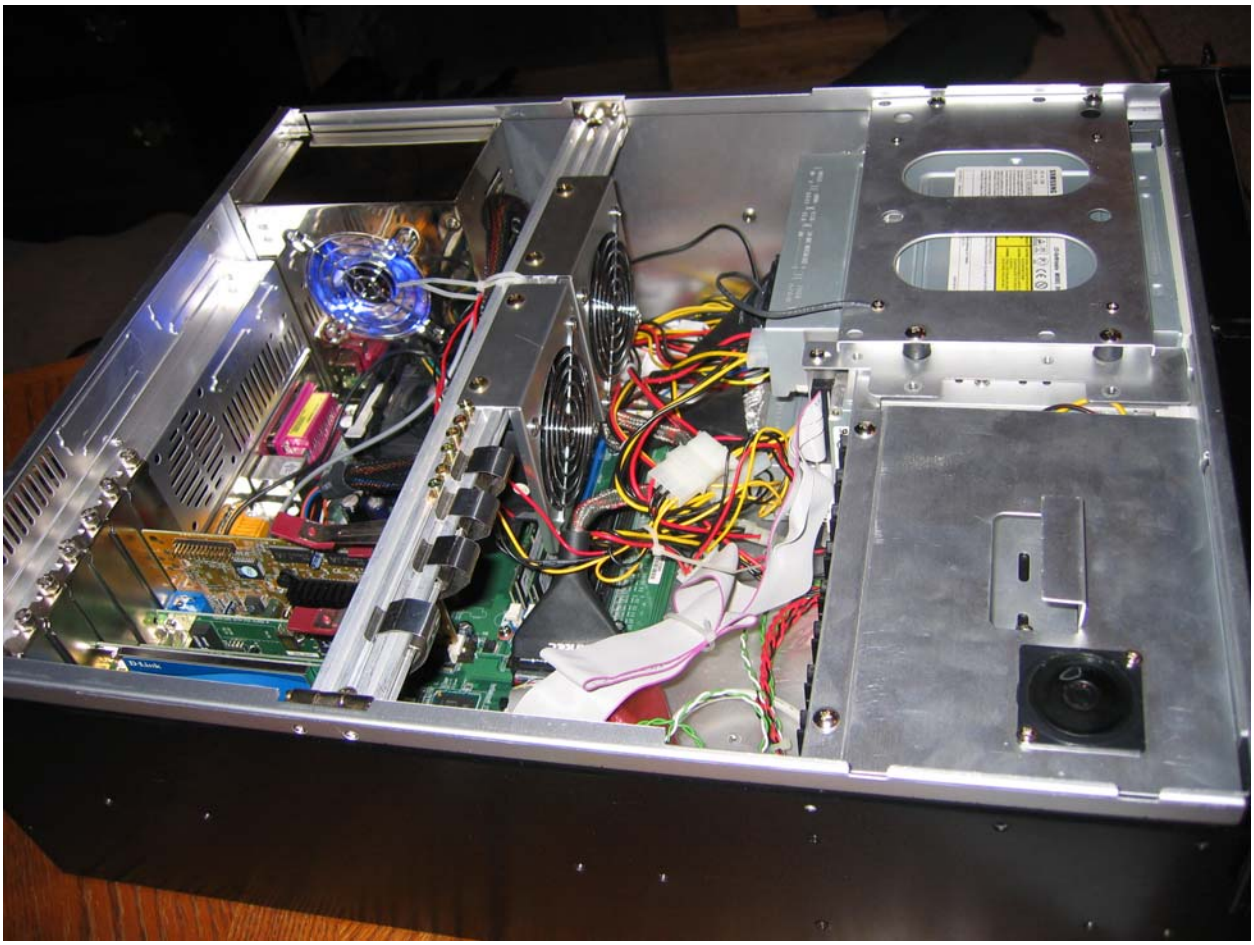

Figure 7 Power Supply Glowing Bluely and Cables Squared Away

With the hardware working, Figure 7, software work could begin. Installing an operating system would be the first step. SuSE 10.0 installation media was available. Since the installation media includes Xen 3.0, it seemed like a good choice, Figure 8. Once again, however, “Murphy” appeared. The once good installation media had become corrupt. Twelve hours later, SuSE 10.1 was downloaded from a nearby mirror. In another 45 minutes, the new OS was “burnt” to a DVD and operating system installation was resumed.

Figure 8 Installing an Operating System

SuSE provides the option to select Xen virtualization during installation. This option was selected. When the installation is completed, the user is given the option of booting native Linux or Xen-enabled Linux. The SuSE tool, Yet Another System Tool, (YAST), has the ability to create and manage Xen virtual machines. YAST was used to create an SuSE virtual machine. The VM was successfully created however, an attempt to install an operating system on the client VM failed, Figure 9.

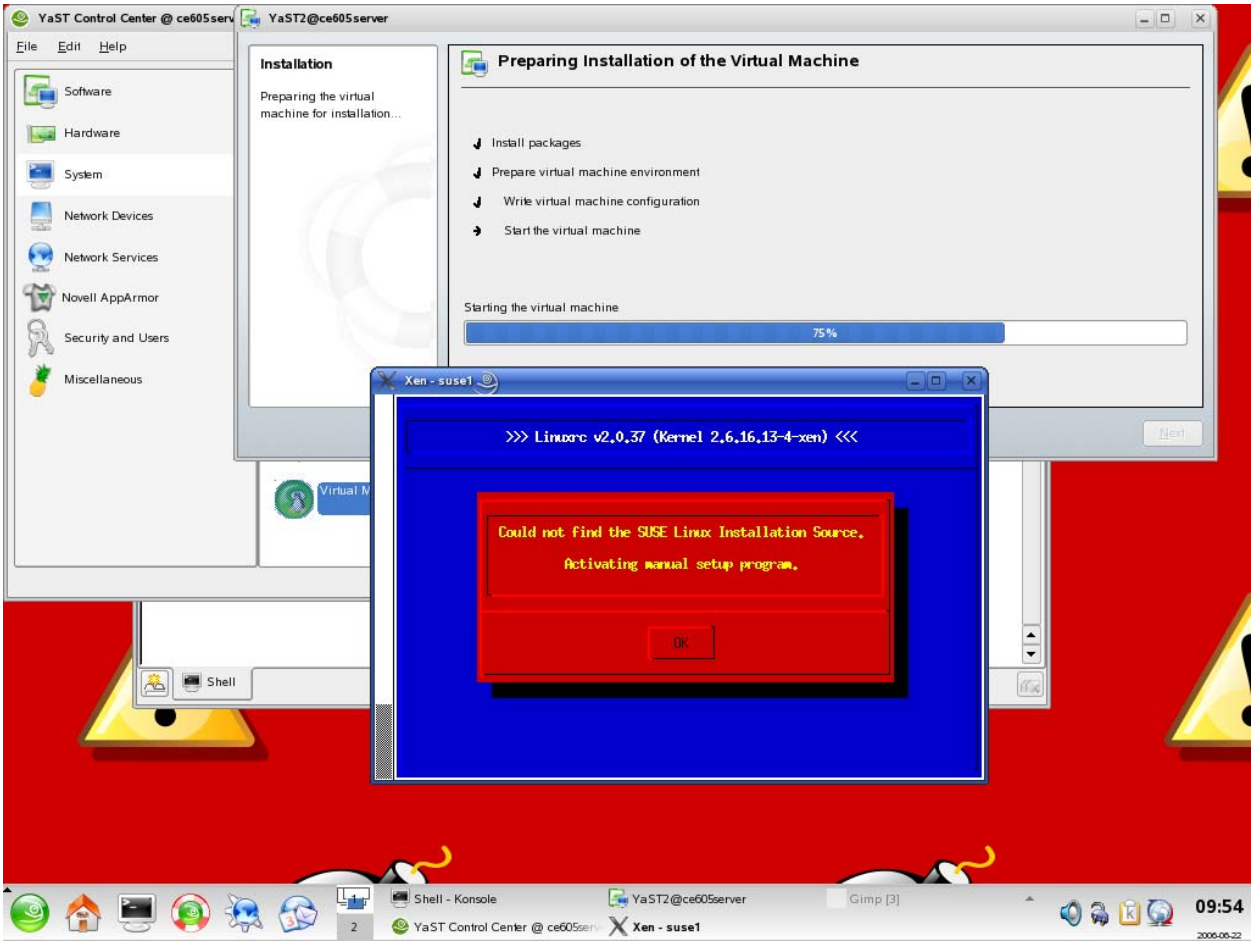

Figure 9 Client VM Installation Comes to an Ignominious End

Having spent much more time than budgeted on the practical implementation of a virtualized server, it was necessary to bring this phase of experimentation to an end. From the experiment, the following conclusions are drawn:

- Xen, using the paravirtualization technique, provides better performance than VMWare, however, the goal of running a Windows 98SE server in a client VM cannot be achieved with Xen. It can be achieved with VMWare (although there was not enough time to demonstrate this during the experiment). This is because

VMWare performs full emulation and, therefore, does not require recompilation of the OS kernel.

- Extensive documentation is available for server virtualization. The automated tools for installing virtual servers and clients, however, are not as mature as the tools available for installing modern operating systems such as Linux or Windows.
- Virtualized server environments are robust and can be implemented by System Administrators but, like installing an operating system from scratch, creating one is beyond the capabilities of a causual user.

Conclusion

Virtual Machine technology has been around since the 1960s. In the "mainframe era", there were compelling business reasons driving the technology. As minicomputers and microcomputers evolved to dominate the market, interest in virtualization waned. The shift towards large, collocated "server farm" data centers is driving a resurgence of interest in virtualization. Again, as in the mainframe era, there are compelling business reasons driving the interest in this technology.

Server virtualization offers many compelling benefits. The ability to dynamically move system images amongst servers facilitates load balancing and restoral of service following a server failure. Virtualization offers the ability to host legacy applications on newer servers and the opportunity to consolidate underutilized servers. To achieve these business benefits, the virtualization technology must be robust and must provide good performance when compared to native execution.

Virtualization technology is robust. This can be seen by comparing benchmark results for the two market-leading virtualization providers. From a practical experiment, it can be seen that the implementation of virtualization technology is not fully mature, however documentation and tools are readily available and rapidly evolving. This is due to the compelling business case for virtualization. It can be expected that installation and deployment of virtual machines will become almost as easy as ordinary operating system installations. Server virtualization is “hot” and it is likely to become the mainstream technology for data centers within the next few years.

References

Baratz, A. (2004), *Virtual Machine Shootout: VMware vs. Virtual PC,* Ars Technica, Retrieved June 23, 2006 from http://arstechnica.com/reviews/apps/vm.ars/1

Creasy, R. (1981). *The Origin of the VM/370 Time-sharing System,* IBM Journal of Research and Development. Volume 25 Number. 5, September 1981. Retrieved May 23, 2006, from http://www.research.ibm.com/journal/rd/255/ibmrd2505M.pdf

Harris, T. (2003), *Xen 2002,* University of Cambridge Computer Laboratory Technical Report #553. Retrieved June 23, 1006 from http://research.microsoft.com/~tharris/papers/2002-xen-tech-report.pdf

Intel Corporation. (No Date). *Virtualization Solutions with Intel® PRO Server Adapters*. Retrieved May 23, 2006, from http://www.intel.com/network/connectivity/solutions/virtualization.htm?ppc_cid=ggl|lad_virtualization|k302E|c

Rosen, R. (2006). *Virtualization in Xen 3.0,* Linux Journal, March 2006. Retrieved June 21, 2006 from http://www.linuxjournal.com/article/8909

Rosenblum, M. (2004). *The Reincarnation of Virtual Machines,* ACM Queue vol. 2, no. 5 - July/August 2004. Retrieved May 23, 2006, from http://acmqueue.com/modules.php?name=Content&pa=showpage&pid=168

Shelden, W. (2005). *Modeling VMware ESX Performance*. Retrieved May 23, 2006, from http://www.perfman.com/resource_center/whitePaperRequest.asp?paperName=7

TigerDirect (2006), *48 Hour Hot Deals*. Retrieved June 23, 2006, from http://www.tigerdirect.com/email/wem1141.asp?SRCCODE=WEM1141TT.

Weilnau, P. (2005). *Measuring Up For Server Virtualization*. Retrieved May 23, 2006, from http://www.perfman.com/resource_center/whitePaperRequest.asp?paperName=6

Wikipedia. (No Date). *Pentium 4*. Retrieved June 22, 2006, from http://en.wikipedia.org/wiki/Pentium_4

Wikipedia. (No Date). *Virtualization*. Retrieved June 23, 2006, from http://en.wikipedia.org/wiki/Virtual_machine

Wikipedia. (No Date). *Virtual Machine*. Retrieved May 23, 2006, from http://en.wikipedia.org/wiki/Virtual_machine

Wikipedia. (No Date). *VMWare*. Retrieved June 23, 2006, from http://en.wikipedia.org/wiki/VMWare