

Computer Sharing Systems using COSMOS Desktop Virtualization

David Peterson, Chief Technology Officer
Fiddlehead Group, C&S Companies

Introduction

Personal computer (PC) replacement and/or cost reduction strategies that use thin client/server-based computing solutions have been a hot topic for enterprises for quite some time. Technologies that drive these solutions include Citrix, Microsoft Terminal Services, and other terminal-based inventions. Although they have the ability to greatly reduce acquisition and operating costs, thin client/server-based computing does not have the same capabilities as a standard PC and usually requires the support of a complex data center. True Computer Sharing, as defined in this paper, will become an important and very different strategy in this area of enterprise solutions and even, as it will be shown, important to the non-enterprise.

Computer Sharing, as defined herein, is *not* a “thin client,” nor is it a clever server, nor is it “fast user switching.” These technologies lack the ability to provide smooth 32-bit video, are unable to launch applications that aren’t designed to be launched with multiple and concurrent users, pirate software, and require more technology to manage. So by being inherently limited, these technologies cannot be true replacements for PCs. On the other hand, Computer Sharing, utilizing the Containerized Operating System for Multiple Operators Simultaneously (COSMOS) does provide all the benefits an actual PC, but without a dedicated PC or any other “thin” hardware. Accomplishing this, Computer Sharing utilizing COSMOS defines a new method of desktop virtualization spawned from a PC, unlike traditional desktop virtualization that originated with a server.

It is, therefore, critical for Computer Sharing and other computer sharing systems developers to understand the power and limitations of traditional desktop virtualization in this space to accurately evaluate what technology is suitable for their products.

There are inherent differences between the server based style of desktop virtualization or traditional desktop virtualization (like VMWARE or XEN) and desktop virtualization as it applies to Computer Sharing or computer sharing systems. The limitations of traditional virtualization approaches will be discussed, specifically as it applies to computer sharing systems and why they are not suitable as an add-on for these computer sharing systems. These limitations relate to the highly-integrated nature of computer sharing systems and their particular reliability and security requirements.

An explanation of how to use a control kernel as a specific approach to new desktop virtualization will also be discussed, as well as why this approach is the only suitable solution for computer sharing

systems. It will detail how a control kernel, especially COSMOS technology, overcame the limitations of plain virtualization. A road map to the future of this technology is also included.

Desktop Virtualizations

A virtual machine (VM) typically used in traditional virtualization provides an environment allowing an operating system to run as an application on another piece of software. To the VM, this “base” software appears as if it is a piece of hardware, which is known as emulation. On the other hand, a control kernel can run as if on bare hardware. In the control kernel environment, such a VM is an efficient, isolated duplicate of the real machine. The control kernel presents an interface that looks like real hardware to the “guest” operating system..

The control kernel has several important characteristics:

1. The control kernel provides an environment that uses as much of the original machine as possible. It is essentially identical with the original machine.
2. Overhead to run the virtual machine is very low. Programs run in this environment show very little decrease in apparent speed.
3. The control kernel is in complete control of system resources.

All three of the aforementioned characteristics are important. The first characteristic ensures that most of the components of the real machine are the same as the components of the VM. This allows the same driver code to work for the control kernel and the VM and is directly related to the second characteristic. Good efficiency ensures the desktop virtualization is usable from the customer’s point of view. The ability of the control kernel to completely control the system resources prevents any software from breaking out of the VM.

How Is It Done?

First, it is important not to use emulation, like QEMU, Bochs, or VirtualPC. Since resources are already shared across multiple seats, any further resource inefficiency will adversely affect the user experience by slowing applications. To be efficient, the vast majority of instructions must be directly executed by the hardware; every interpretation replaces one virtual machine instruction with many instructions, requiring the virtual hardware to be identical to the physical hardware on which the VM is hosted. These instructions, which access physical resources, are intercepted by the VM monitor. Examples of acceptable differences in the instructions include:

- Different memory-management units
- The virtual machine may be an older version of the same basic architecture of the host PC and used to run legacy code

- The virtual machine may be a not yet implemented newer version of the host PC architecture.

As long as the differences are small and the different instructions are not constantly used, the desktop virtualization can be almost as efficient as if the hardware was the same. Any other differences in the directions are not acceptable.

The VM directly executes most instructions; in some cases, this may cause an exception, which invokes the control kernel interpreting the instruction. Because the virtual machine's code is now executing in a non-privileged execution mode on the processor, pure desktop virtualization is achieved. Almost all contemporary systems allowed sensitive instructions that were not privileged to run on the physical, rather than the virtual, machine, causing kernel traps and context errors, not exceptions. This has been addressed by all major processor manufacturers and they have added virtualization extensions allowing the processor to be run in a manner that forces all sensitive instructions to cause exceptions. Unfortunately, this doesn't allow multiple operating systems to run natively; they must be controlled by a virtual machine monitor. Ultimately, the fewer exceptions occurring, the better the environment. Exceptions are expensive and will completely drain the pipeline, delaying processing. Some processors have exception costs that run into the hundreds of CPU cycles.

The two classes of instructions that must be handled by the VM are control sensitive and behavior sensitive instructions. Control sensitive instructions control the machine, and, therefore, interfere with the control kernel's control over resources. Behavior sensitive instructions read the state of the machine, indicating the status of resources, and do not change the resources or their allocations, allowing the instructions to understand what is real and what is virtual. It is important that code running in the VM does not execute sensitive instructions. The control kernel is what ensures that doesn't happen.

Security

Desktop virtualization is used as a method of enhancing security. A virtual machine is, by definition, a subsystem, and as such, it cannot interfere with other subsystems. Desktop virtualization protects critical subsystems, such as the control kernel stack, from a compromised application not procured from a trusted source. CPU core affinities can be set and allow each VM to have its own CPU core, but this threat is relevant even if the application OS runs on its own processor core.

If the operating system is compromised by a buffer or stack overflow, any software running on top of it can be compromised. Other subsystems can be protected by this VM in a properly built control kernel.

The Technology of COSMOS™

Historically, many aspects of computer sharing virtualization have proven to be difficult and seemingly impossible to solve, including high-performance communication, device sharing, security policies, dealing with a trusted computing base, and ensuring the trusted computing base remains small. The technology of COSMOS, overcomes all of these issues. COSMOS is the kernel and user-mode policy module, making it the total trusted computing base. As code progresses, more is possible. The inherent efficiency of the COSMOS code base makes it possible to use mathematical algorithms to enhance the speed and efficiency of the code.

High-Performance Communication

Tight cooperation requires highly-efficient communication between the VM and the host PC. This concept differs from the virtual machine model, in which each VM is considered a system of its own communicating with other systems via file systems or networks. The communication required between components of a computer sharing system uses shared memory and low-latency signaling, demands that simply do not fit the virtual machine model.

This communication requirement has many aspects. The first requirement is the bulk data transfer between subsystems, such as a media file downloaded via the communications subsystem and displayed by a media player. It is important for overall performance, as well as energy conservation, that such data is not copied unnecessarily, which is normally achieved by depositing it in a shared buffer (securely) between subsystems. Although this is not currently supported by the virtual-machine model, COSMOS technology provides a method within the concurrent computer sharing model to allow sophisticated communication.

Device Sharing

The integration of multiple virtual machines requires the sharing of physical devices, which must be accessed in a controlled manner by different VMs and their associated subsystems. A desktop virtualization approach supports running device drivers in their native (guest) OS, but this means a device is owned by a particular guest and not accessible by others, and the guest is trusted to drive the particular device.

Security Policies

Even though virtualization helps to separate different guest operating systems, it does not address security requirements by itself. Even though subsystems must communicate effectively, it must be disabled where it is not needed. If it is not disabled, there could be a leakage of critical information. Specifically, it is absolutely necessary to deliver well-defined security policies defining which communication is allowed across components.

Trusted Computing Base

Many computer sharing systems contain highly security-critical components caused from multiple guest VMs running on the same real machine. Each VM must be particularly well-protected from security compromises.

Since almost all software contains bugs, it is important to keep the security exposure to a minimum by minimizing the amount of code upon which the VMs operate.

Due to its extreme importance, COSMOS's mechanism is highly optimized for minimal latency and maximum efficiency. The control kernel provides the right mechanisms for efficiently supporting desktop virtualization and it serves as the virtual machine manager, catching desktop virtualization traps. Unlike other virtualization approaches, the COSMOS control kernel forwards the exception to a user-mode desktop virtualization component, recompiling the code for future use or forcing it to run in ring 3.

Small Trusted Computing Base

Virtually all of the COSMOS code is built outside of the operating system core. When Linux is used, the kernel itself can be made very small and efficient—around 16,000 lines—enforcing a strict separation of policies and mechanisms. It also ensures high security and helps to prevent denial of service attacks, as well as minimizes the amount of code to be trusted. In contrast to plain virtualization approaches designed to be always used with a guest OS underneath any other software, the amount of trusted user-mode code can be kept much smaller.

The Future: Many Cores

Many questions exist when contemplating the future of computer sharing systems:

- How are computer sharing systems and desktop virtualization going to impact control kernel technology?
- Will control kernel technology become more or less relevant and what is needed to keep it relevant?
- Is this technology heading in the right direction?

One thing that is a certainty is that PCs are becoming much more powerful through the addition of more “cores” to the CPU. The problem with multiple cores in competing technology is the inability of the application software to utilize these cores effectively. Multi-core chips are already common in high-end computing systems and “manycore” (chips containing 16 or more CPUs) are only a few years away. While legacy operating systems will find it increasingly hard to scale to such chips, the COSMOS control kernel technology automatically utilizes these cores and partitions the chip into sub-domains containing a small or moderate number of processors handled by a single guest OS.

Building Value—Computer Sharing and COSMOS Control Kernel Technology

In an environment of almost bare metal desktop virtualization, the guest operating-system is the lowest layer of software on which everything else is built; everything depends on it. It is absolutely essential that computer sharing system developers understand how control kernel technology will meet the challenges of the future. When considering desktop virtualization technology, this understanding doubles in importance. Customer technology requires control kernel technology to adapt to future challenges. COSMOS is unique in several related respects:

- The underlying open-source technology has a long track record of research, development, and testing unmatched by competing products. Recent testing has demonstrated how this code base can be scaled to quad-core processors and beyond.
- The components are evolving and a new, light-weight component technology aimed specifically at computer sharing systems has been developed. These components are the basis for delivering a modern software-engineering framework, along with the control kernel, that will support even higher performance, excellent fault tolerance, and security.
- It is important that results are tested, verified, and ushered (when finished) through triple redundant tollgate procedures using basic operation and installation verification testing, unit testing, regression testing, and failover testing. Verification is also enabled by the efficient and disciplined design of the COSMOS control kernel.
- Most testing is performed for code running on bare metal. It is difficult to test code running on a virtual machine, even if most of the VM is running on bare metal. This is being addressed by examining the methods of running complete timing analyses on the underlying code, which would give the ability to guarantee speeds approaching real-time.
- As development progresses, the current code base will ultimately be replaced with a security enhanced kernel, such as SELINUX, and a smooth upgrade path to this advanced technology will be provided.

COSMOS is the future of computer sharing desktop virtualization technology. The best solution for developers to future-proof their technology is to use COSMOS as the base for their technology. COSMOS, proven in the present, is the future of computer sharing systems desktop virtualization technology.