Linux in the dawn of the virtual era

MANY IN ONE

Virtualization can save you time, money, and effort, but you'll need to find the right tool for the task.

BY WILHELM MEIER, TORSTEN KOCKLER

Of course, you can use CPU cycles to search for extraterrestrial life forms, as the SETI@Home project demonstrates, but that's not all. When one computer is incapable of exploiting the full potential of a modern CPU, you can let multiple computers do so. Multiple virtual servers on a small number of physical hosts put the hardware resources to better use while at the same time consolidating the system landscape.

This month's cover story looks at virtualization in Linux. We'll show you the popular Xen virtual machine monitor, and we'll look at the alternative VServer virtualization tool. We'll also bring you a glimpse of virtualization in the real world with VMware's ESX Server.

En Vogue

Virtualization is one of today's buzz words, although the idea is not new. Since the introduction of the Java programming language, most people have constantly used at least one virtual machine. And maybe some readers will remember the UCSD Pascal p system, one of the first virtual machines for Pascal.

Virtualization at operating system level goes back a long way. The first virtual machine was IBM's VM/CMS from the late sixties. And this technology is still with us today; dubbed z/VM, it supports efficient use of Linux on IBM zSeries servers.

Virtual Machines

A virtual machine typically emulates an execution environment: that is, it emulates the interface to this environment. (In contrast to emulation, simulation would reflect all internal states of the environment at the same time.)

In the Java programming language, we talk about the Java Virtual Machine (JVM), an emulation based on strict specifications [16]. The internal states are not really of interest to the user. The JVM works as a virtual processor within a virtual execution environment.

This kind of emulation is also possible for a complete computing system. The task is not typically handled by the hardware but requires a special supporting software component-- a kind of rudimentary operating system known as a Virtual Machine Monitor (VMM) or hypervisor (Figure 1). However, the hardware, especially the CPU, has to fulfill a few requirements [1].

VmWare [2] and VirtualPC or VirtualServer [3] support this kind of virtualization on the x86 architecture. However, Intel and AMD processors do not provide everything you need to support efficient virtualization [1]. To do so, each machine instruction that allows access to

COVER STORY

Xen 3 26
VServer 32
VMware ESX Server 38

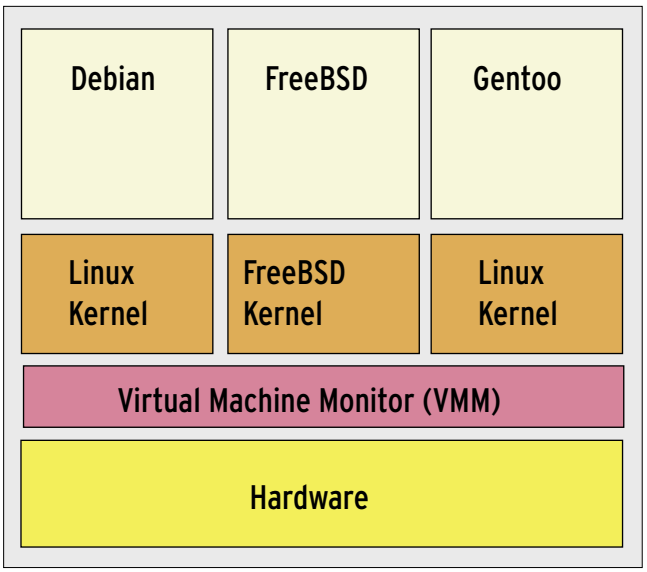

Figure 1: Virtual machine with a Virtual Machine Monitor. VmWare ESX is a practical example of this.

host system resources would need to trigger a software interrupt, unless it is running in one of the processor's privileged modes [17]. Intel's Vanderpool [4] and AMDs Pacifica [5] CPUs will be the first chips to support this ability in an efficient way. Right now, products such as VmWare or VirtualPC use workarounds that can considerably affect performance.

Xen [6] is another hypervisor technique that requires modifications to the guest operating system without Vanderpool or Pacifica processor technology. Xen uses a technique known as para-virtualization, in contrast to the full virtualization used by VmWare or VirtualPC. You'll learn more about Xen and para-virtualization later in this issue.

Full and para-virtualization provide a complete execution environment just like a physical computer system. This is why you need to install an independent operating system kernel on the virtual machine, although it does not need to be the same kernel the host system uses.

The host and guest system kernels can be identical. At first glance, it doesn't seem to make much sense to detour via the virtual machine. After all, the virtual machine monitor simply provides the same execution environment the host system offers. But if you look more closely, you'll see why virtualization makes sense for many environments. For example, you might want to consolidate multiple physical computer systems on a single powerful system to save costs and administrative overhead.

On closer inspection, it is obviously very important to know what you are trying to achieve by introducing virtualization. The trade offs are security, performance, cost, and complexity. You will find an overview of various virtualization approaches at [7].

Defining Goals

If the aim is to provide homogeneous, but separate, virtual execution environments, full virtualization is a useful approach. Even if you have different, physical host systems, the virtual machines can still be set up identically. This approach adds the ability to allocate guest systems arbitrarily, and sometimes dynamically, within a pool of physical systems.

At the same time, over-commitment, that is, multiple planning of physical resources in the virtual machines, can help you save. Of course, this assumes balanced load profiles. Peak loads on multiple virtual machines can't occur at the same time.

The host system and its guests, and the guest systems among themselves, are completely unlinked from a structural point of view, and this is important for security. All of the systems use the same CPU, but it makes no difference if the host system is a separate virtual machine monitor – as is the case with VmWare ESX – on which Linux, FreeBSD, or Windows run as guest systems, or if the virtual machine requires a complete host system, such as Microsoft VirtualPC (Figure 2).

The latter scenario is fairly complex, adding another software component for guest system management on top of the virtualization component.

The CPU performance for non-privileged machine instructions on a virtual machine is theoretically identical to that of the native environment, although the current crop of Intel and AMD CPUs require performance-hitting workarounds. However, the performance of the virtual periphery can be vastly different. In each case, you will need to check whether virtualization will really help you achieve your goals.

Virtual Servers

If requirements do not mandate full virtualization, and if the guest and host operating system kernel can be identical, virtual server environments or operating system partitions may be an option. If an operating system kernel has the ability to allocate processes, and divide the file system and all other resources to an extent where processes on different partitions do not influence each other and resources can't reach other partitions, partitions can be operated more or less like separate physical servers. Applying this principle to a Linux/Unix system, this seems to open up vectors that were previously imperfect or very hard to achieve (Figure 3).

The advantages of partitioning are obvious: less latency and overhead in com-

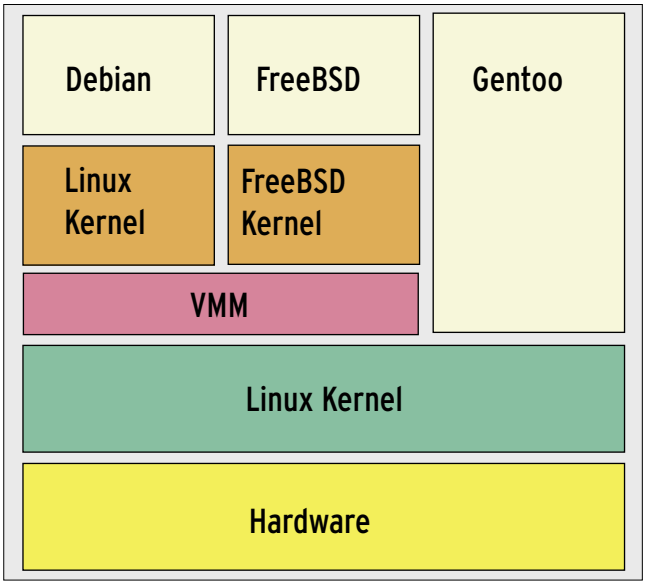

Figure 2: Virtual machine with complete host operating system: this is how Micrososft's VirtualPC works, for example.

parison to virtual machines, where multiple kernels run in parallel or in a hierarchy, and where data are multiply buffered and copied.

Just like with virtual machines, the load profile for the services and applications running on the partitions has to be taken into account. The performance hits in comparison to a native environment may be negligible, but you can't afford to ignore capacity planning, and there are limits to over-commitment.

If threads are not only encapsulated in traditional processes, but additionally assigned to a single partition, root processes on partition A can neither see nor influence processes on partition B. The same thing applies to the filesystem. If partition A is exclusively assigned a subtree below */A*, and partition B a subtree below */B*, the privileges for the omnipotent Unix root user are nicely channeled for one partition. Of course, all other resources, such as network interfaces, need to be allocated in a similar way. Direct hardware access must be ruled out. Access to */dev/kmem* or */dev/sda* must be restricted to root processes on a specific partition.

Projects

A fairly early implementation of this design is to be found in the concept of jails as introduced by FreeBSD [13]. Jails add partitioning of the process space and network infrastructure to the familiar Unix/Linux concept of *chroot()* jails. Privileged processes in a jail environment are no longer capable of performing actions that affect the whole system. For example, it is impossible to load or unload kernel modules, mount filesystems, create device files, or reboot the system within a *jail* environment.

The Linux VServer [8] and OpenVZ [9], along with the commercial variant Virtuozzo [10], which can also be used for Microsoft Windows, are available for the Linux kernel. Sun Solaris 10 and later have a technically comparable product that uses containers or zones [11], and this is also available in OpenSolaris [12], of course. Basically, the same restrictions apply as to FreeBSD jails. On Linux, Linux VServer and OpenVZ use different technical approaches, which are reflected by different kernel patch sets. The userspace tools also differ.

As of this writing, neither Linux VServer nor OpenVZ are part of the official Linux kernel, although the modifications introduced by these projects are quite advanced and stable. Both projects seek the approval of the kernel developers, and it would be nice to see a few basic requirements for virtualization support added to the code base.

Linux VServer

The Linux VServer project is based on mechanisms provided by the current Linux kernel, such as POSIX capabilities for processes [15], namespaces, resource limits, and extended attributes for the filesystem. However, these POISIX features are not sufficient alone to fulfill the requirements. The VServer patches add process contexts, and support binding of processes to network addresses. Besides these critical extensions of basic functionality, the patches add privilege restrictions for all processes within a context, based on POSIX capabilities, as well as the ability to assign files to a specific context.

Practical applications need to handle context-driven accounting and scheduling, and support for many of these tasks is available. Additionally, *util-vserver* provides a very useful userland toolbox.

OpenVZ

Although the commercial Virtuozzo by SWsoft has been on the market for quite some time now, the open source derivative of Virtuozzo, OpenVZ, was released just a few months ago. OpenVZ provides some Linux kernel modifications, adding a few userland tools.

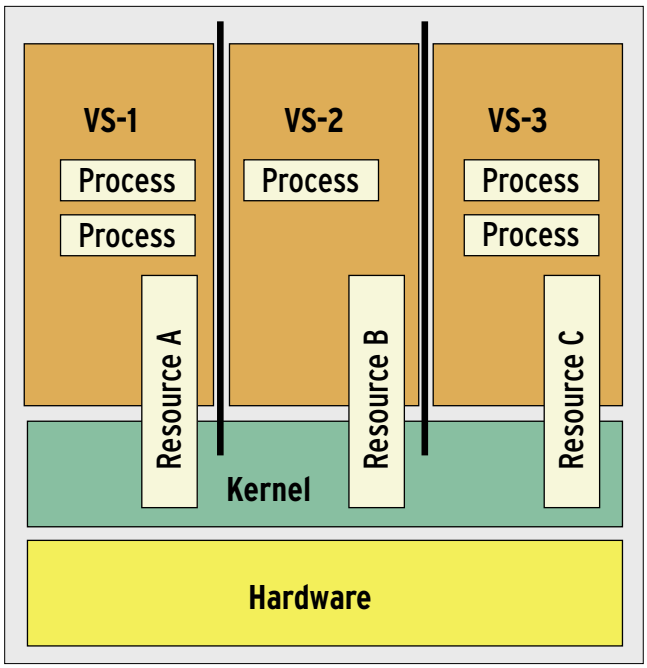

Figure 3: Operating system partitioning. This is the technology used by Linux VServer and OpenVZ, FreeBSD jails and Solaris zones.

Virtuozzo gives commercial customers a full-fledged management suite, including a Management Console (VZMC), and a web-based control center. For professional hosters, SWsoft offers the HSP-complete product, a complete solution for order processing, through provisioning, to billing.

Just like Linux VServer, OpenVZ introduces contexts to isolate processes. Network virtualization does not rely on alias interfaces, meaning that each virtual system can use its own firewall.

Control of the resources used by a VS is based on user bean counters that support a high level of granularity. On the downside, the project currently lacks instruments for syncing the VS filesystems, like Linux VServer's copy-on-write link breaking.

Solaris Zones

Solaris zones are a Solaris 10 container component. Sun has a lot of experience with partitioning on large-scale Unix systems. As far back as 1996, static partitioning was introduced as an option for the E10000 machines. However, partitions were quasi-autarkic machines, each with its own operating system kernel.

The immediate predecessor to the current container was the Resource Manager, which was introduced to Solaris 9. The Resource Manager allowed system administrators to assign processes to pools and to control the resource consumption of these pools. It did not give administrators the ability to isolate execution environments, however; that ability came later with the introduction of zones.

The underlying technology on Solaris Zones is very similar to that used by Linux Vserver and OpenVZ and relies on partitioning the operating system. Powerful tools are available to install and manage zones. Loopback mounts can be used to sync zones, although this is not as efficient as CoW link breaking or UnionFS[18]. ■

THE AUTHOR

Torsten Kockler is an assistant to Professor Wilhelm Meier who teaches operating systems and programming languages in the Department of Computer Science/Microsystem Technology of the Technical University of Kaiserslautern in Zweibrücken, Germany.

Professor Wilhelm Meier teaches operating systems and programming languages in the Department of Computer Science/Microsystem Technology of the Technical University of Kaiserslautern in Zweibrücken, Germany.

INFO

[1] Popek, G.J; Goldberg, R.P; "Formal requirements for virtualizable third generation architectures", Communications of the ACM, Vol. 17, July 1974: *http://portal.acm.org/citation.cfm?id=361011.361073*

[2] VMware: *http://www.vmware.com*

[3] Microsoft Virtual Server and Virtual PC: *http://www.microsoft.com/windows/virtualpc/default.mspx http://www.microsoft.com/windowsserversystem/virtualserver/default.mspx*

[4] Intel virtualization technologies: *http://www.intel.com/technology/computing/vptech/*

[5] AMD Pacifica: *http://www.amdboard.com/pacifica.html*

[6] Xen: *http://www.cl.cam.ac.uk/Research/SRG/netos/xen/*

[7] Overview of virtual machines: *http://en.wikipedia.org/wiki/Comparison_of_virtual_machines*

[8] Linux VServer: *http://linux-vserver.org/*

[9] OpenVZ: *http://openvz.org/*

[10] SW-Soft Virtuozzo: *http://www.virtuozzo.com/*

[11] SUN Solaris 10 containers: *http://www.sun.com/software/solaris/utilization.jsp http://www.usenix.org/events/lisa04/tech/full_papers/price/price_html*

[12] OpenSolaris zones: *http://www.opensolaris.org/os/community/zones/*

[13] FreeBSD jails: *http://www.awprofessional.com/articles/article.asp?p=366888&seqNum=9 http://docs.freebsd.org/44doc/papers/jail/jail.html*

[15] POSIX / IEEE 1003.1e and 1003.2c (withdrawn): *http://wt.xpilot.org/publications/posix.1e/*

[16] JVM-Spec: *http://java.sun.com/docs/books/vmspec/2nd-edition/html/VMSpecTOC.doc.html*

[17] Robin, J.S.; Irvine, C.E.; "Analysis of the Intel Pentium's Ability to Support a Secure Virtual Machine Monitor": *http://www.usenix.org/publications/library/proceedings/sec2000/full_papers/robin/robin.pdf*

[18] UnionFS homepage: *http://www.fsl.cs.sunysb.edu/project-unionfs.html*