Hypervisor

By Contributors to Wikimedia projects

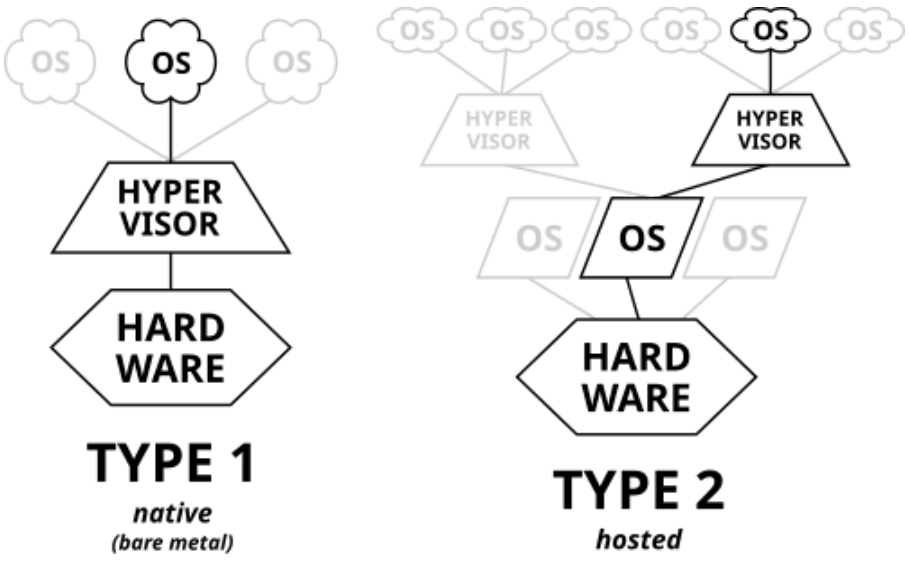
Published: 2004-12-11 · Archived: 2026-04-05 16:02:57 UTC

From Wikipedia, the free encyclopedia

A **hypervisor**, also known as a **virtual machine monitor** (**VMM**), is a type of computer software, firmware or hardware that creates and runs virtual machines. A computer on which a hypervisor runs one or more virtual machines is called a *host machine* or *virtualization server*, and each virtual machine is called a *guest machine*. The hypervisor presents the guest operating systems with a virtual operating platform and manages the execution of the guest operating systems. Unlike an emulator, the guest executes most instructions on the native hardware.[1] Multiple instances of a variety of operating systems may share the virtualized hardware resources: for example, Linux, Windows, and macOS instances can all run on a single physical x86 machine. This contrasts with operating system-level virtualization, where all instances (usually called *containers*) must share a single kernel, though the guest operating systems can differ in user space, such as different Linux distributions with the same kernel.

The term *hypervisor* is a variant of *supervisor*, a traditional term for the kernel of an operating system: the hypervisor is the supervisor of the supervisors,[2] with *hyper-* used as a stronger variant of *super-*.[a] The term dates to circa 1970;[3] IBM coined it for software that ran OS/360 and the 7090 emulator concurrently on the 360/65[4] and later used it for the DIAG handler of CP-67. In the earlier CP/CMS (1967) system, the term *Control Program* was used instead.

Some literature, especially in microkernel contexts, makes a distinction between *hypervisor* and *virtual machine monitor* (*VMM*). There, both components form the overall *virtualization stack* of a certain system. *Hypervisor* refers to kernel-space functionality and VMM to user-space functionality. Specifically in these contexts, a *hypervisor* is a microkernel implementing virtualization infrastructure that must run in kernel-space for technical reasons, such as Intel VMX. Microkernels implementing virtualization mechanisms are also referred to as *microhypervisor*.[5][6] Applying this terminology to Linux, KVM is a *hypervisor* and QEMU or Cloud Hypervisor are VMMs utilizing KVM as hypervisor.[7]



Type-1 and type-2 hypervisors

In his 1973 thesis *Architectural Principles for Virtual Computer Systems*, Robert P. Goldberg classified two types of hypervisor:[1]

Type-1, native or bare-metal hypervisors

These hypervisors run directly on the host's hardware to control the hardware and to manage guest operating systems. For this reason, they are sometimes called bare-metal hypervisors. The first hypervisors, which IBM developed in the 1960s, were native hypervisors.[8] These included the test software SIMMON and the CP/CMS operating system, the predecessor of IBM's VM family of virtual machine operating systems. Examples of Type-1 hypervisor include Hyper-V, Xen and VMware ESXi.

Type-2 or hosted hypervisors

These hypervisors run on a conventional operating system (OS) just as other computer programs do. A virtual machine monitor runs as a process on the host, such as VirtualBox. Type-2 hypervisors abstract guest operating systems from the host operating system, effectively creating an isolated system that can be interacted with by the host. Examples of Type-2 hypervisor include VirtualBox and VMware Workstation.

The distinction between these two types is not always clear. For instance, KVM and bhyve are kernel modules[9] that effectively convert the host operating system to a type-1 hypervisor.[10]

The first hypervisors providing full virtualization were the test tool SIMMON and the one-off IBM CP-40 research system, which began production use in January 1967 and became the first version of the IBM CP/CMS operating system. CP-40 ran on a S/360-40 modified at the Cambridge Scientific Center to support dynamic address translation, a feature that enabled virtualization. Prior to this time, computer hardware had only been virtualized to the extent to allow multiple user applications to run concurrently, such as in CTSS and IBM M44/44X. With CP-40, the hardware's *supervisor state* was virtualized as well, allowing multiple operating systems to run concurrently in separate virtual machine contexts.

Programmers soon implemented CP-40 (as CP-67) for the IBM System/360-67, the first production computer system capable of full virtualization. IBM shipped this machine in 1966; it included page-translation-table hardware for virtual memory and other techniques that allowed a full virtualization of all kernel tasks, including

I/O and interrupt handling. (The "official" operating system, the ill-fated TSS/360, did not employ full virtualization.) Both CP-40 and CP-67 began production use in 1967. CP/CMS was available to IBM customers from 1968 to early 1970s, in source code form without support.

CP/CMS formed part of IBM's attempt to build robust time-sharing systems for its mainframe computers. By running multiple operating systems concurrently, the hypervisor increased system robustness and stability: Even if one operating system crashed, the others would continue working without interruption. Indeed, this even allowed beta or experimental versions of operating systems—or even of new hardware[11]—to be deployed and debugged, without jeopardizing the stable main production system, and without requiring costly additional development systems.

IBM announced its System/370 series in 1970 without the virtual memory feature needed for virtualization, but added it in the August 1972 Advanced Function announcement. Virtualization has been featured in all successor systems, such that all modern-day IBM mainframes, including the zSeries line, retain backward compatibility with the 1960s-era IBM S/360 line. The 1972 announcement also included VM/370, a reimplementation of CP/CMS for the S/370. Unlike CP/CMS, IBM provided support for this version (though it was still distributed in source code form for several releases). *VM* stands for *Virtual Machine*, emphasizing that all, not just some, of the hardware interfaces are virtualized. Both VM and CP/CMS enjoyed early acceptance and rapid development by universities, corporate users, and time-sharing vendors, as well as within IBM. Users played an active role in ongoing development, anticipating trends seen in modern open source projects. However, in a series of disputed and bitter battles[*citation needed*], time-sharing lost out to batch processing through IBM political infighting, and VM remained IBM's "other" mainframe operating system for decades, losing to MVS. It enjoyed a resurgence of popularity and support from 2000 as the z/VM product, for example as the platform for Linux on IBM Z.

As mentioned above, the VM control program includes a *hypervisor-call* handler that intercepts DIAG ("Diagnose", opcode x'83') instructions used within a virtual machine. This provides fast-path non-virtualized execution of file-system access and other operations (DIAG is a model-dependent privileged instruction, not used in normal programming, and thus is not virtualized. It is therefore available for use as a signal to the "host" operating system). When first implemented in CP/CMS release 3.1, this use of DIAG provided an operating system interface that was analogous to the System/360 Supervisor Call instruction (SVC), but that did not require altering or extending the system's virtualization of SVC.

In 1985 IBM introduced the PR/SM hypervisor to manage logical partitions (LPAR).

Operating system support

[edit]

Several factors led to a resurgence around 2005 in the use of virtualization technology among Unix, Linux, and other Unix-like operating systems:[12]

- Expanding hardware capabilities, allowing each single machine to do more simultaneous work
- Efforts to control costs and to simplify management through consolidation of servers

- The need to control large multiprocessor and cluster installations, for example in server farms and render farms
- The improved security, reliability, and device independence possible from hypervisor architectures
- The ability to run complex, OS-dependent applications in different hardware or OS environments
- The ability to overprovision resources, fitting more applications onto a host

Major Unix vendors, including HP, IBM, SGI, and Sun Microsystems, have been selling virtualized hardware since before 2000. These have generally been large, expensive systems (in the multimillion-dollar range at the high end), although virtualization has also been available on some low- and mid-range systems, such as IBM pSeries servers, HP Superdome series machines, and Sun/Oracle SPARC T series CoolThreads servers.

IBM provides virtualization partition technology known as logical partitioning (LPAR) on System/390, zSeries, pSeries and IBM AS/400 systems. For IBM's Power Systems, the POWER Hypervisor (PHYP) is a native (bare-metal) hypervisor in firmware and provides isolation between LPARs. Processor capacity is provided to LPARs in either a dedicated fashion or on an entitlement basis where unused capacity is harvested and can be re-allocated to busy workloads. Groups of LPARs can have their processor capacity managed as if they were in a "pool" - IBM refers to this capability as Multiple Shared-Processor Pools (MSPPs) and implements it in servers with the POWER6 processor. LPAR and MSPP capacity allocations can be dynamically changed. Memory is allocated to each LPAR (at LPAR initiation or dynamically) and is address-controlled by the POWER Hypervisor. For real-mode addressing by operating systems (AIX, Linux, IBM i), the Power processors (POWER4 onwards) have designed virtualization capabilities where a hardware address-offset is evaluated with the OS address-offset to arrive at the physical memory address. Input/Output (I/O) adapters can be exclusively "owned" by LPARs or shared by LPARs through an appliance partition known as the Virtual I/O Server (VIOS). The Power Hypervisor provides for high levels of reliability, availability and serviceability (RAS) by facilitating hot add/replace of multiple parts (model dependent: processors, memory, I/O adapters, blowers, power units, disks, system controllers, etc.)

HPE provides HP Integrity Virtual Machines (Integrity VM) to host multiple operating systems on their Itanium powered Integrity systems. Itanium can run HP-UX, Linux, Windows and OpenVMS, and these environments are also supported as virtual servers on HP's Integrity VM platform. The HP-UX operating system hosts the Integrity VM hypervisor layer that allows for multiple features of HP-UX to be taken advantage of and provides major differentiation between this platform and other commodity platforms - such as processor hotswap, memory hotswap, and dynamic kernel updates without system reboot. While it heavily leverages HP-UX, the Integrity VM hypervisor is really a hybrid that runs on bare-metal while guests are executing. Running normal HP-UX applications on an Integrity VM host is heavily discouraged,[*by whom?*] because Integrity VM implements its own memory management, scheduling and I/O policies that are tuned for virtual machines and are not as effective for normal applications. HPE also provides more rigid partitioning of their Integrity and HP9000 systems by way of VPAR and nPar technology, the former offering shared resource partitioning and the latter offering complete I/O and processing isolation. The flexibility of virtual server environment (VSE) has given way to its use more frequently in newer deployments.[*citation needed*]

Although Solaris has always been the only guest domain OS officially supported by Sun/Oracle on their Logical Domains hypervisor, as of late 2006, Linux (Ubuntu and Gentoo), and FreeBSD have been ported to run on top of the hypervisor (and can all run simultaneously on the same processor, as fully virtualized independent guest

OSes). Wind River "Carrier Grade Linux" also runs on Sun's Hypervisor.[13] Full virtualization on SPARC processors proved straightforward: since its inception in the mid-1980s Sun deliberately kept the SPARC architecture clean of artifacts that would have impeded virtualization. (Compare with virtualization on x86 processors below.)[14]

Similar trends have occurred with x86/x86-64 server platforms, where open-source projects such as Xen have led virtualization efforts. These include hypervisors built on Linux and Solaris kernels as well as custom kernels. Since these technologies span from large systems down to desktops, they are described in the next section.

x86 virtualization was introduced in the 1990s, with its emulation being included in Bochs.[15] Intel and AMD released their first x86 processors with hardware virtualisation in 2005 with Intel VT-x (code-named Vanderpool) and AMD-V (code-named Pacifica).

An alternative approach requires modifying the guest operating system to make a system call to the underlying hypervisor, rather than executing machine I/O instructions that the hypervisor simulates. This is called paravirtualization in Xen, a "hypercall" in Parallels Workstation, and a "DIAGNOSE code" in IBM VM. Some microkernels, such as Mach and L4, are flexible enough to allow paravirtualization of guest operating systems.

Embedded hypervisors, targeting embedded systems and certain real-time operating system (RTOS) environments, are designed with different requirements when compared to desktop and enterprise systems, including robustness, security and real-time capabilities. The resource-constrained nature of multiple embedded systems, especially battery-powered mobile systems, imposes a further requirement for small memory-size and low overhead. Finally, in contrast to the ubiquity of the x86 architecture in the PC world, the embedded world uses a wider variety of architectures and less standardized environments. Support for virtualization requires memory protection (in the form of a memory management unit or at least a memory protection unit) and a distinction between user mode and privileged mode, which rules out most microcontrollers. This still leaves x86, MIPS, ARM and PowerPC as widely deployed architectures on medium- to high-end embedded systems.[16]

As manufacturers of embedded systems usually have the source code to their operating systems, they have less need for full virtualization in this space. Instead, the performance advantages of paravirtualization make this usually the virtualization technology of choice. Nevertheless, ARM and MIPS have recently added full virtualization support as an IP option and has included it in their latest high-end processors and architecture versions, such as ARM Cortex-A15 MPCore and ARMv8 EL2.

Other differences between virtualization in server/desktop and embedded environments include requirements for efficient sharing of resources across virtual machines, high-bandwidth, low-latency inter-VM communication, a global view of scheduling and power management, and fine-grained control of information flows.[17]

Security implications

[edit]

The use of hypervisor technology by malware and rootkits installing themselves as a hypervisor below the operating system, known as *hyperjacking*, can make them more difficult to detect because the malware could intercept any operations of the operating system (such as someone entering a password) without the anti-malware

software necessarily detecting it (since the malware runs below the entire operating system). Implementation of the concept has allegedly occurred in the SubVirt laboratory rootkit (developed jointly by Microsoft and University of Michigan researchers[18]) as well as in the Blue Pill malware package. However, such assertions have been disputed by others who claim that it would be possible to detect the presence of a hypervisor-based rootkit.[19]

In 2009, researchers from Microsoft and North Carolina State University demonstrated a hypervisor-layer anti-rootkit called Hooksafe that can provide generic protection against kernel-mode rootkits.[20]

- Comparison of platform virtualization software
- OS-level virtualization
- Virtual memory

1. ^ *super-* is from Latin, meaning "above", while *hyper-* is from the cognate term in Ancient Greek (ὑπέρ-), also meaning *above* or *over*.

1. ^ Jump up to: **a** **b** *Goldberg, Robert P. (1973). Architectural Principles for Virtual Computer Systems (PDF) (Technical report). Harvard University. ESD-TR-73-105.*
2. ^ *Bernard Golden (2011). Virtualization For Dummies. p. 54.*
3. ^ *"How did the term "hypervisor" come into use?".*
4. ^ *Gary R. Allred (May 1971). System/370 integrated emulation under OS and DOS (PDF). 1971 Spring Joint Computer Conference. Vol. 38. AFIPS Press. p. 164. doi:10.1109/AFIPS.1971.58. Retrieved June 12, 2022.*
5. ^ *Steinberg, Udo; Kauer, Bernhard (2010). "NOVA: A Microhypervisor-Based Secure Virtualization Architecture" (PDF). Proceedings of the 2010 ACM European Conference on Computer Systems (EuroSys 2010). Paris, France. Retrieved August 27, 2024.*
6. ^ *"Hedron Microkernel". GitHub. Cyberus Technology. Retrieved August 27, 2024.*
7. ^ *"Cloud Hypervisor". GitHub. Cloud Hypervisor Project. Retrieved August 27, 2024.*
8. ^ *Meier, Shannon (2008). "IBM Systems Virtualization: Servers, Storage, and Software" (PDF). pp. 2, 15, 20. Retrieved December 22, 2015.*
9. ^ *Dexter, Michael. "Hands-on bhyve". CallForTesting.org. Retrieved September 24, 2013.*
10. ^ *Graziano, Charles (2011). A performance analysis of Xen and KVM hypervisors for hosting the Xen Worlds Project (MS thesis). Iowa State University. doi:10.31274/etd-180810-2322. hdl:20.500.12876/26405. Retrieved October 16, 2022.*
11. ^ See History of CP/CMS for virtual-hardware simulation in the development of the System/370
12. ^ *Loftus, Jack (December 19, 2005). "Xen virtualization quickly becoming open source 'killer app'". TechTarget. Retrieved October 26, 2015.*
13. ^ *"Wind River To Support Sun's Breakthrough UltraSPARC T1 Multithreaded Next-Generation Processor". Wind River Newsroom (Press release). Alameda, California. November 1, 2006. Archived from the original on November 10, 2006. Retrieved October 26, 2015.*
14. ^ *Fritsch, Lothar; Husseiki, Rani; Alkassar, Ammar. Complementary and Alternative Technologies to Trusted Computing (TC-Erg./-A.), Part 1, A study on behalf of the German Federal Office for Information*

Security (BSI) (PDF) (Report). Archived from the original (PDF) on June 7, 2020. Retrieved February 28, 2011.

15. ^ *"Introduction to Bochs"*. bochs.sourceforge.io. Retrieved April 17, 2023.
16. ^ Strobl, Marius (2013). *Virtualization for Reliable Embedded Systems*. Munich: GRIN Publishing GmbH. pp. 5–6. ISBN 978-3-656-49071-5. Retrieved March 7, 2015.
17. ^ *Gernot Heiser* (April 2008). *"The role of virtualization in embedded systems"*. Proc. 1st Workshop on Isolation and Integration in Embedded Systems (IIES'08). pp. 11–16. Archived from the original on March 21, 2012. Retrieved April 8, 2009.
18. ^ *"SubVirt: Implementing malware with virtual machines"* (PDF). *University of Michigan*, *Microsoft*. April 3, 2006. Retrieved September 15, 2008.
19. ^ *"Debunking Blue Pill myth"*. Virtualization.info. August 11, 2006. Archived from the original on February 14, 2010. Retrieved December 10, 2010.
20. ^ Wang, Zhi; Jiang, Xuxian; Cui, Weidong; Ning, Peng (August 11, 2009). "Countering kernel rootkits with lightweight hook protection". *Proceedings of the 16th ACM conference on Computer and communications security* (PDF). CCS '09. Chicago, Illinois, USA: *ACM*. pp. 545–554. *CiteSeerX* 10.1.1.147.9928. *doi*:10.1145/1653662.1653728. *ISBN* 978-1-60558-894-0. *S2CID* 3006492. Retrieved November 11, 2009.

Wikimedia Commons has media related to Hypervisor.

- Hypervisors and Virtual Machines: Implementation Insights on the x86 Architecture
- A Performance Comparison of Hypervisors, VMware
