The MIPS architecture and virtualization

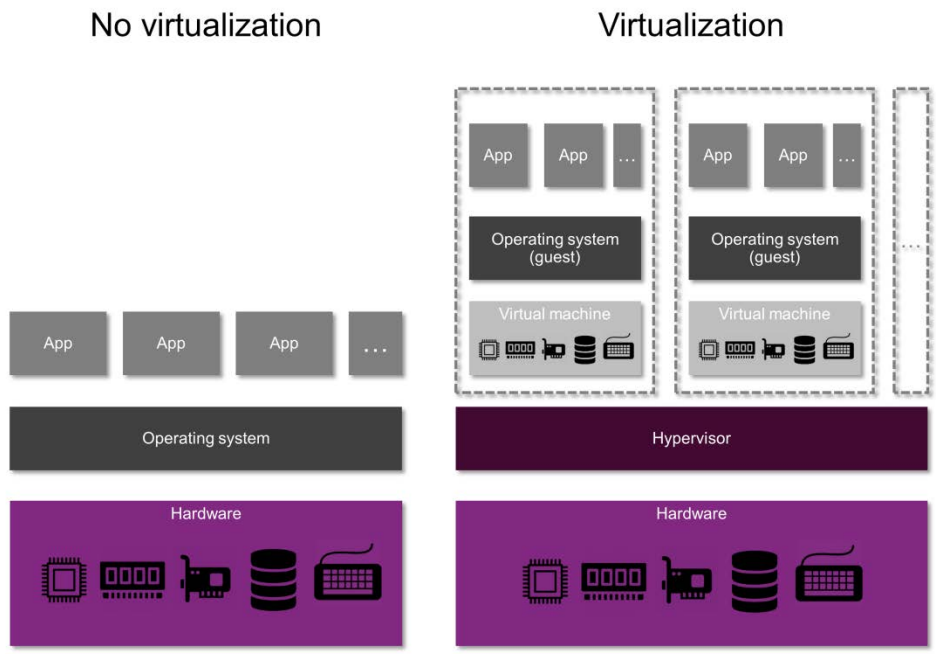
Simply put, virtualization makes one physical device appear as one or more virtual devices. Virtualization can be implemented at the processor level (e.g. CPU virtualization) or at the system level (i.e. SoC virtualization).

Virtualization separates the software running on virtual machines from the underlying hardware resources.

By adding support for virtualization, a platform can create multiple Virtual Machines (VMs), each running its own operating system, embedded programs or a combination of both.

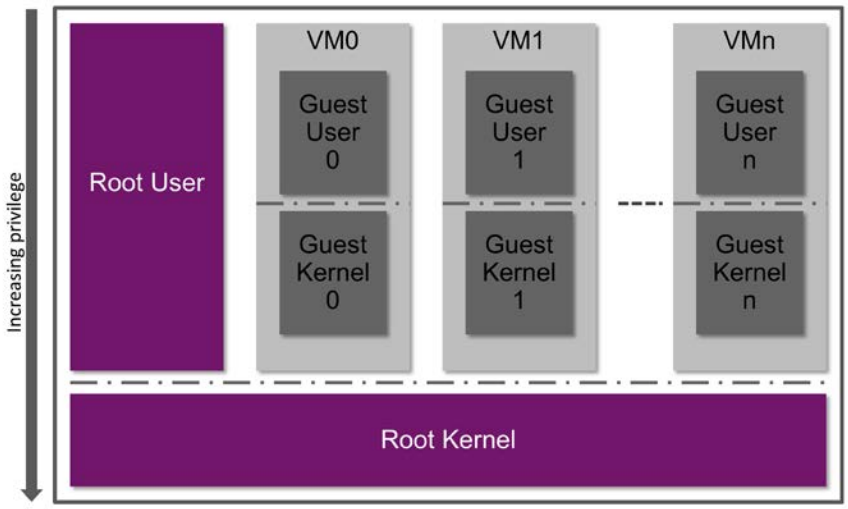
The hardware and software architecture of virtualized platforms

When reading about virtualization, you will come across some commonly-used terms such as host processors, virtual machines, root and guest contexts, and hypervisors.



The host machine represents the actual physical hardware whose resources are to be virtualized. For a MIPS CPU, a host processor is synonymous with a virtual processor (VP) that supports the MIPS our multithreading technology (MIPS MT).

The virtual machine (VM) is the virtual context of a processor created by software. In the case where the host machine supports multiple virtual machines, each guest operating system or computer program will run on a VM in its own context. Therefore, the virtualized system will have one kernel and one user root mode (i.e. the context of the physical system) and multiple pairs of guest kernel and guest user modes, one for each virtual machine.



The software that creates and controls the VM is called a hypervisor. The hypervisor is a crucial component of virtualized systems and represents an intermediate software layer between virtual machines and the actual hardware. Running in the root context, the hypervisor has direct control of the hardware, and thus creates and maintains the trusted execution environments. The main job of a hypervisor it to load and control the software running in each guest context.

It is vital that all software running in a guest context is not able to detect the presence of this intermediate software layer: from the guest context's point of view, the software inside a VM appears to be running directly on a hardware platform.

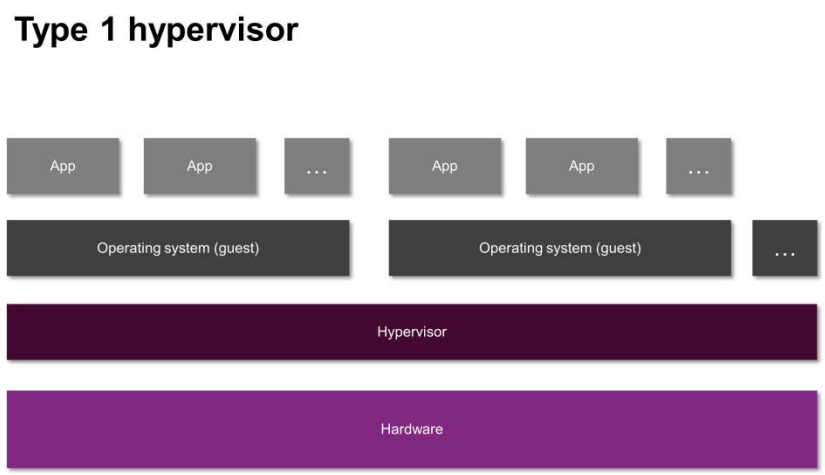
The principles of virtualization

Like multithreading, the idea of virtualization is not new. In 1974, two renowned computer scientists – Gerald J. Popek (UCLA) and Robert P. Goldberg (Harvard University) – wrote an article called *Formal requirements for virtualizable third generation architectures*; this article contains the three pillars for virtualization:

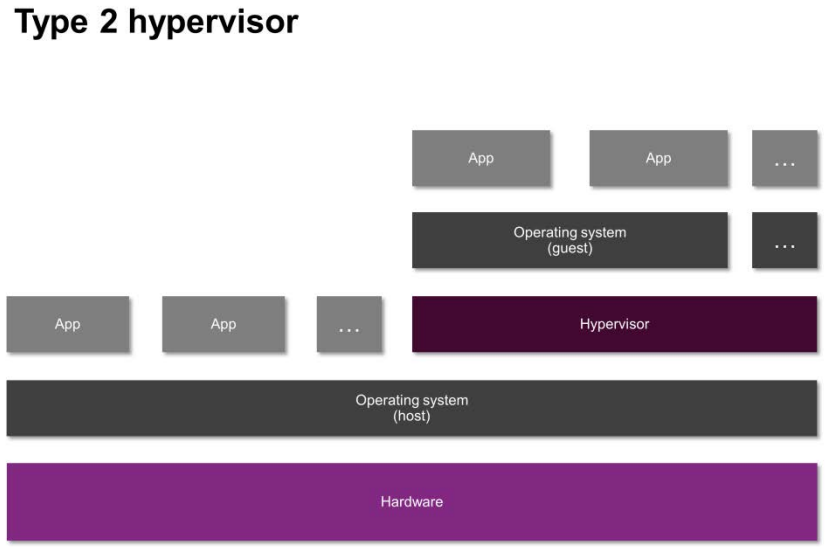
- Equivalence: A program running on a virtual machine should look identical to one running on the actual hardware
- Resource control: A hypervisor must be in complete control of all virtualized resources
- Behaviour sensitive instructions: The majority of instructions executed by virtualized software should not require hypervisor intervention

Hypervisors

There are two types of hypervisors: type 1 (or native) hypervisors runs directly on the hardware while the guest software runs in the VMs. SELTECH FEXEROX and PUCRS Hellfire are two examples of native hypervisors for the MIPS architecture.



A type 2 (or hosted) hypervisor runs within a conventional operating system environment. Type 2 hypervisors are also referred to as trap and emulate hypervisors because all actions by guest software are trapped and the access to hardware is emulated. A common example of a type 2 hypervisor for MIPS CPUs is the KVM hypervisor that runs in the Linux kernel.



On a MIPS CPU, any type 2 hypervisor would run in kernel mode while the guest operating system would run in user mode. In this way, any access that the guest operating system tries to make to a privileged region of memory or execute a privileged instruction will cause an exception that is handled by the hypervisor running in kernel mode.

MIPS virtualization module (MIPS VZ) and OmniShield-ready MIPS CPUs

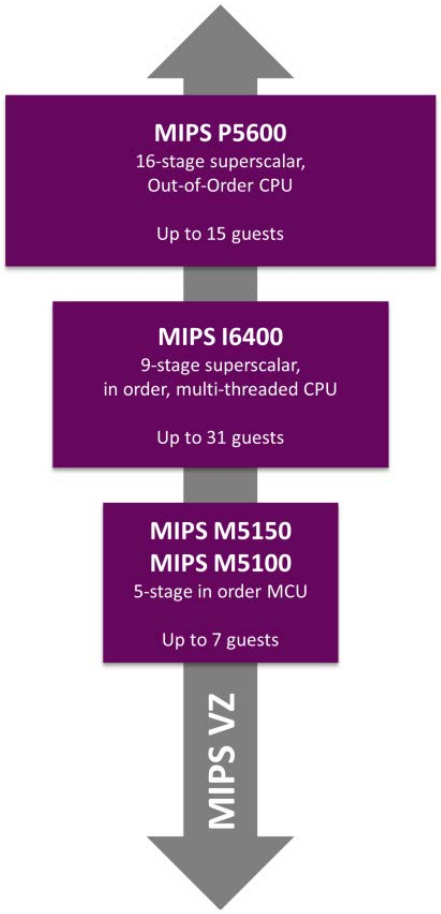
Virtualization can be achieved using software-only methods (e.g. paravirtualization) or by means of hardware assistance (full virtualization). Many MIPS-based processors today support and run paravirtualized operating systems and embedded programs.

Since Release 5, the MIPS architecture has added support for hardware-assisted virtualization. Once incorporated into a MIPS Warrior CPU, full virtualization provides the benefits of improved performance without any modifications of guest software.

MIPS Warrior CPUs support various levels of full, hardware-assisted virtualization. MIPS has implemented this technology across the entire range, from the high-end P- and I-class application processors to M5150 and M5100 microcontroller-class CPUs.

Full virtualization across the range is a unique feature to the MIPS architecture in the CPU IP landscape.

The latest release of the MIPS architecture supports up to 255 guest operating systems or applications. However, the number of maximum guests for each MIPS Warrior CPU has been selected based on a careful analysis of target applications and markets. The diagram below presents an overview of the MIPS Warrior family and the virtualization capabilities for each CPU:

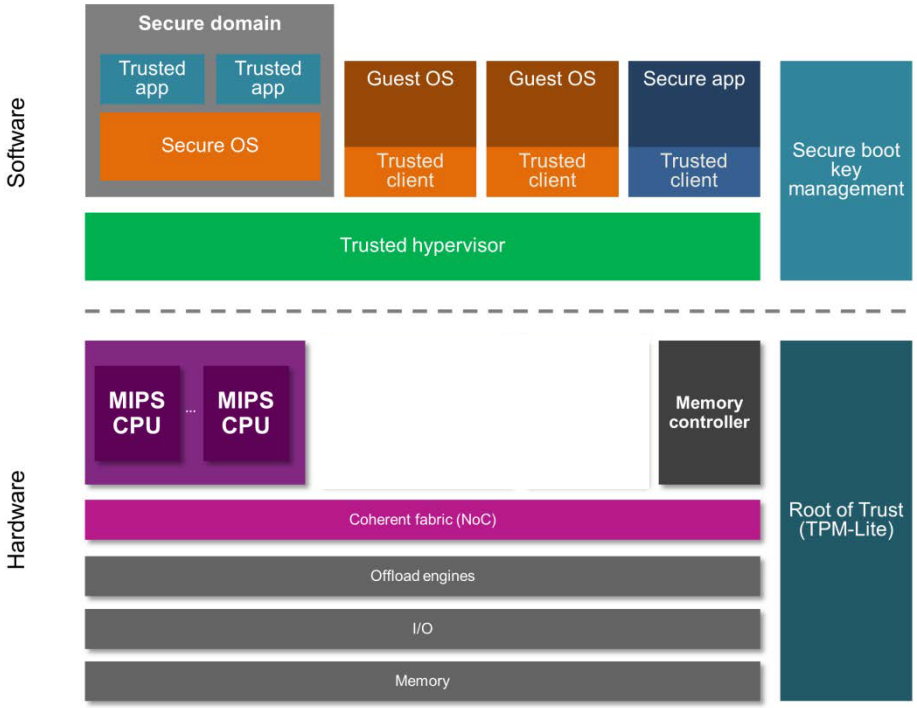


More information on the MIPS VZ module is available at https://www.mips.com/products/architectures/ase/virtualization/.

To address the security and reliability requirements of next-generation connected devices, MIPS has added hardware support for virtualization into its CPU processors.

Implementing full hardware virtualization across MIPS processors enables SoC designers to build OmniShield-ready platforms. OmniShield goes beyond a two-zone approach to create multiple secure domains, where each secure/non-secure application/operating system can operate independently in its own separate environment.

With OmniShield-ready hardware and software IP, Imagination is ensuring that customersôSoCs and OEMsôproducts are designed for security, reliability and dynamic software management.



In addition, OmniShield addresses the scalability that heterogeneous architectures will require. In heterogeneous architectures, application data and resources will be shared between the CPU and other processors in the system. This means that those processors will now face the same level of exposure as the CPU, and must be given the same level of protection ï and currently there is no solution in the market today that addresses this security challenge in its entirety. OmniShield addresses this by protecting all of the processors in an SoC.