

Can hypervisors Stand the Test of Real Time?

Yi Zheng, QNX Software Systems



Hardware virtualization, and the hypervisors that enable it, have become a hot topic in the embedded space. But while virtualization opens up new possibilities for system architects and designers, it also poses new challenges, especially when used with real-time applications. To apply virtualization successfully, system designers must weigh several factors, including resource allocation and ease of implementation; they must also evaluate whether a hypervisor is, in fact, the most effective tool for achieving their design goals.

Hypervisors may be new to embedded systems, but they have a long history in general-purpose computing. For instance, VMWare, a hypervisor for x86 systems, first appeared in 1999. Hypervisors for mainframes appeared even earlier, in the late 1960s.

Hypervisors come in two basic forms: Type 1 and Type 2. Some VMWare products, including VMWare Workstation and VMWare Server, fall into the Type 2 category. As Figure 1 illustrates, a Type 2 hypervisor runs as an application on an operating system (OS) such as Windows or Linux and provides an environment for another operating system, commonly known as the guest operating system. Using this approach, a software designer who uses Windows for corporate applications (email, PowerPoint, etc.) and Linux for design work can quickly switch between the two OS environments.

Can hypervisors Stand the Test of Real Time?

Published on Electronic Component News (<http://www.ecnmag.com>)

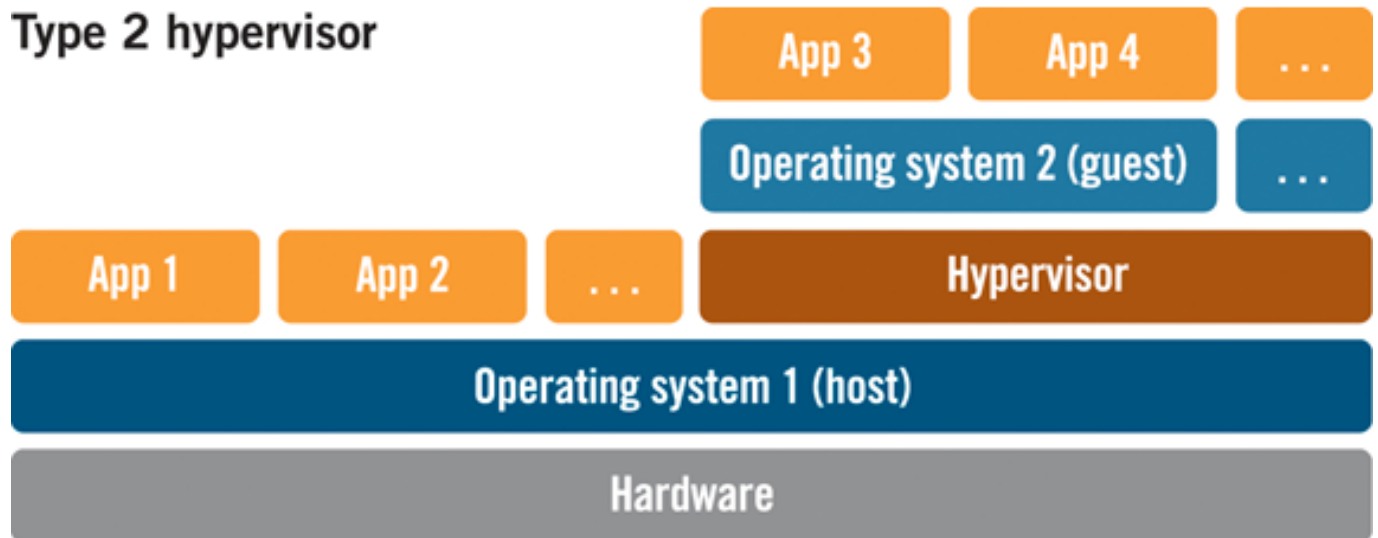


Figure 1. A Type 2 hypervisor runs as an application on a host operating system.

Type 2 hypervisors are easy to use — they typically require no modification to the guest OS. In the case of VMWare, the Workstation software emulates the underlying hardware and provides a full set of virtual hardware resources (mapped by the hypervisor to physical resources) to the guest OS. If the guest OS can run on an x86 machine, it can run within Workstation without modifications. The convenience provided by this type of hypervisor comes with a trade-off: added processing overhead that can compromise real-time performance.

A Type 1 hypervisor has one less layer than a Type 2 hypervisor and sits directly on the host hardware (see Figure 2). Amazon’s well-known Elastic Compute Cloud (EC2) is a web service enabled through a Type 1 hypervisor. Using a Xen hypervisor on top of powerful servers, Amazon can offer “slices” of scalable compute power to its customers, allowing each slice to operate in a virtualized independent environment.

Type 1 hypervisor

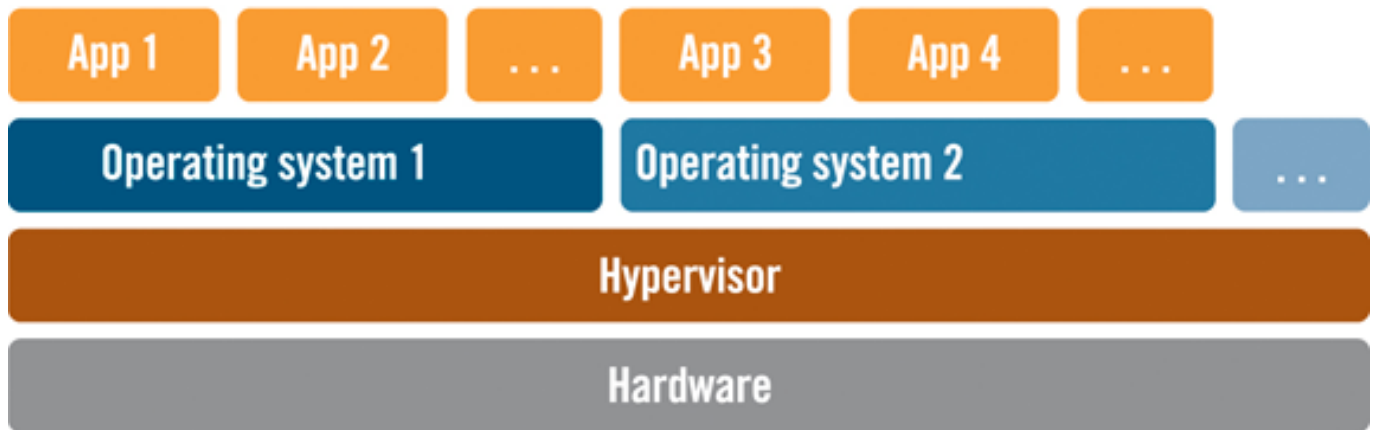


Figure 2. A Type 1 or bare-metal hypervisor sits directly on the host hardware.

A Type 1 hypervisor may be thinner than its Type 2 counterpart, but it typically requires modifications to the guest OS. Thus, the embedded developer must often use a variant of the guest OS created specifically for the hypervisor.

Many Type 1 hypervisors rely on hardware that has virtualization capabilities. Intel's VT, Freescale's embedded hypervisor technology, and ARM's TrustZone are all technologies that provide the foundation for virtualization solutions. As a result, some hardware vendors have joined the hypervisor movement and promote Type 1 hypervisors optimized for their hardware platforms. This approach offers an attractive alternative for systems that demand higher performance.

Intelligent Partitioning

The desire to consolidate hardware or to achieve more efficient usage of hardware is the main driver for hypervisors. In the VMWare Workstation example, virtualization allows the software designer to use just one system instead of two. In the embedded space, the rapid evolution of microprocessors, including multi-core processors, both enables and creates the demand for hardware consolidation. The ideal virtualization solution allows applications traditionally deployed in two or more separate systems to run in a single hardware environment, without software modifications. In the real-time embedded world, the solution must provide this functionality without impacting performance; that is, it must provide intelligent partitioning.

The Amazon EC2 service, with its need to support multiple simultaneous customers from the same server, illustrates the basic requirements for separation and partitioning. Similarly, in the embedded space, the system designer must partition applications to prevent them from interfering with one another. Although virtualization could provide this separation, other options are often easier and cheaper to adopt. For example, a microkernel OS such as the QNX Neutrino RTOS provides full address space separation for applications and system services on both

Can hypervisors Stand the Test of Real Time?

Published on Electronic Component News (<http://www.ecnmag.com>)

multi-core and uni-core processors. A malfunction in one application or service doesn't affect other applications or the microkernel. This ability to protect applications through partitioning is key to safety- or security-critical applications, where the system designer must provide strong evidence that applications operate as expected without interference from neighboring programs on the same hardware.

Distributing resources

For embedded systems, the role of partitioning goes beyond protection of applications from one another. It must deliver another important capability: resource allocation.

Embedded devices typically have limited computing resources, including processor bandwidth and memory space. Hence an effective virtualization solution must provide capabilities to distribute resources in a manner appropriate to each application it supports. For example, a hypervisor should allow a real-time application to process interrupts directly, rather than redirect them on behalf of the application. Meanwhile, it can handle redirects for a general-purpose program without visible side-effects.

Resource allocation becomes critical for applications where operations must complete within a bounded time interval. For such applications, the virtualization technology must be near-transparent, providing resource allocation with little or no overhead. The virtualization layer must also be intelligently discriminative, favoring real-time applications (such as machine control) to those with less stringent real-time requirements (such as a user interface).

The need to ensure sufficient resources for time-critical applications eliminates a Type 2 hypervisor as the virtualization candidate — the extra layer introduces additional latency that violates the real-time requirement. Even a Type 1 hypervisor has added overhead, no matter how thin the hypervisor layer might be. Some hypervisor vendors try to address this problem by giving the real-time OS direct access to the hardware, as described above. For example, the hypervisor from Real-Time Systems lets the user configure the system before runtime so that the real-time OS can perform time-sensitive tasks without added overhead.

Ease of Implementation

Hypervisors are growing in popularity, but their use in embedded systems still requires careful investigation. The choice of hardware (possibly with support for virtualization technology), the choice of hypervisor (which should have minimal impact on real-time performance), the configuration of the hypervisor (to ensure real-time access to hardware resources), the ease of implementation, the need for inter-application communication, and even the availability and usability of tool chains for the hypervisor are all important questions that system designers must address.

For more information, contact QNX Software, 175 Terence Matthews Crescent, Ottawa, Ontario Canada, K2M 1W8; 800 676-0566; www.qnx.com [1]

Can hypervisors Stand the Test of Real Time?

Published on Electronic Component News (<http://www.ecnmag.com>)

Source URL (retrieved on 04/20/2014 - 8:22pm):

<http://www.ecnmag.com/articles/2010/04/can-hypervisors-stand-test-real-time>

Links:

[1] <http://www.qnx.com>