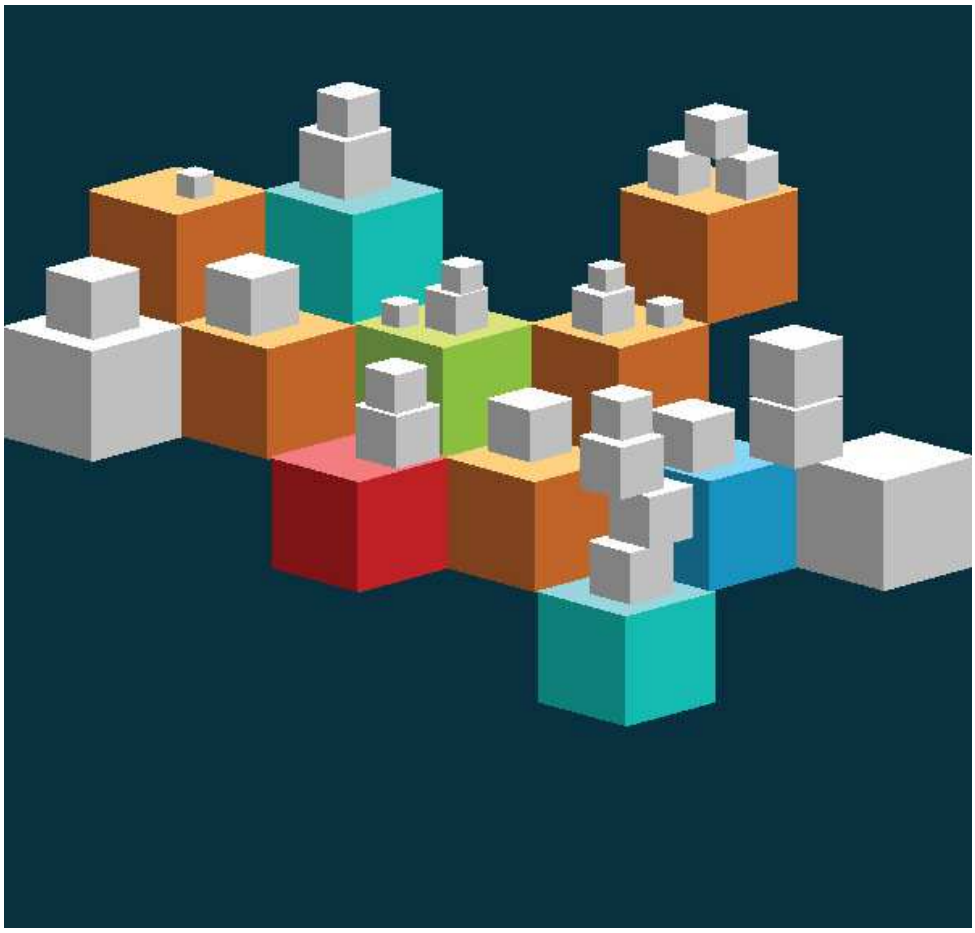
Embedded virtualization: Latest trends and techniques

S. R. Badiger
Department of Electronics & Telecommunication Engineering, SVERI's College of Engineering, Pandharpur



Data center network architectures have been increasingly influencing all areas of embedded systems. Virtualization techniques are commonplace in enterprises and data centers in order to increase capacity and reduce floor space and power consumption. From networking to smart phones, industrial control to point-of-sale systems, the embedded market is also accelerating the adoption of virtualization for some of the same reasons, as well as others unique to embedded systems.

What is Virtualization?

Virtualization is the creation of software abstraction on top of a hardware platform and Operating System (OS) that presents one or more independent virtualized OS environments.

Enterprise and data center environments have been using virtualization for years to maximize server platform performance and run a mix of OS-specific applications on a single machine. They typically take one server blade or

system and run multiple instances of a guest OS and web application server, then load balance requests among these virtual server application environments. This enables a single hardware platform to increase capacity, lower power consumption, and reduce physical footprint for web- and cloud-based services.

Within the enterprise, virtualized environments may also be used to run applications that only run on a specific OS. In these cases virtualization allows a host OS to run a guest OS that in turn runs the desired application. For example, a Windows machine may run a VMWare virtual machine that runs Linux as the guest OS in order to run an application only available on Linux.

How is embedded virtualization different?

Unlike data center and enterprise IT networks, embedded systems span a very large number of processors, OSs, and purpose-built software. So introducing virtualization to the greater embedded systems community isn't just a matter of supporting Windows and Linux on Intel architecture. The primary drivers for virtualization are different as well. Embedded systems typically consist of a real-time component where it is critical to perform specific tasks within a guaranteed time period and a non-real-time component that may include processing real-time information, managing or configuring the system, and use of a Graphical User Interface (GUI).

Without virtualization, the non-real-time components can compromise the real-time nature of the system, so often these non-real-time components must run on a different processor. With virtualization these components can be combined on a single platform while still ensuring the real-time integrity of the system.

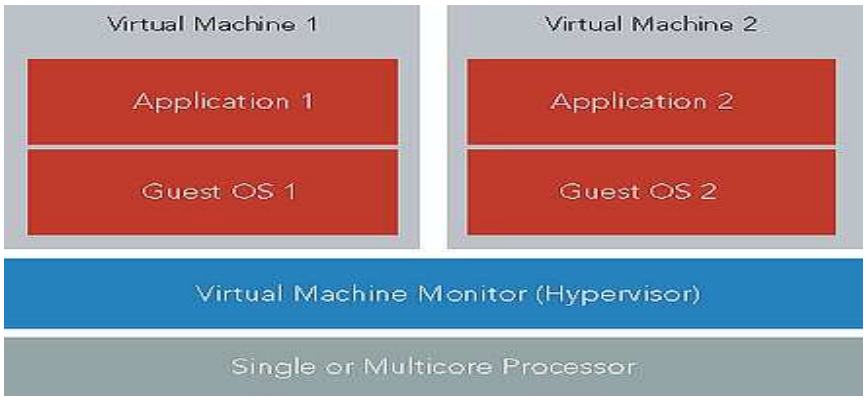


Figure 1: General Embedded Virtualization Architecture

Figure 1 shows general embedded virtualization architecture. For enterprise Virtual Machine (VM) applications, it's typical to have a "host" OS, then running within the host OS a virtual machine running the "guest" OS. Figure 1 shows a component called a Virtual Machine Monitor (VMM) or hypervisor. In embedded systems, this hypervisor is a "to the metal" software layer that abstracts and partitions memory and I/O resources between the virtual machine environments. This approach leads to greater security and isolation between the two virtual environments as well as providing higher performance within each VM.

Technologies enabling embedded virtualization

There are some key capabilities required for embedded virtualization, multicore processors and VM monitors for OSs and processor architectures. In the enterprise/data center world, Intel architecture has been implementing multicore technology into their processors for years now. Having multiple truly independent cores and symmetrical multiprocessing OSs laid the groundwork for the widespread use of virtualization. In the embedded

space, there are even more processor architectures to consider like ARM and its many variants, MIPS, and Free scale/PowerPC/Q or IQ architectures. Many of these processor technologies have only recently started incorporating multi core. Further, hypervisors must be made available for these processor architectures. Hypervisors must also be able to host a variety of real-time and embedded OSs within the embedded world. Many Real-Time Operating System (RTOS) vendors are introducing hypervisors that support Windows and Linux along with their RTOS, which provides an embedded baseline that, enables virtualization.

Where are we in the adoption?

As multi core processors continue to penetrate embedded applications, the use of virtualization is increasing. More complex embedded environments that include a mix of real-time processing with user interfaces, networking, and graphics are the most likely application. Another feature of embedded environments is the need to communicate between the VM environments – the real-time component must often provide the data it's collecting to the non-real-time VM environment for reporting and management. These communications channels are often not needed in the enterprise/data center world since each VM communicates independently.

Linux Works embedded virtualization perspective

Robert Day, Vice President of Sales and Marketing at Linux Works echoed much of this history and current state of the embedded industry and virtualization. "Enterprise systems are nowhere near as diverse as in the embedded systems environment. In addition, embedded environments are constrained – the virtualization layer must deal with specific amounts of memory and accommodate a variety of CPUs and SoC variants."

Day notes that embedded processors are now coming out with capabilities to better support embedded virtualization. Near-native performance is perhaps more important in embedded than enterprise applications, so these hypervisors and their ability to provide a thin virtualization and configuration layer, then "get out of the way" is an important feature that provides the performance requirements the industry needs.

Day references the Type 2 hypervisors that run or depend on another OS, this kind of configuration simply doesn't work in most embedded environments due to losing the near-native performance as well as potential compromise of real-time characteristics. Type 1 hypervisors the software layer running directly on the hardware and providing the resource abstraction to one or more OSs can work, but tend to have a large memory footprint since they often rely on a "helper" OS inside the hypervisor. For this reason, Linux Works coined the term "Type 0 hypervisor" a type of hypervisor that has no OS inside. It's a small piece of software that manages memory, devices, and processor core allocation. The hypervisor contains no drivers – it just tunnels through to the guest OSs. The disadvantage is that it doesn't provide all the capabilities that might be available in the enterprise VM world.

Embedded system developers typically know the platforms their systems run on, what OSs are used, and what the application characteristics are. In these cases, it's acceptable to use a relatively static configuration that gains higher performance at the expense of less flexibility – certainly an acceptable trade-off for embedded systems.

Linux Works has been seeing embedded developers take advantage of virtualization to combine traditionally separate physical systems into one virtualized system. One example Day cited was combining a real-time sensor environment that samples data with the GUI management and reporting system (Figure 2).

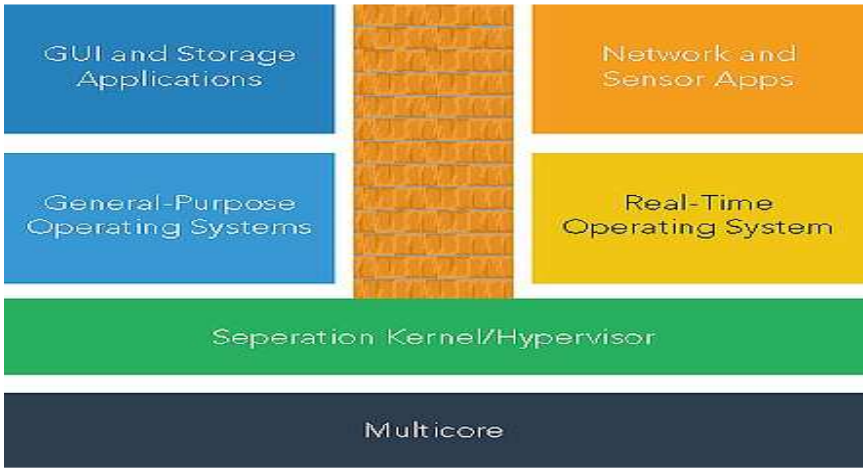


Figure 2: Embedded virtualization example for real-time data sensor sampling and GUI management and reporting.

Processors that incorporate Memory Management Units (MMUs) support the virtualized memory maps well for embedded applications. A more challenging area is the sharing or allocating of I/O devices among or between virtualized environments. “You can build devices on top of the hypervisor and then use these devices to communicate with the guest OSs,” Day says. “This would mean another virtual system vitalizing the device itself.” Here is where an I/O MMU can provide significant help. The IOMMU functions like an MMU for the I/O devices. Essentially the hypervisor partitions devices to go with specific VM environments and the IOMMU is configured to perform these tasks. Cleanly partitioning the IOMMU allows the hypervisor to get out of the way once the device is configured and the VM environment using that device can see near-native performance of the I/O.

Linux Works has seen initial virtualization use cases in the defense applications. The Internet of Things (IoT) revolution is also fueling the embedded virtualization fire.

Virtualization is one of the hottest topics today and its link to malware detection and prevention is another important aspect. Day mentioned that malware detection is built into the Linux Works hypervisor. This involves the hypervisor being able to detect behavior of certain types of malware as the guest OSs run. Because of the privileged nature of the hypervisor, it can look for certain telltale activities of malware going on with the guest OS and flag these. Most virtualized systems have some method to report suspicious things from the hypervisor to a management entity. When the reports are sent, the management entity can take action based on what the hypervisor is reporting. As virus and malware attacks become more purpose-built to attack safety-critical embedded applications, these kinds of watchdog capabilities can be an important line of defense.

Wind River embedded virtualization perspective

Technology experts Glenn Seiler, Vice President of Software Defined Networking and Davide Ricci, Open Source Product Line Manager at Wind River say virtualization is important in the networking world.

A network transformation is underway: The explosion of smart portable devices coupled with their bandwidth-hungry multimedia applications have brought us to a crossroads in the networking world. Like the general embedded world, network infrastructure is taking a page from enterprise and data center distributed architectures to transform the network from a collection of fixed-function infrastructure components to general compute and packet processing platforms that can host and run a variety of network functions. This transformation is called Software Defined

Networking (SDN). Coupled with this initiative is Network Functions Virtualization (NFV), taking networking functionality like bridging, routing, network monitoring, and deep packet inspection and creating software components that can run within a virtualized environment on a piece of SDN infrastructure. This model closely parallels how data centers work today, and it promises to lower operational expense, increase flexibility, and shorten new services deployment.

Seiler mentions that there has been considerable pull from service providers to create NFV-enabled offerings from traditional telecom equipment manufacturers. “Carriers are pushing toward NFV. Wind River has been developing their technical product requirements and virtualization strategy around ETSI NFV specifications. This has been creating a lot of strong demand for virtualization technologies and Wind River has focused a lot of resources on providing carrier-grade virtualization and cloud capabilities around NFV.”

Seiler outlines four important tenets that are needed to support carrier-grade virtualization and NFV:

1. **Reliability and availability**- Network infrastructure is moving toward enterprise and data center architecture, but must do so and maintain carrier-grade reliability and availability.
2. **Performance**- Increasing bandwidths and real-time requirements such as baseband and multimedia streaming requires near-native performance with NFV.
3. **Security**- Intelligent virtualized infrastructure must maintain security and be resistant to malware or viruses that might target network infrastructure.
4. **Manageability**- Virtualized, distributed network components must be able to be managed transparently with existing OSS/BSS and provide the ability to perform reconfiguration and still be resilient to a single point of failure.

Wind River recently announced Wind River Open Virtualization. This is a virtualization environment based on Kernel-based Virtual Machine (KVM) that delivers the performance and management capabilities required by communications service providers. Service provider expectations for NFV are ambitious among them being able to virtualized base stations and radio access network controllers and to support these kinds of baseband protocols at peak capacity, the system has to have significant real-time properties.

Specifically, Wind River looked at interrupt and timer latencies from native running applications versus running on a hypervisor managing the VMs. Ricci mentioned Wind River engineers spent a significant amount of time developing with the KVM open source baseline to provide real-time preemption components with the ability to get near-native performance. Maintaining carrier-grade speeds is especially important for the telecom industry, as performance cannot be compromised.

The future is virtualized

Embedded virtualization is being used in a large number of embedded industry segments for a wide variety of reasons. Near-native performance, maintaining reliability, and the ability to work within constrained environments are the challenges. One thing is clear that virtualization is here and software companies rooted in embedded systems are applying virtualization technologies to meet the demanding requirements of embedded applications.

Source:

http://embedded-computing.com/articles/embedded-virtualization-latest-trends-techniques/