Volume 10 Issue 03 Published, August 10, 2006 ISSN 1535-864X DOI: 10.1535/itj.1003

Intel® Technology Journal

Intel® Virtualization Technology

Intel® Virtualization Technology in Embedded and Communications Infrastructure Applications

Intel® Virtualization Technology in Embedded and Communications Infrastructure Applications

Dean Neumann, Communication Infrastructure Architecture & Planning, Intel Corporation
Dileep Kulkarni, Communication Infrastructure Architecture & Planning, Intel Corporation
Aaron Kunze, Corporate Technology Group, Intel Corporation
Gerald Rogers, Infrastructure Processor Division, Intel Corporation
Edwin Verplanke, Infrastructure Processor Division, Intel Corporation

Index words: virtualization, virtual machine monitor (VMM), hypervisor, real-time operating system (RTOS), multi-core, isolation, consolidation, embedded, communications, industrial control, aerospace

ABSTRACT

Intel® Virtualization Technology$^{\Delta}$ delivers improved computing benefits for home users, business users, and IT managers alike. This paper describes the unique requirements that embedded systems and communications infrastructure equipment place on virtualized environments and shows how Intel is working with a number of third parties to extend the benefits of Intel Virtualization Technology to these market segments. Bounded real-time performance can be maintained while using virtualization to consolidate systems; system uptime can be increased by enabling software failover without redundant hardware; and software migration can be performed without bringing down the application. Virtualization also allows legacy applications to co-exist with new applications by executing both software environments in parallel, and it provides the means for applications to take advantage of multi-core processors without re-architecting for multi-threaded execution.

INTRODUCTION

As in the desktop, mobile, or IT server domains, embedded systems and communications infrastructure equipment can also realize several benefits from virtualizing the hardware execution environment, so that multiple operating systems (OSs) can share the common resources of the hardware platform. These benefits may be realized in terms of cost reduction (either by reducing capital costs or operational costs), in terms of increased performance and functionality, or in terms of increased system reliability and security. There are also several different "usage models" (mechanisms by which virtualization can be used) which provide these benefits. In this paper we survey several of these models within embedded and communication systems.

Regardless of the mechanism by which virtualization is used, one commonality between usage models is that it is always necessary for an additional layer of software to exist that schedules the operating systems which share the hardware platform, manages the resources assigned to each OS, and saves/restores state when context switching between the OSs. In this way each OS executes within a "virtual machine" (VM) rather than on a physical machine. This additional layer of software, the Virtual Machine Monitor (VMM), manages the execution of OSs in much the same way that OSs manage the execution of applications.

Although the existence of a VMM is common to all usage models, we shall see that the architecture of these VMMs is tailored to the constraints of the market segments they address, and that different design decisions must be made to optimize the VMM for the specific requirements of each market segment. It is not only the VMM that is tailored to different market segments, but also the OSs that execute within the VMs that must be tailored to the requirements of these market segments. Whereas a General Purpose-Operating System (GPOS) such as Linux*, Microsoft Windows* or Microsoft Windows Server* addresses the requirements of desktop or IT server environments, a different class of OS—the Real-Time Operating System (RTOS)—is required to address the requirements of embedded and communications systems. As we shall see, providing specific real-time behaviors in a virtualized environment becomes an overriding factor in embedded system design.

EMBEDDED VMM DESIGN CONSIDERATIONS

Embedded systems have requirements that make software design for these systems different than software design for a server or desktop environment. These requirements come from many factors, including the closed nature of the systems and the real-time workloads that often run on these systems. The design requirements of these devices impact the design of VMMs that are targeted to this environment. Although there are some similarities in the requirements of many embedded systems, even within the embedded systems market segments, different vertical market segments have different requirements. Before Intel released processors with Intel Virtualization Technology (Intel VT), the complexity and costs involved in developing a VMM for different embedded systems market segments was extremely high. With Intel VT, however, the unique requirements of embedded systems can be inexpensively met by targeted products like those highlighted in this paper.

In this paper, we highlight three design considerations for VMMs for an embedded system. These design considerations include the unique isolation requirements, the prevalence of static VMs, and support for real-time workloads.

Unique Isolation Requirements

One difference between the requirements of an embedded VMM and a general-purpose VMM affects how well VMs are isolated from one another. Most embedded systems are closed systems, where all of the software is either written by or installed by a single vendor, and where end users do not have the flexibility to run their own software. In some cases this can reduce the isolation requirements that exist in general-purpose VMMs. In embedded systems, this reduced isolation often increases performance or increases the predictability of performance. In other cases such as security-critical applications, the isolation requirements are increased rather than reduced, often to the point of requiring formal analysis and certification.

There are two examples of optimizations supported by embedded VMMs that capitalize on reduced isolation requirements. First, many real-time OSs run without paging [1]. Page walks make it difficult for real-time systems to predict the performance of code and meet real-time guarantees efficiently. They can also result in a reduction in overall performance. In an Intel VT-enabled system, running guests with paging disabled reduces the isolation of that guest—the guest can read and write all of physical memory, including that of devices or other guests. This is a tradeoff that would not be acceptable in a general-purpose environment, but may be preferred in some embedded environments. Second, I/O performance can be critical to the success of an embedded system. This has caused some embedded VMM vendors to allow VMs to have direct access to Direct Memory Access (DMA)-capable devices. This increases performance, but would allow a malicious application or device driver to use the device to read and write the memory of the other guest OSs on the system. Again, in environments where the software environment is fixed, this tradeoff might be acceptable.

Even though some embedded environments will accept reduced isolation between guests, there are exceptions with other embedded environments. Specifically, security-critical and safety-critical environments like those supported by LynuxWorks would not allow these types of tradeoffs [2]. Again, Intel VT reduces the cost of producing a VMM so that the development of targeted VMMs with different design tradeoffs is possible.

Static Virtual Machines

Many embedded systems are designed with the knowledge of exactly what workloads will be run at all times. They are also designed knowing exactly what hardware will be available on the system. This allows designers to make design-time choices about how to allocate hardware to the tasks running in the workload. As such, when using an embedded OS it is typical for the designer to statically allocate specific cores and specific regions of memory to processing tasks. This reduces the emphasis on dynamic scheduling and dynamic memory management that is often important in a GPOS. Although scheduling multiple tasks that are running on the same core is still important, especially in an RTOS, deciding which cores will run which particular tasks is less important.

In a virtualized environment, the VMM is responsible for performing processor scheduling and memory management for the guest OSs just as the OS is responsible for processor scheduling and memory management for applications. Therefore, many of the requirements that apply to OSs in an embedded environment also apply to VMMs in an embedded environment. With this in mind, VMMs can be designed so that system designers can make static configuration choices at design time with respect to how resources are allocated to VMs. This changes the design of VMMs for embedded environments in two ways. First, the design of the scheduler is simplified since the designer will likely statically map tasks to processor cores manually. The scheduler must still schedule between VMs running on the same core, and must do so using different scheduling policies than would be used in desktop or IT data center domains, but scheduling between cores is simplified. Second, memory management is designed more for configurability than for dynamic reallocation. VMMs rely on the designer to specify which memory ranges should be given to each VM. This simplifies the design of memory managers within the VMM, but increases the complexity of configuring the VMM.

Real-time System Support

Many embedded systems are required to support real-time workloads. One of the reasons why GPOSs such as Microsoft Windows cannot provide real-time control is due to the limitations of the scheduler. The scheduler in this case is the entity which determines how much time a specific process or thread is allowed to spend on a given processor. Microsoft Windows uses a quantum-based, preemptive priority scheduling algorithm. This basically means that threads with equal priority are scheduled round robin and threads with higher priority are serviced above threads with a lower priority. Linux uses a priority-based scheduling algorithm as well. An RTOS, however, often schedules tasks using a strict priority scheduler [3]. A priority scheduler will let higher priority tasks run for as long as they have work to do, whereas a general-purpose scheduler tries to guarantee that no task can prevent another from progressing.

Supporting real-time workloads in a virtualized environment is a challenge. A VMM does not typically have visibility into the individual tasks running in an RTOS, nor does it have knowledge about their priority. The solutions that embedded VMM vendors are using today is to isolate RTOSs alone on a processor core, or to only allow the RTOS to share a core with a GPOS and give the RTOS strict priority over the GPOS.

There are also differences between a real-time environment and a general-purpose environment in the way interrupts are handled. In many embedded applications, interrupts generated by external hardware have to be serviced within a specific time frame. GPOSs do not necessarily have this capability built in as they can frequently turn off interrupt processing for an unbounded amount of time. This kind of behavior is unacceptable for applications that require real-time control.

For real-time interrupt handling, Intel VT plays a key role. With Intel VT, the system can be configured such that when a guest disables interrupts, only the delivery of interrupts to itself is disabled and not the delivery to other guests. Therefore, if a GPOS guest were to disable interrupts, and an interrupt for a real-time device were to be asserted, the VMM could still decide to interrupt an RTOS that required small, predictable interrupt latencies. On the other hand, if an interrupt targeted for the GPOS would occur while the RTOS is executing a critical task, the VMM should delay delivery of the interrupt until the RTOS has finished execution of the time-critical task to maintain determinism.

Many embedded systems also require strict prioritization of I/O traffic between code running at different priority levels [4]. For example, if multiple VMs are to access a network device, the application may require some quality of service (QoS) guarantees be enforced between the guests. These guarantees may come in the form of bandwidth or latency guarantees. Such requirements are not typically found in a general-purpose system. This complicates the design of the infrastructure used to share devices between VMs.

Embedded systems have unique design criteria that have a large impact on how VMMs are designed for such systems. Intel VT reduces the cost of producing a VMM such that targeting an individual embedded environment is a possibility. It allows VMM vendors to focus on how they meet the requirements of their market segment without dealing as much with the complexities of virtualizing system hardware.

VIRTUALIZATION IN INDUSTRIAL CONTROL

Many industrial control systems feature highly visual human interfaces that depict the process under control, which may be a medical device, an industrial plant, an assembly line, etc. These displays may also involve rapidly changing data, or they may include interfaces to network-accessible databases to access schematic diagrams or diagnostic and maintenance procedures. Such systems therefore benefit from the ubiquity and richness of GPOSs such as Microsoft Windows and the computational performance of powerful CPUs. Yet they also require strict real-time control to ensure that robotic machines assemble parts with exacting precision, move gantries and X-ray machines to exact locations, operate switches and actuators at precise times, or perform functions for exact durations. Many require closed-loop feedback control systems: for example, if a sensor detects that a machine has reached a specific point, that sensor sends a signal that must be acted upon by the control software within a timeframe that is measured in microseconds in order that the machine can be halted in that exact position without variance.

As we noted above, GPOSs such as Microsoft Windows are not suitable for performing this level of real-time control because they are designed to share processor resources fairly between running processes, thereby preventing "starvation" of some processes. Traditional industrial control systems therefore typically separate their processing and control functions into a Master Station component that implements the human computer interface, database interface, and other non-real-time management functions, and separate Remote Terminal Units (RTUs) which are small, rugged computers that implement the real-time control functions, traditionally using separate Programmable Logic Controllers for that purpose. These RTUs contain the sensors and actuators to detect and control the operating environment, along with sufficient computing performance to react to environmental changes and control commands extremely quickly, and to execute the communications protocol stack to exchange data with the Master Station. RTUs frequently must also operate within extreme temperature, humidity, vibration, or other environmental conditions that are more challenging for the electronics than for the mechanical components. These environmental conditions restrict the choice of computing components that can be employed, or they increase the cost of those components, particularly if ample performance headroom is desired to support future functionality. Therefore, for reasons of cost reduction and the desire for a common, scalable infrastructure upon which greater functionality can be layered, trends in industrial control are toward a consolidated platform that can provide the required separation of the real-time control functionality from the non-real-time functions [5]. In this way RTUs can be simplified, while the Master Station provides real-time control that can be given absolute priority so that signals can be operated upon within strictly defined bounded timeframes, while still providing the rich graphical environment, databases, and device support of a GPOS.

Scheduler and Interrupt Latencies for Virtualized Solutions in Industrial Control

Although the graphical user interface of a GPOS such as Microsoft Windows or Linux combined with XWindows meets the requirements for control of medical equipment or industrial equipment, the applications' response times are often not acceptable. The *average* interrupt latency provided by a GPOS may indeed be within acceptable limits (on the order of 5-20 microseconds), but the *worst-case* latency must be designed for, and this may be orders of magnitude too long in a GPOS.

The main reasons why GPOSs are not suitable are due to the policies of the scheduler and the design of the kernel. The scheduler is the entity that determines how much time a specific task/process/thread is allowed to spend on a given processor. Modern GPOS schedulers allow users to provide an element of application scheduling control. Both Windows and Linux OSs provide such features. Still, even when equipped with these kinds of features, the scheduling algorithms utilized by GPOSs do not provide sufficient real-time control.

The scheduler behavior has an immediate effect on interrupt handling. When controlling a critical process, interrupts generated by the equipment under control must be serviced within a very specific and bounded time frame. GPOSs do not have this capability, as high priority tasks/processes or threads can take priority. This behavior is unacceptable for applications that require real-time control. The design of the OS kernel may also include critical sections of code that must execute atomically, and therefore interrupts must be disabled during these critical sections, thereby causing the worst-case latency to increase by the length of the longest critical section in the OS.

Today there are different solutions available to provide this element of real-time control. Some solutions provide a real-time kernel and run the GPOS and graphical user interface within a complete thread. An API is defined that allows the GPOS to interact with the real-time kernel. If the requirement for thread scheduling exists, an application within the GPOS would utilize the thread mechanism offered by the real-time kernel.

Another approach is to run a small real-time kernel along with the GPOS. Both OSs share the CPU, memory, and interrupt controller. However, each version of the kernel has its own context (descriptor tables, memory management etc.). The real-time kernel determines when specific processes have to be executed to maintain determinism, and interrupt delivery is also controlled so that it does not affect the behavior of the system. If an interrupt occurs during the execution of a real-time task, the real-time kernel will not necessarily execute the interrupt service routine if it is associated with the GPOS. Instead it will continue execution of the real-time task and on completion it will hand over control to the GPOS, which will then deal with the interrupt.

Designing in determinism in a GPOS this way can be very complex. The OS source may not be available: with this in mind, some assumptions must be made in order to predict the behavior of the general-purpose kernel.

Commercial Virtualization Solutions for Industrial Control Applications

Commercial solutions exist that meet the stringent latency constraints of real-time control while still providing the rich and ubiquitous development framework of Microsoft Windows. Products such as TenAsys Corporation's INtime* employ the hardware capabilities of Intel's processors with Intel VT to provide extremely low latencies (worst-case measured as low as 3 microseconds) and real-time capabilities for Microsoft Windows, while allowing Visual Studio* to be used for development in both the Windows environment and the INtime environment. This enables developers to create and deploy sophisticated real-time applications without trying to force a GPOS or device driver to achieve real-time performance.

VIRTUALIZATION IN COMMUNICATIONS NETWORKS

Communication Usage Models

As Intel Corporation and other vendors migrate towards multi-core processors, communications equipment manufacturers are changing their programming paradigms to take advantage of these additional cores. Communications equipment tends to utilize highly specialized software that has been optimized and validated to execute as sequential logic. Thus, it is not easily ported to a multi-core platform. By eliminating the need for equipment manufacturers to refactor their software for multi-threaded execution, Intel VT makes this migration simpler. Equipment manufacturers can instead execute multiple instances of their single-threaded software, each within a separate VM, each processing a portion of the total workload. A suitably architected VMM provides the software infrastructure necessary to distribute the workload between VMs. Examples of multi-core migration include multiple Home Location Registers in a cellular network; or splitting workloads between intrusion detection systems.

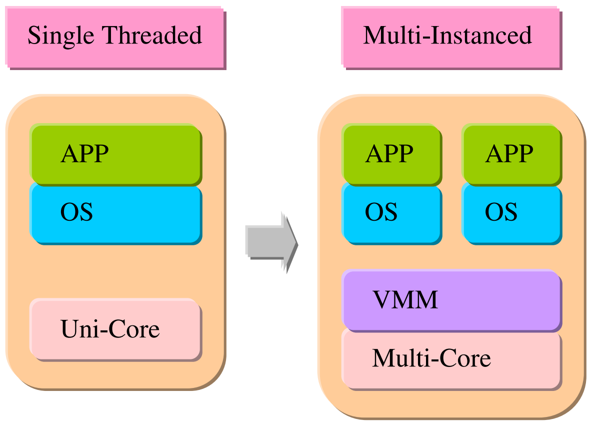

Figure 1: Virtualized vs. non-virtualized environment

Consolidation is common across all market segments, but offers unique benefits in communication market segments. Telecommunications Equipment Manufacturers could utilize a VMM to consolidate multiple instances of an older legacy single threaded application on a multi-core platform, avoiding the need to spend expensive R&D cycles on modifying legacy code to take advantage of multi-core architectures (see Figure 1). Much of the communication equipment processing is split between Data Plane, Control Plane, and Management Plane processing. Each plane has different processing requirements, memory latency and bandwidth requirements, and network I/O requirements. By using Intel VT and a real-time VMM, a manufacturer can consolidate these different planes onto fewer processing elements. This reduces equipment and operational costs, and these savings allow the equipment manufacturers as well as their customers (the service providers) to remain competitive. An example of such a consolidation is in the Mobile Wireless business where a system for determining the current location of a mobile unit, called a Home Location Register (HLR), exists. Many of these systems are proprietary in nature, and restricted to 32-bit addressing. Using Intel VT, more than one HLR can be collocated onto a single system. The VMM allows for the splitting of workloads to multiple HLRs, and allows for a HLR database to be greater than 4 GB in size.

A unique requirement of communication systems is their extremely high reliability. Communication systems may be required to be available to process calls 99.999% of the time. This corresponds to less than five minutes *per year* of downtime, which includes all scheduled maintenance, software and hardware upgrades, and system corrective actions. In comparison, we may spend five minutes *per day* brushing our teeth, so communication systems permit approximately 1/300th the maintenance that we perform on our teeth. Due to the implications on software design, today only high-end communication systems can provide this level of reliability. With Intel VT, communication systems can provide greater availability without the traditional software infrastructure costs. Many of these reliability issues arise from the customized nature of the communication software. Intel VT provides for software fault isolation on all levels of communication systems. This is achieved by allowing Active and Standby instances of the executing software, each within its own VM. In the event of a software failure, the Standby instance will continue execution and assume Active status, while the failed instance is restarted by the VMM. With this capability, the cost of a software fault, which has traditionally been protected against via redundant hardware, is eliminated.

In addition to redundancy, the ability to perform live upgrades of software is accomplished by providing redundant hardware components. As indicated in Figure 2, a Standby partition could be used for either hot upgrades or fault tolerance. With Intel VT, the need for redundant hardware is eliminated. Now simply upgrading the standby instance, restarting it, and designating it the Active instance accomplishes the software upgrade. In the event the new software fails, the previous software version is still available to fall back on.

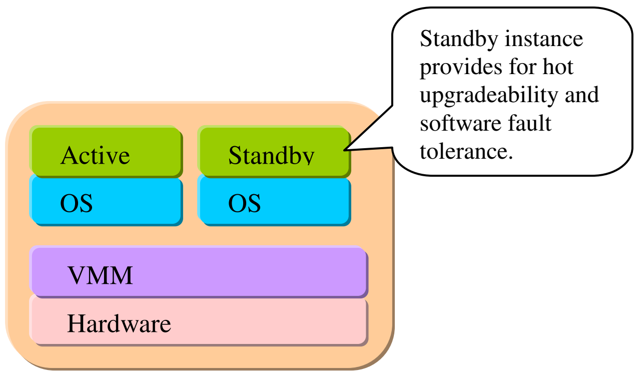
Standby instance provides for hot upgradeability and software fault tolerance.
Active
Standby
OS
OS
VMM
Hardware

Figure 2: Hot upgradeability and fault tolerance

Workload migration is a more common feature of virtualized enterprise servers; however, it also has applicability to the communications market segment. For instance, in many Voice over Internet Protocol (VoIP) implementations, there is a device called a Soft Switch. This switch handles all aspects of call establishment and management. This switch has a set level of capacity, and once exceeded must be replaced or augmented with a new switch. The process of configuring the new switch is very time consuming due to its manual nature. Intel VT simplifies this process by allowing for the migration of a complete switch instance from one hardware platform to another. In addition, expanding a network can be simplified by first performing all configuration in a controlled lab environment and then pushing that configuration to the live switch, thus reducing the risks associated with expansion. Using a test harness and traffic patterns from the live environment, an expansion switch can be fully configured and tested in the lab prior to deployment in the field. Once the expansion switch configuration has been tested, and a migration strategy put into place, the live upgrade can proceed. This migration is shown in Figure 3, where the expansion switch has been added into the network, and a Region, from the installed switch, is being migrated to the expansion switch. This makes for a simpler management model than existing solutions.

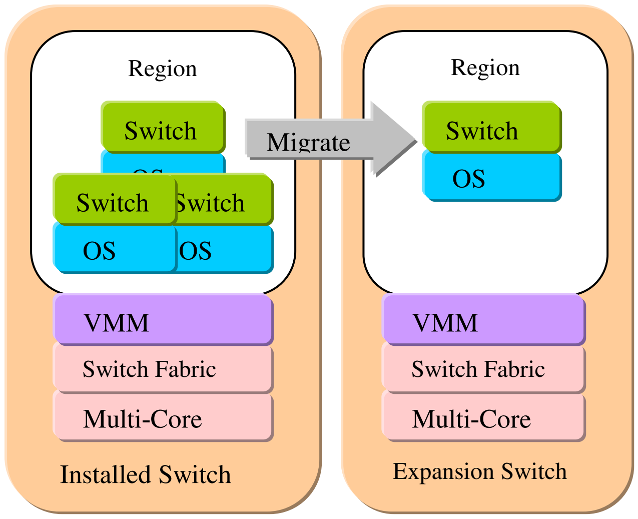

Figure 3: Virtual machine migration

Communication-Oriented Operating Systems

More so than any other market segment, the communications market segment contains many customized home grown OSs. Many times these systems are developed with a specific product in mind and don’t lend themselves well to maintainability either due to complexity or lack of original knowledge. Virtualization allows a company to take advantage of this valuable intellectual property while still moving forward with new technology. By providing an environment within which the proprietary OS can operate, Intel VT allows new development to occur on general-purpose or modern OSs, while providing a link back to the proprietary OS. Intel VT offers the first step in providing support for these legacy OSs. It provides migration to advanced hardware technologies such as multi-core, without requiring multi-processor support within the OS. It also eliminates the need for modification of the OS, and it improves performance by eliminating the need for binary translation. With this capability, the proprietary technology is utilized for the purpose it was intended, and it is saved from costly revalidation and software development efforts.

Sharing vs. Assigning I/O Devices

The communications market segment demands high I/O performance from the hardware/software solution. Cost is always a factor in the design, and obtaining the most performance per watt is a driving fact for every design. In virtualized solutions, two methods exist for providing access to high-performance I/O, namely Shared I/O and Direct Assignment models (i.e., driver domains).

In Shared I/O the VMM (or its host OS) provides access to an I/O device by multiplexing that access through emulation. The guest OSs are presented with a virtual device through which they communicate. The VMM then multiplexes the access from those virtual devices to the real I/O device below. The Shared I/O mechanism results in a performance loss due to the introduction of a multiplexing and emulation layer; yet provides for the most flexibility in migration. Due to this performance impact, shared I/O in communication systems is limited to non-performance critical tasks, such as the management plane.

In Direct I/O Assignment, the VM is assigned an I/O device exclusively. Intel VT for Directed I/O (Intel VT-d) addresses this requirement, and today this assignment occurs on the PCI bus within commercial VMMs architected to address this need. The VMM hides access to PCI devices that are not assigned to a particular guest OS.

Technical challenges exist for Direct I/O Assignment. The biggest challenge comes with those devices that perform DMA operations. Since a guest OS is unaware that it has been moved to a location in memory above its known starting point, it will provide addresses to DMA devices that may reside outside its memory range. To overcome this problem, it is necessary for either the VMM to remap these memory accesses or for hardware to dynamically do so. In the case where the VMM remaps addresses, this either will require that the guest OS be aware of the fact that it will be relocated into a new memory location, or that the VMM restrict the relocation accordingly. In the case where the hardware remaps DMA addresses (as with Intel VT-d), it is necessary that the VMM program the hardware with the VM base address, and that VM's device assignments. Direct I/O Assignment provides an order of magnitude performance improvement over Shared I/O, at the expense of VM dynamic migration ability. This performance improvement is mandatory for all high throughput interfaces in communications equipment and thus the tradeoff is warranted.

Partitioning the Platform for Better Communication Performance

When designing for general-purpose architectures, communication systems designers are often forced into a paradox: They want to leverage GPOSs, various operator interface options, and other general-purpose software, but the networking performance provided by GPOSs is less than acceptable. Virtualization can be used to solve this paradox by creating one partition that executes a minimal OS containing just what is needed to run the performance-critical parts of the application and provide direct access to networking devices, while another partition runs a GPOS that executes those parts of the system that are not performance-critical, such as operator interfaces or management agents for configuration, monitoring, and statistics and alarm reporting. Intel has prototyped an application running on such a system and found that it outperforms the same application running on a GPOS on the same hardware by 24%.

Commercial Virtualization Solutions for Communication Networks

Commercial products such as Jaluna OSware* offer solutions that are optimized to meet the stringent demands of communications equipment. OSware provides a robust platform that offers the key ingredients: Direct and Shared I/O, hard real-time guarantees, bounded interrupt latencies (measured at 21 microseconds), efficient memory virtualization, and the ability to execute both commercial as well as proprietary OSs without requiring them to be modified. Figure 4 shows that OSware provides identical network I/O performance of benchmark applications on RedHat Enterprise Linux* when executing in virtualized and non-virtualized environments.

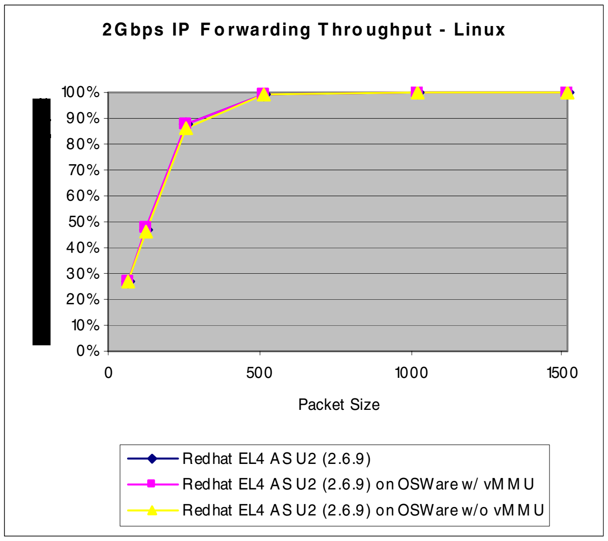

Figure 4: Virtualized vs. native OS networking performance

VIRTUALIZATION IN SAFETY-CRITICAL APPLICATIONS

The US government is migrating its Department of Defense, Department of Energy, and Homeland Security infrastructures from proprietary systems developed solely for government specifications to commercial off-the-shelf (COTS)-based systems with incremental security and reliability requirements. It is easy to imagine that the efficiencies and cost savings resulting from migrating to COTS systems would easily run into the billions of dollars, but the real benefits lie beyond that. Rapid deployment of new technologies allows the US armed forces to retain the technological superiority so vital to their military and intelligence actions. Modern COTS-based systems permit increasingly sophisticated security methods to be employed to safeguard data while permitting the sharing of data that has proven very difficult across different proprietary architectures of the past. Safety-critical systems are also found in many other non-governmental applications where human life is at stake, such as aerospace (flight control systems).

A major challenge in migrating to COTS architectures is ensuring the security of both the hardware and software elements. The Federal Aviation Administration (FAA) has established criteria for certifying software for safety-critical aviation systems, and likewise the National Institute of Standards and Technology (NIST) and the National Security Agency (NSA) have established a common criteria for evaluation of technology products for security-critical systems. An enabling architecture known as Multiple Independent Levels of Security (MILS) is in the process of dramatically reducing the size and complexity of security-critical code, thus allowing faster and more cost-effective development and evaluation.

The MILS architecture defines four conceptual layers of separation:

- separation kernel and hardware
- middleware services
- trusted applications
- distributed communications

Our focus in this discussion is mainly on the MILS separation kernel. The separation kernel must be mathematically verified and evaluated. This practically limits kernel size to less than 5,000 lines of code. Also, the separation kernel must be completely isolated from other layers of software including OS services, which themselves must also be separated from other middleware components.

Intel VT is ideally suited to meet these separation kernel requirements. Figure 5 illustrates how Intel's family of virtualization technologies provides the foundation for an implementation of the MILS architecture.

Benefits of Intel Virtualization Technology

In summary, the benefits of Intel VT are these:

- It provides the separate root ring structure necessary for isolation of separation kernel from non-separation kernel services.
- Just as we would not expect a minivan to do the same job as a pickup truck, we cannot expect a desktop-oriented OS or a desktop-oriented VMM to operate within the constraints of embedded, communications or safety-critical environments, and still provide the functionality, configurability, separation, or performance of solutions that have been architected specifically for those attributes.
- It simplifies VMM design keeping the separation kernel code very small and thus making it possible to build a mathematically verifiable separation kernel.
- It simplifies the migration of single-threaded legacy software to multi-core processors by allowing virtualization of unmodified OSs. This gives end customers an option to simultaneously run multiple instances of non-SMP OSs.
- Intel VT-d allows for direct access to assigned devices. Separation of network interfaces is an essential component of system security. Intel's family of virtualization technologies will be extended to allow efficient sharing of physical I/O devices among VMs without requiring a "service" partition that has access to all network traffic, thus allowing the directing of network traffic to the specific guest OS and application for which it is intended.
- Intel VT also supports the use of a Trusted Platform Module (TPM) to provide the ability to authenticate both the VMM and the guest OSs and applications, to ensure that their image on disk has not been tampered with between reboots. The TPM is a microcontroller that stores keys, passwords, and digital certificates. Microcontrollers that adhere to the TPM specification as defined by the Trusted Computing Group [6] are available from a number of manufacturers.

Commercial Virtualization Solutions for Safety-Critical Applications

Safety-critical systems and security-critical systems are being developed using Intel VT by companies such as LynuxWorks, which provides its LynxOS* RTOS and LynxOS-178* safety-critical RTOS and corresponding development tools. Intel and LynuxWorks are working together to demonstrate the MILS architecture shown in Figure 5 using Intel® Core™ Duo processors. The LynuxWorks separation kernel has been developed to be mathematically verifiable, and it utilizes Intel VT and Intel® EM64T$^{\Phi}$ technologies to support virtualization and both 32-bit and 64-bit operating modes. It provides SMP support and is architected to take full advantage of Intel®

multi-core processors and their various platform-enhancing technologies.

CONCLUSION

We have shown that virtualization has many varied uses in embedded systems, communications infrastructure, and safety- and security-critical environments. The requirements of these domains are however quite different from one another and from the more familiar desktop, mobile, and IT data center environments. These requirements dictate that different architectural and design tradeoffs be made within the VMM and the guest OSs executing within the VMs.

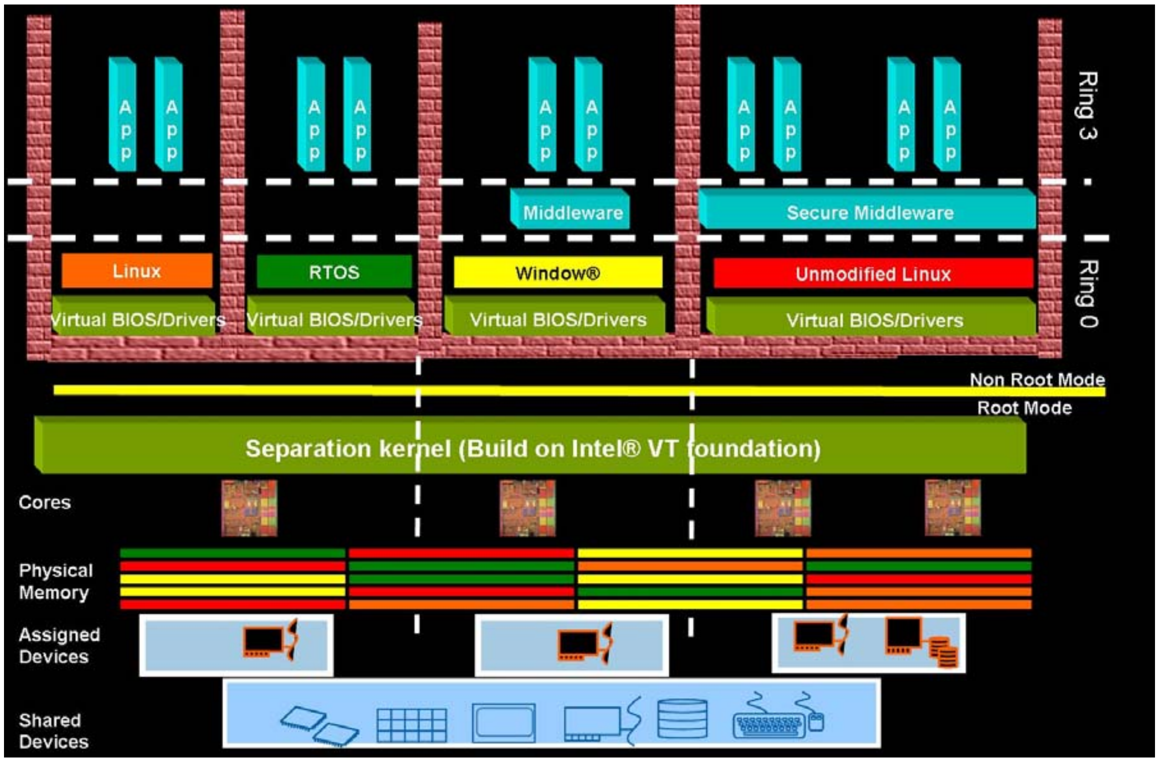

Figure 5: Example of MILS architecture with Intel Virtualization Technology

REFERENCES

[1] Abrossimov, E., Rozier, M., and Shapiro, M., "Generic virtual memory management for operating system kernels," in *Proceedings of the twelfth ACM symposium on Operating systems principles*, 1989, pp. 123–136.

[2] United States Department of Defense, *Department Of Defense Trusted Computer System Evaluation Criteria*, 1985.

[3] Ramamritham, K., Stankovic, J.A., "Scheduling algorithms and operating systems support for real-time systems," in *Proceedings of the IEEE,* January 1994, pp. 55–67.

[4] Kuhns, F., Schmidt, D., Levine, D., "The Design and Performance of a Real-time I/O Subsystem," in *Proceedings of the Fifth IEEE Real-Time Technology and Applications Symposium*, June 1999, pp. 154–163.

[5] Falco, J, Gilsinn, J., and, Stouffer, K., "IT Security for Industrial Control Systems: Requirements Specification and Performance Testing," *2004 NDIA Homeland Security Symposium & Exhibition.*

[6] Trusted Computing Group, at http://www.trustedcomputinggroup.org*.

AUTHORS' BIOGRAPHIES

Dean Neumann specializes in strategic planning of advanced technologies for Intel's Communication Infrastructure Group. He holds Bachelor's and Master's degrees in Computer Science, and has 20 years of experience in engineering fault tolerant and highly available systems for the communications and financial industries. His e-mail is dean.neumann at intel.com.

Dileep Kulkarni specializes in strategic planning of advanced technologies for Intel's Communication Infrastructure Group. He holds an Engineering degree from the Indian Institute of Technology and an M.B.A.

degree from the University of Michigan He has broad experience in planning, marketing, sales, and finance. His e-mail is dileep.kulkarni at intel.com.

Aaron Kunze is a network software engineer in Intel's Corporate Technology Group. He has a Bachelor's degree from Purdue University and a Master's degree from Oregon Graduate Institute. Aaron's area of research is in the design of communication systems, and he co-wrote two books about Intel's IXP Network Processors. His e-mail is aaron.kunze at intel.com.

Gerald Rogers is a software engineer specializing in Performance Analysis of Intel® Architecture for Communications, and he works for Intel's Infrastructure Processor Division. He holds a Bachelors degree in Electrical Engineering and a Masters degree in Computer Science. He has 16 years of embedded software development experience in the Telecommunications and Networking industry. His e-mail is gerald.rogers at intel.com.

Edwin Verplanke is a platform solution architect and is part of Intel's Infrastructure Processor Division. He holds a Bachelor's and Master's degree in Computer Science and Electrical Engineering, respectively. For the past 12 years Edwin has focused on communications board design, participated in various standards development covering high-speed interconnects, and more recently, researching multi-core architectures for the embedded market. His e-mail is edwin.verplanke at intel.com.

For further information visit:

developer.intel.com/technology/itj/index.htm