2010 NDIA GROUND VEHICLE SYSTEMS ENGINEERING AND TECHNOLOGY SYMPOSIUM
VEHICLE ELECTRONICS & ARCHITECTURE (VEA) MINI-SYMPOSIUM
AUGUST 17-19 DEARBORN, MICHIGAN

Software Virtualization in Ground Combat Vehicles

Daniel McDowell
BAE Systems
U.S. Combat Systems
Minneapolis, MN

John Norcross
BAE Systems
U.S. Combat Systems
Minneapolis, MN

ABSTRACT

Increased complexity of ground combat vehicles drives the need to support software applications from a broad variety of sources and seamlessly integrate them into a vehicle system. Traditionally, hosting GOTS and COTS applications required burdening a vehicle's computing infrastructure with dedicated hardware tailored to specific processors and operating systems, heavily impacting the vehicles space, weight and power systems. Alternatively, porting the applications to the vehicles native computer systems is often cost prohibitive and error prone.

Advancements in commercially developed software virtualization capabilities bring an entirely new approach for dealing with this challenge. Virtualization allows real-time and non-real-time applications running on different operating systems to operate seamlessly alongside each other on the same hardware. The benefits include reduced software porting costs and reduced qualification efforts. Additionally, elimination of specialized computing environments can produce significant gains in space, weight, power and cooling demands.

INTRODUCTION

Ground combat vehicles are becoming increasingly more complex. They are transforming from simple personnel and weapons platform carriers to complex machines employing high powered weapons systems containing complex robotic automation; fully integrated maintenance, diagnostics and prognostics systems; integrated networks of command, control and communications systems; and sophisticated user interfaces resembling those more typically found within a fighter cockpit than in a ground vehicle. Ground combat vehicles are now required to support and integrate with an ever increasing spectrum of legacy and evolving systems and applications to include everything from command and control applications to maintenance diagnostics and logistics systems to management and configuration software.

As the complexity of ground combat vehicles continues to grow, the need to support software applications from a broad variety of sources and then seamlessly integrate them into a single vehicle system solution becomes a significant driving design constraint. Add to this the requirement to also host both Commercial off-the-shelf (COTS) and Government off-the-shelf (GOTS) applications, and the task of designing and building a modern combat vehicle quickly becomes an especially daunting undertaking.

The traditional approach used to consolidate and install all of these required applications onto a single vehicle would burden the vehicle's computing infrastructure with dedicated hardware tailored to meet each system's software needs. Such an approach, while straight forward and easy to implement from a software standpoint, would heavily impact the space, weight and power constraints planned for by ground combat vehicle designers. An often used alternative to mitigate this impact would be to consolidate as many applications as possible onto the vehicle's main computer, which would involve costly porting efforts. This would typically result in substandard performance, load balancing issues and numerous other problems. Such software porting efforts would often lead to cost and schedule overruns that would exceed what a project could readily absorb.

Recent advancements in commercially developed software and hardware virtualization technology bring forth an entirely new approach towards dealing with this design and integration challenge. Virtualization technology allows disparate applications running on non-native operating systems to be hosted alongside applications operating in an entirely different environment on the same hardware. Real-time and non real-time applications can now effectively share a single computing resource using such a technology –

something that would have been unheard of as a viable design solution just a few years ago. This means Windows based applications, Linux based applications, and real-time applications can be seamlessly integrated onto a single computing platform and still meet their intended functional requirements within a system.

The benefits of virtualization appear to be many. They include eliminating costly porting efforts required to host a particular application on a designated target environment, to insulating existing software from the impact of migrating to newer hardware, to providing data and process separation which allows for increased fault tolerance within a given design. In addition to the software engineering benefits derived through virtualization, the ability to eliminate the need for numerous specialized computing environments results in a direct and significant impact to a vehicle's design in the areas of space, weight, power and cooling requirements.

WHAT IS VIRTUALIZATION?

So what exactly is virtualization, and how does it help? Virtualization simply refers to the ability to abstract away the physical computing environment and its associated resources (memory, I/O lines, peripherals, etc.) and replace them with a synthetic or virtual copy that can be used in their stead. Virtualization is accomplished via software, or by a combination of software and dedicated hardware support. [1][2]

Software Virtualization

Software virtualization is accomplished via a specialized application called a hypervisor, also known as a Virtual Machine Monitor (VMM). A hypervisor abstracts the computer's hardware (i.e., memory, I/O, disks, etc.) into a virtual platform. The hypervisor then presents that virtual platform to one or more guest (or hosted) operating systems, which can include multiple instances of the same operating system. The hypervisor monitors these guest operating systems, intercepting and redirecting calls from the virtual platform to that of the real hardware it is insulating the operating systems from.

A hypervisor can run on bare hardware (known as a Type 1 or native VMM) or on top of an operating system (known as a Type 2 or hosted VMM). [3][4]

In software only virtualization, the Hypervisor must redirect the hardware calls made by the guest OS. This often required either the modification of the guest OS binaries or by creating a full emulation of the hardware and associated I/O resources the guest operating system has access to.

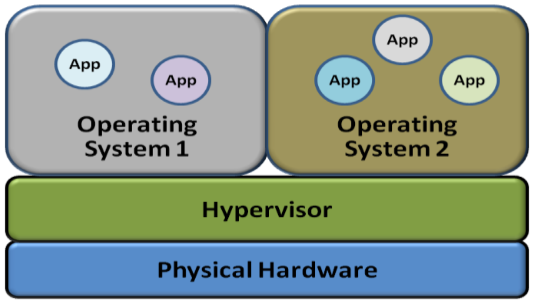
The term *paravirtualization* is typically used to describe a situation in which a guest OS has been specifically modified to run on top of a hypervisor. In such a case, the virtualization environment that embedded applications run on would have been optimized for performance, both within the guest OS as well as within the hypervisor. The use of paravirtualization, which typically also makes use of specialized processing hardware virtualization extensions, offers applications running under its direction near-native execution performance ability. This approach is also more appropriately suited for open source OSes, such as Linux, where the OS source code is readily available and can easily be modified and recompiled for use in such an arrangement.

The term *full virtualization* is used to describe a situation involving a guest OS that has been presented with a virtualized environment that is indistinguishable from the native hardware it would normally interact with. Full virtualization requires no changes to the guest OS, but, because the hypervisor is doing the heavy lifting of translating all calls between the virtual environment and the underlying real hardware, there is typically a performance hit involved over the paravirtualized approach.

Hypervisors typically employ paravirtualization and/or full virtualization in their implementation approaches.

Type 1 Hypervisor

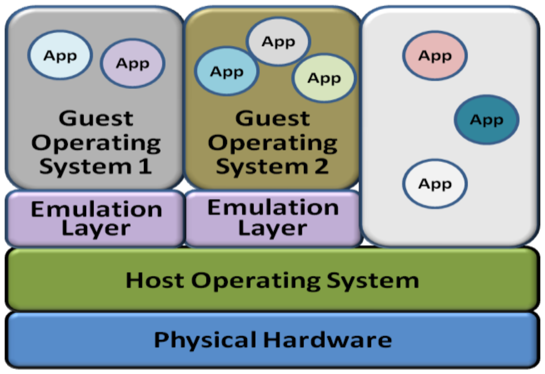
A type 1 hypervisor, also known as native virtualization, relies on a small kernel of specialized software running directly on top of the hardware. This provides a small, efficient, and lightweight virtualization environment on which guest OSes can be run. A type 1 hypervisor typically requires a very close tie to and intimate knowledge of the architecture of the host processor.



Type 1 Hypervisor Architecture

Type 2 Hypervisor

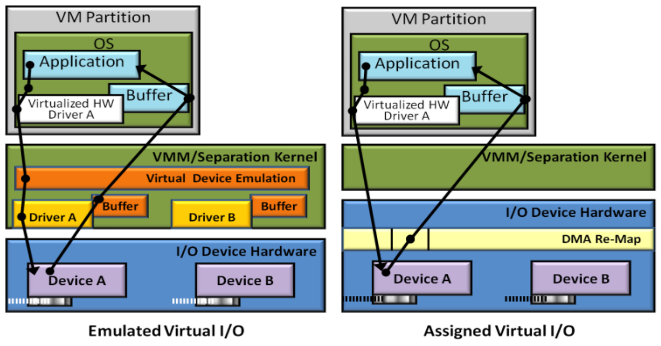
A type 2 hypervisor, also known as hosted virtualization, provides a complete emulation of the hardware platform each hosted OS expects to see. This approach again has the advantage that each guest OS does not need to be modified to run, but has the disadvantage that the emulation overhead required in implementation typically has a measurable impact on performance (i.e., it tends to have a significant impact on processor overhead, causing applications to run slower than they typically would in a native environment). Each OS's call to hardware must be redirected one or more times before it finally reaches the real hardware. This tends to add to overall memory and processor utilization due to the extra software translation required to emulate a given platform.



Type 2 Hypervisor Architecture

Hardware Assisted Virtualization

Hardware chipsets made by manufactures like Intel® (Intel Virtualization Technology or Intel VT) and AMD (AMD V^{TM}) can assist in the virtualization of a hardware platform by supporting extensions that make it easier for the hypervisor to abstract the hardware. Instead of having to patch the binaries of the hosted operating system to redirect hardware calls, or requiring that they run in a protected mode, hardware assisted virtualization allows the virtualized operating system / software to run in its intended user mode, while redirecting calls to hardware based on a flag set by the hypervisor for the guest OS. This is accomplished by having the hypervisor run with its flag set for direct hardware access and the guest operating systems set with their flags set to recognize virtual hardware access. This speeds up hardware calls, by properly routing them to either the real hardware or to the hypervisor, depending on who is making the call.



Emulated VS Hardware Assisted Virtualization [5]

Processors and OSes supporting hardware virtualization technology makes it much simpler for a hypervisor to virtualize a given platform, avoiding the need to modify a guest OS and avoid the time impacts usually associated with redirected I/O and hardware calls from a virtualized OS.

Separation Kernel

A separation kernel is a kernel designed to provide time-space partitioning and information flow control. The job of a separation kernel is to control the flow of information between hosted applications. It does this by explicitly controlling the communication channels and resources available to the applications being hosted on the kernel.

With security critical applications, it can be important that one component be prevented from accessing the resources available to another component. Historically this has been accomplished by separate computers, or through the use of split backplanes, where resources are physically separated and no unintended cross communication or data flow exists. Data flow external to a component is typically managed via the use of a separate Cross-Domain Solution specific hardware item. This combination tends to increase the space, weight, cost, complexity and power required in a given design while reducing its flexibility. The use of a separation kernel is a design solution that accomplishes process and data separation through the use of software, rather than hardware.

The concept of a separation kernel was originally introduced by John Rushby [6]. According to Rushby, "the task of a separation kernel is to create an environment which is indistinguishable from that provided by a physically distributed system: it must appear as if each regime is a separate, isolated machine and that information can only flow from one machine to another along known external communication lines. One of the properties we must prove of a separation kernel, therefore, is that there are no channels for information flow between regimes other than those explicitly provided."

In 2007, the Information Assurance Directorate of the U.S. National Security Agency published the Separation Kernel Protection Profile (SKPP). In this document, it states that: "The separation kernel provides to its hosted software programs high-assurance partitioning and information flow control properties that are both tamperproof and non-bypassable. These capabilities provide a configurable trusted foundation for a variety of system architectures."[7]

HOW DOES THIS ALL APPLY TO GROUND COMBAT VEHICLES?

To understand how virtualization technology applies to ground combat vehicles, we should first take a look at how it applies to the commercial information technology (IT) sector. Virtualization has been successfully applied to servers in a business or IT environment for many years. Many IT servers in large companies are using virtualization software such as VMware™ to minimize the number of servers required by their organization. The application of this technology has also allowed divergent OS applications (Windows, Linux, Solaris, etc.) to be run on a single piece of hardware (or even an abstracted group of hardware). It allows the applications, including license servers, drive storage, etc. to be seamlessly moved from server to server (as upgrades are necessary for memory, processing, disk space) without impacting the end user. The users or applications are presented with a single platform that matches their needs. This allows the IT community to support numerous COTS and Open source applications using a minimal number of optimized hardware platforms.

Why Would Virtualization be Useful for Ground Combat Vehicles?

As stated in the beginning of this paper, ground combat vehicles are becoming increasingly complex and are now required to interface to and/or integrate with a large variety of technically diverse Commercial off-the-shelf (COTS), Government off-the-shelf (GOTS) and Government Furnished Equipment (GFE) products. This typically requires dramatic increases to a vehicle's weight profile to offset the space and power requirements imposed by the additional required computing infrastructure needed. In addition to weight and space claim issues, this brings along additional challenges associated with how to configure vehicle crew compartments to allow for access to all the equipment while still allowing room for the activities associated with simply operating the vehicle. Historically, integrating these applications and their dependent computing platforms often meant one of the following, all of which could introduce unacceptable risks to a program:

- **Port the software to the vehicle's computing system**
  This option would involve a porting effort onto the vehicle's primary computer system. The software applications would be modified without impacting their internal design or structure such that they could run in an entirely different computing environment then the one they were originally designed to operate in. This often involves changing the target operating system, the processing unit, middleware, support libraries, and/or interfaces. Such an undertaking typically requires gaining access to the source code of the application (which is not traditionally available to a vehicle integrator), or paying the vendor/developer of the application to modify them to work within the new environment. Depending on the changes required, this often can result in an outlay of resources (money or schedule) that exceed those that would have been spent simply rewriting the application from scratch. Such a porting effort also brings along inherent risks associated with issues that may develop by the introduction of the new computing environment that the original application never had to contend with.

- **Rewrite the software to work on the vehicle's hardware**
  Another option is to rewrite the application itself, re-implementing its internal design in such a fashion as to have it natively supported by the vehicle computing hardware. Rewriting an application is also a costly endeavor, both monetarily and in schedule. Unlike a simple porting effort, such redesign is typically expense and invasive, requiring in-depth testing of the application to verify that all of its intended functionality remains correctly implemented. Again, this brings along possible risks associated with issues caused by the introduction of the new environment as well as those errors introduced during the applications redesign.

- **Integrate the native hardware and software**
  This option is the simplest from a software development and integration perspective, but while it typically eliminates many software schedule concerns, it simply transfers the problem, transforming it into burdens for other engineering domains. In this approach, the required target hardware and software are simply brought into the vehicle and incorporated into its overall design. But, this approach brings along several challenges, the first being able to find adequate space for all the hardware, which not only means the computing hardware itself, but also its associated peripherals and user interfaces such as displays, keypads or other user input devices. Even adding something as seemingly innocuous as a simple laptop computer involves finding a space to store it, a location to put it while it is in use, a method to power it, some special packaging or ruggedization to incorporate, and some way to connect it to its required inputs and outputs (i.e., Ethernet, radios, intercom, etc.). The second challenge is providing the power required for all the added equipment, which places additional burdens upon the vehicle's power resources. Such burdens typically also negatively impact cabling requirements as well as power distribution elements of the system. Obviously, this overall approach can also dramatically impact vehicle weight, as each integrated hardware and software package adds to the bottom line. In addition to these primary challenges, there are numerous other indirect impacts to other areas such as thermal environment management, RFI/EMI, human factors, etc. Finally, each hardware/software system incorporated into the

vehicle is most likely provisioned with requirements that mandate it cover scenarios in which the worst case maximum processing, memory, and power utilization profiles required by all its applications are met, even though such conditions are rarely realized in an operating environment in which the systems are being used concurrently.

All of these options bring with them potentially significant risks which vary from impacts to the cost, weight and size of the vehicle to the introduction of significant schedule impacts in the vehicle's development.

Virtualization: Bringing in an Entirely New Option

The maturation of Hardware and Software Virtualization technologies in recent years brings to the table an entirely new option that can be explored. Instead of adding boxes, porting applications, or rewriting applications compatible with the existing COTS/GOTS products, the vehicle integrator can utilize virtualization technologies to reuse the applications as written and leverage a single computing infrastructure on which to support them all. A hypervisor is used to provide a virtual platform that simulates the native platform required by an application. This allows the application (and the OS it was designed for) to run on the ground combat vehicle's computer system. Using a hypervisor, the application can be integrated into the vehicle without modification to the application or the OS. This includes Windows applications, Linux applications, and even real-time applications. This approach has several advantages to its use which include:

- **Reduced cabling**
  Reducing the number of boxes has the potential to significantly reduce cabling, both to each individual box, and potentially between boxes if they were networked or interconnected.

- **Reduced Power requirements**
  The power required to run a single computing platform is typically less than that required to power multiple non-heterogeneous computing platforms.

- **Improved crew station layout**
  Virtualization technologies, coupled with remote control capabilities allow the vehicle integrator to optimize the computing platform's displays and user interfaces to better match the vehicle's mission. The hypervisor and remote control capabilities allow access to multiple applications off the same monitor, rather than requiring multiple monitors, keypads, etc.

- **Reduced integration and test time**
  The use of a hypervisor allows COTS and GOTS products to be integrated along with their native environment into the system. This has the potential to greatly reduce integration and test time, by allowing the vehicle integrator to reuse the application directly without having to worry about operating environment compatibility.

CHALLENGES TO ADOPTION

Although virtualization sounds like the answer to many of the issues associated with reusing and integrating COTS / GOTS applications within ground combat vehicles, there are challenges to adopting the technology that need to be addressed.

Information Assurance Concerns

In a ground combat vehicle, information is a critical resource. The use of virtualization technology brings with it concerns about managing data to including its availability, integrity, authentication, confidentially, and non-repudiation (i.e., ensuring validity in the verification of signatures). Information assurance in a ground combat vehicle also involves ensuring the vehicle is protecting its data from external attack via open ports (Ethernet, USB, etc.), Trojans and viruses, and any other form of unauthorized access.

Virtualization must also address data separation concerns, ensuring that it is maintained between applications such that the data associated with one application can't be accessed or acted upon by another either intentionally or unintentionally.

Virtualization also brings along issues associated with virtual networks, such as ensuring that information on a virtual network inside a processor isn't making the system vulnerable to attack. Communication connections, such as Ethernet, USB, eSATA, CANbus, Firewire, etc. that were previously physically separate and secure in their native environments may now be virtualized across the platform. This can lead to difficulties in the management of granting access to the network and in ensuring compliance with established information assurance guidelines.

The Red Hat Enterprise Linux 5 OS includes Security-Enhanced Linux (*SELinux*), which is a framework developed jointly with the National Security Agency (NSA) and the Linux community that enforces limits on files that programs can access and the actions that programs can take.

As described previously, the separation kernel approach being implemented in the new generation of hypervisors from companies like Green Hills, Wind River, and LynuxWorks helps address many of the existing information assurance issues that are associated with data separation by limiting the access to and controlling the flow of data within the hypervisor.

Software Updates / Maintenance

Virtualization brings along unique challenges in the area of application installation and application update as well. Since the applications are running within a virtualized platform (referred to as a domain or container), rather than directly on the native hardware and OS, the applications installer provided by vendors typically are not designed to handle the virtualized environment. Even if a CD-ROM drive is available in the platform (or on a maintenance laptop), the maintainer can't simply take a new release CD from the supplier and run the install application to simply update or reinstall it. Instead, a new method must be developed to deploy or update applications within the vehicle.

Additionally, since ground combat vehicles typically have limited processing power to begin with due to imposed space, power and weight restrictions, changes to an application's memory, processing, and/or storage requirements may require reworking the memory and space allocations across the entire vehicle's computing assets.

Parallel Computing Needs

Space, weight, power and environmental considerations all conspire to force the computing platform utilized within a ground combat vehicle to be as optimized a solution as possible. Adding applications or increasing the size of existing applications running within such a computing environment will typically impact areas such as process prioritization and load balancing in order to maintain the performance required by the system's specifications. A hypervisor provides access to a set limit of the computer's actual physical resources. Changing how many of these resources a hypervisor can make available to an application can impact the performance of the installed application.

Additionally, the paradigm behind traditional virtualization is to make full use of all available resources. Typically this means that all of the resources are distributed between the applications virtualized on the platform. If multiple applications are executing at the same time, the performance of one application could suffer based on the demands of another. This is often the case as the available processing power (or other resources) available in the virtualized system may not be equivalent to the combined amount available in the standalone systems the applications were originally designed for. When virtualization technology is leveraged in unique solutions that also address security and safety concerns, the availability of multicore resources are often usurped, limiting their specific application and use at the OS and user application level. While moving from single core to multicore hardware is often seen as a performance enhancer at the OS level and above in a system's design, its impact is far more beneficial in terms of design flexibility and capability with the introduction and use of virtualization technologies.

Real-Time constraints

Embedded systems typically found in ground combat vehicles usually require hard real-time response to support combat operations. Typical commercial hypervisors, designed for workstations and IT servers include additional overhead to virtualize the operating environment and as such don't address such real-time response requirements.

Recent advancements in embedded hypervisors, such as LynxSecure and VxWorks (MILS) support legacy Real-Time Operating Systems (RTOS) such as LynxOS and VxWorks to support hard real-time response requirements while at the same time supporting the use of High Level OSes, such as Windows and Linux on a given hardware platform. Embedded hypervisors typically make use of paravirtualization to minimize impacts on the RTOS, and to speed up execution of Linux domains, as well as supporting full virtualization for Windows applications.

These embedded hypervisors are a relatively new technology on the market. Their ability to successfully fulfill all of the timing and memory separation requirements of a real-time operating system while extending the virtualization of a given platform to include hosting Windows and Linux OSes and their respective applications is still being evaluated, but looks very promising.

A SURVEY OF CURRENT VIRTUALIZATION PRODUCTS AVAILABLE

The addition of hardware assisted virtualization in modern processors (i.e. Intel VT-x and AMD-V) has brought a dramatic increase in the number of Software Virtualization options available. Below is a sampling of several virtualization products that seem to be viable choices for possible incorporation into a ground combat vehicle's computing infrastructure design.

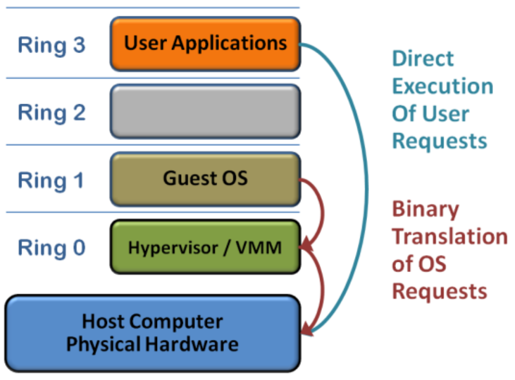
VMware

VMware is one of the most widely recognized virtualization (i.e., hypervisors or Virtual Machine Monitors) tools in the world today. VMware was founded in 1998, and has been producing VMware applications to support virtualization primarily targeted towards the Information Technology (IT) community for the past 12 years. VMware was one of the first companies to figure out how to effectively virtualize the x86 platform, creating a market for x86 virtualization. [8] VMware efforts primarily focus on the business community's IT sector by providing virtualization capabilities for both servers and desktops. They provide virtualization solutions that allow IT departments to make more efficient use of machines by deploying multiple "Virtual Machines" on single hardware platforms. This significantly consolidated expenditures for hardware and allows for a more flexibility approach in the deployment of applications across the available hardware resources within a business.

VMware provides both type 1 and type 2 hypervisor solutions. VMware's desktop software products (including VMware Workstation, VMware Fusion, and VMware Player) typically run as type 2 hypervisors (i.e. hosted applications) on Microsoft Windows, Linux, and Mac OS X. VMware's server software products (including VMware ESX, VMware Server, and VMware vSphere) typically run as type 1 hypervisors directly on server hardware without requiring an additional underlying operating system. [9]

VMware primarily supports full virtualization, where the guest OS runs without modification in the VMware virtual machine and is provided a virtualized set of hardware to utilize. VMware software virtualizes the hardware typically found on a system, such as network and video adapters, hard disks, USB ports, etc. This allows VMware virtual machines to become highly portable between computers, because the host looks identical to the guest. In practice, this allows an IT system administrator to suspend operations on a virtual machine guest, move or copy that virtual machine to another physical computer, and then resume execution exactly at the point of suspension.

What is somewhat unique about the VMware solution is that it does not emulate an instruction set for the hardware that is not physically present. Instead, VMware uses a technique called Binary Translation (BT) to dynamically rewrite the non-virtualizable instructions into new sequences of instructions that have the same intended effect on the virtual hardware [8]. The translated code gets stored in spare memory, typically at the end of the address space, which segmentation mechanisms can protect and make invisible. This binary translation, combined with direct execution of hardware requests can significantly boost performance, but can also, in theory, cause problems when moving virtual machine guests between hardware hosts that employ different instruction-sets.


Ring 3
User Applications
Ring 2
Ring 1
Guest OS
Ring 0
Hypervisor / VMM
Host Computer Physical Hardware
Direct Execution Of User Requests
Binary Translation of OS Requests

VMware's Binary Translation Approach

VMware does not currently address the flow control aspects that a separation kernel would address. VMware, at least in its current form, does not support the Multiple Independent Levels of Security (MILS) architecture which allows applications with different levels of security to be deployed on the same platform securely.

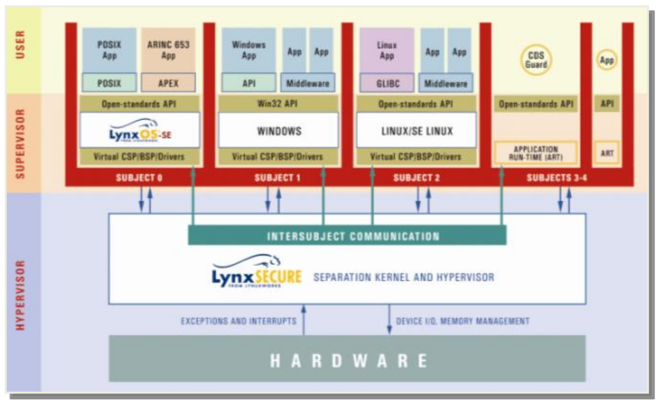
LynuxWorks LynxSecure

LynuxWorks offers a product called LynxSecure, which is an embedded virtualization solution for single and multicore processors that makes use of a high-assurance separation kernel and an embedded hypervisor. LynxSecure version 4.0, which was released in May of 2010, is referred to by LynuxWorks as a Separation Kernel Hypervisor (SKH). [10] It was developed based on the need to address both separation of virtual machines via the hypervisor (supporting running multiple operating systems on the same hardware), and flow control via the separation kernel (managing the resources and flow of information between components in the system). The hypervisor functionality maintains an abstraction layer between the resources of the target and the hosted or virtualized operating system. The separation kernel functionality permits the hardware resources of the platform to be securely shared as the resources that it exports and partitions.

LynxSecure currently supports virtualization of Windows operating systems (such as Windows XP), Linux Operating Systems (such as Red Hat Enterprise Linux 5.x), and Real-time Operating Systems (Such as LynxOS SE). This allows applications developed for these three operating systems to run on LynxSecure concurrently in separate partitions, while maintaining 100% application binary compatibility with the non-virtualized operating systems. LynxSecure also supports the use of multiple instances of the same operating system, each running within its own partition. LynxSecure provides virtual memory, virtual addressing, and time space partitioning, to guarantee resource availability.

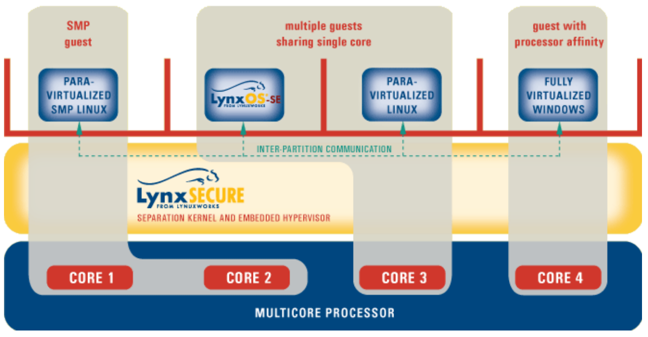
Additionally, since LynxSecure is a separation kernel, it addresses concerns associated with security critical applications, especially those that handle multiple enclaves or different levels of information (such as Confidential, Secret, and Top Secret).

LynxSecure claims conformance to the Multiple Independent Levels of Security/Safety (MILS) architecture, claiming strict adherence to data isolation, damage limitation and information flow policies identified by this architecture. LynxSecure allows multiple components to run concurrently on the same hardware through the hypervisor, and uses the separation kernel to enforce isolation and information flow control through the configuration of the kernel. This capability, combined with a software based Cross Domain Solution (running on top of the kernel) allows data flow to be managed at the item level. This is a desired feature in military applications, where information assurance/security is critical. LynxSecure supports the capability to be certified for EAL 6+ (in theory it was designed to be certifiable to EAL 7).



LynxSecure's Separation Kernel and Hypervisor [10]

As discussed earlier, virtualization can have a significant impact on performance and LynxSecure provides both paravirtualization and full virtualization mechanisms that are intended to be leveraged to help minimize negative performance impact. Paravirtualization uses modified components of the given operating system/environment to run more efficiently on LynxSecure. Full virtualization makes it possible to run an operating system such as Windows XP where the source code is not available for paravirtualization or when it is desired to run the guest operating system and its application software in a completely unmodified environment. Additionally, LynxSecure supports hardware assisted virtualization which allows additional improvements in performance by making use of a processor's hardware acceleration / virtualization capabilities. Currently LynxSecure makes use of virtualization capabilities in both Intel and PowerPC processors. Most recently, LynxSecure added support for Intel Core i7 processors which allows for near native performance in a virtualized environment. This is critical when running an RTOS, such as LynxOS SE on LynxSecure, where timing and performance are critical.


SMP guest
multiple guests sharing single core
guest with processor affinity
PARA-VIRTUALIZED SMP LINUX
LynxOS-SE
PARA-VIRTUALIZED LINUX
FULLY VIRTUALIZED WINDOWS
INTER-PARTITION COMMUNICATION
LynxSECURE
SEPARATION KERNEL AND EMBEDDED HYPERVISOR
CORE 1
CORE 2
CORE 3
CORE 4
MULTICORE PROCESSOR

LynxSecure Virtualization for High-Assurance Embedded Systems on Multi-core Hardware

LynxSecure also supports Symmetric Multi Processing (SMP), and runs as a multi-core Separation Kernel Hypervisor. The cores utilized by LynxSecure are configurable via XML and components can also be configured via XML to have direct access to a given set of devices and host controllers. Fully virtualized Windows XP environments for example can be installed to run directly from an assigned hard drive device or from a virtual block device.

Red Hat Enterprise Linux Virtualization

Red Hat Enterprise Linux provides a virtualization system known as the Red Hat Enterprise Virtualization (RHEV) which is multi-layered and driven by a privileged hypervisor. RHEV is capable of hosting multiple guest operating systems where each guest OS runs within its own Virtual Machine. RHEV schedules virtual CPUs (called vCPUs) within each virtual machine to make the best use of the available physical CPUs in a system, with each guest operating system handling its own applications and scheduling its applications accordingly on the virtual CPUs that it sees.

Red Hat Virtualization can be deployed using either full virtualization or paravirtualization. With full virtualization, total abstraction of the underlying physical system is achieved and a new virtual system (a virtual machine or VM) is provided in its place in which the guest operating systems resides and runs. Using full virtualization, no modification to the guest OS (nor the applications it runs) are required since it is not aware that it is running in a virtualized environment.

The use of paravirtualization allows the utilization of high performance virtualization on architectures that are potentially difficult to virtualize, such as those based on x86 hardware, and requires modification of the guest operating system so that it is aware that it is running within a virtual environment. Near-native performance can be realized by taking this approach, and while an operating system's kernel must be configured to support this form of virtualization, it is not necessary to modify individual user applications or libraries. Paravirtualization and full virtualization can be deployed independently or simultaneously across the virtualization infrastructure that Red Hat Enterprise Linux provides.

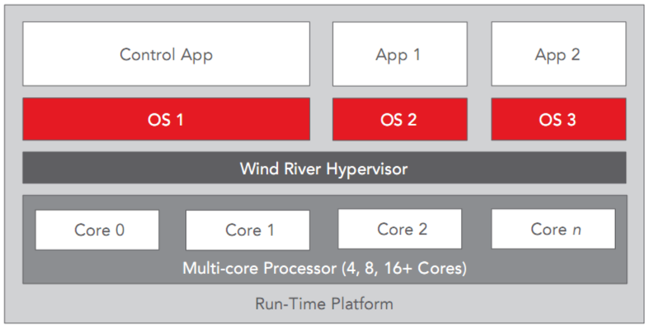
Current Red Hat Enterprise Virtualization (RHEV) is based on a RHEL kernel implementing a Kernel-based Virtual Machine (KVM) and can be thought of as a hybridized type 1 hypervisor. It is approximately 100MB in size and is bootable from a Preboot eXecution Environment (PXE), flash memory, local disk or Storage Area Network (SAN). It supports up to 96 processing cores and 1TB of RAM, with VMs containing up to 16 vCPUs and 256GB of RAM. RHEV utilizes high-performance virtual input/output drivers and PCI-pass through direct I/O to allow it to achieve up to 98% of the performance of a physical (bare metal) solution.

RHEV currently supports guest operating systems from RHEL3 through RHEL5, as well as Windows 2000, Vista and Server 2008. It is expected that future releases will also incorporate support for Windows 7 and Server 2008 R2 as well. RHEV is an x64 only solution and makes extensive use of hardware assisted virtualization found in P6 class processors, with directed I/O (Intel VT-d/AMD IOMMU) used for secure PCI pass-through together with PCI single root I/O virtualization so that multiple virtual operating systems can achieve near native I/O performance for network and block devices.

While Red Hat seems to be gearing its future virtualization efforts towards the adoption and use of a KVM approach, past Red Hat virtualization technology successfully employed the use of a Xen hypervisor, which makes use of paravirtualization and can also leverage Hardware-Assisted Virtualization (HVM) on more modern processors.

Wind River Hypervisor

Wind River introduced its own brand of virtualization technology to the market in June 2009 via its release of an embedded hypervisor that provides a virtualization layer that can partition single or multi-core chips into multiple partitions, each with varying levels of protection and capabilities. Wind River's products are traditionally associated with embedded systems and RTOS offerings, so their entry into the realm of virtualization further endorses its viability as a technology. Unlike many other hypervisors, Wind River's solution was developed with the demands of real-time systems foremost in mind. Recognizing the industry trend towards the adoption of multi-core hardware, their hypervisor offering is specifically targeted at clients in the defense and aerospace industry. Their solution is focused on being small, scalable, deterministic and event-driven, all of which are attributes actively sought after within the embedded systems market. Wind River's solution is also a type 1 hypervisor that runs directly on the target hardware. Like other vendor solutions, their hypervisor separates a system into multiple virtual machines or virtual boards as they are referred to within Wind River literature.

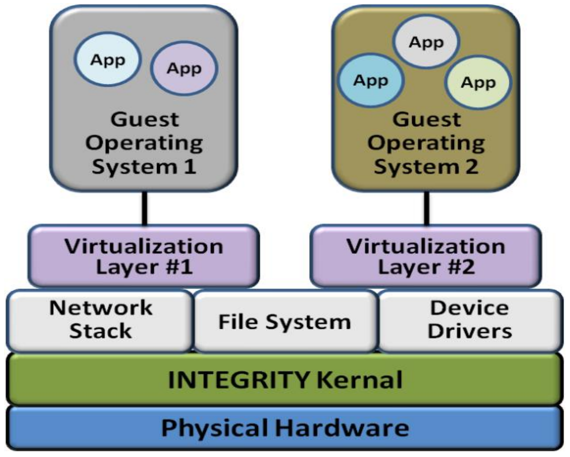


Wind River Hypervisor [11]

Wind River's hypervisor executes guest operating systems, which in turn host applications. The guest operating systems can execute at near native performance levels and like other hypervisor solutions, the Wind River hypervisor uses a mix of hardware assist and paravirtualization in order to deliver optimized performance and determinism. Currently, the Wind River Hypervisor supports Wind River Linux and VxWorks, and it provides an open interface that allows other operating systems and executives to run, to include those that are open-source based, proprietary, or internally developed. The Wind River hypervisor also allows for the execution of a virtual machine without an operating system. This is useful for specialized tasks that do not require the capabilities of a full operating system, such a tasks utilizing a fast polling loop to monitor hardware. This "no operating system" capability, known as a virtual board application, allows the Wind River hypervisor solution to implement a fast-boot scenario which is another often sought after feature in the embedded domain space.

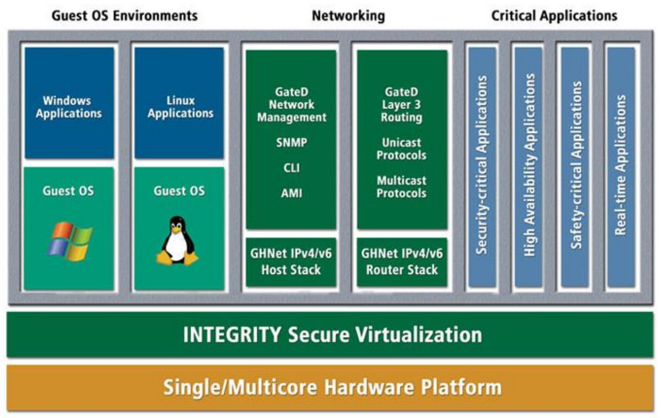
Green Hills INTEGRITY

Green Hills INTEGRITY is an RTOS employing virtualization technology that has evolved significantly since it was first introduced in 1997. INTEGRITY employs a microkernel-based hypervisor architecture where virtualization takes place in user-mode processes that are outside the trusted operating system microkernel. Separate instances of the virtualization layer are used for each guest operating system that is hosted. In this approach, the virtualization layer only needs to meet the equivalent robustness level required by the guest operating system. The kernel runs in the highest privilege mode, with device drivers, network stacks, and file systems all executing as non-privileged applications. INTEGRITY also makes use of hardware memory protection to isolate and protect itself and user tasks from incorrect operation caused by accidental errors or malicious tampering.



INTEGRITY's Enhanced Type-1 Hypervisor Architecture

In 2003, Green Hills deployed INTEGRITY Secure Virtualization (ISV), a product based on an enhanced version of a type-1 hypervisor architecture. INTEGRITY Secure Virtualization allows a platform to host arbitrary guest operating systems alongside real-time applications and middleware. Applications and guest operating systems are scheduled across one or more processor cores, can communicate with each other, and utilize system peripherals according to an access control model implemented by the hypervisor. [12]



INTEGRITY Secure Virtualization

Green Hills also offers a product known as INTEGRITY-178B, which is a certifiable version of INTEGRITY for Safety-Critical applications. INTEGRITY-178B has distinguished itself as the first operating system certified by the NSA-managed NIAP lab to achieve an EAL6+ High Robustness rating. No other commercial operating system has attained this level of security.

INTEGRITY Multivisor is Green Hills Software's ISV implementation for multicore processors. Besides provides mechanisms for managing discrete cores, the Multivisor can statically bind guest operating systems to cores, in an Asymmetric Multiprocessing (AMP) model, or dynamically schedule workloads in a Symmetric Multiprocessing (SMP) model, depending on system requirements.

INTEGRITY supports all common general purpose operating systems, such as Windows, Linux, Solaris, and Android and runs on a wide range of hardware platforms to include variants of ARM, XScale, Blackfin, Freescale ColdFire, MIPS, PowerPC, and x86 computer architectures.

CONCLUSION

Virtualization has been used for years in the commercial world to reduce server space and increase flexibility, and is widely accepted as common place in the IT world. Recent advances in Virtualization and Separation technologies provide an exciting and new way to support the adoption and seamless integration of COTS and GOTS products alongside legacy real-time applications into ground combat vehicles. Their use promises significant space, power and weight reductions while allowing true software reuse at the binary level. While there are some risks and obstacles associated with their adoption within a ground combat vehicle's design, the benefits derived help marginalize many of the risks associated with their use and bolster their serious consideration as an enabling technology.

REFERENCES

[1] Robert Day "Virtualization makes better use of Open-source OSes and apps". EE-Times 3/23/2009 http://www.eetimes.com/news/design/showArticle.jhtml?articleID=215901013

[2] Steve Blackman "Secure virtualization technology can extend the life of legacy systems". Military Embedded Systems, January February 2009. http://www.mil-embedded.com/articles/id/?3738

[3] Goldberg, Robert P. (February 1973). Architectural Principles for Virtual Computer Systems. Harvard University. pp. 22–26. http://www.dtic.mil/cgi-bin/GetTRDoc?AD=AD772809&Location=U2&doc=GetTRDoc.pdf

[4] Wikipedia – Hypervisor – http://en.wikipedia.org/wiki/Hypervisor#cite_note-0

[5] Peter Carlston "Advances in virtualization aid information assurance". Embedded Computing Design, January 13th, 2008. http://embedded-computing.com/advances-virtualization-aid-information-assurance

[6] John Rushby, "The Design and Verification of Secure Systems," Eighth ACM Symposium on Operating System Principles, pp. 12-21, Asilomar, CA, December 1981. (ACM Operating Systems Review, Vol. 15, No. 5).
[7] Information Assurance Directorate, National Security Agency, Fort George G. Meade, MD. "U.S. Government Protection Profile for Separation Kernels in Environments Requiring High Robustness," Version 1.03, June 2007.
[8] VMware "Understanding Full Virtualization, Paravirtualization, and Hardware Assist" http://www.vmware.com/files/pdf/VMware_paravirtualization.pdf
[9] From VMWare corporate web site. Products overview - http://www.vmware.com/products
[10] From LynxWorks corporate web site. http://www.lynuxworks.com/virtualization/hypervisor.php
[11] From Wind River corporate web site. http://www.windriver.com/products/hypervisor/index.html
[12] From Green Hills corporate web site. http://www.ghs.com/products/rtos/integrity_virtualization.html