Cross-OS HYPERVISOR™

A Next Generation Hypervisor for the Embedded Market

Whitepaper

Table of Contents

Background: Current Generation of Hypervisors

Embedded developers turn to hypervisor solutions in order to extend the life cycle of a product using aging hardware or an obsolete OS. Further, they use a hypervisor solution to transition to newer computing platforms, while co-hosting their legacy systems in parallel on separate partitions. However, they are not always aware of the future hassles associated with continuing product development once the transition is completed. Selecting the right type of hypervisor solution will not only support this transition, but also will streamline and simplify future product development while enabling effective application-level interoperability and software re-use.

A hypervisor (also called a Virtual Machine Manager) is a software, firmware or hardware component that creates and runs virtual machines. Virtual machines are the virtual representation of a hardware platform with an OS. A hypervisor provides a software execution environment, which is separated from the underlying physical hardware resources.

The hypervisor divvies up the underlying hardware into partitions each containing dedicated CPUs, memory and devices and installs one OS in each partition. The applications run on the operating systems that are installed in these partitions. Typical hypervisors are classified as Type 1 and Type 2.

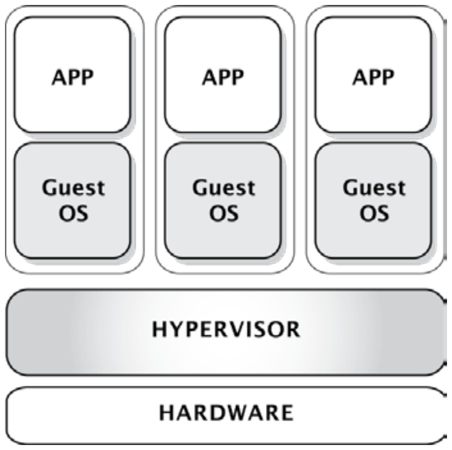

Fig 1.1: Type 1 Hypervisor

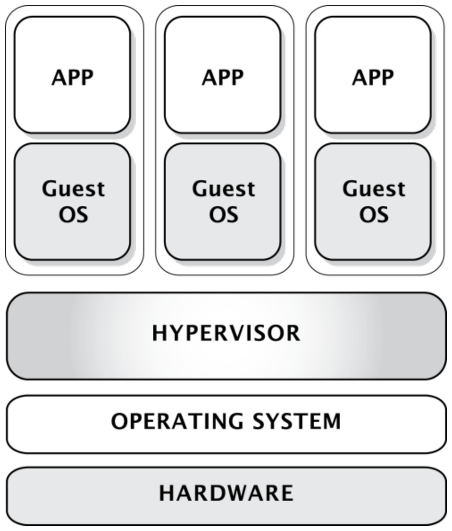

Fig 1.2: Type 2 Hypervisor

Type 1 Hypervisors run directly on host hardware and manage guest operating systems in different partitions as shown in Fig 1.1 above. In this approach, virtualization software is needed to provide services like BSPs and device driver calls for each OS they host. This type of hypervisor is generally not preferred in the embedded domain.

Type 2 Hypervisors run within a host operating system environment as shown in Fig. 1.2 above. Here, the hypervisor layer is a distinct second software level and the guest operating systems run at the third level above the hardware. In this approach, the virtualization software relies on the host operating system to provide the services to talk directly to the underlying hardware. Type 2 hypervisors are suitable for the embedded domain.

Limitations of Current Generation Hypervisors

Typical hypervisor solutions effectively address consolidation of hardware platforms, but do not consolidate the OS platforms. This leaves users with a lot of constraints when choosing a typical hypervisor solution:

- **Hypervisor Availability** - Embedded products use a wide variety of CPU architectures and models ranging from low-end to high-end in terms of computer architecture, power, cost, specific use, etc. However, typical hypervisor solutions are not available for a majority of embedded CPUs and users may be forced to choose more expensive hardware.

- **Hardware Expenses** - Even if your existing CPU is powerful enough to run multiple applications, you may not find a hypervisor that supports it and therefore may be forced to purchase new hardware.

- **Project Costs** - Hypervisors minimize the cost of hardware by consolidating them to a single, cheaper hardware platform. However, they will leave you having to pay high licensing and possibly royalty fees for your software platforms. While helping you to change to less expensive hardware, hypervisors don't help you reduce your software expenses (e.g. software, tools, device drivers, middleware, etc.).

- **Working with an Obsolete OS** –Hypervisor solutions may not be available for your obsolete OS or versions (e.g. there is no embedded hypervisor available for the pSOS® operating system).

- **Device Management** - Hypervisors need a mechanism to decide which application will have access to the shared physical devices, when to grant access and also what the status of the device needs to be. Creating such a mechanism that ensures a decision is made in a timely manner to satisfy the needs of real-time embedded systems can be difficult.

- **Performance** - Hypervisors introduce an additional layer of software that makes the environment more complex due to various integration and communication issues and further impacts the application's performance and real-time requirements.

Next Generation Hypervisor: Cross-OS Hypervisor

The Cross-OS Hypervisor is similar to a Type 2 Hypervisor in that it needs a host operating system. However, the hypervisor itself is capable of offering the services of various guest OS to support hosting multiple heterogeneous applications. The Cross-OS Hypervisor (ref. Fig 2.1) provides multiple OS interfaces, thereby eliminating the need for multiple guest OS's. Hence the hypervisor and guest operating systems are on the same layer at the second level above the hardware, while the host operating system provides the service calls to talk to the underlying hardware.

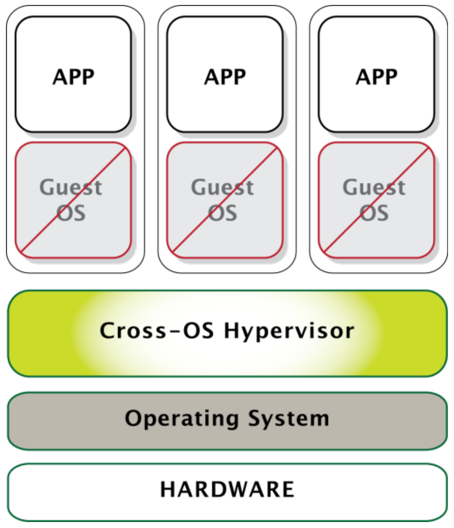

Fig 2.1: Cross-OS Hypervisor Design

Cross-OS Hypervisor is redefining virtualization from wrapper and layered based implementations to a virtualization that is accomplished at the application source code level. Improved performance is achieved as a result of eliminating multiple layers of OS schedulers and device I/O.

Cross-OS Hypervisor is a next-generation hypervisor that eliminates the need for multiple operating systems by providing the OS functionalities required by the applications. By consolidating or confining your development to a single hardware and OS, the current generator hypervisor limitations are virtually eliminated.

Cross-OS Hypervisor design ensures that applications are not locked into a particular OS platform and it simplifies development, reduces bill-of-material costs, and utilizes the system resources more effectively.

Cross-OS Hypervisor provides virtualization interfaces for Linux POSIX/POSIX, micro-ITRON, Windows®, VxWorks®, Nucleus®, ThreadX®, pSOS®,µC/OS™ and FreeRTOS™ applications as show in fig 2.2 below:

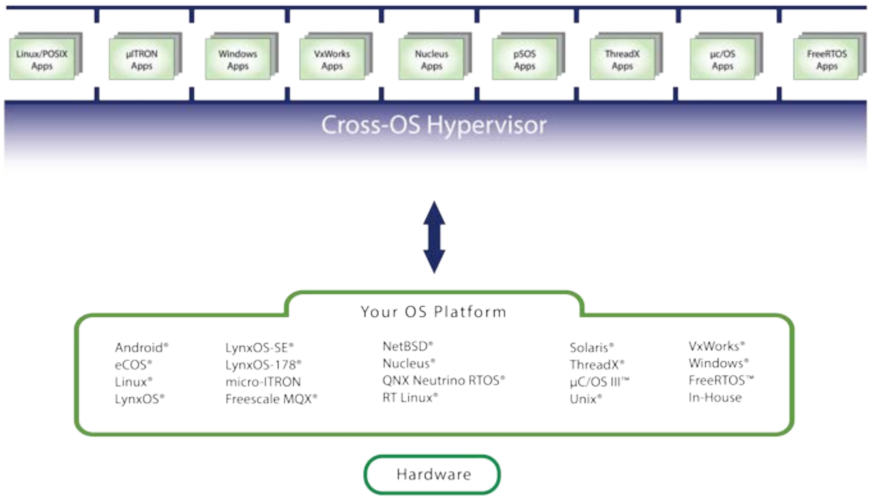

Fig 2.2: Cross-OS Hypervisor Overview

In the subsequent sections, you will find that MapuSoft's innovative Cross-OS Hypervisor solution offers hardware and OS consolidation by consolidating multiple OS platforms to a single OS, without having to go through the effort of re-writing your software.

Cross-OS Hypervisor Benefits

There are many reasons for consolidating applications to a single OS. The benefits include:

- **Legacy code migration**: Cross-OS Hypervisor allows you to run unmodified legacy code, even on your existing hardware platform, by eliminating the application's dependency on the legacy OS (e.g. pSOS). This provides time and money savings associated with the new hardware design. Optionally, it also allows for switching to a newer hardware platform, while greatly reducing the actual migration effort.

- **Advanced OS features:** Cross-OS Hypervisor provides advanced and mission-critical OS features irrespective of whether the underlying host OS offers them or not. It also enhances real-time performance with extended OS features like soft reset, MMU, Tired Memory Pool, Task Pooling and more. Developers can also add advanced platform features and extend the hypervisor functionality for their platform.

- **Simpler development**: Consolidating to a single OS allows for a simpler, more streamlined development on one OS by using one set of tools/device drivers/BSPs. It also makes debugging easier, since you will be using the native debugger tools and do not have to debug multiple layers of software.

- **Cost savings:** Cost savings result from reduced software, hardware and development costs. Software licensing costs are reduced for un-needed operating systems, tools and middleware. Hardware costs are reduced for memory and disk space. And finally, development costs are reduced for the BSPs/device drivers as you only need to develop drivers for one OS.

- **Integration of applications**: Applications can allow sharing of certain individual resources (tasks, timers, queues, etc) and directly communicate between them using the virtualization interfaces' shared block memory manager. This allows for more efficient communication and a better integration of the applications. This also allows applications to bypass hardware devices or network interface layers, while being integrated and tested on a host environment.

- **Better performance**: When applications are running natively on one OS platform, engineers will have greater source-level control for performance optimization. Consolidating to a single OS means your system needs to set aside fewer OS resources and CPU resources can be dedicated to a single application utilizing SMP features.

Cross-OS Hypervisor Architecture

Cross-OS Hypervisor is part of MapuSoft's AppCOE™ (Application Common Operating Environment), an integrated development platform built on the powerful open source Eclipse® framework, supporting C, C++ and Ada programming languages. Cross-OS Hypervisor is built upon the OS Abstractor® platform and offers OS Interfaces for VxWorks, pSOS, Nucleus, ThreadX, Windows, Linux/POSIX and micro-ITRON applications.

OS Abstractor is a component of the Cross-OS Development Platform™, a C/C++ source-level virtualization technology that provides a robust and industry standard OS interface architecture for flexible real-time application development, while allowing the user to protect the software from being locked to one OS. This negates future porting issues because the software will support multiple operating systems and versions from the beginning. It also eliminates the risk associated with the OS selection process, since the same application can be tested on multiple platforms for comparison and won't be tied to the chosen OS.

OS Abstractor acts as a seamless separator from the application code to the operating system, supports over 20 commercial OS's, minimizes switching of applications, and provides the inter-process, inter-task communication layer for applications to communicate. OS Abstractor also alleviates key issues, including the support of new hardware.

Cross-OS Hypervisor is available for installation on Windows and Linux Host machines for the development, simulation and testing of embedded applications without the target hardware. It also provides source level debugging, an App/Platform Profiler and code generation options to support a wide range of target OS platforms and configurations.

The App/Platform Profiler is based on The Eclipse Test and Performance Tools Platform (TPTP) which allow developers to build unique test and performance tools, both open source and commercial, that can be easily integrated with other tools. This profiling tool identifies the APIs used heavily by the application and automatically optimizes them by eliminating their function wrappers.

Cross-OS Hypervisor's code generator builds a custom virtualization interface package that is optimized for specific applications and target environments. It reads application source code to determine the OS services used by your application and produces an interface code highly optimized for your specific application and target OS platform. Cross-OS Hypervisor gives you the ability to generate code even for your in-house OS with only minor customization effort required. The interface code is compiled and linked with the applications using native target tools to generate highly optimized application executables.

The following diagram depicts the architecture of the Cross-OS Hypervisor.

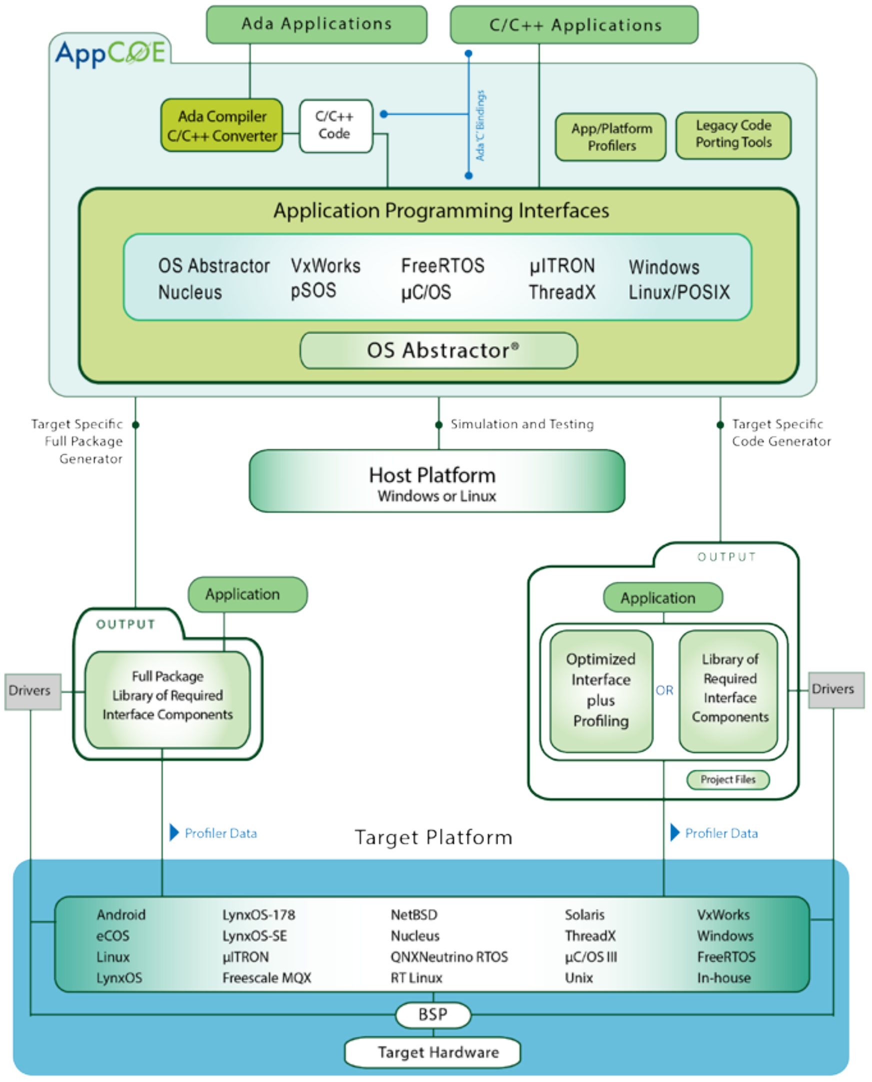

Fig 3.1: Cross-OS Hypervisor – Architecture Diagram

Current versus Next Generation Hypervisors

Current Generation Hypervisors	Next Generation Cross-OS Hypervisor
Require Guest OS. Increased cost in terms of OS royalty and licensing fees for using the guest OS.	Does not require the Guest OS. Hypervisor offers the guest OS features itself and enhances the application's performance by eliminating multiple OS and device I/O layers
Multiple guest OS configurations increase the hardware resource requirements. Further, the OS resources are not effectively used across the application.	Reduced hardware requirements (i.e. memory, disk space, etc.) because applications are consolidated to a single OS. Also, the OS resources are pooled and re-used across multiple applications to reduce system requirements.
Requires multiple set of development tools to support each guest OS	Streamlined development by using one set of tools, device drivers and BSPs for one OS.
Hypervisors are not generally available in full source code format	Hypervisor source code is available and can be used for: • customization and integration • application specific optimization • simplified debugging • compiling using native tools
Hypervisor available for limited target OS. Also, the design is locked-in with a specific OS vendor's solution, preventing you from easily switching the OS in the future.	Hypervisor available for over twenty target OS and the design ensures that the solution is not locked into a particular OS.
Not possible to integrate the Hypervisor solution with in-house OS platform.	Hypervisor solution can easily be integrated with in-house OS platform.
Profiling of hypervisor code not possible.	Hypervisor can be profiled to identify the API level of use for optimization purposes. Further, it can identify bottlenecks within applications for further performance improvement.
Direct interaction between guest applications is not possible.	Hypervisor allows applications to directly interact by allowing them to set their selected OS resources in shared mode. For example, applications can directly trigger a timer or an event resource belonging to another application. This also eliminates the need for setting up virtual device for communication and also avoid layers of data transfer via use of shared memory.

OS resources are not shared between applications and as such are not effectively used.	OS resources are pooled for re-use across applications to minimize the overall system resource requirements and to enhance performance.
Hypervisor not available with a host-based environment	Hypervisor available with a host-based environment. Further, applications can also take full advantage of the powerful host resources like CPU, memory, drivers, middleware and tools for better development, system testing and application simulation.
Hypervisor is not integrated with SMP and as such does not offer effective load sharing of multiple CPUs.	CPU resources can be pre-defined and dedicated to a single application utilizing the SMP features offered by the target OS. Host applications can be either independent applications running as separate processes or grouped as a standalone application running as a fully independent dedicated process on single core under SMP.
Hypervisor does not enhance the guest OS to improve application performance.	Hypervisor hardens the underlying target OS platform by providing application specific optimizations. It can also add real-time features to non real-time target OS like Windows and Linux to enhance application stability and performance.

Additional Information

- The Cross-OS Hypervisor datasheet can be downloaded here:
  http://www.mapusoft.com//wp-content/uploads/documents/cross-os-hypervisor.pdf

- For the latest release notes about the API coverage provided by the OS Interfaces visit this link:
  http://www.mapusoft.com/wp-content/uploads/documents/Release_Notes.pdf

- A free evaluation can be downloaded here:
  http://mapusoft.com/downloads/

- For user manuals & technical documentation visit this link:
  http://www.mapusoft.com/techdata/

- For technical or sales questions please submit a ticket at the MapuSoft support site at this link:
  http://mapusoft.com/support/