Tech Note: Embedded Hypervisors Fall Short of INTEGRITY-178 tuMP for Security and Multicore Features

Contents

Executive Summary 1

Hypervisor Architecture 2

INTEGRITY-178 tuMP Architecture 2

Overhead Comparison 3

- Hypervisors Add Overhead 3
- INTEGRITY-178 tuMP has no Virtualization Overhead 3

Advanced Partitioning 4

- Time Partitioning 4
- Resource Partitioning 4
- Robust Partitioning 4

Multicore Interference Mitigation 5

Multicore Interference Mitigation in INTEGRITY-178 tuMP 5

Advanced Scheduling 6

- Scheduling Overview 6
- Scheduler Layers 6
- Flexible Multi-Processing 7
- Moving Applications across Cores and Time Windows 8

Security Functionality and Assurance 8

- Common Criteria 9

Conclusion 10

References 10

Executive Summary

Some embedded software companies promote hypervisors as the leading solution to provide security on a multicore processor. Although hypervisors can be a quick way to provide a basic level of isolation between applications, hypervisors are less secure and impose many more architectural constraints than the INTEGRITY-178 tuMP high-assurance real-time operating system (RTOS), which is based on a hardened separation kernel.

Initially, some of the companies promoting those hypervisors tried to extend their single-core RTOS to be a multicore RTOS but found it much more challenging than expected. As a result, the expedient alternative was to maximally leverage the more novel, and sometimes unproven, capabilities of hardware virtualization and create another layer of software in the form of a hypervisor to take care of running applications on different cores.

Most embedded hypervisors aim to increase security by keeping the software functionality to a minimum when advanced functionality is actually needed to meet high-assurance security requirements, especially for multicore systems. Hypervisors also lack the advanced functionality to address the toughest challenges facing system integrators of multicore systems for embedded security and airborne safety, such as:

- Multicore interference mitigation (which enables robust partitioning),
- Task scheduling optimized across partitions and cores,
- Rate monotonic analysis across applications and cores,
- Mixing different multi-processing (AMP, BMP, SMP) models in the same schedule to achieve maximum multicore utilization, and
- Movement of applications across time windows and cores.

The INTEGRITY-178 tuMP RTOS solves those issues and more for multicore systems. Compared to a minimal hypervisor or separation kernel, this advanced functionality enhances both safety and security. Even with those enhanced features, the INTEGRITY-178 tuMP kernel remains small enough to evaluate to Common Criteria EAL 6+ while also achieving DO-178C/ED-12C DAL A safety.

Hypervisor Architecture

A hypervisor is a construct designed to provide hardware virtualization. Hardware virtualization creates virtual machines (VMs) that emulate the physical hardware and run their own guest operating system (OS) isolated from other VMs. The VMs are controlled by a virtual machine monitor, more commonly referred to as a hypervisor. Each VM mimics the same underlying hardware but can have a different guest OS. The hypervisor allocates each VM a separate address space, which the hypervisor enforces using the CPU's memory management unit (MMU). All the software inside the VM executes as if it were running directly on the physical hardware isolated from other VMs and their software stacks.

When a VM needs access to some shared physical resource, such as the network, display, or disk drives, either the hypervisor intervenes and controls that access directly or a novel form of hardware I/O virtualization is utilized. The alternative is to dedicate an entire VM to managing the device and acting as an I/O server for that device. Hypervisors can introduce additional latency for real-time applications running on a guest RTOS compared to when the RTOS runs directly on the hardware (Figure 1).

The combination of hardware I/O virtualization technology and hypervisors has yet to undergo and complete the rigorous process of formal safety evaluation, such as DO-178C Level A, by an authorized certification authority, such as the FAA, Transport Canada, or EASA, or formal security evaluation by the US National Security Agency. The novel form of unproven virtualization technology translates into extra effort and increased risk for systems requiring certification.

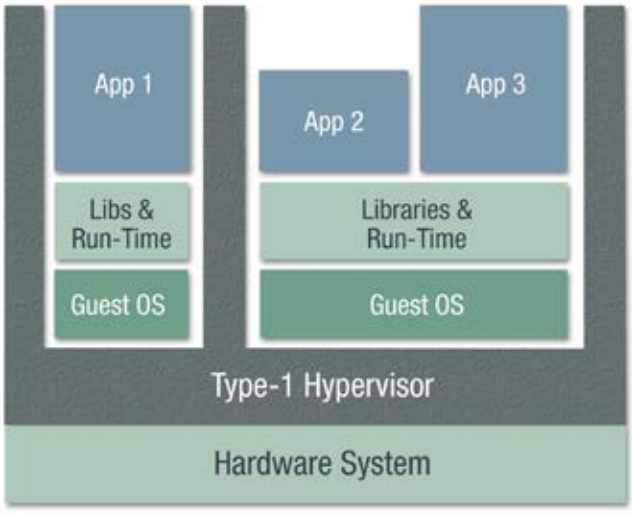


Figure 1: Hypervisor with some real-time applications

INTEGRITY-178 tuMP Architecture

The INTEGRITY-178 tuMP RTOS is a safety- and security-critical real-time operating system based on a secure separation kernel. A separation kernel is a minimized OS microkernel whose function is to enforce four basic security policies: data isolation, fault isolation, control of information flow, and resource sanitization. Other OS services are moved to user space so that both the size and attack surface of kernel-mode code are the absolute minimum (Figure 2). Software components with the largest attack surfaces and the most vulnerabilities, such as the networking stack, file system, and virtualization, can execute just fine in user mode, so the security policy should not allow them to execute in privileged kernel mode.

Separation kernels are typically used as the base layer of a secure operating system, with most OS services executing in user space instead of kernel space. The combined separation kernel and OS services in INTEGRITY-178 tuMP are the foundation of achieving the highest levels of safety and security certifications. INTEGRITY-178 tuMP does not rely on novel hardware virtualization techniques for high-assurance applications, which translates into less risk from a certification perspective and less effort for customers not needing to provide assurance evidence for novel hardware features.

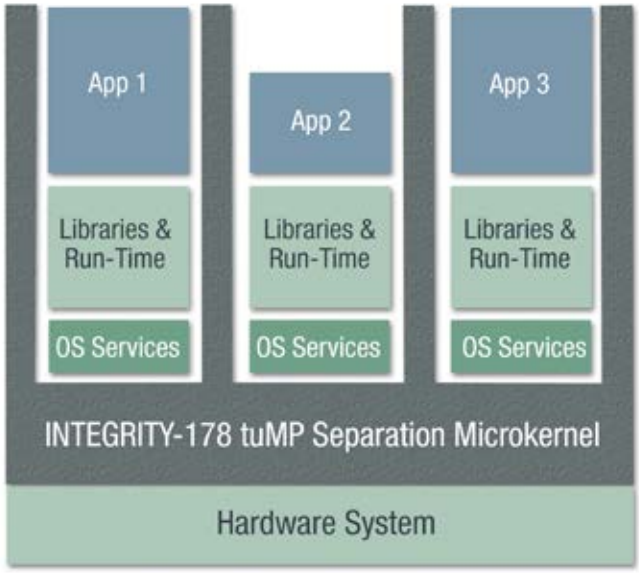


Figure 2: Separation kernel with OS services running in user space

Overhead Comparison

Hypervisors Add Overhead

Hypervisors add an extra layer of software below the application and the OS. If virtualization is not required, there is significant overhead for unneeded functionality. The additional software layer also reduces determinism, which can be an issue for safety-critical applications.

Some hypervisors support unikernels as an attempt to partially offset that overhead. Unikernels collapse the guest OS into the application running in a VM by using a library OS (Figure 3). However, implementing an application as a unikernel is complex in that it requires expertise in both the application and the library OS.[1] Generally, that burden falls to the application programmer.

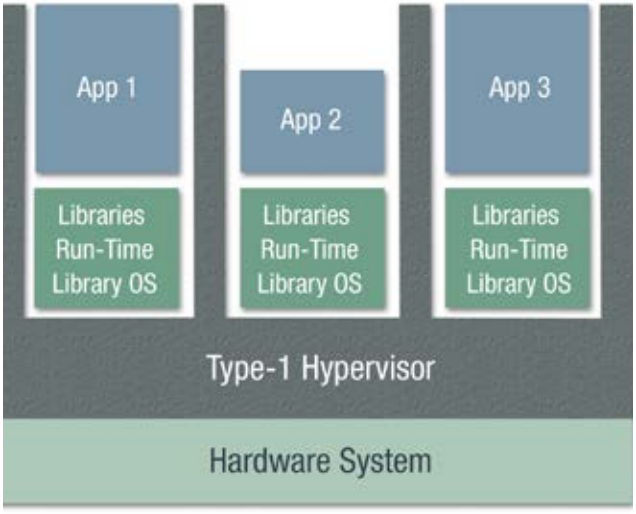


Figure 3: Unikernels use a library OS and run in a VM

If a unikernel supports real-time requirements, then it is really no different than the traditional flat RTOS from decades ago. One of the primary reasons operating systems added support for running application code in user space was to benefit from memory protection rooted in hardware using the memory management unit (MMU). This memory protection eliminated many bugs that plagued single address space operating systems, such as crashing the entire system by dereferencing a stray or null pointer.

For communication between VMs, embedded hypervisors generally set up the approved communication links and then tout that they "get out of the way." While this sounds like it improves performance, the opposite is actually true because this forces the communications to be asynchronous, and the applications have to implement a polling architecture instead of blocking and waking up when data is available.

INTEGRITY-178 tuMP has No Virtualization Overhead

INTEGRITY-178 tuMP isolates each partition without forcing each application to run on top of a virtualization layer. All real-time applications see real-time performance. A virtualization layer implemented in user space is available for legacy applications and general-purpose operating systems, such as Linux. Only the applications that actually require virtualization get the virtualization overhead (Figure 4).

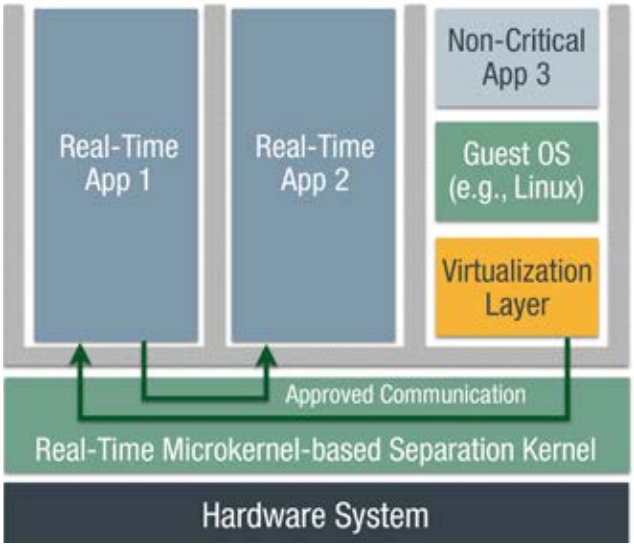


Figure 4: With INTEGRITY-178 tuMP, only applications that require virtualization get the overhead

INTEGRITY-178 tuMP prevents communication among partitions unless specifically enabled. All communication enabled between partitions is limited to authorized sources and destinations in a specified mode, such as read-only or read/write. Those information flows are statically defined and loaded at boot time.

INTEGRITY-178 tuMP provides secure synchronous communication between partitions that can block and wake up when data is available. This has higher performance and lower overhead than an asynchronous polling architecture, especially when required for low latency system event handling such as a classified data sanitization function or power-down event.

Advanced Partitioning

Both safety and security depend on partitioning the system to isolate applications from each other and the system software. There are three dimensions to partitioning: space, time, and system resources. The ultimate goal is to enable robust partitioning, in which no partition can affect another partition's access to resources or execution time and no partition can propagate a fault condition.

Space partitioning is the basic level of partitioning, and it is used broadly across computer systems. Space partitioning uses the hardware MMU to provide a separate address space for each partition. Hypervisors and INTEGRITY-178 tuMP both use the hardware MMU to enforce memory space partitioning.

Time Partitioning

Time partitioning divides a fixed time interval, called a Major Frame, into a sequence of fixed sub-intervals referred to as partition time windows or Minor Frames. Each application is allocated one or more partition time windows, with the length and number of windows being factors of the application's worst-case execution time (WCET) and required period. The ARINC 653 standard addresses both space and time partitioning of real-time operating systems for safety-critical avionics applications.

Some hypervisors support a rudimentary form of time partitioning by allowing VMs to be scheduled in fixed time windows. Any full implementation of time partitioning is left to a guest OS that supports ARINC 653 running inside the VM. If a hypervisor supports fixed time windows, all cores generally are required to swap VMs simultaneously.

INTEGRITY-178 tuMP implements a partition time scheduler modeled after ARINC 653 Part 1 plus extended functionality. INTEGRITY-178 tuMP provides additional flexibility with simultaneous support for both rate-driven and event-driven requirements. The "time-variant" capability of INTEGRITY-178 tuMP supports varying major frame rates across user-defined groups of cores, and the mode change capability can swap the major frame schedule to a predefined alternative schedule. The latter capability is also supported under ARINC-653 Part 2 Multiple Modules Schedules. Not only can applications change from one module schedule to another, but also the core assignments for an application can change as well, including the number of cores assigned to a particular application.

Resource Partitioning

System software contains buffers, tables, and other objects used to control the partitions or VMs. Resource partitioning allocates a fixed amount of those software resources to each partition so that one partition cannot use up the software resources required by another partition.

In multicore systems, there is also a significant amount of hardware resources that are shared among cores. Hardware resource partitioning needs to allocate those resources such that one partition cannot affect access to the resources needed by another partition. Part of hardware resource partitioning includes mitigation of multicore interference, which can deny access to shared resources for a period of time. Multicore interference mitigation is discussed in detail below.

INTEGRITY-178 tuMP implements resource partitioning of both software and hardware resources. Most hypervisors do not address resource partitioning beyond cores, memory, and I/O devices.

Robust Partitioning

Robust partitioning is a capability that enables verification of a safety-critical application independently from other applications as well as an independent measurement of worst-case execution time (WCET). Robust partitioning thereby reduces the time and cost of adding new or modified applications into existing systems. For systems without robust partitioning, both regression testing and analysis must be performed to assess the impact of the new or modified applications on the continued correct operation and margins of the unchanged applications.

Robust partitioning on a multicore processor is much more complex than on a single-core processor. In addition to time and space partitioning for each core, partitioning must account for contention when different processor cores access shared resources simultaneously. The CAST-32A position paper on multicore processors specifies several requirements for robust partitioning. The two most challenging are "Software partitions cannot consume more than their allocations of shared resources" and "No software partition

consumes more than its allocation of execution time on the core(s) on which it executes, irrespective of whether partitions are executing on none of the other active cores or on all of the other active cores." The key to multicore robust partitioning is multicore interference mitigation, which is explained in a section below. Effective multicore interference mitigation can guarantee each processor core and the partitions running on it get access to the allocation of resources and execution time, thereby enabling robust partitioning.

No hypervisor has robust partitioning for multicore processors, and some hypervisor vendors even claim that robust partitioning is impossible. Hypervisor vendors have only recently started their research into a solution. Conversely, Green Hills Software started in 2010 to invest over 100 staff years to develop a general solution to multicore interference mitigation, thereby enabling robust partitioning.

Through robust partitioning in time, space, and resources, INTEGRITY-178 tuMP minimizes regression testing for unmodified applications. This reduced effort translates into considerable cost savings and faster time-to-market.

Multicore Interference Mitigation

All multicore hardware architectures include shared resources, such as memory controllers, I/O, last-level cache, and the internal fabric that connects them. Contention for these shared resources results when a processor core attempts to access a resource already in use by another core. In safety-critical applications, the principal concern is how such shared resource contention can cause an application running on one core to interfere with an application running on another core, negatively affecting determinism, quality of service, and, ultimately, safety.

The impact of such multicore interference can be significant. For example, with just one other core contending for the same memory controller, the worst-case execution time (WCET) of an application can be up to 8 times longer than with no interference for some architectures. With a second and third interfering core, the WCET can increase to over 13 times longer (Figure 5).

There are a variety of techniques to reduce multicore interference, such as consolidating all I/O to a single core. Such techniques cannot eliminate multicore interference, only reduce it. Other techniques, such as hand-tuning the

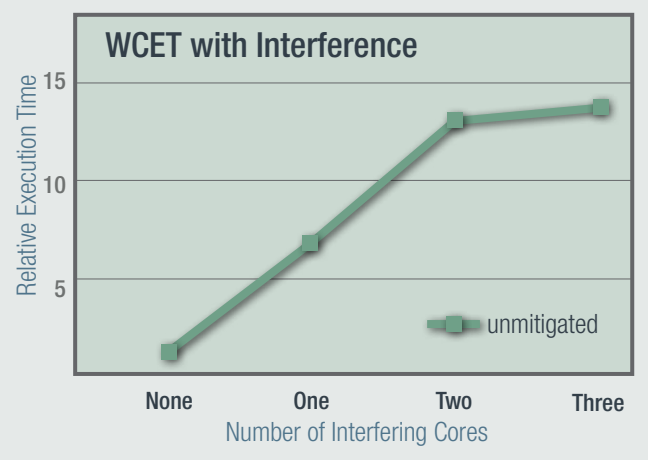


Figure 5: WCET increases faster than the number of interfering cores

schedule of application execution to try to avoid contention, are limited in effectiveness and require extensive, repetitive testing and verification when any of the applications are modified or a new application is added. A general solution to multicore interference is needed to achieve robust partitioning on a multicore processor and avoid virtually endless retesting and reverification during development, integration, and maintenance.

Multicore Interference Mitigation in INTEGRITY-178 tuMP

INTEGRITY-178 tuMP is the only multicore operating system solution that provides a comprehensive and built-in approach that not only thoroughly identifies interference but also mitigates that interference such that critical applications are guaranteed to meet their deadlines. Calculating worst-case execution times (WCET) for critical applications requires the identification of all interference as well as a guaranteed method for the system integrator to limit it. All other operating systems and hypervisors push the multicore interference problem onto the system integrator, sometimes suggesting that the customer use a third-party tool to partially assist them by attempting to identify where interference may exist. Mitigating that interference then becomes a system integrator's responsibility and juggling act.

Some vendors tout that they enable access to performance counters on multicore processors. Opening such access barely scratches the surface of what is needed to use

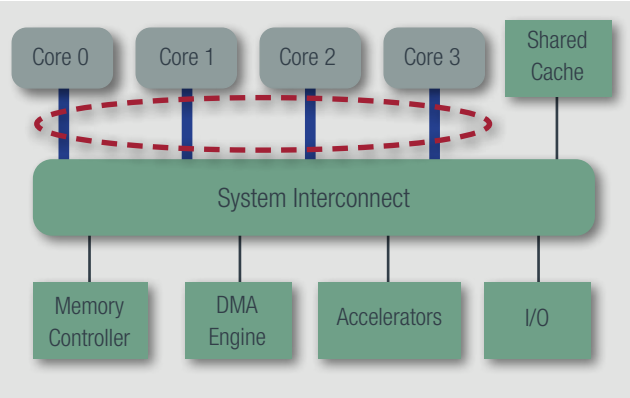


Figure 6: BAM regulates the bandwidth from processor cores to shared resources

performance counters to mitigate multicore interference. Without a built-in DAL A compliant solution to interference mitigation, the addition of future functionality and system sustainment also becomes a costly nightmare.[2]

The INTEGRITY-178 tuMP RTOS includes bandwidth allocation and monitoring (BAM) functionality to limit interference between cores by regulating the bandwidth to the shared resources (Figure 6). Based upon more than 100 staff-years of research and development into multicore interference analysis and mitigation strategies, BAM enables the software architect to allocate a portion of the system bandwidth to each core. BAM then regulates core access to shared resources over very small time quantums to provide smooth and consistent performance independent of the amount of interference generated by the other cores. BAM was developed to the highest design assurance level (DAL A) and can be used to meet the CAST-32A multicore objectives.

By fully mitigating multicore interference, BAM reduces certification risk and enables faster time-to-market by simplifying the verification and analysis activities while also significantly lowering the cost of long-term sustainment and growth of the system. Without such multicore interference mitigation capabilities, system reverification after adding or modifying applications incurs a very high risk of needing to retest and reverify every application.

Advanced Scheduling

Scheduling Overview

Both safety-critical systems and security-critical systems isolate applications into partitions, and then those partitions need to be scheduled to run on one or more CPU cores. To achieve high levels of assurance, time partitioning assigns a fixed duration time window for execution. The open standard for time partitioning is ARINC 653, "Avionics Application Software Standard Interface."

Applications can be divided into multiple interconnected tasks, and those tasks need to be scheduled within the partition schedule. Those tasks should be scheduled with a typical real-time algorithm, such as priority pre-emptive scheduling. Ideally, the scheduling algorithm should be mathematically optimal, meaning that it best meets the real-time deadlines with the available cycles. Rate monotonic scheduling is a common optimal scheduling algorithm.

Scheduler Layers

Hypervisors execute applications inside of a VM. The hypervisor is responsible for scheduling the execution of the VMs and assigning them to processor cores, and each hypervisor implements different scheduling algorithms. The guest OS running in the VM is responsible for scheduling applications and tasks inside that VM. If the guest OS is a safety-critical OS, some have separate schedulers for partitions and tasks within a partition. The result is 2-3 layers of scheduling in a hypervisor-based system, with no optimization across the layers (Figure 7). Beyond the lack of optimization, multi-level scheduling also generally prevents multiple applications from being part of the same time partition.

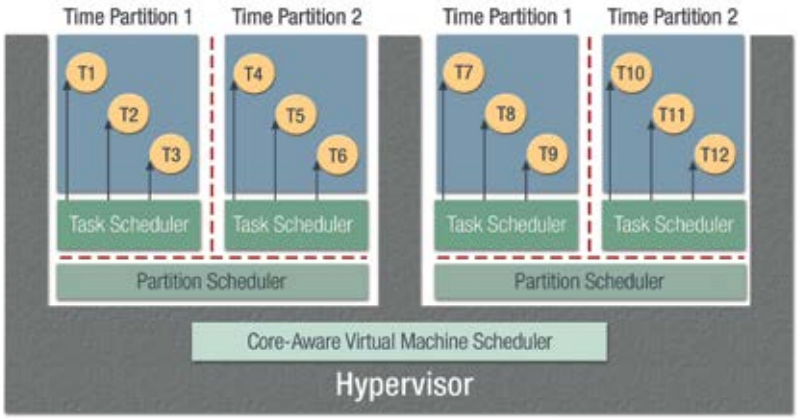


Figure 7: A hypervisor-based solution can have three separate layers of schedulers that do not talk to each other.

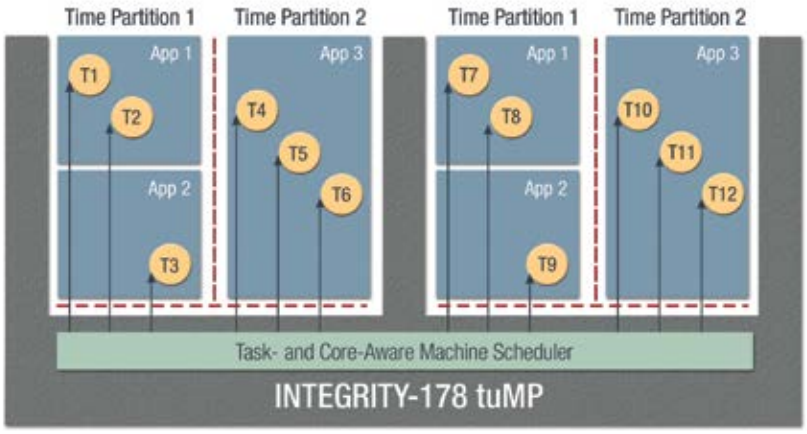


Figure 8: INTEGRITY-178 tuMP has a unified partition scheduler that is aware of both tasks and cores

INTEGRITY-178 tuMP is "unified" in that a single operating system runs across multiple cores. The partition scheduler is fully aware of tasks, partitions, and cores, resulting in a single-layer scheduler that globally optimizes both task and partition schedules (Figure 8). Each core can move to the next time window at times that are different from other cores, enabling further optimization of the partition and task scheduling.

Having a single scheduler handle both partitions and tasks enables INTEGRITY-178 tuMP to have multiple applications per partition with the ability to optimize task schedules across those applications. That optimization includes, but is not limited to, rate monotonic analysis across those applications, yielding a mathematically optimal real-time schedule.

Flexible Multi-Processing

The software architecture on multicore processors can be classified by the amount of sharing and coordination among cores. The simplest software architecture for a multicore-based system is Asymmetric Multi-Processing (AMP), where each core runs independently, each with its own OS or hypervisor/Guest OS pair. Each core runs a different application with little meaningful coordination between the cores.

The use of AMP can result in underutilization and the inability to perform coordinated activity across cores, such as required for comprehensive built-in test.

The modern alternative is Symmetric Multi-Processing (SMP), where a single OS controls all the resources, including which application threads are run on which cores. Processes and threads can be coordinated across cores, and utilization can be maximized. Bound Multi-Processing (BMP) is an enhanced and restricted form of SMP that is particularly useful for safety-critical systems. BMP statically binds an application's tasks to specific cores, allowing the system architect to tightly control the concurrent operation of multiple cores.

Most embedded hypervisor-based systems do not support SMP or BMP, just AMP, reducing the options for a software architect to use all of the cores efficiently. Some hypervisors can schedule copies of the same VM to run on different cores simultaneously, but that is not the same as a single application running across multiple cores in BMP or SMP mode.

INTEGRITY-178 tuMP has full support for SMP, BMP, and AMP. Its BMP implementation directly follows the multicore requirement in ARINC 653 Supplements 4 and 5 section 2.2.1, which requires the ability to have "multiple processes within a partition scheduled to execute concurrently on different processor cores."

INTEGRITY-178 tuMP goes further in that a partition schedule can have any mix of AMP, BMP, and SMP grouping across cores in the same time window. The time-variant capability of INTEGRITY-178 tuMP allows the grouping of cores and applications to vary over time, thereby providing the added flexibility to change the assignment of tasks to cores or even change between AMP, BMP, and SMP for different time windows (Figure 9).

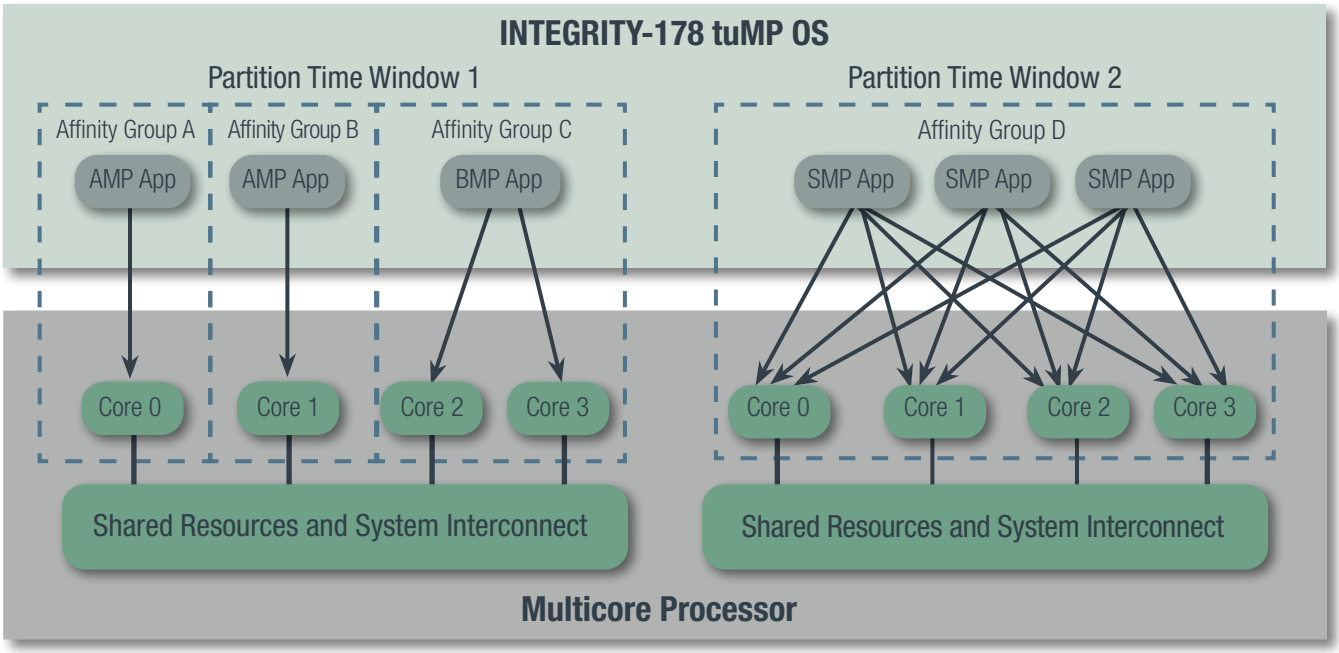


Figure 9: INTEGRITY-178 tuMP supports mixing AMP, BMP, and SMP as well as changing between time windows

Moving Applications across Cores and Time Windows

System architects and integrators can give an application extra execution time by assigning it an open time window on any core or adding a core to an existing time window in the INTEGRITY-178 tuMP configuration file for native applications. Both options are simple and easy because INTEGRITY-178 tuMP supports inter-process communication (IPC) mechanisms that work seamlessly across cores. For example, an application can release a semaphore on any core and obtain it on another core, receive an interrupt on one core and signal a task on another core, or run AMP in one window and SMP in another.

For an integration-level example, consider an application initially designed to utilize 100% of a single core and uses traditional OS services for task communication. What happens a couple of years after the application was developed and now requires more than one core? One solution is to run SMP just for that time window, which is easy with INTEGRITY-178 tuMP but not possible with a hypervisor. A hypervisor does not perform any task scheduling, and guest operating systems, including unikernels, cannot transition between single and multicore scheduling when switching between partition time windows. If using a hypervisor in this scenario, integrators would likely need to re-architect and re-analyze the application since it will now have to run entirely as SMP on an SMP-enabled guest OS or unikernel. As a result, a hypervisor can present significant challenges during integration and functional upgrades, as the granularity in which integrators can assign CPU resources is very coarse. Integrators might be able to give VMs longer or additional time windows to the currently assigned cores, but they cannot easily change the number of cores on a partition window-by-window basis because the VMs need to be a given allocation to those resources at boot time.

Security Functionality and Assurance

Hypervisors are assumed to be secure because their initial implementations relied on proven hardware constructs such as the memory management unit (MMU) and IOMMU to provide separation. However, the advent of newer hardware virtualization techniques, such as I/O virtualization, fundamentally changes that argument. Given the lack of design assurance in commercially available hardware solutions, this puts a tremendous burden on systems developers to provide the required assurance that demonstrates these capabilities may be relied upon to provide partitioning. Although such approaches result in less hypervisor software, this minimalist approach to security leaves open many potential attack

vectors, particularly through the hardware virtualization functionality.

Because applications running on a hypervisor generally have no awareness of running in a virtualized system, any compromises to the virtualization system are very hard to detect. When a hypervisor runs on bare metal, it has the unique position as the software layer that all others depend upon to provide the base security. Hypervisors often have a complex code base with a large number of entry points and complicated interconnections, all of which encourages a growing amount of attack vectors.[3] If the hypervisor security is breached, the attackers have access to all the VMs, all the applications, and all application data inside of them. This makes the hypervisor a prime target for an attack. Virtualization vulnerabilities are many and varied, such as VM escape, hyperjacking, denial-of-service (DoS), and side-channel attacks. Any claims of being secure need to be backed up with security certifications such as the Common Criteria.

Some embedded hypervisors are scaled-down versions of enterprise hypervisors like Xen. Such hypervisors still have a code base that is an order of magnitude larger than a separation kernel, and over 600 common vulnerabilities and exposures (CVEs) have been reported for Xen. The other approach for an embedded hypervisor is a minimalist hypervisor that aims to have a code base size that approaches that of a separation kernel. The small size is accomplished by relying almost entirely on hardware virtualization features to provide the isolation. After configuring the VMs, the hypervisor loads and boots the guest OS and then, as one simple hypervisor puts it, "gets out of the way." The hypervisor itself does not directly engage in protecting the system with respect to loss of function, data integrity and data confidentiality. Such simple hypervisors claim that they have reduced the threat surface due to the small code base. However, the threat surface is not limited to software. The hardware virtualization is a large threat surface, and the system is now relying on novel hardware features that are based on less rigorous design assurance. The result is less system security due to the VMs unconstrained access to the hardware virtualization threat surface.

INTEGRITY-178 tuMP takes a different approach and allows the system architect to utilize the proper partitioning mechanisms based on the overall system safety/security analysis and architecture. Such analysis considers functional partitioning mechanisms in association with the assurance data for those mechanisms. Many systems require an active security monitor that provides high levels of security functionality and assurance as defined by the Common Criteria.[4]

Common Criteria

The security robustness of computer hardware and software platforms can be specified by evaluation to the international standard "Common Criteria for Information Technology Security Evaluation" (ISO/IEC 15408).[5] Typically, Common Criteria targets of evaluation (TOE) are evaluated against a government-defined protection profile that includes both functional and assurance requirements. Evaluations can be done to different levels of depth and rigor, called Evaluation Assurance Levels (EAL), with EAL 1 being the least rigorous and EAL 7 being the most rigorous.

Starting with EAL 2, each of those EALs requires a vulnerability analysis to a specific level. EAL 5 is the first level that protects against attackers of "moderate" potential, and it is only at EAL 6 that the resistance to penetration attacks raises to "high" attack potential (Figure 10). The only government-defined protection profile ever designed for EAL 6 and above that applies to operating systems or hypervisors is the "U.S. Government Protection Profile for Separation Kernels in Environments Requiring High Robustness" (SKPP). The NSA-defined SKPP is a rigorous and detailed specification that contains 29 groups of security functional requirements and 36 groups of security assurance requirements.

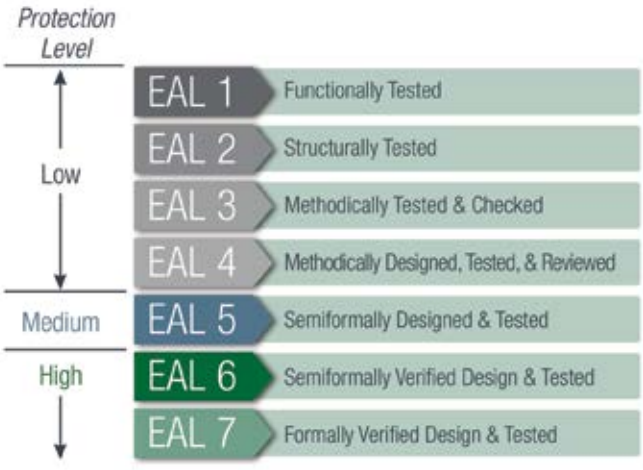


Figure 10: Common Criteria EAL 6 and above protects from attackers of high potential

The INTEGRITY-178 separation kernel and RTOS have been certified multiple times to EAL 6+ and "High Robustness" using the SKPP. No other commercial operating system or hypervisor has been certified to EAL 6+ or High Robustness. In fact, no embedded hypervisor has completed any Common Criteria certification, and no enterprise hypervisor has achieved a level to address even attackers of moderate potential.

As part of certification to the SKPP, INTEGRITY-178 underwent independent vulnerability analysis and penetration testing by the NSA to demonstrate that it does not allow hostile and well-funded attackers with high attack potential to violate the security policies. Extending that security design to multicore processors, INTEGRITY-178 tuMP continues to meet the SKPP's rigorous set of functional and assurance requirements.

Conclusion

Based on a hardened separation kernel, the INTEGRITY-178 tuMP RTOS achieves the highest possible levels of safety and security assurance for multicore processors. Being a true RTOS, INTEGRITY-178 tuMP provides many more OS services than a hypervisor. Most multi-level security applications, such as cross domain guards, require a rich set of OS functions, including tasking services, semaphores, and message passing.

Together, the safety and security assurance certifications of INTEGRITY-178 are proven in real-world customer applications with over 75 DO-178B/C Level A and/or EAL 6+ unique customer certification packages delivered across more than 40 different single and multicore processors, including Arm, Intel, and Power Architecture. INTEGRITY-178 tuMP directly addresses the toughest issues of system coupling and multicore interference instead of leaving those to the system integrator. It does this without introducing reliance on novel hardware virtualization capabilities except when required by low assurance level applications

Many of those successful customer deployments have included avionics systems that required compliance with DO-178C/ED-12C and CAST-32A/A(M)C 20-193, including four different technical standing orders (TSOs). Certification of those systems depended upon the multicore robust partitioning and multicore certification support provided by INTEGRITY-178 tuMP. To date, no other OS or hypervisor has achieved civil certification of multicore avionics or Common Criteria certification to address attackers of medium potential, let alone persistent high potential.

References

1. S. Kuenzer, et al., "Unikraft: Fast, specialized unikernels the easy way," ser. EuroSys'21. New York, NY, USA: ACM, 2021
2. "Solving Multicore Interference for Safety-Critical Applications," Green Hills Software, 2019. https://www.ghs.com/download/whitepapers/GHS_multicore_interference.pdf
3. Wu J, Lei Z, Chen S, Shen W, "An access control model for preventing virtual machine escape attack." Future Internet 9:2, 2017.
4. "Creating a Trusted Platform for MLS Applications," Green Hills Software, 2020. https://www.ghs.com/download/whitepapers/GHS_Trusted_Platform_MLS.pdf
5. Common Criteria for Information Technology Security Evaluation, Version 3.1, revision 5, April 2017. Part 3: Security assurance components.



Corporate Headquarters

30 West Sola Street • Santa Barbara, CA 93101
ph: 805.965.6044 • fax: 805.965.6343 • email: info@ghs.com • www.ghs.com

European Headquarters

Fleming Business Centre • Leigh Road • Eastleigh • Hampshire S050 9PD • United Kingdom
ph: +44 (0)2380 649660 • fax: +44 (0)2380 649661 • email: info-emea@ghs.com

Safety & Security Critical Products

34125 US Hwy 19 North • Suite 100 • Palm Harbor, FL 34684
ph: 727.781.4909 • fax: 727.781.3915 • email: info-sscp@ghs.com

Green Hills, the Green Hills logo, INTEGRITY, and tuMP are trademarks or registered trademarks of Green Hills Software in the US and/or internationally. All other trademarks (registered or otherwise) are the property of their respective owners. © 2023 Green Hills Software. v0423