Asymmetric Multi-Processing (AMP) Systems vs. Symmetric Multi-Processing (SMP) Systems

"Nothing is more fairly distributed than common sense: no one thinks he needs more of it than he already has."

- René Descartes, Discourse on Method

This quote can be applied to how Symmetric Multi-Processing (SMP) distributes applications for multi-core processers, or that many do not understand the nature, or common sense, behind a real-time OS (RTOS). The general purpose applications running under a General Purpose Operating System (GPOS), are fairly distributed across the multi-core resources. But this distribution of the GPOS applications is in conflict with the absolute deterministic needs of real-time applications. Given that most hard real-time applications are inherently single threaded, why surrender that explicit control to a black box SMP scheduler with "fair" and non-deterministic execution? Real-time system solutions are engineered to be deterministic. To be deterministic, the work needs to be prioritized (free from interruptions and interference) and reliable (fixed to hardware resources) with coordinating critical services that scale over the nodes of the system. By definition, this is an asynchronous and asymmetric problem ideally solved with an appropriate AMP system solution.

Multi-core processors were developed to increase processing power in computers when the clock rate reached a practical limit. To address maximum utilization of the multi-cores, GPOS, including Microsoft* Windows*, uses SMP to distribute the workload dynamically or "on the fly" to the multiple cores in a system. For many applications this works just fine, and increases the overall throughput of the computer system. SMP supports the scalability of applications using processors with different numbers of cores, and enables the easy addition of new applications to the workload, but there is a performance hit and some unpredictability for the core management overhead. SMP relies on a shared environment of memory and I/Os, as well as identical cores so each process can execute on any of the available cores without any change or limitation. The implementation of SMP used in Windows is not concerned about when a process delivers results, but that the processes in use remain responsive to the user, leaving background processes to languish. When a system gets overloaded, all processes are affected, but remain "fair". The GPOS also must perform housekeeping and typically many other background tasks periodically, affecting all processes. All of these variables adds to the unpredictability of an already inherently non-deterministic SMP system.

Hard real-time applications, however, demand deterministic timing for results. An application is deterministic if the timing requirements can be met all the time, and also with bounded timing jitter. This deterministic requirement affects both the process scheduling priority and the use of I/O devices. Historically, hard real-time embedded controllers used dedicated processors and resources required to perform the task in real-time. Separate, dedicated processors are a form of AMP. Combining

embedded hard real-time applications on multi-core systems presents some challenges to determinism, but also offers great cost savings, and also allows the addition of non-real-time applications onto a shared system. There are three major considerations that enable combining hard real-time embedded applications on a shared system.

- One consideration is the need to assign fixed workloads to each node (core) in the multi-core host system, or AMP.
- The second consideration is to partition or assign system resources to each individual node.
- The third consideration is to provide an inter-process communication (IPC) mechanism.

Asymmetric Multi-Processing (AMP) provides workload separation

AMP is the method used for systems with multiple cores of different architectures. For example, a computer with a DSP, GPU, co-processor, or a core with enhanced micro-code or accelerators, has different cores to handle specific tasks. While these are examples of customized cores, real-time workloads are also assigned to a specific core, not for different core capability, but to keep the time-critical results deterministic. Unlike SMP systems which typically run one instance of a GPOS or node on multiple cores, AMP for an RTOS assigns a different operating system instance or node for each core. When each node has a tailored set of characterized and debugged processes, the workload does not change, therefore the application will have deterministic timing.

The workload for each RTOS node needs to be balanced to not only meet the required minimum response time but also to avoid any conflicts between tasks that would introduce non-determinism. If needed, the real-time solution is statically redistributed over multiple real-time nodes. Each AMP node needs to be characterized for bounded on-time performance with its entire workload, to guarantee a minimum response time. With workload separation on distinct nodes, one workload cannot have an impact on another workload. The goal for RTOS is to be on time, all the time, unlike the goal for SMP which is to fill all processor cycles with activity.

INtime® Distributed RTOS from TenAsys® provides multiple hard RTOS nodes for real-time applications on a multi-core host. INtime® for Windows* from TenAsys® addresses the need to combine multiple RTOS nodes for hard real-time applications with a GPOS node running Microsoft* Windows*. The Windows node provides the Human Machine Interfaces (HMI) and other non-real-time applications on the same multi-core host. All nodes run directly on the hardware with no hypervisor to slow the I/O or require any driver customization.

AMP explicit partitioning is key to determinism

Explicit partitioning reserves and assigns system resources: memory, I/O, and interrupts, for the exclusive use of an OS node. Exclusive use of these resources insures that a node never needs to arbitrate for these resources, which would introduce a variable timing delay, and break the deterministic timing. Memory for an RTOS node is typically locked into non-page pooled or physical memory, avoiding memory block fetches.

INtime for Windows partitions these resources by modifying the boot configuration for the Windows environment, reserving cores and memory for the INtime nodes. Partitioning is done explicitly with the help of standard Windows APIs. I/O ports and interrupts are also dedicated to each INtime node to preserve performance and allow determinism. Each Windows and INtime node runs natively on their assigned nodes (Figure *1*). None of the operating system nodes are affected by virtualization, as there is no hypervisor to take context away, and because Windows runs natively, there is no violation of the Microsoft licensing restrictions of running embedded versions (e.g. Windows Embedded Standard 7) on a virtualized platform.

In INtime Distributed RTOS, the system resources are reserved for each node. Explicit hardware partitioning is a proven solution used in TenAsys products since 1997.

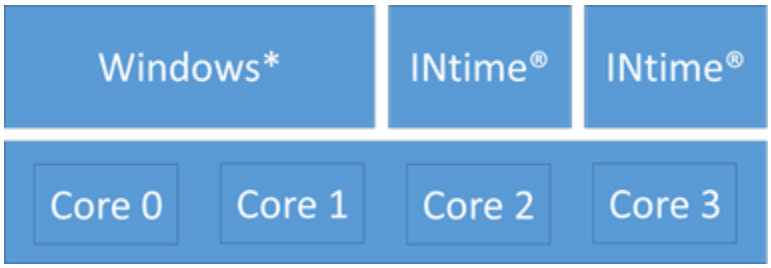

Figure 1: Explicit hardware partitioning of OS environments with Microsoft Windows and INtime on a 4-core processor.

Global objects facilitate Inter-Process Communication (IPC)

With application processes executing on different nodes with different operating systems and even remote or distributed systems, a reliable means of inter-process communication (IPC) that is more robust than simply passing messages in shared memory regions is required. Shared memory can work within a single SMP system where the one GPOS provides a master communication service for all processes, but in an AMP environment a mastered inter-process communication resource would restrict the scalability and reliability of the system. Changing to platforms with more cores or adding new processes or remote systems would require changes to the system architecture, impacting system cost and reliability. A solution for an AMP system is an OS that incorporates the concept of global objects, supporting a flexible distributed IPC that does not require a master resource from one OS. Distributing the IPC across all cores makes the IPC expandable without changes.

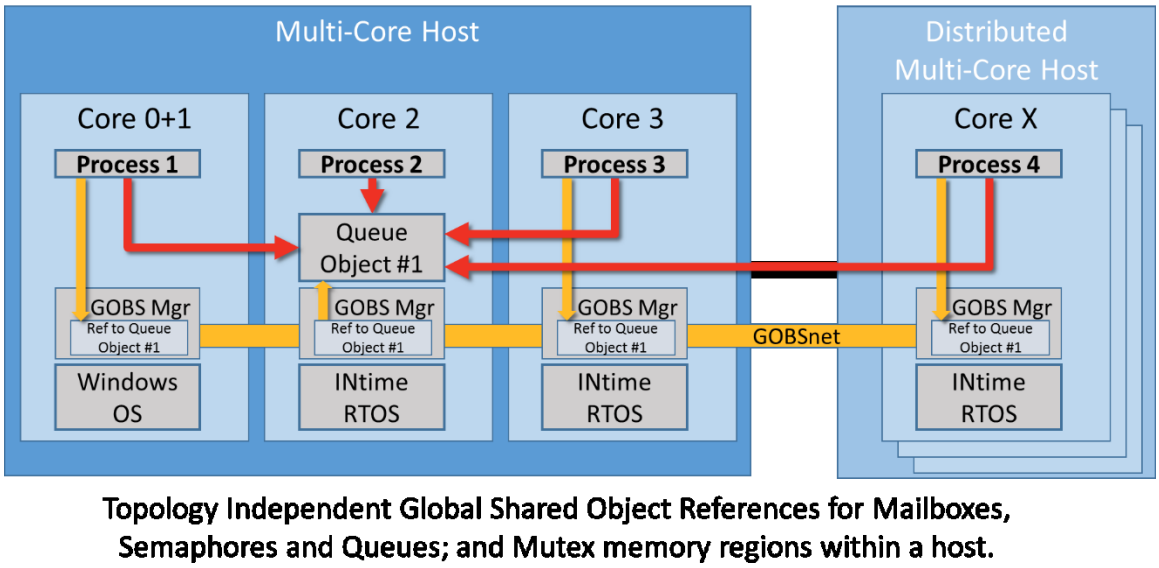

Figure 2: The use of global objects facilitates the interaction between tasks running on different hosts.

Figure 2 illustrates how global objects (GOBS) in INtime RTOS work. Using a multi-host multi-node system as an example, a process can acquire a reference to a global object that is owned by another process. The distributed GOBS manager keeps the location of the object up to date so other processes can reliably reference that object.

The use of global objects to enable communications between the processes is a fundamental element of how the TenAsys® INtime® RTOS handles multiple schedulers and multiple environments in an AMP system. Communication functions include time synchronization and alarm services along with mailboxes, queues, semaphores and controlled shared memory regions. The IPC is also used by built-in functions to share Windows resources, such as file I/O with the disk drive and human machine interfaces (HMI) with the RTOS nodes. I/O stacks employing media such as virtual Ethernet between processes are also used to connect nodes.

Summary

System consolidation using multi-core processors and specialized hard real-time operating system (RTOS) software that is designed to deliver determinism with AMP principles will reduce the cost and complexity and continue to give OEMs a competitive advantage. SMP is simply not suited for the demands of hard real-time applications. The INtime RTOS Software from TenAsys achieves the requirements for hard real-time applications and provides for more flexibility in deployments than other solutions. The INtime RTOS Software provides a rich set of built in capabilities to support fast real-time application development. Solutions designed to consolidate multiple processes with AMP, distributed on multiple cores (e.g. Intel's Atom* processors) offer broad scalability and price performance options to embedded system suppliers.