WIND RIVER

An Introduction to Multicore Technology and VxWorks SMP

Table of Contents

Executive Summary

The growth in multiprocessing is the beginning of a very important trend. More and more devices are being designed using multicore processors because getting to the next level of device performance is necessary. This paper explores multicore technology and the choices and challenges it presents to organizations and developers building next-generation multicore devices. It also describes Wind River's VxWorks SMP product and how it addresses the opportunities presented by multicore technology to enable Wind River VxWorks platform users to move into this exciting new area of embedded computing.

Introduction to Multiprocessing

Multiprocessing is the use of two or more processors in a system, in which the processors cooperate and distribute among themselves the total processing load. Multiprocessing is not new. Servers, enterprise systems, and telecom carrier equipment providers have used multiprocessing for many years. Some high-end embedded devices use many discrete boards on backplane busses and even several discrete processors on a board.

Today's new technology is not multiprocessing by itself, but multiprocessing using multicore processors. Multicore processors have more than one processor engine (core) on the same chip. Each core has its own program counter (i.e., its own instruction stream). Chip architectures allow the cores to "talk" over an interconnect medium of some type, most typically local memory.

Every multicore processor allows the execution of more than one piece of software simultaneously. Multiprocessing gives you true concurrency, meaning that more than one piece of software is running at exactly the same instant on one or more other processors, but their behavior when they run is fundamentally different. Multitasking and multithreading on single-processor systems has typically interleaved the execution of more than one thread at different points in time.

Terms such as hyperthreading, simultaneous multithreading, and chip multithreading are all architectural techniques applied to a single processor. Instructions go into the processor through a pipeline waiting to execute. Processors are often blocked while waiting for instructions to be fetched from memory. Multiple instructions are in progress, but still there is a sequential stream of instructions going in and a sequential stream of completed results coming out. In chip multithreading, the concept is expanded a little more because there are more likely to be dependencies between instructions in the same thread than between instructions in separate threads.

Chip multithreading switches between multiple threads at the hardware level, much like an operating system does in software. This is still not the same as multicore because the different hardware threads are being multiplexed onto one core. Hyperthreading, simultaneous multithreading, and chip multithreading concepts are not mutually exclusive. You can have a chip multithreaded multicore processor running a multitasking operating system.

These technological developments put multiprocessing within the cost, power, and space budgets of many more device applications than is possible with multiple discrete processors or boards. This is significant because more and more device companies will be motivated to enter the multiprocessing world on the promise of increased device performance, cost and space savings, and feature richness made possible by using multicore processors.

Multicore Processor Trends

Semiconductor manufacturers are driving the trend toward multicore processors. For years, higher performance was achieved by packing more transistors on chips and by increasing clock frequencies. These advances could be realized within a power budget and a cost envelope that were acceptable to customers. But the semiconductor industry is now at the point where the physics behind that strategy no longer allows those advances. Semiconductor manufacturers cannot raise the frequency as high and keep the power consumption of the device at a reasonable level. Figure 1 shows the effect of decreasing frequency on the power consumption.

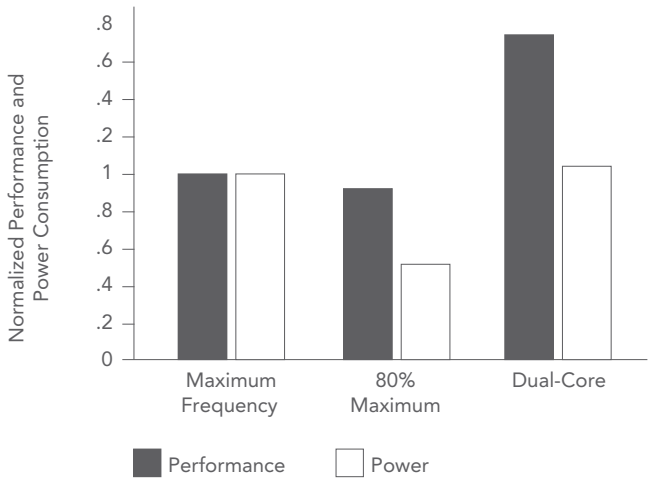

Figure 1: Normalized performance and power consumption for a given processor shown at left

In a processor the power consumption is proportional to the cube of the voltage, and the voltage proportionally relates to the frequency. So at a lower frequency you can run the processor at a lower voltage and the power consumed is substantially reduced. If the clock frequency is dropped by 10%, performance goes down by 10% but the power reduction is much bigger, as seen in the center bars in Figure 1. A little sacrifice in performance saves a lot on power consumption. Now if two of those processors can be put on the same chip with a small reduction in clock frequency, in theory, there would be the situation shown on the right of Figure 1: a processor with about the same power consumption but with a lot more theoretical processing power.

Shrinking geometries and an increasing number of transistors on a die made it economical to place multiple CPU cores on dies operating at slightly lower frequencies. It is these trends that have resulted in multicore processors. But for software engineers to exploit increased processing power delivered in this way is fundamentally different than exploiting increased processing power resulting from higher clock frequencies.

Multicore Technology and the Device Software Market

Survey data from Venture Development Corporation (VDC), an independent technology market research and strategy consulting firm, shows strong interest in and increasing use of multicore technology in the device software market. Many device manufacturers are already using multicore processors and multiprocessing for high-performance systems. In the enterprise space, multicore processors are already dominant: nine out of 10 Intel processors shipped for servers are already multicore. The enterprise market has already tipped toward multicore processors.

The device software market is not going to tip nearly that fast. But as a trend, multicore is going to keep on marching forward. The semiconductor manufacturers, because they are in the business of providing compute power in smaller packages at lower costs, are all manufacturing their own multicore processors. Some, such as Intel, have directed their entire product line toward multicore technology; while others, such as Freescale, maintain a more balanced portfolio of single-core and multicore processors, depending on the markets they serve.

More and more devices will require higher-performance processors, and architects and developers will find themselves with no choice but to adopt multicore processors to achieve higher-performance devices within the constraints of power, weight, space, and cost. The commercial off-the-shelf (COTS) board manufacturers are also supporting this trend. There are many COTS multicore processor–based boards available today from companies such as Curtiss-Wright, Kontron, Radstone, and Mercury Computer.

Multicore Business Issues

The benefits of multicore processors are the same as the benefits of any other high-performance processors at the same level of power for the same level of cost. They provide the ability to run more software concurrently and consequently to speed up devices, add more feature content, optimize performance, and ultimately increase customer value.

But there is a real cost in adopting multicore for most if not all device manufacturers. Most developers will have to deal with true software concurrency for the very first time. Writing new software or reusing existing software investments for a multiprocessing environment is more complicated, takes longer, and introduces more risk to projects. The transition to multicore technology will be relatively simpler for those device manufacturers that have already been building multiprocessing systems. However the majority of device manufacturers that have only ever created software for single-processor systems are going to step into a fundamentally different environment.

The Difference in Multicore Technology

Three aspects make developing applications for multicore environments fundamentally different than developing for single-core environments:

1. It's not guaranteed that your application will gain performance from multiprocessing. Not all problems can be solved with multiprocessing. You may have to find a fundamentally different algorithm to do the work. Existing algorithms may work in uniprocessor environments but may not scale to more than one processor.
2. The system design is more complicated than a single-processor system, assuming you have an algorithm that can be concurrently executed. There is a strong interaction between the way the hardware supports multiprocessing, the way the operating system and system software support multiprocessing, and the way the application is partitioned to take advantage of multiple processors. Users must select the right combination of hardware, OS, and application design to realize a system that delivers higher performance than a single-processor system. Often there is a fair amount of trial and error involved in getting the right combination that optimizes performance.
3. There is a great deal of complexity in the software development environment. Programmers for years have learned the concepts of and have practiced sequential programming. For example, when multiple threads of an application are operating concurrently it exposes latent programming defects. It violates some fundamental assumptions against which the software was originally written: that concurrently executing pieces of code were never expected to run concurrently. Developers must deal with synchronization issues (i.e., adding synchronization on shared data that did not exist before) and race conditions exposed by these assumptions. The behavior of software can become less deterministic because of the interaction of all these different threads, the interrupt activity in the background, and the way the operating system schedules the threads.

Application Suitability

If you have a job to do and you have two people to do it, you can get it done faster than with only one person. Cores in a multicore processor are like more workers you can apply to the job. Obviously you have to have enough work to keep all the workers busy. Then you have to be able to split the work into enough pieces to give every worker something to do. But sometimes it can be like having too many cooks in the kitchen, where more workers does not solve the problem faster. The workers have to work well together or efficiency can suffer.

Assembly lines are an example of multiprocessing. A lot of work can be accomplished by carefully coordinating multiple activities so they happen in parallel as much as possible. Yet there are dependencies between the actual stages through which the work goes. If the dependencies are not right or the work is not done correctly, the subsequent steps stall and output suffers. A delivery warehouse that sorts packages and loads them on to the right delivery truck is an example of an application with almost no dependencies. Each package is processed independently from every other package. This system has a very high level of parallelism. The challenge is that of shared resources. In the example of the delivery warehouse, if there were too few carts available to move the packages around, every worker would be waiting. This situation is called resource contention.

Some tasks simply cannot be done faster by adding more people or resources, such as reading a book. It is strictly a serial operation. Supercomputers and massively parallel computers are used only for certain kinds of applications such as weather forecasting, financial modeling, and so on, but not necessarily for every computing problem.

Amdahl's Law: Realistic Performance Expectations

Amdahl's law is another limit that must be understood. Every problem has a percentage of work that can be done in parallel and a percentage of work that must be done in order (i.e., serially). Think about cooking; certain things have to be done in a specific order and some things can be done in parallel. Depending on how much of the work can be done in parallel, there is a limit on how much faster the work can be done no matter how many workers (or processors) are applied to the problem. Amdahl's law states the following:

> Parallel speedup = 1/(Serial% + (1-Serial%) / Number of processors)

Here Serial% is the percentage of work that must be done serially, and (1-Serial%) is the percentage of work that can be done in parallel. Figure 2 shows the effects of Amdahl's law on applications with varying degrees of serialization and number of available processors.

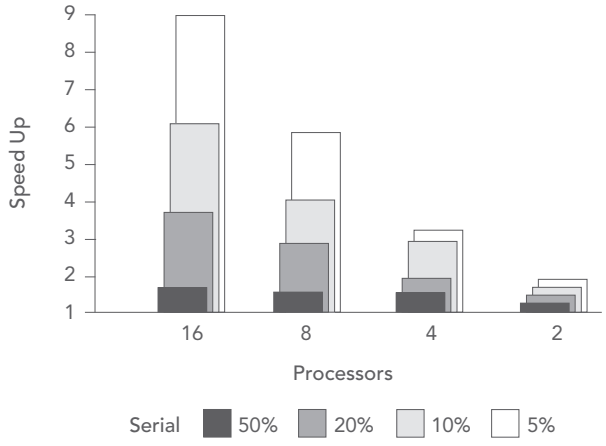

Figure 2: The effects of Amdahl's law on applications with varying degrees of serialization and number of processors

With 5% serialization, the performance speedup is relatively linear as the number of processors increases. But with 16 processors, the maximum speedup achieved is nine times. The 5% serialization limits the maximum speedup possible to well below 16 times for a 16-processor system. As the percentage of serialization increases, the maximum speedup gained as well as the rate of performance gained is reduced significantly. At 50% serialization, performance gain almost flattens beyond four processors. Developers building a device with 10% serialization will not feel that they are getting good value from an eight-core system that is only performing at four times the speed of a single core device.

Even small increases in the amount of work that has to be done serially can reduce your maximum performance gains by large amounts. Therefore, there is a limit to obtaining more performance by adding more CPUs for many applications. Tools and techniques for reducing the amount of software serialization become key to optimizing the device for multiprocessing.

Problems that can be parallelized more (or completely) are the ones where multiprocessing delivers significant gains in throughput and performance. One example is in networking on the data plane where the system is handling multiple independent streams of incoming packets. Each stream is independent of the others; as many processors can be applied as there are data streams. Multicore processors help such systems because more processors on a chip yields a system that is smaller and lower power yet allows more work to be done as compared to a single-processor system.

Multiprocessing, and multicore in particular, heavily influence the architecture of the hardware, which in turn determines the architecture of the system software, and in turn influences the architecture of the application.

There are many ways in which work in a given application can be partitioned for parallelism. Application partitioning is one of the most important aspects device manufacturers have to worry about when designing multiprocessing systems.

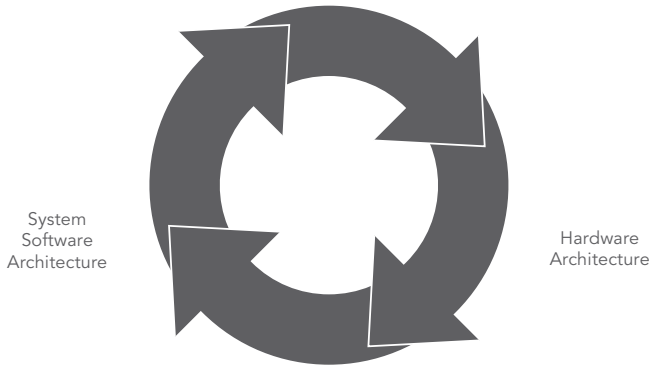

Figure 3: Application architecture

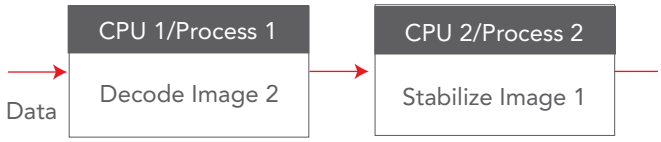

Figure 4: Coarse-grained serial execution

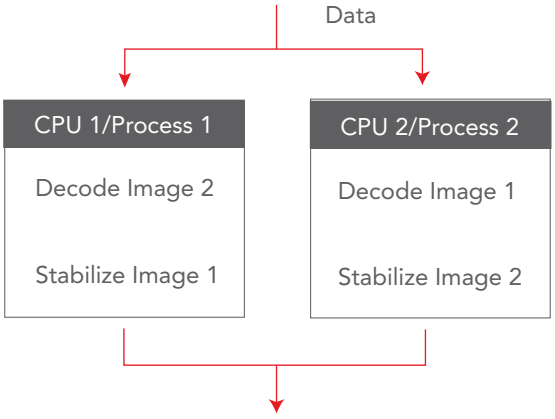

Figure 5: Coarse-grained parallel execution

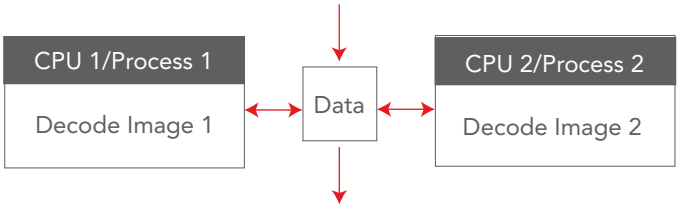

Figure 6: Fine-grained simultaneous data access

All forms of application partitioning shown in the figures are valid and appropriate for different problems. The system designer's task is to find the right application partitioning design for his problem and his hardware design. On the top is a pipelinelike design, much like an assembly line. On the bottom is the distribution of independent work to more than one worker (such as in the example of the delivery warehouse).

The most complicated partitioning design is shown in Figure 6. This design deals with fine-grained partitioning. Here there is one set of data, and to get the performance required, multiple operations must be completed on that data; there is no strict ordering in which these operations need to be completed. Maximizing performance requires more than one worker working on the same problem at the same time. This is like putting many workers on the job but it must be done in a way that they are all able to work effectively on the same job and get it done faster. Each worker takes up a little piece of the larger job and works on it independently from other workers; the work is partitioned on a fine granularity as compared to assigning bigger blocks of work to each worker.

Operating System Configurations for Multiprocessing

There are two configurations of operating systems suitable for multiprocessing systems: symmetric multiprocessing (SMP) and asymmetric multiprocessing (AMP).

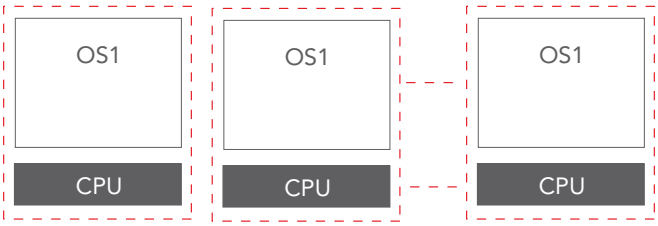

Figure 7: An asymmetric multiprocessing OS configuration

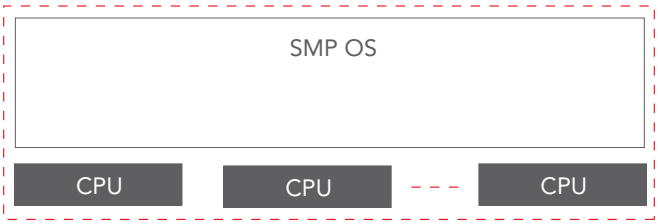

Figure 8: A symmetric multiprocessing OS configuration

In an AMP configuration (shown in Figure 7), an operating system runs on each processor, or core, of the multiprocessing system. Each processor/OS combination is really a single-processor computer in its own right. The system as a whole is comprised of more than one of these computers (also referred to as nodes), which act in concert to do a given job. What ties together these independent nodes into a larger system is an interconnect mechanism the processors use to communicate with each other. The interconnect mechanism is most typically a bank of shared memory between the processors but can sometimes be a network connection or other peripheral bus. In an AMP system, the job performed by each individual node must be defined when the system is designed. The job of the whole system is thus partitioned into work items assigned to each individual node in the system.

AMP systems can come in a variety of combinations. You can choose to have dissimilar processors in the system. You can choose different operating systems in an AMP system, depending on your needs. For example, VxWorks could handle real-time sensitive tasks while Wind River Linux could handle the media interface. Sometimes some nodes may perform a very limited function that does not require them to have a complete operating system but just a simple executive program leveraging the hardware directly.

AMP systems typically need more memory than SMP systems to hold all the different operating systems and their applications. AMP systems by their very nature are very dependent on the precise hardware configuration you design for. Moving to another hardware configuration with fewer or more processors or different interconnect mechanisms, as when creating a family of devices with different price points, causes you to repartition your applications and reverify the system to work on the new hardware configuration.

An SMP configuration (shown in Figure 8) is where one operating system controls more than one identical processor (or core in a multicore processor). Applications "see" and interact with only one operating system, just as they do in single-processor systems. The fact that there are several processors in the system is a detail that the OS hides from the user. In this sense, an SMP operating system abstracts the hardware details from the user. An SMP operating system is also symmetric; it needs to work with multiple identical processors, each of which can access all memory and devices in the system in a uniform fashion. Therefore, any processor (or core) in an SMP system is capable of executing any task just as well as any other processor in the system can. This symmetry in the hardware allows an SMP operating system to dispatch any work to any processor in the system. An SMP operating system tries to keep all processors busy running application threads, in effect load balancing the system's work.

As a result of hardware abstraction and load balancing in the system, the SMP operating system simplifies the task of developing software to run on SMP hardware. From the programmer's view, there is only one OS to write an application for, which will automatically distribute the workload to all available processors. An SMP operating system therefore provides the same programming semantics as a uniprocessor system.

All processors in an SMP system share one pool of memory under the control of one operating system. The memory and local caches for each individual processor are kept synchronized by the chip hardware. Therefore applications in an SMP system can share data with each other in memory very easily and efficiently. This makes SMP systems suitable for applications that need to share large amounts of data with low latency.

Since any SMP operating system must synchronize and control more than one processor, it will necessarily have higher internal overheads than an operating system that only controls one processor. On individual benchmark comparisons, SMP systems are almost always slower than uniprocessor systems, but SMP outperforms uniprocessor systems when there is a high degree of parallelism in the application that allows the system to do more work in a given time than a uniprocessor system can.

Choosing Between SMP and AMP

An SMP operating system is the best choice when the OS is expected to effectively load-balance differing parallel workloads. One of the essential properties of the hardware for SMP to run is hardware-enforced cache coherency. This causes the hardware to automatically update copies of data in a processor's cache when it is modified by another processor. For running programs, this update is transparent so that modified data can be immediately visible in memory and accessible by fast memory. When tasks share a lot of data located in memory, accessing this data at memory access speeds is both fast and easy in an SMP operating system. SMP is also the right choice when the application needs to be portable between systems with a differing number of cores. In all these cases, the hardware abstraction and load balancing performed by SMP kernels will free application developers from having to tune their applications for different hardware configurations.

AMP is best suited for systems that are cleanly partitioned into different subsystems that are mostly autonomous but have limited well-defined data communication needs with other nodes. This gives system designers the choice of designing their hardware with a heterogeneous mix of processors that are ideally suited for the function each node in the system performs. AMP is also the most suitable choice for redundant systems needing high availability, as each node in an AMP system is a standalone single-processor computer in its own right.

VxWorks SMP

Wind River's VxWorks SMP is an add-on product to all Wind River VxWorks platforms (6.6-based and later) that provides SMP capabilities to VxWorks. VxWorks SMP leverages multi-core processors to achieve true concurrent execution of applications, allowing applications to improve performance through parallelism.

The VxWorks SMP add-on, when combined with a VxWorks platform, provides users with a complete symmetric multiprocessing–ready platform comprising the run-time, middleware, and Wind River's market-leading development tools suite.

The following table explains the advantages and drawbacks of each OS configuration.

AMP	SMP
When to Choose • Application is easy to partition into loosely coupled nodes • Redundancy is needed for reliability • Hardware configuration is not suitable for SMP • Multiple processor/OS types are needed • Assignment of applications to hardware resources must be explicit • Minimum scheduling overhead is required	When to Choose • Application exhibits a high enough degree of parallelism to keep the processors busy • Tasks/threads need to share a lot of data frequently • Load balancing is important • Application portability across different hardware is essential
What to Watch Out For • AMP design may use more memory (multiple OS images) • AMP system needs to be tuned for that hardware configuration • Setting up communications between cores and debugging is complex	What to Watch Out For • SMP kernels have higher overhead than uniprocessor kernels • Applications making a lot of OS calls will run slower than on a uniprocessor system • Latent software defects arising from concurrency assumptions are likely to be exposed when migrating code from uniprocessor systems

Along with providing SMP capabilities, VxWorks SMP has been designed to be compatible with the uniprocessor version of VxWorks. This is to allow VxWorks users to more easily migrate their VxWorks applications, drivers, and other legacy software from a uniprocessor environment (VxWorks UP in short), to an SMP environment.

VxWorks SMP is purchased as a separate add-on product to any of the Wind River VxWorks 6.6 platforms (versions 3.6 of Wind River General Purpose Platform, Wind River Platform for Automotive Devices, Wind River Platform for Consumer Devices, Wind River Platform for Industrial Devices, or Wind River Platform for Network Equipment). Once installed, users can develop either uniprocessor or SMP applications as needed.

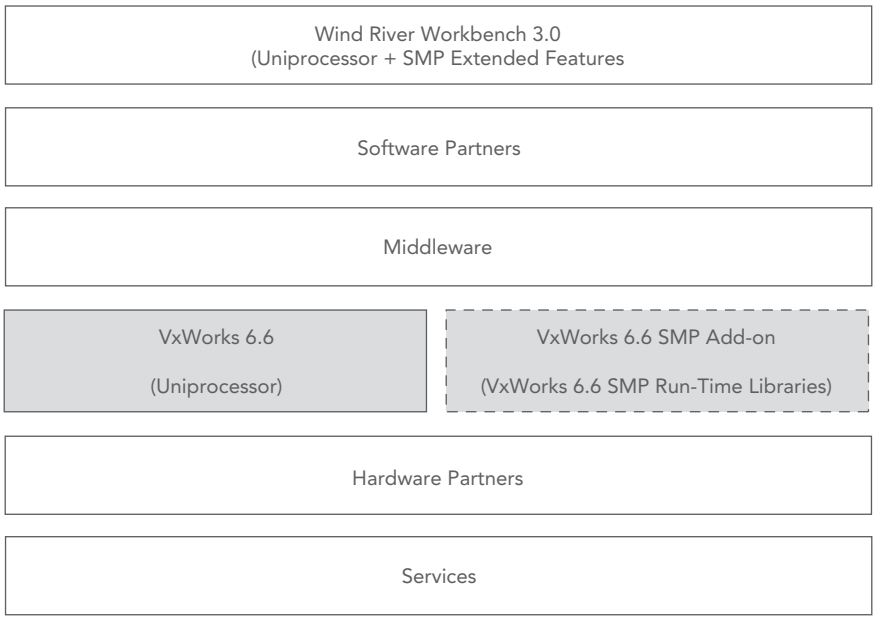

Figure 9: VxWorks SMP installation with Wind River VxWorks Platforms

Figure 9 depicts a VxWorks 6.6 installation with SMP.

VxWorks SMP Features

VxWorks SMP provides symmetric multiprocessing capabilities to the world-leading VxWorks RTOS. VxWorks SMP is built from the same source base as VxWorks. Specifically, VxWorks SMP provides the following:

1. SMP capability
   VxWorks SMP provides VxWorks users with the capability to run applications in an SMP environment on industry-leading multicore processors. This enables the delivery of more functionally rich, higher-performing devices than is possible with single-processor systems.
2. Compatibility
   VxWorks SMP is API compatible with the uniprocessor version of VxWorks (with a few exceptions). API compatibility promotes reuse of existing VxWorks-based software, allowing companies to leverage previous VxWorks investments. Major features such as real-time processes and error detection and reporting are fully supported even in an SMP environment.
3. Real-time behavior
   VxWorks SMP adds SMP capabilities to VxWorks without compromising on the real-time behavior and characteristics of VxWorks. This allows VxWorks SMP to be applicable in the same target markets and target applications as uniprocessor VxWorks systems are. VxWorks SMP has a deterministic scheduler that guarantees task execution in priority order. The scheduler dispatches the N highest priority tasks that are ready to run, where N is the number of processors in the system. Deterministic priority-based scheduling is also one of the critically differentiating features of VxWorks on single-processor systems; this is what differentiates an RTOS from a general-purpose operating system. VxWorks SMP implements interrupt-level parallelism, namely the ability for more than one core in the system to handle different interrupts simultaneously. This increases the responsiveness of the system as whole. Many critical operations within the OS are protected by spin-locks. These operations complete in deterministic time, which is a function only of the time for which the spin-lock is held. Spin-locks also protect critical data during interrupt processing. These spin-locks are held for a deterministic length of time, which in turn makes the system's interrupt latency deterministic as well.
4. Support for leading multicore silicon
   VxWorks SMP runs on all the leading multicore processors that are in the market today. This includes multicore processors from ARM, Broadcom, Cavium, Freescale, Intel, and Raza Microelectronics. This offers users a wide choice of hardware platforms on which to build their next generation of devices.
5. Cross-architecture portability
   This has been a VxWorks hallmark since its inception. Like its uniprocessor sibling, VxWorks SMP is a portable operating system running on a wide variety of processors with uniform characteristics. SMP truly extends this unique VxWorks capability to the multiprocessing domain.
6. Familiar development environment
   Wind River Workbench and its constituent tools work with both uniprocessor and SMP versions of VxWorks. Also, the tools have additional capabilities for VxWorks SMP, which are designed to visualize parallel execution and loading patterns in all cores. The following sections provide more details on the capabilities of the various tools.

Applications VxWorks SMP Is Suited For

VxWorks SMP is suited for problems that need both an SMP operating system and RTOS capabilities such as determinism, low latency, and small footprint. The preceding sections show the distinctions and unique capabilities of an SMP operating system configuration. To recap, VxWorks SMP would be the right solution when one or more of the following conditions are true:

1. The application is (or will be) designed in terms of a set of parallel threads, allowing the system to perform more work simultaneously than a uniprocessor VxWorks system would. There must be enough parallelism in the application to allow meaningful performance gains from using multiple processors. Recall Amdahl's law and how it puts an upper limit on how much performance may be gained for a given level of parallelism.
2. Threads share large amounts of data, or have a fine-grained data-sharing design.
3. The system designer prefers not to explicitly partition applications to individual processors, instead relying on the operating system to schedule ready-to-run threads on all available processors (also known as load balancing).

For example, a network router can get better throughput by having multiple cores process packets from many different streams in parallel. Network throughput can also be increased by having some cores act like security processing hardware offload engines, in which one core can run the main TCP/IP stack logic, while others concentrate exclusively on routing or crypto functions. In the example of the network router, multiple cores can process additional independent data streams in parallel, while in the other example, a part of the network processing load is performed in parallel by other cores. Both of these examples exhibit the possibilities offered by multiprocessing, getting more work done in a given quantum of time than possible with just one processor.

New Features in VxWorks SMP

Some of the major new features and APIs in VxWorks SMP are described here. This is not an exhaustive list of features, so for more information visit www.windriver.com or contact your Wind River account team.

1. Spin-locks
   Spin-locks are lightweight, fast, multiprocessor-safe mechanisms used to synchronize threads and interrupt handlers running on different processors in an SMP system. Spin-locks can be thought of as lightweight mutex semaphores for multiple processor environments. Spin-locks are of two basic types (task-level and interrupt-level) and can be used to provide synchronization both between different tasks and between tasks and interrupt service routines.
2. Atomic operators
   Atomic operators are a class of APIs that provide fast multiprocessor-safe operations on simple memory variables, for example, increment, decrement, bitwise logical operations, and simple arithmetic operators. These operators update the variables atomically with respect to other processors in the system and obviate the need to use slow, higher latency mechanisms such as interrupt locks or preemption locks (as are often used in unicore processor or uniprocessor systems). The atomic operators developed for VxWorks SMP are also available in the uniprocessor version of VxWorks. Their use is encouraged for all VxWorks application development in VxWorks 6.6.
3. Reader-writer semaphores
   A reader-writer semaphore is a new type of semaphore in VxWorks 6.6 (available in both uniprocessor systems and SMP). A reader-writer semaphore allows many reader threads to gain fast read access on a shared data structure. Only one writer thread is allowed to modify the data structure at a time. These semaphores are optimized for situations with multiple readers and one writer. Their use is encouraged in producer-consumer type software algorithms and is especially suited for multiprocessing systems. Like VxWorks mutex semaphores, reader-writer semaphores also support priority inversion protection as an option.
4. Thread-local storage APIs
   Thread-local storage is a new multiprocessor-safe facility to manage thread-local variable storage. It supersedes the legacy VxWorks taskVarLib facility in kernel mode and the tlsLib facility in user mode, neither of which is available in an SMP environment.

Wind River Workbench with VxWorks SMP

VxWorks UP and SMP share the same Wind River Workbench development suite. The Workbench suite was upgraded as needed to support debugging and analyzing an SMP environment. Workbench 3.0 has the ability to debug the VxWorks SMP kernel and real-time processes (RTPs) in both system or task mode. Workbench enhancements simplify the diagnosis of race conditions and deadlocks, two common issues that arise when developing programs for multicore environments.

In Workbench, VxWorks SMP projects are set up in exactly the same way as they are for the uniprocessor version of VxWorks. Users can choose to develop with either VxWorks UP or SMP on a project-by-project basis. Since not every project or application is suited for SMP, users can choose the OS configuration that best suits their needs. Workbench features Workbench Debugger, System Viewer, Workbench Performance Profiler, Workbench Code Coverage Analyzer, Workbench Memory Analyzer, Workbench Data Monitor, and Workbench Function Tracer were updated so they can be used equally well in SMP as in uniprocessor systems.

Debugging SMP Systems

There is nothing inherently different between SMP and uniprocessor systems when it comes to most debugging and tuning workflows. One workflow that is unique about SMP is that it ensures the maximum throughput and CPU utilization is achieved. For this specific problem, the System Viewer was enhanced to help users understand CPU utilization and concurrency interactions.

The bugs and problems that a developer will see in an SMP system are the same kind of problems that appear in uniprocessor multithreaded systems. Some kinds of bugs are more likely to occur in an SMP system than in a uniprocessor system, especially if the software makes certain assumptions about concurrency, but they are not strictly SMP issues. Debugging SMP is not inherently different than in uniprocessor systems. The following is some guidance about problems that developers are more likely to see in SMP systems and how to find them:

- Problems arising from unprotected or incompletely protected critical sections, such as race conditions and memory corruption, are more likely to occur when threads are running concurrently. This happens because concurrently running software is far more likely to execute such erroneous code than software that runs with little or no concurrency. Memory profiling and System Viewer help catch these errors, same as in uniprocessor systems.
- Priority inversion, for example, a low-priority thread inherits priority over a higher-priority thread waiting on a resource, could be caused when other threads make assumptions about thread priority. System Viewer can help visualize this just as on uniprocessor systems.
- Task-interrupt concurrency issues result from handling interrupts on one core while another core is running a task. Legacy VxWorks applications could assume that task and interrupt processing are mutually exclusive, which is not true in SMP. System Viewer can help visualize this just as on uniprocessor systems.

In general, all the common issues that the run-time analysis tools let you observe and understand work equally well with uniprocessor and SMP systems. It is just as easy to use the tools in a multicore environment as in a uniprocessor environment.

System Viewer

System Viewer contains new analysis capabilities to aid SMP kernel and application developers. For SMP, the System Viewer event graph is annotated with core information providing core awareness to help developers identify and isolate race conditions, deadlocks, and starvation in the same way they would on a uniprocessor system. This can be useful to minimize performance losses. Using System Viewer, users can see parallel activity going on in the system: task-level parallelism, task-interrupt parallelism, and interrupt-interrupt parallelism, all synchronized to a common time source for timing accuracy. This can be useful to visualize the execution characteristics of the application and aids in diagnosing software defects arising from concurrent execution. In addition, System Viewer shows the level of CPU loading on every core in the system. This is useful for performance tuning and effective load balancing across all cores in the system.

VxWorks Simulator

VxWorks Simulator can simulate SMP applications on single-CPU or multi-CPU host machines. Using Simulator, users can perform basic to advanced levels of application porting, simulation, and characterization before moving to real multicore hardware. System Viewer is also available on the host machine to enable the user to collect and visualize SMP data even when run using Simulator. Note, however, that while Simulator provides an accurate view of SMP application behavior, it cannot be used to provide accurate estimates of SMP application performance because the underlying host hardware is likely to be significantly different from the desired target hardware.

Wind River Analysis Tools

Wind River Run-Time Analysis Tools (formerly known as ScopeTools) work with VxWorks SMP just as they do on the uniprocessor version of VxWorks.

Conclusion

In the past, increasing processor clock frequency and shrinking geometries provided the path to huge advances in CPU performance. But of late this strategy is reaching its physical limits. In its quest for ever higher performance, the semiconductor industry has now switched to putting more than one processing core on a single chip.

Multiprocessing isn't new to computer science but its use in device software is a new and growing trend driven by the increasing use of multicore processors. While the promise of higher-performance, rich-featured devices is obvious, multicore technology does pose challenges and complications to software developers. Unlike single-processor systems, multiprocessing systems exhibit true concurrency. This exposes many programming assumptions and latent defects that might never be seen in the single-processor systems for which most legacy software is written. Migrating software from single-processor systems to those with multiple processors is a challenge.

Software speedup from using multiple processors is very dependent on the amount of parallel software execution possible in a system. Depending on the inherent ratio of serial to parallel work in it, not every algorithm will provide notable performance gains from using multiple processors.

Operating systems for multiprocessing systems can be classified as either SMP or AMP in nature. SMP systems rely on hardware symmetry to provide symmetrical software execution across more than one processor. AMP systems use more than one OS in a system consisting of many individual nodes that come together to form a system. Which configuration to use depends on the problem to be solved. SMP systems are relatively easier to program for than AMP, though the nature of some problems make an AMP configuration the better choice for performance.

VxWorks SMP brings symmetric multiprocessing capabilities to the world's leading commercial RTOS. In doing so, it retains the same core, distinguishing real-time characteristics as does VxWorks on a single-processor system. API compatibility and a single set of tools in the same Workbench environment ensure a minimal learning curve and reuse of existing VxWorks applications. The tools support new capabilities for SMP environments. VxWorks SMP is designed to coexist in the same platform environment for VxWorks on a single processor. The VxWorks SMP run-time and tools technology make the only RTOS platform solution with both breadth and depth, backed up by the best support and services teams in the device software industry.

About Wind River

Wind River is the global leader in Device Software Optimization (DSO). Wind River helps develop, run, and manage device software faster, better, at lower cost, and more reliably, to accelerate time-to-market for highly differentiated devices. Only Wind River can deliver on all of your requirements across the DSO life cycle, from development through deployment and management, from hardware optimization through middleware and application integration. To learn more, visit www.windriver.com or call 800-545-WIND.

WIND RIVER

Wind River is the global leader in Device Software Optimization (DSO). We enable companies to develop, run, and manage device software faster, better, at lower cost, and more reliably. www.windriver.com