White Paper

A TimeSys Perspective on the Linux Preemptible Kernel

Version 1.0

A TimeSys Perspective on the Linux Preemptible Kernel

Introduction

One of the most basic aspects of any modern operating system is its ability to support multiple tasks, commonly referred to as processes. Within the heart of the operating system itself, interrupting one process for the benefit of another is known as **preemption**.

In early 2002, Linus Torvalds, the originator of Linux, announced that enhancements had been made to the Linux kernel to support preemption. This set of changes (a patch to the Linux kernel) was originally introduced by MontaVista Software and has most recently been championed by Robert Love, a well-known contributor to the Linux kernel. This patch was incorporated into the source code for the main Linux development kernel and will be available in all versions of the kernel later than version 2.5.4-pre6.

While TimeSys Corporation's enhancements to Linux already include support for kernel preemptibility, like all enhancements to the Linux kernel, the changes made by TimeSys are freely available in source form under the terms of the GNU Public License under which the Linux kernel is being developed.

This white paper explores TimeSys' perspective on the changes made to the mainline Linux development kernel to improve its preemptibility and discusses the implications of these changes for companies that are interested in using Linux for embedded and real-time development projects.

Concurrent Processes

Operating systems support multiple tasks by managing the interaction between the multiple processes running on the machine at any given time. These processes coexist on a computer system by constantly switching from one to the other in order to share resources such as processing power, peripherals, and so on.

For example, after performing calculations, a process may have nothing else to do until additional input is received. Until more data arrives, that process can relinquish the system's CPU, making it available to other processes that have calculations to perform.

Preemption interrupts one process for the benefit of another. The amount of elapsed time between when an event that requires some action by the operating system occurs and the point at which that request is actually serviced is known as latency. The ability to preempt a lower-priority process by a higher priority process is a fundamental mechanism in operating system design that allows a system to reduce latencies for more important or time-sensitive events.

Latency has a variety of different implications depending on the purpose of a particular operating system. Linux was designed as a multi-user system that supports large numbers of processes executing at any given time.

Most desktop computer users have experienced the phenomenon of their desktop computer freezing for a significant amount of time while the system is busy processing a task. While merely annoying in a desktop setting, this kind of unpredictable response time in an embedded environment is unacceptable.

When used as a desktop or server operating system, the goal of Linux is to provide the best general performance for all processes, thereby providing the best average latency for all processes. However, when used in real-time or embedded systems, Linux must be able to guarantee minimum latency for the processes that are critical to one or more specific embedded or real-time applications.

By definition, embedded systems must maintain some interaction with the external world. For an embedded system to function satisfactorily, this interaction must be performed in a timely manner. Latency of response to external events is thus a key way of measuring the performance and acceptability of an embedded operating system.

The new Linux kernel preemption patch decreases latency in the Linux kernel, which is a very positive development for the overall Linux community. On the other hand, the patch does not do enough for the needs of embedded system designers.

The TimeSys Linux kernel already provides more extensive enhancements to the preemptibility of the Linux kernel, as well as other fundamental improvements in Linux scheduling that make the TimeSys Linux kernel a more powerful and predictable solution for using Linux in embedded and real-time development projects. Analyzing the capabilities provided by the new kernel patch and comparing them to the preemptibility enhancements that TimeSys has already made to the Linux kernel will help clarify the specific effects on performance provided by both approaches.

Comparing Preemptibility in the TimeSys and Generic Linux Kernels

The preemptibility enhancements provided by the recent kernel patch provide fundamental yet isolated improvements to the Linux kernel, a major step forward for Linux. The original Linux kernel is not preemptible anywhere. Both TimeSys Linux and the new Linux preemptible kernel begin by leveraging the Symmetric Multi-Processor (SMP) capabilities provided by Linux in order to enhance preemptibility, but take very different approaches under the covers.

The new Linux preemptible kernel enables preemption except when an SMP spinlock would have been invoked. Spinlocks are simple synchronization mechanisms that cause a process that wants access to a locked resource to continuously test whether the resource is available until the existing lock on that resource is released. Spinlocks enable multiple processes to simultaneously (and safely) execute in the kernel on an SMP machine – with at most one on each processor.

The preemptible kernel changes the spinlock to disable preemption because disabling preemption on a uniprocessor system is logically equivalent to the spinlock in an SMP – the process can't be interrupted until the lock is released, which is the point at which the SMP spinlock would have been released. Unfortunately, this approach disables preemption in a large part of the Linux kernel; no kernel thread or any part of the kernel itself can continue execution until preemption is re-enabled (which is done at the point at which the SMP spinlock would have been released).

The TimeSys approach is conceptually similar, but maximizes the number of kernel threads that can continue execution because the TimeSys Linux kernel replaces SMP spinlocks with kernel mutex locks instead of disabling preemption. Kernel mutex locks behave exactly as their name suggests; they implement a mechanism for mutually exclusive access to a shared resource that does not affect other kernel threads and processes that may be executing, unless they need access to the same shared resource.

In the new Linux preemptible kernel, whenever a kernel thread needs exclusive access to a resource, all other kernel threads must wait, resulting in maximum latency that is as long as the duration of the longest non-preemptible interval.

As an analogy, in a city with many traffic lights, using a spinlock or disabling preemption to ensure that only one car is in any intersection at any time would freeze all traffic in a city whenever any car entered any intersection. In contrast, using a mutex would only stop traffic from entering the specific intersection that the car is passing through. Traffic would continue to move everywhere else.

Comparing Latency in the TimeSys and Generic Linux Kernels

As mentioned earlier in this whitepaper, latency has different implications depending on the ways in which Linux is used. On desktop and server systems, decreasing average latency is the right thing to do, because it benefits the overall performance of the system. However, in embedded application development, minimizing the maximum latency of specific Linux processes is much more critical. TimeSys Linux kernel enhancements focus on this aspect of Linux preemptibility.

The following table provides some empirical performance comparisons to highlight the difference between these approaches to preemptibility, showing that adding kernel preemption to Linux helps the average latency of Linux processes but does not reduce the maximum latency of any Linux process. Unfortunately, simply adding preemption doesn't help many of the situations in which the Linux kernel must wait for other operations to complete.

The table below shows that the average preemption latency for the Linux kernel before the preemption patches are applied is around 10,000 microseconds. The new kernel reduces the average preemption latency to about 1,000 microseconds. This improvement will make Linux desktop and server systems noticeably more responsive, but only reduces the average latency; worst-case latencies as long as 100,000 microseconds are still visible.

OPERATING SYSTEM	AVERAGE LATENCY (MICROSECONDS)	MAXIMUM LATENCY (MICROSECONDS)
Standard Linux	<10,000	100,000
Linux with Preemptible Kernel	<1,000	100,000
TimeSys Linux GPL	<50	1,000
TimeSys Linux/Real-Time	<10	51

By contrast, the longest delay observed in TimeSys Linux/Real-Time on similar hardware is about 51 microseconds, while the average latency is less than 10 microseconds. This difference of 2,000 times faster performance when using TimeSys Linux can have enormous benefits to your applications.

Beyond Preemptibility: What Else Does TimeSys Linux Provide?

As shown by the table above, both the Linux preemptible kernel patch and TimeSys Linux provide improvements to the preemptibility and performance of Linux systems, though there is a substantial difference in scale. However, embedded and real-time application developers need more than just increased preemptibility from an operating system.

Contrary to some recent industry comments, the new Linux preemptible kernel does not make Linux into a Real-Time Operating System (RTOS). The key to a real-time operating system is not its average responsiveness, but in the predictability of the system's response to events. A faster, more responsive system is always useful, but the fundamental requirement of a real-time system is that it demonstrates well-known, predictable (and fast) response so that processes that must occur in real-time can be correctly scheduled. If a system takes varying amounts of time to respond to similar events, it's impossible to precisely schedule the execution of any given process or thread.

An RTOS must have certain characteristics in order to provide true predictable performance, including:

- Correct priority handling and associated resource management. RTOSs must allocate resources based on the time constraints placed on processes and threads that are running on the system. Unlike the TimeSys Linux kernel, which provides an unlimited number of priorities and schedules threads in constant time unaffected by the number of threads, the standard Linux kernel uses the existing Linux priority mechanism.

- Better control over interrupt handling. The preemptible patch to the standard Linux kernel does not improve the standard Linux interrupt handlers. The TimeSys kernel features a new, significantly enhanced interrupt scheduling mechanism that considers application and system time constraints and responds appropriately.

These two capabilities are provided by the TimeSys Linux/GPL kernel, at no cost. However, even these enhancements are not sufficient for true real-time performance. TimeSys Linux/Real-Time adds the other requirements of a true Linux RTOS, which include the following:

- Extremely fine resolution clocks and timers. RTOSs need to support clocks and times whose granularity is as close as possible to the system clock cycle. Providing clocks and timers whose resolution is as fine-grained as possible increases the ability of the system to precisely schedule events and take advantage of the execution cycles that are provided by the RTOS. The preemptible kernel patch makes no changes to the clocks and timers used by Linux. By contrast, the TimeSys Linux kernel supports clocks and timers at the resolution of the system clock.

- Protection against priority inversion. RTOSs need integrated mechanisms to protect high-priority processes from having unbounded waiting time on lower-priority processes that hold locks on shared resources. The priority management capabilities of TimeSys Linux automatically provide multiple mechanisms for protection against this type of situation.

TimeSys Linux/GPL gives you significantly improved performance and capabilities today, and also provides a flexible solution for future growth as your project requirements increase. As your project requirements grow, you can then add the proprietary TimeSys Loadable Kernel Modules (LKMs) provided by TimeSys Linux/Real-Time to provide true real-time behavior when necessary.

When used with TimeSys Linux/Real-Time kernel modules, TimeSys Linux is a full-featured RTOS in every sense. As a true Linux kernel, it provides the full Linux application program interface (API) while also providing internal enhancements that guarantee it can satisfy the hard and soft requirements of a truly predictable real-time system.

Summary

The recent announcement of the incorporation of preemptibility enhancements into future versions of the Linux kernel validates the notion that kernel preemptibility is a fundamental underpinning of any embedded operating system. TimeSys Linux already provides a greatly improved implementation of this capability and also satisfies the other fundamental requirements of an embedded or real-time operating system.

TimeSys Linux provides a complete solution for embedded and real-time development that is tested, supported, and available today on all important chip architectures. For more information about using Linux as a real-time operating system, see other white papers from TimeSys Corporation.

Corporate Headquarters
925 Liberty Avenue
6th Floor
Pittsburgh, PA 15222

888.432.TIME
412.232.3250
Fax: 412.232.0655

www.timesys.com

Satisfying Your Embedded Requirements

TimeSys Corporation is a pioneer and leader in Embedded Linux and Java solutions for creating robust, high-performance embedded systems. TimeSys provides complete and certified "out-of-the-box" embedded Linux and cross-platform development tools for a full range of processor architectures and system boards.

Only TimeSys Linux offers revolutionary reservation technologies that can guarantee system response for critical tasks, even in overloaded systems. TimeSys also offers a full spectrum of Linux training and professional embedded development services. For more information on TimeSys, visit www.timesys.com or send email to sales@timesys.com.