

Preemptible Linux: A Reality Check

Kevin Morgan
Vice President, Engineering
MontaVista Software Inc.

The MontaVista Software preemptible kernel patch for x86-based systems improves process level responsiveness in Linux 2.4 by a factor of twenty. The patch is very small (only 1000 lines in length). In the MontaVista Hard Hat™ Linux® 2.0 Professional Edition product, a kernel can be configured with or without preemptibility merely by clicking on a “preemptible kernel” button in the Target Configuration Tool. The preemptible kernel patch has been maintained and reissued through the kernel mailing list for every stable minor revision of Linux 2.4, and will continue to be maintained by MontaVista Software. Our effort is to maintain this patch against the evolving Linux base has been very minimal. By the end of the summer, MontaVista will have preemptible kernel technology available for all supported target architectures, including x86, PowerPC, StrongARM, XScale, MIPS, and SH.

MontaVista has a long-term and ongoing commitment to the availability of a preemptible Linux kernel patch as open source. It is also available as a commercially supported technology in MontaVista’s Hard Hat™ Linux® 2.0 Professional Edition. MontaVista customers (and Linux users in general) can choose to use it if their process responsiveness requirements exceed the responsiveness capabilities of non-preemptible Linux. There are no changes whatsoever to Linux API’s or semantics induced by the preemptible kernel patch. The preemptible kernel patch strictly affects the system performance domain.

Concerns

Recently at the July 2001 meeting of the The Real-time and Embedded Systems Forum (The Open Group) in Austin, Texas, Michael Tiemann (CTO of Red Hat) raised a number of concerns about the MontaVista preemptible kernel technology, including it’s probable longevity, the overall viability of having Linux provide real-time capabilities, and the cost of maintenance of this patch.

MontaVista Software would like to let the facts speak to these concerns.

The Patch Specifics

The preemptible kernel patch modifies the definition (implementation) of a spinlock, changing it from its symmetric multiprocessing (SMP) specific implementation to a preemption lock. In both cases, the locking function acts as a control on reentrancy to a critical section of kernel software. Additionally, the preemptible kernel patch modifies the interrupt handling software to allow rescheduling on return from interrupt if a higher priority process has become executable, even if the interrupted process was running in

Preemptible Linux: A Reality Check

kernel mode (provided the process is not in a critical preemption locked region). Spin unlocks are redefined to return the system to a preemptible state, and check if an immediate context switch is needed. Lastly, the kernel build definition for a uniprocessor target system is modified to include the spinlocks (implemented as preemption locks). Through these four basic changes, the Linux kernel becomes generally preemptible (with short non-preemptible regions corresponding to the spinlocked regions in an SMP kernel). Process level responsiveness is dramatically improved, both on average and in the worst case.

Maintenance Cost and Longevity

All of the changes for the preemptible kernel patch directly leverage the SMP spinlocks, which are themselves fundamental in Linux for symmetric multiprocessing. The code modifications in the preemptible kernel patch are thereby limited to the four areas above. New kernel code that functions correctly in an SMP kernel requires absolutely *no* additional changes in the preemptible kernel patch. Maintenance of the patch against the evolving Linux base is low cost. MontaVista has released a version of the patch to the Linux kernel mailing list for every stable revision of Linux since 2.4.2.

MontaVista is committed to the continued development of this technology across all Hard Hat Linux supported target architectures and boards. The reason for this is simple: an improvement in Linux process level responsiveness is a “must” requirement for many of our current customers, and for many embedded system designers considering the use of Linux as an OS platform. MontaVista customers have a simple choice: enable kernel preemption if needed by the demands of their responsiveness requirement, or continue to use non-preemptible Linux if sufficient as is.

Can Linux Be A “Real-Time Operating System?”

Real-time is defined by application requirements. Hard real-time means an application fails catastrophically if deadline requirements are not met. Soft real-time means an application suffers degradation in quality, but not catastrophic failure, if deadline requirements are not met. Hard and soft real-time are terms that characterize an application. Both terms are time independent. A hard real-time requirement might be relatively slow (i.e., if an event is not serviced within one minute, an engine overheats and breaks down). A soft real-time requirement might be relatively fast (i.e., if a buffer on a networking card is not drained within 200 microseconds of an interrupt event it will overflow, but it will recover from the loss of data). The important point is that soft and hard real-time are attributes of the impact of failures to meet deadlines, and the specific timing requirements are independent of the “hardness” of the requirement.

Linux is capable of meeting a wide variety of real-time requirements, in terms of specific timing needs, addressed by specific levels of software (interrupt service routine versus user application level). Interrupt service routine software is delayed by interrupt off periods in the kernel, and the Linux 2.4 kernel has very short interrupt off timings, with none greater than 60 microseconds on an 800 Mhz Pentium III class system. This level

Preemptible Linux: A Reality Check

of performance meets the vast majority of real-time requirements for interrupt level software. This is particularly true given modern system designs, where extremely fast I/O response requirements tend to be serviced by dedicated hardware in the form of intelligent I/O controllers, dedicated micro-controllers, or custom dedicated hardware. In the rare remaining cases where Linux interrupt off periods cannot be tolerated, RTLinux and RTAI are available. These are sub-kernel technologies providing simple multithreaded interrupt handling environments for driver level software. These environments emulate (virtualize) interrupt management requests from Linux, and thereby reduce the worst case interrupt off timings for the driver software written for these environments from the 60 microsecond level down to approximately 10 microseconds.

Modern real-time environments typically involve substantial control and monitoring software in the real-time control path. Such software resides at the user application level. For example, consider real-time control software written in Java, running on a JVM, an increasingly common design choice. Such a system would never be structured as driver level software. Response requirements for applications are directly tied to the kernel's ability to preempt a running process and switch to a higher priority process (newly awoken) very quickly. The lack of kernel preemption in Linux means that long system calls can delay high priority user process execution for relatively long periods, running into the ten's of milliseconds in a 2.4 kernel. MontaVista's preemption patch today reduces this time down to one to two milliseconds, with further improvements in the future.

Whether an operating system capable of these levels of responsiveness guarantees is considered "real-time" or not is a positioning, not a technical, issue. MontaVista has found this level of improvement in Linux moves it from "problematic" to "very acceptable" for the vast majority of applications that have real-time requirements (soft or hard). MontaVista is completely comfortable with a position that Hard Hat Linux is a real-time operating system.

What About Throughput Impacts?

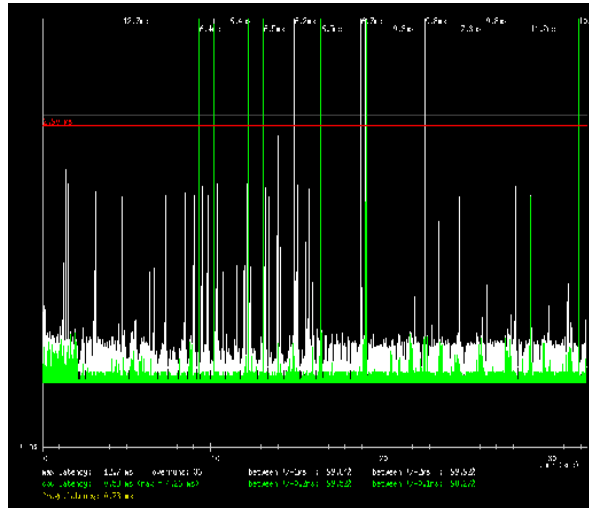
At a simplistic level, changing a uniprocessor kernel to add internal reentrancy management means "more code" and hence "more time". Superficially, a preemptible kernel will have reduced throughput. MontaVista will be providing benchmarks in August and September to help quantify the impacts on throughput of the preemptible kernel changes.

At the heart of the throughput issue is the question of a balanced system design, and the overall design objectives. How important is a responsive Linux? In a world of streaming media, responsiveness is quite important. MontaVista's demonstration of the preemptible kernel doing simple audio processing shows that even a trivial load on non-preemptible Linux causes user process delays that exceed the threshold of the human ear, and audio glitches are heard. With preemption enabled, these delays are vastly reduced, and no audible glitching is heard.

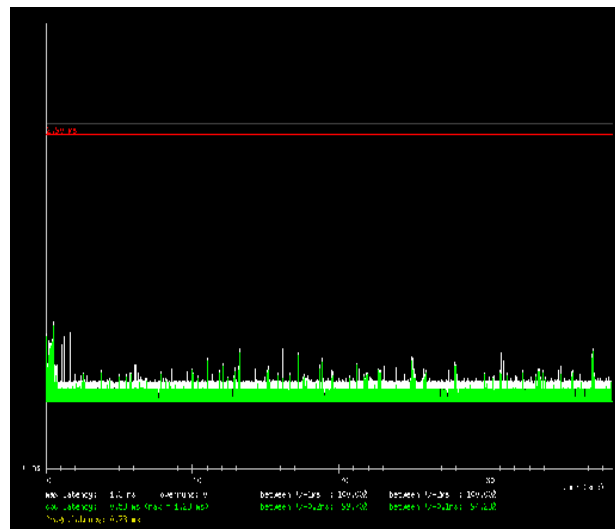
Audio Processing Under Load

The red (horizontal) line represents the audible threshold, or point of real-time failure.

**Standard
Kernel**



**Preemptible
Kernel**



Preemptible Linux: A Reality Check

In order to achieve over 20x improvements in process level responsiveness, what level of throughput loss would be acceptable? MontaVista believes that if throughput loss is less than 2-3%, the cost is outweighed by the improvement in system responsiveness. This tradeoff does not have to be made if every ounce of throughput is critical, and process level responsiveness is not. MontaVista's Hard Hat Linux product allows the user to select preemption or not, as they see fit.

It is important to note that in some application environments, kernel preemption can actually improve throughput. I/O channels can be kept busier over time by prioritizing I/O bound applications higher, and occasionally re-launching them more quickly through kernel preemption. The complete set of computing resources are more highly utilized over time, by being more responsive to the service needs of resources when they go idle.

Inclusion in the Official Linux Kernel Source

MontaVista promotes the inclusion of preemptible kernel technology (as a build option, similar to SMP) in the Linux source code, as provided at kernel.org, starting with the 2.5 kernel base. Our perspective is that this is a fundamental improvement in Linux, which has value to all Linux user communities (desktop, server, and embedded), and should be provided with this central distribution.

Continued development and deployment of preemption technology in Linux will not slow down if the technology is not integrated into the official source tree. Many Linux technologies are available and in widespread use today which are not part of the source code, and which may never be included. This is one of the key benefits of open source; new and innovative technologies can be developed and provided when necessary, with the best getting extensive usage and support.

Independent of Linux 2.5 and beyond, Linux kernel preemption technology is available today as an open source patch, and is offered by MontaVista software as a commercially supported capability in Hard Hat Linux. MontaVista is committed to the long-term availability and support of this capability in Hard Hat Linux, and committed to bringing leading edge Linux advancements across all of MontaVista's target platforms.

Why Are Some Opposed to a Preemptible Kernel?

The Linux kernel community is large and diverse. In every technical area, there is lively discussion and debate. Preemptible kernel technology is no different.

Some oppose a preemptible kernel because of throughput concerns. MontaVista's perspective on this is covered above. Others oppose preemptibility because of concerns about growing complexity in the kernel. This argument is specious, because the preemption approach takes advantage of already required and in place SMP locking. No additional complexity is created. All Linux kernel engineering must already take into account SMP requirements. Some oppose continued refinement of SMP locking to achieve better SMP scaling (on higher way SMP systems); such refinement has the

Preemptible Linux: A Reality Check

beneficial side effect of also reducing preemption off periods in a preemptible kernel. Preemptibility on 2.4 already provides dramatic improvements in user process responsiveness, and while further improvement would be beneficial, the current level of improvement is already of tremendous value. Hence, the pro's and con's of improving SMP scaling in Linux can be debated relatively independently of preemptibility improvement opportunities.

Finally, some are opposed to preemptible kernel technology for the simple reason that they are business people, and would rather have customers utilize a product other than Linux to meet real-time requirements. eCos, RTLinux, RTAI, LynxOS, VxWorks, QNX, and others are all non-Linux products around which there may exist such individuals.

Recently, the FreeBSD developers have shared their intent to transform the FreeBSD kernel into a completely preemptible kernel, utilizing similar technology to that developed for Linux by MontaVista Software. The FreeBSD developers appear to understand the criticality of preemption in a modern, competitive kernel.

Next Steps for Preemptible Linux

MontaVista is continuing to enhance preemptible Linux. Availability of the technology as a commercially supported product from MontaVista on all embedded target architectures is next. Providing an alternative semaphore implementation that utilizes priority inheritance is an improvement under design. Continuing to refine long spinlock held regions is an ongoing effort, both within and outside of MontaVista. Characterizing throughput impacts (positive and negative) on a number of workloads is under progress and will be shared shortly. And of course, supporting numerous commercial customers in their utilization of preemptible Hard Hat Linux is a core MontaVista task. MontaVista expects a number of application success stories over the course of the next year as this technology is widely designed in and deployed by embedded system product organizations.

Responsibility to the Community

Mr. Tiemann commented at The Open Group Real-Time Forum that Red Hat has “a responsibility to the community”, and that the MontaVista preemptible kernel technology is not “a go-forward solution.” It appears that Mr. Tiemann wishes to preempt (no pun intended) the normal process of the open source bazaar, by dictating what are and are not “go-forward solutions” in advance of the natural processes of review, usage, and feedback.

MontaVista also has responsibilities, to the open source and Linux communities, as well as to the embedded system product development communities. We have a responsibility to innovate and release our innovations early and often, for public comment and contribution. We have a corporate responsibility to do our best to enable Linux to be a superior operating system platform for embedded system design and implementation.

Preemptible Linux: A Reality Check

Our customers find extensive value in our exercise of that responsibility, through the delivery of such product technologies as a preemptible Linux 2.4 kernel.

As long time supporters of the Open Source community we are pleased that the embedded systems marketplace, and the Linux community itself, will decide the relative merits of MontaVista Software preemptible kernel technology.

***Kevin Morgan** is Vice President, Engineering at MontaVista Software. He has 20 years of experience developing embedded and real-time computer systems for Hewlett-Packard Co. Experienced in operating systems and development. Kevin was a member of the HP 1000 computer software design team. While at Hewlett-Packard, he worked as an engineer, project manager and section manager spanning the development of five operating systems. Most recently serving as HP-UX Operating System Laboratory Manager, Kevin was responsible for overall HP-UX release planning, execution and delivery for Hewlett-Packard server computers.*

Linux is a registered trademark of Linus Torvalds. Hard Hat is a trademark of MontaVista Software Inc. All other names mentioned are trademarks, registered trademarks or service marks of their respective companies