Real–Time Embedded Linux Study on ARM Cortex-A8

The latest versions of the Linux kernel enhanced with a Real-Time patch show fast response times with latencies below 58µsec, a response some 40-times better on system events than a standard pre-emptive Linux kernel. Real-Time Linux is suitable for many applications including, communication stacks, IP-PBXs, and industrial automation.

The Linux kernel is designed to deliver first class performance for personal computers and servers. However, over the last few years, another market where the Linux kernel has shown its potential is in Real-Time Embedded systems.

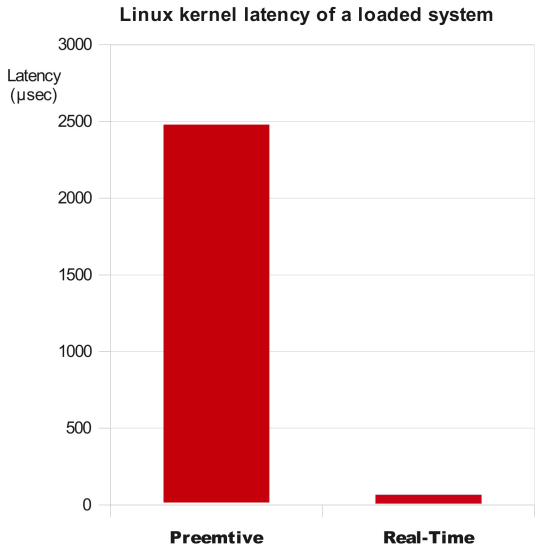
The key characteristic of a Real-Time Operating System is determinism, something that implies quick and bounded response to certain events, and accurate scheduling of periodic tasks. The standard Linux kernel lacks a framework that demonstrates the Real-Time performance required by hard real-time applications.

The Linux community has shown solid interest in addressing the lack of Real-Time properties in the Linux kernel, and has developed a number of enhancements making the response time for events in the Linux kernel suitable for Real-Time applications. Real-Time Linux improvements are distributed from kernel.org, the Linux source code repository.

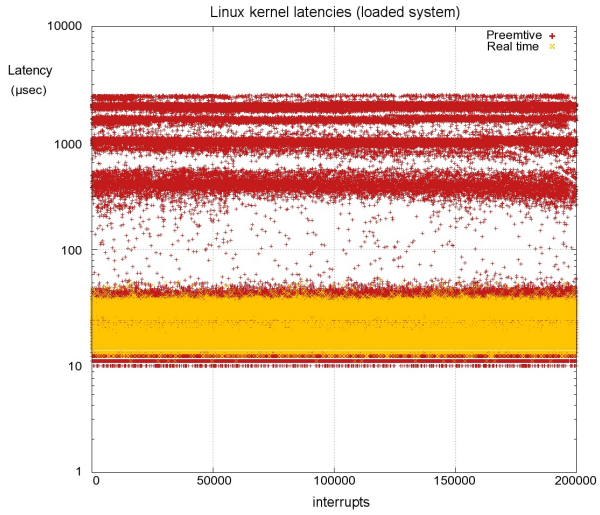
In benchmarking the Linux kernel, we collected latency data from a Freescale MX515 development board. The MX515 uses a Cortex-A8 processor from ARM running at 800MHz. The version of the Linux kernel in our tests was 2.6.31.12, modified with a Real-Time patch. We ported the Real-Time patch to work on Cortex-A8 and also made a number of changes to support the MX515 architecture. In order to examine and compare the response times for events, we used two different kernel configurations; one with the Real-Time features enabled and one with a standard Linux pre-emption. We generated a set of hardware timing events and captured the latencies from the Real-Time applications running in the user space.

The results:

The results from our tests are very encouraging, with the Real-Time kernel performing latencies up to 58µsec maximum, whereas the standard pre-emptive Linux configuration brings about 2465µsec worst-case response latency; 40-times worse than the latency measured in the Real-Time kernel.

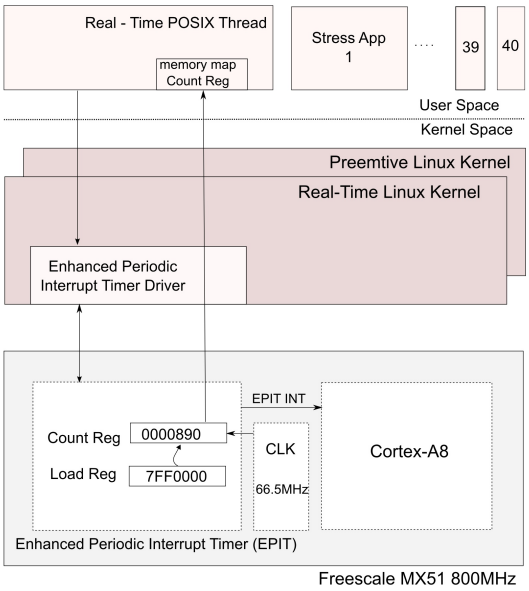


The graph below shows latencies (on a logarithmic scale) for a Real-Time Linux versus a pre-emptive one. In the Real-Time system it is clear, that the event response latencies are quite consistently within the region of 12 to 60 µsec.

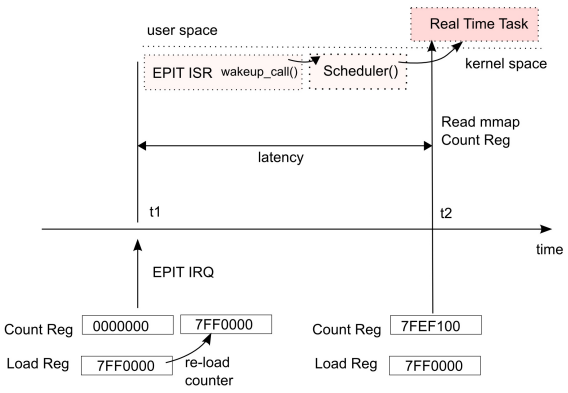


The test-bench:

Our test-bench for measuring latencies used a periodic interrupt timer (from the MX515 configured as free-running timer). The interrupt latency was measured from the point in time when an interrupt happened to the time when a real-time task was scheduled to run.



The tests were run for a day on a fully-loaded system collecting millions of samples. In order to load the system the “Stress” utility program was used. The Stress utility takes a number of parameters and loads the CPU, Input/Output, and the virtual memory.



What does the Linux Real-Time patch do?

The Real-Time patch converts the Linux kernel into a fully pre-emptive OS in the following manner:

- Implements POSIX high resolution timers.
- Re-implementation of locking primitives in the kernel.
- Implements priority inheritance for spinlocks and semaphores.
- Converts interrupt handlers into pre-emptible kernel threads.
- Converts the old Linux timer API for high resolution kernel timers support.

Design your Real-Time application using POSIX API:

Developing applications with Real-Time Linux may include the following steps:
Partition your system into two different domains; Real-Time and non Real-Time. Work out the time constraints of those domains and calculate the MIPS required. Allow 30% spare processing room, and adjust the CPU clocks. Develop your Linux device drivers efficiently using rt-mutexes and the Linux kernel pre-emptible infrastructure. Measure driver's performance. Design and develop your Real-Time domain applications using POSIX Real-Time scheduling policies in user space. Follow up with your non-real-time applications. Finally profile, and benchmark your system.

Conclusions:

The Real-Time Linux kernel has become mature and robust over the last few years. It has been extensively tested on X86 architectures and is a strong fit to a range of applications where fast and deterministic response is required. Real-Time Linux is gradually getting into other architectures such as ARM, and PPC. Porting the Real-Time patch to work on the MX515 with Cortex-A8, and developing a test-bench with tools and utilities was quite a valuable exercise for us. Our findings show that Linux kernel behaviour changes dramatically when the Real-Time patch is enabled in Linux, with interrupt response time reduced down to a few microseconds from 2.5 milliseconds. The Linux high resolution timers also allow scheduling Real-Time tasks in the user space quite accurately. The use of the standard POSIX APIs to develop applications in user space accelerates development time and enables portability. Multiple – core ARM architectures, such as Cortex-A9, may bring even better performance as Real-Time Linux supports multiple cores, leading to the ability to group Real-Time applications to run on CPU-A, and non Real-Time on CPU-B, thus increasing both responsiveness and throughput.

Consultancy services:

Hedera Innovations provides you with Open-Source software development and hardware design services over a wide range of application spaces. Our specialist services allow you to differentiate, increase functionality and reduce cost in your project with cutting-edge Open-Source software components plus a custom embedded system design service.

Hedera Innovations Ltd
23 Cambridge Science Park
Cambridge
CB4 0EY, UK
Tel.: +44 (0) 1223 437142
Fax: +44 (0) 1223 437143
Email: info@hedera-innovations.com
Web: www.hedera-innovations.com