FSMLabs White Paper

The RTLinux® embedded enterprise versus the Linux "RT" patches.

Abstract:

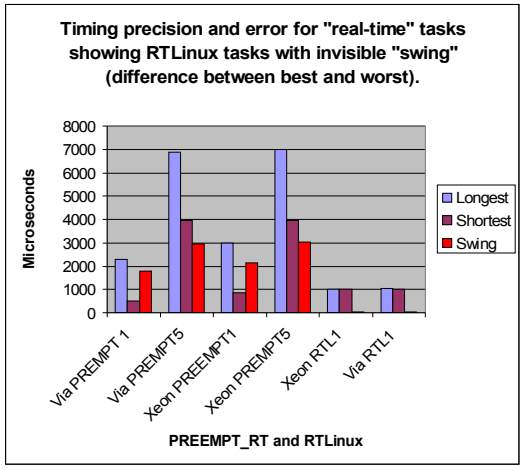
FSMLabs patented "decoupled real-time" continues to hold a sustained performance, reliability, and programmability advantage over the result of ten years of non-modular Linux real-time efforts.

Introduction and performance summary

The patented RTCore dual kernel model found in RTLinux and in RTCore BSD has been used in production for more than a decade and is incorporated in products such as Infineon's single core cell-phone handset (the first such system to be built), Samsung Heavy Industry's spider robot for welding ship hulls, Raytheon's missile simulators, Juniper Networks J-series routers,

On machine with best Preempt_RT Performance – two threads test.		
Period (micro-seconds)	Time to failure	
	PREEMPT_RT	RTLinux
200	20 minutes	Never
1000	50 minutes	Never

and Hollywood animatronic robots like those in the film Charlotte's web. RTCore runs an enterprise operating system, like Linux or BSD, as a client instead of following a much older technical approach which is to try to integrate real-time and enterprise in a "mixed use kernel" (MUK).



Over the last six years, there have been repeated efforts to produce a mixed use Linux, something like the old Irix operating system of Silicon Graphics. Victory in this quest has been announced many times. For example, Timesys and Montavista argued bitterly over which one had first put real-time into MUK Linux in August 2000 (see this press release for example). With a recent spate of new announcements about how this time MUK Linux is really, really, real-time, it's worth making a comparison.

We tested Linux PREEMPT_RT for Linux 2.6.17 on multiple machines. Bottom line: a nice, if delicate, system for playing audio and video on programmer workstations, but not useful for mission critical projects. Limitations include high cost in enterprise performance (for example the MySQL benchmarks show between 10% and 35% slowdown), imprecise scheduling, and unclear API limitations. For example, on the best machine tested, a lightly loaded 1GHz Via, a pair of one kilohertz tasks that were supposed to alternate operation

tangled up within 50 minutes. In this test, two "real-time" threads increment a shared variable and failure occurs when one increments twice in a row – showing that scheduling failed completely. The same test, running the same code ran for a week at a 5 times shorter period under RTLinux with no error. We couldn't find a combination of configuration points that would get good performance on a standard Dell Xeon Workstation and just booting the system on a 64bit multi-core was impossible due to fatal preemption lock errors.

We did see surprising improvements in PREEMPT_RT performance in best case compared with previous versions of this patch. We looked deeper and saw that PREEMPT_RT is starting to look very familiar – in fact, the developers must have realized the impossibility of success in a pure MUK environment because they have adapted many of the basic parts of the RTLinux design from 1995. Linux MUK developers are welcome to use the RTLinux method under the terms of the Open RTLinux Patent License, which is royalty free as long as all code using the method is released under the GPL as specified in the license.

Lets put some racing wheels on these babies and see how well they corner

Motivation for RTCore

RTCore's design was inspired by the difficulties of the MUK approach. While it might seem obvious that you can just add real-time to an enterprise OS by improving some processes priorities, in practice, it's like putting roller skates on an elephant. In spite of an impressive engineering effort, including work by some of the best operating systems development teams ever assembled, IRIX real-time was limited and required increasingly expensive hardware (not a disadvantage for companies like SGI that sell expensive hardware). Furthermore, application programmers had to navigate an increasingly complex set of rules about what calls were permissible in what contexts.

Wind River's VxWorks started as a simple real-time OS (RTOS) and had enterprise features added later and it suffers from a problem of interference between real-time and non-real-time components. This problem has lead to two near disasters on VxWorks controlled Mars Missions, problems that were not uncovered during extensive testing but showed up during the mission. One problem was system failure due to an interaction of a flash file system with real-time threads. Working around this problem required some high risk in-space reprogramming.

The first versions of RTLinux, running on low cost PCs were able

to significantly outperform mainframe SGI systems on simple real-time tasks and they immediately offered a rich enterprise programming environment from the “decoupled” client Linux kernel. As commodity PC and embedded computer hardware has improved, and as FSMLabs has built up the technology, RTLinux and RTCore BSD systems have been able to apply this performance advantage to more sophisticated real-time control applications. Now rack-mounted 16 core AMD Opteron based RTLinux servers are being used to directly replace SGI mainframes on graphics intensive simulators and RTCore is running deeply embedded low power systems.

The POSIX threads API of RTCore real-time is familiar and the rules for what goes in real-time and what does not are clear. Our experience and that of many of our customers is that the dual kernel model reduces configuration time and helps programmers avoid surprises

In terms of engineering payoff for effort, the dual kernel approach just works better.

To understand why the MUK path is so difficult, think about the difference between long haul freight delivery powered by trucks and a motor-cycle based urban courier service. What standard enterprise class operating systems do is similar to long haul freight delivery. The important thing is to get large volumes of cargo from one place to another in the shortest economically reasonable period of time. A long haul trucker can speed up going down hill and slow down going uphill – the final time is what counts and those slowdowns can be “amortized” by faster periods. City couriers have much more rigorous timing requirements – if they are late delivering a package to one customer, they cannot make up for it by delivering the next parcel early. Imagine trying to retrofit a long haul freight service to also provide urban courier service. The methods and routes and vehicles that are well suited to the one service are uneconomical for the other. In brief, the technical argument for MUK is that drivers of the big trucks don’t want to learn to ride motorcycles or drive city streets so the trucks should be able to spin around corners on two wheels. As you can imagine, making trucks like that is a complex undertaking.

Summary and conclusion

- RTLinux and RTCore BSD dual kernel technology is proven and production validated aimed at mission critical applications and validated in the field on everything from memory constrained low power ARM7s to sixteen-core AMD Opteron-64's.
- Linux "PREEMPT" real-time is a continuing experiment aimed at audio and video playing with unreliable results and a detrimental affect on "enterprise" performance.