Real-Time Linux

Shane Tucci, Ethan Blatti

I. Abstract

This document is meant to give a comparative study on the different implementations of real-time Linux. It will detail the various methods of real-time Linux and how there are similar and different ways of having real-time on Linux. There is no single correct method in to implementing real-time systems on a Linux Machine. At standard, Linux will never be a real-time machine but with the help of the following methods there are ways for a Linux machine to obtain a real-time status.

Keywords – real-time, Linux, preempt, spinlocks, processor affinity

II. Introduction

Linux is not a real-time operating system. There are a few reasons as to why Linux is not a real-time operating system. Applications run in “user space”, and all hardware interactions take place in the “kernel space”. Traditionally, the Linux kernel will only allow one process to preempt another only under certain circumstances: When the CPU is running user-mode code, or when kernel code returns from a system call or an interrupt back to user space, or when kernel code blocks on a mutex, or explicitly yields control to another process. If kernel code is executing when some event takes place that requires a high priority thread to start executing, the high priority thread cannot preempt the running kernel code, until the kernel code explicitly yields control. This could cause a latency of up to hundreds of milliseconds. So, Linux cannot be defined as a real-time operating system on its own. However, there are implementations that can make Linux in to a real-time operating system. In this paper, we will discuss a few possible implementations in making Linux a real-time operating system. To do this, you can turn on the CONFIG_PREEMPT -RT kernel patch, RTLinux, and Xilinx Real-Time Linux approach.

III. CONFIG_PREEMPT_RT Kernel Patch

The CONFIG_PREEMPT_RT kernel patch is maintained by a group of small developers, that allows nearly all the kernel to be preempted. The key point of the patch is to reduce the amount of kernel code that is non-preemptible, also minimizing the amount of code that must be changed to provide this added preempt ability. There are a handful of features the PREEMPT_RT patch gives. The first feature is “preemptible critical sections”. For this feature, normal spinlocks are now preemptible. Normally, what occurs is that when a task needs a resource, it obtains the “lock” for it while using that resource. If another task needs that resource, but that resource is in use, that task essentially “spins” in place, waiting for the resource to be available. This is caused because Linux does not support multitasking. So, allowing the critical section to be preempted, this makes the spinlocks be able to be interrupted and used by another task if the priority is more critical. One just needs to be cautious as now that the critical section can be preempted, some tasks will not run on a single-CPU but may get moved to a different CPU since it is preemptible. Another feature is making interrupt handlers preemptible. Any code that interacts with an interrupt handler must be prepared to deal with that interrupt running concurrently on another CPU. A third feature of the PREEPMT_RT patch is priority inheritance. Priority inheritance is when a high-priority task gives a low-priority task its’ high priority. This

idea is to help with suppressing preemption. Finally, a feature that helps make Linux real-time is the latency-reduction measures that the patch has. Its purpose is to reduce scheduling or interrupt latency. Latency is the time it takes data to get from one destination to another. Decreasing latency helps processes run faster and more efficiently. Overall, the PREEMPT_RT kernel patch is a nicely built patch that can allow Linux to run in real-time. The patch is always being updated and worked on so one can expect some changes in the patch over time, but what was described in this section was the basic outline of what the patch does and how it allows Linux to run in real-time. However, this is not the only method to letting Linux run in real-time.

IV. RTLinux

RTLinux is a hard-real-time operating system microkernel that runs the Linux operating system in a full preemptive process. Having the hard-real-time property allows Linux to control time-sensitive operations. RTLinux was developed by Victor Yodaiken, Michael Barabanov, Cort Dougan and others at the New Mexico Institute of Mining and Technology and then as a commercial product at FSMLabs. Wind River Systems acquired FSMLabs embedded technology in February 2007 and made a version available as Wind River Real-Time Core for Wind River Linux. As of August 2011, Wind River has discontinued the Wind River Real-Time Core product line, effectively ending commercial support for the RTLinux product. The key RTLinux design objective was to add hard real-time capabilities to a commodity operating system to facilitate the development of complex control programs with both capabilities. RTLinux was designed to share a computing device between a real-time and non-real-time operating system so that the real-time operating system could never be blocked from execution by the non-real-time operating system and components running in the two different environments could easily share data. For implementing RTLinux, RTLinux provides the capability of running special real-time tasks and interrupt handlers on the same machine as standard Linux. These tasks and handlers execute when they need to execute no matter what Linux is doing. Overall, RTLinux is a small threaded environment for the real-time tasks to work in, inside a standard Linux machine.

V. Xilinx Real-Time Linux

Xilinx has an existing kernel to allow Linux to run in real-time, where low latency and preemption enhancements are made to the Linux kernel, or through an Asymmetric Multi Processing (AMP) approach where a real-time operating system can pre-empt the whole Linux OS, or it has the ability to run independently on one of the two processor cores. There are several different companies that Xilinx is partnered with and some of the different companies actually use different methods in making Linux a real-time operating system.

The first method is a native Linux-API based solution to making Linux real-time. This method to improving real-time performance on Linux is by utilizing pre-existing features on Linux or modifying the kernel patch that address low-latency and pre-emtibility. By doing this, you can refer to the “CONFIG_PREEMPT_RT Kernel Patch” section of this white paper. Xilinx partners who include Native-Linux real-time capabilities in their Linux machines are: Denx, Enea, Mentor, MontaVista, SYSGO, and Wind River.

The next method is SMP Affinity which is an open source solution to improving real-time processes on a Linux machine. This approach uses existing features and capabilities of the Xilinx Git kernel in order to improve system-level responsiveness. Some of the benefits is that this method can allow you to

assign tasks to specific CPU according to their PID. This is also called processor affinity.

Enea Light Weight Run Time is a commercial solution to this problem. It offers user-space implementation of scheduling, message passing, and resource management. This exposes a Linux API to the developer for support of real-time tasks. This method requires fewer kernel patches than preempt_rt and should still give low-latency just like the preempt_rt patch.

Asymmetric Multi Processing (AMP) is another real-time solution for Linux. This method is where a separate real-time operating system runs on one of the processors. This redirects one of the ores to run solely real-time tasks. This is also called a dual-kernel method. Not all CPUs are treated equally here. Xilinx partners that offer this dual-kernel method are Denx, Sierraware, SYSGO, and Xilinx PetaLinux.

The last solution to this real-time Linux solution is using Xenomai. Xenomai is a dual-kernel open source solution. This is very similar to AMP but with Xenomai, real-time tasks can be deployed within Linux user-space, regardless that it runs on a separate kernel. One other benefit to using Xenomai is that Xenomai offers what they call "skins" and these skins are essentially just recreations of some traditional APIs for some commercial real-time operating systems. The original vendor does not support the skins of APIs. Linux is also protected from being corrupted with the RTOS. Xenomai is probably one of the most versatile methods of making Linux a real-time operating system. Xenomai is supported by Denx, and SYSGO.

Overall, Xilinx is a very helpful source in getting support and solutions for real-time on Linux. They help outline the different methods to creating a real-time solution for Linux.

VI. Conclusion

What makes Linux not real-time is the fact that priority is not so much a factor, but its' resources are first come first serve. When a resource is being used, it is locked and cannot be used by another task until the task using the resource is complete. With this causing latency times of up to hundreds of milliseconds, Linux cannot be defined as a real-time operating system. But what was stated in this paper are the methods and ideas behind the solutions in to making Linux a real-time operating system.

Each of the available approaches to improve the real-time performance of Linux also bring various trade-offs related to programming API, kernel porting, divergence from standard Linux source tree, modification of device drivers, and IO/throughput. As stated in this paper, each approach handles real-time in a different manner. Some of these methods require more work and possibly a better background or understanding of Linux itself. Without first having a knowledge in the Linux operating system, it would be challenging to comprehend and implement these real-time patches if one did not.

It is hard to say if any one method is better than the others, but what can be said is that it is possible in multiple ways to create a real-time solution for Linux. For one to obtain these solutions is easy. They are all over the web where anyone with an interest in using Linux as a real-time operating system. These solutions are still being developed and worked on, so it would be no surprise if one came across any sorts of errors. But I believe that some time in the future, there will be a better method or a new method that can seamlessly make Linux a real-time operating system. Other than that Linux could potentially even release a version of Linux with real-time capabilities built in to their hardware already. In the technological industry, the game changes within just years so the possibilities are endless.

VII. References

Dougan, Cort (2004), "Precision and predictability for Linux and RTLinuxPro", Dr. Dobbs Journal, February 1, 2004

A Real-Time Linux. Victor Yodaiken and Michael Barabanov, New Mexico Institute of Technology.

RTLinux HOWTO. Dinil Divakaran, Copyright (C)2002 Dinil Divakaran.

Intro to Real-Time Linux for Embedded Developers. Linux Foundation, Linux Foundation, March 21, 2013.

Real-Time Linux. Xilinx, Xilinx.com

Release news – Xenomai

Bell, C. Gordon, Mudge, J. Craig, McNamara John E. "The PDP-10 Family". (1979). Part V of Computer Engineering: A DEC View of Hardware Systems Design. Digital Equipment Corp.