SKYCOMPUTERS
THE MULTICOMPUTING EXPERTS

A SKY Computers White Paper

Real Time Linux

SKY Computers, Inc.
8 Centennial Drive.
Peabody, MA 01960
978 977 3000
www.skycomputers.com

Linux was originally written as a general purpose operating system without any consideration for real time applications. Recently Linux has become attractive to the real time community due to its low cost and open standards. In order to make it more practical for the real time community, patches have been written to effect things such as interrupt latency and context switch. These patches are public domain and are becoming part of the main Linux tree.

This paper talks about one of these patches, the preemptive scheduler patch, and its effect on the interrupt latency of a Linux system. The patch reduces the measured interrupt latency of the system making it more appropriate for real time applications.

Background

While applications run, some are going to reach a point where they wait for an interrupt to occur. That is, some piece of data must be received or something must occur before the application can proceed. The interrupt latency is a measure of the amount of time needed for the application to respond to the interrupt and proceed. If an application responds to many interrupts, it will experience many latencies. In real-time systems, the engineer needs to design to the worst case so the largest of these interrupts must be known.

Linux is both a multiprocess and multithread operating system. The numbers presented in this paper represent the results of two different processes running so the interrupt latency numbers represent the time to both respond to the interrupt and change processes. This change of process requires cache data of the operating process to be stored and cache data of the blocked process to be reloaded. Latency sensitive and security insensitive applications can be written in a single process multithread fashion which would produce smaller latency numbers because data does not need to be moved. Many operating systems only offer the single process multithread option and will automatically offer smaller latency times. Because we chose to run an open source benchmark which could only do things in a multiprocess fashion, these are the numbers which are presented. Future work will include a single process multithread benchmark.

For this work, interrupt latency is measured with an open benchmark called “Realfeel”, written by Mark Hahn. Realfeel issues periodic interrupts and measures the time needed for the computer to respond to these interrupts. Response time will vary from interrupt to interrupt. Realfeel will measure these interrupts and produce a histogram by putting the measurements into bins. The measurement focused on is not the histogram itself but the largest interrupt latency measurement. The performance of an operating system may depend on the average interrupt latency for some applications but real-time applications are more dependent on the largest interrupt latency. The largest interrupt latency is a prediction of the worst case scenario. For many of these applications, if the latency were over the limit one time, this would result in a complete failure of the system so the purpose of the benchmark is to give the latency which would never be exceeded.

In a real application, the system will not be dormant when the interrupt is issued. Some kind of background operation will be occurring with the operating system or another application will be in process. The benchmark is run under various load conditions ranging from processor computation to system and network operations. This work looks at the maximum under three load conditions which are run as script files on the system. The “Find” script searches the hard drive for a file by using the UNIX find command. The “Launch” script runs a script which continually launches a trivial executable program and prints a small amount of text to the screen. The “File move” script continually copies two large files on the disk over each other. To come up with truly worst case numbers, we made an effort subject to the worst load as possible. The loads are run continuously and are probably worse than anything the system would see in actual use.

The application is run with two kernels; one is the Linux 2.4.19 kernel from the Galileo tree and the other is the same kernel with the “preemptive scheduler” patch applied. The preemptive scheduler patch was created by MontaVista software and maintained by Robert Love. The patch is public domain and expected to be included in the next version of the standard kernel. The intent of the work is to examine the effect of the patch on the interrupt latency time.

The benchmark was run on a Power PC operating at 312.5 MHz. The system controller chip was a Marvell Discovery operating at 125 MHz. The system was connected to the network and had no disk drive.

Results

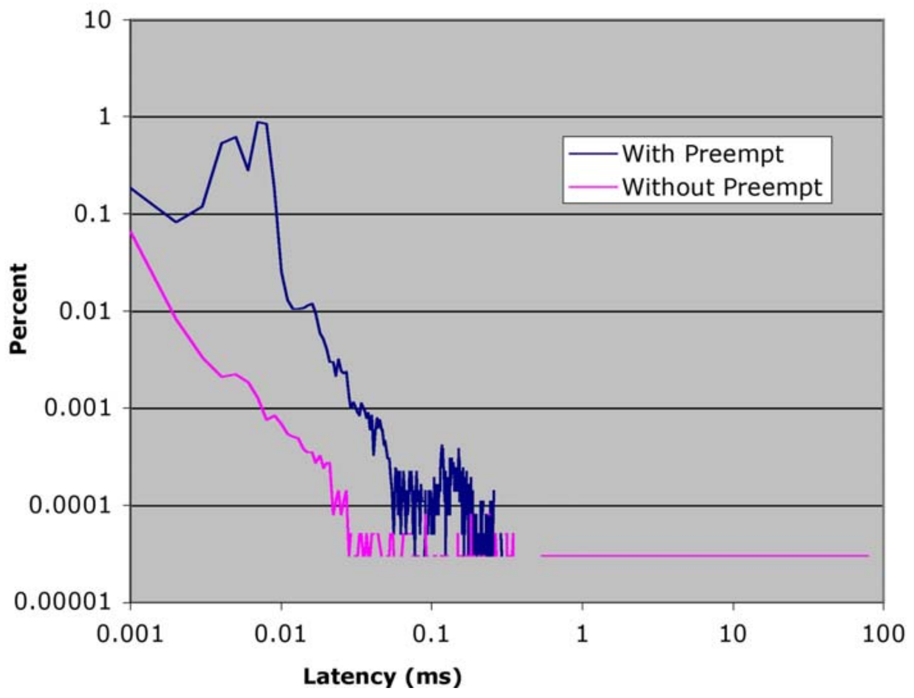
Results are presented for both kernels run with all three load scripts run in background. These results are presented in Table 1. Results are stored in the form of a histogram. Because the purpose of this work is to examine the potential to use Linux in a real time environment, only the largest interrupt latency numbers are reported.

Table 1
Largest interrupt latency time measured over 3.5 million samples

Script	Without preemptive kernel patch	With preemptive kernel patch
Find script	78.51 ms	0.48 ms
Launch script	0.61 ms	0.41 ms
File move script	0.62 ms	0.31 ms

The results show that the preemptive kernel patch reduces the largest interrupt latency for all load scripts. The patch reduced the maximum latency for the launch script and file move script by about 50% and reduced the maximum latency for the find script by a factor of more than 100. This difference could easily mean the difference between a real time application succeeding and failing. The maximum interrupt latency will always vary with application and there is no standard telling what number separates real time from non real time.

While the focus is on the worst case latencies, it is also useful to examine the histogram of latencies shown in figure 1. The plot is log log and the bulk of the latencies occur within 50 microseconds on the plot with and without the preemptive kernel patch. The numbers with the patch are actually slightly less than the numbers without the patch until the very end of the plot. The major effect of the patch is the end or tail of the plot. The patch essentially shortens the tail and lowers the largest interrupt latency times.



Conclusions

The preemptive kernel patch can dramatically reduce interrupt latency times and thereby make Linux more of a real-time operating system. The move to an open operating system such as Linux offers advantages to both vendors and users. It lowers costs for vendors while offering more portability for users allowing them to avoid “lock in” for programs.

The numbers presented represent a benchmark run in multiprocess mode and should only be compared to other benchmarks run in multiprocess modes. In the future, single process multithread benchmark numbers will be measured.