Running Realtime Scheduling Analysis

Ilse Monserrat Sánchez Genereux and Mark Octavio Rivera Acosta

November 21, 2016

Abstract

The Linux kernel controls the way tasks (or processes) are managed in the running system. The task scheduler, sometimes called process scheduler, is the part of the kernel that decides which task to run next. In this project its analyzed the behavior of scheduler by changing a default value from the runtime scheduling. The default value is 950000μs, or 0.95 seconds for the sched_rt_runtime_us or scheduler realtime running variable. Meaning that 5% of the CPU time is reserved for processes that don't run under a real-time or deadline scheduling policy. This value in this file specifies how much of the "period" time can be used by all real-time and deadline scheduled processes on the system. The AIO-Stress which shows the obtained results in the different tests is an a-synchronous I/O benchmark created by SuSE which is is a German Linux distribution provider and business unit of Novell, Inc.

1 Introduction

The Linux kernel controls the way tasks (or processes) are managed in the running system. The task scheduler, sometimes called process scheduler, is the part of the kernel that decides which task to run next. It is one of the core components of a multitasking operating system (such as Linux), being responsible for best utilizing system resources to guarantee that multiple tasks are being executed simultaneously.

A typical real-time task is composed of a repetition of computation phases (task instances, or jobs) which are activated on a periodic or sporadic fashion.

The utilization of a real-time task is defined as the ratio between its WCET and its period (or minimum inter-arrival time), and represents the fraction of CPU time needed to execute the task.

In this project we will analyze the behavior of scheduler changing the default value of run time scheduling in terms of how many megabits per second change depending on how many microsecond are specified.

2 Theoretical Framework

2.1 The Kernel

The kernel is the essential center of a computer operating system, the core that provides basic services for all other parts of the operating system. A synonym is nucleus. A kernel can be contrasted with a shell, the outermost part of an operating system that interacts with user commands. Kernel and shell are terms used more frequently in Unix operating systems than in IBM mainframe or Microsoft Windows systems.

2.2 The Scheduler

The scheduler is responsible for keeping the CPUs in the system busy. The Linux scheduler implements a number of scheduling policies, which determine when and for how long a thread runs on a particular CPU core. Scheduling policies are divided into two major categories:

1. Realtime policies
    - SCHED_FIFO
    - SCHED_RR
2. Normal Policies
    - SCHED_OTHER
    - SCHED_BATCH
    - SCHED_IDLE

In this project we are dedicated to analyze the behavior of the Realtime policies by changing parameters in /proc/sys/kernel/sched_rt_period_us which is described later.

2.2.1 Realtime scheduling policies

Realtime threads are scheduled first, and normal threads are scheduled after all realtime threads have been scheduled. The realtime policies are used for time-critical tasks that must complete without interruptions.

SCHED_FIFO

This policy is also referred to as static priority scheduling, because it defines a fixed priority (between 1 and 99) for each thread. The scheduler scans a list of SCHED_FIFO threads in priority order and schedules the highest priority thread that is ready to run. This thread runs until it blocks, exits, or is preempted by a higher priority thread that is ready to run. Even the lowest priority realtime thread will be scheduled ahead of any thread with a non-realtime policy; if only one realtime thread exists, the SCHED_FIFO priority value does not matter.

In the Linux kernel, the SCHED_FIFO policy includes a bandwidth cap mechanism. This protects realtime application programmers from realtime tasks that might monopolize the CPU. This mechanism can be adjusted through the following /proc file system parameters:

- **/proc/sys/kernel/sched_rt_period_us**: Defines the time period to be considered one hundred percent of CPU bandwidth, in microseconds ('us' being the closest equivalent to 'μs' in plain text). The default value is 1000000μs, or 1 second.
- **/proc/sys/kernel/sched_rt_runtime_us**: Defines the time period to be devoted to running realtime threads, in microseconds ('us' being the closest equivalent to 'μs' in plain text). The default value is 950000μs, or 0.95 seconds.

3 Objetives

- Learn about Kernel scheduling, specifically running real time scheduling.
- Observe the change in AIO Stress when the microseconds increased.

4 Justification

The need to learn about how the Linux kernel schedulers works, primarily to make an analysis of the changes to be made in the values of this. Also understand the meaning of the values that we will change and the relationship between the microseconds (run time) and the AIO Stress average.

5 Development

The default value is 950000μs, or 0.95 seconds for the sched_rt_runtime_us or running realtime threads meaning that 5% of the CPU time is reserved for processes that don't run under a real-time or deadline scheduling policy. This value in this file specifies how much of the "period" time can be used by all real-time and deadline scheduled processes on the system. The value in this file can range from -1 to INT_MAX-1. Specifying -1 makes the run-time the same as the period; that is, no CPU time is set aside for non-real-time processes.

Figure 1 shows the test results by changing the default value for this section of the scheduler. The microseconds μs were changed from 1,000,000 to 100,000 with a difference of 100,000 for each different test except by the default value. In this figure are included different parameters as Standard Error and Standard Deviation because the average AIO Stress can't be exact. Figure 2 shows the relation between Microseconds and AIO Stress only. This shows how the results are varying going up and down.

5.1 System Specifications

This project was developed in a DELL computer with the following specifications:

- Processor: Intel Core i7-4500U @ 3.00GHz (4 cores)
- Motherboard: Dell 03VVKX
- Chipset: Intel Haswell-UTL DRAM
- Memory: 2 x 4096 MB DDR3-1600MHz
- Disk: 1000GB Western Digital WD10JPVX-75J
- Processor: Intel Core i7-4500U @ 3.00GHz (4 cores)
- Audio:Intel Haswell-ULT HD Audio
- Network: Realtek RTL8101/2/6E + Qualcomm Atheros QCA9565 / AR9565
- OS: Ubuntu 16.04
- Kernel:4.4.0-43-generic (x86_64)
- Desktop: Unity 7.4.0
- Display Server: X Server 1.18.4
- Display Driver: intel 2.99.917
- OpenGL: 3.3 Mesa 11.2.0
- Compiler: GCC 5.4.0 20160609
- File-System: ext4
- Screen Resolution: 1366x768

Microseconds	AIO Stress	Standard Error	Standard deviation
1,000,000	111.06 mb/s	0.55	0.86%
950,000	**93.71 mb/s**	**4.23**	**11.08%**
900,000	108.6 mb/s	1.54	2.46%
800,000	105.89 mb/s	1.19	1.95%
700,000	106.84 mb/s	1.10	1.78%
600,000	104.71 mb/s	1.83	3.03%
500,000	95.76 mb/s	5.21	13.32%
400,000	101.25 mb/s	1.51	2.99%
300,000	103.39 mb/s	1.58	2.64%
200,000	103.13 mb/s	1.29	2.17%
100,000	94.87 mb/s	3.34	8.64%

Figure 1: Results for the different test values

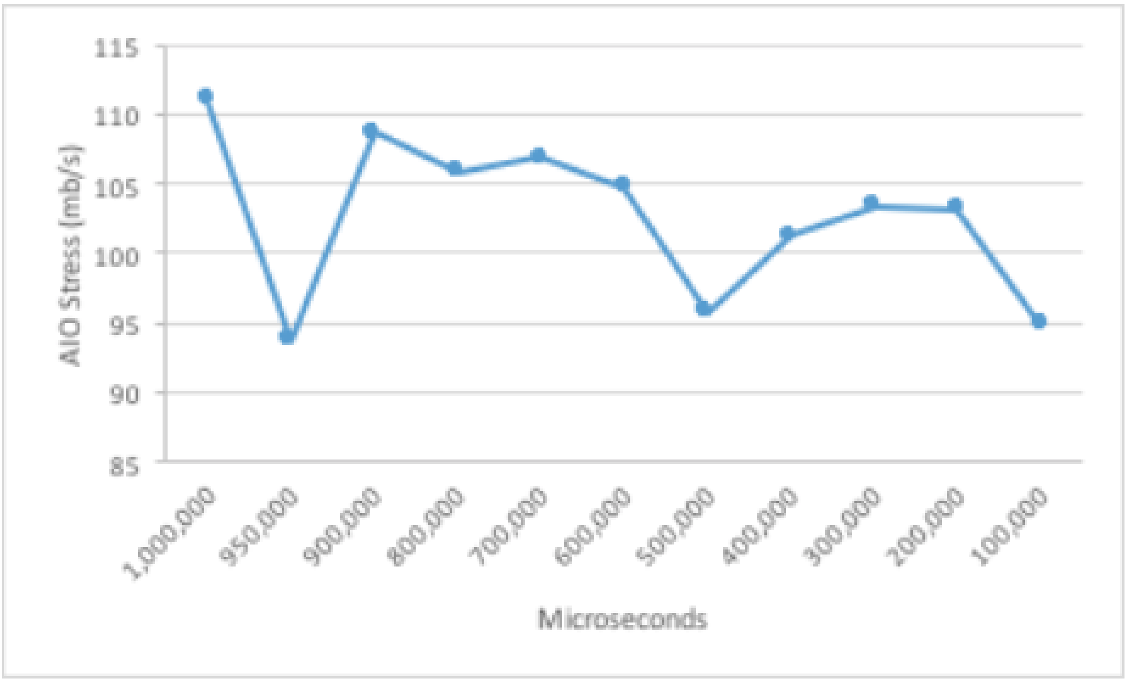

Figure 2: Relation Microseconds-AIO Stress

6 Conclusion

The project was completed and thus further studies on the equipment, the use and importance of the scheduler in something so important to the computer as the kernel, although the kernel is the essential center of a computer operating system, the core that provides basic services for all other parts of the operating system, a kernel can be contrasted with a shell, the outermost part of an operating system that interacts with user commands. But what is noteworthy between these two is that the kernel has the scheduler, the unit responsible to keep busy the CPUs in the system, but this in a fully efficient manner. As seen throughout the development of this class, a scheduler may be a small part of the scheduler, but is very important because it coordinates the smooth operation between the processes occurring in the system, which operate the computer the way the user needs.

Importantly, the scheduler has within its configuration variables, which can be modified by a user in order to obtain an improvement in system performance and that was what was done in this project. It was learned to modify the scheduler to know and take notes on how to change the computer's performance and thus, together, find a specific value to achieve optimization of the computer.

It was very important practice because, although what was done can be found in books, is not the same know what can be done within a computer to really do it in practice and check for yourself how it changes so much with so little bit.

More importantly it was learned to appreciate what is the area of operating systems because if behind this project there was much research, imagine what must have researched and tested those who made the Linux OS.

7 References

Kernel. (s.f.). Deadline Task Scheduling. Recovered by: https://www.kernel.org/doc/Documentation/scheduler/sched-deadline.txt

Open Suse.(s.f.). Tuning the time scheduler. Recovered by: https://doc.opensuse.org/documentation/html/openSUSE_121/opensuse-tuning/cha.tuning.taskscheduler.html

Phoronix Test Suite. (2016). OpenBenchmarking.org. Retrieved from AIO-Stress [pts/aio-stress] : https://openbenchmarking.org/test/pts/aio-stress

Red Hat. (s.f.). CPU Scheduling. Recovered by: https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Performance_Tuning_Guide/s-cpu-scheduler.html