Reduce Linux power consumption, Part 1: The CPUfreq subsystem

Meet all the power-efficiency tuning components for Linux on System x

Skill Level: Intermediate

Jenifer Hopper (jhopper@us.ibm.com)
Software Engineer
IBM

15 Sep 2009

This three-part series is your starting point for tuning your system for power efficiency. In Part 1, get up to speed on the components and concepts you need to fine-tune a Linux-based System x server for power efficiency. Learn how to enable the Linux CPUfreq subsystem, get instruction on C and P states, and determine which of the five in-kernel governors you need to boost power efficiency on your system.

About this series

In this series, learn how to tune your Linux-based IBM System x server for power efficiency. You'll learn about the in-kernel governors and their settings and how to use them; you'll also see the effects of the tuned governors on a power performance and e-commerce workload. The examples are based on a System x server running Red Hat Enterprise Linux version 5.2 (RHEL 5.2), but the same guidelines apply to any of the 2.6.x kernels, as well as any processor type that supports frequency scaling.

Part 1 introduces the components and concepts you'll need to tune your system for power efficiency, including the Linux CPUfreq subsystem, C and P states, and the five in-kernel governors.

Part 2 will give more details on the general settings of the Linux CPUfreq subsystem and the five in-kernel governors—*performance*, *powersave*, *userspace*, *ondemand*, and *conservative*—and their settings.

Part 3 will compare the performance of the five in-kernel governors in both a tuned and an untuned state to show you what results you can achieve by power tuning your system.

Power efficiency is an important consideration for anyone concerned with business costs or environmental issues. In this article, let's look at how to use the Linux CPUfreq subsystem and in-kernel governors to change the processor's operating frequency to improve your system's power efficiency without having a major impact on performance. However, power efficiency tuning has its limits based on the actual hardware (more on this in Part 2 of this series).

The Linux CPUfreq subsystem

Starting with the 2.6.0 Linux kernel, you can dynamically scale processor frequencies through the CPUfreq subsystem. When processors operate at a lower clock speed, they consume proportionately less power and generate less heat. This dynamic scaling of the clock speed gives some control in throttling the system to consume less power when not operating at full capacity.

The CPUfreq structure makes use of governors and daemons for setting a static or dynamic power policy for the system. The dynamic governors, which I discuss later in this article, can switch between CPU frequencies based on CPU utilization to allow for power savings while not sacrificing performance. These governors also allow for some user tuning so you can customize and easily change the frequency scaling. In addition, the `sched_mc_power_savings` and `sched_smt_power_savings` settings make use of consolidating threads to save power.

C states and P states

Join the green groups on My developerWorks
Discuss topics and share resources about energy, efficiency, and the environment on the GReen IT Report space and the Green computing group on My developerWorks.

Before we begin the CPUfreq discussion, let's review C and P states.

C states: Almost all idle

C states, with the exception of the *C0* state where the processor is running, are idle states in which the processor will unclock and shut down components to save power. The deeper the C state, the more power saving steps are taken—steps like stopping the processor clock or stopping interrupts from coming in. These states can provide power savings when the system is idle.

There is also a mode called *C1E* (alternately known as *Enhanced C1* or *C1 Enhanced Mode*) that can help power savings at idle. C1E tries to provide more power savings than the traditional C1 state (which only halts the clock signal) by also lowering the voltage and frequency. In fact, C1E has the ability to lower the voltage/frequency faster than any of the CPUfreq governors.

Not all processors have all these options, but to use C states and C1E, make sure that you have the BIOS options `CPU C State` and `C1E` (or anything similar) enabled to get the greatest power savings at idle. Some systems support a *C3* and even a *C6* deep-sleep state.

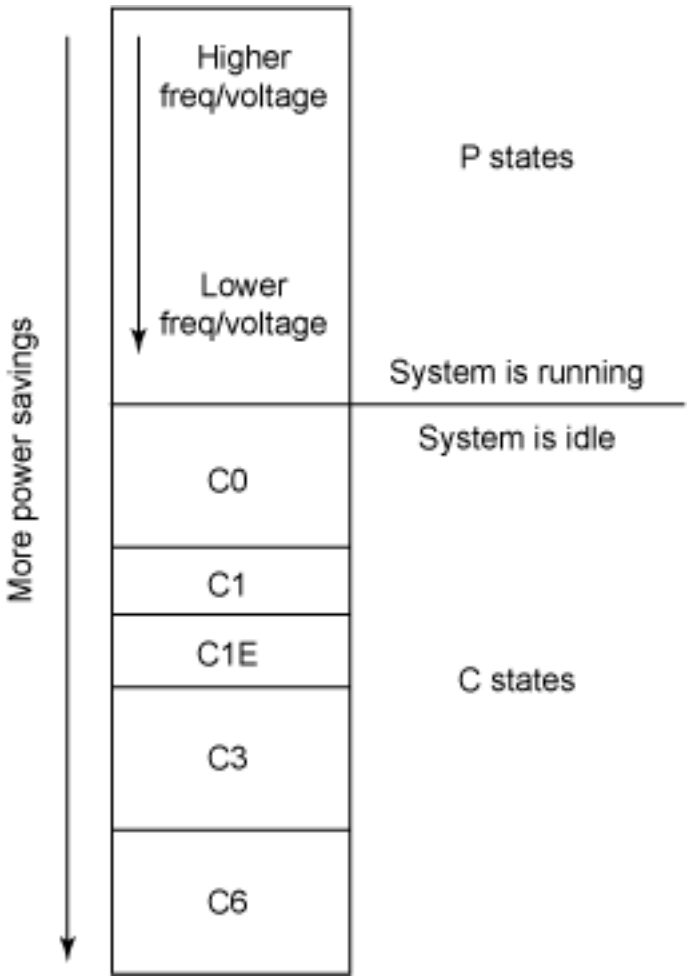
Remember, the deeper the C state, the more power savings.

P states: In operation

P states are operational states that relate to CPU frequency and voltage. The higher the P state, the lower the frequency and voltage at which the processor runs. The CPUfreq governors use P states to change frequencies and lower power consumption.

You need to have the `Processor Performance States` BIOS option (or anything like it) enabled on your system to use P states and the CPUfreq governors. Figure 1 is a simple diagram of C and P states.

Figure 1. C and P states



Prerequisites for the CPUfreq subsystem

Before you can make use of the CPUfreq subsystem, you need the prerequisites described in this section. CPUfreq is enabled by default in RHEL 5.2 (it's normally enabled as well in other distributions). One quick way to check if CPUfreq is already enabled is to look in the /sys filesystem. If you see the cpufreq directory listed under /sys/devices/system/cpu/cpu*/cpufreq/, then your system currently has CPUfreq enabled. If you do not see this directory listed, follow these instructions to ensure that you have the required pieces.

First, make sure that your processor can support frequency scaling. See the list of hardware that supports the CPUfreq subsystem in Resources later in this article.

Next, look at your kernel config file. All of the required configurations are usually set by default for the RHEL 5.2 kernel, but you may want to change some of the settings to achieve the desired startup state for your system. The following options are located in the *CPU Frequency scaling* section of the config file:

`CONFIG_CPU_FREQ`
This option must be set to `y` to make use of the kernel's CPU frequency scaling.

`CONFIG_CPU_FREQ_GOV_PERFORMANCE, CONFIG_CPU_FREQ_GOV_POWERSAVE, CONFIG_CPU_FREQ_GOV_USERSPACE, CONFIG_CPU_FREQ_GOV_ONDEMAND, CONFIG_CPU_FREQ_GOV_CONSERVATIVE`
These options are for each of the available CPUfreq governors. To use a governor, set the config option to `y` or `m`. If you set the option to `y`, that governor's module will be built into the kernel. If you set the option to `m`, you will have to load the module yourself for each boot by issuing one or all of the following commands:

```
modprobe cpufreq_performance
modprobe cpufreq_powersave
modprobe cpufreq_userspace
modprobe cpufreq_ondemand
modprobe cpufreq_conservative
```

Alternatively, if you chose `m`, you can have the module loaded at boot time by adding the governor modules to /etc/rc.local. Also note that you can set either the userspace or the performance governor to the default by setting either `CONFIG_CPU_FREQ_DEFAULT_GOV_USERSPACE` or `CONFIG_CPU_FREQ_DEFAULT_GOV_PERFORMANCE` to `y`.

Also, to use `sched_mc_power_savings` and `sched_smt_power_savings`, which are discussed later, make sure that options `CONFIG_SCHED_MC` and `CONFIG_SCHED_SMT` are set to `y` under the *Processor type and features* section of the config file.

For any changes in the config file to take effect, you must rebuild and boot your kernel. You probably know how to do that, but if you're fuzzy on any details, check any of the many sources for instructions on rebuilding a Linux kernel (see Resources for suggestions).

Here come the governors

There are five in-kernel governors available for use with the CPUfreq subsystem. These governors set the processor frequency based on certain criteria; some dynamically change the frequency as inputs are changed either by the system or the user. This articles focuses on RHEL 5.2, which is based on the 2.6.18 kernel, so all of these governors are available for use. Let's meet them. (Part 2 and Part 3 in this series takes you into deeper detail on the governors.)

Performance governor: Highest frequency

The performance governor statically sets the processor to the highest frequency

available. You can adjust the range of frequencies available to this governor. As the name implies, this governor's goal is to get the maximum performance out of a system by setting the processor clock speed to the maximum level and leaving it there. This governor does not attempt to provide any power savings by default, although you can tune the governor to change the frequency it selects.

Powersave governor: Lowest frequency

On the flip side, the powersave governor statically sets the processor to the lowest available frequency. Again you can adjust the range of frequencies available to this governor. The purpose of this governor is to run at the lowest speed possible at all times. Obviously this can affect performance in that the system will never rise above this frequency no matter how busy the processors are.

In fact, this governor often does not save any power since the greatest power savings usually come from the savings at idle through entering C states. Using the powersave governor will prolong a running process since it will be running at the slowest frequency; therefore, it will take longer for the system to go idle and get the C state savings.

Userspace governor: Manual frequencies

Next there is the userspace governor, which allows you to select and set a frequency manually. This governor also works with processor frequency daemons running in userspace to control frequency (we'll talk more about daemons and provide examples in Part 2). This governor is useful for setting a unique power policy that is not preset or available from the other governors; you can also use it to experiment with policies.

Note that the userspace governor itself does not dynamically change the frequency; rather, it allows you or a userspace program to dynamically select the processor frequency.

Ondemand governor: Frequency change based on processor use

Introduced in the 2.6.10 kernel, the ondemand governor was the first in-kernel governor to dynamically change processor frequency based on processor utilization. The ondemand governor checks the processor utilization and if it exceeds the threshold, the governor will set the frequency to the highest available. If the governor finds the utilization to be less than the threshold, it steps down the frequency to the next available. If the system continues to be underutilized, the governor will continue stepping down the frequency until the lowest available is set.

You can control the range of frequencies available, the rate at which the governor checks utilization on the system, and the utilization threshold.

Conservative governor: A more gradual ondemand

Based on the ondemand governor, the conservative governor (which was introduced in the 2.6.12 kernel) is similar in that it dynamically adjusts frequencies based on processor utilization; however, the conservative governor behaves a little differently and allows for a more gradual increase in power. The conservative governor checks the processor utilization and if it is above or below the utilization thresholds, the governor steps up or down the frequency to the next available instead of just jumping to the highest frequency as ondemand does.

You can control the range of frequencies available, the rate at which the governor checks utilization on the system, the utilization thresholds, and the frequency step rate.

Next time

In Part 2, we'll go deeper into setup and usage of the subsystem. I will provide some general setup and usage settings for the Linux CPUfreq subsystem and some different interface options. I will also cover governor-specific settings and administration schedulers to help you zero in on the right tools for your usage.

In Part 3, I will discuss the effects each of the governors can have on different workloads using two popular configuration workloads.

Your thoughts

In Part 3, the author will discuss energy/performance tradeoffs in terms of Watts. How important would CPU energy usage be in your decision to implement these power-saving measures? What other factors would you consider? What other power-saving measures would you consider? Add your comments below to compare notes with other readers.

Resources

Learn

- Review the list of hardware that supports the CPUfreq subsystem.
- Check out these additional materials on power consumption:
  - The tutorial "How to make use of Dynamic Frequency Scaling"
  - The tutorial "Enhanced Intel SpeedStep Technology and Demand-Based Switching on Linux"
  - The article "Making power policy just work" (on power schedulers)
  - The article "CPU frequency scaling in Linux"
  - The documentation "Linux CPUfreq Governors" (on CPU frequency and voltage scaling code in the Linux kernel)
  - The Gentoo "Power Management Guide" (comes with a caveat — for laptops, so don't apply to servers unless you know what you're doing!)
  - The tutorial "How to use CPU frequency scaling (cpufreq)"
  - This wiki entry on CPU Frequency Scaling
  - The tutorial "Scheduler tunables for multi-socket systems"
  - And data on the CPUfreq subsystem from kernel.org
- Need help rebuilding/rebooting your kernel? Try Kwan Lowe's "Kernel Rebuild Guide."
- In the developerWorks Linux zone, find more resources for Linux developers, and scan our most popular articles and tutorials.
- See all Linux tips and Linux tutorials on developerWorks.
- Stay current with developerWorks technical events and Webcasts.

Get products and technologies

- With IBM trial software, available for download directly from developerWorks, build your next development project on Linux.

Discuss

- Get involved in the My developerWorks community; with your personal profile and custom home page, you can tailor developerWorks to your interests and interact with other developerWorks users.

About the author

Jenifer Hopper

Jenifer Hopper is a software engineer for the IBM Linux performance group in Austin, Texas. Her current focus is on High Performance Computing (HPC) and energy workloads, as well as system profilers and data analysis tools.