

WES 237A Linux Governors Guide

1. Overview

These three links are useful if you want to tune processor frequency, be sure to read through these to get comfortable with Linux cpufreq subsystem concepts:

<http://www.ibm.com/developerworks/linux/library/l-cpufreq-1/index.html>

<http://www.ibm.com/developerworks/linux/library/l-cpufreq-2/index.html>

<http://www.ibm.com/developerworks/linux/library/l-cpufreq-3/index.html>

Below is a digest of what you should pay attention to.

2. 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 article 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.)

A. 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.

B. 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.

C. 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.

D. 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.

You can communicate with the Linux `cpufreq` subsystem using the `/sys` file system, in Section 3.

3. Governor Utilities

A. Checking the contents of /sys/devices/system/cpu/

```
# cd /sys/devices/system/cpu/

#ls
online
possible
present
kernel_max
offline
cpufreq
cpuidle
cpu0
cpu1
perf_events
```

B. Checking the CPUfreq directory

```
[root@systemx cpu]# cd cpu0/cpufreq/
[root@systemx cpufreq]# ls -l
total 0
-r--r--r-- 1 root root 4096 Oct 31 14:53 affected_cpus
-r----- 1 root root 4096 Oct 31 14:53 cpuinfo_cur_freq
-r--r--r-- 1 root root 4096 Oct 31 14:53 cpuinfo_max_freq
-r--r--r-- 1 root root 4096 Oct 31 14:53 cpuinfo_min_freq
-r--r--r-- 1 root root 4096 Oct 31 14:53 scaling_available_frequencies
-r--r--r-- 1 root root 4096 Oct 31 14:53 scaling_available_governors
-r--r--r-- 1 root root 4096 Oct 31 14:53 scaling_cur_freq
-r--r--r-- 1 root root 4096 Oct 31 14:53 scaling_driver-rw-r--r-- 1 root root
0 Nov 5 11:44 scaling_governor
-rw-r--r-- 1 root root 4096 Oct 31 14:53 scaling_max_freq
-rw-r--r-- 1 root root 4096 Oct 31 14:53 scaling_min_freq
```

C. Checking frequencies

```
# cat scaling_cur_freq
384000
```

D. Checking max, min frequencies

```
# cat cpuinfo_max_freq
1188000

# cat cpuinfo_min_freq
192000
```

E. Checking Available Governors

```
# cat scaling_available_governors
ondemand powersave conservative userspace performance
```

F. Checking the Current Governor & Changing Governors

```
# cat scaling_governor
performance

# echo conservative > scaling_governor
# cat scaling_governor
conservative
```

G. Changing Frequencies in Userspace Governor

Note: These commands *ONLY* apply to the “userspace” governor

```
# echo 1188000 > scaling_setspeed
```