Lenovo

Using Cpufreq on Linux Servers to Manage Power Consumption

Introduces the cpufreq subsystem in Linux

Describes the use of the acpi-cpufreq and intel-pstate drivers

Shows how to set the power mode in UEFI on ThinkSystem servers

Provides guidances on driver selection based on application type

Huaisheng Ye

LENOVO PRESS

Abstract

One of the most effective ways to reduce power consumption and heat output on a server system is the cpufreq subsystem. Cpufreq, also referred to as CPU frequency scaling or CPU speed scaling, is the infrastructure in the Linux kernel space that enables users to scale the CPU frequency in order to save power.

CPU scaling can be initiated in different ways:

- Automatically based on the system loading
- In response to ACPI events
- Automatically by the hardware
- Manually by userspace programs

This paper show users how to enable cpufreq drivers through correct BIOS settings, and introduces the interface of the cpufreq subsystem that allows flexible control through processor frequency. In addition, this document offers a guide to users, showing them how to choose proper drivers based on different scenarios.

At Lenovo® Press, we bring together experts to produce technical publications around topics of importance to you, providing information and best practices for using Lenovo products and solutions to solve IT challenges.

See a list of our most recent publications at the Lenovo Press web site:

http://lenovopress.com

Do you have the latest version? We update our papers from time to time, so check whether you have the latest version of this document by clicking the **Check for Updates** button on the front page of the PDF. Pressing this button will take you to a web page that will tell you if you are reading the latest version of the document and give you a link to the latest if needed. While you're there, you can also sign up to get notified via email whenever we make an update.

Contents

Introduction

Intel's Enhanced Intel SpeedStep and AMD's PowerNow Technology have been introduced in modern x86 platforms to allow the management of processor power consumption via performance state transitions. These states are defined as discrete operating points associated with different voltages and frequencies. Based on these technologies, the operating system (OS) cpufreq subsystem can develop power management functions, allowing users to further reduce electricity consumption, without significantly affecting processor performance.

Modern CPUs support P-states, C-states and S-states for power saving. Figure 1 shows these relationships.

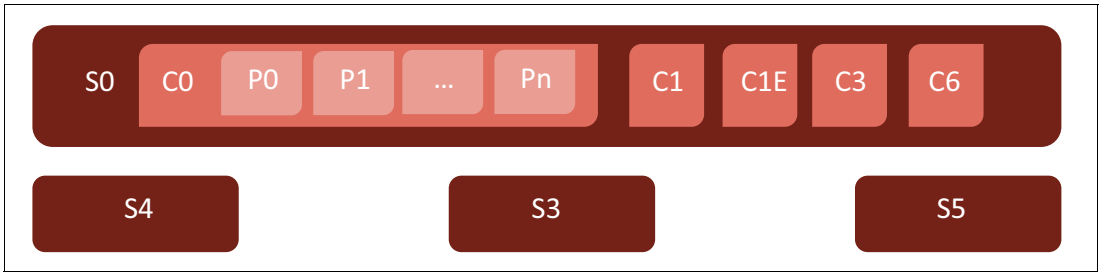

Figure 1 The Relationship of CPU P-states, C-states and S-states

C-states

Generally, the CPU power states C0 to C3 are defined as follows:

- C0 is the operating state.
- C1 (often known as Halt) is a state where the processor is not executing instructions, but can return to an executing state almost instantaneously. All ACPI-conformant processors must support this power state. Some processors, such as the Pentium 4, also support an Enhanced C1 state (C1E or Enhanced Halt State) for lower power consumption.
- C2 (often known as Stop-Clock) is a state where the processor maintains complete all software visibility, but may take longer to wake up. This processor state is optional.
- C3 (often known as Sleep) is a state where the processor does not need to keep its cache coherent, but maintains other states. Some processors have variations on the C3 state (Deep Sleep, Deeper Sleep, etc.) that differ in how long it takes to wake the processor. This processor state is optional.

Additional states are defined by manufacturers for some processors. For example, Intel's E5 v3 (Haswell) platform has states up to C10, which are divided into core states and package states.

P-states

While the CPU operates in C0, it can be in one of several power-performance states. These states are implementation-dependent. However, P0 is always the highest-performance state, with P1 to Pn being successively lower-performance states.

P-states have become known as SpeedStep in Intel processors and as PowerNow! or Cool'n'Quiet in AMD processors.

- P0: max power and frequency

- P1: less power than P0, voltage and frequency scaled
- P2: less power than P1, voltage and frequency scaled
- ...
- Pn less power than P(n-1), voltage and frequency scaled

S-state means sleeping state. Server products should always be at S0 when powered on.

The significance of the kernel cpufreq subsystem

This paper only introduces the cpufreq subsystem within the P-state scope. Other drivers are responsible for C-state adjustments and they aren't discussed in this paper.

The main goals of the cpufreq subsystem are:

- Reduce overall processor consumption to save costs by means of P-states switching
- Provide a flexible and standardized interface to control each logical core independently

The benefits of proper use of the cpufreq subsystem include:

- Heat reduction for servers and computing centers
- Reduced secondary costs, including cooling, space, cables, and generators
- Lower electricity fees
- Lower carbon dioxide output
- Meeting government regulations or legal requirements regarding Green IT
- Meeting company guidelines for new platforms

Cpufreq subsystem

One of the most effective ways to reduce power consumption and heat output on your system is the cpufreq subsystem. Cpufreq, also referred to as CPU speed scaling, is the infrastructure in the Linux kernel that enables users to scale the CPU frequency in order to save power.

CPU scaling can be done automatically based on the system load and in response to ACPI events, automatically by the hardware, or manually by userspace programs. It allows the clock speed of the processor to be adjusted on the fly, and enables the system to run at a reduced clock speed to save power.

The rules for shifting frequencies, whether to a faster or slower clock speed, and when to shift frequencies, are defined by the cpufreq governor or policy.

As of the time of writing, the mainstream Linux kernel has two different drivers which are used to implement independent processor frequency scaling:

- intel-pstate
- acpi-cpufreq

The intel-pstate is the newer driver. These two driver types have different implementation mechanisms, but most importantly, they have unique features based on different mechanisms. Therefore, users should choose the proper driver based on their application environment and needs.

Figure 2 on page 5 shows the main structure of the kernel cpufreq subsystem.

The two power management drivers use the same framwork of Linux kernel, however acpi-cpufreq uses cpufreq's cpu frequency scaling algorithms, whereas intel-pstate uses its own algorithms to choose proper frequency based on different policies.

In Figure 2, when acpi-cpufreq is used as the cpufreq driver, the performance, ondemand, powersave and userspace governors are loaded first. acpi-cpufreq makes use of methods defined in the ACPI tables including PCT(Performance Control), which describes the processor registers and firmware locations that allow the operating system to change Pstate and to check the status of Pstate requests.

Intel-pstate is different from acpi-cpufreq in that it doesn't need firmware to provide an ACPI table to determine CPU frequency levels. Intel-pstate has capability to get the levels by itself through checking related MSR registers.

Intel-pstate has two working models, one is Hardware-Controlled Performance States (HWP) and the other is via software p-states. Whether it will work with HWP mode or software p-states mode, depends on a parameter in the CPUID (IA32_PM_ENABLE) to determine whether the current CPU has enabled HWP or not.

- If HWP feature has been enabled, intel-pstate will pass the responsibility of adjusting frequency to the CPU itself. All it needs to do is simply offer a hint to CPU hardware.
- If HWP feature hasn't been enabled, intel-pstate will check the family of CPU has been added to its supported list or not. If it is added, intel-pstate will work in software mode, and it will take responsibility as choosing proper frequency based on actual workload. If the family of CPU hasn't been added, then intel-pstate will exit and return "No such device", and the OS will then attempt to load acpi-cpufreq instead.

Note: Regardless of HWP or software p-states mode, intel-pstate doesn't need to use cpufreq subsystem's governors. This is because it has implemented them as 'performance' and 'powersave' policies internally.

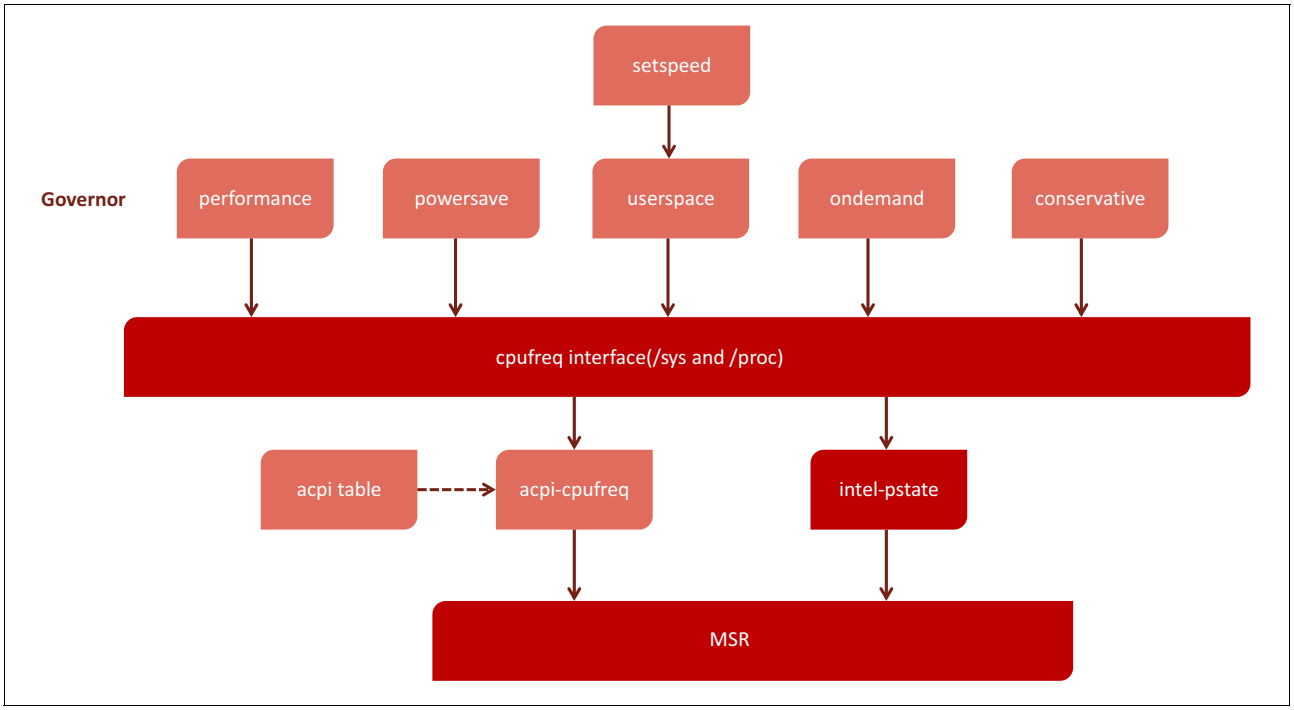

Figure 2 Structure of the cpufreq Subsystem

In this paper, we provide guidance and information regarding the cpufreq subsystem driver for users' systems that run with RHEL 7 and SLES1 2.

SetPolicy and Target mechanisms

For the drivers that belong to the SetPolicy mechanism, like intel-pstate, users need to select powersave or performance, as well as whether they want more aggressive power-saving or more instantly available processing power. Then, the driver will decide the proper frequency based on the current policy.

For a performance policy, the driver will operate a related MSR, like MSR_IA32_PERF_CTL or MSR_HWP_REQUEST, to offer a hint to processor, such as "run at maximum performance and do not consider power saving".

For a powersave policy, the driver will periodically calculate its current frequency and CPU utilization. If the utilization exceeds or is below a predefined threshold, the driver or hardware P-state will take action to increase or decrease the current P-state.

The classic acpi-cpufreq driver belongs to Target implementation and a governor must be selected. Such a governor decides at what speed the processor shall run, within limitations such as maximum and minimum values. One such governor is the userspace governor. It allows the user to decide the specific speed at which the processor should run.

Governors

The governor defines the power characteristics of the system CPU, which in turn affects CPU performance. Each governor has its own unique behavior, purpose, and suitability in terms of workload.

This section describes how to choose and configure a CPUfreq governor, the characteristics of each governor, and what kind of workload each governor is suitable for. Different types of CPUfreq governors available in RHEL 7 are listed in the file /sys/devices/system/cpu/cpu*/cpufreq/scaling_available_governors as shown in Figure 3.

Note: Governors only should be involved with Target implementation drivers like acpi-cpufreq. For SetPolicy implementation, drivers like intel-pstate do not need any governors.

```
[root@localhost cpufreq]# ls
affected_cpus               freqdomain_cpus               scaling_governor
bios_limit                  related_cpus                  scaling_max_freq
cpuinfo_cur_freq            scaling_available_frequencies scaling_min_freq
cpuinfo_max_freq            scaling_available_governors   scaling_setspeed
cpuinfo_min_freq            scaling_cur_freq
cpuinfo_transition_latency  scaling_driver
[root@localhost cpufreq]# cat scaling_available_governors
conservative userspace powersave ondemand performance
```

Figure 3 Contents of scaling_available_governors

The available governors are as follows:

- cpufreq_performance:

  The Performance governor forces the CPU to use the highest possible clock frequency. This frequency will be statically set, and will not change. As such, this particular governor

offers no power saving benefits. It is only suitable for heavy workloads, and even then, only during times when the CPU is rarely (or never) idle.

- cpufreq_powersave:

  By contrast, the Powersave governor forces the CPU to use the lowest possible clock frequency. This frequency will be statically set, and will not change. As such, this particular governor offers maximum power savings, but at the cost of the lowest CPU performance.

  The term *powersave* can sometimes be deceiving, though, since (in principle) a slow CPU on full load consumes more power than a fast CPU that is not loaded. As such, while it may be advisable to set the CPU to use the Powersave governor during times of expected low activity, any unexpected high loads during this time can cause the system to actually consume more power.

  The Powersave governor is, in simple terms, more of a “speed limiter” for the CPU than a “power saver”. It is most useful in systems and environments where overheating can be a problem.

- cpufreq_ondemand:

  The Ondemand governor is a dynamic governor that allows the CPU to achieve maximum clock frequency when the system load is high, and also minimum clock frequency when the system is idle. While this allows the system to adjust power consumption with respect to system load, it does so at the expense of latency between frequency switching. As such, latency can offset any performance/power saving benefits offered by the Ondemand governor if the system switches between idle and heavy workloads too often.

  For most systems, the Ondemand governor can provide the best compromise between heat emission, power consumption, performance, and manageability. When the system is only busy at specific times of the day, the Ondemand governor will automatically switch between maximum and minimum frequency depending on the load, without any further intervention.

- cpufreq_userspace:

  The Userspace governor allows userspace programs, or any process running as a root process, to set the frequency. Of all the governors, Userspace is the most customizable. Depending on how it is configured, it can offer the best balance between performance and consumption for your system.

- cpufreq_conservative:

  Like the Ondemand governor, the Conservative governor also adjusts the clock frequency according to usage. However, while the Ondemand governor does so in a more aggressive manner (that is from maximum to minimum and back), the Conservative governor switches between frequencies more gradually.

  This means that the Conservative governor will adjust to a clock frequency that it deems fitting for the load, rather than simply choosing between maximum and minimum. While this can possibly provide significant savings in power consumption, it does so at an ever greater latency than the Ondemand governor.

Sysfs Interfaces of the cpufreq subsystem

The preferred interface for the cpufreq subsystem is located in the sysfs filesystem. If the user has mounted it at /sys, the cpufreq interface is located in a subdirectory cpufreq within the cpu-device directory (e.g. /sys/devices/system/cpu/cpu*/cpufreq/ for the first CPU). The contents of the directory is shown in Figure 4.

```
[root@localhost cpufreq]# pwd
/sys/devices/system/cpu/cpu63/cpufreq
[root@localhost cpufreq]# ls
affected_cpus               freqdomain_cpus                scaling_governor
bios_limit                  related_cpus                   scaling_max_freq
cpuinfo_cur_freq            scaling_available_frequencies  scaling_min_freq
cpuinfo_max_freq            scaling_available_governors    scaling_setspeed
cpuinfo_min_freq            scaling_cur_freq
cpuinfo_transition_latency     scaling_driver
```

Figure 4 Content of the cpufreq directory

Here are the descriptions of each of these interfaces:

- affected_cpus: List of Online CPUs that require software frequency coordination.
- cpuinfo_cur_freq: Current frequency of the CPU as obtained from the hardware, in kHz. This is the frequency the CPU actually runs at.
- cpuinfo_min_freq: This file shows the minimum operating frequency the processor can run at (in kHz).
- cpuinfo_max_freq: This file shows the maximum operating frequency the processor can run at (in kHz).
- cpuinfo_transition_latency: The time it takes to switch between two frequencies on this CPU in nanoseconds. If unknown or known to be so high that the driver does not work with the Ondemand governor, -1 (CPUFREQ_ETERNAL) will be returned. This information can be useful in choosing an appropriate polling frequency for a kernel governor or userspace daemon. Be sure not to switch the frequency too often, as this can result in performance loss.
- related_cpus: List of Online + Offline CPUs that need software frequency coordination.
- scaling_available_frequencies: List of available frequencies, in kHz.
- scaling_available_governors: This file shows the CPUfreq governors available in this kernel. You can see the currently activated governor in
- scaling_cur_freq: Current frequency of the CPU as determined by the governor and cpufreq core, in kHz. This is the frequency the kernel thinks the CPU runs at.
- scaling_driver: This file shows which cpufreq driver is used to set the frequency on this CPU scaling_governor, and by “echoing” the name of another governor you can change it. Please note that some governors won't load - they only work on certain specific architectures or processors.
- scaling_min_freq and scaling_max_freq: Show the current “policy limits” (in kHz). By echoing new values into these files, you can change these limits. NOTE: When setting a policy you need to first set scaling_max_freq, then scaling_min_freq.
- scaling_setspeed: This can be read to get the value currently programmed by the governor. This can be written to change the current frequency for a group of CPUs, represented by a policy. Currently, this is supported only by the userspace governor. If you have selected the “userspace” governor, which allows you to set the CPU operating frequency to a specific value, you can read out the current frequency in scaling_setspeed. By “echoing” a new frequency here, you can change the speed of the CPU, but only within the limits of scaling_min_freq and scaling_max_freq.
- bios_limit: If the BIOS tells the OS to limit a CPU to lower frequencies, the user can read out the maximum available frequency from this file. This typically can happen through (often not intended) BIOS settings, restrictions triggered through a service processor or

other BIOS/HW based implementations. This does not cover thermal ACPI limitations which can be detected through the generic thermal driver.

Example 1 shows the output which comes from the sysfs interfaces.

Example 1 Contents of the sysfs governor

```
[root@localhost cpufreq]# ls
affected_cpus               freqdomain_cpus               scaling_governor
bios_limit                  related_cpus                  scaling_max_freq
cpuinfo_cur_freq            scaling_available_frequencies  scaling_min_freq
cpuinfo_max_freq            scaling_available_governors    scaling_setspeed
cpuinfo_min_freq            scaling_cur_freq
cpuinfo_transition_latency  scaling_driver
[root@localhost cpufreq]# cat affected_cpus
63
[root@localhost cpufreq]# cat bios_limit
2101000
[root@localhost cpufreq]# cat cpuinfo_cur_freq
2101000
[root@localhost cpufreq]# cat cpuinfo_max_freq
2101000
[root@localhost cpufreq]# cat cpuinfo_min_freq
1000000
[root@localhost cpufreq]# cat cpuinfo_transition_latency
10000
[root@localhost cpufreq]# cat freqdomain_cpus
63
[root@localhost cpufreq]# cat related_cpus
63
[root@localhost cpufreq]# cat scaling_available_frequencies
2101000 2100000 2000000 1900000 1800000 1700000 1600000 1500000 1400000 1300000 1200000
1100000 1000000
[root@localhost cpufreq]# cat scaling_available_governors
conservative userspace powersave ondemand performance
[root@localhost cpufreq]# cat scaling_cur_freq
2101000
[root@localhost cpufreq]# cat scaling_driver
acpi-cpufreq
[root@localhost cpufreq]# cat scaling_governor
performance
[root@localhost cpufreq]# cat scaling_max_freq
2101000
[root@localhost cpufreq]# cat scaling_min_freq
1000000
[root@localhost cpufreq]# cat scaling_setspeed
<unsupported>
```

Using the acpi-cpufreq driver

The ACPI CPUfreq driver is a kernel driver that controls the frequency of a particular CPU through ACPI, which enables the communication between the kernel and the hardware.

Users can use the **cpupower** tool to get detailed frequency information, as shown in Example 2.

Example 2 Using cpupower to get frequency information

```
# cpupower -c all frequency-info
analyzing CPU 0:
  driver: acpi-cpufreq
  CPUs which run at the same hardware frequency: 0
  CPUs which need to have their frequency coordinated by software: 0
  maximum transition latency: 10.0 us
  hardware limits: 1000 MHz - 2.10 GHz
  available frequency steps:  2.10 GHz, 2.10 GHz, 2.00 GHz, 1.90 GHz, 1.80 GHz, 1.70 GHz,
1.60 GHz, 1.50 GHz, 1.40 GHz, 1.30 GHz, 1.20 GHz, 1.10 GHz, 1000 MHz
  available cpufreq governors: conservative userspace powersave ondemand performance
  current policy: frequency should be within 1000 MHz and 2.10 GHz.
                  The governor "performance" may decide which speed to use
                  within this range.
  current CPU frequency: 2.10 GHz (asserted by call to hardware)
  boost state support:
    Supported: yes
    Active: yes

analyzing CPU 1:
  driver: acpi-cpufreq
  CPUs which run at the same hardware frequency: 1
  CPUs which need to have their frequency coordinated by software: 1
  maximum transition latency: 10.0 us
  hardware limits: 1000 MHz - 2.10 GHz
  available frequency steps:  2.10 GHz, 2.10 GHz, 2.00 GHz, 1.90 GHz, 1.80 GHz, 1.70 GHz,
1.60 GHz, 1.50 GHz, 1.40 GHz, 1.30 GHz, 1.20 GHz, 1.10 GHz, 1000 MHz
  available cpufreq governors: conservative userspace powersave ondemand performance
  current policy: frequency should be within 1000 MHz and 2.10 GHz.
                  The governor "performance" may decide which speed to use
                  within this range.
  current CPU frequency: 2.10 GHz (asserted by call to hardware)
  boost state support:
    Supported: yes
    Active: yes
```

To modify the driver's governor to another supported governor, use the cpupower command as shown in Example 3.

Example 3 Using cpupower to set a new governor

```
[root@localhost cpufreq]# cpupower -c all frequency-set -g ondemand
Setting cpu: 0
Setting cpu: 1
Setting cpu: 2
Setting cpu: 3
Setting cpu: 4
Setting cpu: 5
Setting cpu: 6
Setting cpu: 7
Setting cpu: 8
Setting cpu: 9
```

```
Setting cpu: 10
Setting cpu: 11
…
Setting cpu: 60
Setting cpu: 61
Setting cpu: 62
Setting cpu: 63
Setting cpu: 63
```

Use with Lenovo ThinkSystem

To set acpi-cpufreq on a Lenovo ThinkSystem™ server, use the following steps:

1. Boot the server into XClarity Provisioning Manager (system setup) by pressing F1 when prompted.
2. Click the UEFI Setup menu and click **System Settings** → **Operating Modes** as shown in Figure 5 on page 11.

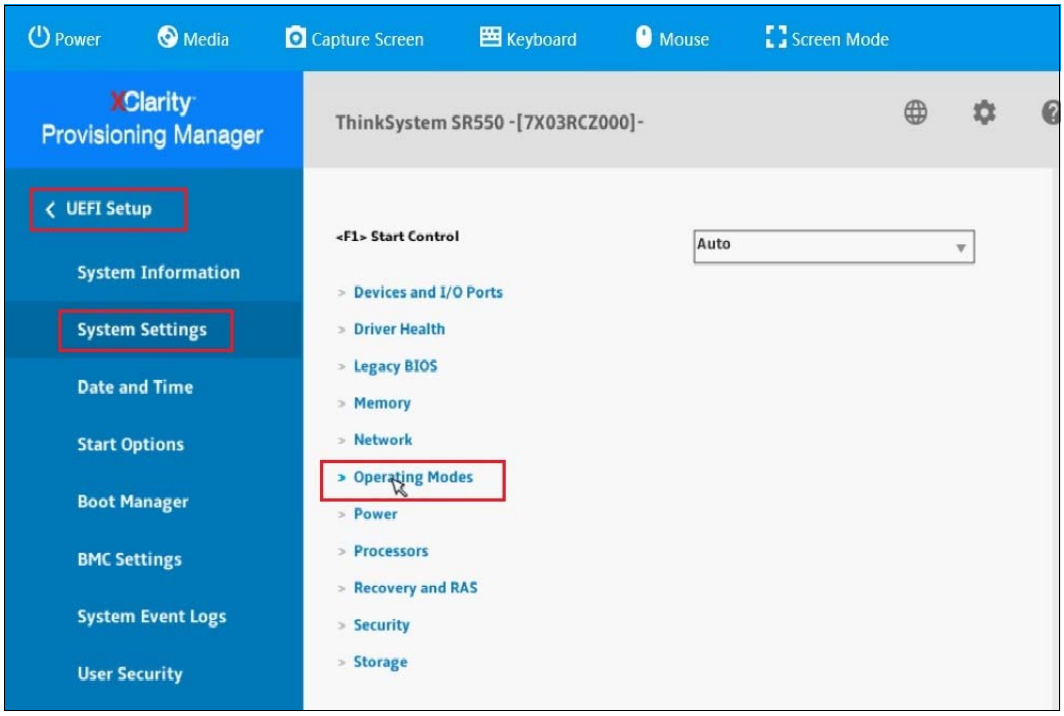

Figure 5 Navigating Provisioning Manager to Operating Modes

3. Select the following, as shown in Figure 6 on page 12:
   - Operating Mode: **Custom Mode**
   - CPU P-state Control: **Legacy**

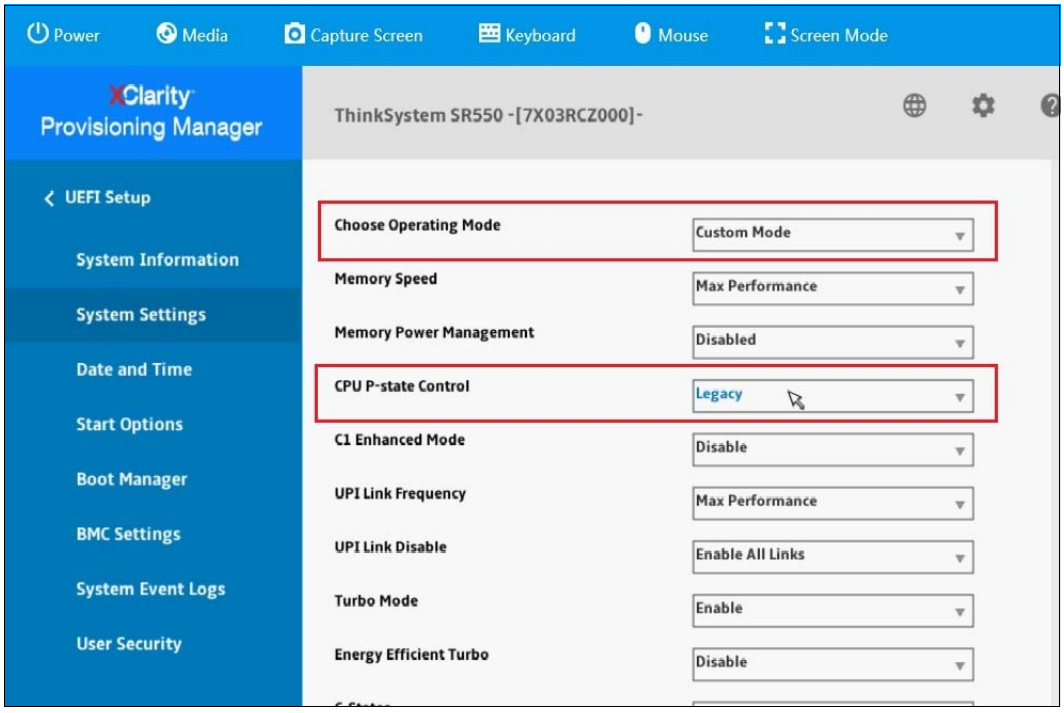

Figure 6 Operating Mode settings for acpi-cpufreq

4. Click **Save**, exit Provisioning Manager and reboot the server.

Using the intel-pstate cpufreq driver

Newer OS kernels, like RHEL 7 and SLES 12, support the Intel P-state driver. This driver provides an interface for controlling the P-state selection for processors based on the Intel Xeon E series architecture or a newer architecture.

A lot of cpufreq drivers or even most CPU frequency scaling algorithms only allow the CPU frequency to be set to predefined fixed values. Acpi-cpufreq is this way and uses the ->target() callbacks to get allowed CPU frequency. In addition, there are some cpufreq-capable processors that switch the frequency between certain limits on their own. These use the ->setpolicy() callback to offer a hint to CPU hardware and that is how intel-pstate works. This driver decides what P-state to use based on the policy requested from the cpufreq core. With HWP feature, the CPU can autonomously select P-states while utilizing OS supplied performance guidance hints.

Figure 7 shows intel-pstate's HWP working mode.

- At the bottom of the diagram Tstate (Throttling States) has capability to decrease power consumption by limiting CPU working at lower frequency. It is different from P-state that it depends on limiting CPU not working for a period of time. But P-state just adjusts frequency to switch at relatively low or high value.
- Lowest Frequency means the minimum frequency that is supported by CPU.
- Most Efficient Frequency means current value of the most efficient performance level. This value can change dynamically as a result of workload characteristics.

- Guaranteed Frequency means current value for the guaranteed performance frequency. This value can change dynamically as a result of internal or external constraints, e.g. thermal or power limits.
- Highest Frequency means the value for the maximum non-guaranteed performance frequency. It is determined by the physical maximum power limitation.
- EPP is Energy Performance Preference; it will dictate how aggressive the algorithm will adjust CPU frequency and it depends on system, workload, OS.

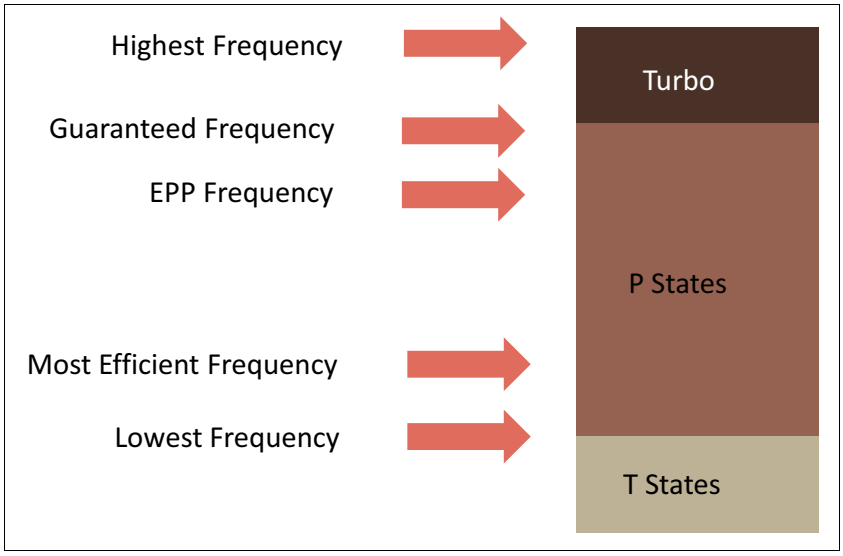

Figure 7 Intel-Pstate Diagram

Sysfs interfaces of Intel-Pstate

Intel P-state provides its own sysfs files to control the P-state selection. These files are located in the /sys/devices/system/cpu/intel_pstate/ directory. Any changes made to the files are applied to all CPUs. This directory contains five files that are used for setting P-state parameters, as shown in Figure 8.

```
[root@localhost intel_pstate]# pwd
/sys/devices/system/cpu/intel_pstate
[root@localhost intel_pstate]# ls
max_perf_pct  min_perf_pct  no_turbo  num_pstates  turbo_pct
[root@localhost intel_pstate]# cat max_perf_pct
100
[root@localhost intel_pstate]# cat min_perf_pct
27
[root@localhost intel_pstate]# cat no_turbo
0
[root@localhost intel_pstate]# cat num_pstates
28
[root@localhost intel_pstate]# cat turbo_pct
58
```

Figure 8 Contents of the intel_pstate directory

The p-state parameters are as follows:

- max_perf_pct: Limits the maximum P-state requested by the driver, expressed in a percentage of available performance. The available P-state performance can be reduced by the no_turbo setting.
- min_perf_pct: Limits the minimum P-state requested by the driver, expressed in a percentage of the maximum (no-turbo) performance level.
- no_turbo: Limits the driver to selecting P-states below the turbo frequency range.
- turbo_pct: Displays the percentage of the total performance supported by hardware that is in the turbo range. This number is independent of whether turbo has been disabled or not.
- num_pstates: Displays the number of P-states that are supported by hardware. This number is independent of whether turbo has been disabled or not.

Currently, Intel P-state is used by default for supported CPUs and it has been compiled into kernel image, and acpi-cpufreq has been compiled as kernel module. Users can switch to ACPI CPUfreq by adding the following to the kernel parameter line:

```
Intel_pstate=disable
```

Use with Lenovo ThinkSystem

To set intel-pstate on a Lenovo ThinkSystem server, use the following steps:

1. Boot the server into XClarity Provisioning Manager (system setup) by pressing F1 when prompted.
2. Click the UEFI Setup menu and click **System Settings** → **Operating Modes** as shown in Figure 9 on page 14.

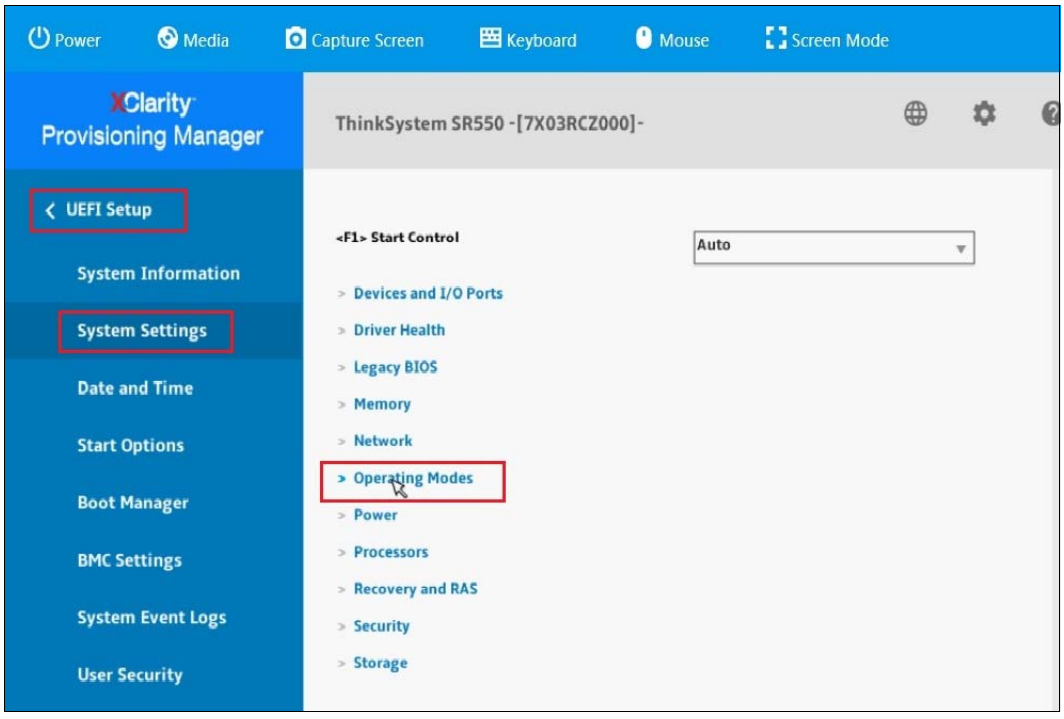

Figure 9 Navigating Provisioning Manager to Operating Modes

3. Select the following, as shown in Figure 10 on page 15:
   - Operating Mode: **Custom Mode**
   - CPU P-state Control: **Cooperative**

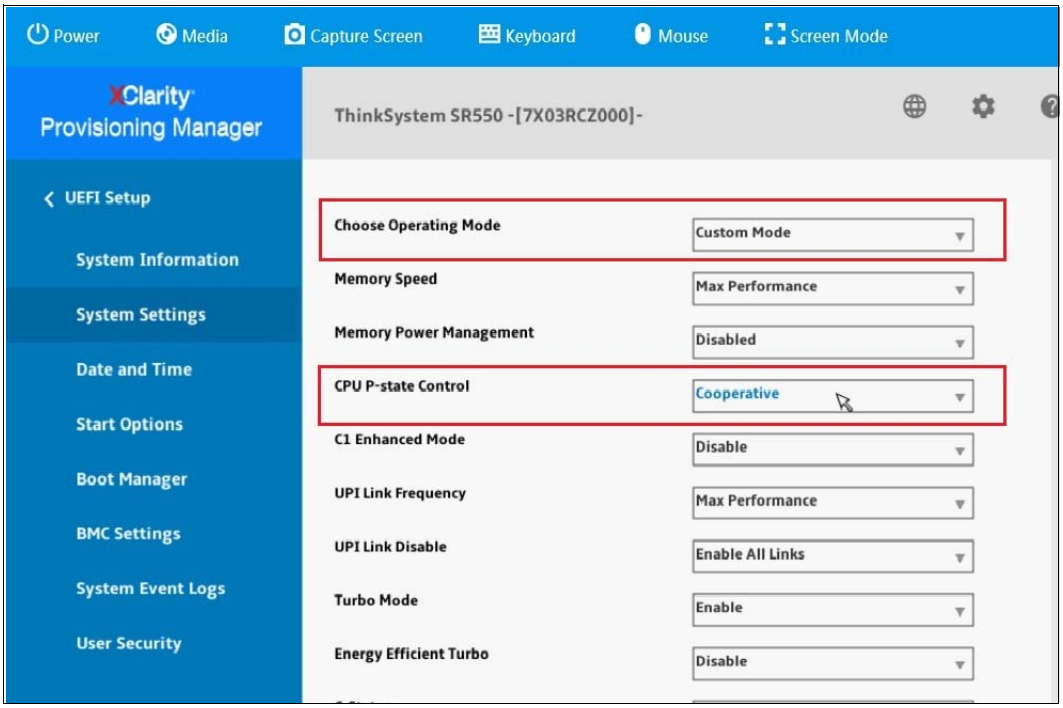

Figure 10 Operating Mode settings for intel-pstate

4. Click **Save**, exit Provisioning Manager and reboot the server.

Driver selection based on application scenarios

Nowadays, a typical standard server comes with basically all of the necessary hardware features supported in RHEL 7. The first thing to take into consideration is the main kinds of workloads for which the server will be used. Based on this information, you can decide which components to optimize for power saving.

- Web server

  A web server needs network and disk I/O. Depending on the external connection speed, 100 Mbit/s might be enough. If the machine serves mostly static pages, CPU performance might not be very important.

  Power-management choices might therefore include:

  - No adjustment of disk or network plugins.
  - ALPM turned on.
  - Ondemand governor turned on.

- Compute server

  A compute server mainly needs CPU resources. Power management choices might include:

  - Depending on the jobs and where data are stored, disk or network plugins can be adjusted; or for batch-mode systems, fully active adjustment.
  - Depending on utilization, perhaps the performance governor should be used.

- Mail server

A mail server mostly needs disk I/O and CPU resources. Power management choices might include:

- Ondemand governor turned on, because the last few CPU performance percentages are not important.
- No adjustment of disk or network plugins.
- The network speed should not be limited, because mail is often internal and can therefore benefit from a 1 Gbit/s or 10 Gbit/s link.

- File server

  File server requirements are similar to those of a mail server, but depending on the protocol used, might require more CPU performance. Typically, Samba-based servers require more CPU than NFS, and NFS typically requires more than iSCSI. Even so, users should be able to use the ondemand governor.

- Directory server

  A directory server typically has lower requirements for disk I/O, especially if equipped with enough RAM. Network latency is important, although network I/O less so. Users might consider network latency adjustment with a lower link speed, but they should test this carefully for the particular network. Therefore, the ondemand governor might be a suitable choice.

Autonomous P-states

Lenovo ThinkSystem servers also support Autonomous P-states, which Intel calls HWPM OOB mode (Hardware Power Management out-of-band mode). With this mode, the decision about which P-state to use is made entirely by system hardware (Package Control Unit, PCU) with no input from the operating system. The priority can be tuned somewhat with the power/performance UEFI setting.

Modern Intel processors use Hardware Power Management (HWPM), a new optional processor power management feature in the hardware that frees the OS from making decisions about processor frequency. HWPM allows the platform to provide information on all available constraints, allowing the hardware to choose the optimal operating point. Operating independently, the hardware uses information that is not available to software and is able to make a more optimized decision in regard to the p-states and c-states.

The Intel Xeon Scalable Family expands on HWPM by providing a broader range of states that it can affect as well as a finer level of granularity and microarchitecture observability via the PCU. Intel Xeon Scalable family processors offers the option for a collaboration between the HWPM and the operating system, known as native mode. For more information, see the *Intel Xeon Processor Scalable Family Technical Overview*, available from:

https://software.intel.com/en-us/articles/intel-xeon-processor-scalable-family-technical-overview

References

For more information, see the following documents

- Managing power consumption on Red Hat Enterprise Linux 7

  https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Power_Management_Guide

- Intel 64 and IA-32 Architectures Software Developer's Manual

  https://software.intel.com/en-us/articles/intel-sdm?wapkw=64-ia-32-architectures-software-developer-manual
- Linux kernel source code repository

  https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git
- Wikipedia ACPI global states

  https://en.wikipedia.org/wiki/Advanced_Configuration_and_Power_Interface#Global_states

Authors

Huaisheng Ye is a Linux Engineer at the Lenovo Data Center Group in Beijing, China. He has experience with Linux kernel power and memory management subsystems, and kernel drm scope. Before joining Lenovo, he worked for AMD as a graphics driver engineer. He has almost nine years work experience, and currently focuses on Linux kernel memory, storage and power management subsystems.

Special thanks to the following people for their contributions and suggestions for this project:

- Carol Cheng, Lenovo Sr. Manager for OS Enablement and Preload
- Ocean He, Lenovo Advisory Engineer for Linux Enablement
- Robert Wolford, Lenovo SESM for System x® Power Management & Efficiency

Notices

Lenovo may not offer the products, services, or features discussed in this document in all countries. Consult your local Lenovo representative for information on the products and services currently available in your area. Any reference to a Lenovo product, program, or service is not intended to state or imply that only that Lenovo product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any Lenovo intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any other product, program, or service.

Lenovo may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

Lenovo (United States), Inc.
1009 Think Place - Building One
Morrisville, NC 27560
U.S.A.
Attention: Lenovo Director of Licensing

LENOVO PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. Lenovo may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

The products described in this document are not intended for use in implantation or other life support applications where malfunction may result in injury or death to persons. The information contained in this document does not affect or change Lenovo product specifications or warranties. Nothing in this document shall operate as an express or implied license or indemnity under the intellectual property rights of Lenovo or third parties. All information contained in this document was obtained in specific environments and is presented as an illustration. The result obtained in other operating environments may vary.

Lenovo may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Any references in this publication to non-Lenovo Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this Lenovo product, and use of those Web sites is at your own risk.

Any performance data contained herein was determined in a controlled environment. Therefore, the result obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

This document was created or updated on February 27, 2018.

Send us your comments via the **Rate & Provide Feedback** form found at
http://lenovopress.com/lp0826

Trademarks

Lenovo, the Lenovo logo, and For Those Who Do are trademarks or registered trademarks of Lenovo in the United States, other countries, or both. These and other Lenovo trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by Lenovo at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of Lenovo trademarks is available on the Web at http://www.lenovo.com/legal/copytrade.html.

The following terms are trademarks of Lenovo in the United States, other countries, or both:

Lenovo®	System x®
Lenovo(logo)®	ThinkSystem™

The following terms are trademarks of other companies:

Intel, Intel SpeedStep, Pentium, Xeon, and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.