Lenovo

Understanding P-State Control on Intel Xeon Scalable Processors to Maximize Energy Efficiency

Introduces the P-state control mechanisms of Intel Xeon Processor Scalable Family

Explains how SPECpower benchmarking can be used to analyze performance

Analyzes both performance and efficiency impact among the P-state controls

Provides tuning guidelines to suit customer workloads on Lenovo ThinkSystem servers

Jenseng Chen

Robert Wolford

Abstract

“Energy Prices Continues to Rise.”
“Natural Disaster Puts Strain on Electrical Grid.”
“Economic Slowdown: Who will pay the Lighting Bill.”

These types of headlines are becoming more commonplace in recent years. Lenovo® is committed to tackling the challenges centered around power consumption by producing servers that conserve power and operate efficiently. One specific area of focus within the server is processor power.

For many workloads, the processor is the largest consumer of power within the server. Compared to older processor architectures, new generations of processors offer many more power management choices.

Understanding which choices to pick can influence many aspects of a server: workload performance, response times & latency, power consumption, and operating efficiency. If customers do not understand how the various choices operate, it can be difficult for them to pick the right ones and many times, it just becomes an “educated guess”.

This paper explains the processor active power management state (aka P-states) choices on Lenovo ThinkSystem™ servers with Intel Xeon Scalable processors and when it is best to pick each P-state choice.

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

When a server is working on daily jobs, it is desirable for the processors to complete the work requested quickly, but use power prudently. The more precisely the power is used, the more efficient the system is.

The main method of frequency control in Intel Xeon processors is P-states (performance states). Lenovo ThinkSystem servers include several methods of controlling the P-states based on a customer's preference of efficiency or performance.

The general strategy for tuning P-state control on a server for either high efficiency or high performance is as follows:

1. Tune UEFI settings
2. Tune OS settings
3. Apply power or frequency caps

After each step, measurements are taken to understand the impact of the changes. Then, once each step is optimized, the next step is performed.

SPECpower benchmark

The Standard Performance Evaluation Corporation (SPEC) has developed the SPECpower_ssj2008 benchmark suite (referred to hereafter as SPECpower) to measure the power and performance characteristics of a server and report the overall performance/watt efficiency metric of the server. SPECpower can be used to compare power and performance among different servers and serves as an industry standard toolset for measuring server efficiency.

This benchmark is targeted for use by hardware vendors, IT industry, computer manufacturers, and governments. Also, since it is an industry standard benchmark, the results from SPECpower are more credible compared to custom efficiency benchmarks.

For more information about the benchmark, see the SPECpower web page:

http://spec.org/power_ssj2008/

During the benchmark run, firstly, it runs the system under test (SUT) at the maximum throughput possible. This is determined by running the workload unconstrained for at least 3 calibration intervals. The maximum throughput is set as the arithmetic average of the throughputs achieved during the final two calibration interval runs.

The workload is then run in a controlled manner, with delays inserted into the workload stream, to obtain total throughputs of 100%, 90%, 80%, 70%, 60%, 50%, 40%, 30%, 20%, and 10% of the maximum throughput. During each of these target loads, the power characteristics of the SUT as well as the temperature are recorded.

Finally, the power characteristics and temperature are measured and recorded during an idle interval during which the SUT processes no Java transactions. When the benchmark completes, we will have the efficiency score (performance per watt), maximum performance, and power consumptions, etc for each load level.

Using the Lenovo ThinkSystem SR850 as an example, its efficiency score is 13,010[1]. This score means that the system can reach 13,010 ssj_ops per watt with the specified hardware, software and tunings documented in the full disclosure form. At 100% load level, it reached a

performance level of 10,052,756 ssj_ops and consumed 688 watts of power. At a 90% load level, the system power was 614 watts. We can infer hints from the other load levels. Its idle system power is 95.9 watts. In the later investigations, these will be key data points in the comparisons among different configurations.

The results of the SR850 tests are listed in Table 1.

Table 1 SPECpower results of the ThinkSystem SR850

Performance			**Power**	**Performance to Power Ratio**
Target Load	Actual Load	ssj_ops	Average Active Power (W)	
100%	98.2%	10,052,756	688	14,617
90%	90.5%	9,263,423	614	15,079
80%	80.0%	8,194,534	532	15,399
70%	70.0%	7,163,627	467	15,354
60%	60.0%	6,146,537	416	14,780
50%	50.0%	5,119,675	374	13,693
40%	40.0%	4,096,209	337	12,151
30%	29.9%	3,065,282	301	10,177
20%	20.0%	2,051,162	266	7,718
10%	10.0%	1,022,957	228	4,495
	Active Idle	0	95.8	0
			$\sum$ssj_ops / $\sum$power =	**13,010**

With respect to performance/watt efficiency, one of the key factors that affects efficiency is processor frequency. The processors in a server typically consume the largest portion of power for the total server. As the processor frequency is increased, the processor power generally increases exponentially which leads to higher system power. Therefore, monitoring the processor frequency is critical in analyzing the overall efficiency of the server.

However, the SPECpower benchmark suite does not monitor or report processor frequency. To collect the processor frequency while running the scenarios in this paper, Intel's Power Thermal Utility (PTU) was used.

There are many different tuning parameters available to tune for energy efficiency such as UEFI tunings, OS tunings, power or frequency capping. Since we are discussing the efficiency impact among different P-state control mechanisms, we are expecting users don't want to have any performance drop to get better efficiency in their daily work.

For customers who want to have better efficiency by reducing performance using methods such as power capping, they still can apply the same concepts from this paper and cap power or performance to suit their needs in the real scenarios. Power capping can be performed by following the steps described in "Power capping" on page 17 of the Lenovo Press paper, *Energy Efficiency Features of Lenovo ThinkSystem Servers*:

https://lenovopress.com/lp0780-energy-efficiency-features-thinksystem

[1] Lenovo ThinkSystem SR850 SPECpower_ssh2008 result can be downloaded from the web at: https://www.spec.org/power_ssj2008/results/res2017q4/power_ssj2008-20171011-00792.html

System UEFI P-state control methods

Figure 1 on page 5 shows the P-state control methods menu available in UEFI in Lenovo ThinkSystem servers. This menu can be accessed by pressing F1 while the server is booting through POST.

Alternatively, P-state control method can be set out-of-band with Lenovo's OneCLI utility which can be downloaded from the following web page:

https://support.lenovo.com/us/en/solutions/lnvo-tcli

Figure 1 P-state control in UEFI

The P-state control method you can select here are as follows:

- **Autonomous (default)**

  This mode is part of Intel's Hardware Power Management (HWPM) feature and is the default in ThinkSystem servers. With the Autonomous selection, the P-states are completely controlled by system hardware. From the operating system's perspective, there are no P-states and the OS always thinks the processor is running at its rated frequency. The processor's efficiency is totally dependent on Intel's hardware management engine (ME) implementation.

- **Legacy**

  When Legacy is selected, the processor P-states will be presented to the OS and the OS power management (OSPM) will directly control which P-state is selected. Legacy

provides slightly higher latency than Autonomous since its desired frequency is calculated by the OS. It is also useful on older OSes that don't support Cooperative mode and older hardware without HWPM support.

- **Cooperative**

  Cooperative mode is also part of Intel's HWPM feature. Cooperative mode is a combination of Autonomous and Legacy modes. Like Autonomous mode, the P-states are still controlled by the system hardware. However, in Cooperative mode, the OS is aware that different processor operating frequencies exist and it can provide hints to the system hardware for the minimum, desired, and maximum P-states.

  The system hardware does not always have to honor the OS requested P-state however. The final P-state chosen is influenced by the Power/Performance Bias setting which is configurable under the Operating Modes menu panel as shown in Figure 1 on page 5 and also the current condition of the processor (power draw, temperature, number of active cores).

  For more details about Power/Performance Bias, see "Power bias and performance bias" on page 13 of the Lenovo Press paper, *Energy Efficiency Features of Lenovo ThinkSystem Servers*:

  https://lenovopress.com/lp0780-energy-efficiency-features-thinksystem

  Newer OSes (for example, Windows Server 2016, RHEL 7.x, Ubuntu 15.x, Linux kernel 3.19 and higher) are required to take advantage of cooperative P-states. If Cooperative is selected on an older OS that doesn't support it, the system will fall back to Autonomous mode.

- **None**

  When P-states are set to None, P-states are completely disabled and the processors run at either their rated frequency or in turbo mode (if Turbo Boost is enabled). None is the best choice when the absolute lowest latency and highest performance are desired. The trade-off is that power consumption is higher.

Tuning Windows Server 2016 for energy efficiency

To have a better energy efficiency rating, you have to pay attention to both OS-level tuning and UEFI options. Windows provides the power plan mechanism to modify the maximum and minimum processor state, PCIe idle state (ASPM) and hard disk drive (HDD) turn off time in idle state. Different power plan settings result in different performance and power behavior.

The UEFI tunings include system operating behaviors such as P-state, C-state, Memory Power Control, and QPI speed. We focus on processor P-state control in this paper.

Our test configuration was as follows:

- Operating system: Windows Server 2016
- Server: Lenovo ThinkSystem SR650 with 2x 8180 processors, 12x16GB 2Rx8 DDR4 DIMMs, 1x 120GB M.2 SSD
- Java virtual machine: Oracle Java7 update 80

Power plans

Within the context of Windows Server 2016, the most significant factor for energy efficiency is the “Power Plan”. Under all P-state control mechanisms in UEFI, there are three pre-defined power plans as shown in Figure 2 on page 7.

- Balanced
- High performance
- Power saver

Users are able to change the power plan or create a new power plan on-the-fly to change the tuning parameters as needed.

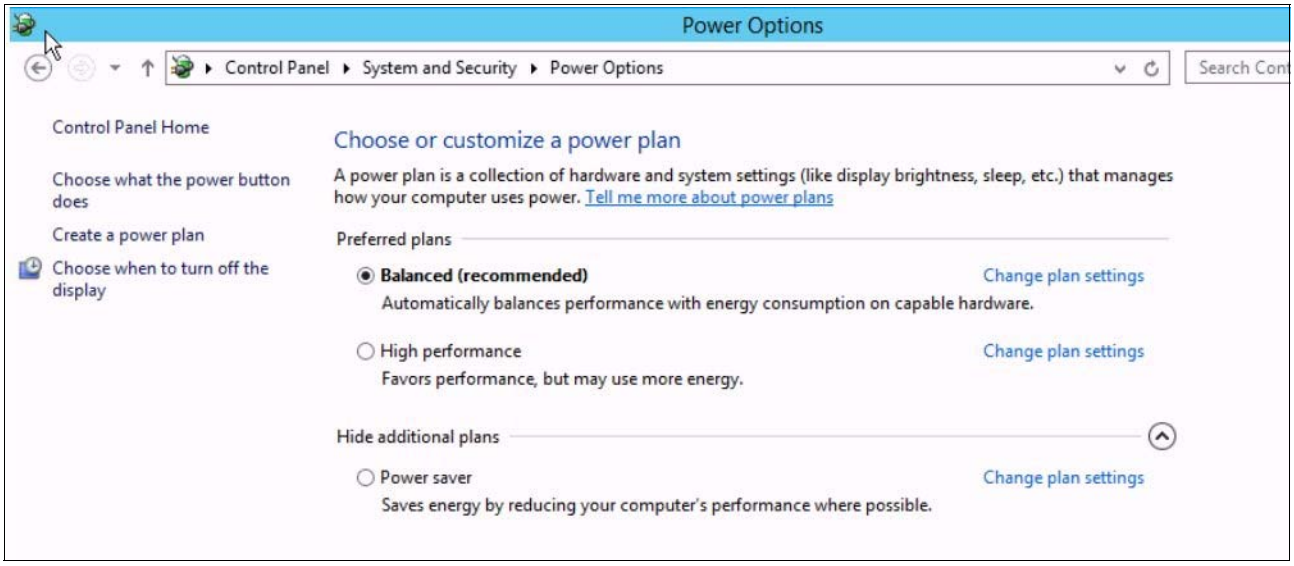

Figure 2 Windows power plans

Each power plan includes sub-options. For our benchmark testing, these sub-options were set as follows:

- Turn off the display: 1 min
- Turn off hard disk after: 1 min
- Minimum processor state: 0%
- Maximum processor state: 100%

It is possible to modify the existing power plan or to create a new power plan using an existing power plan as a template. For example, you may wish to set the minimum processor state value to 5%.

When UEFI P-state control is set to Legacy, there are three standard power plans as listed in Figure 2. From a performance point of view, both High Performance and Balanced power options are able to achieve similar performance. However, the Power Saver plan will only achieve 96% of the peak performance compared to the other two plans. This performance is shown in Figure 3 on page 8.

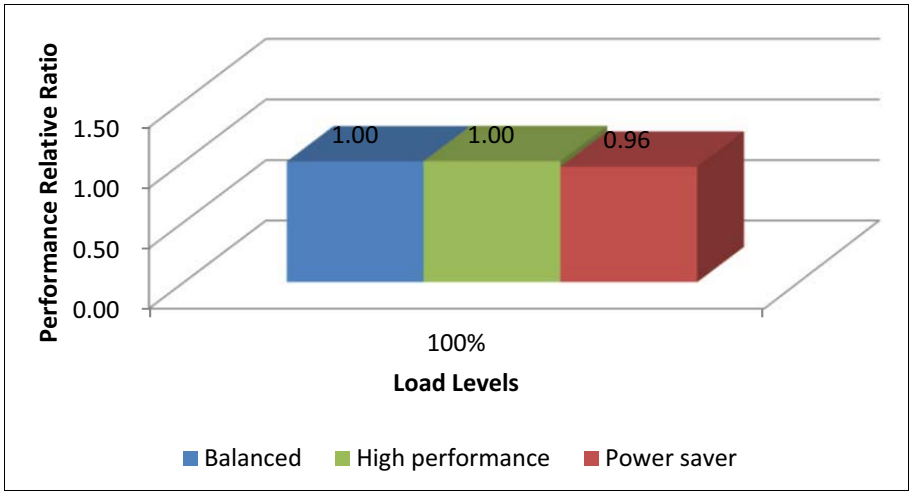

Figure 3 Comparing peak performance of the three standard power plans

The reason why the Power Saver plan's peak performance is lower is because it does not engage turbo frequencies on the processor. This is illustrated in Figure 4 which was created by running SPECpower_ssj2008 on a processor rated at 2.5 GHz. As you can see, the green line (Power Saver plan) does not exceed 2.5 GHz on the Y-axis.

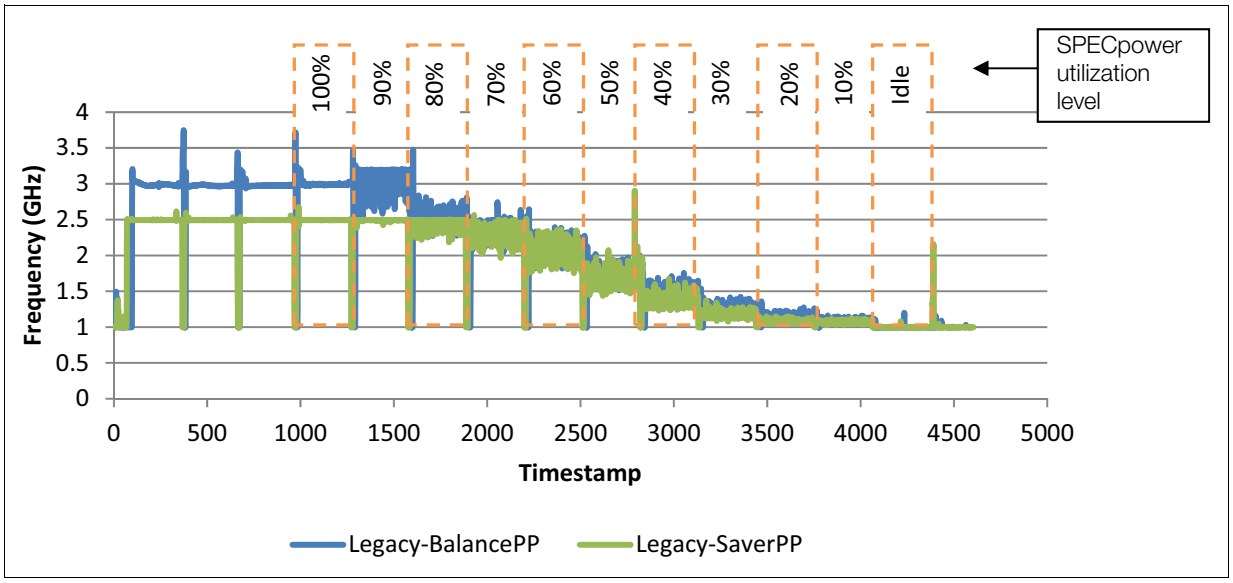

Figure 4 Comparing Balanced and Power Saver plans against different SPECpower processor utilization levels

The Power plan under Windows has an impact on performance and energy efficiency due to the running P-states. Reducing performance may yield higher energy efficiency — this is a tuning consideration.

Since we are talking about the power efficiency among P-state control mechanisms today, we would like to keep the same performance among configurations and watch for differences in efficiency. Referring to Figure 3, both High Performance and Balanced power plans yield a 4% performance boost compared to the Power Saver plan. Under Legacy P-state control, we will use the Balanced power plan instead of others in the later investigation.

If customers want to fine tune efficiency for a specific workload, they can still use the hints from this document along with the power plan (Windows platform) or governor (Linux platform) tunings, power capping and frequency limiting.

The effect of P-state control on efficiency and performance

While using the Balanced Windows power plan, let's study the energy efficiency details under different P-state control mechanisms measured by the SPECpower_ssj2008 benchmark.

As we show in Figure 5, Legacy P-state control is the most efficient selection, while selecting None yields the worst performance. Of interest is Cooperative mode, where Intel's Management Engine (aka ME) makes the final processor frequency selection based on hints from the OS. It makes sense that the relative efficiency of Cooperative mode lies between Legacy, where the OS has total P-state control, and Autonomous, where the OS has no P-state control.

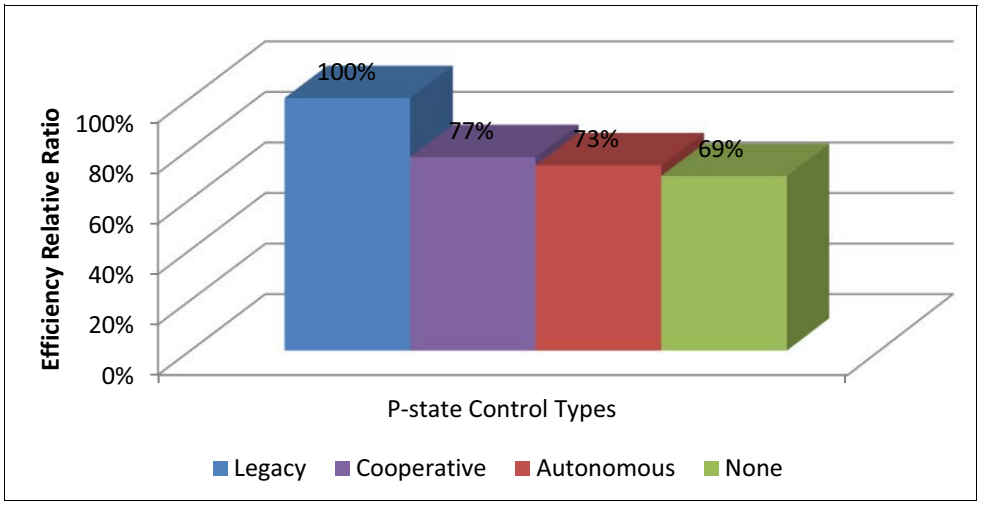

Figure 5 Relative energy efficiency of the P-state Control modes

In addition to efficiency, we also looked at peak performance. As shown in Figure 6 on page 10, the 100% performance is almost the exactly same regardless of the P-state control methods. In other words, P-state control mechanisms will negligible impact on peak performance.

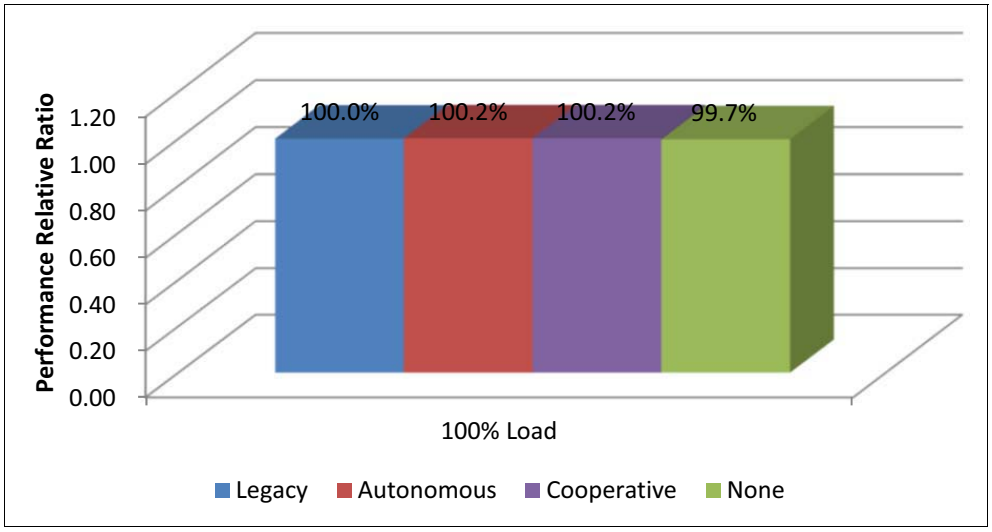

Figure 6 The effect of P-state control on peak performance

Now we perform a deeper investigation on the processor frequency trend among different P-state control mechanisms. During a SPECpower_ssj2008 run, the processor frequency drops as the SPECpower_ssj2008 load level is reduced from 100% down to 0% for all cases except when P-state control is set to None (in that mode, Turbo mode is engaged when the load increases).

For a given load level (performance level), the lower the processor frequency is, the lower the processor power is. Since efficiency is equal to performance/watt, this also means that as the processor power is reduced for a given load level, the efficiency increases. As shown in Figure 7, the highest processor frequency of P-state control in each load level is in the None mode. The slightly better (lower frequency) one comparing to None is Autonomous mode and the best one is Legacy mode. It validates what was shown with overall efficiency in Figure 5 on page 9.

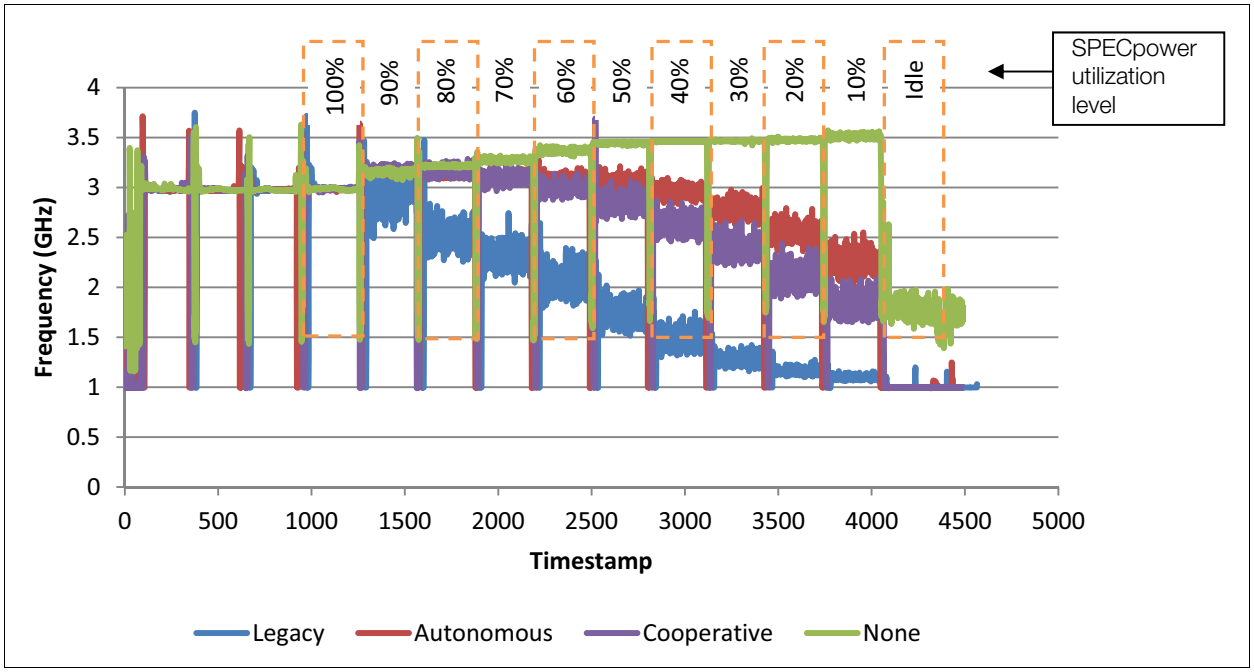

Figure 7 Comparing P-State Control modes against different SPECpower processor utilization levels

Note that when P-state control is set to None (the green line in Figure 7 on page 10), the frequency goes up at lighter loads. In all load levels, processors are trying to engage P0 (turbo state) with no regard to whether the workload is light or heavy. This occurs because when P-state control is set to None because the None mode is performance centric.

Figure 7 on page 10 also shows that the processor frequency for the 90% and 80% loads in Autonomous mode is higher than 100% load. This was unexpected. As the load was smaller at 90% & 80%, the frequency should have been lower. It implies that the design of Autonomous P-state control does not target power efficiency, but rather performance in high loads. But there is a tipping point at the 60-70% load point. At that point, the Intel Management Engine (ME) pulls down the frequency as it senses the loads are lighter.

Tuning RHEL 7.4 for energy efficiency

For an evaluation of Linux, we first used Red Hat Enterprise Linux (RHEL) 7.4. As was mentioned in the Windows tuning section, we have to pay attention to both OS level tunings and UEFI tunings to obtain better power efficiency. The Linux OS provides a governor mechanism to adjust processor frequency in order to reduce power consumption.

Our test configuration was as follows:

- Operating system: RHEL 7.4
- Server: Lenovo ThinkSystem SR650 with 2x 8180 processors, 12x 16GB 2Rx8 DDR4 DIMMs, 1x128GB SSD
- Java virtual machine: Oracle Java8 update 144

CPU frequency drivers and governors under RHEL 7.4

Newer versions of Linux support the Cooperative P-state control. RHEL7.4 is more sophisticated in its frequency control compared to previous RHEL versions. Different P-state control mechanisms result in a different P-state driver being loaded into the OS. Different P-state drivers support different governors as well.

The effect of P-state control on RHEL 7.4 performance and efficiency is described in the following subsections:

- “None P-state control”
- “Autonomous P-state control” on page 12
- “Legacy P-state control” on page 12
- “Cooperative P-state control” on page 14

None P-state control

When the P-state control is set to None in UEFI, there is no CPU frequency driver loaded into the OS. Also, no processor frequency info is implemented into the ACPI related tables. As shown in Figure 8 on page 12, we observe neither CPU frequency driver nor P-state info is available to the OS.

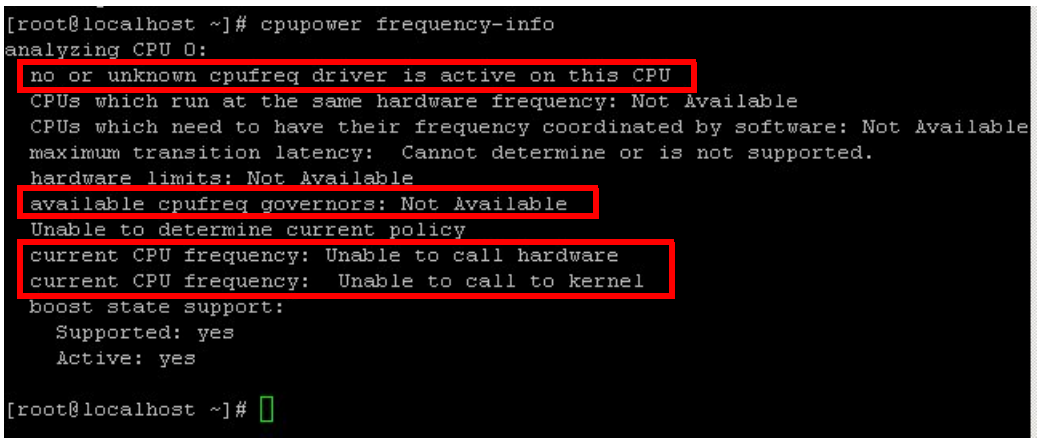

Figure 8 CPU and P-state information presented to Linux with P-state Control = None

Autonomous P-state control

When the P-state control is set to Autonomous in UEFI, there will be no CPU frequency driver loaded into OS either. As shown in Figure 9, it is similar to None, but processor frequency selection will be handled by the Intel ME alone and independent of any software including the OS.

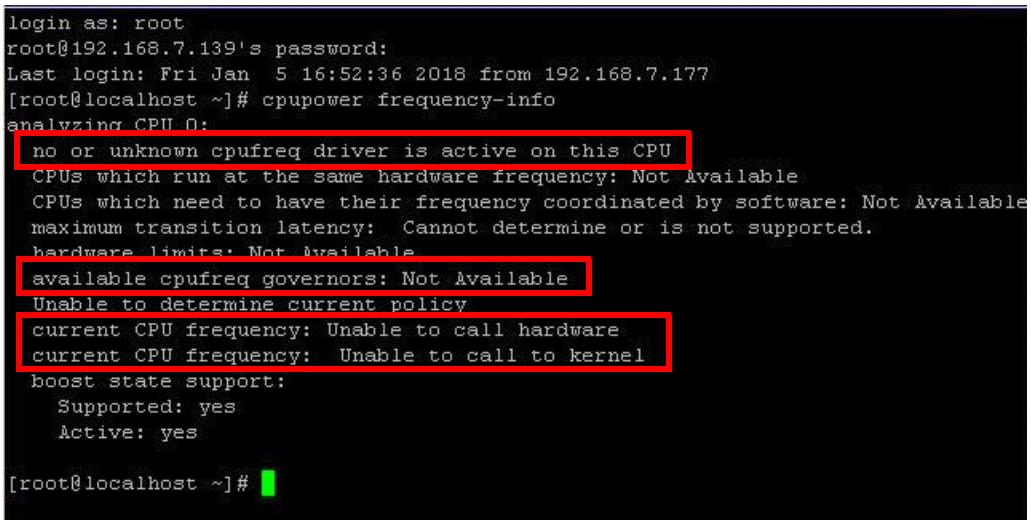

Figure 9 CPU and P-state information presented to Linux with P-state Control = Autonomous

Legacy P-state control

When the P-state control is set to Legacy in UEFI, the acpi-cpufreq driver will be loaded into the OS as shown in Figure 10 on page 13. P-states will be enabled by UEFI and five governors can be selected. The available governors are:

- conservative
- userspace
- powersave
- ondemand
- performance

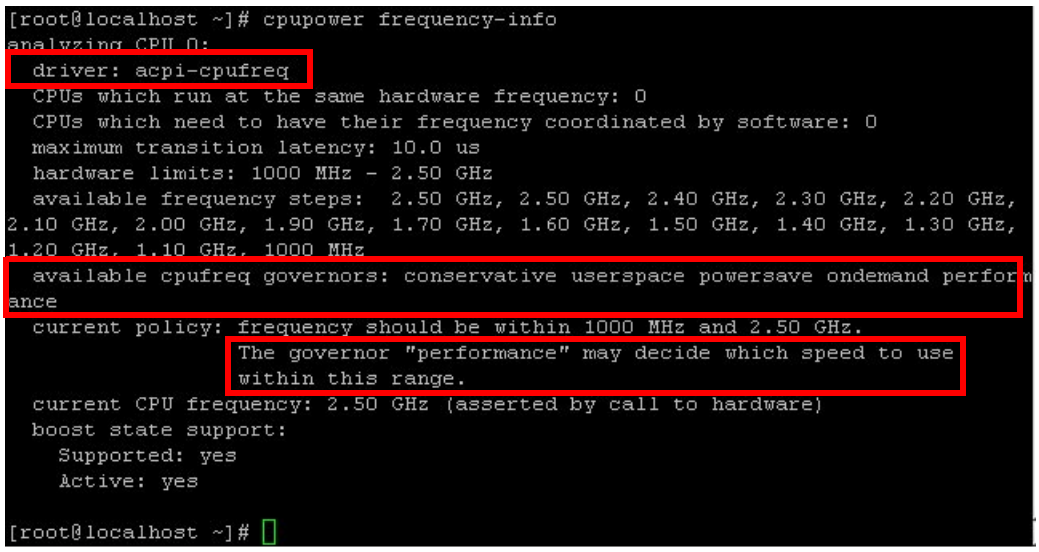

Figure 10 CPU and P-state information presented to Linux with P-state Control = Legacy

Table 2 lists each of the five available governors and the effect each has on energy efficiency.

Table 2 A comparison of the different governors in the linux acpi_cpufreq driver

Governor selection	Efficiency impact
performance	Not good for efficiency since it forces the processors to stay at the highest frequency regardless of the load. Overall, power consumption will be higher.
powersave	Not good for efficiency either since it forces the processors to stay at the lowest frequency no matter how much workload is present. Overall, performance will be lower.
ondemand	Scales the processor frequency between minimum and maximum. The final operating frequency is based on the workload. Efficiency will benefit.
conservative	Operates the same as the ondemand governor but switches between P-states more gradually. This can have a detrimental effect on system latency.
userspace	Requires a lot of intensive workload tuning to determine the optimal processor frequency that achieves the highest efficiency. It's an iterative process and is not automatic. Most datacenter administrators don't have the time to manually optimize the userspace governor.

If there is more than one governor available such as in Figure 10, the following command can be used to change the governor we want on-the-fly:

```
cpupower frequency-set -g governor
```

For example, we can use the following command to change current governor to ondemand governor:

```
cpupower frequency-set --governor ondemand
```

Based on testing with the different governors under Legacy mode as shown in Figure 11 on page 14, it's apparent that the powersave governor cannot achieve the same level of performance as the other governors. This is because the powersave governor is designed for minimum power at the expense of higher performance.

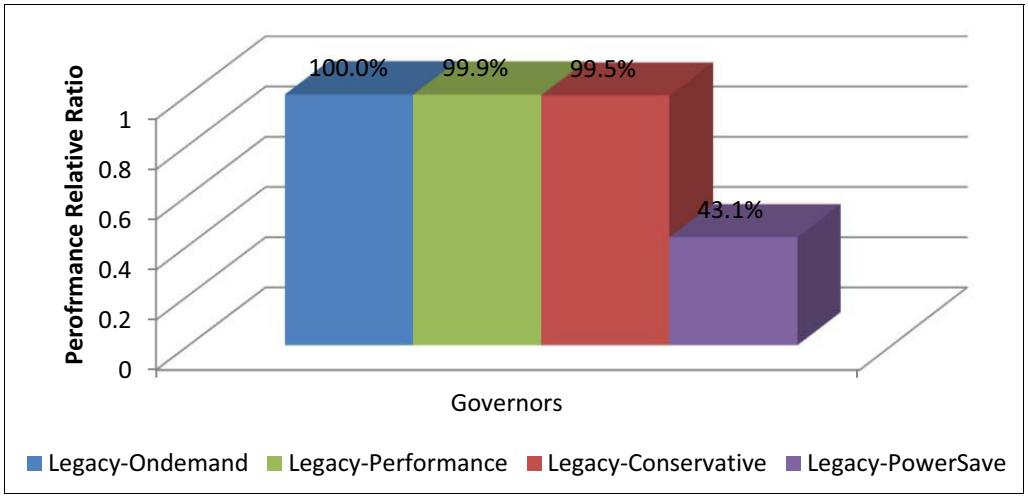

Figure 11 Comparing the performance of governors when used in Legacy P-state mode

From the viewpoint of energy efficiency as shown in Figure 12, we see that the ondemand governor is the best of all.

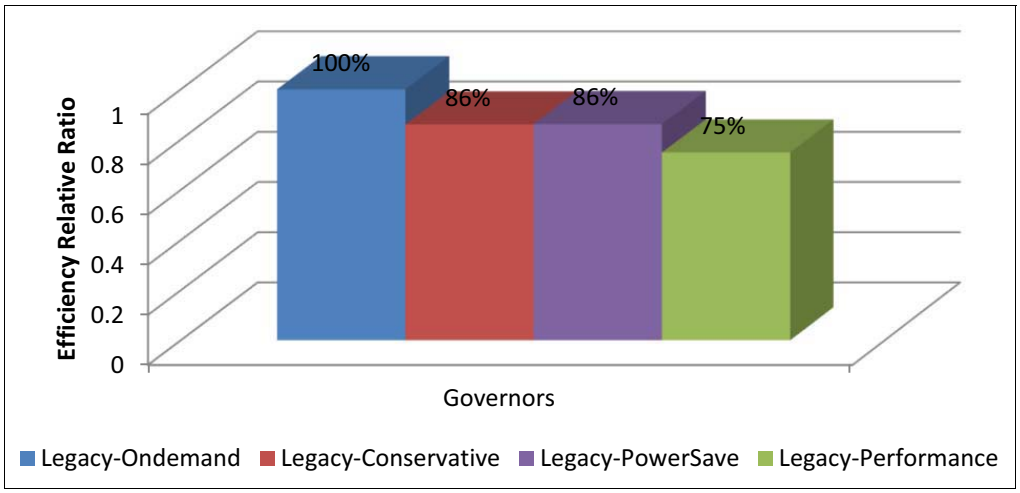

Figure 12 Comparing the energy efficiency of governors when used in Legacy P-state mode

In the later discussions, we will use the ondemand governor in Legacy mode to obtain both performance and efficiency metrics.

Cooperative P-state control

When the P-state control is set to Cooperative in UEFI, the intel_pstate driver will be loaded in the OS. The OS is aware of the processor frequencies from the minimum frequency to maximum turbo frequency as shown in Figure 13 on page 15. There are only two governors:

- performance (the default)
- powersave

```
[root@localhost ~]# cpupower frequency-info
analyzing CPU 0:
  driver: intel_pstate
  CPUs which run at the same hardware frequency: 0
  CPUs which need to have their frequency coordinated by software: 0
  maximum transition latency:  Cannot determine or is not supported.
  hardware limits: 1000 MHz - 3.80 GHz
  available cpufreq governors: performance powersave
  current policy: frequency should be within 1000 MHz and 3.80 GHz.
                  The governor "performance" may decide which speed to use
                  within this range.
  current CPU frequency: 1000 MHz (asserted by call to hardware)
  boost state support:
    Supported: yes
    Active: yes

[root@localhost ~]#
```

Figure 13 CPU and P-state information presented to Linux with P-state Control = Cooperative

From a maximum performance point of view, both governors are able to reach similar performance as shown in Figure 14. This was surprising since the powersave governor yielded less performance in Legacy P-state control (Figure 11 on page 14).

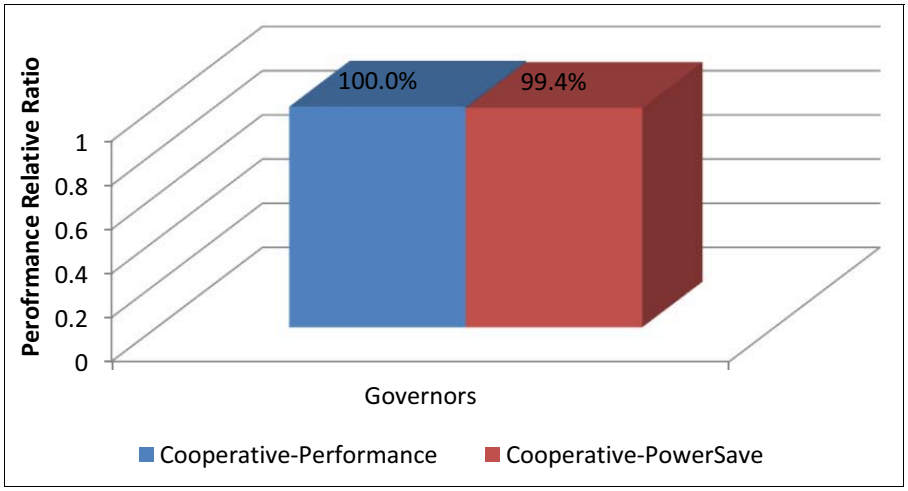

Figure 14 Comparing the performance of governors when used in Cooperative P-state mode

As we mentioned in the previous section, the control mechanism for Cooperative P-state control is the system hardware. The P-state is decided based on related performance counters and referring to OS hints. Even though the governor is set as powersave and it implies that the OS hints are not very aggressive, Intel ME does not fully honor OS hints and relies more on its performance counter.

However, the powersave governor is better than the performance governor with respect to power efficiency as shown in Figure 15 on page 16.

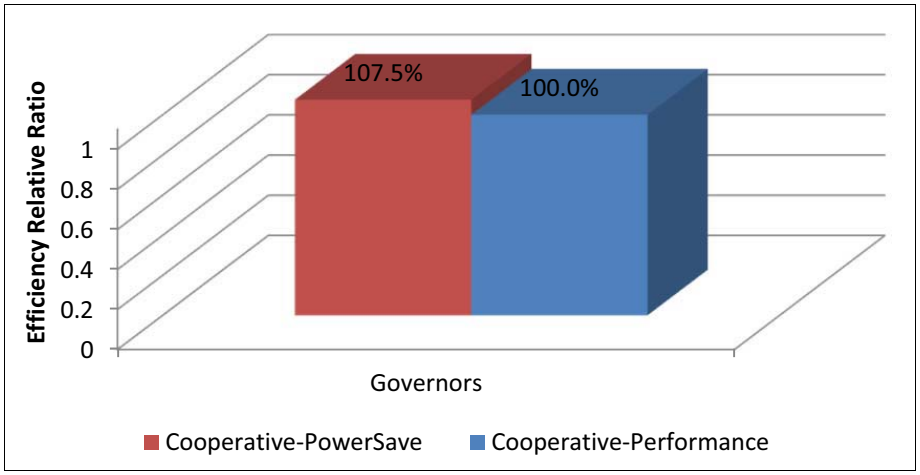

Figure 15 Comparing the energy efficiency of governors when used in Cooperative P-state mode

In the later discussions, we will use powersave governor in Cooperative mode to obtain both performance and efficiency.

The effect of P-state control on efficiency and performance

We now compare all four P-state Control modes. We have determined from the previous section that the ondemand governor is the best choice for Legacy mode, and the powersave governor is the better choice for Cooperative mode.

Looking at the best governor selection (most efficient and no performance drop) for each P-state Control and then comparing them, we see that the peak performance is almost exactly the same among all targets. This is shown in Figure 16.

Overall, when a customer chooses a specific P-state control mechanism, there is no performance drop with the correct governors and tuning.

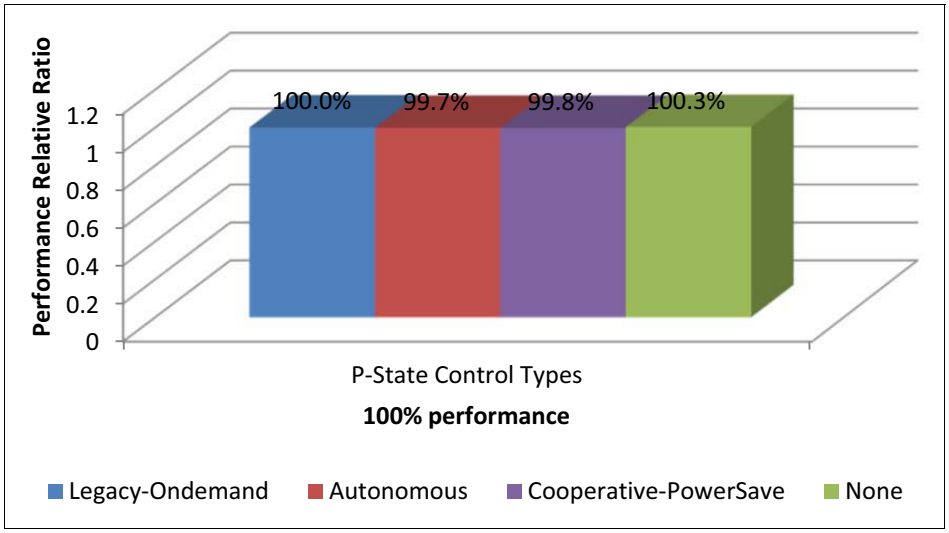

Figure 16 The effect of P-state control on peak performance

Switching to an energy efficiency point of view, we can see in Figure 17 on page 17 that we can arrive at the same conclusion as the Windows environment discussed earlier: the UEFI modes from most efficient to least efficient are:

- Legacy (most efficient)
- Cooperative
- Autonomous
- None (least efficient)

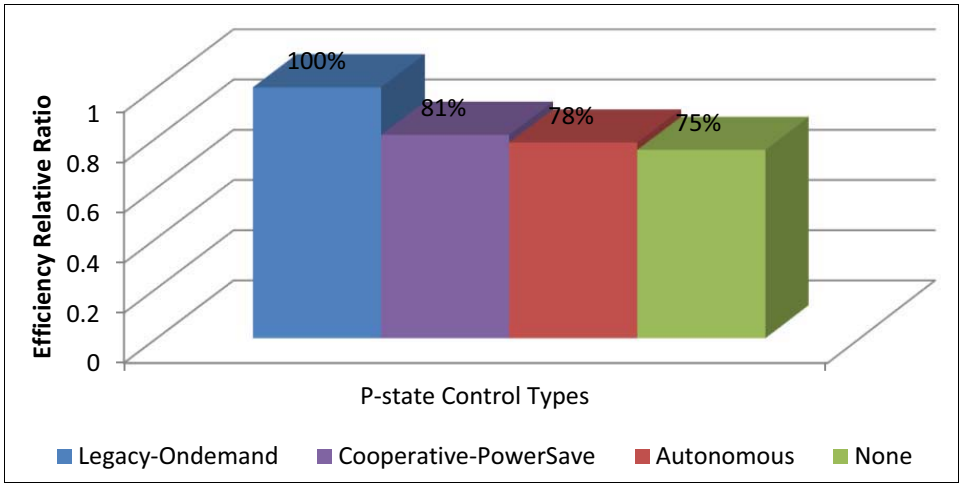

Figure 17 The effect of P-state control on energy efficiency

Looking at the power impact in each load level, it is easy to illustrate the efficiency difference mentioned above as shown in Figure 18. In the middle to light loads, there is a significant power delta between Legacy mode and any of the other three P-state control methods. During 60%~30% loads, for example, Autonomous mode consumed 1.4x times more power comparing to Legacy control. With incorrect tunings, competitive performance can be achieved, but the energy efficiency will be terrible.

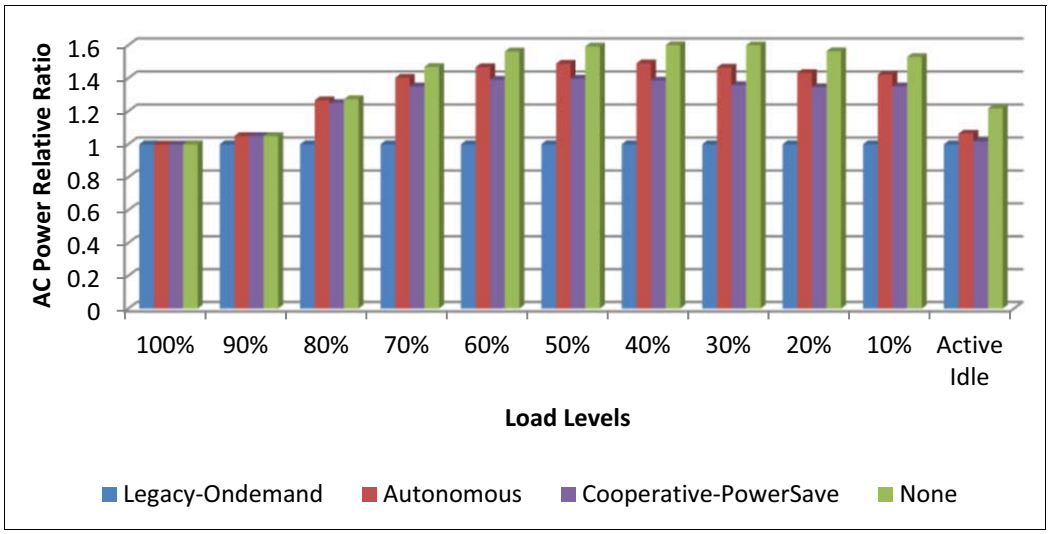

Figure 18 The effect of P-state control on energy efficiency at various load levels

From another aspect, if we observe the processor frequency impact in each load level, it is easy to illustrate the efficiency difference as well as shown in Figure 19 on page 18. Compared to Legacy P-state control, the other three P-state control methods more aggressively increase the processor frequency.

From this analysis, we conclude that Legacy P-state control is the best selection from a performance/watt point of view.

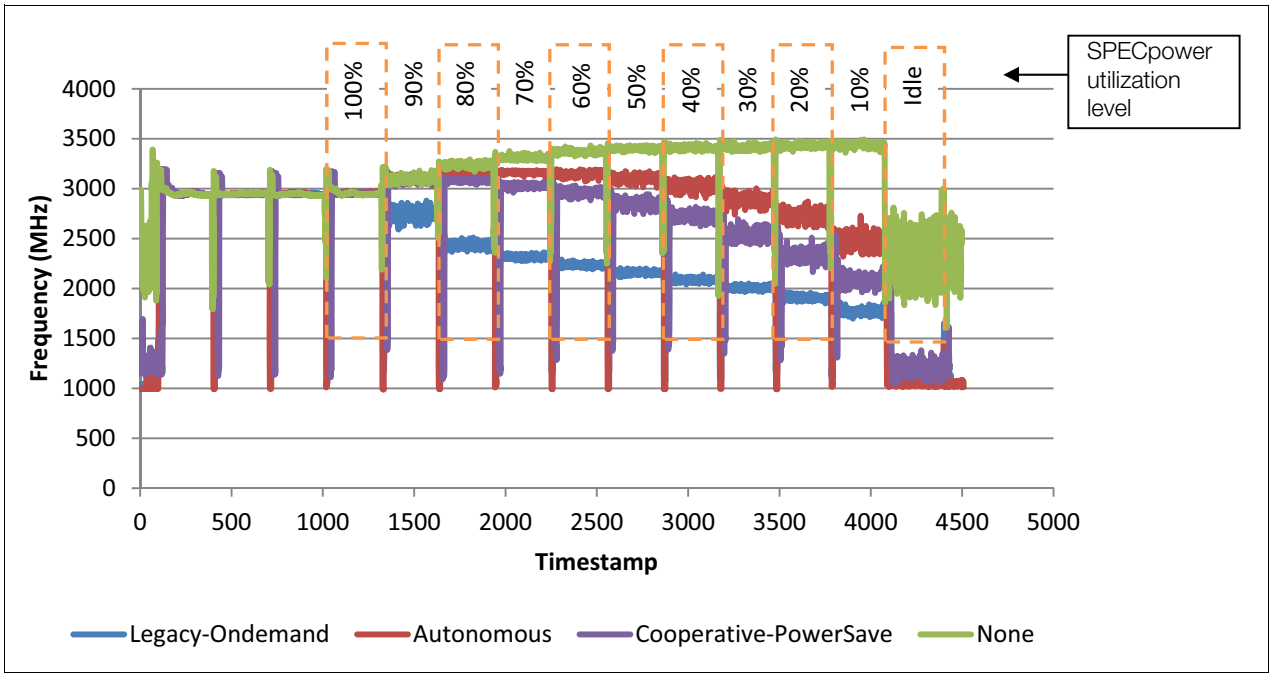

Figure 19 Comparing P-State Control modes against different SPECpower processor utilization levels

Tuning SLES 12 SP4 for energy efficiency

For a further evaluation of Linux, we also evaluated SUSE Linux Enterprise Server 12 SP4. As with other operating systems, special attention should be paid to both OS level tunings and UEFI tunings to obtain optimal power efficiency. SLES provides CPU frequency drivers and corresponding governors to adjust processor frequency in order to reduce power consumption.

The test configuration was as follows:

- Operating system: SLES 12 SP4
- Server: Lenovo ThinkSystem SR650 with 2x 8280 processors, 12x 16GB 2Rx8 DDR4 DIMMs, 1x 128GB SSD
- Java virtual machine: Oracle Java7 update 80

CPU frequency drivers and governors under SLES 12 SP4

SLES 12 SP4 is formally supported on second-generation Intel Xeon Scalable Processors. What we learned from “Tuning RHEL 7.4 for energy efficiency” on page 11 is that there are different P-state control mechanisms in UEFI and different P-state drivers and governors in the OS.

The effect of P-state control on SLES 12 SP4 performance and efficiency is described in the following sections:

Legacy P-state control

When the P-state control is set to Legacy in UEFI, the intel_pstate driver will be loaded into the OS by default as shown in Figure 20. Its implementation is different compared to RHEL 7.4 described in “Tuning RHEL 7.4 for energy efficiency” on page 11.

With the intel_pstate driver, the hardware registers (APERF, MPERF) and algorithms were adopted rather than the ACPI tables exposed by UEFI. It is able to change the OS power management algorithm to acpi-cpufreq governor as needed. Unlike acpi-cpufreq power management (PSS objects in the ACPI tables), intel_pstate always exposes the entire range of available P-states, including the whole turbo range, to the cpufreq core to generic scaling governors.

The available governors are performance and powersave under the intel_pstate driver. They are not generic (legacy) scaling governors, but their names are the same as the names of some of the legacy governors. Even more confusing, they generally do not work in the same way as the generic governors they share the names with. For example, the powersave P-state selection algorithm provided by intel_pstate driver is not a counterpart of the generic powersave governor.

```
cybrog:~ # cpupower frequency-info
analyzing CPU 0:
  driver: intel_pstate
  CPUs which run at the same hardware frequency: Not Available
  CPUs which need to have their frequency coordinated by software: Not Available
  maximum transition latency:  Cannot determine or is not supported.
  hardware limits: 1000 MHz - 4.00 GHz
  available cpufreq governors: performance
  current policy: frequency should be within 1000 MHz and 4.00 GHz.
                  The governor "powersave" may decide which speed to use
                  within this range.
  current CPU frequency: Unable to call hardware
  current CPU frequency: 1.00 GHz (asserted by call to kernel)
  boost state support:
    Supported: yes
    Active: yes
```

Figure 20 CPU, P-state driver and governors with P-state Control = Legacy under SLES 12 SP4

From the graph shown in Figure 21 on page 20, the interesting thing is that the powersave governor will not harm any performance -- the performance ratio between Legacy-intel_pstate-performance and Legacy-intel_pstate-powersave are similar (100%). Comparing the graph to the RHEL equivalent in Figure 12 on page 14 of RHEL 7.4, it is easy to see that the powersave governor implemented by the intel_pstate driver is totally different from the acpi_cpufreq one. Powersave is not a universal algorithm.

Although the peak performance between the two governors are close, the efficiency is different. The powersave governor is better than the performance one. The linux kernel selects the maximum P-state it is allowed to use in the performance governor. With powersave, however, the P-state selection algorithm is based on the values read from the APERF and MPERF. The provide hints about utilization to select a suitable P-state.

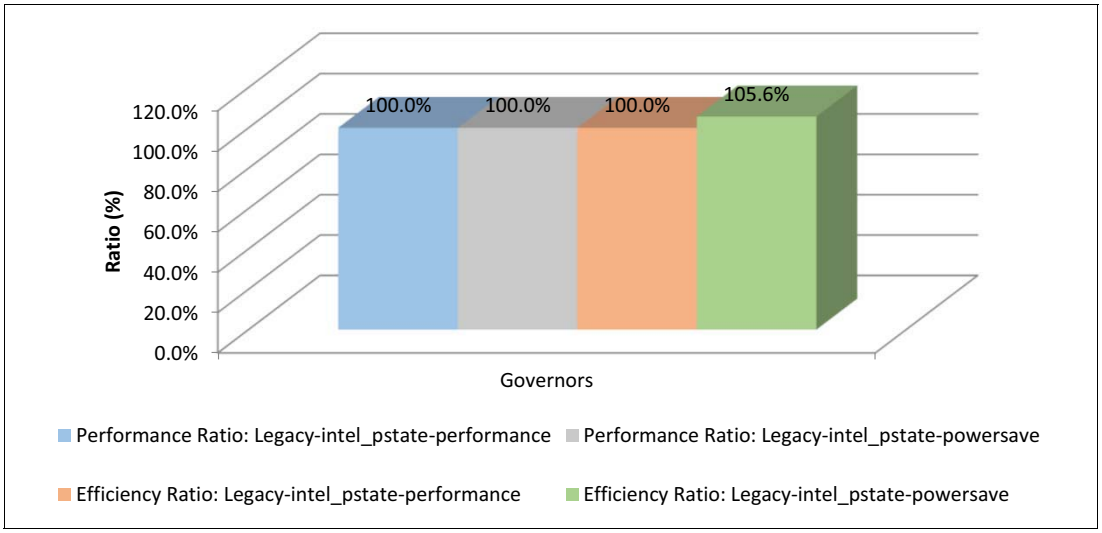

Figure 21 The performance and efficiency ratio among governors with UEFI P-state Control = Legacy and intel_pstate driver under SLES 12 SP4.

As we discuss later in “The effect of P-state control on efficiency and performance” on page 28 we will use powersave governor under the intel_pstate driver in Legacy mode to obtain both performance and efficiency.

If the acpi-cpufreq driver is loaded instead of intel_pstate driver, we may have to disable intel_pstate driver in grub. Versions of Linux with grub2 support provide more sophisticated steps to make system configuration consistent. First, we need to disable intel_pstate in /etc/default/grub file as shown in Figure 22 on page 21.

```
# Uncomment to set your own custom distributor. If you leave it unset or empty,
the default
# policy is to determine the value from /etc/os-release
GRUB_DISTRIBUTOR=
GRUB_DEFAULT=saved
GRUB_HIDDEN_TIMEOUT=0
GRUB_HIDDEN_TIMEOUT_QUIET=true
GRUB_TIMEOUT=8
GRUB_CMDLINE_LINUX_DEFAULT="resume=/dev/sda2 splash=silent quiet showopts"
GRUB_CMDLINE_LINUX="intel_pstate=disable"
# Uncomment to automatically save last booted menu entry in GRUB2 environment
```

Figure 22 Disabling intel_pstate in grub

Then, we need update bootable GRUB to make the system consistent to the related configuration settings with the command as shown in Figure 23.

```
grub2-mkconfig -o /boot/efi/EFI/sles/grub.cfg
```

```
cybrog:~/ptu # grub2-mkconfig -o /boot/efi/EFI/sles/grub.cfg
Generating grub configuration file ...
Found theme: /boot/grub2/themes/SLE/theme.txt
Found linux image: /boot/vmlinuz-4.12.14-94.41-default
Found initrd image: /boot/initrd-4.12.14-94.41-default
done
cybrog:~/ptu #
```

Figure 23 grub2-mkconfig command

The modifications will take effect after the system reboots. We then observe that the acpi-cpufreq driver is enabled as in Figure 24 on page 21. There are three available governor options:

- ondemand
- performance
- schedutil

The ondemand algorithm is enabled by default. An interesting thing is there is no powersave governor like RHEL 7.4 as shown in Figure 11 on page 14.

```
cybrog:~ # cpupower frequency-info
analyzing CPU 0:
  driver: acpi-cpufreq
  CPUs which run at the same hardware frequency: Not Available
  CPUs which need to have their frequency coordinated by software: Not Available
  maximum transition latency: 10.0 us
  hardware limits: 1000 MHz - 2.70 GHz
  available frequency steps:  2.70 GHz, 2.70 GHz, 2.60 GHz, 2.50 GHz, 2.30 GHz,
2.20 GHz, 2.10 GHz, 2.00 GHz, 1.80 GHz, 1.70 GHz, 1.60 GHz, 1.50 GHz, 1.40 GHz,
1.20 GHz, 1.10 GHz, 1000 MHz
  available cpufreq governors: ondemand performance schedutil
  current policy: frequency should be within 1000 MHz and 2.70 GHz.
                  The governor "ondemand" may decide which speed to use
                  within this range.
  current CPU frequency: 1000 MHz (asserted by call to hardware)
  boost state support:
    Supported: yes
    Active: yes
```

Figure 24 CPU and acpi-cpufreq P-state driver and governors with P-state Control = Legacy under SLES 12 SP4

In Table 2 on page 13 we described the ondemand and performance governors. Schedutil is the newest Cpufreq governor introduced in Linux 4.7 as an alternative to ondemand and

performance. What makes schedutil different and interesting is that it makes use of CPU scheduler utilization data for its decisions about CPU frequency control.

The main formula to schedutil governor is:

```
next_freq = 1.25 * curr_freq * util / max *
```

The coefficient 1.25 corresponds to the frequency tipping point a (util / max) = 0.8.

You can lear more about the schedutil frequency scaling governor for the Linux kernel in the following YouTube video:

https://www.youtube.com/watch?v=b-RrZi2AHXs

Let's walkthrough what will be happen:

1. Assume the system utilization is sampled and it is at 80%
2. If the system utilization remains at 80%, the next_freq is the same as current_freq, there is no need to be change anything. Next_freq and current_freq remain unchanged.
3. If the system utilization becomes higher than 80%, the next_freq will be higher than current_freq. Due to next_freq being higher, more instructions will get executed in a given period of time. And, if the workload remains constant, the system utilization will drop the next time it is sampled.
4. If the system utilization becomes lower than 80%, the next_freq will be lower than current_freq. Due to next_freq being lower, less instructions will get executed in a given period of time. And, if the workload remains constant, the system utilization will increase the next time it is sampled.

80% CPU utilization was selected as the tipping point because it was likely deemed to be the most efficient by Linux developers. To achieve good power and performance, trying to keep the system utilization at 80% is the best way. In the schedutil formula, it is similar to the concept used in Lenovo Efficiency Mode[2] released by Lenovo several years ago. Essentially, the CPU frequency can scale up and down around a sweet spot for system utilization.

Due to a newer Linux kernel in SLES 12 SP4, the OS power management behaviors are different between RHEL 7.4 and SLES 12 SP4. Based on testing with the different governors under Legacy mode as shown in Figure 24 on page 21 there is no powersave governor. Per the Figure 25, there is no performance shortage with the new schedutil algorithm compared to other two governors. However, from the viewpoint of energy efficiency, schedutil governor is much better, as will be demonstrated next.

[2] See the paper *Power Management with Lenovo Efficiency Mode*, https://lenovopress.com/lp0548

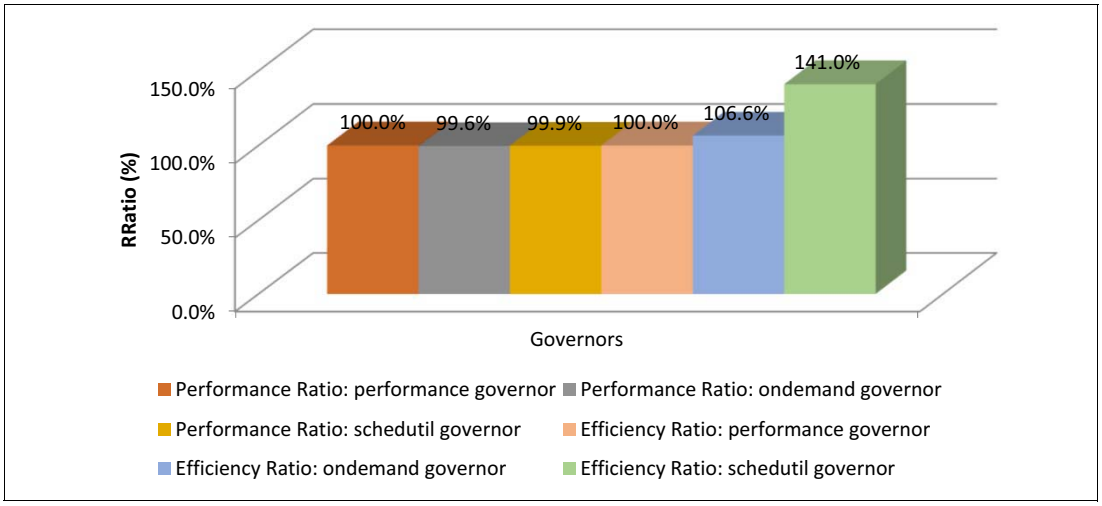

Figure 25 Comparing to the performance and efficiency of governors when used in Legacy P-state mode

Let's look inside to have more details in Figure 26, the performance governor keeps the CPU frequency as high as possible. When the loading is reduced, the processor is boosted to a higher CPU frequency. This is because the performance governor is targeted to maximize absolute performance with no consideration on power consumption. The ondemand governor is shown second. The CPU frequency associated with it is lower than with the performance governor, but still much higher than schedutil. It is quite easy to see why schedutil governor introduced in the new Linux kernel is optimized more for power efficiency versus raw performance.

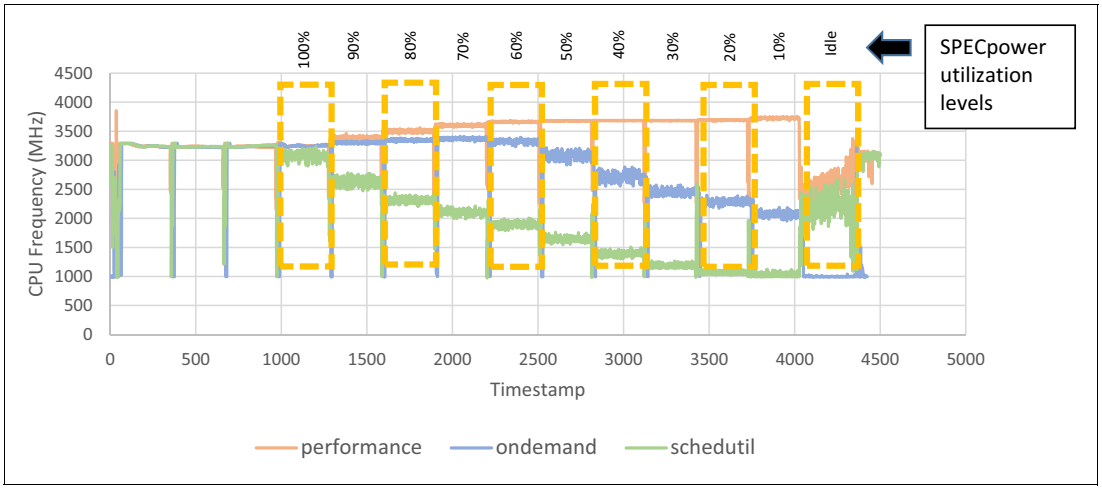

Figure 26 Frequency trend of governors when used in Legacy P-state mode

Because utilization monitoring is the main concept behind schedutil, it is interesting to study the underlying operation. Referring to Figure 27, when we consider a utilization point of view, schedutil does a much better job at keeping the CPU highly utilized. In other words, for a given CPU frequency, schedutil squeezes the same performance from the cores but, since the frequency is lower, it does it at a lower power level. In schedutil, CPU frequency and utilization are opposites. Therefore, the CPU frequency will be lower when its utilization is higher.

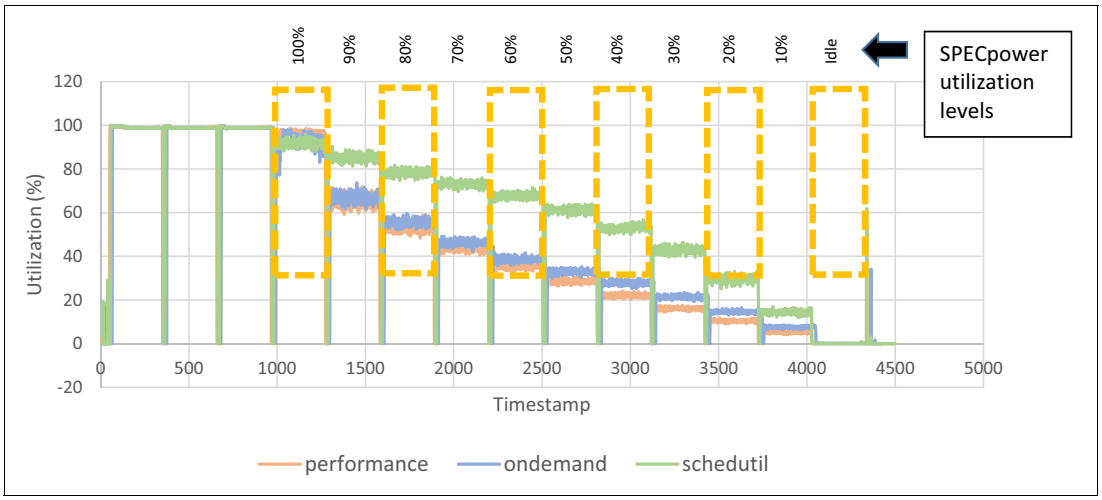

Figure 27 Utilization trend of governors when used in Legacy P-state mode

In “The effect of P-state control on efficiency and performance” on page 28, we will use the schedutil governor under acpi-cpufreq driver in Legacy mode to obtain both performance and efficiency metrics.

None P-state control

When the P-state control is set to None in UEFI, UEFI will not initialize any ACPI tables and objects. Without ACPI support, SLES 12 SP4 loads intel_pstate driver by default. It is different than what we observed in legacy RHEL 7.4 -- as described in “None P-state control” on page 11, None P-state control in RHEL 7.4 does not load drivers, whereas SLES 12 SP4 loads the intel_pstate driver.

Referring to Figure 28, there are two governors: performance and powersave. Both the performance and efficiency metrics are the same between the performance and powersave algorithms shown in Figure 29 on page 25.

```
cybrog:~ # cpupower frequency-info
analyzing CPU 0:
  driver: intel_pstate
  CPUs which run at the same hardware frequency: Not Available
  CPUs which need to have their frequency coordinated by software: Not Available
  maximum transition latency:  Cannot determine or is not supported.
  hardware limits: 1000 MHz - 4.00 GHz
  available cpufreq governors: performance
  current policy: frequency should be within 1000 MHz and 4.00 GHz.
                  The governor "powersave" may decide which speed to use
                  within this range.
  current CPU frequency: Unable to call hardware
  current CPU frequency: 3.80 GHz (asserted by call to kernel)
  boost state support:
    Supported: yes
    Active: yes
```

Figure 28 CPU and P-state information presented to Linux with P-state Control = None

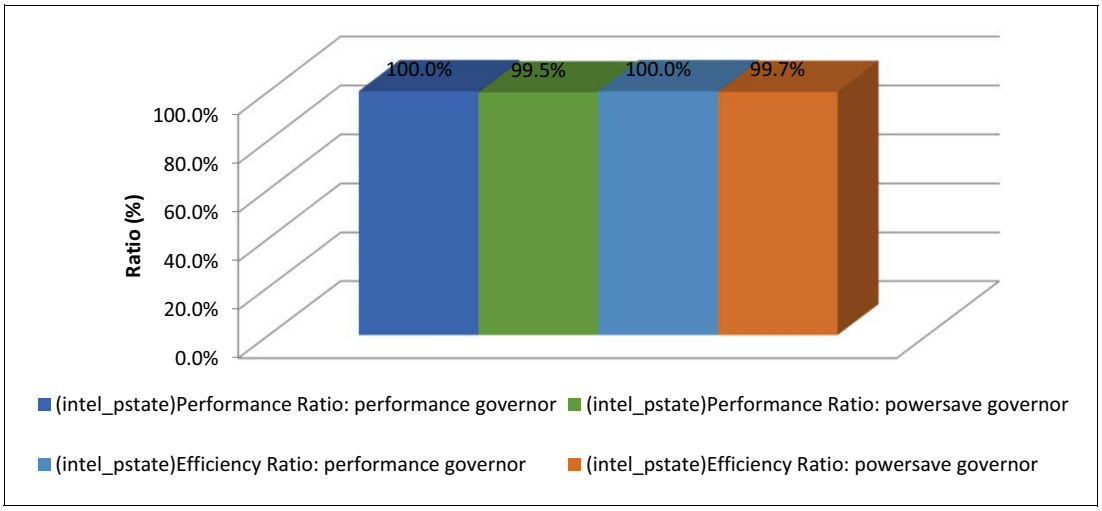

Figure 29 Performance and Efficiency presented to Linux with P-state Control = None

Breaking down the details, although the intel_pstate driver supports two power management algorithms, their frequency trends are the same. It means their power performance is the same as well.

When we change the CPU frequency driver to acpi-cpufreq, the driver can't be loaded since UEFI did not expose any ACPI tables to the OS. There are no power management mechanisms due to no p-state driver. Therefore, we don't need to consider it in later comparisons.

Setting P-state control to None was developed by Lenovo to maximize CPU performance and minimize latency by eliminating P-state transitions, regardless of workload. Figure 30 shows the CPU frequency increases to achieve maximum performance while the load is reduced.

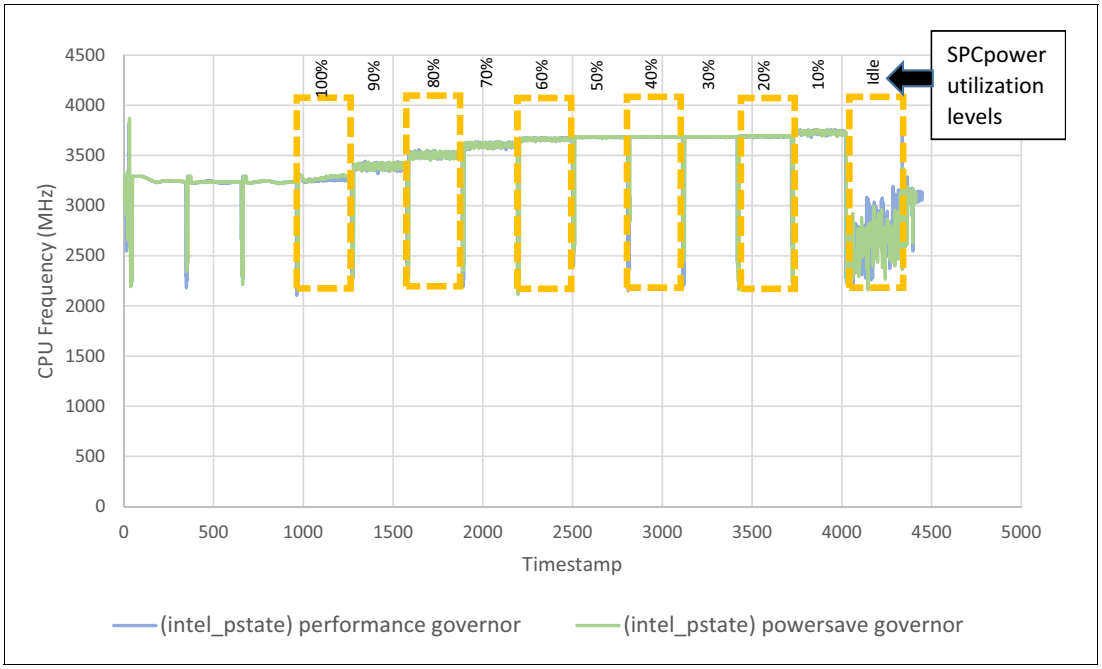

Figure 30 CPU frequency trend presented to Linux with P-state Control = None

There is limited impact on the governors. In “The effect of P-state control on efficiency and performance” on page 28, we will use the powersave governor in None mode to obtain both performance and efficiency.

Autonomous P-state control

When the P-state control is set to Autonomous in UEFI, there will be no CPU frequency driver loaded into the OS, just like when P-states are set to None in UEFI. As shown in Figure 31, it is similar to None, but processor frequency selection will be handled by Intel ME alone and independent of any software in the OS.

```
cybrog:~ # cpupower frequency-info
analyzing CPU 0:
  no or unknown cpufreq driver is active on this CPU
  CPUs which run at the same hardware frequency: Not Available
  CPUs which need to have their frequency coordinated by software: Not Available
  maximum transition latency:  Cannot determine or is not supported.
  hardware limits: Not Available
  available cpufreq governors: Not Available
  Unable to determine current policy
  current CPU frequency: Unable to call hardware
  current CPU frequency:  Unable to call to kernel
  boost state support:
    Supported: yes
    Active: yes
```

Figure 31 CPU and P-state information presented to Linux with P-state Control = Autonomous

Cooperative P-state control

When the P-state control is set to Cooperative in UEFI, the intel_pstate driver will be loaded in the OS by default. The OS is aware of the processor frequencies from the minimum frequency to the maximum turbo frequency as shown in Figure 32.

In Cooperative mode, there are only two governors:

- performance
- powersave (the default)

```
cybrog:~ # cpupower frequency-info
analyzing CPU 0:
  driver: intel_pstate
  CPUs which run at the same hardware frequency: Not Available
  CPUs which need to have their frequency coordinated by software: Not Available
  maximum transition latency:  Cannot determine or is not supported.
  hardware limits: 1000 MHz - 4.00 GHz
  available cpufreq governors: performance
  current policy: frequency should be within 1000 MHz and 4.00 GHz.
                  The governor "powersave" may decide which speed to use
                  within this range.
  current CPU frequency: Unable to call hardware
  current CPU frequency: 1000 MHz (asserted by call to kernel)
  boost state support:
    Supported: yes
    Active: yes
```

Figure 32 CPU and P-state information presented to Linux with P-state Control = Cooperative

From a maximum performance point of view, both governors are able to reach similar performance as shown in Figure 33 on page 27. Its design is the same as mentioned in the governors with the intel_pstate driver under UEFI Legacy P-state control. It also proved that the performance and powersave governors are not the same as the ones implemented in Legacy acpi-cpufreq drivers as was described previously.

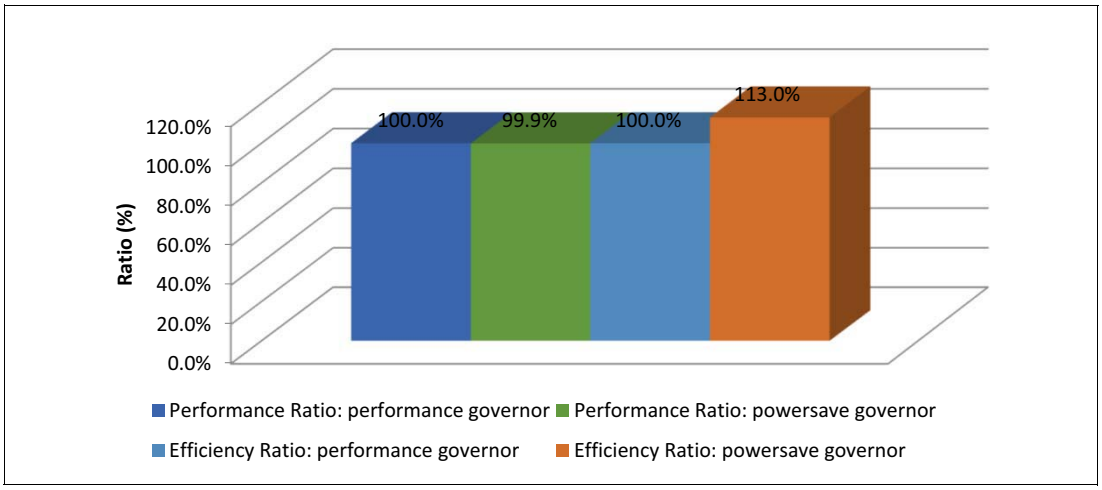

Figure 33 Comparing the performance and efficiency of governors when used in Cooperative P-state mode

The control mechanism for Cooperative P-state control is the system hardware. The P-state is decided based on related performance counters. The OS also provides hints and suggestions, however the system hardware ultimately decides the final P-state.

Since the decided P-state depends on OS hints, the powersave governor is better than the performance governor with respect to power efficiency as shown in Figure 33. Due to the design of performance governor, the processor will raise up its maximum frequency as shown in Figure 34, thereby making it less efficient.

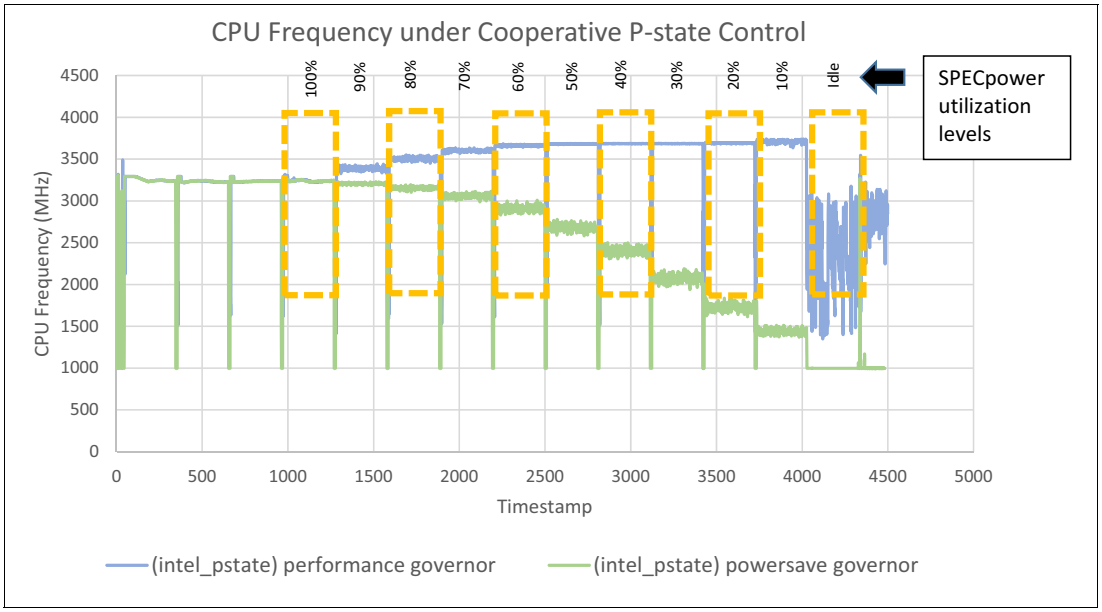

Figure 34 Comparing the energy efficiency of governors when used in Cooperative P-state mode

If we switch the CPU frequency driver to acpi-cpufreq, the driver does not load since UEFI does not enumerate P-states and T-states in the ACPI related tables such as _PCT, _PPC, _PSS, _PTC, _TSS, and _TPC entries in Cooperative P-state Control. There is no CPU frequency selecting algorithm to handle P-state transition and we don't consider it in our analysis in the next section. We will use powersave governor in Cooperative mode to obtain both performance and efficiency.

The effect of P-state control on efficiency and performance

In “CPU frequency drivers and governors under SLES 12 SP4” on page 19, we gave a detailed walkthrough on the relationships among P-state modes, CPU frequency drivers and the corresponding governors. We know the best power management implementations in each P-state control. We can now compare all four UEFI P-state control modes.

We have determined that the best combinations for optimal power efficiency under each UEFI P-state mode are:

- Legacy P-state Control: The schedutil governor is the best choice under acpi-cpufreq, and the powersave governor is the best under intel_pstate.
- None P-state Control: Both powersave and performance governors yield the same efficiency. Powersave was selected as the point of comparison.
- Autonomous P-state Control: Since P-state control is entirely in hardware, neither acpi-cpufreq nor intel_pstate drivers are loaded into the OS. We keep the default condition for our later discussion.
- Cooperative P-state Control: The powersave governor is the best with the intel_pstate driver. The acpi-cpufreq driver is not available with cooperative mode.

Looking at the best governor selection mentioned above, there is almost no performance delta as shown previously. Because of the implementation in the new Linux kernel, there is no powersave governor under acpi-cpufreq driver under Legacy P-state Control. Therefore, we only need to be concerned with efficiency. As shown in Figure 35, the best choice is schedutil governor under acpi-cpufreq driver while setting P-state Control to Legacy.

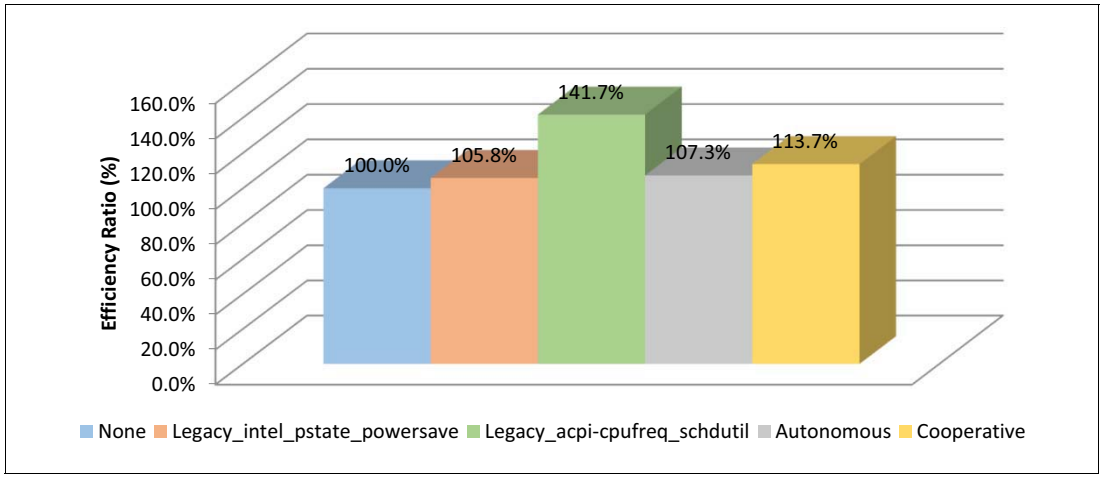

Figure 35 The effect of P-state control on power efficiency

Looking at the efficiency impact in each load level as Figure 36 on page 29 and using “None” as the baseline, it is easy to see that the combination UEFI legacy P-state mode + acpi_cpufreq driver + schedutil governor is the best choice for performance per watt efficiency. Overall, that combination is 41.7% better than the baseline.

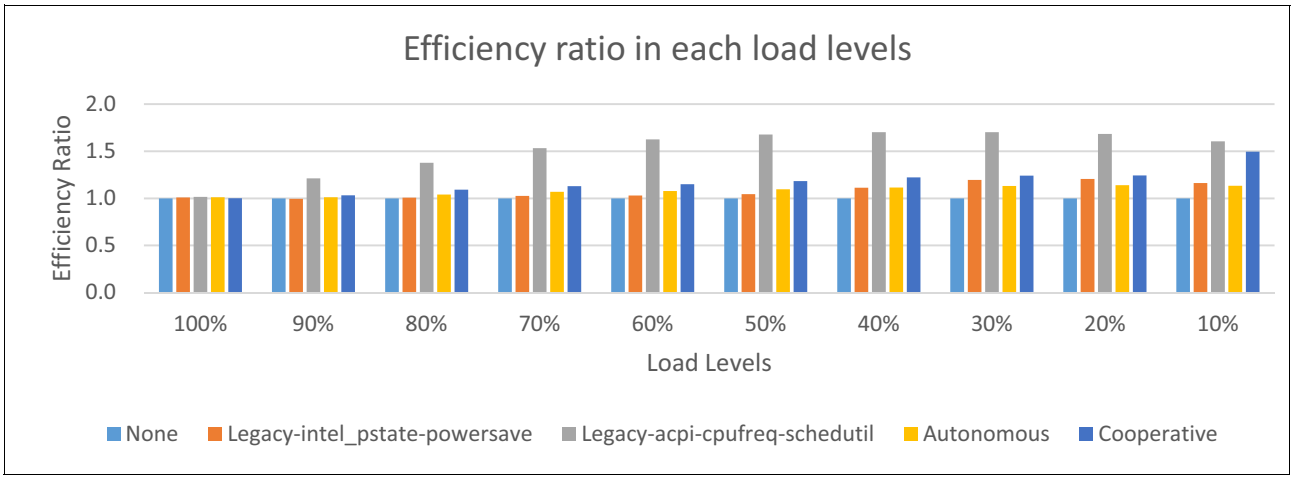

Figure 36 The effect of P-state control on energy efficiency at various load levels

Looking at the 50% load level as an example, the efficiency of schedutil governor is 1.68 times higher compared to the 50% load of None P-state control. Since it is an index of efficiency, larger is better.

We can glean some hints from Figure 36:

- Schedutil governor is the best in all load levels except 100%. To reach the system maximum performance, there is no room to save power in 100% load and each power management implementation is almost the same.
- Cooperative P-state control is the second. However, its efficiency ratio (comparing to None P-state Control) gets much higher in light load (10% load). It means there is some headroom in high to medium loads to be improved in Cooperative P-state Control.

From another aspect, if we observe the processor frequency impact at each load level, it is easy to illustrate the efficiency difference as well as shown in Figure 37. The CPU frequency dropped aggressively under schedutil governor. That's why schedutil is a good choice in power efficiency. In idle, it's easy to impact the CPU frequency because it's susceptible to interrupts and monitoring tools (e.g. performance & power monitors). Overall, a judgement on efficiency cannot be made based only on the idle CPU frequency measured.

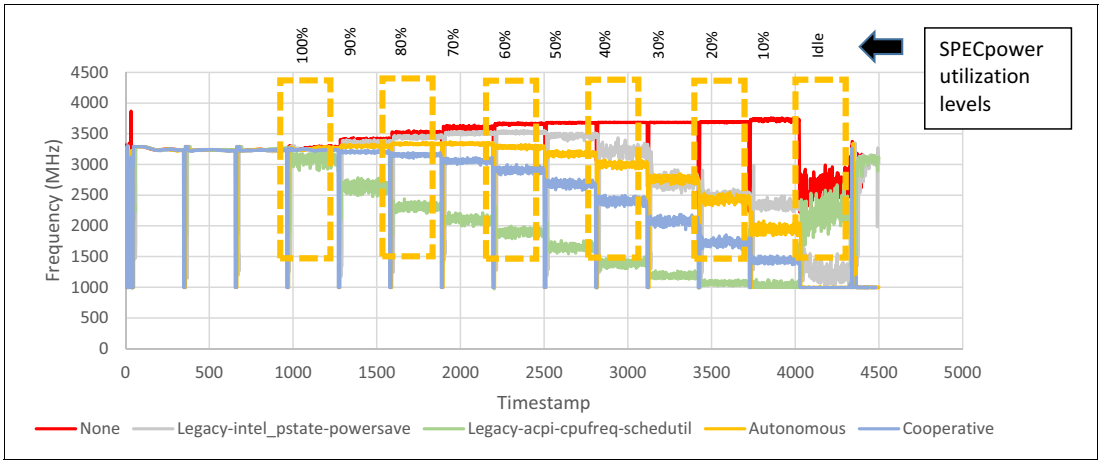

Figure 37 Comparing P-State Control modes against different SPECpower processor utilization levels

Table 3 shows a global view of UEFI P-state Control, the available CPU frequency drivers, and the available OS governors. An asterisk (*) indicates the default option under SUSE. To change the default, additional settings must be made as described in the previous paragraphs.

Table 3 Relationships among UEFI P-state Control, CPU frequency driver, and OS governors (= default)*

UEFI P-state control	Available cpufreq driver	Available OS governors	Key notes
None P-state control	intel_pstate	powersave*	The behaviors are the same due to no CPU frequency control.
		performance	
Legacy P-state control	intel_pstate*	powersave*	CPU frequency will be changed according to load. It is better for power efficiency.
		performance	CPU frequency will be maximized with no regard for power consumption. It is worse for power efficiency.
	acpi-cpufreq	ondemand*	CPU frequency will changed according to load. It has a medium effect on power efficiency.
		performance	CPU frequency will be maximized with no regard for power consumption. It is worse for power efficiency.
		schedutil	The best for power efficiency.
Autonomous P-state control*	NA	NA	System hardware will decide the CPU frequency.
Cooperative P-state control	intel_pstate	powersave*	CPU frequency will be changed according to load. It is better for power efficiency.
		performance	CPU frequency will be maximized with no regard for power consumption. It is worse for power efficiency.

Conclusion

Using the industry standard SPECpower_ssj2008 benchmark workload, this paper has shown that the Legacy P-state control method in the server's UEFI setup menu is the best selection when peak performance/watt efficiency is desired. This is true whether running Windows or Linux.

In addition to the optimal UEFI settings, the best OS settings were also determined:

- For Windows, the Balanced Power Plan provides peak efficiency.
- For Linux, acpi-cpufreq driver is much better than intel_pstate driver no matter which one is the OS's default recommendation. It needs to be changed manually if it is not default. However, the enablement depends on the Linux kernel. As was mentioned, it is different in RHEL 7.4 versus SLES 12 SP4. The best governor choice is schedutil, if it is available under the acpi-cpufreq driver. If the schedutil governor is not supported due to a kernel restriction, the ondemand governor is the second choice for best efficiency with no performance impact.

Based on our analysis, in general, power and processor frequency drop as the workload is reduced. This is the nature of SPECpower_ssj2008 workload. Additionally, the processor frequency tends to drop the most aggressively in Legacy mode.

However, there is one trade-off to be aware of with Legacy mode: The sluggish processor frequency change that occurs in Legacy mode increases latency when P-states are changing. If the workload is bursty, increased latency can become noticeable due to the transient nature of the workload where the load constantly transitions between high and low load levels.

To know which P-state control mechanism is best in real world environments, we need to know the purpose of the real usage. Listed below is some predefined scenarios to offer guidance:

- **Legacy**: Fits most real use cases and has the best energy efficiency without a performance impact. Most customers can choose it as the first step and do further tunings.
- **None**: Suitable for customers targeting the highest absolute performance regardless of energy consumption. For example, in a high frequency trading workload, the latency of frequency change can be very costly if it is too high. The lowest latency and maximum performance are the overriding goals.
- **Autonomous**: The processor frequency drops in correspondence with the load levels. Its operation lies between Legacy and None. In real cases, Autonomous is suitable for customers who want to reduce power in light loads but still target higher performance.
- **Cooperative**: This mode is a compromise between Legacy and Autonomous modes. It is targeted to the customers who want to reduce power in light loads but prefer power savings. However, both UEFI and the OS support have to be present for it to work.

Table 4 summarizes the recommendations.

Table 4 P-state control recommendations

P-state control	Description	Common applicable scenarios	Implementation highlights
Legacy (recommended)	Default setting. Targets good energy efficiency with minimal performance impact.	► General computing	It is implemented in UEFI and is the best in most cases for obtaining the best efficiency.
None	Increases performance at the cost of high energy consumption. Power and thermal limitations, operating expenses, and reliability considerations apply.	► Low latency applications ► Application code that is sensitive to processor performance changes	Processors are always kept at the P1 or P0 (including “turbo” frequencies).
Autonomous	Obtain both power save and processor frequency with a preference towards performance.	► Efficiency – Favor Performance	Supported by Lenovo ThinkSystem servers (Intel Scalable Family processors with Intel ME). No OS or external software is required.
Cooperative	Obtain both power save and processor frequency with a preference towards power savings.	► Efficiency – Favor Power	Supported by Lenovo ThinkSystem servers (Intel Scalable Family processors with Intel ME). Requires OS support to provide hints to the system hardware.

Change history

May 2019:

- New section, “Tuning SLES 12 SP4 for energy efficiency” on page 19

April 2018

- Initial version

Authors

This paper was produced by the following specialists:

Jenseng Chen is an Advisory Software Engineer in the Lenovo Server Performance lab. He joined Lenovo in October 2014, after having worked at IBM for 8 years. He started his career in IBM as an operating system engineer, having worked on Linux development and support. Since then, he has been working on Server Performance Validation in energy efficiency. He holds three patents in server performance and system energy efficiency areas. Jenseng holds a Master degree in Computer Information Science from National Chiao Tung University in Taiwan.

Robert R. Wolford is the Principal Engineer for power management and efficiency at Lenovo. He covers all technical aspects that are associated with power metering, management, and efficiency. Previously, he was a lead systems engineer for workstation products, video subsystems building block owner for System x®, System p, and desktops, signal quality and timing analysis engineer, gate array designer, and product engineering. In addition, he was a Technical Sales Specialist for System x. Bob has several issued patents centered around computing technology. He holds a Bachelor of Science degree in Electrical Engineering with Distinction from the Pennsylvania State University.

Thanks to the following people for their contributions to this project:

- Joseph Jakubowski, Lenovo Performance Senior Engineering Staff Member
- David Watts, Lenovo Press

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 May 16, 2019.

Send us your comments via the **Rate & Provide Feedback** form found at
http://lenovopress.com/lp0870

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, 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.

Windows, Windows Server, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

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