Using Processor Performance P-States with Linux on Intel-based ThinkSystem Servers

Planning / Implementation

Overview of Processor P-States

In modern processors, CPU frequency is not fixed. It can be adjusted within the range constrained by power and thermal budget. The range are often composed of some predetermined clock frequency and voltage configurations, which are defined as P-states in the ACPI specification.

As a rule, the higher the voltage and the higher the clock frequency are, and the more instructions can be executed by the CPU over a unit of time. But a higher voltage means the more power is consumed over a unit of time and a higher pressure for the cooling system. Therefore, there is a trade-off between the CPU performance and the power consumption and cooling condition. And this is the reason for performance management.

Normally, we do not always run CPU at the highest frequency, because it is not power efficient and may also not be permitted by the thermal budget. Instead, how fast the CPU should run is often affected by the computing usage scenario. In some scenarios like critical network services, CPU is expected to run as fast as possible. In some other scenarios like mobile applications, CPU is restricted to a lower speed to extend the battery life. This determines that there is not a universal CPU frequency adjustment scheme. Instead, performance management subsystem has to support various performance management policies.

In the following sections, the performance management techniques available with Intel processors are introduced. Then it is explained how Linux power management subsystem is designed to support those hardware techniques to meet multiple computing usage scenarios. Finally, the way to do performance management with Linux power management utilities are covered.

Intel Processor Performance Techniques

Intel has implemented a series of mechanisms in the x86 architectures to extract additional performance within the constraints, which include the following:

- **Enhanced Intel SpeedStep Technology (EIST)**

  Enables the management of processor power consumption via performance state (P-state) transitions. The states are defined as discrete operating points associated with different voltages and frequencies.

- **Turbo Boost Technology (TBT)**

  Uses the same principle of leveraging thermal headroom to increase processor performance dynamically and opportunistically for single-threaded and multi-threaded/multi-tasking environment.

- **Hardware Coordination Feedback**

  Provides the feedback mechanism using IA32_MPERF MSR and IA32_APERF MSR. OS can use the two MSRs to calculate CPU utilizations and then make P-state decisions.

- **Hardware-Controlled Performance States (HWP)**

  Autonomously selects performance states as deemed appropriate for the applied workload and with consideration of constraining hints that are programmed by the OS. The states are a continuous, abstract unit-less, performance value scale that is not tied to a specific performance state/frequency by definition.

Linux CPUFreq Subsystem

CPU performance management is to adjust CPU frequency according to CPU running status changes. This incurs two steps: performance level selection and performance level application.

CPU performance level is determined by the factors that are critical to the computing usage scenario, and there are various computing usage scenarios to support. Since performance management techniques are not unique even on the same platform, multiple performance level applications need support.

CPUFreq subsystem is the Linux's answer to the complexities of CPU performance level selections and applications. It is composed of the CPUFreq core, CPUFreq governors, and CPUFreq drivers. Governors handle various computing usage scenarios. Each governor implements a specific performance level selection mechanism. Drivers interact with the underlying performance management hardware to apply the selected performance level.

CPUFreq core manages governors and drivers' registration and switching and provides sysfs interfaces for user management works. Multiple governors can be registered with the core, and one of them is designated as the default governor that is used when no governor is chosen explicitly. For drivers, only one is allowed. Once a driver registered with the core, the core does not accept new drivers.

Governor

A governor monitors system running status and chooses a new performance level when the status changes. Then it dictates the CPUFreq driver to apply the new performance level via the core. Below are some of the governors.

- **performance** requests the highest frequency to be used.
- **powersave** requests the lowest frequency to be used.
- **schedutil** uses CPU utilization data from the scheduler and calculates the new frequency by the utilization and the scheduling class.
- **userspace** passes through the frequency from user space and does nothing by itself.

Normally all CPUs are managed by the default governor and their frequencies are within the range of the hardware minimum and maximum frequencies. CPUFreq allows user. to set the frequency range and the governor for each CPU. Different CPUs may have different frequency ranges and governors. CPUFreq saves the settings in the structure called CPUFreq policy. Every CPU under the management of CPUFreq has a such policy bound with it. A policy may be shared by multiple CPUs in some cases where the hardware interface for P-state control is shared by CPUs.

A policy records the governor bound, CPU hardware frequency range, current frequency, P-state transition latency, and so on. Besides the user settings. The policies are exposed to user via sysfs.

Driver

CPUFreq defines a unified interface for all CPUFreq drivers. Among the interface functions, the policy initialization and frequency switching functions are the most important. The policy initialization interface function gives the driver the opportunity to expose hardware information to upper layers. The frequency switching interface function accepts a frequency and applies to the hardware.

There are two CPUFreq drivers, acpi-cpufreq and intel_pstate, available for Intel platforms. By default, intel_pstate is used as the driver, and acpi-cpufreq is used only when the platform is not supported by intel_pstate. The kernel parameter intel_pstate=disable can be used to disable intel_pstate.

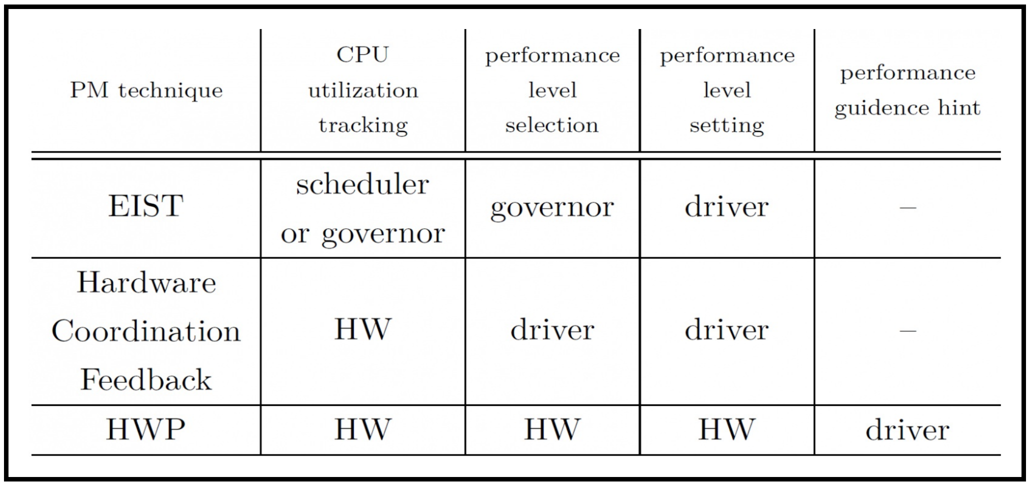
acpi-cpufreq Driver

acpi-cpufreq supports ACPI based CPU performance management. The available P-states are reported by the ACPI Performance Supported States object _PSS. The ACPI Performance Control object _PCT declares an interface that allows the OS to transition the processor into a performance state. Depending on _PCT type, acpi-cpufreq communicates with hardware either by system IO ports or by EIST MSR IA32_PERF_CTL and IA32_PERF_STATUS.

intel_pstate Driver

intel_pstate supports only CPUs with either **Hardware-Controlled Performance States (HWP)** feature or APERF/MPERF feature. The former feature indicates that CPU supports HWP. The latter feature indicates that CPU supports hardware coordination feedback.

Depending on the kernel and hardware configurations, intel_pstate plays different roles in CPU performance management. When HWP is not available, intel_pstate may work as acpi-cpufreq doing only performance level setting work; or it can use APERF and MPERF MSRs to calculate CPU utilization and do performance level selection by itself. When HWP is available, CPU does performance adjustment autonomously, and so it only needs to provide performance guidance hints to CPU.



PM technique	CPU utilization tracking	performance level selection	performance level setting	performance guidence hint
EIST	scheduler or governor	governor	driver	–
Hardware Coordination Feedback	HW	driver	driver	–
HWP	HW	HW	HW	driver

Figure 1. Performance mechanisms and their policies

Therefore, the kernel module intel_pstate includes two CPUFreq drivers, intel_pstate and intel_cpufreq, to handle the different models above. intel_cpufreq is a normal CPUFreq driver that accepts frequency targets from governors. It uses Enhanced Intel SpeedStep Technology (EIST) MSRs to adjust CPU performance level.

On the contrary, intel_pstate is both a driver and a governor. When HWP is available, it depends on CPU to autonomously adjust performance level, and only gives performance hints when necessary. If HWP is not available, it leverages hardware coordinate feedback to decide performance levels by itself instead of relying on external governors. intel_pstate provides two performance governing algorithms, performance and powersave, which are equivalent to schedutil and powersave governors respectively.

Which of the two drivers is used depends on the hardware and kernel configurations. By default, if CPU has HWP feature, intel_pstate module enables it and registers intel_pstate as the CPUFreq driver; otherwise, intel_cpufreq is registered as the CPUFreq driver.

Users can change the default behavior by kernel parameter intel_pstate. intel_pstate=passive and intel_pstate=active choose intel_cpufreq and intel_pstate as the driver respectively.

Once enabled by either BIOS or OS, HWP cannot be disabled again unless system reboots. intel_pstate=no_hwp can be used to inhibit HWP being enabled by the module.

When HWP is not enabled, intel_cpufreq works like acpi-cpufreq. intel_pstate leverages MPERF and APERF MSRs to calculate CPU utilization, selects CPU performance, and uses EIST MSRs to apply the target performance level.

Both drivers can work with HWP enabled, but their working models are different from those without HWP enabled. When HWP is enabled, both drivers can not directly control CPU performance level. They convey their performance levels to CPU via HWP hint mechanism. Note that an excursion of the performance level actually chosen by CPU below the level specified is possible due to hardware constraints.

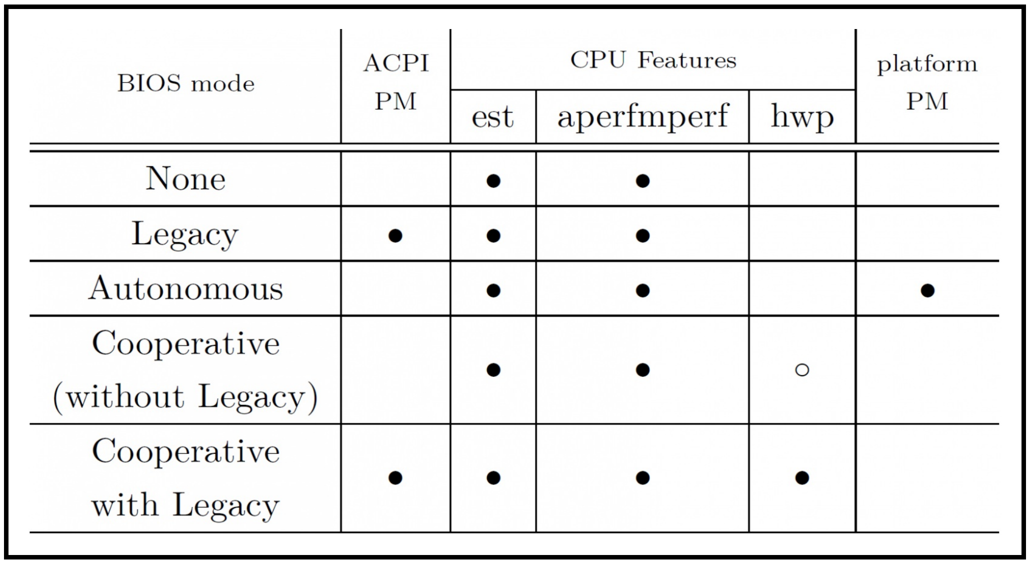
intel_cpufreq accepts CPU frequency targets from governors as its hints. The situation is more complicated for intel_pstate, since it also has to handle different governing modes.

HWP allows the OS to set its preference for performance or power saving via the MSR field of Energy Performance Preference (EPP). With performance governing mode, intel_pstate sets EPP to performance mode, and relies on HWP autonomous performance adjustment most time and only hints for more performance to keep a high IO throughput when IOs are waited for completion. With powersave governing mode, intel_pstate sets EPP to balance performance mode, and leverages MPERF and APERF MSRs to selects CPU performance levels.

Intel P-State Configuration with Lenovo UEFI

On Lenovo ThinkSystem platforms, UEFI provides multiple operating modes under **System Settings > Operating Modes > CPU P-State Control** for users to control the way CPU P-States is supported.

Below is a list of the operating modes and their impact on FW or CPU states observed on ThinkSystem servers with Intel processors.



BIOS mode	ACPI PM	CPU Features			platform PM
		est	aperfmperf	hwp	
None		●	●		
Legacy	●	●	●		
Autonomous		●	●		●
Cooperative (without Legacy)		●	●	○	
Cooperative with Legacy	●	●	●	●	

Figure 2. BIOS settings and the corresponding policies

Below is a summary of the CPU P-State Control setting and the CPUFreq driver used.

- **None** BIOS disables ACPI PM, but since CPU APERF/MPERF feature is available, intel_cpufreq driver is used.
- **Legacy** intel_cpufreq is used, if intel_pstate module is not disabled; otherwise, acpi-cpufreq is used.
- **Autonomous** BIOS takes over performance management. As of Linux v5.1 and new kernels, intel_pstate module will not load at this circumstance. And so does acpi-cpufreq as lacking of ACPI PM.
- **Cooperative (without Legacy)** ACPI PM is not supported in this mode. CPU HWP feature is available, but BIOS does not enable HWP at bootstrap. intel_pstate driver is used, and HWP will be enabled unless it is prohibited by the kernel parameter intel_pstate=no_hwp.
- **Cooperative with Legacy** ACPI PM is supported and CPU HWP feature is available and enabled by BIOS at bootstrap. intel_pstate driver is used by default.

Managing Linux Processor P-States

Processor P-states can be managed with sysfs interface of CPUFreq and with the cpupower utility.

Common Sysfs interface

If CPUFreq subsystem is available, then it presents its sysfs interfaces under directory /sys/devices/system/cpu/cpufreq/.

Directory cpufreq contains a sub-directory policy*n* for each CPU*n*'s policy structure that records CPU performance management related information.

Under policyn are attributes files:

- **cpuinfo_max_freq** Maximum possible operating frequency CPU*n* can run at (in kHz).
- **cpuinfo_min_freq** Minimum possible operating frequency CPU *n* can run at (in kHz).
- **scaling_max_freq** Maximum frequency CPU*n* are allowed to be running at (in kHz). This attribute is read-write. Writing an integer (must not be lower than scaling_min_freq) to it will cause a new limit to be set.
- **scaling_min_freq** Minimum frequency CPU*n* are allowed to be running at (in kHz). This attribute is read-write. Writing an integer (must not be higher than scaling_max_freq) to it will cause a new limit to be set.
- **scaling_driver** The CPUFreq driver currently in use.
- **scaling_available_governors** List of governors that can be attached to CPU *n*.
- **scaling_governor** The CPUFreq governor currently attached to CPU *n*. This attribute is read-write. Writing to it will cause a new CPUFreq governor to be attached to CPU*n*.

Intel_pstate Sysfs interface

intel_pstate module also exposes sysfs interface under directory /sys/devices/system/cpu/intel_pstate/.

The attribute files under the directory affects all CPUs.

- **max_perf_pct** Maximum P-state the driver is allowed to set in percent of the maximum supported performance level.
- **min_perf_pct** Minimum P-state the driver is allowed to set in percent of the maximum supported performance level.
- **no_turbo** If set to 1, the driver is not allowed to set any turbo P-states. If set to 0, which is the default, turbo P-states can be set by the driver.
- **status** Operation mode of the driver: “active” means that intel_pstate is used, “passive” means that intel_cpufreq is used, and “off” means that none of the drivers is used. This attribute can be written to change the driver’s operation mode or to unregister it.

cpupower

Cpupower is the Linux utility to do various power management works. It has two subcommands for P-state management:

- `cpupower frequency-info`
- `cpupower frequency-set`

The `cpupower frequency-info` subcommand prints out CPUFreq information.

```
# 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 Avail
able
  maximum transition latency:  Cannot determine or is not supported.
  hardware limits: 1.20 GHz - 2.70 GHz
  available cpufreq governors: performance
  current policy: frequency should be within 1.20 GHz and 2.70 GHz.
                  The governor "performance" may decide which speed to use
                  within this range.
  current CPU frequency: 1.68 GHz (asserted by call to hardware)
  boost state support:
    Supported: yes
    Active: yes
    2500 MHz max turbo 4 active cores
    2500 MHz max turbo 3 active cores
    2600 MHz max turbo 2 active cores
    2700 MHz max turbo 1 active cores
```

The `cpupower frequency-set` subcommand allows user to modify CPUFreq settings without using sysfs interfaces. Below options are provided.

- **-g <governor>** sets a new governor and is equivalent to write sysfs attribute file scaling_governor.
- **-d <frequency>** sets a new minimum CPU frequency and is equivalent to write sysfs attribute file scaling_min_freq.
- **-u <frequency>** sets a new maximum CPU frequency and is equivalent to write sysfs attribute file scaling_max_freq.

By default, the settings are applied on all CPUs. Option “-c cpu” can be used to select CPU(s) to do frequency-set.

Authors

Peng Liu is an experienced Linux Engineer at the Lenovo Infrastructure Solutions Group, based in Beijing, China. He focuses on storage device drivers and but his interest areas include topics such as I/O frameworks and memory management.

Xiaochun Li is a Linux engineer at the Lenovo Infrastructure Solution Group in Beijing, China. He specializes in the development of Linux kernel storage and memory management, as well as virtualization. Before joining Lenovo, he worked in INSPUR as an OS engineer for several years. With ten years of industry experience, he now focuses on Linux kernel RAS, storage, security, and virtualization.

Thanks to Adrian Huang from Lenovo ISG in Taipei for his contributions to the development of this paper.

Related product families

Product families related to this document are the following:

- Processors

Notices

This document, LP1946, was created or updated on April 26, 2024.

Send us your comments in one of the following ways:

- Use the online Contact us review form found at:
  https://lenovopress.lenovo.com/LP1946
- Send your comments in an e-mail to:
  comments@lenovopress.com

This document is available online at https://lenovopress.lenovo.com/LP1946.