CPU frequency scaling

CPU frequency scaling enables the operating system to scale the CPU frequency up or down in order to save power. CPU frequencies can be scaled automatically depending on the system load, in response to ACPI events, or manually by userspace programs.

CPU frequency scaling is implemented in the Linux kernel, the infrastructure is called *cpufreq*. Since kernel 3.4 the necessary modules are loaded automatically. For older kernels or CPUs, the recommended **ondemand governor** is enabled by default, whereas for newer kernels or CPUs, the **schedutil governor** is enabled by default. However, userspace tools like **cpupower**, **acpid**, **Laptop Mode Tools**, or GUI tools provided for your desktop environment, may still be used for advanced configuration.

Related articles

- **Power saving**
- **Laptop Mode Tools**
- **Undervolting CPU**
- **PHC**

Contents

Userspace tools

thermald

thermald (https://archlinux.org/packages/?name=thermald) is a Linux daemon used to prevent the overheating of Intel CPUs. This daemon monitors temperature and applies compensation using available cooling methods.

By default, it monitors CPU temperature using available CPU digital temperature sensors and maintains CPU temperature under control, before HW takes aggressive correction action. If there is a skin temperature sensor in thermal sysfs, then it tries to keep skin temperature under 45C.

The associated systemd unit is `thermald.service`, which should be **started** and **enabled**.

i7z

i7z (https://archlinux.org/packages/?name=i7z) is an i7 (and now i3, i5, i7, i9) CPU reporting tool for Linux. It can be launched from a Terminal with the command `i7z` or as GUI with `i7z-gui`.

cpupower

cpupower (https://archlinux.org/packages/?name=cpupower) is a set of userspace utilities designed to assist with CPU frequency scaling. The package is not required to use scaling, but is highly recommended because it provides useful command-line utilities and a **systemd** service to change the governor at boot.

The configuration file for *cpupower* is located in `/etc/default/cpupower`. This configuration file is read by a bash script in `/usr/lib/systemd/scripts/cpupower` which is activated by *systemd* with `cpupower.service`. You may want to **enable** `cpupower.service` to start at boot.

cpupower-gui

cpupower-gui (https://aur.archlinux.org/packages/cpupower-gui/)[AUR] is a graphical utility designed to assist with CPU frequency scaling. The GUI is based on **GTK** and is meant to provide the same options as *cpupower*. *cpupower-gui* can change the maximum/minimum CPU frequency and governor for each core. The application handles privilege granting through **polkit** and allows any logged-in user in the `wheel` **user group** to change the frequency and governor.

turbostat

turbostat (https://archlinux.org/packages/?name=turbostat) can display the frequency, power consumption, idle status and other statistics of the modern Intel and AMD CPUs.

CPU frequency driver

Note:

- The native CPU module is loaded automatically.
- The `intel_pstate` CPU power scaling driver is used automatically for modern Intel CPUs instead of the other drivers below. This driver takes priority over other drivers and is built-in as opposed to being a module. This driver is currently automatically used for Sandy Bridge and newer CPUs. The `intel_pstate` may ignore the BIOS P-State settings. `intel_pstate` may run in "passive mode" via the `intel_cpufreq` driver for older CPUs. If you encounter a problem while using this driver, add `intel_pstate=disable` to your kernel line in order to revert to using the `acpi-cpufreq` driver.
- Even P State behavior mentioned above can be influenced with `/sys/devices/system/cpu/intel_pstate`, e.g. Intel Turbo Boost can be deactivated with `echo 1 > /sys/devices/system/cpu/intel_pstate/no_turbo` as the root user for keeping CPU-Temperatures low.
- Additional control for modern Intel CPUs is available with the **Linux Thermal Daemon (https://01.org/linux-thermal-daemon)** (available as **thermald (https://archlinux.org/packages/?name=thermald)**), which proactively controls thermal using P-states, T-states, and the Intel power clamp driver. thermald can also be used for older Intel CPUs. If the latest drivers are not available, then the daemon will revert to x86 model specific registers and the Linux 'cpufreq subsystem' to control system cooling.

cpupower requires modules to know the limits of the native CPU:

Module	Description
intel_pstate	This driver implements a scaling driver with an internal governor for Intel Core (Sandy Bridge and newer) processors.
intel_cpufreq	Starting with kernel 5.7, the intel_pstate scaling driver selects "passive mode" aka intel_cpufreq for CPUs that do not support hardware-managed P-states (HWP), i.e. Intel Core i 5th generation or older.
acpi-cpufreq	CPUFreq driver which utilizes the ACPI Processor Performance States. This driver also supports the Intel Enhanced SpeedStep (previously supported by the deprecated speedstep-centrino module).
speedstep-lib	CPUFreq driver for Intel SpeedStep-enabled processors (mostly Atoms and older Pentiums)
powernow-k8	CPUFreq driver for K8/K10 Athlon 64/Opteron/Phenom processors. Since Linux 3.7 'acpi-cpufreq' will automatically be used for more modern AMD CPUs.
pcc-cpufreq	This driver supports Processor Clocking Control interface by Hewlett-Packard and Microsoft Corporation which is useful on some ProLiant servers.
p4-clockmod	CPUFreq driver for Intel Pentium 4/Xeon/Celeron processors which lowers the CPU temperature by skipping clocks. (You probably want to use a SpeedStep driver instead.)

To see a full list of available modules, run:

```
$ ls /usr/lib/modules/$(uname -r)/kernel/drivers/cpufreq/
```

Load the appropriate module (see **Kernel modules** for details). Once the appropriate cpufreq driver is loaded, detailed information about the CPU(s) can be displayed by running

```
$ cpupower frequency-info
```

Setting maximum and minimum frequencies

In some cases, it may be necessary to manually set maximum and minimum frequencies.

To set the maximum clock frequency (*clock_freq* is a clock frequency with units: GHz, MHz):

```
# cpupower frequency-set -u clock_freq
```

To set the minimum clock frequency:

```
# cpupower frequency-set -d clock_freq
```

To set the CPU to run at a specified frequency:

```
# cpupower frequency-set -f clock_freq
```

Note:

- To adjust for only a single CPU core, append `-c core_number` .
- The governor, maximum and minimum frequencies can be set in `/etc/default/cpupower` .

Alternatively, you can set the frequency manually:

```
# echo value > /sys/devices/system/cpu/cpu*/cpufreq/scaling_max_freq
```

The available values can be found in `/sys/devices/system/cpu/cpu*/cpufreq/scaling_available_frequencies` or similar. **[1] (https://software.intel.com/sites/default/files/comment/1716807/how-to-change-frequency-on-linux-pub.txt)**

Disabling Turbo Boost

intel_pstate

```
# echo 1 > /sys/devices/system/cpu/intel_pstate/no_turbo
```

acpi-cpufreq

```
# echo 0 > /sys/devices/system/cpu/cpufreq/boost
```

x86_energy_perf_policy

With **x86_energy_perf_policy (https://archlinux.org/packages/?name=x86_energy_perf_policy)**:

```
# x86_energy_perf_policy --turbo-enable 0
```

The change is temporary.

Scaling governors

Governors (see table below) are power schemes for the CPU. Only one may be active at a time. For details, see the **kernel documentation (https://www.kernel.org/doc/Documentation/cpu-freq/governors.txt)** in the kernel source.

Governor	Description
performance	Run the CPU at the maximum frequency.
powersave	Run the CPU at the minimum frequency.
userspace	Run the CPU at user specified frequencies.
ondemand	Scales the frequency dynamically according to current load. Jumps to the highest frequency and then possibly back off as the idle time increases.
conservative	Scales the frequency dynamically according to current load. Scales the frequency more gradually than ondemand.
schedutil	Scheduler-driven CPU frequency selection **[2] (https://lwn.net/Articles/682391/)**, **[3] (https://lkml.org/lkml/2016/3/17/420)**.

Depending on the scaling driver, one of these governors will be loaded by default:

- `powersave` for Intel CPUs using the `intel_pstate` driver (Sandy Bridge and newer)..
- `powersave` (for Linux < 5.10) or `schedutil` (since Linux 5.10) for CPUs using the `acpi-cpufreq` driver.

Note: The `intel_pstate` driver supports only two governors: `powersave` and `performance`. Although they share the name with the generic governors, they do not work in the same way as the generic governors. Both `intel_pstate` governors provide dynamic scaling similar to the `schedutil` or `ondemand` generic governors. The `performance` governor provided by `intel_pstate` **should give better power saving functionality than the old ondemand governor (https://www.phoronix.com/scan.php?page=news_item&px=MTM3NDQ)**.

Warning: Use CPU monitoring tools (for temperatures, voltage, etc.) when changing the default governor.

To activate a particular governor, run:

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

Note:

- To adjust for only a single CPU core, append `-c core_number` to the command above.
- Activating a governor requires that specific **kernel module** (named `cpufreq_governor`) is loaded. As of kernel 3.4, these modules are loaded automatically.

Alternatively, you can activate a governor on every available CPU manually:

```
# echo governor | tee /sys/devices/system/cpu/cpu*/cpufreq/scaling_governor
```

where `governor` is the name of the governor, mentioned in the above table, that you want to activate.

Tip: To monitor cpu speed in real time, run:

```
$ watch cat /sys/devices/system/cpu/cpu[0-9]*/cpufreq/scaling_cur_freq
```

Tuning the ondemand governor

See the **kernel documentation (https://www.kernel.org/doc/Documentation/cpu-freq/governors.txt)** for details.

Switching threshold

To set the threshold for stepping up to another frequency:

```
# echo -n percent > /sys/devices/system/cpu/cpufreq/<governor>/up_threshold
```

To set the threshold for stepping down to another frequency:

```
# echo -n percent > /sys/devices/system/cpu/cpufreq/<governor>/down_threshold
```

Sampling rate

The sampling rate determines how frequently the governor checks to tune the CPU. `sampling_down_factor` is a tunable that multiplies the sampling rate when the CPU is at its highest clock frequency thereby delaying load evaluation and improving performance. Allowed values for `sampling_down_factor` are 1 to 100000. This tunable has no effect on behavior at lower CPU frequencies/loads.

To read the value (default = 1), run:

```
$ cat /sys/devices/system/cpu/cpufreq/ondemand/sampling_down_factor
```

To set the value, run:

```
# echo -n value > /sys/devices/system/cpu/cpufreq/ondemand/sampling_down_factor
```

Make changes permanent

To have the desired scaling enabled at boot, **kernel module options** and **systemd-tmpfiles** are regular methods.

For example, changing the up_threshold to 10:

```
/etc/tmpfiles.d/ondemand.conf
```

```
w- /sys/devices/system/cpu/cpufreq/ondemand/up_threshold - - - - 10
```

However, as noted in **systemd-tmpfiles**, in some cases race conditions may exist and one can use **udev** to avoid them.

For example, to set the scaling governor of the CPU core `0` to performance while the scaling driver is `acpi_cpufreq`, create the following udev rule:

```
/etc/udev/rules.d/50-scaling-governor.rules
```

```
SUBSYSTEM=="module", ACTION=="add", KERNEL=="acpi_cpufreq", RUN+="/bin/sh -c 'echo performance > /sys/devices/system/cpu/cpufreq/policy0/scaling_governor'"
```

To have the rule already applied in the *initramfs*, follow the example at **udev#Debug output**.

Tip:

- Since Linux 5.9, it is possible to set the `cpufreq.default_governor` kernel option.**[4] (https://kernelnewbies.org/Linux_5.9#CPU_Frequency_scaling)**
- Alternatively, configure the **cpupower** utility and enable its systemd service.

Control Intel CPUs energy performance policy

Install **x86_energy_perf_policy (https://archlinux.org/packages/?name=x86_energy_perf_policy)**:

Enable Hardware P-States:

```
# x86_energy_perf_policy -H 1
# x86_energy_perf_policy -U 1
```

Set "default" policy:

```
# x86_energy_perf_policy default
```

Set "performance" policy:

```
# x86_energy_perf_policy performance
```

Set "balance-performance" policy:

```
# x86_energy_perf_policy balance-performance
```

Set "balance-power" policy:

```
# x86_energy_perf_policy balance-power
```

Set "power" policy:

```
# x86_energy_perf_policy power
```

The changes are temporary. See **x86_energy_perf_policy(8) (https://man.archlinux.org/man/x86_energy_perf_policy.8)** for more info.

CPU idle driver

The `intel_idle` CPU idle driver is used automatically for modern Intel CPUs instead of the `acpi_idle` driver. This driver is currently automatically used for Sandy Bridge and newer CPUs. The `intel_idle` may ignore the BIOS C-State settings. If you encounter a problem while using this driver, add `intel_idle.max_cstate=0` to your kernel line.

Interaction with ACPI events

Users may configure scaling governors to switch automatically based on different ACPI events such as connecting the AC adapter or closing a laptop lid. A quick example is given below, however it may be worth reading full article on **acpid**.

Events are defined in `/etc/acpi/handler.sh` . If the **acpid (https://archlinux.org/packages/?name=acpid)** package is installed, the file should already exist and be executable. For example, to change the scaling governor from `performance` to `conservative` when the AC adapter is disconnected and change it back if reconnected:

```
/etc/acpi/handler.sh
```

```
[...]

ac_adapter)
    case "$2" in
        AC*)
            case "$4" in
                00000000)
                    echo "conservative" >/sys/devices/system/cpu/cpu0/cpufreq/scaling_governor
                    echo -n $minspeed >$setspeed
                    #/etc/laptop-mode/laptop-mode start
                ;;
                00000001)
                    echo "performance" >/sys/devices/system/cpu/cpu0/cpufreq/scaling_governor
                    echo -n $maxspeed >$setspeed
                    #/etc/laptop-mode/laptop-mode stop
                ;;
            esac
        ;;
        *) logger "ACPI action undefined: $2" ;;
    esac
;;

[...]
```

Privilege granting under GNOME

Note: systemd introduced logind which handles consolekit and policykit actions. The following code below does not work. With logind, simply edit in the file `/usr/share/polkit-1/actions/org.gnome.cpufreqselector.policy` the <defaults> elements according to your needs and the polkit manual **[5] (https://www.freedesktop.org/software/polkit/docs/latest/polkit.8.html)**.

GNOME has a nice applet to change the governor on the fly. To use it without the need to enter the root password, simply create following file:

```
/var/lib/polkit-1/localauthority/50-local.d/org.gnome.cpufreqselector.pkla
```

```
[org.gnome.cpufreqselector]
Identity=unix-user:user
Action=org.gnome.cpufreqselector
ResultAny=no
ResultInactive=no
ResultActive=yes
```

Where the word *user* is replaced with the username of interest.

The desktop-privileges (https://aur.archlinux.org/packages/desktop-privileges/)[AUR] package in the AUR contains a similar `.pkla` file for authorizing all users of the `power` user group to change the governor.

Troubleshooting

- Some applications, like ntop, do not respond well to automatic frequency scaling. In the case of ntop it can result in segmentation faults and lots of lost information as even the `on-demand` governor cannot change the frequency quickly enough when a lot of packets suddenly arrive at the monitored network interface that cannot be handled by the current processor speed.
- Some CPU's may suffer from poor performance with the default settings of the `on-demand` governor (e.g. flash videos not playing smoothly or stuttering window animations). Instead of completely disabling frequency scaling to resolve these issues, the aggressiveness of frequency scaling can be increased by lowering the *up_threshold* sysctl variable for each CPU. See how to change the on-demand governor's threshold.
- Sometimes the on-demand governor may not throttle to the maximum frequency but one step below. This can be solved by setting max_freq value slightly higher than the real maximum. For example, if frequency range of the CPU is from 2.00 GHz to 3.00 GHz, setting max_freq to 3.01 GHz can be a good idea.
- Some combinations of ALSA drivers and sound chips may cause audio skipping as the governor changes between frequencies, switching back to a non-changing governor seems to stop the audio skipping.

BIOS frequency limitation

Some CPU/BIOS configurations may have difficulties to scale to the maximum frequency or scale to higher frequencies at all. This is most likely caused by BIOS events telling the OS to limit the frequency resulting in `/sys/devices/system/cpu/cpu0/cpufreq/bios_limit` set to a lower value.

Either you just made a specific Setting in the BIOS Setup Utility, (Frequency, Thermal Management, etc.) you can blame a buggy/outdated BIOS or the BIOS might have a serious reason for throttling the CPU on its own.

Reasons like that can be (assuming your machine's a notebook) that the battery is removed (or near death) so you are on AC-power only. In this case a weak AC-source might not supply enough electricity to fulfill extreme peak demands by the overall system and as there is no battery to assist this could lead to data loss, data corruption or in worst case even hardware damage!

Not all BIOS'es limit the CPU-Frequency in this case, but for example most IBM/Lenovo Thinkpads do. Refer to thinkwiki for more thinkpad related info on this topic (https://www.thinkwiki.org/wiki/Problem_with_CPU_frequency_scaling).

If you checked there is not just an odd BIOS setting and you know what you are doing you can make the Kernel ignore these BIOS-limitations.

Warning: Make sure you read and understood the section above. CPU frequency limitation is a safety feature of your BIOS and you should not need to work around it.

A special parameter has to be passed to the processor module.

For trying this temporarily change the value in `/sys/module/processor/parameters/ignore_ppc` from `0` to `1`.

For setting it permanently Kernel modules#Setting module options describes alternatives. For example, you can add `processor.ignore_ppc=1` to your kernel boot line, or create

```
/etc/modprobe.d/ignore_ppc.conf

# If the frequency of your machine gets wrongly limited by BIOS, this should help
options processor ignore_ppc=1
```

See also

- **Linux CPUFreq - kernel documentation (https://www.kernel.org/doc/html/latest/cpu-freq/index.html)**
- **Comprehensive explanation of pstate (https://www.reddit.com/r/linux/comments/1hdogn/acpi_cpufreq_or_intel_pstates/)**
- **Processor boosting control (https://www.kernel.org/doc/Documentation/cpu-freq/boost.txt)**
- **intel_pstate kernel documentation (https://www.kernel.org/doc/html/latest/admin-guide/pm/intel_pstate.html)**
- **intel_pstate/intel_cpufreq documentation kernel 5.7+ (https://linrunner.de/tlp/settings/processor.html)**