CPU frequency scaling in Linux with cpufreq

Here are some notes on getting CPU frequency scaling working on Linux. CPU frequency scaling does what it sounds like. It will raise and lower the frequency of your processor depending on a set level of demand being made on the processor at the time. One of the reasons you might want to do this would be to save energy. This could save you money on your electric bills or battery life on a laptop. It will also lower the temperature of your processor(s) to keep your machine cooler.

The following examples are done using K/X/Ubuntu distribution using a 2.6.24 kernel. Most of the settings in the examples should be the same or close to the same for any other distros. I will use apt-get the package manager installed with K/X/Ubuntu for the software install examples. You will need to use whatever package manager yours system uses. Yum (RPM) and apt-get (dpkg) are 2 very popular ones.

Before we start if you have a RedHat or Fedora machine you might want to try to install cpuspeed (yum install cpuspeed) to get frequency scaling. In K/X/Ubuntu or another Debian based machine you might just try to install cpufrequtils (sudo apt-get install cpufrequtils) to get frequency scaling working. If that does for you then you don't need to go through the parts of finding and loading the correct modules below. Just skip to the section on configuring the scaling governor.

Kernel version

Make sure your kernel version is at least 2.6.12 to make use of all the possible governors that will be mentioned. All of the frequency scaling will be done with kernel modules and not user space governors.

Enable support in your BIOS

Enter your BIOS and make sure Cool'n'Quiet (AMD) or SpeedStep (Intel) is enabled for you CPU. Some BIOS's may not have an option for either. If you don't find the option it is probably enabled by default. Unfortunately your BIOS may have the option but it is listed as another name altogether. If that is the case check your BIOS's manual for more information.

Remove any userspace CPU scaling programs

There are some userspace programs that can be run to scale the processors frequency. We will be scaling the CPU with the kernel so we don't need these. So if you have any we are going remove them now. You may want to look into these as they can be helpful in certain situations. This article will deal with just using the kernel modules to scale.

sudo apt-get remove powernowd cpudyn cpufreqd powersaved speedfreqd

Install the module for your CPU

When you installed your system there is a very good chance your CPU was detected by default and the module you need for for scaling is already running. Below is a command that will help you identify what type of processor(s) you have.

cat /proc/cpuinfo

After you know this then you will know what kernel module you will need to load for it. Here is the command to see what kernel modules are loaded.

lsmod

Below are CPU descriptions and the commands used to load the kernel modules based on what processor you have. Look at the output from lsmod above and use the modules names after the word "modprobe" below to see if you already have a module loaded. If you do then just move on to the next step. If not then use the CPU info you found and figure out which module you need to load. Then run the command to load it.

CPU: PIII-M or P4 without est. 2 module types for this.

sudo modprobe speedstep-ich
or
sudo modprobe speedstep-smi

CPU: Intel Core Duo, Intel Core2 Duo or Quad, or Intel Pentium M. This has been merged into the acpi-cpufreq module in later kernels.

sudo modprobe speedstep-centrino

CPU: Intel Celeron, Xeon, and Pentium 4 processors

sudo modprobe p4_clockmod

CPU: AMD K6. Socket Type: Socket 7

sudo modprobe powernow-k6

CPU: AMD Sempron/Athlon/MP (K7). Socket Types: A, Slot A.

sudo modprobe powernow-k7

CPU: AMD Duron/Sempron/Athlon/Opteron 64 (K8). Socket Types: 754, 939, 940, S1 (638), AM2 (940), F (1207).

sudo modprobe powernow-k8

CPU: VIA CentaurHauls* or Transmeta GenuineTMx86*

sudo modprobe longhaul

As a last resort if any of these don't work you can try the generic one for ACPI. More drivers are getting moved to this module in later kernels like speedstep-centrino after 2.6.20.

sudo modprobe acpi-cpufreq

Inserting the scaling modules

Now that the CPU frequency module is loaded we can now insert the scaling modules. To see which scaling modules you have a available you can use this command (using a Bash shell).

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

If you have these modules then they may already be running. To check if they are try the following command.

lsmod | grep freq

If you see most or all of modules that where listed in the cpufreq directory then your done. Move on to the next section. If not here are the commands to load the modules.

sudo modprobe cpufreq_conservative cpufreq_ondemand cpufreq_powersave cpufreq_stats cpufreq_userspace freq_table

Now that they are loaded you will want to load them on boot. To do this on a Debian based system like K/X/Ubuntu put the following lines in the /etc/modules file. You will have to check on where to put them on other distros like RedHat (/etc/modules.conf?). Remember to put the name of your CPU's module (you found above) in here also so it loads on boot. That is only if it is not loaded on boot. If you did not find it with lsmod when you first looked then it did not load automatically.

cpufreq_conservative
cpufreq_ondemand
cpufreq_powersave
cpufreq_stats
cpufreq_userspace
freq_table

Configuring the scaling modules

Now that they are loaded we can configure the governor. First you have to choose the governor you want to use. Below is a list the governors and how each works. You can decide which one you fits your needs best. If the module for a governor is loaded then you can use it. Remember you can see what modules are loaded with the lsmod command. Commands are done with sudo below like in K/X/Ubuntu world. You can switch to root and run the same commands just without the sudo sh -c " " if you like.

To show the available governors you can use.

sudo cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_available_governors

Ondemand governor - sets the CPU frequency depending on the current usage. To do this the CPU must have the capability to switch the frequency very quickly. This would be good for systems that do a lot of work (high load) for a short periods of time and then don't do much (low load) the rest of the time.

sudo sh -c "echo ondemand > /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor"

Ondemand governor configuration options

sampling_rate - This is measured in microseconds (one millionth of a second). This is how often you want the kernel to look at the CPU usage and to make decisions on what to do about the frequency. Typically this is set to values of around '10000' or more. If you wanted to set the sampling rate to 1 second you would set it to 1000000 like in the following example.

sudo sh -c "echo 1000000 > /sys/devices/system/cpu/cpu0/cpufreq/ondemand/sampling_rate"

show_sampling_rate_(min|max) - This is minimum and maximum sampling rates available that you may set 'sampling_rate' to. I believe in microseconds also. I've seen discussion on getting rid of this in later kernels don't count on it being there in the future. To see both just do the following.

sudo cat /sys/devices/system/cpu/cpu0/cpufreq/ondemand/sampling_rate_min
sudo cat /sys/devices/system/cpu/cpu0/cpufreq/ondemand/sampling_rate_max

up_threshold - This defines what the average CPU usage between the samplings of 'sampling_rate' needs to be for the kernel to make a decision on whether or not it should increase the frequency. For example when it is set to its default value of '80' it means that between the checking intervals the CPU needs to be on average more than 80% in use to then decide that the CPU frequency needs to be increased. To set this to something lower like 20% you would do the following.

sudo sh -c "echo 20 > /sys/devices/system/cpu/cpu0/cpufreq/ondemand/up_threshold"

ignore_nice_load - This parameter takes a value of '0' or '1'. When set to '0' (its default), all processes are counted towards the 'cpu utilization' value. When set to '1', the processes that are run with a 'nice' value will not count (and thus be ignored) in the overall usage calculation. This is useful if you are running a CPU intensive calculation on your laptop that you do not care how long it takes to complete as you can 'nice' it and prevent it from taking part in the deciding process of whether to increase your CPU frequency. To turn this on do the following.

sudo sh -c "echo 1 > /sys/devices/system/cpu/cpu0/cpufreq/ondemand/ignore_nice_load"

Conservative governor - CPU frequency is scaled based on current load of the system. It is similar to ondemand. The difference is that it gracefully increases and decreases the CPU speed rather than jumping to max speed the moment there is any load on the CPU. This would be best used in a battery powered environment.

sudo sh -c "echo conservative > /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor"

Conservative governor configuration options

freq_step - This describes what percentage steps the CPU freq should be increased and decreased smoothly by. By default the CPU frequency will increase in 5% chunks of your maximum CPU frequency. You can change this value to anywhere between 0 and 100 where '0' will effectively lock your CPU at a speed regardless of its load whilst '100' will, in theory, make it behave identically to the "ondemand" governor. For example to have it step up and down in increments of 10% you would do the following.

sudo sh -c "echo 10 > /sys/devices/system/cpu/cpu1/cpufreq/conservative/freq_step"

down_threshold - This is same as the 'up_threshold' found for the "ondemand" governor but for the opposite direction. For example when set to its default value of '20' it means that if the CPU usage needs to be below 20% between samples to have the frequency decreased. For example to set the down threshold to 30% you would do the following.

sudo sh -c "echo 30 > /sys/devices/system/cpu/cpu0/cpufreq/conservative/down_threshold"

sampling_rate - same as ondemand.
sampling_rate_(min|max) - same as ondemand.
up_threshold - same as ondemand.
ignore_nice_load - same as ondemand.

Performance governor - CPU runs at max frequency regardless of load. This module might not be listed in the running modules but is still available. My guess is it is built into the kernel for K/X/Ubuntu. Yours may be the same way.

sudo sh -c "echo performance > /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor"

Powersave governor - CPU runs at min frequency regardless of load.

sudo sh -c "echo powersave > /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor"

Cpufreq stats about your CPU

The cpufreq module lists stats about your CPU. These will help you find out things like the current frequency of your processor or what available frequencies your CPU can scale to. Check them out below.

cpuinfo_cur_freq - Show the current frequency of your CPU(s). You can also find this out by doing a "cat /proc/cpuinfo".

sudo cat /sys/devices/system/cpu/*/cpufreq/cpuinfo_cur_freq

cpuinfo_max_freq - Show the maximum frequency your CPU(s) can scale to.

sudo cat /sys/devices/system/cpu/*/cpufreq/cpuinfo_max_freq

cpuinfo_min_freq - Show the minimum frequency your CPU(s) can scale to.

sudo cat /sys/devices/system/cpu/*/cpufreq/cpuinfo_min_freq

scaling_available_frequencies - Show all the available frequencies your CPU(s) can scale to.

sudo cat /sys/devices/system/cpu/*/cpufreq/scaling_available_frequencies

scaling_cur_freq - Show the available frequency your CPU(s) are scaled to currently.

sudo cat /sys/devices/system/cpu/*/cpufreq/scaling_cur_freq

scaling_driver - Show the cpufreq driver the CPU(s) are using.

sudo cat /sys/devices/system/cpu/*/cpufreq/scaling_driver

scaling_max_freq - Set the maximum frequency your CPU(s) are allowed to scale to. Look at the output from scaling_available_frequencies above. Then you can pick one of those numbers (frequencies) to set to be the maximum frequency the CPU(s) will be allowed to scale to. For example if your CPU output from scaling_available_frequencies was 2000000 1800000 1000000 then you might set this to 1800000. So when the CPU scales it will only go to a max of 1800000 and not 2000000. An example on how to set this would be the following.

sudo sh -c "echo 1800000 > /sys/devices/system/cpu/cpu0/cpufreq/scaling_max_freq"

scaling_min_freq - Same as scaling_max_freq but setting a value that will not allow the CPU(s) to go below. For example.

sudo sh -c "echo 1800000 > /sys/devices/system/cpu/cpu0/cpufreq/scaling_min_freq"

Keeping any of your settings for cpufreq on reboot

Since all the settings we have been doing are in the /sys virtual file system they will not be saved after a reboot. You can go about setting these on reboot a few ways.

The first way is to put the lines you have been executing in /etc/rc.local. Since root executes rc.local on boot you don't need to sudo before each line. Your rc.local could look like the following example which sets the ondemand governor and the up_threshold to 20%. Don't forget to make sure the module for your CPU and the cpufreq scaling modules are set to load on boot also.

```
echo ondemand > /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor
echo 20 > /sys/devices/system/cpu/cpu0/cpufreq/ondemand/up_threshold
```

The second way to keep your settings on reboot is to install sysfsutils (sudo apt-get install sysfsutils). Then you could add the following lines to /etc/sysfs.conf which do the same thing as in the /etc/rc.conf example.

```
devices/system/cpu/cpu0/cpufreq/scaling_governor=ondemand
devices/system/cpu/cpu0/cpufreq/ondemand/up_threshold=20
```

Notes on cpufreq from my experiences

From what I have seen if you have 2 or more CPU's and you set the governor for one it will set that governor for all CPU's. At least it's that way on my AMD Athlon X2.

Every time the governor type changes all values get reset.

Completely Fair Scheduler (task scheduler) introduced in 2.6.23 seems to be causing problems (I think it's the scheduler??) on trying to set the up_threshold with ondemand. Let's say you have 2 or more processors and a task that is running that wants 100% of a processor. The scheduler is now so "fair" it bounces the task from one CPU to the next which causes each CPU to never get above the default ondemand up_threshold of 80%. This causes the ondemand governor to never let any processor get to it's full frequency and complete the task faster. Lowering the up_threshold to something like 20% fixes it. The task seems to stay on one processor long enough to get it above 20%. Then when it bounces back it's already scaled up from the last bounce. This keeps it up until the process is complete for all cores.