

Scaling CPU Speeds In Ubuntu (8.04)

Please note: this is a first draft and is liable to change after feedback, but it works! Please report any mistakes – Noel Slevin, 26/08/2008.

I am writing this guide to scaling CPU speeds not because there isn't any documentation out there already – there is, although it can be hard to find – and not because I am a technical genius – I know a fair bit, I guess, but I'm no Einstein. No, I'm writing this because it's something that I wanted to do but had to look in a number of places in order to find out how to do it. Furthermore, some of the documentation that I found was a little muddled and probably not easy enough for some people to follow. Thus, I am going to try and explain how to scale CPU speeds in Ubuntu's latest release, 8.04 (Hardy Heron). If you're not using Ubuntu 8.04, you may need to change a few things, but this guide should still be helpful as long as you are using a 2.6 kernel.

Before we start, a little disclaimer. Some of this guide will have you executing powerful commands and you could accidentally wreck your system. If you do this, it's not my fault. I'm not forcing you to do this, nor am I there to make sure you do it correctly. If you wreck your system, you've likely done something wrong. If so, you're best off going somewhere like the Ubuntu Forums for help, not here. If, however, you follow this guide and it just doesn't work, feel free to leave a comment or contact me and I can see if I can help this work for you, and perhaps update the guide. Like I said, I'm no expert, but I have, after some study, made CPU scaling work very nicely on both my laptop and my desktop, which are very different machines.

So, before we actually start doing anything, what exactly is CPU speed scaling, or as it is more technically known, CPU frequency scaling? It's quite simple really. A CPU is a central

processing unit – it processes everything as computer does. The faster the processor, the quicker it can execute commands – although it can, of course, be constrained by other elements of your hardware, like insufficient RAM. CPUs have a speed rating – my laptop has a Core2Duo T7300 rated at 2.00Ghz, so basically, it has 2 CPU cores, and they run at 2.00Ghz. My rather ageing desktop has a Pentium 4 HT processor running rated at 2.8Ghz, meaning it has 1 core, running at 2.8Ghz. However, whilst this is what these processors are rated at, this does not mean to say that the processors must run at this speed. Far from it. My laptop CPU can also run at 1.6Ghz, 1.2Ghz and 800Mhz. My desktop CPU can run at all sorts of speeds down to 350Mhz. This is known as speed stepping. Please note, this is not the same as overclocking (or underclocking)! I will explain later about how overclocking fits in with speed stepping. One final note before moving on to the “why” - CPUs do not have to run at a set speed – they can dynamically change speeds. This can be very useful, as will be explained next...

So why might you actually want to speed step your CPU? You might want to save power. This would be a good way to do that. You may not need all the processing power your CPU has, and thus not want to use it unnecessarily. Neither of these are particularly likely reasons, but there are two more likely reasons for speed stepping. Firstly, if you have a problem with overheating, turning the processor's speed down will reduce the effect of overheating. Secondly, you may be using a laptop and trying to save power to extend battery life. Speed stepping can, when used correctly, save power and extend battery life. Considering laptops can often overheat because of their compact nature and people's lack of care for them, the reduced heat could potentially prolong the life of the laptop too, whilst still leaving you with processing power when required. So, enough of the background, let's get on with actually implementing speed stepping on our Ubuntu system!

The first thing we need to do is make sure that we have a couple of packages installed. One of these is “powernowd”, and the other is “cpufrequtils”. To see if these two packages are installed, go to System → Administration → Synaptic Package Manager and type in the package names.

Once you have made sure that these two packages are installed, go to Applications → Accessories → Terminal. In the terminal, type “/sys/devices/system/cpu/”. Then press the “tab” button twice. If you have two processors, or a multi-core processor, you should see “cpu0” and “cpu1”. If you have more than two cores or processors, you will see more (one for each that you have); if you have just the one processor with one core, you should just see “cpu0”. If you still have the command on the line, type “cpu0/cpufreq”. Hopefully, it should find a cpufreq/directory. If it does, press the “tab” button twice again. A whole list of new options should be available, such as “scaling_driver”, “scaling_governor”, “scaling_max_freq”, “cpu_max_freq”, etc... I have 13 options on my system. We will use these later to probe the processor and to change our speed step settings.

Now, you need to copy and paste the following command into the terminal and press “enter”:

```
ls /lib/modules/$(uname -r)/kernel/drivers/cpufreq/ \  
/lib/modules/$(uname -r)/kernel\  
/arch/x86/kernel/cpu/cpufreq/
```

Please note: if you are using an earlier version of Ubuntu, you may need to replace the “x86” in the last line with “i386”. If you get an error message in the output from the above text, try replacing the text. Unfortunately, I don't know when that changed in the kernel – I just know it did. Anyway, what we're interested in is the output. The first output lists the available speed step drivers we can use to speed step the CPU. The second output lists the governors, which I will talk about later. The first list will consist of drivers such as “acpi-

cpufreq.ko”, “speedstep-centrino.ko”, “p4-clockmod.ko” and “powernow-k8.ko”. Hopefully, one of these drivers is the one we need – we just need to find out which one. With some logical thinking, and by paying attention to the terminal's output, we can actually do this quite easily. The format we need to use is the following:

```
sudo modprobe driver
```

Please note, although the drivers in the previous output had the extension “.ko”, we *do not* add that extension in the modprobe! So, if we are trying to add the p4-clockmod driver, we enter the following:

```
sudo modprobe p4-clockmod
```

If I accidentally add the “.ko” extension, I will get the following error:

```
FATAL: Module p4_clockmod.ko not found.
```

So the next question is, how do we know if we have loaded the correct driver? If we load the correct driver, there will be no output from the terminal. If we load the incorrect driver, however, we will see a message that says something along the lines of “Device not found”. And so, having evaded simple errors, we come to the million-dollar question: which driver is the right driver?

Good question! The simply answer is, “I don't know”! However, with some logical thinking, we can get somewhere. The best driver to try first is “acpi-cpufreq” as it is the most likely to work. This is the driver I use for my Core2Duo. If you have an nforce2 chipset, try “cpufreq-nforce2”. If you have a Centrino processor, try “speedstep-centrino”. If you have p4, like my desktop, try “p4-clockmod”. If you have an AMD processor, try one of the “powernow-6/7/8” drivers (use the number that's most appropriate, if you know which one that is).

But what if you don't know what kind of processor you have? That's not a position I'm familiar with, but I can still help you. For more information about your processor, type in the following command in the terminal:

```
cat /proc/cpuinfo
```

This will tell you lots of information you don't need, and a little that's helpful. Take a look at “vendor_id” and “model name” - these two labels will tell you what you need to know about your processor. Also look at “cpu Mhz” - this tells you the speed your processor is currently running at. Please note: it *does not* display the speed it is supposed to run at, or can run at, but the speed that *it is currently running at*.

Anyway, once you have successfully loaded a driver, you need set the speed or set a governor to automatically govern the speed. These are two very different things. Setting the speed is, in effect, manually throttling the CPU – especially effective if you're trying to stop the CPU from overheating. Setting a governor is a dynamic way of changing the speed of the CPU in accordance with the amount of power it needs at a given time – more effective for saving power on laptops. Decide which of these methods you want to take – for now – as they take different approaches (although we will be able to alter both later).

First, I'm going to go through how to manually set the speed of the CPU to a set level. Before we can set the level, however, we need to know what speeds the CPU is capable of running at. We do this by executing the following command:

```
$cat  
/sys/devices/system/cpu/cpu0/cpufreq/scaling_available_frequencies
```

This command will output a series of numbers, separated by spaces. These are the

speeds (in hertz, Hz), that the CPU is capable of running at. On my laptop, the CPU can run at 2Ghz, 1.6Ghz, 1.2Ghz and 0.8Ghz. Make sure you count the number of 0's! Decide which of the available frequencies you want to run the CPU at, and then execute it with the following command:

```
sudo cpufreq-selector -f value
```

Remember, 1Ghz == 1000000! The “-f” argument simply tells the program to set the frequency. You can select a particular CPU by using “-c” followed by the CPU number. That's it, if you were just wanting to set a new, constant CPU frequency! And, just to check the new frequency has been applied, we can issue the following command:

```
cpufreq-info
```

This will output a fair amount of information, one piece of which will be “current CPU frequency is...” If, however, you want *just* the current CPU frequency, you can issue a more long-winded command that will just output the frequency in hertz (Hz):

```
cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_cur_freq
```

If, however, you are wanting to set a governor, it's a little more long-winded, but not complicated. First, we want to go to the cpufreq/ directory:

```
cd /sys/devices/system/cpu/cpu0/cpufreq/
```

Now that we're here, we want to see which governor is currently set:

```
cat scaling_governor
```

The output will almost certainly be “userspace”, which basically means that the CPU has been set to a particular, constant speed. If you're reading this, I'm assuming that speed is

full speed! Anyway, what we want to do next is to find out the scaling_governors we are able to set, which we query using the following command:

```
cat scaling_available_governors
```

The output will hopefully be “ondemand”, “userspace”, “conservative”, “powersave” and “performance”. “Userspace” allows the user to set the CPU frequency manually, “powersave” runs the CPU at the lowest frequency, “performance” sets it to the maximum frequency. “Ondemand” and “conservative” are very similar – they both set the CPU frequency quite low where possible, and then raise the frequency as necessary to run commands. The difference, however, is that the “conservative” governor gradually steps up the speed, whereas the “ondemand” governor is capable of jumping straight to the fastest frequency step. Depending upon what you are doing, either of the two methods could save more or less power than the other, and it is probably in part down to personal taste. So, now you're an expert in the options, pick one and execute it in the terminal! I'm going to execute the “ondemand” governor.

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

The “sh -c” bit is telling bash to run the command in a shell. This is needed because the command is run as a superuser. You can run the command as root, but I really wouldn't advise it when you can do it like this. Because the command has to be run through a shell, the actual command has to be run within quotation marks. Anyway, once this has been successfully run, your system will be running the “ondemand” governor! But you can't see that yet, and you want to be able to see that this is working, right? Yes, you do. Read on. We're almost there!

Apologies to all non-GNOME users out there, but the following section is applicable to GNOME (but there's more after this section to interest you!). I don't use KDE or any other desktop, so I can't help you when it comes to a graphical utility for CPU frequency and governors outside of GNOME. However, if you do use the GNOME desktop, there is an applet for the GNOME panel that can help us. However, first of all, we need to reconfigure the "gnome-applets" package so as to give it superuser privileges. We do this using the following command:

```
sudo dpkg-reconfigure gnome-applets
```

You will be asked a question, to which you need to answer "yes". Then, right-click on the GNOME panel, click "add to panel" and select "CPU Frequency Scaling Monitor". Once you have placed it in the panel, you can left-click on the applet and select either a frequency, or a scaling governor. We're done!

However, for your piece of mind, I have added a few notes on the bottom here to explain a few other things. Take a quick look here if you experience any difficulties.

The command "cat cpuinfo_cur_freq" and "cat scaling_cur_freq" output the same information, but the former requires superuser privileges whereas the latter doesn't. The commands "cat cpuinfo_max_freq" and "cat scaling_max_freq", and "cat_cpuinfo_min_freq" and "cat scaling_min_freq" give the same relative output, but in this case, none of the commands require superuser privileges. No, I don't know why either!

If you want to find out which driver you are currently using, use the command "cat scaling_driver". To check the current governor, use the command "cat scaling_governor". To check how long your computer has been in each CPU frequency state, execute the

command "cat stats/time_in_state".

You can artificially set a minimum and maximum CPU frequency state using these commands too. To check the current maximum state, execute the command "cat scaling_max_freq", and for the minimum, execute "cat scaling_min_freq". If you wanted to set the minimum frequency to 1.2Ghz, you would execute the following command:

```
sudo sh -c "echo 1200000 >
/sys/devices/system/cpu/cpu0/cpufreq/scaling_min_freq"
```

And to set the maximum frequency at, say, 1.6Ghz, you would execute the following command:

```
sudo sh -c "echo 1600000 >
/sys/devices/system/cpu/cpu0/cpufreq/scaling_min_freq"
```

Easy! I don't know what happens if you accidentally set the maximum frequency to a lower value than the minimum – you'll probably get an error message though.

As far as I am aware, that is everything and this tour through CPU frequency scaling is complete! If you use this and it works, please leave a comment to let me know. If you tried and it didn't, leave a comment and let me know. If you don't understand, or you find a mistake, leave a comment and let me know! If you're still reading, you're a committed reader. Leave a comment and let me know. Maybe I'll buy you a drink sometime!