Homepage of Axel Kohlmeyer

- Welcome to my Homepage.
- Axel's Old News
- About Myself
- Curriculum Vitae
- Publications
- In the Press
- Images and Movies
- Lectures and Talks
  - ICTP/TWAS Caribbean School 2012
  - Math 8210 - Spring 2012
  - Temple HPC Intro Summer 2011
  - Introduction to HPC Fall 2010
  - GPU Technology Conference 2010
  - ICTP Schools on HPC and Grids
- Software
  - LAMMPS-ICMS
  - LAMMPS VMD plugin
  - TopoTools
  - HOOMD VMD plugin
  - LJMD
  - TclMPI
  - VRPN-ICMS
- Random Hacks
  - Text-mode 2D-Graphics for Fortran
  - XRandR Chooser GUI with Presets
  - VMD Init Script
  - Irregular VMD Representation Update
  - Nvidia GPU Coolness
  - AMD GPU Coolness
  - Pimp Your S1070

Recent site activity

Curriculum Vitae
edited by Axel Kohlmeyer

Random Hacks >

Nvidia GPU Coolness

Contents

Background

There are quite a few people that use Nvidia GPUs for computing purposes these days, and some of them - like me - are concerned about reliability and longevity of their devices due to the way how the fan speed is regulated by default. In short the GPUs are running too hot without any real need. G200 generation GPUs are run around 70C and for GF100 (Fermi) generation GPUs the "magic" temperature seems to be 85C. However, fan speed at those temperatures is often very low and even in a typical desktop environment, one could easily tolerate an increased fan speed without noticing it. In my specific case, it would have to be about 75% until I can can tell it apart from the air conditioning and other noises of the computer.

nvclock

A few years ago, nvclock, appeared and that seemed to solve the problem (and also satisfy those crazy people for whom fast is not fast enough), but it produces a lot of warnings from the kernel, is very machine and GPU specific and - most importantly - seems to be abandoned and thus does not support newer GPU models.

nvidia-settings

At some point Nvidia developers must have understood, that it is impossible to keep people from messing with the GPU settings, and rather than having them use strange and uncontrolled hacks, they exposed some of the GPU control features (clock/memory speeds and fan speed) in their driver via the NV-CONTROL extension to client programs like the nvidia-settings tool. In order to keep users on shared machines from doing harm to the hardware, these options have to be explicitly enabled by the system administrator through the Coolbits option in the `xorg.conf` X server configuration file. The tool also has a command line interface, as shown here, which makes it a prime target, although there is an additional problem that needs to be overcome.

The Problem

Having to use X

The problem with the `nvidia-settings` tool is that, unlike `nvidia-smi`, it connects to an X server and thus controls the GPUs that are connected to this X server. No big deal when you are using a desktop, but quite a problem, when you use

Redirects
removed by Axel Kohlmeyer
attachment removed by Axel Kohlmeyer

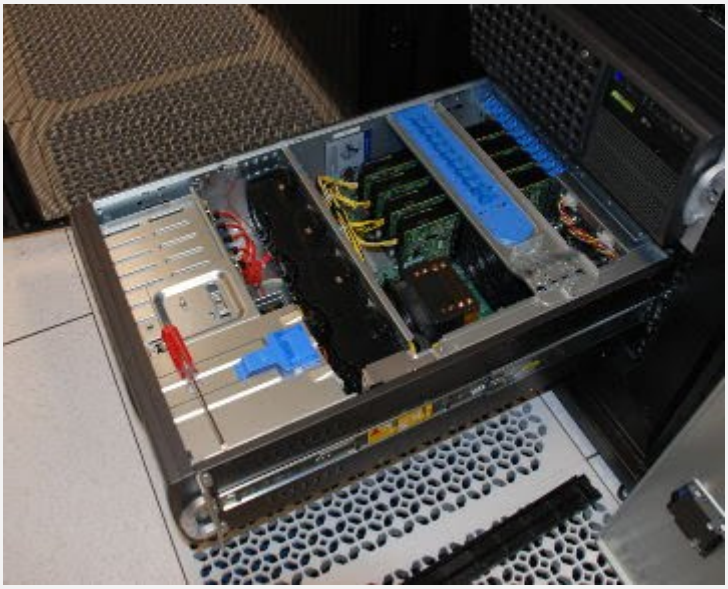
Curriculum Vitae
edited by Axel Kohlmeyer
attachment from Axel Kohlmeyer

Welcome to my Homepage.
edited by Axel Kohlmeyer

Curriculum Vitae
edited by Axel Kohlmeyer

View All

Disclaimer



remote GPUs. If you run `nvidia-settings` on those machines, you will still control the GPU(s) on your desktop and not those on the remote (headless) server.

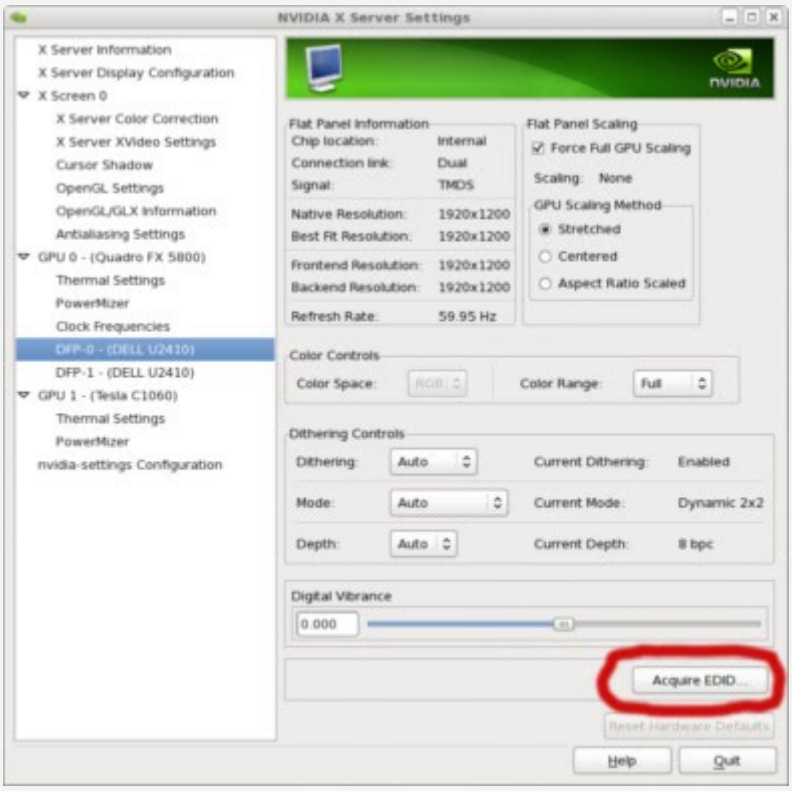
Need a Real Display?

To add insult to injury, the Nvidia driver for the X.org X server checks if there is a display connected and aborts, if there is none. So if you would think that you could bypass the issue by simply running an X server on the remote machine, that looked like a dead end, too. Next, if you are running Tesla type GPUs for computing, only one of them will be allowed to connect to a display, which in turn makes it impossible to control the GPUs without display via the NV-CONTROL extension.

Not Losing It

And the final issue to cope with is that the driver is "losing" all settings and going back to its default settings when the first device context is created. When running an X server, the X server will be holding this context on desktops, but you don't want to keep running an X server on a compute node, as that also will add a watchdog timer that will keep GPU kernels from hogging the GPU too long. On a compute node, that should not be a problem and thus allowed.

The Hack



In order to achieve GPU coolness on headless compute nodes, now a series of steps are required that address each of the issues listed in the section above. Please note that this hack has only been tested with CUDA 3.2 as well as CUDA 4.1 and the associated drivers. Apparently Nvidia engineers don't like their customers to worry about protecting their investments, and rather would like them to purchase - less effective and more expensive - passively cooled GPU hardware for data center use.

Keeping a GPU Context

For CUDA 3.2, this is easiest done through the log function of the `nvidia-smi` tool: we launch it in the background, make it poll the GPUs infrequently and send the resulting log directly to `/dev/null`. Keeping a GPU context open will not only preserve the changes that we apply to the GPU settings, it will also remove the several seconds long delay resulting from the re-initialization of the GPU devices when the first device context is created and thus makes using other tools that use the GPU via opening and closing the GPU devices much more efficient. The drivers bundled with CUDA 4.x allow to run the GPUs in a so-called "persistence mode", which has the same effect, but doesn't require some application to run.

Running nvidia-settings

To be able to use `nvidia-settings`, we have to have an X server configured and running, but thanks to keeping a GPU context on the GPUs open, we only need to run it until the settings are applied and then have it shut down again. The `xinit` command is the perfect tool to do this.

Faking a "Head" for a Headless X Server

The biggest remaining challenge is now to make the X server launch properly without having a display attached. Nowadays, display settings are negotiated between the X server and the display via EDID, and this is how we can simulate a display. The X server allows to override EDID settings and to define which display to configure through settings in the /etc/X11/xorg.conf file. All that is missing is a valid EDID file and this can be obtained from `nvidia-settings` through the "Acquire EDID" button, when examining the properties of a currently attached display (doesn't matter which one). In the xorg.conf file, something along the lines of the following has to be set.

```
Section "Screen"
    Identifier     "Screen0"
    Option         "UseDisplayDevice" "DFP-0"
    Option         "ConnectedMonitor" "DFP-0"
    Option         "CustomEDID" "DFP-0:/etc/X11/dfp-
edid.bin"
    Option         "Coolbits" "5"
    ....
End Section
```

Dealing With Multiple C2050 Tesla Cards

As mentioned above, if your headless GPU server contains only Tesla C2050 or similar GPUs, you cannot run (or fake) multiple screens. Thus it is necessary to run multiple X servers, one for each GPU. To select the right GPU, we need to set the BusID keyword to uniquely identify each display. To make life easier, one can just

write a little script that parses the output of `lspci` and inserts the bus id into a template xorg.conf file and then loops just overy all individual GPUs.

Putting it All Together

Setting the GPU fan speeds is best done at boot time, so the steps from above are best scripted and then executed during boot. This way there is no disruption of ongoing GPUs jobs and the overhead of launching one or more X servers has no significant impact. The easiest way to deal with this, is to write a script that can (also) be managed like other scripts in /etc/init.d/ and activated through `service cool_gpu start` or managed via `chkconfig` for automatic execution during the boot process.

The Result

GPU temperatures are significantly lower on our GPU servers with 4xTesla C2050 each. The GPU temperature under full load drops on average from around 85C with the default settings to less than 60C with the fan speed set to 85%. This is quite a pleasing outcome of this little hack.

The Code

A tar.gz bundle with the final chkconfig compatible script, xorg.conf, and a suitable EDID file is available from: http://klein-group.icms.temple.edu/akohlmey/files/set-gpu-fans.tar.gz

Installation

The tar archive unpacks itself into a directory `set-gpu-fans`. You can move that, for example, into `/opt`. Check out the first part of the `cool_gpu` script and apply any changes that might be needed. The most convenient way to launch this would be to make a symbolic link to the cool_gpu script in /etc/init.d/ and then run chkconfig --add cool_gpu (on redhat or fedora machines). This will launch it automatically at each boot.

Comments

Commenting disabled due to a network error. Please reload the page.

You do not have permission to add comments.