User Guide for Ubuntu 10.10

This paper contains practical information for students attending the Ph. D summer school in Scientific GPU Computing, May 2011. After reading this, you should be able to compile and run the exercises using Ubuntu 10.10 installed on the machines in the VR-databar. This tutorial can also serve as inspiration for those who wishes to use their own computer running Linux with a CUDA capable graphics card, however this is only for experienced users.

Access to workstations

Workstations that can be used for completing the exercises are equipped with Ubuntu 10.10 and NVIDIA Quadro FX 5800 graphics cards. The lab is located in room 017 in Building 305. The login to Ubuntu on the VR-databar machines is not connected to the DTU login, so anyone who wants to use Ubuntu must contact the teacher assistant (TA) for a personal login which can be used for the entire week.

Starting up Ubuntu

Turn on the machine or restart it if the machine is already in Windows. After the BIOS has finished its checks, you will be presented with a couple of options. The first option, which is `Boot from hard disk`, will boot Windows and this option will automatically be selected after 10 seconds. Since you are reading this guide because you want to use Ubuntu, select the bottom option which is called `Linux Desktop`.

Logging in

Once Ubuntu has started up, you will be presented with the login screen. As already mentioned, do not use your DTU login. Contact the TA for a login. Should you mistype the username or password, the machine will respond with `No response from server, restarting...` which is quite misleading. Please ensure you are typing the correct username and password before contacting a TA.

Getting the exercises

Once you have logged in, use Firefox to download the exercises from CampusNet or the PhD school website at
(`http://gpulab.imm.dtu.dk/PhDschool2011/Materials.html`). The downloaded file will by default be stored in the `Downloads` folder in the home directory, so click on `Places` and then `Downloads`. Simply extract the file by right-clicking it and choosing `Extract here`.

Editing the exercises

You are free to use whichever of the installed text editors you prefer to edit the code.

There is `gedit` (Text Editor) for basic text editing, `geany` for those who want more a bit than just syntax highlighting as well as `Eclipse` for those who want a full-blown IDE, although you will have to set-up an Eclipse project for the exercises yourself. There is also `VIM` and `nano` for those who want to use the terminal.

Building the exercises

All the exercises come with a Makefile which will build the exercise. Simply open a terminal and navigate to the directory for the exercise you wish to build. Then you can simply run `make` to build the exercise.

The Makefile contains a couple of variables which are described in table 1 below. For this course, it should be sufficient to simply change these variables. Modify the `*FLAGS` variables when you wish to change compiler or linker options, such as specifying CUDA compute architecture, and add an object file, eg. `name.o`, for each source file, eg. `name.c/name.cpp/name.cu`, you add.

Table 1: *Makefile variables*

Variable	Description
TARGET	This is the name of the binary output file.
OBJS	This is a space-separated list of object files to link.
CFLAGS	This is the flags to pass to the gcc compiler for all the .c files.
CPPFLAGS	This is the flags to pass to the g++ compiler for all the .cpp files.
NVFLAGS	This is the flags to pass to the nvcc compiler for all the .cu files.
LDFLAGS	This is the flags to pass to the linker.

Compute Visual Profiler

The Compute Visual Profile is located `/usr/local/cuda/computeprof/bin/computeprof`. However, the environment should be setup such that you can run it by simply executing `computeprof` from the terminal.

Environment variables

A couple of environment variables have been defined to ease the use of CUDA on the machines, which are listed in table 2. These are the same environment

variables which are available on the Windows installation. If you use nvcc to compile and link, it will automatically include the correct include and library paths, but should you for some reason need to include or link with CUDA with a different compiler, then you can use these environment variables.

Table 2: *Environment variables*

Variable	Description
CUDA_BIN_PATH	Path to the CUDA binary files such as nvcc.
CUDA_INC_PATH	Path to the CUDA include path.
CUDA_LIB_PATH	Path to the CUDA library path.
NVSDKCOMPUTE_ROOT	Path to CUDA SDK.
NVSDKCUDA_ROOT	Path to CUDA SDK examples (sdk/C).