CS-206 - CUDA Assignment

---

1 Introduction

In this assignment, you will compare the speed of a serial matrix multiplication code with a parallel one on GPU. To do this, you need to write the kernel code for matrix multiplication in CUDA.

2 Preparation

You need to connect to Deneb cluster to compile and run your code. To connect to Deneb cluster, you need to use your GASPAR username and password as follow:

```
ssh -Y YOUR_USERNAME@deneb1.epfl.ch
```

`YOUR_USERNAME` is the username of your GASPAR account.
You also need to have your code on the server. So, open a new terminal and go to the path where the `Q1` directory in the assignment is located. Copy the `Q1` directory in the assignment from your computer into your home directory on the server:

```
cd PATH_TO_Q1_DIRECTORY
scp -r ./Q1 YOUR_USERNAME@deneb1.epfl.ch:/home/YOUR_USERNAME
```

On the server, go to the `Q1` directory:

```
cd Q1
```

In order to compile a CUDA code, you need to load cuda module:

```
module load cuda/6.0
```

Now you can compile CUDA code. To run it, you need to send a batch job to the server. The file `q1_username.run` is a script you need to use to submit your job. You need to modify this file to specify your username. By the following instruction you can change `username` to your GASPAR username:

```
sed 's/username/YOUR_USERNAME/g' q1_username.run > q1.run
```

`YOUR_USERNAME` is the username of your GASPAR account.

3 Modify the code

In this assignment, we want to multiply two N*N matrices (a and b) sequentially and in parallel using the GPU, and then measure the speedup. You have to modify the q1.cu file. Open it in an editor (e.g., `vim q1.cu`). Sequential code is given. You need to write the kernel code for matrix multiplication in function `__global__ void matrix_mul(float *a, float *b, float *c)`. `a` and `b` are input matrices and `c` is the matrix that you should write the result in.

Besides, you also need to copy the data from the CPU to the GPU, run the kernel, and copy the result back from the GPU to the CPU. (The section that you should add your code is shown in q1.cu).

You can find more information about CUDA programming in the Links on the course web page.

4 Compile and run your code

First you need to be sure that you have loaded the cuda module, run the following command:

```
nvcc --version
```

You should see the following response:

```
nvcc:  NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2013 NVIDIA Corporation
Built on Thu_Mar_13_11:58:58_PDT_2014
Cuda compilation tools, release 6.0, V6.0.1
```

If you see an error, load again the cuda module by the instruction in Preparation section.
Now you are ready to compile CUDA code. You can compile the `q1.cu` file by running:

```
make
```

(You will receive a warning, just ignore it).
This command make a `q1` file (if you don't receive any error).
Now you can submit a job to the server:

```
sbatch -A cs206 --reservation=cs-206 q1.run
```

You will get a response:

```
Submitted batch job JOB_ID
```

`JOB_ID` is a number that shows your job ID. You can find the result of your job in the file: `slurm-JOB_ID.out`.
You can see the result:

```
cat slurm-JOB_ID.out
```

If your code runs correctly, you can see the time spent on each function (sequential and parallel) and the speedup. Note that it takes time for the program to complete.

5 Report

Write a short description of your code and, for N = 1024 and N = 2048, report the time spent on each function and the speedup. The report must be short (at most 2 pages).

6 Bonus

The top 5 groups who get the highest speedup will get a 20 point bonus. To be eligible for the bonus a group has to explain the gained speedup in the report.

7 Submission

Deadline: 19.05.2015

Submit a pdf for your report and the modified q1.cu in a zipped folder.

Version 1.0 of 26th April 2015, EPFL ©2015