GPU Computing Assignment

May 31, 2017

Abstract

This is the third project of our course. The project is released on June 1st, 2017. The deadline is 5:00pm, June 18th, 2017. Please send the report to xuelinq@163.com. The late submission is also acceptable; however, you will be penalized 10% of total scores for EVERY ONE DAYS' delay (by 5:00pm of that day).

(2) Note that if you are not satisfied with the initial report, the updated report will also be acceptable given the necessary score penalty of late submission.

(3) OK! That's all. Please let me know if you have any additional doubts of this project. Enjoy!

1 Question 1: Common Errors (30 points)

The GPU coding will make heavy use of low-level C constructs and concepts, especially pointers and memory management. Thus there are a few quick samples of code and their intended specifications. Each such piece of code is incorrect. Identify what is wrong with the code, and how it should be fixed. Note that many of these problems allude to common errors encountered while writing both GPU and CPU code.

1.1 Pointer-1

Creates an integer pointer, sets the value to which it points to 3, adds 2 to this value, and prints said value.

```
void test1()
{
int *a = 3;
*a = *a + 2;
printf("%d\n", *a);
}
```

1.2 Pointer-2

Creates two integer pointers and sets the values to which they point to 2 and 3, respectively.

```
void test2()
{
int *a, b;
a = (int *) malloc(sizeof (int));
b = (int *) malloc(sizeof (int));
if (!(a && b))
{
printf("Out of memory\n");
exit(-1);
}
*a = 2;
*b = 3;
}
```

1.3 Pointer-3

Allocates an array of 1000 integers, and for i = 0, ..., 999, sets the i-th element to i.

```
void test3()
{
int i, *a = (int *) malloc(1000);
if (!a)
{
printf("Out of memory\n");
exit(-1);
}
for (i = 0; i < 1000; i++)
*(i + a) = i;
}
```

1.4 Pointer-4

Creates a two-dimensional array of size 3x100, and sets element (1,1) (counting from 0) to 5.

```
void test4()
{
int **a = (int **) malloc(3 * sizeof (int *));
a[1][1] = 5;
}
```

1.5 Pointer-5

Sets the value pointed to by a to an input, checks if the value pointed to by a is 0, and prints a message if it is.

```
void test5()
{
int *a = (int *) malloc(sizeof (int));
scanf("%d", a);
if (!a)
printf("Value is 0\n");
}
```

2 Question 2: Vector addition (70 points)

Important note: Use *cudaSetDevice(1)* or *cudaSetDevice(2)*, as the first statement in the GPU program, to execute your code on device 1 or 2 depending on which one is free.

2.1 CPU implementation

Code, in C, the vector addition program discussed in class, such that all execution is done on the CPU. The vector length n should be a command line input argument (argv). Use the random number generator (rand) to create the input vectors A and B. Define A, B, and $C = A + B$ as single precision (float) arrays. Output the sum s of all elements in C, $s = \sum_{i=1}^{n} C_i$. You may use as a starting point, the code in the CUDA programmers guide *http://docs.nvidia.com/cuda/cuda-c-programmingguide/index.html.*

2.2 GPU implementation

Modify the above code so that the computation $C = A + B$ is done on the GPU. Setting up the input vectors A and B, and the computation of s should still be done on the CPU as before. You may use, the code in the CUDA programmers guide *http://docs.nvidia.com/cuda/cudac-programmingguide/index.html*, as a guide.

2.3 CPU vs GPU comparison

Use the Linux "time" command to execute the CPU and GPU code for $n = 10^6, 5 \times 10^6, 10^7, 5 \times 10^7$. Verify that the values of s are the same for the CPU and GPU runs.

Plot execution time t as a function of n, for the CPU and GPU runs. Fit a line $t = c_1 + nc_2$ to the data and show it on the plot.

2.4 Conclusions

Assume that the total computation time is the sum of (1) setup time, (2) computation time, and in the case of the GPU, (3) data transfer time between the CPU and GPU. What do c_1 and c_2, calculated above, represent for the CPU? For the GPU? Now, also assume that the GPU is able to parallize the computation such that computation time is approximately 0, and that the setup time is the same for the CPU and GPU. Explain the difference in run times between the CPU and GPU. (Hint: c_1 and c_2 represent two different constants for the CPU and GPU.)

3 Deliverables

Please submit the following:

1. C code for CPU and GPU.
2. Report in pdf format including multiple plots and explanation of the results, *i.e.* comparisons of CPU vs GPU. What is your over-all conclusion as to when one would and would not want to use the GPU?