

**ME759**  
**High Performance Computing for Engineering Applications**  
**Assignment 4**

Date Assigned: September 30, 2013  
Date Due: October 7, 2013 – 11:59 PM

**Problem 1.** Read the article <http://www.nytimes.com/2013/09/26/science/researchersbuild-a-working-carbon-nanotube-computer.html> and

- a) Explain why this discovery is relevant
- b) Indicate two points/issues/observations brought up in the article that have been also discussed in ME759

**Answer:**

(a) This article answers the question to the speculation about what will be the next step when Silicon, the standard material in today's microelectronic circuits, reaches its fundamental limits for use in increasingly small transistors.

**Supporting the Moore's Law**, the constant shrinking of transistor size over the last half-century has been important because it has significantly lowered the cost of computing, making it possible to build ever more powerful computers that are faster and cheaper, and consume less power with each generation. This will eventually lead to improvement in the performance of Computing at lower cost.

(b) The issues common to ME759 brought up in this article are:

- 1. **Emphasis on improving the performance of Computing at significantly reduced cost.**
- 2. **Emphasis on low power consumption for high performance computing.**

**Problem 2.**

Write a program that multiplies a matrix **A** of dimension 16 (height) by 32 (width) with a vector **b** of dimension 32. All entries in **A** and **b** are integers. Specifically,

$A[i][j]=i+j$ , and  $b[i]=i$ . Your program should allocate memory on the device, copy the matrix **A** and vector **b** into the global memory of the device, invoke a kernel to carry out the multiplication on the device, bring back the result to the host, write the result out to file *problem1.out* (one value per row), and finally free host and device memory.

Please report on the forum, under topic "HW4 – Timing Results" the inclusive time required to finish the problem (inclusive time is time for solving the problem that includes the time required to pass data down into the device and then copy back to host the result).

When you present timing results in your report, make sure you indicate the GPU card that you worked off. Remember that there are some C2050, C2070, and GTX480 GPU cards that combine for the set of 56 GPUs available on Euler. To figure out which card you ran on, use the function **cudaGetDeviceProperties**. Look at slide 9 of the 09/27 handout to see how this function is called.

**Answer:** Inclusive Timing : 0.061632 ms on GeForce GTX 480

**Please find the program file named 'hw4p2t.cu' attached in the .zip folder**

### Problem 3.

a) Given the provided four files: *matrixadd.cu*, *matrixadd.h*, *matrixadd\_gold.cpp*, *matrixadd\_kernel*, use CMake to generate a *makefile* that is subsequently used in Eclipse to build a CUDA executable. Information about the CMake utility is available here:

<http://sbel.wisc.edu/Courses/ME964/Literature/CMake.pdf>

b) Edit the *MatrixAddOnDevice( ... )* function in *matrixadd.cu* and the *MatrixAddKernel( ... )* function in *matrixadd\_kernel.cu* to complete the functionality of the matrix addition on the device. Do not change the source code elsewhere. The size of the matrices is fixed to 4096.

c) There are several modes of operation for the application.

i) No arguments: The application will create two randomly initialized matrices to add. After the device multiplication is invoked, it will compute the correct solution matrix using the CPU, and compare that solution with the device-computed solution. If it matches (within a certain tolerance), it will print out "Test PASSED" to the screen before exiting.

ii) One argument: The application will use the random initialization to create the input matrices and write the device-computed output to the file specified by the argument.

iii) Two arguments: The application will initialize the two input matrices with the values found in the files provided as arguments. No output is written to file.

iv) Three arguments: The application will read its inputs from the files provided by the first two arguments and write its output to the file provided in the third.

Note that if you wish to use the output of one run of the application as an input, you must delete the first line in the output file, which displays the accuracy of the values within the file. The value is not relevant for this application.

As part of your homework, provide a text file, Word Document, or PDF file with your answers to the following questions.

**Please find the folder hw4 in the .zip folder which has all the cmake files attached.**

1. How many times is each element of the input matrices loaded in the kernel code?

**Answer:** Each element of the input matrices is loaded only once in the kernel mode.

2. What is the memory-access to floating-point computation ratio in each thread?

Assume that any local variables are in registers.

**Answer:** Since for each thread, there are two reads and one write operation, so the ratio of the memory-access to floating-point computation is **3:1**.

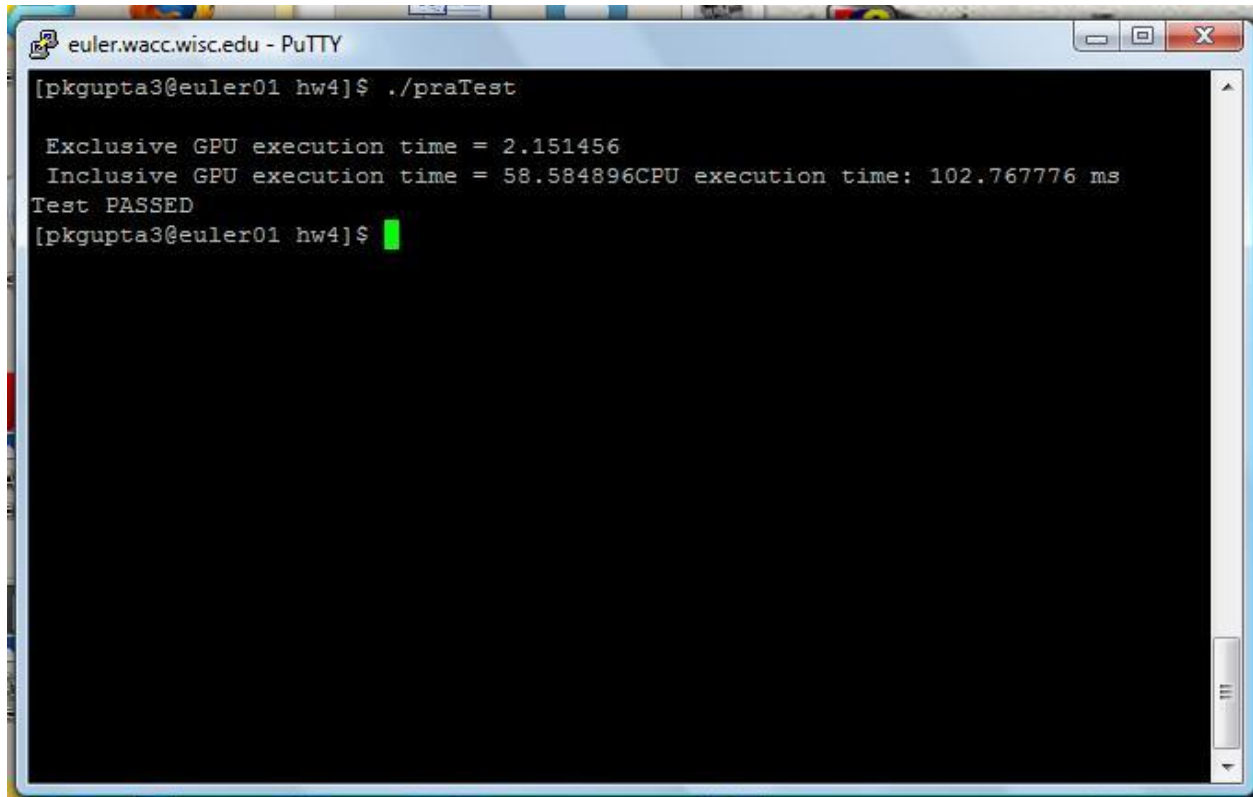
3. Report the amount of time it takes the host to carry out the matrix addition.

**Answer:** The amount of time is the CPU execution time: 102.77 ms.

4. Report the amount of time it takes the device to carry out the matrix addition. Two different execution times are of interest. The first execution time reported below, called "Exclusive GPU time", does not include the copying of the two input matrices to, and one output matrix from, the device. The second time reported is called "Inclusive GPU

time” and does include the copying time. Please paste in your report a picture similar to the one below:

**Answer:**

A screenshot of a PuTTY terminal window titled "euler.wacc.wisc.edu - PuTTY". The terminal shows the command `./praTest` being executed. The output is as follows:  
`Exclusive GPU execution time = 2.151456`  
`Inclusive GPU execution time = 58.584896CPU execution time: 102.767776 ms`  
`Test PASSED`  
The prompt `[pkgupta3@euler01 hw4]$` is visible at the bottom with a green cursor.

5. When comparing CPU to GPU performance for a problem such as matrixadd above, should the “CPU execution time” in the picture above be compared to the “Exclusive GPU execution time” or the “Inclusive GPU execution time”? In other words, what would be a fair comparison here?

**Answer: Comparison with Exclusive GPU execution time is fair because it is the time taken for computation process.**

**Grading for this last problem:**

Correctness: 25%

- Produces correct result output file for provided inputs

Programming: 20%

- Correct usage of CUDA library calls and C extensions.

- Correct usage of thread id's in matrix computation.

Report: 55%

- Answer to question 1: 15%, answer to question 2: 10%, answer to question 3: 10%, answer to question 4: 10%, answer to question 5: 10%.