



(../index.php)

**CS 475: Parallel  
Programming**  
*Computer Science  
Department*  
(<http://www.cs.colostate.edu>)  
Fall 2016  
Assignment 5 - CUDA First  
Steps



(<http://welcome.colostate.edu/>)

## Introduction

The purpose of this exercise is for you to learn how to write programs using the CUDA programming interface, how to run such programs using an NVIDIA graphics processor, and how to think about the factors that govern the performance of programs running within the CUDA environment.

For this assignment, you will modify two provided CUDA kernels and empirically test and analyze their performance. The first is part of a program that performs vector addition. The second is part of a program to perform matrix multiplication. Download and untar the PA5.tar (PA5.tar) file. Follow the instructions in Lab-6 for compiling and running CUDA program.

Resources that may be useful for this assignment include:

- The NVIDIA CUDA Developer web site (<http://developer.nvidia.com/page/home.html>)
- The NVIDIA CUDA Zone web site ([http://www.nvidia.com/object/cuda\\_home.html](http://www.nvidia.com/object/cuda_home.html))
- Documentation for gnu make (<http://www.gnu.org/software/make/manual/make.html>)
- Also, you can check out the latest version of the NVIDIA CUDA Programming Guide (<https://developer.nvidia.com/nvidia-gpu-programming-guide>).

## Vector add: coalescing memory accesses

As discussed in class, vecadd is a micro benchmark to determine the effectiveness of coalescing. You are provided with a non-coalescing version, and it is your job to create a coalescing version, and to measure the difference in performance of the two codes.

Here is a set of files provided in PA5.tar for Vecadd:

- Makefile
- vecadd.cu

- vecaddKernel.h
- vecaddKernel00.cu
- timer.h
- timer.cu

In vecadd01(new executable name) you will use a new device kernel, called vecAddKernel01.cu, to perform vector addition using coalesced memory reads. Change the makefile to compile a program called vecadd01 using your kernel rather than the provided kernel.

## Shared / shared CUDA Matrix Multiply

For the next part of this assignment, you will use matmultKernel00 as a basis for another kernel with which you will investigate the performance of matrix multiplication using a GPU and the CUDA programming environment. Your kernel should be called matmultKernel01.cu. Your code files must include internal documentation on the nature of each program. You can have intermediate stages of this new kernel, eg with and without unrolling, but we will grade you on your final improved version of the code and its documentation.

Here is a set of provided files in PA5.tar for Matmult:

- Makefile
- matmult.cu
- matmultKernel.h
- matmultKernel00.cu
- timer.h
- timer.cu

When run with a single parameter, the provided code multiplies that parameter by FOOTPRINT\_SIZE (set to 16 in matmult00) and creates square matrices of the resulting size. This was done to avoid nasty padding issues: you always have data blocks perfectly fitting the grid. Here is an example of running matmult00 on figs:

```
figs 46 # matmult00 32
Data dimensions: 512x512
Grid Dimensions: 32x32
Block Dimensions: 16x16
Footprint Dimensions: 16x16
Time: 0.016696 (sec), nFlops: 268435456, GFlopsS: 16.077853
```

Notice that parameter 32 value creates a  $32 \times 16 = 512$  sized square matrix problem.

In your new kernel each thread computes four values in the resulting C block rather than one. So now the FOOTPRINT\_SIZE becomes 32. ( Notice that this is taken care of by the Makefile.) You will time the execution of this new program with matrices of the sizes listed above to document how your changes affect the performance of the program. To get good performance, you will need to be sure that the new program coalesces its reads and writes from global memory. You will also need to unroll any

loops that you might be inclined to insert into your code.

Here is an example of running my matmult01 on figs:

```
figs 28>matmult01 16
Data dimensions: 512x512
Grid Dimensions: 16x16
Block Dimensions: 16x16
Footprint Dimensions: 32x32
Time: 0.012185 (sec), nFlops: 268435456, GFlopsS: 22.030248
```

Notice that parameter 16 now creates a  $16 \times 32 = 512$  square matrix problem, because the footprint has become  $32 \times 32$ . Also notice a nice speedup.

## Report

Use any GPU found in CS department machines. (GPUs found in 120(state-capitals), 225 and 325 lab machines have compatible CUDA driver and runtime versions)

1. Collect and compare the time vecadd00 and vecadd01 take with the following number of values per thread: 500, 1000, 2000.
2. Collect and compare the time matmult00 and matmult01 take for square matrices of size:  $256 \times 256$ ,  $512 \times 512$ ,  $1024 \times 1024$ .
3. For matmult00 and matmult01, investigate how each of the following factors influences performance:
  - The size of matrices to be multiplied
  - The size of the block computed by each thread block
  - Any other performance enhancing modifications you made
4. Can you formulate any rules of thumb for obtaining good performance when using CUDA?
5. How does the performance (GFLOPsS) of your improved versions compare to the theoretical machine peak? Can you come up with a heuristic as how to improve it or reason why it is not possible? (Hint: Roofline)


## Submission

Submit your report(PA5.R.pdf) through Checkin.

Submit a tarball(PA5.1 and PA5.2) of your code that contains atleast

- Makefile
- matmult.cu
- matmultKernel.h
- matmultKernel01.cu
- vecadd.cu
- vecaddKernel.h
- vecaddKernel01.cu

You can submit a README file if you want.

Secs.	Contact CSU ( <a href="http://www.colostate.edu/info-contact.aspx">http://www.colostate.edu/info-</a>	
Originating IP	<a href="http://www.colostate.edu/info-disclaimer.aspx">contact.aspx</a>   Disclaimer	
123.230.131.249	( <a href="http://www.colostate.edu/info-disclaimer.aspx">http://www.colostate.edu/info-disclaimer.aspx</a> )	
User: Guest (../..	Equal Opportunity ( <a href="http://www.colostate.edu/info-equalop.aspx">http://www.colostate.edu/info-</a>	
/ztools	<a href="http://www.colostate.edu/info-equalop.aspx">equalop.aspx</a> )	
/authenticateLogin.php)	Colorado State University, Fort Collins, CO 80523	
	USA	
	© 2016 Colorado State University	

---