Lab 2

Keith Evan Schubert

August 31, 2018

1 Objective

The purpose of this lab is to implement a basic dense matrix multiplication routine.

2 Activity

1. Login to kodiak. cd to your mpplabs directory and type `git pull`.
2. Edit the file `<lab-directory>/main.cu` to implement the following where indicated:
   (a) Allocate device memory
   (b) Copy host memory to device
   (c) Copy results from device to host
   (d) Free device memory
3. Edit the file `<lab-directory>/kernel.cu` to initialize the thread block and kernel grid dimensions and invoke the CUDA kernel, and to implement the matrix multiplication kernel code..
4. Compile and test your code.

```
cd <lab-directory>
make
nano sgemm.sh  # add sgemm commands per below
    ~/<lab-directory>/sgemm # Uses the default matrix sizes
    ~/<lab-directory>/sgemm <m># Uses square m x m matrices
    ~/<lab-directory>/sgemm <m> <k> <n># Uses (m x k) and
                                  # (k x n) input matrices
qsub -q tardis sgemm.sh
```

3 Turn in

Upload to the course Canvas site:

1. a report that includes :
   (a) the output
   (b) analysis of the performance: Try the code for several sizes, square and non-square matrices, and matrices that fit and don't fit (neatly) in the blocks How does the time change? Does each part change the same?
   (c) answer section where you answer the following:
      i. How many times is each element of each input matrix loaded during the execution of the kernel?
      ii. What is the memory-access to floating-point computation ratio in each thread? Consider multiplication and addition as separate operations, and ignore the global memory store at the end. Only count global memory loads towards your off-chip bandwidth.
2. main.cu
3. kernel.cu

The cuda code will be graded for completeness, correctness, handling of boundary, and style (5pts). The report will be graded on readability, clarity, analysis, and solution to the questions (5pts).

4 Going Further

We will be looking at some areas soon, so think about how you could group the sections to minimize memory loads.