EE/CSCI 451
Spring 2016
Programming Homework 6

Assigned: April 2, 2016
Due: April 10, 2016, before 11:59 pm, submit via blackboard
Total Points: 30

1 Examples

"vector_add.cu" implements the vector addition using 64K threads. There are two approaches to run it.

- Approach 1

  1. login hpc-login3.usc.edu
  2. source /usr/usc/cuda/5.5/setup.sh
  3. Go to your working directory which has 'vector_add.cu'.
  4. nvcc -o go vector_add.cu
  5. Modify the queue.pbs using your own information (working directory, email, etc.)
  6. qsub queue.pbs (if you see 'qsub:script is written in DOS test format', try:
     dos2unix queue.pbs
     then
     qsub queue.pbs)
  7. You can check your job progress using 'qstat -u your_usr_name'.
  8. After your job is completed, check 'cudajob.output' for output and 'cudajob.error' for any possible error.

  Approach 2

  1. login hpc-login3.usc.edu
  2. Reserve a computing node which has gpu, 'qsub -d. -*l* nodes=1:ppn=8:gpu,walltime=01:00:00'
  3. source /usr/usc/cuda/5.5/setup.sh
  4. Go to your working directory which has 'vector_add.cu'.
  5. nvcc -o go vector_add.cu
  6. ./go

2 Matrix Multiplication [30 points]

In the lecture and discussion, we discussed two approaches to compute matrix multiplication ($C = A \times B$) using CUDA: (1) unoptimized implementation using global memory only and (2) block matrix multiplication using shared memory.

In this assignment, your task is implementing 1024×1024 matrix multiplication using these two approaches.

- Approach 1 (unoptimized implementation using global memory only) [10 points]:
  - Name this program as 'p1.cu'
  - The value of each element of A is 1
  - The value of each element of B is 2
  - Thread block configuration: 16×16
  - Grid configuration: 64×64
  - After computation, print the value of $C[451][451]$
- Approach 2 (block matrix multiplication using shared memory) [15 points]:
  - Name this program as 'p2.cu'
  - The value of each element of A is 1
  - The value of each element of B is 2
  - Thread block configuration: 32×32
  - Grid configuration: 32×32
  - More details of this algorithm can be found in the paper 'Matrix Multiplication with CUDA' under the 'Readings' category of blackboard.
  - After computation, print the value of $C[451][451]$
- Report [5 points]: measure the execution time of the kernel of Approach 1 and Approach 2, respectively. Briefly discuss your observations.

3 Submission

You may discuss. However, the programs have to be written individually. You need submit your CUDA programs, 'p1.cu','p2.cu' and your report via blackboard.