CSCE 435 Fall 2015

HW 6: GPU Programming

Due: 11:59pm Monday, November 23, 2015

Given a set of *n* points in a plane, you need to determine the distance between the closest pair of points. Distance between points $p_i=(x_i, y_i)$ and $p_j=(x_j, y_j)$ is computed as

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

You are provided with a program `nbody.cu` that has the following capability:

- The code generates coordinates for n points on the host;
- The coordinates are copied to the device memory;
- An incomplete kernel function is provided that is intended to compute pairwise distances between all pairs of points, determine the minimum distance, and save the value in a device variable;
- The device variable is copied to the host;
- The value is compared with the minimum distance calculated on the host;
- The code reports the time spent in the kernel function, data transfer between host and device, and host computation.

You need to modify the kernel function to compute the minimum distance. You are allowed to make other changes to the code, which facilitate parallelization on the GPU.

1. (70 points) You need to develop CUDA-based parallel code to compute the distance between the closest pair of points on a GPU. 50 points will be awarded if the code compiles and executes the following commands successfully.
   ```
   ./nbody.exe 16
   ./nbody.exe 1024
   ./nbody.exe 2028
   ```
   10 points are reserved for a brief write-up describing the changes you made to the code. Additional 10 points are reserved for performance of the code: speed improvement obtained by the code over the host code.
2. (20 points) Execute the code for $n=2^k$ for k = 4,…,10. Plot GPU and CPU execution time versus k on the same plot to demonstrate how execution time varies with the problem size on these platforms. Use logarithmic scale for the x-axis. Next, plot the GPU and CPU execution time for $n=2^k$ for k = 11,…,16. For what value of n does the GPU code become faster that the CPU code?
3. (10 points) Plot the data transfer time from host to device and from device to host on the same plot for $n= 2^k$ for k = 4,…,16.

Submission: You need to upload the following to ecampus:

1. Problem 1: submit the file `nbody.cu`.
2. Problem 1, 2 & 3: Submit a single PDF or MSWord document with your response.

Helpful Information:

1. Source file(s) are available on ecampus.
2. Information on compiling and running CUDA programs on a GPU are available at http://sc.tamu.edu/wiki/index.php/Ada:Compile:CUDA

3. To develop code interactively, log on to one of the five GPU-equipped login nodes on ada; these nodes are named adaX.tamu.edu, where X is to be replaced by the numbers 1, 2, 3, 4, or 5.
4. Load the modules for compiler and CUDA prior to compiling your program. Use:
   ```
   module load intel
   module load CUDA
   ```
5. Compile C programs using `nvcc`. Specify intel C compiler as recommended by using `–ccbin=icc`. For example, to compile `code.c` to create the executable `code.exe`, use
   ```
   nvcc -arch=compute_35 -code=sm_35 –ccbin=icc –o nbody.exe nbody.cu
   ```
6. The run time of a code should be measured when it is executed in dedicated mode. A description of the batch system along with sample batch files is provided at http://sc.tamu.edu/wiki/index.php/Ada:Batch. To execute the code on a GPU-equipped node, you must use the submission options for GPUs.