2020.1 [Multicore Computing] Project 4: OpenMP/CUDA/Thrust Practice

(Deadline : June 20th)

Summision method : eClass

- create a directory 'proj4' and create subdirectory 'problem1' and 'problem2' in the directory 'proj4'
- In 'problem1', insert
  (i) source code files (openmp_ray.c, cuda_ray.cu) and executable files, (ii) report pdf file
- In 'problem2', insert (i) source code file (thrust_ex.cu) and executable file, (ii) report pdf file
- compress the directory 'proj4' into proj4.zip and submit it to eclass.
- Important Notice: CUDA codes and THRUST codes are compilable/executable only on a computer equipped with NVIDIA graphics card. If your computer does not support CUDA and THRUST, and if you cannot find a safe way to compile and test your CUDA/THRUST code, you still need to submit source codes, but it is OK that you do not include the test results of your CUDA and THRUST codes (cuda_ray.cu and thrust_ex.cu) on the report. Instead, you may state on your report that you could not test your CUDA/THRUST codes because your computer does not support CUDA/THRUST. (Normally, you can use our computer lab room on 4th or 5th floor of the building 208 to test your CUDA/THRUST code. However, I understand that you may not feel safe to use our lab room on campus because of Corona virus outbreak.)

Problem 1. Implementing two versions of Ray-Tracing (openmp_ray.c, and cuda_ray.cu) that utilizes OpenMP and CUDA library, respectively.

(i) Write OpenMP and CUDA programs.

- Look at the code [raytracing.cpp] for ray-tracing of random spheres, which is available on our class webpage, and modify the code for your purpose.
- You may assume the simplest form of Ray tracing that renders a scene with only spheres.
- Program input :
  (i) openmp_ray.c: [number of threads] [output filename]
  (ii) cuda_ray.cu: [output filename]
- Program output :
  (i) print ray-tracing processing time of your program using OpenMP or CUDA.
  (ii) generate image file (format: .ppm or .bmp, image size: 2048X2048) that shows rendering result

Execution example of openmp_ray.c) > openmp_ray.exe 8 result.ppm
OpenMP (8 threads) ray tracing: 0.41 sec
[result.ppm] was generated.

Execution example of cuda_ray.cu) > cuda_ray.exe result.ppm
CUDA ray tracing: 0.15 sec
[result.ppm] was generated.

(ii) write a report (pdf) that includes

- (i) execution environment (OS type, CPU type, grahics card/GPU type, memory size) (ii) how to compile, (iii) how to execute
- entire source code (openmp_ray.c and cuda_ray.cu) with appropriate comments
- program output results (i.e. screen capture) and ray-tracing result pictures
- experimental results : measuring the performance (execution time) of your OpenMP/CUDA implementation and the given single threaded CPU implementation. Show the performance results with respect to various number of threads. Include screen captures of output results.

Problem 2. Use thrust that is an open-source library

- Thrust is an open-source C++ template library for developing high-performance parallel applications, which is based on CUDA technology and brings a familiar abstraction layer to GPU.
- For information of thrust library, visit https://docs.nvidia.com/cuda/thrust/ and https://www.sie.es/wp-content/uploads/2015/12/Intro-to-Thrust-Parallel-Algorithms-Library.pdf

(i) Understand thrust library and write your C++ source code thrust_ex.cu that does one of following problems. Choose (a) or (b). You do not need to do both.
(a) counting the number of prime numbers between 1 and N, where N is program input argument using CUDA thrust library. Choose N as sufficiently large integer number. (e.g. N=200000)

(b) approximate computation of the integral $\int_0^1 \frac{4.0}{(1+x^2)} dx$ by computing a sum of rectangles $\sum_{i=0}^{N} F(x_i)\Delta x$ where each rectangle has width Δx and height $F(x_i)$ at the middle of interval i and $F(x) = \frac{4.0}{(1+x^2)}$. Choose N as sufficiently large integer number. (e.g. N=200000)

(ii) Write a report that includes (i) your source code, (ii) execution time table of sequential program using only CPU and your program using thrust, and (iii) your explanation on the results.