Colby Crutcher, Riley Rudolfo

GPU Computing

Lab 4 Matrix Multiply

1. Read the provided code in src folder, specifically read the main function in the source file matrix_multiplication.cu. Answer the questions below,

a. What tasks the program performs?

- Compares the speed of matrix multiplication on a CPU versus a GPU. It reads a square matrix from a file specified in the command line arguments and allocates memory on both the host (CPU) and the device (GPU).

b. What are the dependency C files and header files? How to call functions that is defined in another source file?

- mul.h: Contains the prototype for the CPU-based multiplication function mul().
- arrayUtils.h: Contains prototypes for readNewArray, writeArray, and printArray.
- timing.h contains elapsedTime, timeCost, and currentTime
- matrix_multiplication.cu: The main entry point containing the CUDA kernel and the logic for the benchmarking loop.
- timing.c: Contains the implementation for measuring time using gettimeofday.
- mul.c: Implements the sequential triple-nested loop for matrix multiplication.
- arrayUtils.c: Implements the logic for reading matrices from files using fscanf and writing them using fprintf

C files:

2. Based upon the lecture notes, please write the simple kernel function to perform matrix multiplication on GPU, on top of the source file matrix_multiplication.cu.
3. Explore the Makefile, how shall we jointly compile .cu and .c files in a single project?

- Compiling a project with mixed .cu and .c files involves a two-stage process, compiling source files into individual object files and then linking them together. We need to use the -c flag to generate object files. The Makefile creates object files for mul.o, timing.o, and arrayUtils.o individually. We then use nvcc, a compiler to link all the files together.

4. Check the APIs Docs and find out what cudaEventRecord() and cudaEventSynchronize() do?

- Both of these are used to monitor device progress and synchronize the host (CPU) with specific points in a GPU execution stream.
- The cudaEventRecord function “records” an event into a specified CUDA stream.
- The cudaEventSychronize is a blocking call used to wait for an event to complete.

5. In the main function, what is the equation that the program uses to compute the throughput (in unit of GLOPS)? Please interpret the equation.
6. Run your program after you finish your kernel on the dataset 1024.mat and 2048.mat, how much speedups do you obtain compared with CPU time cost? What are the GPU throughput and CPU throughput you observed in these cases?

Output:

Matrix size: 1024 x 1024 Tile size: 16 x 16 Throughput of simple kernel: 5451.66 GFLOPS Throughput of cpu code: 0.99 GFLOPS Performance improvement: simple_throughput / cpu_throughput = 5531.50 x Speedup in terms of time cost: 5531.50 x

-The GPU implementation achieved a speedup of 5531.50 x over the CPU. Throughput for the GPU kernel was 5451.66 GFLOPS. The observed throughput for the sequential CPU code was 0.99 GFLOPS.

7. Can you find some specification data about the peak performance (GFLOPS) for the GPU that we are using in the Lab (NVIDIA RTX 3070) ? Is the GPU device throughput that you observed in step 6 close to their Peak performance? Guess the reason why they are close or why they are far away?