Korea Advanced Institute of Science and Technology

School of Electrical Engineering

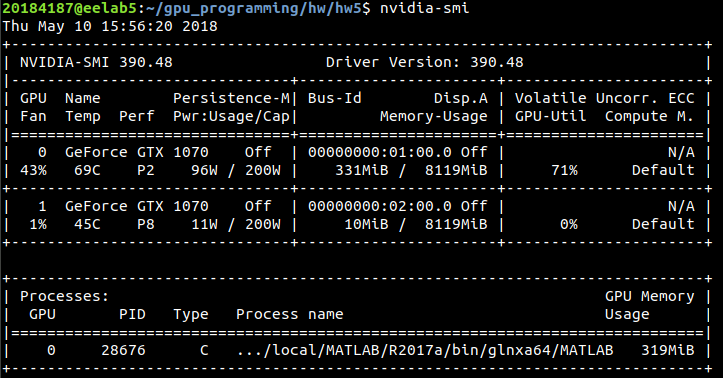
EE817 GPU Programming and Its Applications Spring 2018

Student’s Name: Dinh Vu

Student’s ID: 20184187

**Homework 5**

The computer, used in my homework 4, contains NVIDIA GeForce GT 1070 based on Pascal GP104 architecture.



**Figure 1.1.** Graphic card information

1. **Without Stream**

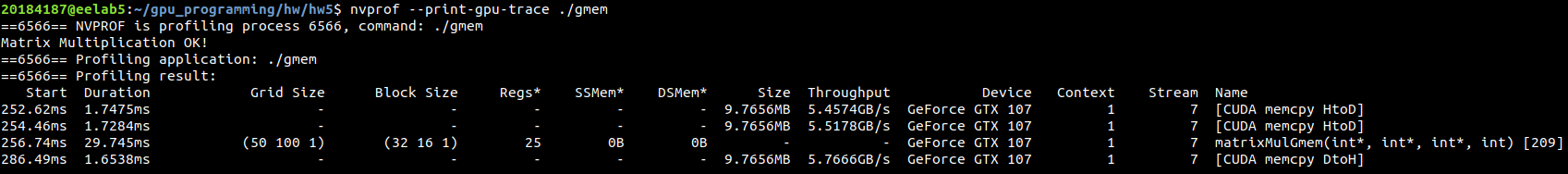
The source code for matrix multiplication using only global memory without stream is matrixMulGmem.cu file.

Each thread computes value of each element in matrix C. Each row of matrix A and the corresponding column of matrix B is read from global memory. Then each element in matrix C with thread index is calculated parallel. This programming strategy is presented in Figure 1.2.

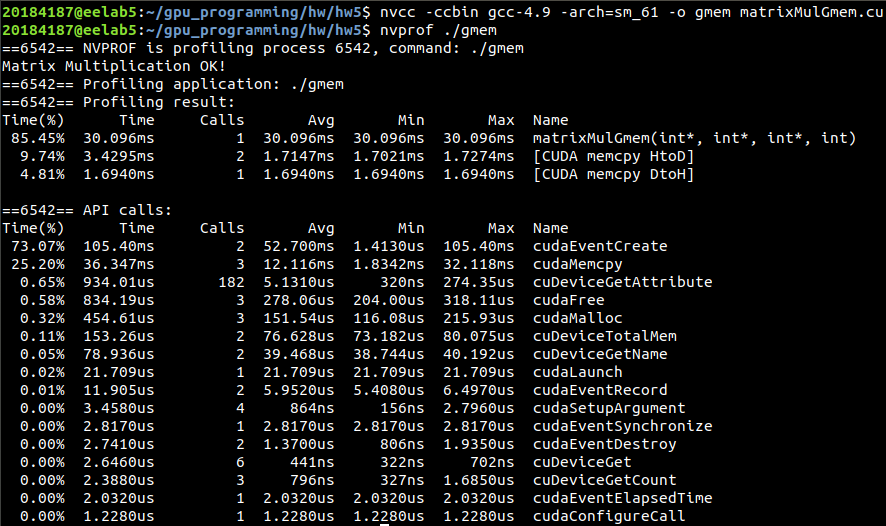
Figure 1.3 shows the procedure of the program while Figure 1.4 displays the summary of the execution time.



**Figure 1.2.** Matrix multiplication only using global memory



**Figure 1.3.** The procedure of the program using only global memory



**Figure 1.4.** The execution time of the matrix multiplication using only global memory

1. **Depth-first**

The matrixMulDepth.cu is source code for matrix multiplication overlapping kernel execution with depth-first data transfer. Because I remote the computer in Haedong Lounge, **nvvp** cannot display the profile of my program. Therefore, instead using **nvvp**, the option --print-gpu-trace of **nvprof** command is used to display the sequence of streams.

Matrix B is still load entirely from global memory. Because the required number of streams is 8, matrix A is divided to 8 slides in order to split into each stream. Each stream will read each slide A[i] from global memory then calculate the output slide C[i], where i = 0, 1, 2,…, 8. Hence, each slide of the output matrix C also is computed following each stream as Figure 2.1.

All streams using the same kernel matrixMulDepth. The programming strategy of kernel matrixMulDepth is presented in Figure 2.2.

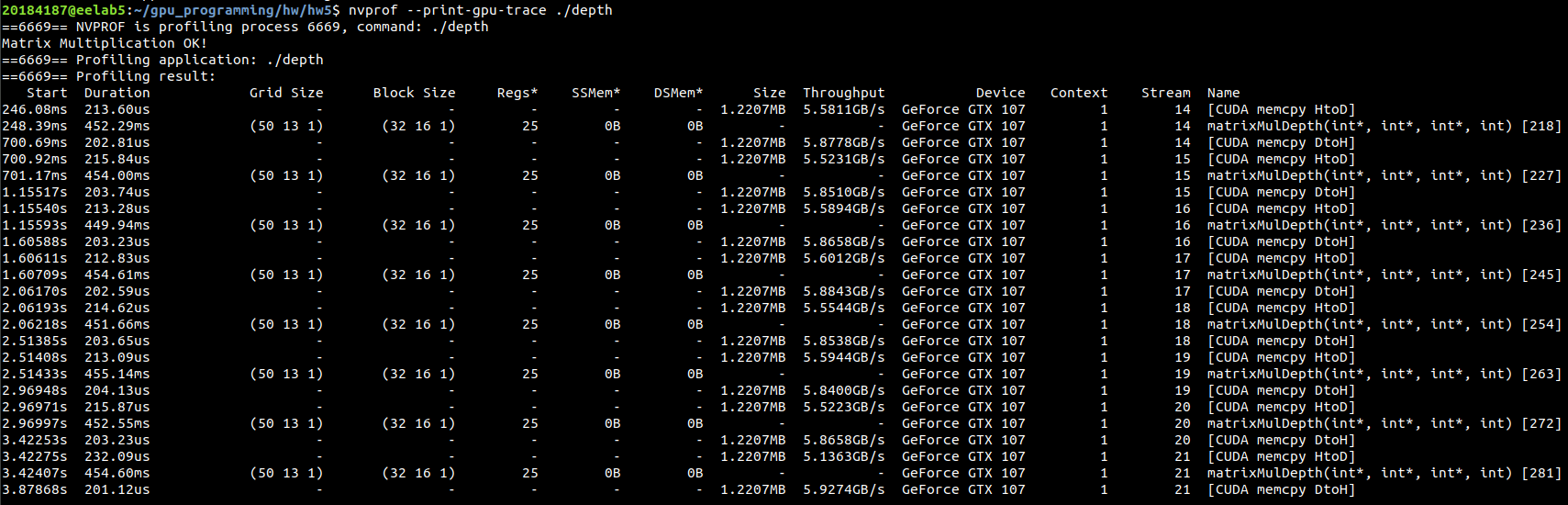
The order operation of the streams and the summary of execution time of each function are shown in Figure 2.3 and Figure 2.4, respectively.



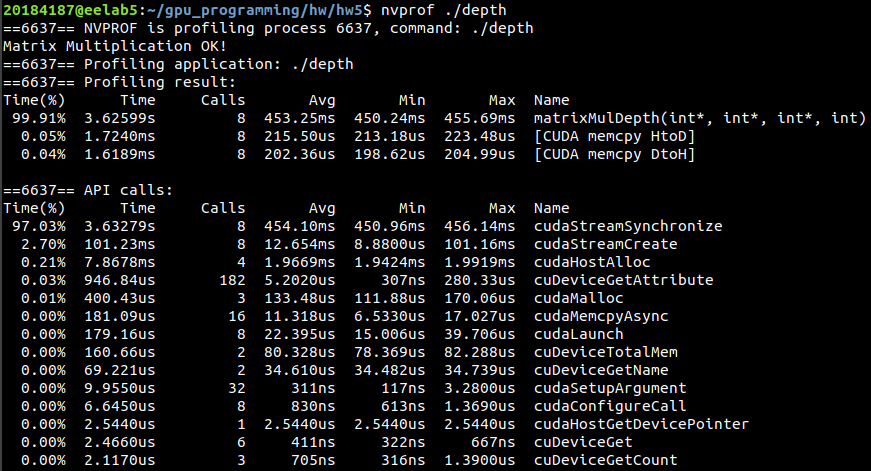
**Figure 2.1.** The programming strategy for matrix multiplication using streams



**Figure 2.2.** The operation of kernel matrixMulDepth



**Figure 2.3.** The procedure of streams in matrix multiplication with depth-approach

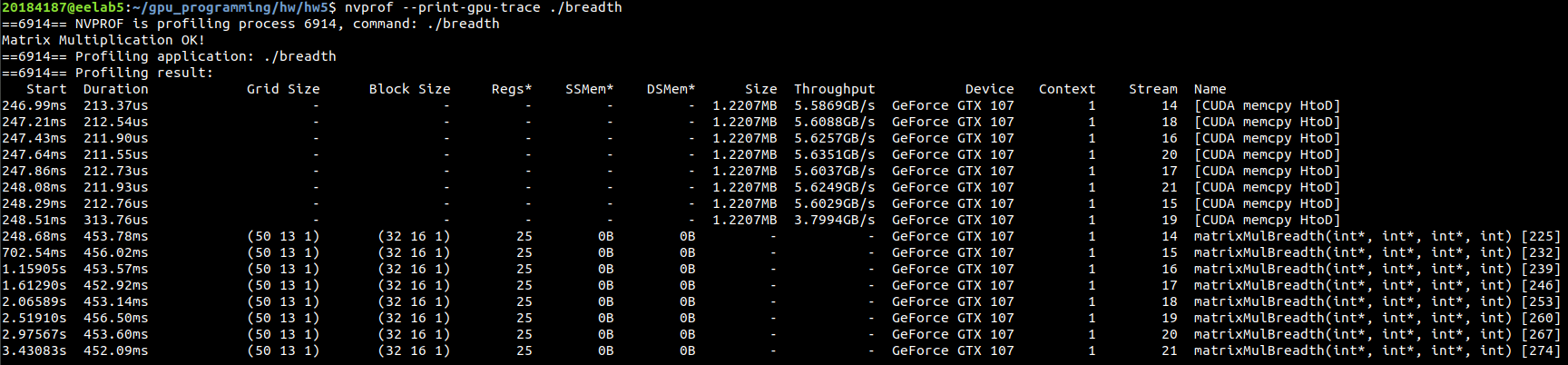


**Figure 2.4.** Summary of the activities on GPU of matrix multiplication

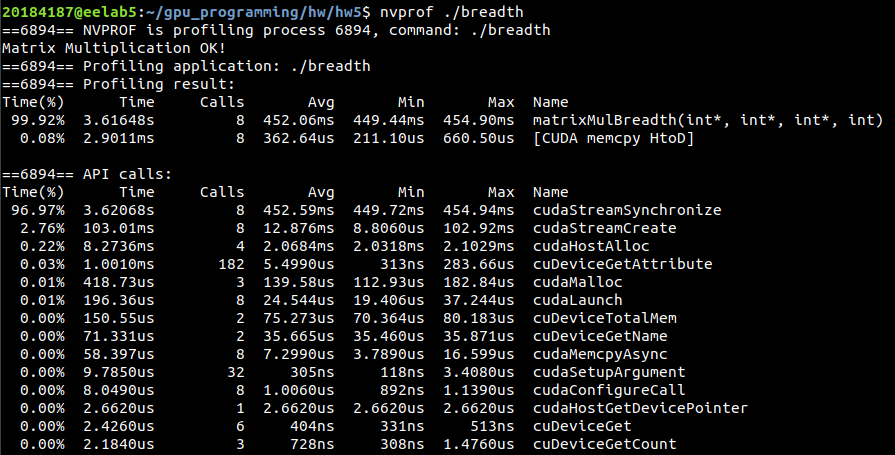
with depth-approach

1. **Breadth-first**

The source code for matrix multiplication using breadth-first data transfer is matrixMulBreadth.cu. The programming strategy is similar to the depth-first. The only difference is that each activity copy data from host to device, kernel and copy result from device to host have each for-loop.



**Figure 3.1.** The procedure of streams in matrix multiplication with bread-approach



**Figure 3.2.** Summary of execution time for each activity on GPU processing matrix multiplication with breadth-approach