ECE 285, Spring 2018
GPU Programming

Department of Electrical and Computer Engineering
University of California, San Diego

LAB 3: Tiled Matrix Multiplication

In this lab, we'll multiply **P** and **Q** matrices which was derived in the lab 2. In practice, you can only multiply the ith row of **P** with the jth column of **Q** to predict a unknown value x_{ij} of **R** for a given query (x_{ij} is a movie rate of the ith user to the jth movie). In this lab, you need to do full matrix multiplication to derive a dense matrix **R** in Figure 1. The matrices **P**, **Q** and **R** are not fit into your 6GB GPU memory. They should be partitioned into multiple tiles as depicted in Figure 2. One key technique for fast matrix multiplication utilizes a double-buffered shared memory to hide transfer latency from the global memory as described in the reference 1. Here we'll also apply the technique. Thus, you should use the double buffered shared memory scheme to read and write data to the global memory. All the data format should be half-precision floating point except computations in the GPU kernel as in Figure 3. As we did in the lab 2, a RMS value of the test result should be around 0.92, which also matches with one of the reference 2.

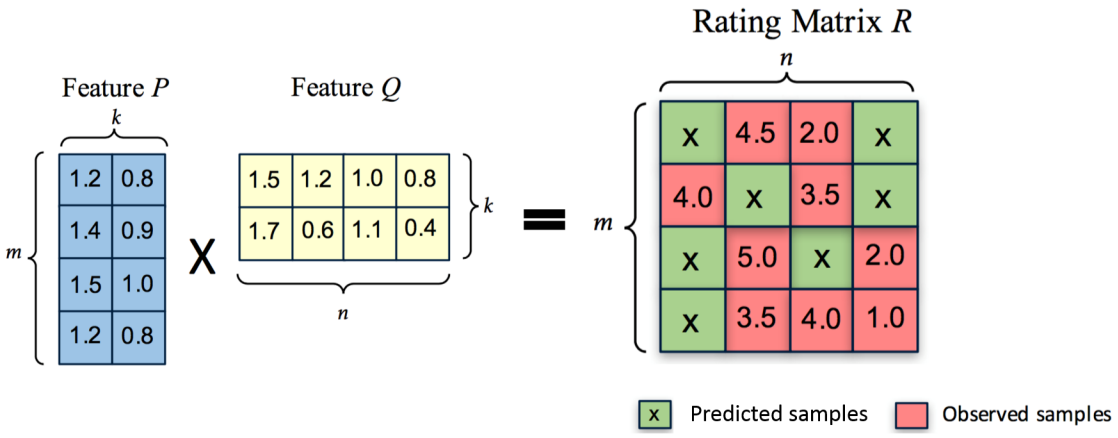

Figure 1: Dense matrix multiplication.

You should try to maximize block partitions in order to minimize the transfer overhead between CPU and GPU. Thus, you need to check an available GPU memory size using the following CUDA API:

```
#define MByte (1024*1024)
    size_t remainMem, totalMem;
    cudaMemGetInfo(&remainMem, &totalMem);
    printf("GPU total memoroy: %d MB, remaining memory:%d MB\n",
    (totalMem / MByte), (remainMem / MByte));
```

Programming language: CUDA

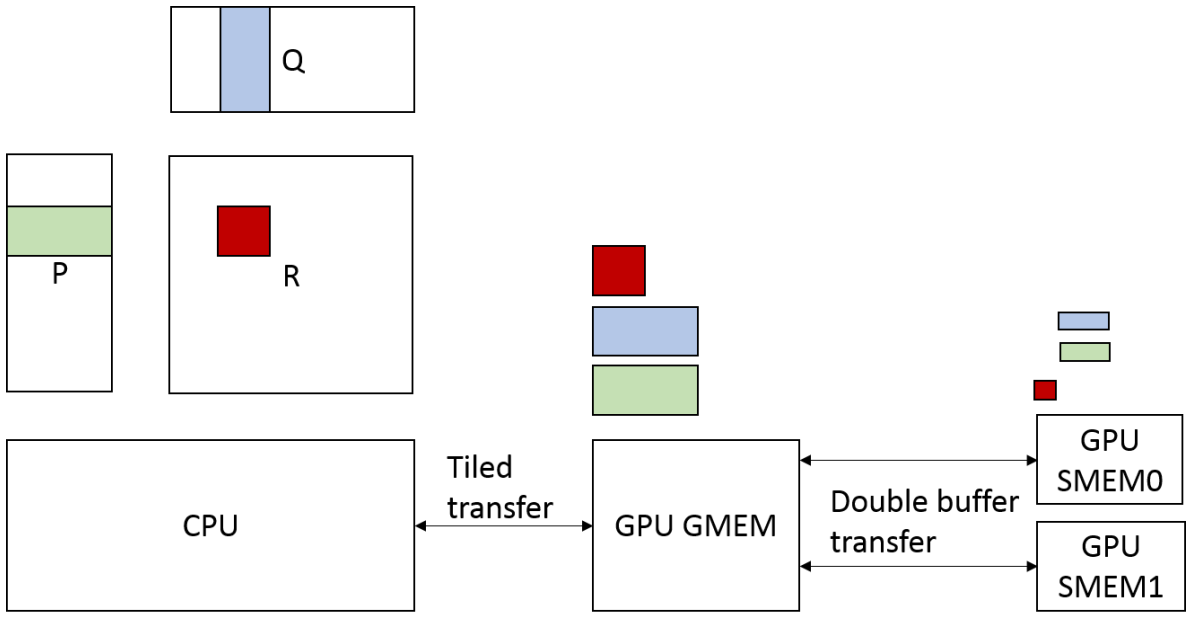

Figure 2: Data processing diagram.

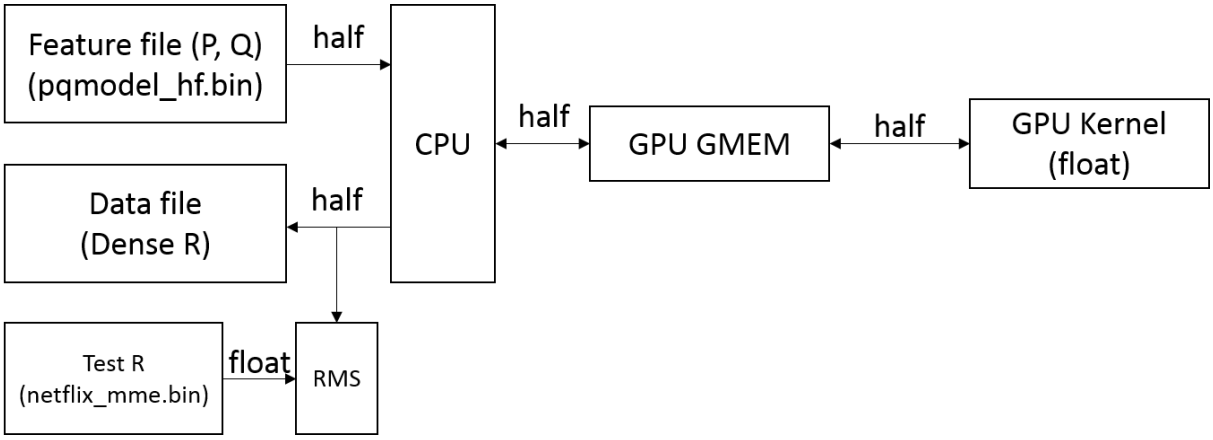

Figure 3: Data precision diagram.

References

1. Andrew Kerr, Duane Merrill, Julien Demouth and John Tran, CUTLASS: Fast Linear Algebra in CUDA C++, Nvidia, 2017
`https://devblogs.nvidia.com/cutlass-linear-algebra-cuda/`

2. Xiaolong Xie, Wei Tan, Liana L. Fong and Yun Liang, CuMF_SGD: Fast and Scalable Matrix Factorization, CoRR, 2016
`https://arxiv.org/abs/1610.05838`