CS433-Parallel and Distributed Computing

Assignment#2 on CUDA Programming

1. There are many problems, which can be effectively solved with CUDA, are suitable for a novice as exercise. Here are two of them.
   a) Matrix multiplication, also known as matrix product and the multiplication of two matrices, produces a single matrix.
   PROBLEM:
   Please write CUDA kernel functions to effectively compute matrix multiplication of two matrices which have **odd** column number and row number.
   b) As you may know, Kernel Method (KM) is used to avoid computation and complexity in Support Vector Machine (SVM). RBF kernel is one of the famous kernel functions. Its definition is
   $$K(x_i, x_j) = e^{-\frac{\|x_i - x_j\|^2}{2\sigma^2}}$$
   Where xi and xj are n-dimensional vectors with each representing a piece of n-dimensional feature. It's clear that calculating RBF kernel function using single CPU thread may be slow and it can be effectively parallelized using CUDA.
   PROBLEM:
   Please write CUDA kernel functions to effectively compute the RBF kernel of two matrices. You may need to write wrapper functions to accomplish the task. (Hint: if you have any problems, please first refer to matrix multiplication algorithms)
2. To achieve the maximum performance, there are many tricks based on the processor and memory architecture. The memory hierarchy of CUDA architecture brings us global memory (including constant memory, texture memory and surface memory), shared memory and registers. You must realize that shared memory should be used, if possible, to avoid high latency accessing global memory. But shared memory is again far slower than registers. How can we achieve the extreme performance? Please read reference [1] and optimize your program to get more speedup.

NOTICE:

1. You'd better use Ubuntu and you have to write a **makefile** to compile your code. If you don't have NVIDIA GPU environment, please contact TA at Shine1999@sjtu.edu.cn.
2. What you write should be standalone functions to perform the computation, i.e., everybody can reuse your function to do the similar job with variable configurations.
3. Send your final version to TA at Shine1999@sjtu.edu.cn. You should archive your source code and makefile with StudentID_Name_HW2.tar.gz(or any archive file types). Do not include binary file.
4. Should you have any questions, please feel free to contact TA at Shine1999@sjtu.edu.cn.

Reference

[1] http://dmacssite.github.io/materials/volkov10-GTC.pdf