Bilgin AKSOY
2252286
MMI 713 Applied Parallel Programming On GPU

REPORT(LAB EXECISES)

1. Problem Definition : During the exercise, three parallelization algorithm (vector product, matrix summation and matrix-vector multiplication) has been implemented. All three problem can easily be parallelize on GPU in order gain performance augmentation.
   a) First problem was dot product of two same length vectors.
   b) Second was element wise summation of two matrices.
   c) Last was a matrix and a vector multiplication.

2. Algorithm Description :
   a) First algorithm is a straightforward one that summing each element in a separate thread.
   b) Second algorithm is also a straightforward like the first one but in this algorithm uses a 2D kernel function.
   c) Third algorithm is multiplying each element in a separate thread and uses a 2D kernel function.

3. Different size matrices (Benchmarking) : All three algorithm work any size of matrices and vectors without any error. See calculation times in Table1,2,3.

4. Pros-Cons of Solution : It is obvious that GPU codes need bigger arrays, vectors, matrices or any kind of container for beat CPU only code.

5. Discussion :
   a) Vector Product: The algorithm which operates on CPU is better than the which operates on GPU for smaller vectors but GPU calculations are better for bigger vectors.
   b) Matrix Summation: The algorithm which operates on CPU is better than the which operates on GPU for smaller matrices but GPU calculations are better for bigger matrices.
   c) Matrix-Vector Multiplying: The algorithm which operates on CPU is better than the which operates on GPU for smaller matrices but GPU calculations are better for bigger matrices. (I really couldn't find any reason how CPU only code can have better performance for **980X720-720**)
6. Environment :
   a) AMD Ryzen 7 1700 CPU
   b) 32 Gb RAM
   c) GTX 1070 GPU 8 Gb RAM
   d) CUDA 8 - NVCC 8.0.61
   e) Ubuntu 14.04
   f) GCC 4.8.4
   g) I used CMake(2.8.12.2) as build generator and GNU Make (3.81) as builder.

Vector Product		
Vector Size	CPU Time	GPU Time
5	0.001888 ms	0.026624 ms
10	0.001856 ms	0.030176 ms
100	0.001024 ms	0.028672 ms
10000	0.008032 ms	0.026080 ms
100000	0.066752 ms	0.076896 ms
250000	0.201728 ms	0.193504 ms
400000	0.313408 ms	0.280192 ms
550000	**0.435424 ms**	**0.342464 ms**
650000	**0.526336 ms**	**0.397536 ms**

Table-1: Vector Product Calculation Times

Matrix Summation		
Matrix Sizes	CPU Time	GPU Time
5X5	0.001024 ms	0.027072 ms
30X28	0.002848 ms	0.034336 ms
300X280	0.055296 ms	0.067552 ms
350X300	0.074752 ms	0.082464 ms
400X300	0.090112 ms	0.095360 ms
400X350	0.097216 ms	0.106880 ms
450X350	0.107680 ms	0.117152 ms
550X400	**0.192512 ms**	**0.191872 ms**
600X800	**0.381952 ms**	**0.325760 ms**

Table-2: Matrix Summation Calculation Times

Matrix-Vector Multiplication		
Matrix-Vector Sizes	CPU Time	GPU Time
5X3-3	0.001056 ms	0.030336 ms
50X30-30	0.002912 ms	0.022080 ms
500X300-300	0.083968 ms	0.118784 ms
800X600-600	0.118784 ms	0.306176 ms
900X650-650	0.356160 ms	0.370400 ms
920X700-700	**0.378880 ms**	**0.365856 ms**
980X720-720	**0.388256 ms**	**0.391232 ms**
980X750-750	**0.447520 ms**	**0.420384 ms**