CS610 : Assignment 5

Parth Sharma
20111039
prthshrma20@iitk.ac.in

NOTE : All codes are compiled and run on IIT Kanpur GPU server gpu1.cse.iitk.ac.in .
Compile : nvcc -g -G $-arch = sm\_52$ -std=c++11 20111039.cu -o 20111039
Run : ./20111039

Configuration of gpu1.cse.iitk.ac.in

- Processor : Intel(R) Core(TM) i7-6850K CPU @ 3.60GHz
- Number of cores : 6 per socket
- Thread(s) per core : 2
- 64-bit Operating System
- L1d cache : 32 KB
- L1i cache : 32 KB
- L2 cache : 256 KB
- L3 Cache : 15360 KB

Problem 1

Solution

Naive parallel CUDA implementation for SIZE1 = 8192 and SIZE2 = 8200.
For both size gpu2.cse.iitk.ac.in server was used.

SIZE	Implementation	Time in milliseconds	Speedup
8192	Serial version on CPU	29652.2	-
8192	Naive Parallel CUDA kernel	919.273	32.25
8200	Serial version on CPU	29372.9	-
8200	Naive Parallel CUDA kernel	956.62	30.70

Table 1: Performance of kernels on two different sizes

Observation

- As it can be observed that kernel mode is getting better performance than serial version.
- SIZE2 being 8200 is more than 8192. Hence, slight variation can be seen in their speedup.

Problem 3

Solution

Three different implementation of multiplying the transpose of a square matrix A with itself (i.e. $A^T A$) were performed. First serial implementation on CPU, parallel implementation on CUDA kernel and optimized parallel implementation (tiling) on CUDA kernel.

On CUDA kernel, threads per block is taken as (32,32), and in optimized version tiling of tile size 32 is performed.

Performance of different implementations are as follows:

Implementation	**Time in milliseconds**	**Speedup**
Serial version	243417	-
Parallel version on CUDA kernel	15986	15.22
Optimized parallel version on CUDA kernel	7026.79	34.64

Table 2: Performance of multiplying the transpose of a square matrix A with itself

OBSERVATION :

- As it can be seen from data that parallel version on CUDA kernel are performing well.
- Doing tiling on parallel version improves the performance.

Problem 4

Solution

Two versions of matrix multiplication with CUDA are denoted as :

- Naive kernel : Kernel 1
- Optimized kernel : Kernel 2

For size=4096 gpu2.cse.iitk.ac.in server was used.

Performance of different implementations on different SIZE of matrix and different size of tile are as follows:

Matrix SIZE	**Implementation**	**Time in milliseconds**	**Speedup**
1024	Serial version	6564.86	-
	Naive kernel	1072.7	6.11
	Optimized kernel with tile size 8	495.803	13.24
	Optimized kernel with tile size 16	385.513	17.02
	Optimized kernel with tile size 32	338.985	19.36
2048	Serial version	165441	-
	Naive kernel	8746.57	18.91
	Optimized kernel with tile size 8	3805.94	43.46
	Optimized kernel with tile size 16	3408.71	48.53
	Optimized kernel with tile size 32	3114.06	53.12
4096	Serial version	1.13392e+06	-
	Naive kernel	42274.4	26.82
	Optimized kernel with tile size 8	16472.7	68.83
	Optimized kernel with tile size 16	14332.7	79.11
	Optimized kernel with tile size 32	14036.1	80.78

Table 3: Performance of different versions of matrix multiplication