ME 766
Assignment 2
Roll no: 123079015

System Specification
The system used for this assignment is listed in 3 files
Cpuspecs.txt – It contains the CPU info of the System
memspecs.txt – It contains the Memory info of the System
CUDAdevices.txt – It contains information about the NVIDIA GPU used

CPU - Intel(R) Core(TM) i7-2600 CPU @ 3.40GHz
8 cores with HyperThreading resulting in 16 virtual cores

Memory - 16 GB RAM

CUDA devices: Main GPU – NVIDIA Tesla K20c
Secondary GPU – NVIDIA GeForce GTX 560 Ti
GPU on test machine – NVIDIA GeForce GT 520 (1GB)

The main sytem was a RedHat system with GCC 4.4.7

Results

Time with data transfer = Time measured for kernel execution along with copying data from host to device and from device to host

Time without data transfer = Time measured for only kernel execution (excluding data transfer of any sort between host and device)

	Time (in ms)						
	N=100	N=500	N=1000	N=2500	N=5000	N=10000	N=9984
With data transfer	0.180704	6.368160	35.793728	487.703613	3765.069580	29625.306641	20439.548828
Without data transfer	0.069856	3.820192	29.454369	454.980682	3638.965088	29123.472656	19940.408203

Comments

The code was originally tested on an NVIDIA GT 520 with 1 GB memory. It was noted that erroneous zero matrix was produced as a result when the dimensions of the matrix was pushed over 2000x2000.
By writing down the matrices to a file it was noted that even a 5000x5000 matrix produced a single matrix with size 237 MB (The 10000x10000 case produces a single matrix file of 957MB and the Product matrix of size 1.4 – 1.6 GB). Thus, the erroneous result was generated when matrix size was increased as all the matrices could not be accomodated into the GPU memory at the same time.

One more thing that was noted was that zero matrices were produced even when the matrix dimensions were well within the range. This, was solved by understanding the results of devQuery. When the number of threads within a given block exceeded 1024, the errorneous results were observed. Thus, the number of threads per block was limited to 20x20 (The last case was set up especially to push the threads/block to 32x32). The limits on the number of blocks that can be called is very high, thus, a huge number of blocks was instantiated every time.