Heterogeneous Computing - HOMEWORK 0 - ECGR-6090

Observed and Documented by: Sumukh Raghuram Bhat – 801131997

The testing platform information is as follows:

CPU: Dual core Intel Core i5-4200U
Software environment: Linux (64-bit), Ubuntu 18.04
Development Language: Cuda C and C.
CUDA Version: 9.2 using MAMBA cluster.

1. A) The "Vector Add" program was written and compiled in C for the CPU and in Cuda C for the GPU.

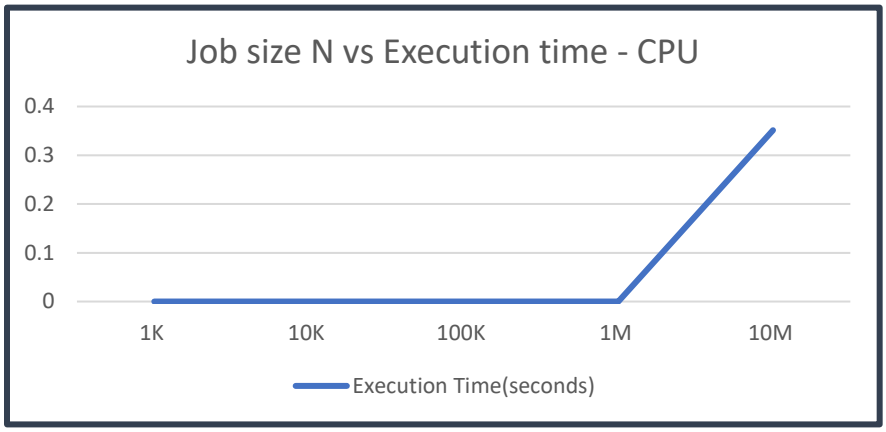
The execution time of GPU is calculated in milliseconds and the execution time for CPU is calculated in seconds.

We increase the value of N (job size) and get the execution time for all the values of N. The tables below describe the execution times obtained for both CPU and GPU

The table with values containing the execution time for variable N values when run in CPU is as follows:

N (Job Size)	Execution Time(seconds)
1K	0.000036
10K	0.001071
100K	0.007887
1M	0.039018
10M	0.351384

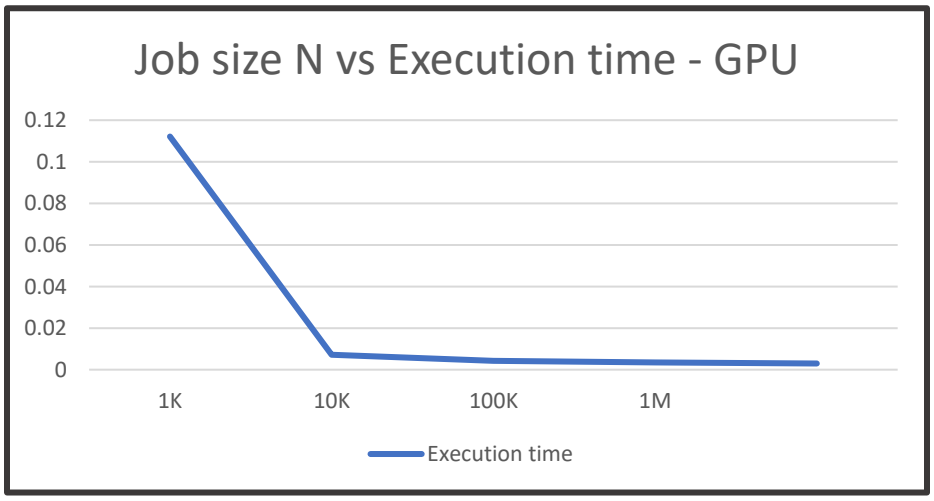
The graph showing the execution time for various job sizes for CPU is:



The table with values containing the execution time for variable N values when run in GPU is as follows:

N (Job Size)	Execution Time(milliseconds)
1K	0.112132
10K	0.007163
100K	0.004257
1M	0.003539
10M	0.002962

The graph showing the execution time for various job sizes for GPU is:



In a GPU, as we increase the job size, there are threads that can perform the operations given to the GPU.

The execution time for CPU will increase as we increase the job size or the amount of data that has to be computed whereas the execution time of GPU reduces.

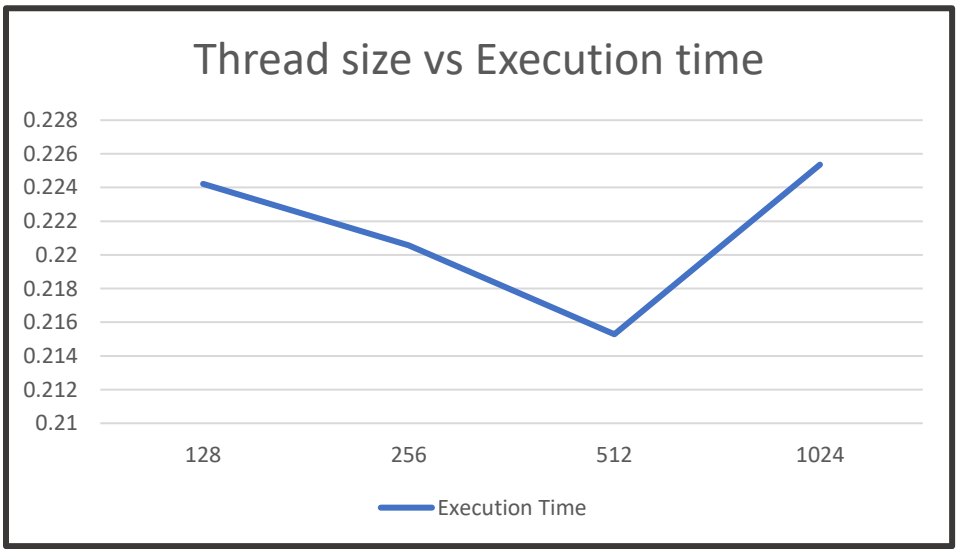
Increasing the job size reduces the execution time because a great amount of parallelism takes place in a GPU and hence all the data gets executed in a parallel fashion and data computation time is reduced.

1. **B)** For N= 1M, thread block size is varied. We then find the execution time for various thread block sizes.

The Table is as follows:

Thread Size	Execution Time(milliseconds)
128	0.224227
256	0.220564
512	0.215286
1024	0.225361

The graph for the execution time with the thread sizes is:



The execution time decreases as we increase the thread size. When the thread size increases beyond 512TB, the execution time again increases. This is because small thread sizes limit the performance of the GPU due to occupancy.

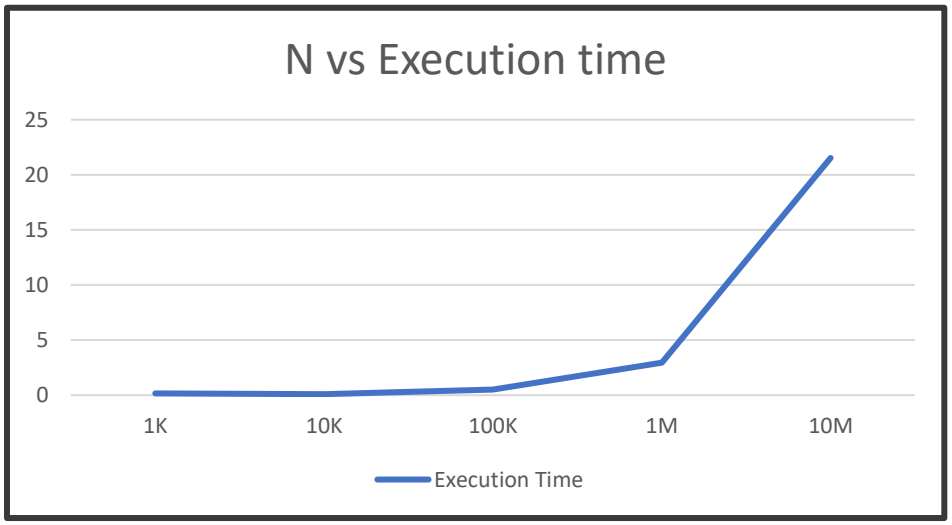
Using very large block sizes will also limit the performance of the GPU due to resource limits such as the number of registers being used per thread or if shared memory is being used or not.

Choosing enough threads to saturate the machine and give the machine the best chance to hide latency is the best way to efficiently use the memory subsystems.

1. **C)** Here, we find that the execution time drastically increases as shown in the table. We consider the additional overhead of moving data from the host to the device.

N (Job Size)	Execution Time(milliseconds)
1K	0.151040
10K	0.092416
100K	0.500000
1M	2.952256
10M	21.528671

The graph of the execution time taken for various job sizes is as follows:



Large vectors need to be copied from CPU memory to GPU memory through relatively slow PCIe bus and it overshadows the higher computational capability of GPU. Hence the cost to move data to and from the GPU will require at least one read of the inputs from the CPU and one write of the outputs to the CPU.

Most of time on GPU are used to copy data. In simple words, as we increase the data that has to be sent to the GPU, the execution time increases.

2. **A)**1D Stencil is executed on the GPU with constant radius (2) and thread block size (128) and with varying input array size.

The tables for execution times with & without memory copy and graph for the execution time with & without memory transfers:

Table for execution time without memory copy:

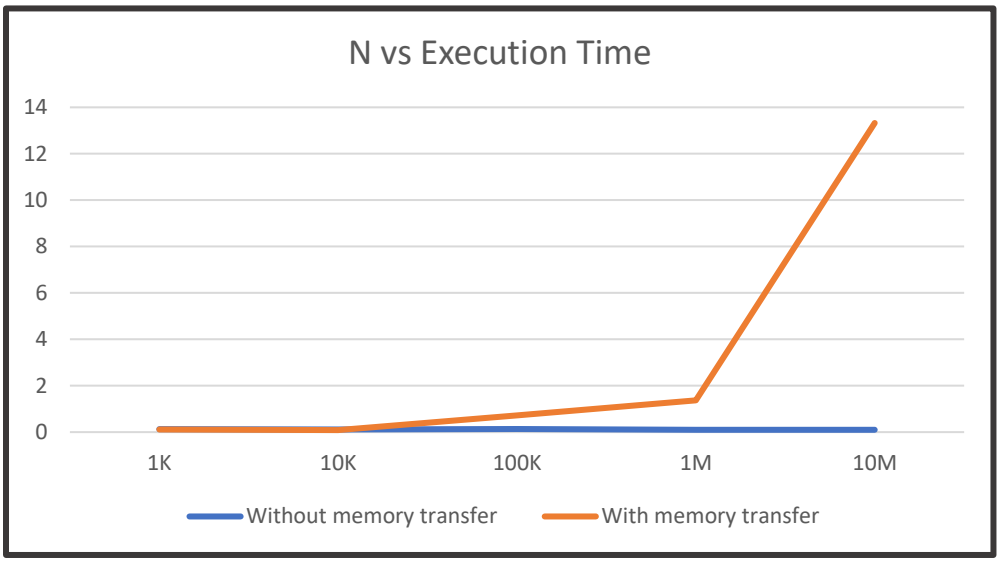
N (job size)	Execution Time (milliseconds)
1K	0.122529
10K	0.113361
100K	0.134076
1M	0.101300
10M	0.100203

Table for execution time with memory copy:

N (job size)	ExecutionTime (milliseconds)
1K	0.108910
10K	0.087421
100K	0.723724
1M	1.372123
10M	13.32531

With memory transfer, execution time is higher in the case of a GPU than CPU because the data has to be copied from the host to the device. The data is transferred at a slower rate.

When we ignore the memory transfer, the execution time is much faster for a GPU as threads execute the data in a parallel fashion.



2.B) The input N is 10K and the radius takes varying values 2, 4, 8 and 16.

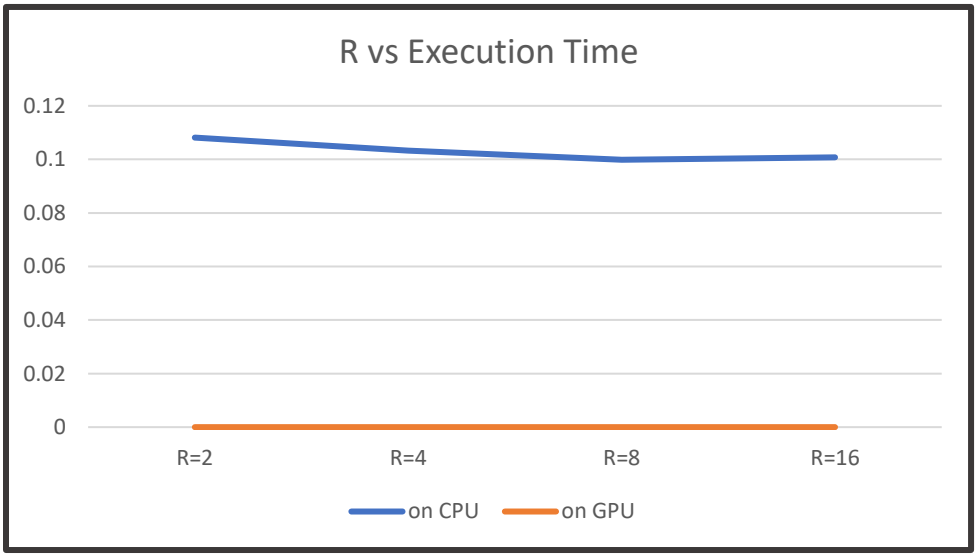
The tables for various execution times and the graph is as below:

Table for execution time when N=10K and TB=128 for a GPU:

Radius	Execution Time (milliseconds)
2	0.108142
4	0.103229
8	0.099864
16	0.100732

Table for execution time when N=10K and TB=128 for a CPU:

Radius	Execution Time (milliseconds)
2	0
4	0
8	0
16	0



From the graph, The execution time in the case of a CPU is just 0. we see the execution in a CPU is much faster than that of a GPU. CPUs use spatial locality which improves the performance of the CPU.

The concept of spatial locality is not used by GPUs and hence the execution time is longer.

Here, the execution time decreases till the radius is 8 and then the execution time increases and the radius increases.

2. **C)** The tables of the execution times with and without shared memory is as follows:

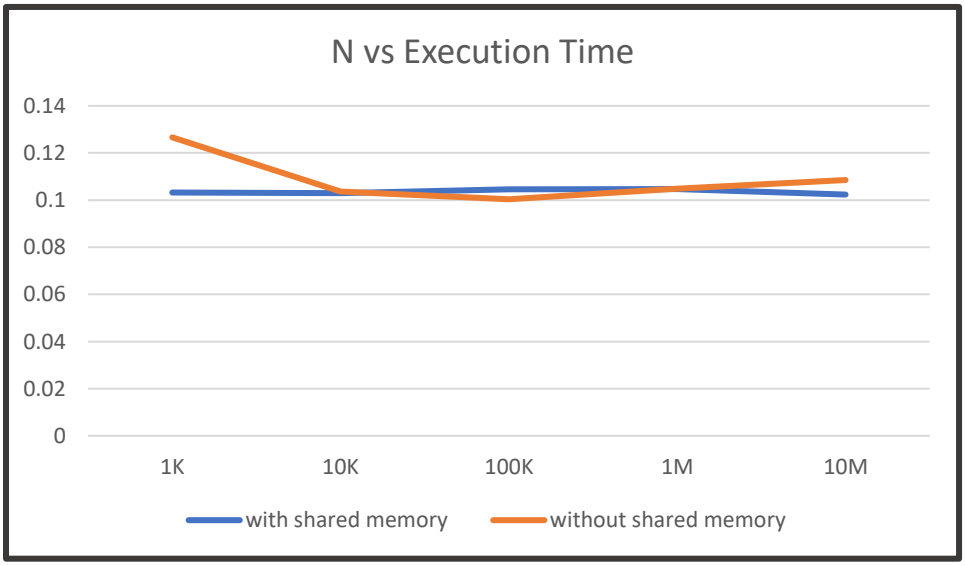
With shared memory:

N (job size)	Execution Time (milliseconds)
1K	0.126624
10K	0.113664
100K	0.113660
1M	0.112545
10M	0.113488

Without shared memory:

Radius	Execution Time (milliseconds)
2	0.103286
4	0.102941
8	0.104620
16	0.112690

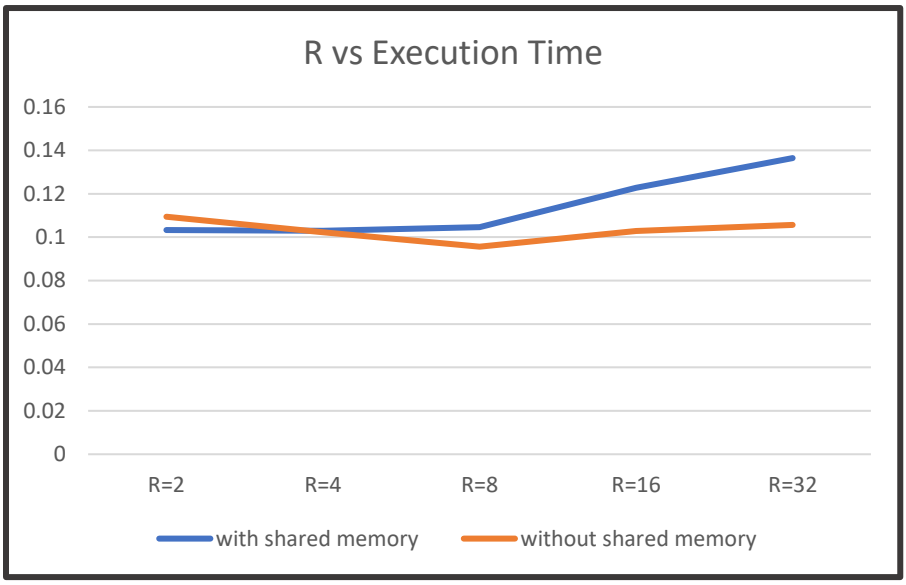
N vs Execution Time for with and without Shared Memory:



As we increase the input, we can notice that the execution time without shared memory is more than the execution time with shared memory.

We see the execution time is more in the case of a GPU with shared memory.

A variable amount of time is taken by the shared memory for the synchronization of all the threads. This happens before the actual execution takes place.



When we ignore the memory transfer, the execution time in a GPU is very fast. This is because the threads execute the data in a parallel fashion.

From the above graph, when the transfer of memory is considered, the execution time in case of a GPU is higher than a CPU.

This is because the data transfer bus is slow and has a slow rate of transfer. This causes the data from the host to the device to be transported at a slow rate.