Major Project

Nimoshika Jayaraman

UIN: 631000852

1. Compiling and executing the code in parallel computer:

* **Load the modules for compiler and CUDA prior to compiling your program:**

module load intel CUDA/11.3.1

module load intel/2018b

module load CUDA

* **Compiling the C program:**

nvcc GPR.cu -o GPR.exe

./GPR.exe 30 1 0.5 2

* **For batch file:**

sbatch job.grace\_job

1. **Strategy to parallelize in single processor and design choices:**

I have used dynamic thread allocation based on the number of threads provided as a runtime

argument to parallelize the **Cholesky algorithm**.

* There are 3 for loops which are almost independent in their execution. From the

research papers referred for Mini project, the outermost loop could not be parallelized

and can only be executed in series.

* The outermost loop runs on the host and the two inner loops runs on the GPU device.
* I have parallelized the inner two loops, across multiple blocks in x-dimension. The blocks

are configured dynamically from the number of threads. Each element calculation is

disjoint and can be scheduled across different blocks. But one of the major drawbacks of scheduling across different blocks is the overhead associated with scheduling the blocks over different SM.

* One of the major drawbacks of parallelizing using the GPU, is the global memory

bandwidth, since we use non uniform memory accessing and the cache across threads

are very low, every memory access needs to be acquired from the Global Memory, as a

result there is a maximum limit to which we can parallelize this algorithm

* We could use the shared memory per block, but it would make it very complex, and it

would require extensive synchronization and data portioning across several blocks.

* Compared to OpenMP , CUDA was difficult and the speedup achieved was not better

than openMP. One of the main bottlenecks were the Global memory Access(GDDRAM),

and the non- coalescing memory access, which further reduces the speedup.

**3) FLOP RATE CALCULATION:**

m= 30

Floating Point Operations = 1700997900

Time Taken in GPU = 1300ms

FLOPS = approx.(1.3GFLOPS).

Actual GPU FLOPS = 1.7TFLOPS

**Speedup:**

No of CUDA Cores = 2496

M=30:

Time CPU = 6.6s

Time GPU = 1.6s Threads = 900. Speedup = 4.2

Efficiency = 3.1% (Used Threads as Cores).

M=40:

Time GPU = 8.2s Threads = 1600

Time CPU = 45.1s

Speedup = 5.152

Efficiency = 3.22% (Used threads as cores)