PROJECT REPORT

IMPLEMENTATION INFORMATION

1. The modules to be loaded are:
   module load gcc/4.9.0
   module load cmake/3.9.1
   module load openmpi
   module load cuda
2. Run times considered for plotting are done for cases of excluding and including the writing to output file.

NODE CONFIGURATIONS

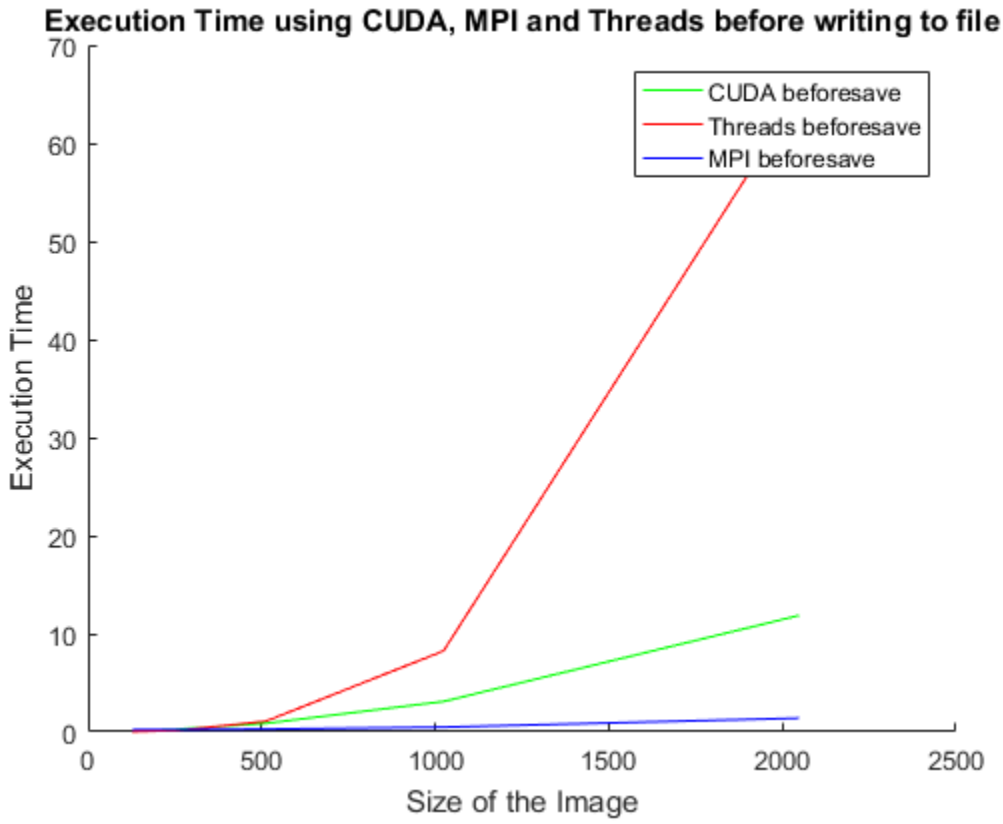
Node = 1
PPN = 8
Nvidia Tesla p100 GPU

RUN TIME GRAPHS

1 - Run times excluding Writing to File

N		128	256	512	1024	2048
Threads	**DFT**	0.02316	0.1454	1.0766	8.2502	64.9472
MPI	**FFT**	0.2170	0.2298	0.28008	0.48155	1.3980
CUDA	**DFT**	0.1834	0.2698	0.8644	3.0976	11.863

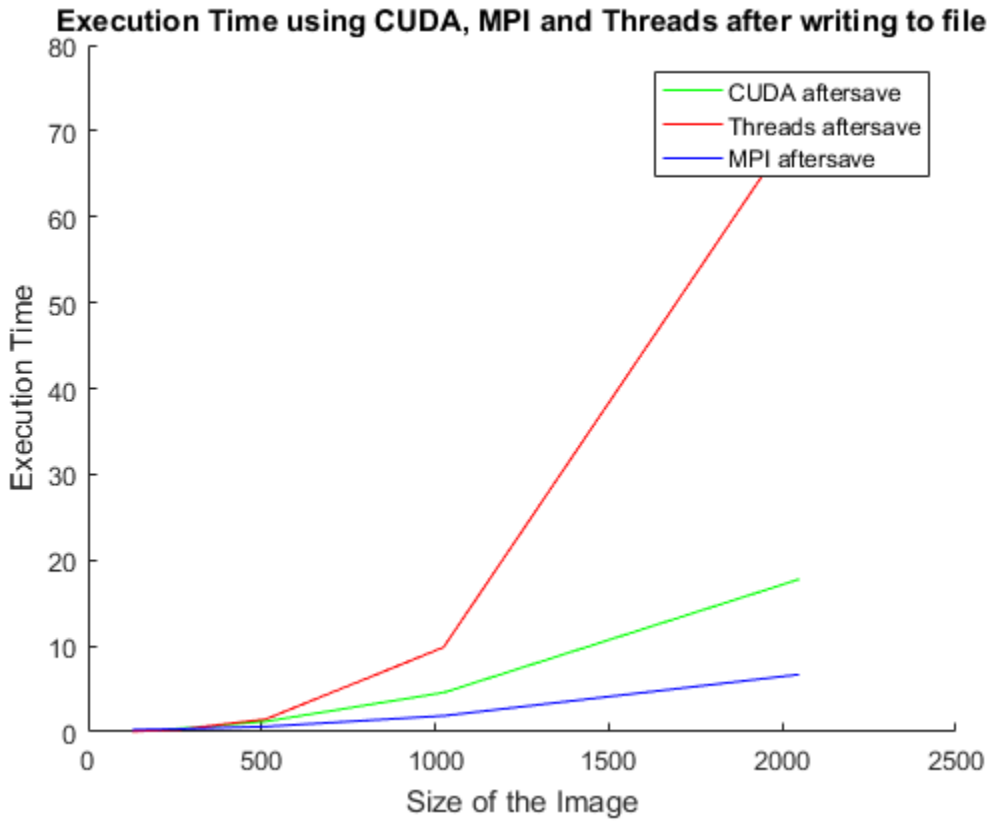
(Time units in seconds)



2 - Run times including Writing to File

N		128	256	512	1024	2048
Threads	**DFT**	0.0489	0.25334	1.4768	9.8465	71.1266
MPI	**FFT**	0.2549	0.3328	0.6172	1.8701	6.68
CUDA	**DFT**	0.20498	0.36308	1.2124	4.5724	17.7696

(Time units in seconds)



INSIGHTS

- The general observation is that MPI implementation is the fastest, followed closely by CUDA implementation, with C++ thread implementation in last place.
- MPI implementation is the fastest as it uses FFT algorithm which reduces complexity from $O(N^2)$ to $O(NlogN)$.
- For N = 128, we observe that C++ thread implementation has faster time performance than both CUDA and MPI implementations. This can be attributed to the fact that sending and receiving data between processors in MPI, and transfer of control from device to host in CUDA, take up comparable amount of time as to that of the DFT computation.
- The FFT implementation in MPI has been done using the iterative method rather than recursive method. No significant time improvements are seen in using recursive method. Furthermore, recursive method uses stack space, whereas iterative method has $O(1)$ and uses no stack space.