Name: Patel Aayush
Enroll no. 183109002
Email id: 183109002@iitb.ac.in

1 First Exercise: Matrix Multiplication and Upper Triangular Matrix formation using OpenMp

To perform numerical task code is written in C++ using OpenMp library, and compilation of Openmp is done using Intel Core i7-9750H Hexa Core on windows.

1.1 Performance study

To quantify the performance different number of sizes(N*N) starting from 80 to 3200. The sets of N is taken as {100,500,1000,2000,3000}.For each N the code was run at different threads aka. 1,2,4,6,8. The performance time is tabulated in is table 1.1 to 1.5. Also figure are attached so to quantify the behaviour of OpenMp.

Sr No.	Size(N)	multiplication (.millisec)	upper triangular(.millisec)
1	100	3	2
2	500	803	320
3	1000	8010	2330
4	2000	89491	19099
5	3000	299384	66284

Table 1.1 time study data for one thread

Sr No.	Size(N)	multiplication (.millisec)	upper triangular(.millisec)
1	100	2	1
4	500	375	148
3	1000	3987	1236
4	2000	43287	9823
5	3000	145454	34649

Table 1.2 time study data for two threads

Sr No.	Size(N)	multiplication (.millisec)	upper triangular(.millisec)
1	100	3	1
4	500	296	85
3	1000	2182	720
4	2000	21830	6419
5	3000	79775	21349

Table 1.3 time study data for four threads

Sr No.	Size(N)	multiplication (.millisec)	upper triangular(.millisec)
1	100	3	10^-1
4	500	190	74
3	1000	1511	588
4	2000	19359	5265
5	3000	62023	18201

Table 1.4 time study data for six threads

Sr No.	Size(N)	multiplication (.millisec)	upper triangular(.millisec)
1	100	4	$0.65*10^{-}1$
4	500	188	69
3	1000	1399	491
4	2000	16145	4403
5	3000	53659	15525

Table 1.5 time study data for eight threads

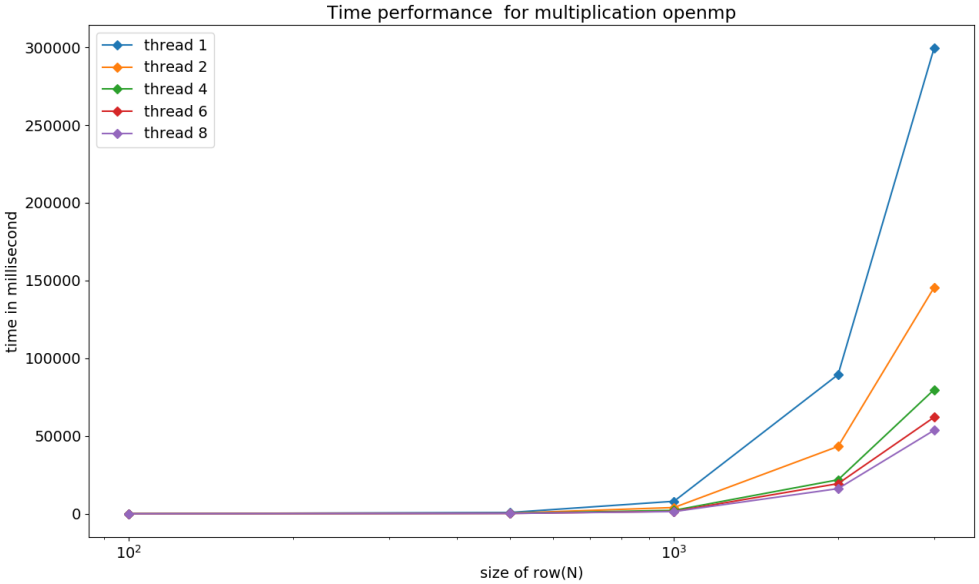

figure 1.1

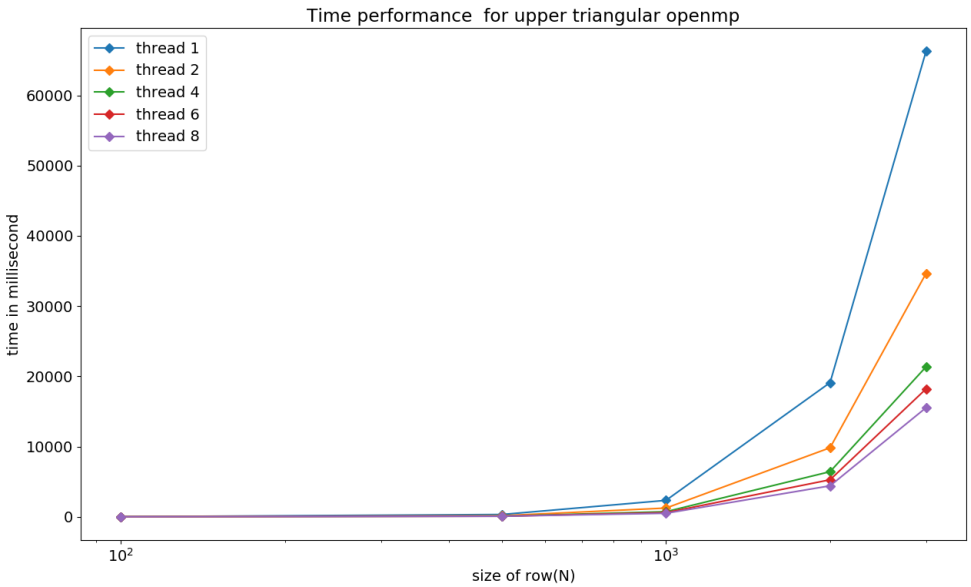

figure 1.2

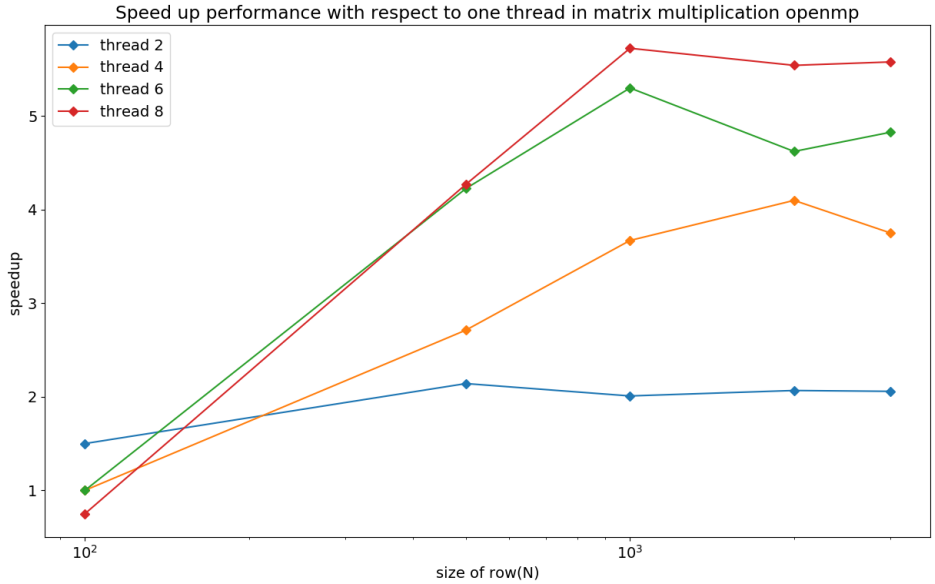

figure 1.3

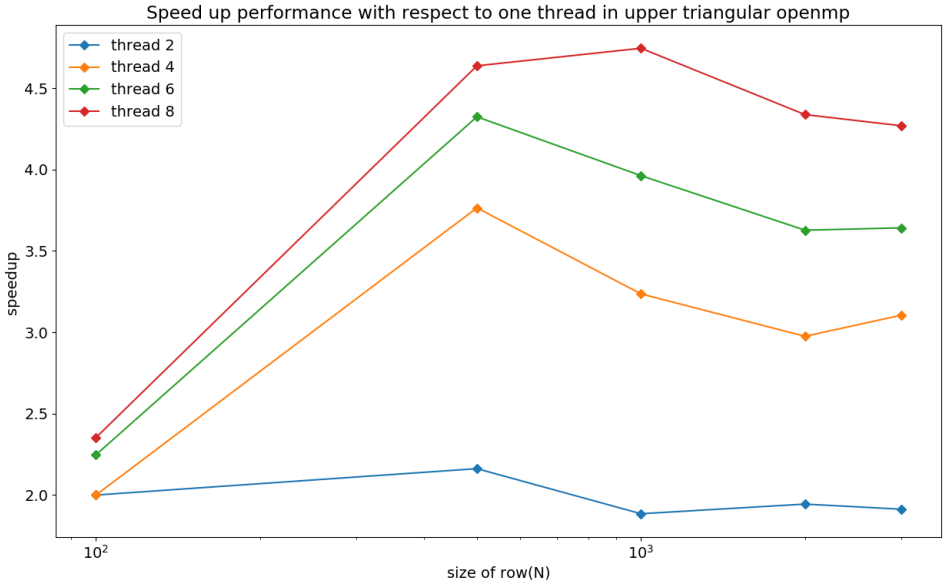

figure 1.4

1.2 Observation

The code is written on shared memory parallelisms. It uses the library to include functionality of thread parallelisation.The performance is quantified using 1,2,4,6 & 8 number of threads using different number of sizes.It is evident that increasing the number of threads the code is speeding up for higher number sizes and speed up is proportional to number of threads upto 4 but after that speed up is not proportional because of hyper threading.

2 Second Exercise: Matrix Multiplication and Upper Triangular Matrix formation using MPI

To perform numerical task code is written in C++ using OpenMp library and the code is ran on ”PARAMSANGANAK”, supercomputing facility of CDAC.

2.1 Convergence study

To quantify the performance different number of sizes(N*N) starting from 100 to 5000. The sets of N is taken as {100,1000,5000}.For each N the code was run at different cores aka. {2,5,10}. The performance time is tabulated in is table 2.1

Sr No.	Size(N)	Multiplication time(.sec)	Upper triangular time(.sec)	total time(.sec)
1	100	0.00243391	0.0025772	0.005011
2	1000	2.98265	1.90207	4.88472
3	5000	498.046	235.999	734.036

Table 2.1 time study for 2 MPI processes(max time out of 2 cores)

Sr No.	Size(N)	Multiplication time(.sec)	Upper triangular time(.sec)	total time(.sec)
1	100	0.00107115	0.00270804	0.00377
2	1000	1.57469	0.896963	1.6643863
3	5000	201.705	99.2697	300.9747

Table 2.2 time study for 5 MPI processes(max time out of 5 cores)

Sr No.	Size(N)	Multiplication time(.sec)	Upper triangular time(.sec)	total time(.sec)
1	100	0.0066769	0.0103018	0.0169787
2	1000	0.8724216	0.71931	1.5917316
3	5000	100.476	57.6462	158.122

Table 2.3 time study for 10 MPI processes(max time out of 10 cores)

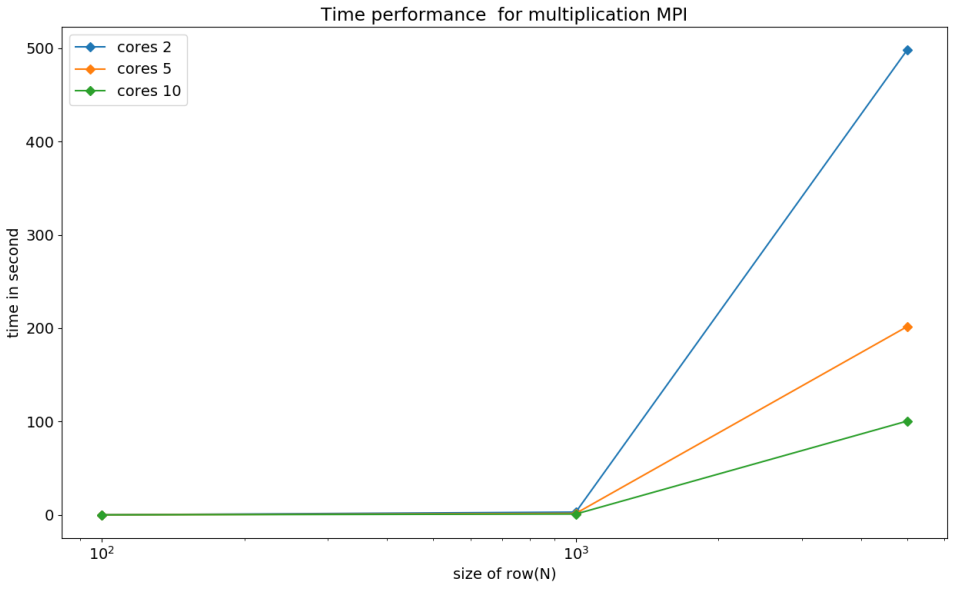

figure 2.1

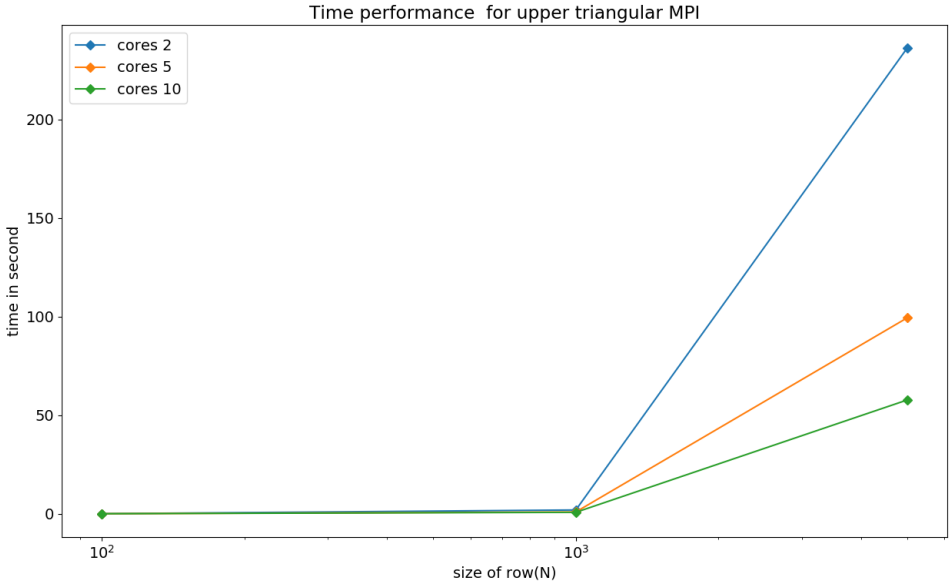

figure 2.2

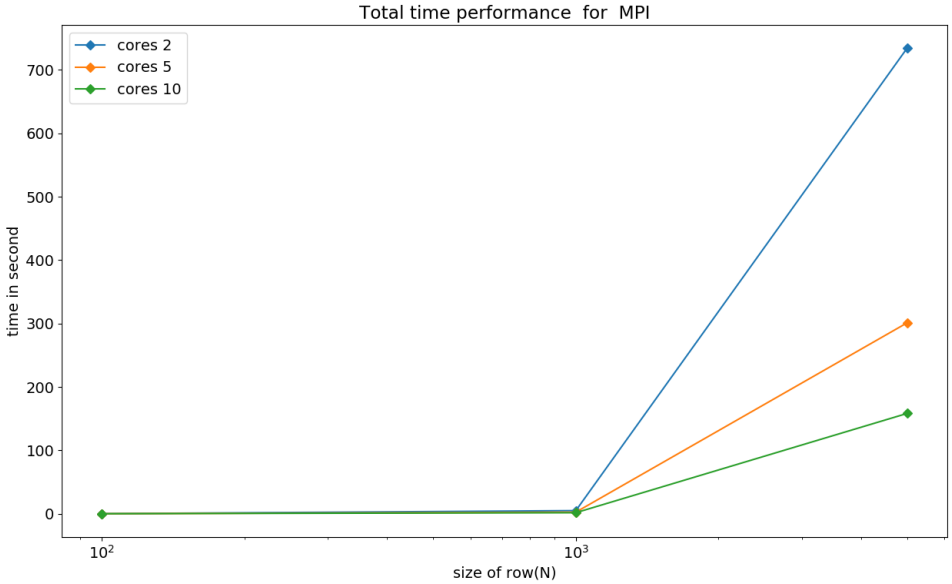

figure 2.3

2.2 Observation

The code is written on distributed memory parallelisms. It uses the mpi library to include functionality of distributed memory parallelisation.The performance is quantified using 2,5 & 10 number of cores using 100,1000,5000 grid points.It is evident from the table that incearsing the number of cores the time completion reduces. Below graph shows the time performance using different cores. Additionally data is provided in tabulated format.