High Performance Computing
Assignment 7

Jahnavi Suthar(201301414) Deeksha Koul(201301435)

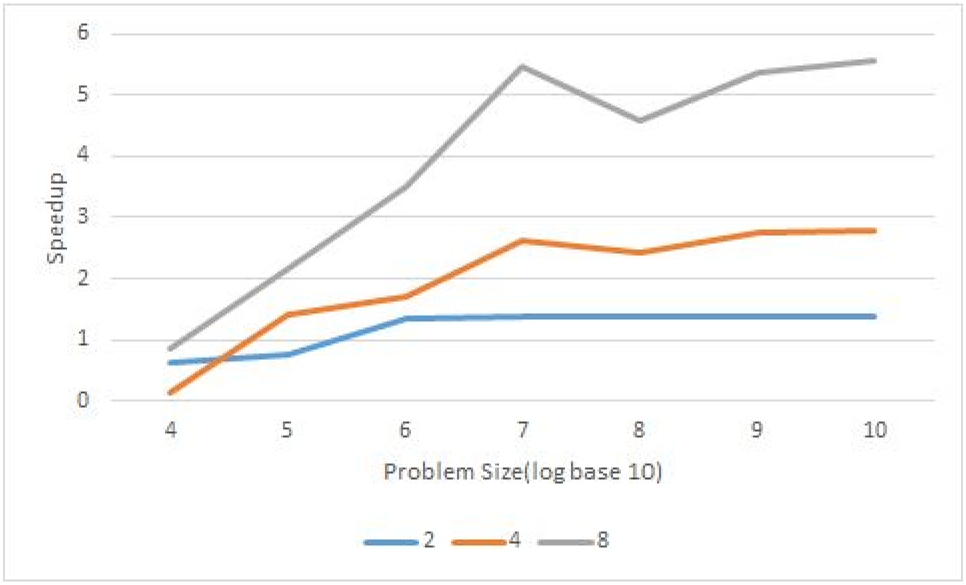
Hardware Details

CPU Model : INTEL(R) Core i5-4590
No. of cores: 16
Memory: 7.6 GiB
Compiler: gcc
Optimization flags if used: None
Precision: double

Calculation of PI using Series

$$\pi = \sum_{k=1}^{n} \frac{(-1)^k}{(2k-1)}$$

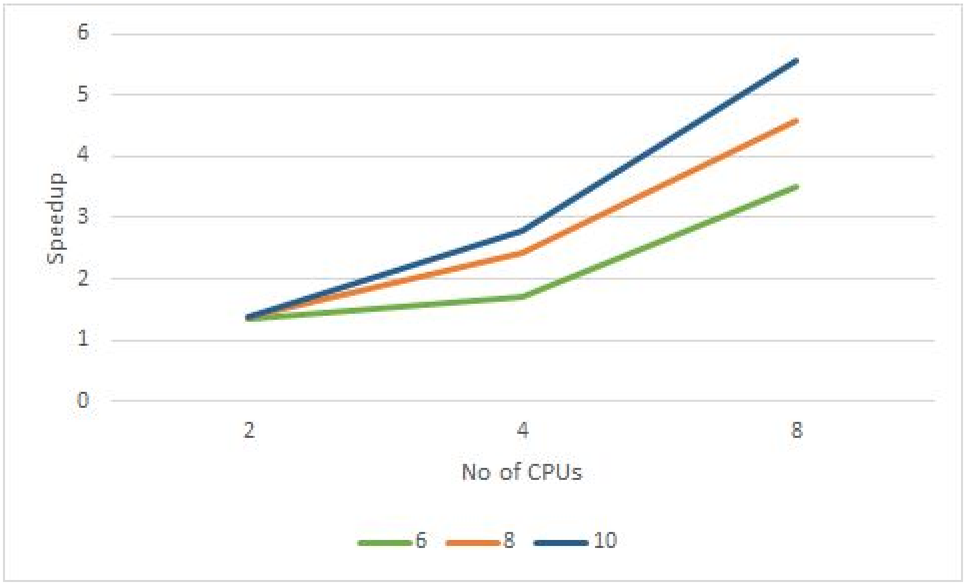
Complexity of the Problem (Serial):
The complexity of the problem is O (n), where n is number of terms which we take in the series for the calculation of pi.
Possible Speedup (Theoretical):
Speedup = 1 / (S + (P/M)) = M(approx.)(Code can be fully parallelized. P = 1)
where M is no. of cores, P is fraction of code that can be parallelized and S is serial fraction of code. P+S=1.
Optimization:
OpenMP: We divide the n(number of terms of series mentioned above) among the p (number of threads) and each thread will calculate its partial sum and then add it to the global sum.By using the reduction operator we have avoide any possible race condition that can happen at sum variable due to each thread accessing it.
MPI: We divide the n (number of terms of series mentioned above) among the "p" processors and each processor with its own local memory , will calculate its own partial sum and then broadcast it. The master processor(Rank=0) calculates global sum by using the MPI_Reduce call therefore we have avoided any possible race condition that can happen at sum variable as each processor calculates its own sum independent of each other.



Problem Size(log base 2) vs Speedup for 2, 4 and 8 CPUs in MPI

Observation:

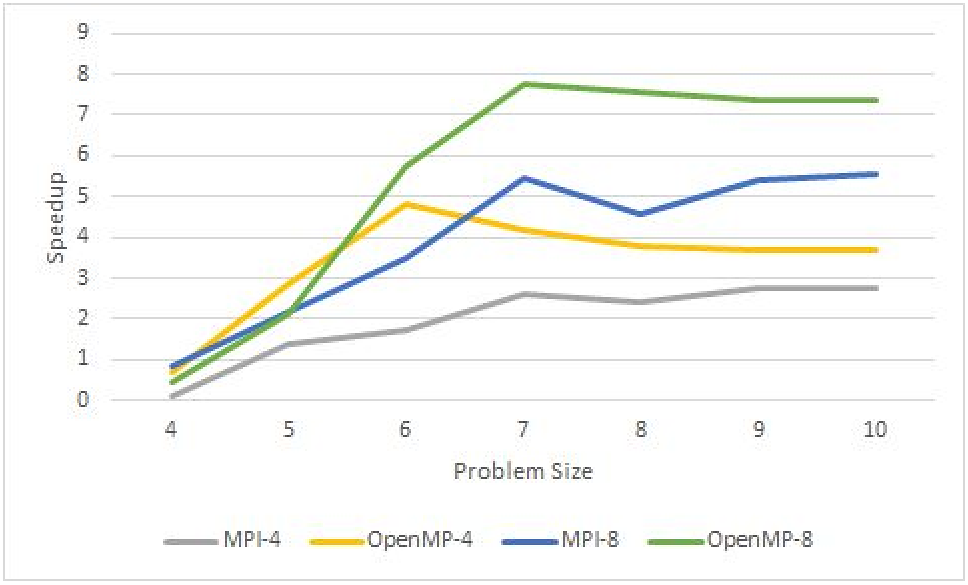
As the input increases in MPI-code the speedup tends to increase ,for a particular number of cores the speedup is proportional to input size but the speedup is not ideal and hence maximum attained speedup is for processors.The Openmp works better than MPI but difference is not large.



No of CPUs vs Speedup for Problem Sizes 10^6, 10^8 and 10^{10} in MPI

Observation:

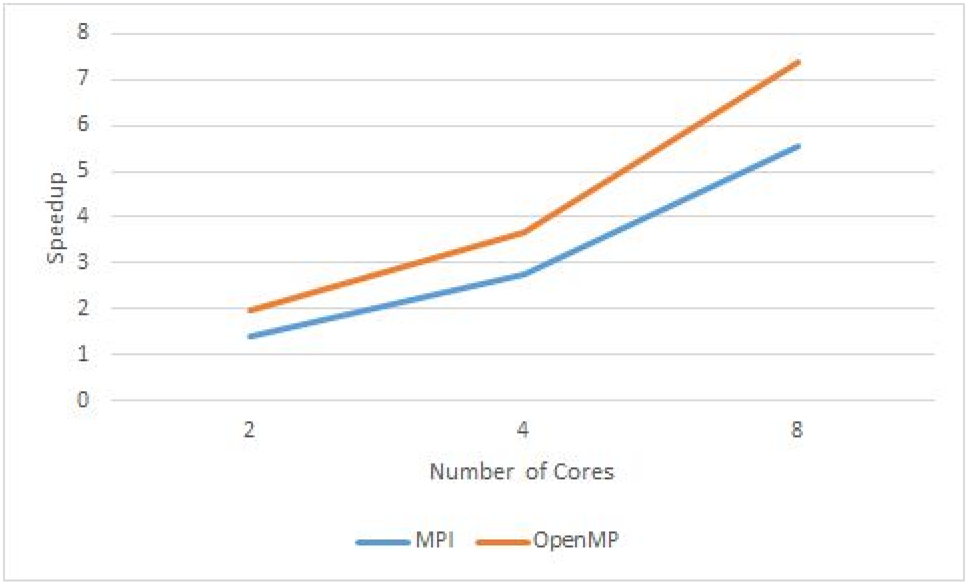
As we increase the number of processors, the speedup increase. The slope of the graph decreases as we go to higher number of processors showing slower increase in speedup due to communication overhead. For the larger problem size we get more speedup for each number of processors.



Problem Size vs Speedup for 4 and 8 cores in OpenMP and MPI

Observation:

For both 4 and 8 cores OpenMP performs better than MPI. This is due to communication overhead in MPI. After Certain problem size speedup starts decreasing in OpenMP, while it is still increasing in MPI. This shows MPI can work better for larger problem sizes.



Number of cores vs Speedup in MPI and OpenMP for problem size 10^{10}

Observation:

Speedup increases as number of cores increases in both OpenMP and MPI. But increase in speedup in MPI is less than in the case of OpenMP due to communication overhead.

Problems faced in Parallelization :

A. In MPI, synchronization and system overhead(the time spent when copying the message data from sender's message buffer to network and from network to the receiver's message buffer) plays an important role but still we are able to get some speedup. This is because in MPI, we create processes and not threads. Each process uses its own variables for generating the sum, and hence there is no overhead due to racing conditions on the shared variables, as in the case of OpenMP.

B. Synchronization problem is faced when we increase the number of processors as all the processor send their data to master process there is a possibility of synchronization overhead which reduces speedup.

C. System overhead:the time spent when copying the message data from sender's message buffer to network and from network to the receiver's message buffer..