HPC Assignment 7

Comparison between OpenMP and MPI Implementation for Approximating value of PI using trapezoid rule

Submitted by:

Saumya Bhadani 201301100

Shruti Singh 201301452

1. Problem statement : We have to compute the value of pi using integration of 4/(1+x^2) in the interval [0, 1]. For computing this, we can run a for loop which iterates over values of x, starting from 0 to 1. After every iteration, the value of x is incremented by a small amount. This leads to a large running time. Since the values x takes is independent of each other, the portion can be parallelized to achieve speedup.

• **Complexity** of the serial algorithm is O(n), where n is the number of divisions, the interval [0, 1] is divided into.

• **Possible speedup(theoretical)** :

Speedup S = 1/ (P/n + s)

n – number of cores
P – percentage of code that can be parallelized
s – percentage of serial code(which is not parallelized)

For our code, P ~ 1 and s ~0
n=4, So theoretical speedup = 4

2. Hardware details:

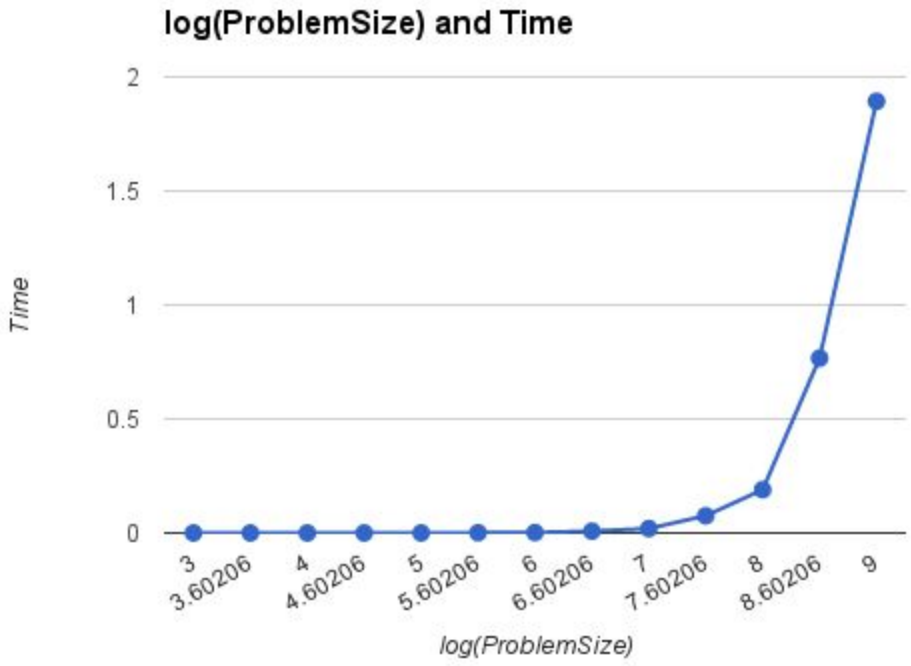
CPU : Intel® Core™ i5-4200U CPU @ 1.60GHz × 4
Compiler : gcc
Precision : Double
Peak performance = 4 FLOPs/cycle * 1.6GHz * 4 = 25.6 GFLOPS

For this particular assignment, we compare the MPI implementation with the OpenMP Implementation.

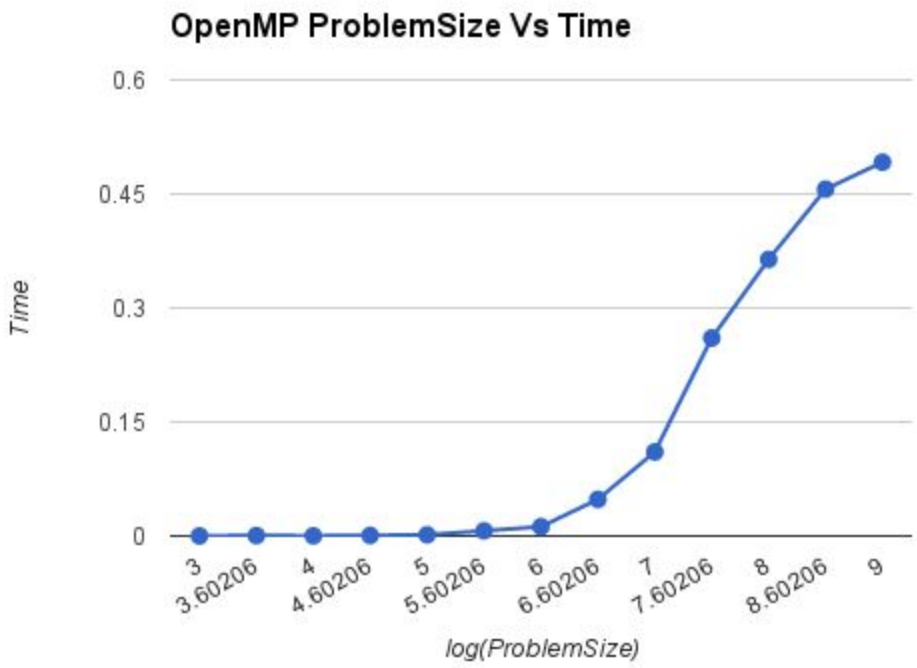
3. Output:

ProblemSize VS Time

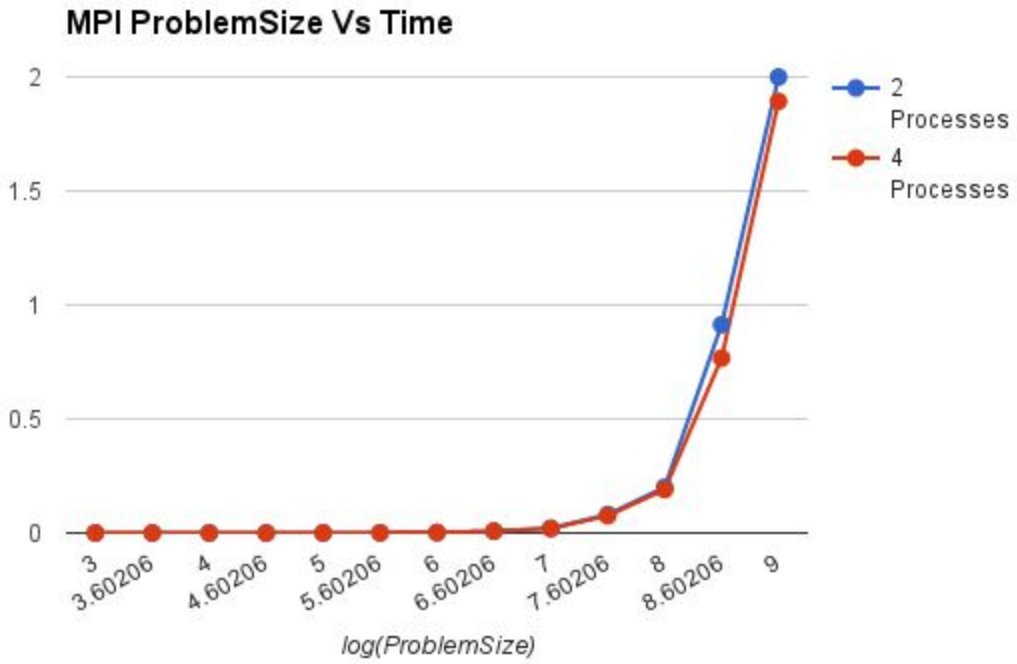
Serial implementation



OpenMP

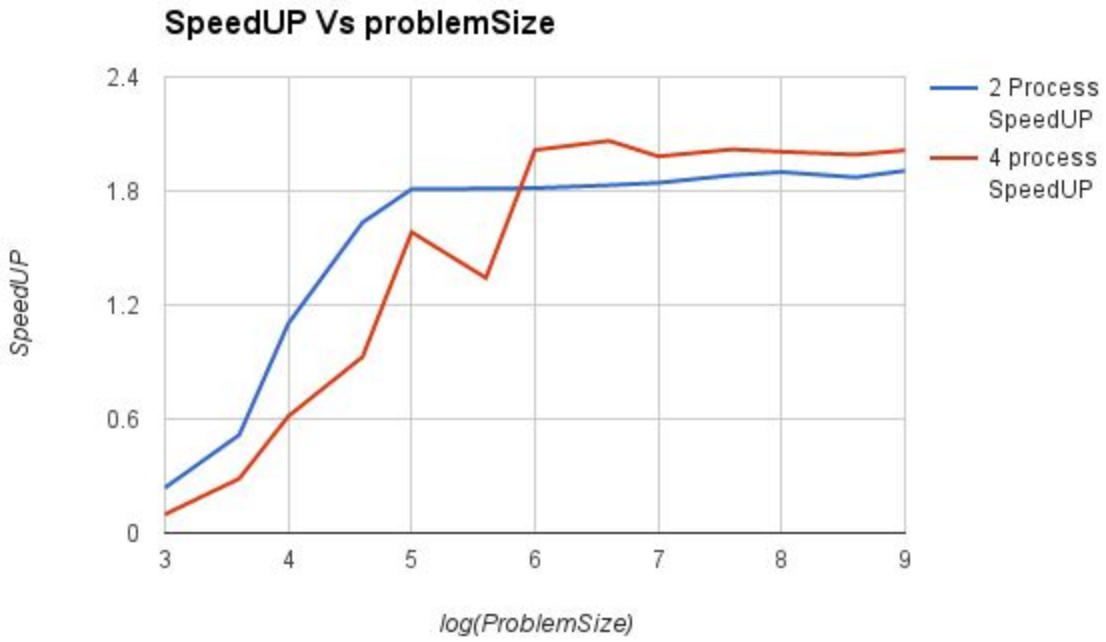

OpenMP ProblemSize Vs Time
0.6
0.45
0.3
0.15
0
Time
3
3.60206
4
4.60206
5
5.60206
6
6.60206
7
7.60206
8
8.60206
9
log(ProblemSize)

MPI

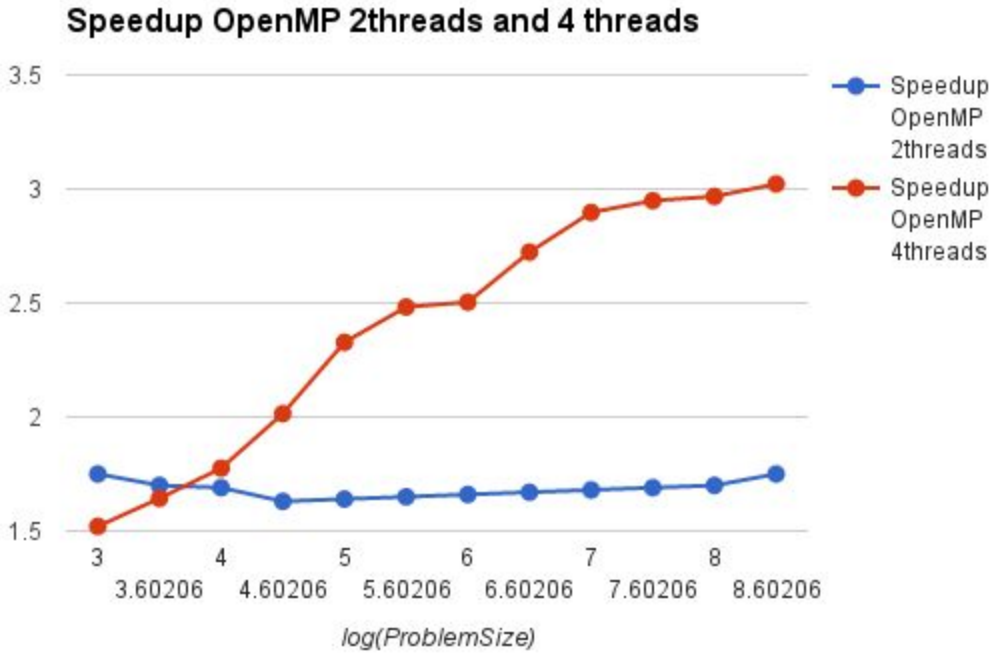

MPI ProblemSize Vs Time
2
1.5
1
0.5
0
3
3.60206
4
4.60206
5
5.60206
6
6.60206
7
7.60206
8
8.60206
9
log(ProblemSize)
2 Processes
4 Processes

Speedup for different threads

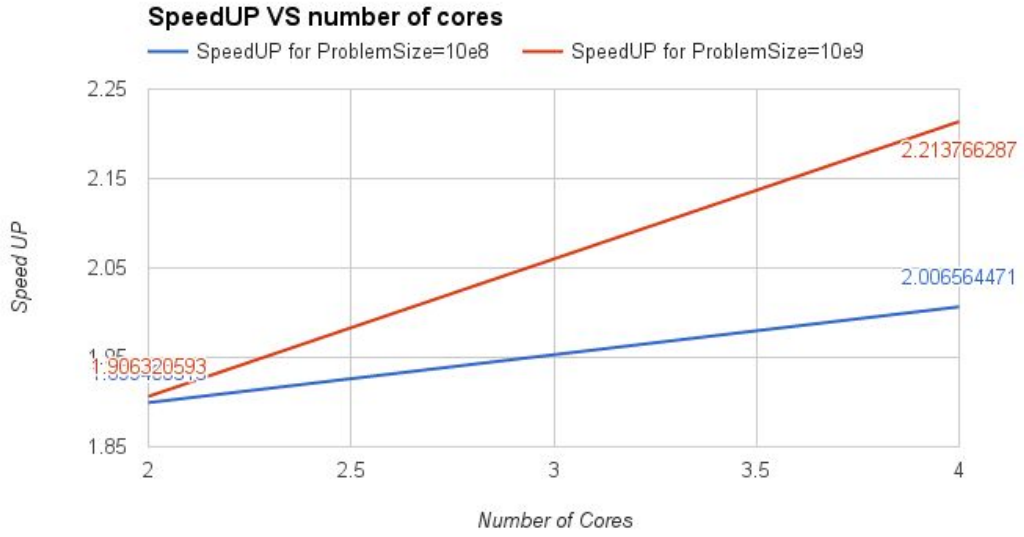
MPI



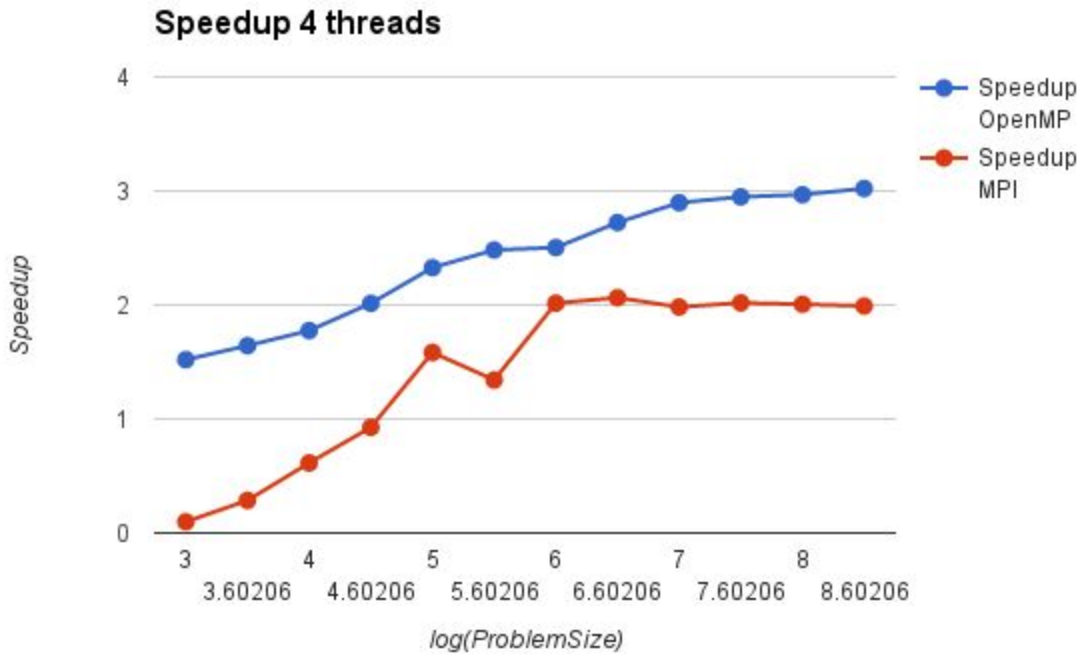
OpenMP

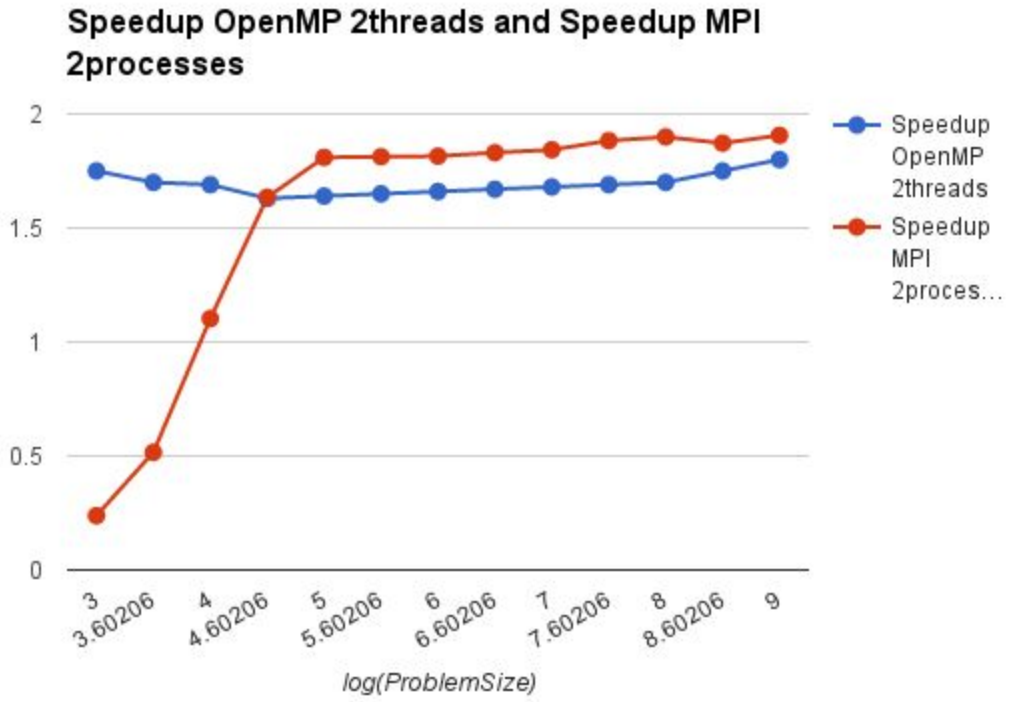


MPI



COMPARISON





OBSERVATIONS :

For 4 processes, speedup of OpenMP is better than MPI for all the problem sizes, while for 2 processes speedup of MPI increases than OpenMP after a particular problem size.

For 2 processes, speedup of MPI is better than OpenMP. For shared memory, passing messages between processes is slightly better than switching threads, which might be the reason.

For 4 processes, the openmp implementation is better than the MPI implementation as the communications involved in the MPI implementation might be dominating. In OpenMP, we used reduction, which might have been faster as compared to communications between 4 processes.

Also we can observe that in case of MPI, maximum speedup of 2 processes is 1.98 which is very good, while maximum speedup of 4 processes is 2.02 which is not good.

KARP-FLATT METRIC

Processes	**2**	**4**
e	**0.01**	**0.125**

From the Karp-Flatt metric, we can observe that values of e are increasing and we can conclude that reason for poor speedup is overheads involved due to processes.

The same thing is validated from the above obtained speedups, for 2 and 4 processes.