HPC Assignment 5

Approximating value of PI using trapezoid rule (using Open MPI)

Submitted by:
Shruti Singh 201301452
Saumya Bhadani 201301100

1. Problem statement : Approximate value of pi

- We have to calculate the value of PI using the trapezoid rule.We compute the value of pi using definite integral of 4/(1+x^2) in the interval [0, 1].
- This can be visualised as dividing the area under the function 4/(1+x^2) into n rectangles and then finding the summation of area of all rectangles.

  We implement the above method in three ways. We write a purely serial code, and two parallel codes, using OpenMP and Open MPI parallel directive.
  We compare the running times of the above three algorithms and record the observations and speedup as follows.

- **Complexity :**
  Complexity of the serial algorithm is O(n), where n = number of divisions, the interval [0, 1] is divided into(i.e. problem size).
  Complexity of the parallel algorithm is O(n/p), where p = number of cores on the machine.

- **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

- Optimization strategy

For the naive parallel code using OpenMP

We have four cores, so the code can be parallelized into 4 segments. The interval of x, i.e. [0,1] if divided into n steps, then n/4 iterations can be performed by each thread. Theoretically, this will increase the speedup of the process 4 times the serial code.

- Open MPI

  Open MPI is a Message Passing Interface (MPI) library project. Here, different processes communicate with each other using MPI communication protocol. MPI codes run on shared-memory multi-processors, distributed-memory multicomputers, cluster of workstations, or heterogeneous clusters of the above.

  The six main functions of MPI are:

MPI_Init()	MPI_Comm_rank()
MPI_Finalize()	MPI_Recv()
MPI_Comm_size()	MPI_Send()

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

3. Output: The value of pi approximated by the machine. The time taken for computation of the value.

For 8 cores:

Problem Size	Serial Time (seconds)	Naive parallel Time OpenMP (seconds)	Parallelization using Open MPI	Value of pi approximated
1000	0.000023;	0.000083	0.000161	3.1415927369
4000	0.000081;	0.000121	0.000863	3.1415926588
10000	0.000196;	0.000236	0.000909	3.1415926544
40000	0.000783;	0.000714	0.000423	3.1415926536
100000	0.001954;	0.002511	0.000884	3.1415926536
400000	0.008733;	0.007107	0.002828	3.1415926536
1000000	0.019460;	0.020635	0.006725	3.1415926536
4000000	0.077808;	0.094888	0.013651	3.1415926536
10000000	0.194256;	0.160862	0.051374	3.1415926536
40000000	0.776629;	0.353436	0.137977	3.1415926536
100000000	1.941913;	0.421661	0.340176	3.1415926536
400000000	7.767228;	0.509591	1.362883	3.1415926536
1000000000	19.422764;	0.703818	3.386835	3.1415926536

4 cores

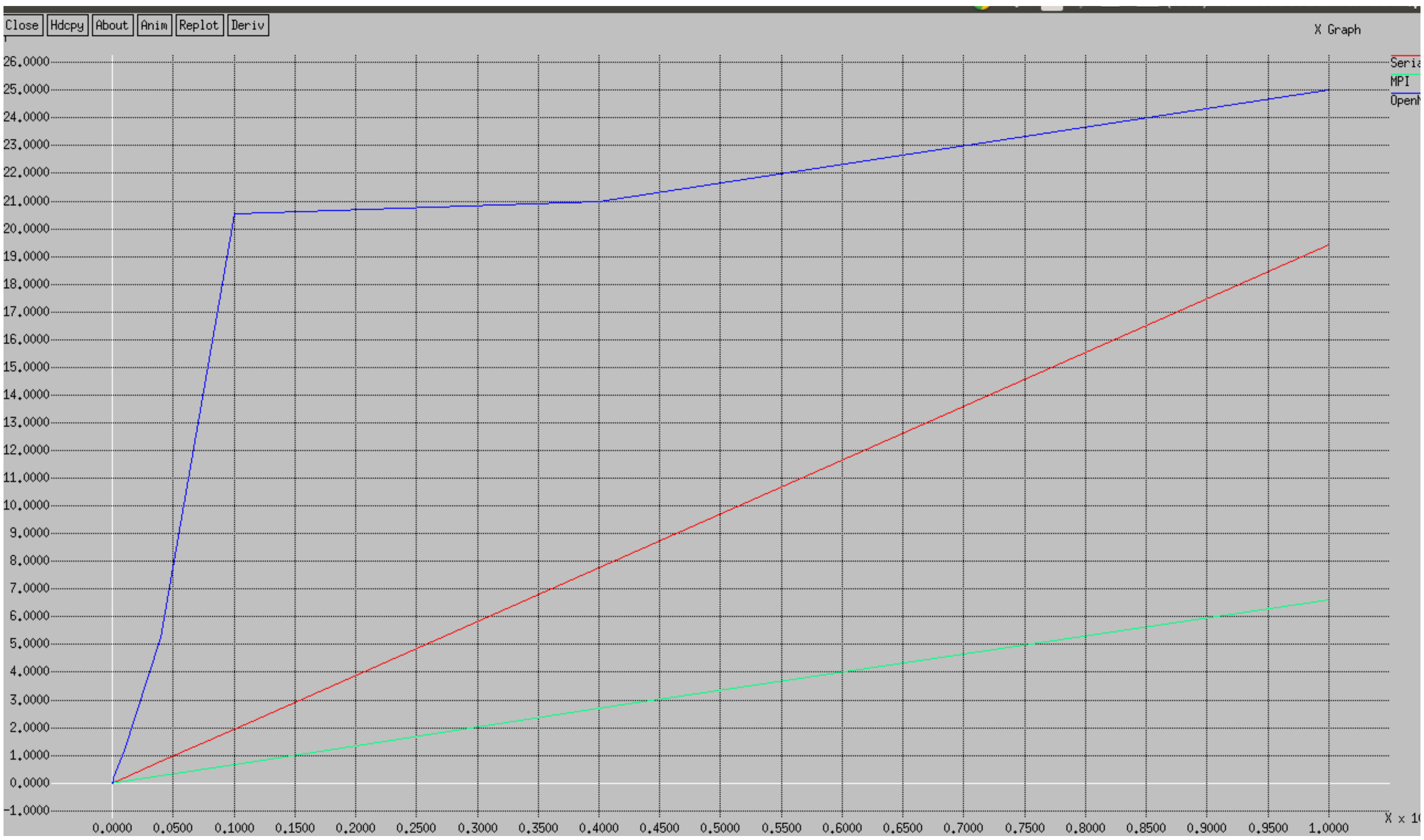
Problem Size	**Serial Time (seconds)**	**Naive parallel Time OpenMP (seconds)**	**Parallelization using Open MPI**	**Value of pi approximated**
1000	0.000023;	0.000424;	0.000139;	3.1415927369
4000	0.000081;	0.000449;	0.000072;	3.1415926588
10000	0.000196;	0.002377;	0.000229;	3.1415926544
40000	0.000783;	0.003479;	0.000721;	3.1415926536
100000	0.001954;	0.013778;	0.001387;	3.1415926536
400000	0.008733;	0.025040;	0.004886;	3.1415926536
1000000	0.019460;	0.195865;	0.007495;	3.1415926536
4000000	0.077808;	0.601682;	0.028812;	3.1415926536
10000000	0.194256;	1.193032;	0.070461;	3.1415926536
40000000	0.776629;	5.251369;	0.268422;	3.1415926536
100000000	1.941913;	20.53164;	0.673055;	3.1415926536
400000000	7.767228;	21.000424;	2.707683;	3.1415926536
1000000000	19.422764;	25.000449;	6.601967;	3.1415926536

Speedup

1. Serial Vs Open MPI

4 cores	**8 Cores**
0.16547	0.142857
1.125	0.093859
0.8559	0.215622
1.08599	1.851064
1.4088	2.210407
1.78735	3.088048
2.5964	2.89368
2.70054	5.699802

2.75693	3.781212
2.89331	5.628684
2.88522	5.708554
2.86859	5.699116
2.94197	5.734783



Speedup Vs ProblemSize