Computer Architecture HW-2

1. Pi Estimated by using Monte-Carlo techniques to solve the Buffon’s Needle problem

Source code :- **MpiSendRecv.c**

Time Elapsed during execution: 0.021038s for total trials of 400000.

1. pi is estimated again by using MPI reduce instead of MPI\_Send and MPI\_Recv

Source Code: **MPIReduce.c**

Time Elapsed during Execution: 0.022795s for total trials of 400000.

**Performance Analysis:** The MPI implementation using MPI \_Send and MPI\_Revc is faster but there are lot of dependencies on which implementation time depends on. There are few instances when MPI\_Reduce is faster than the MPI\_Send and MPI\_Recv. So It is difficult to state one implementation is faster than other very specifically.

**An embarrassingly parallel** is called when little or no effort is required to separate the problem into parallel tasks. Here in implementation of Buffon’s needle problem there is no dependency for any process to get or send data to other processes. Splitting the task into parallel execution requires very little effort.

1. A parallel multiplication of size 32 x 32 is done

Source code: **MatrixParallel.c**

Time taken for executing the matrix multiplication:

Source code for serial Matrix Multiplication: **MatrixSerial.c**

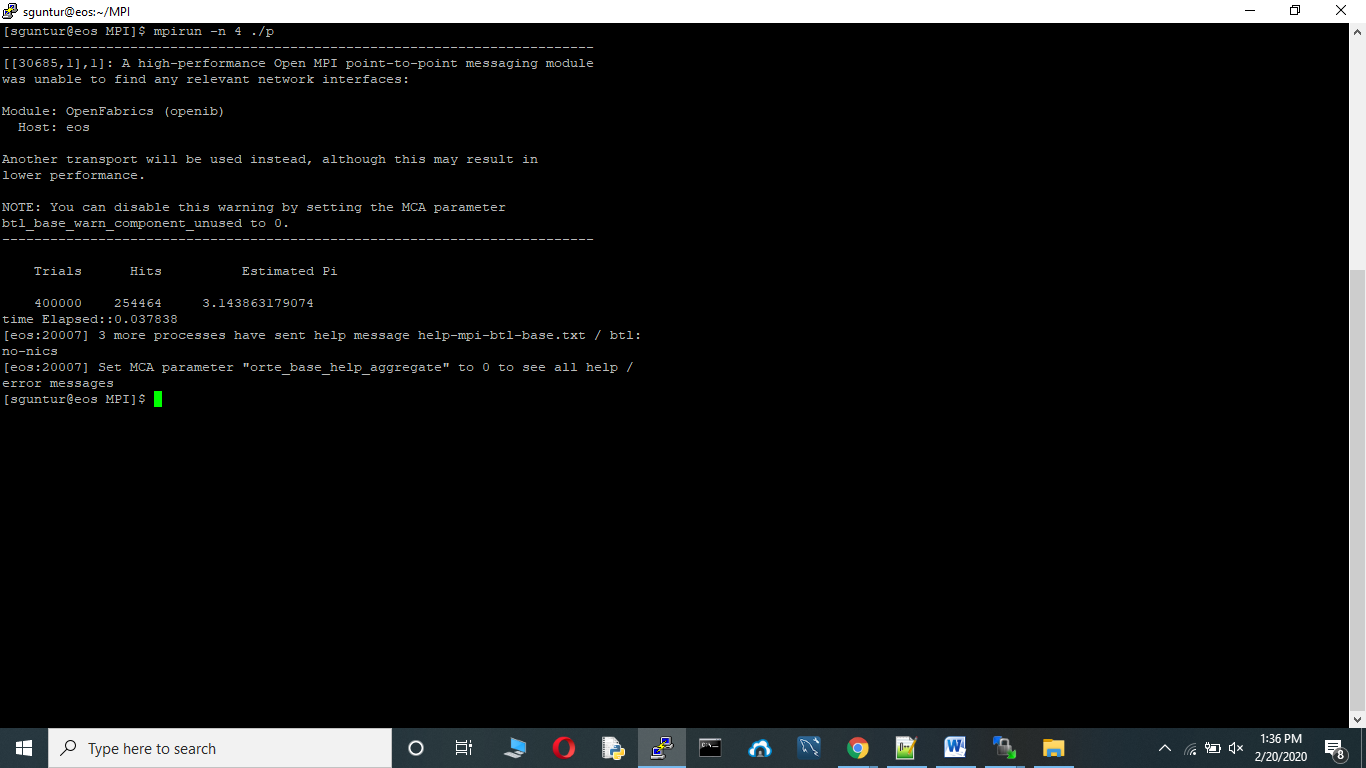
Correctness of the code is verified by Serial matrix multiplication by taking same values for the all the elements in the matrix as ‘1’

Performance comparison:

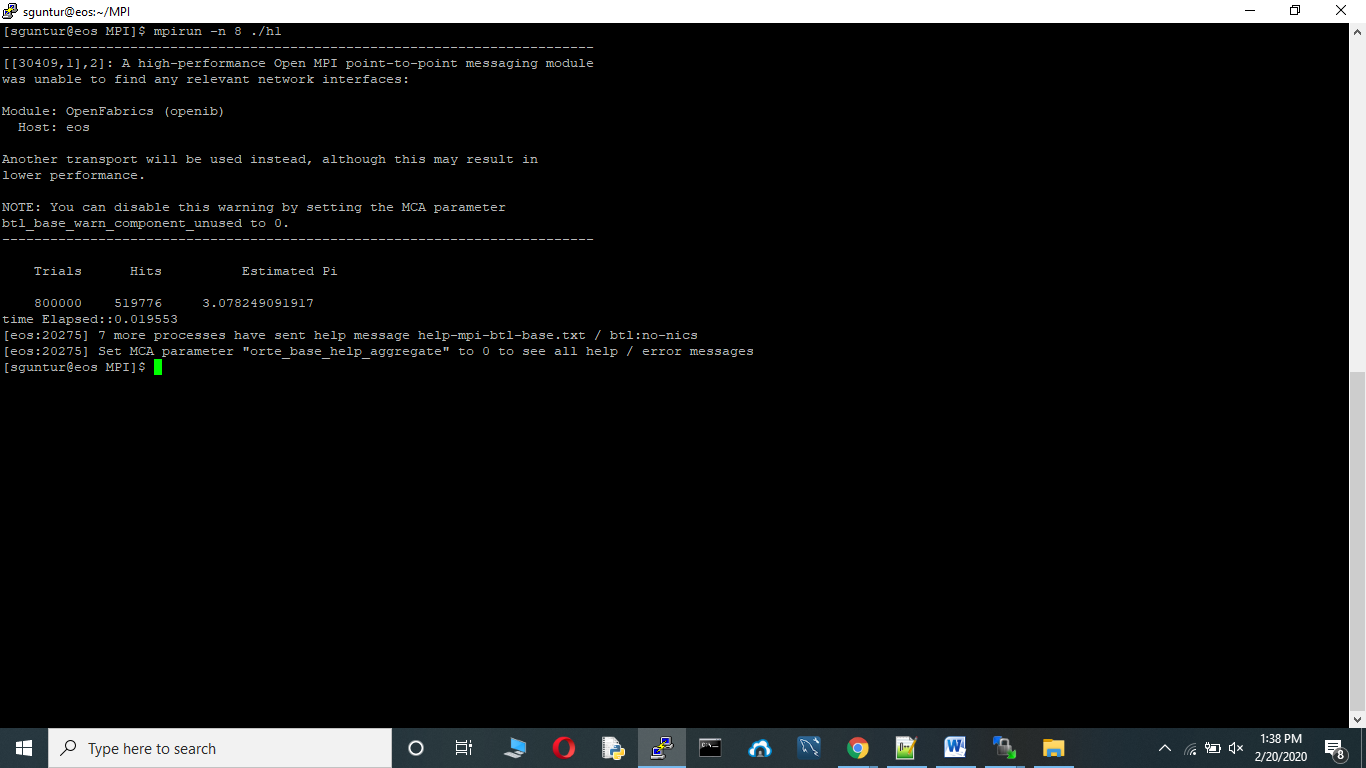
|  |  |  |
| --- | --- | --- |
| Type | Size | Performance in Time (wall clock time) |
| Parallel | 32 x 32 | 0.002757s |
| Parallel | 512 x 512 | 2.001577s |
| Serial | 32 x 32 | 4.578s |
| Serial | 512 x 32 | 12.483s |
| Serial | 1024 x 1024 | 40.163s |

As Observed the time taken for Parallel multiplication is far less than the time taken for serial multiplication. Hence, Improvement in performance is clearly visible.

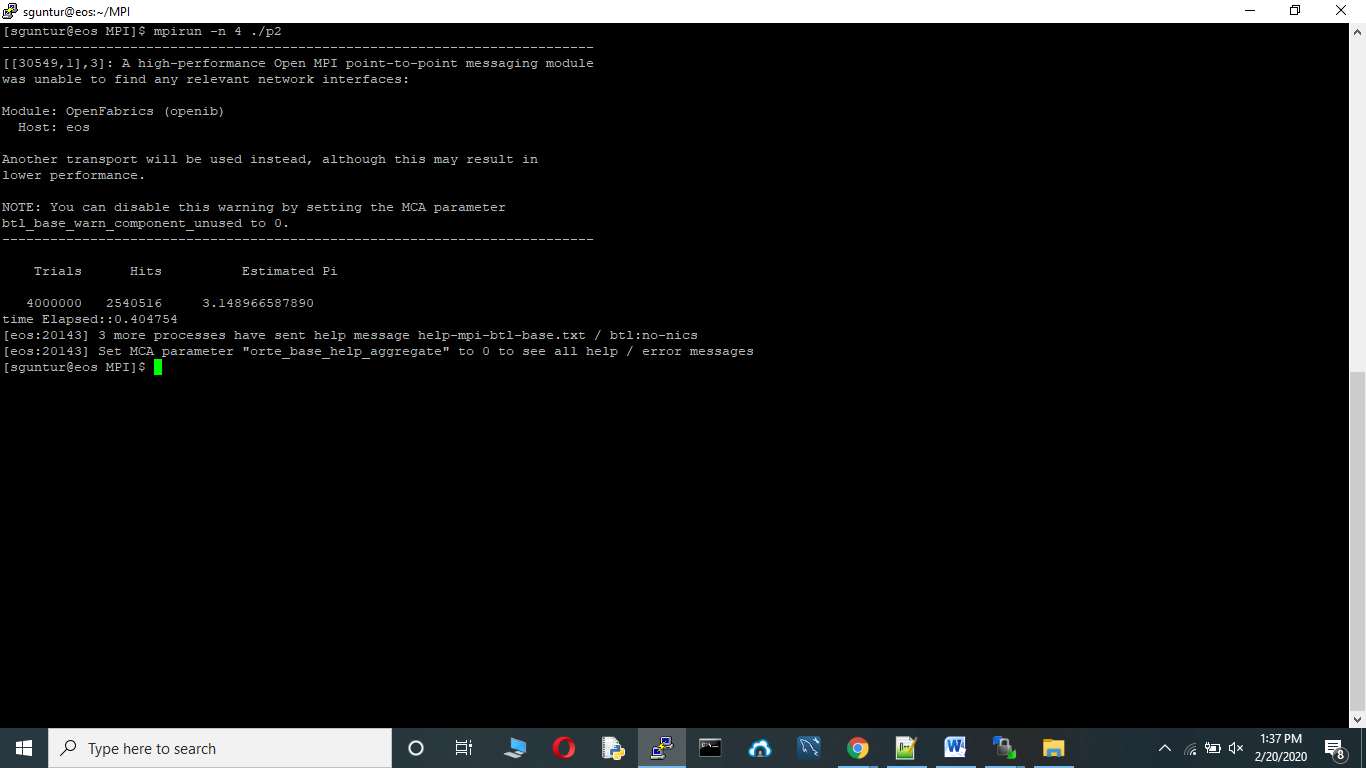
Results for The programs for one of my implementation:



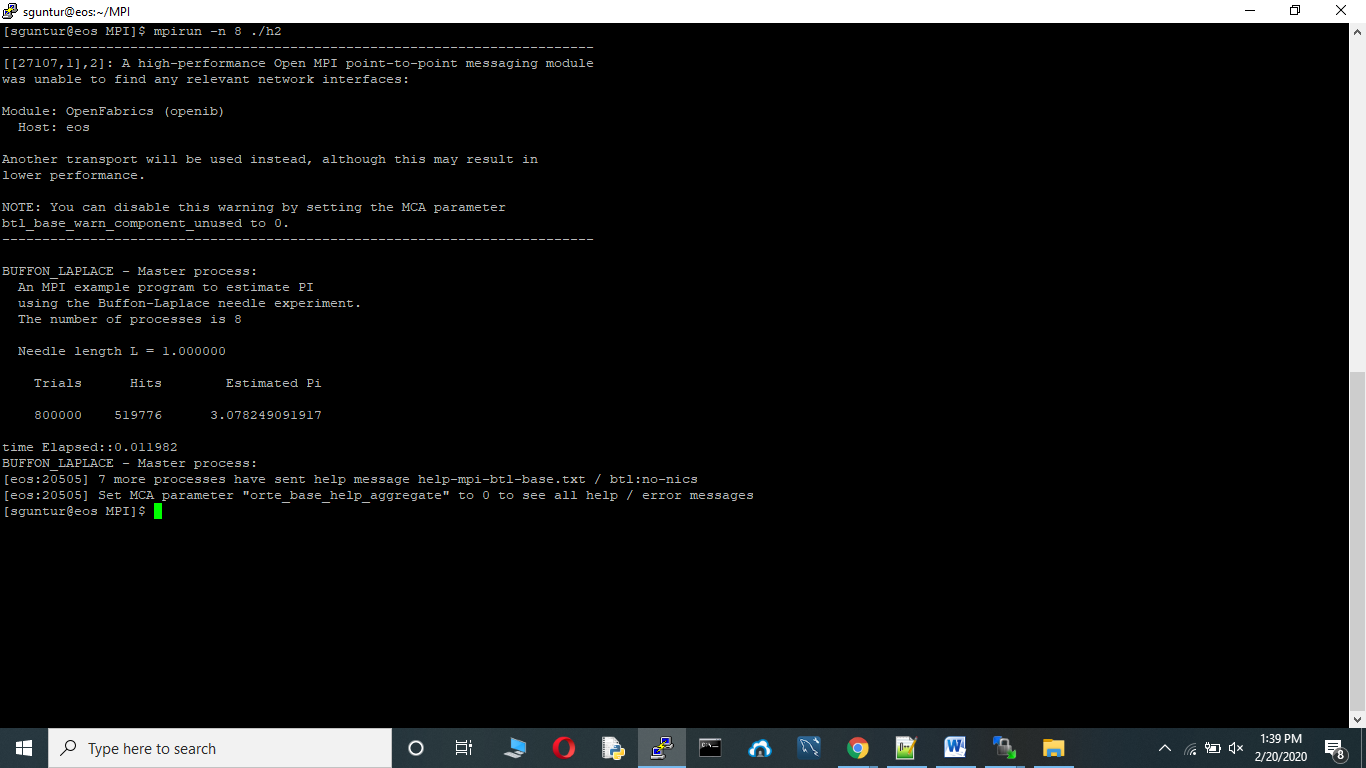
Other implemtation:



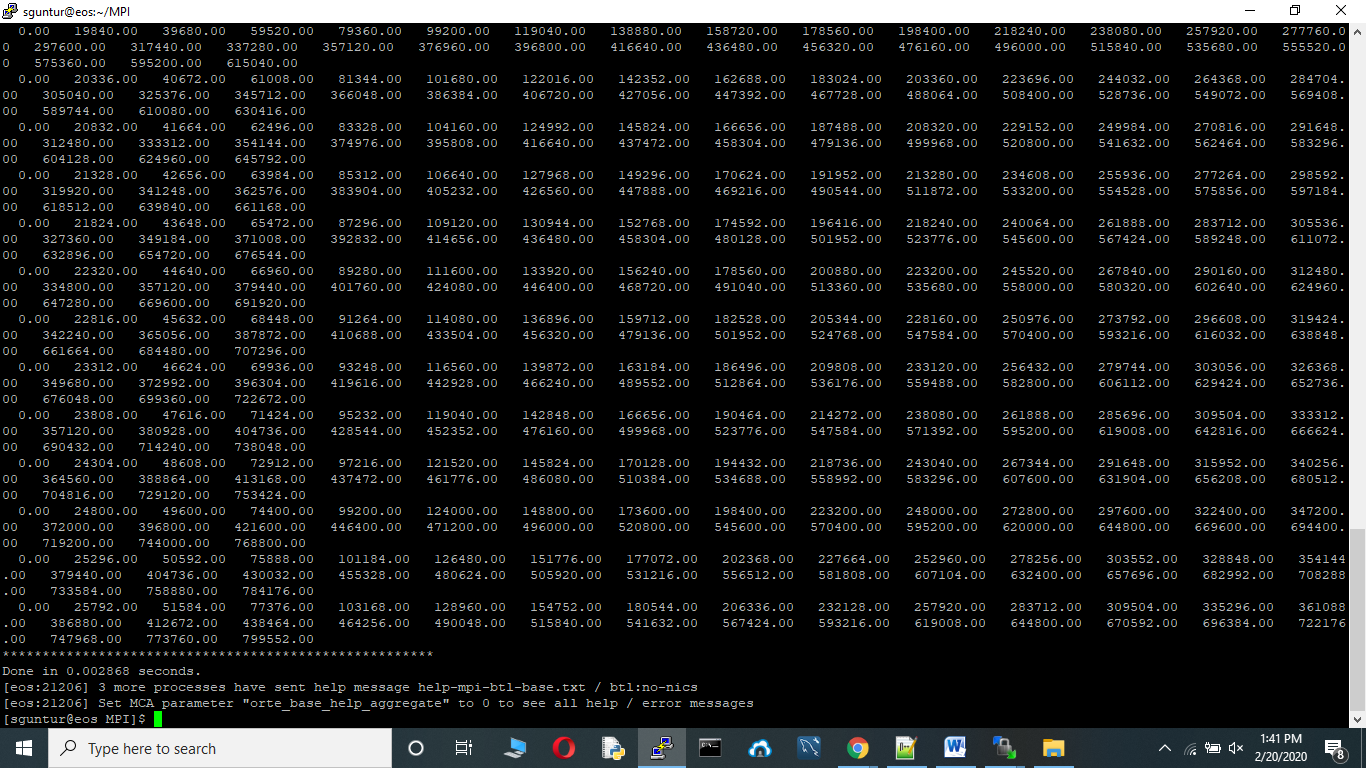
MPI\_Redude:-

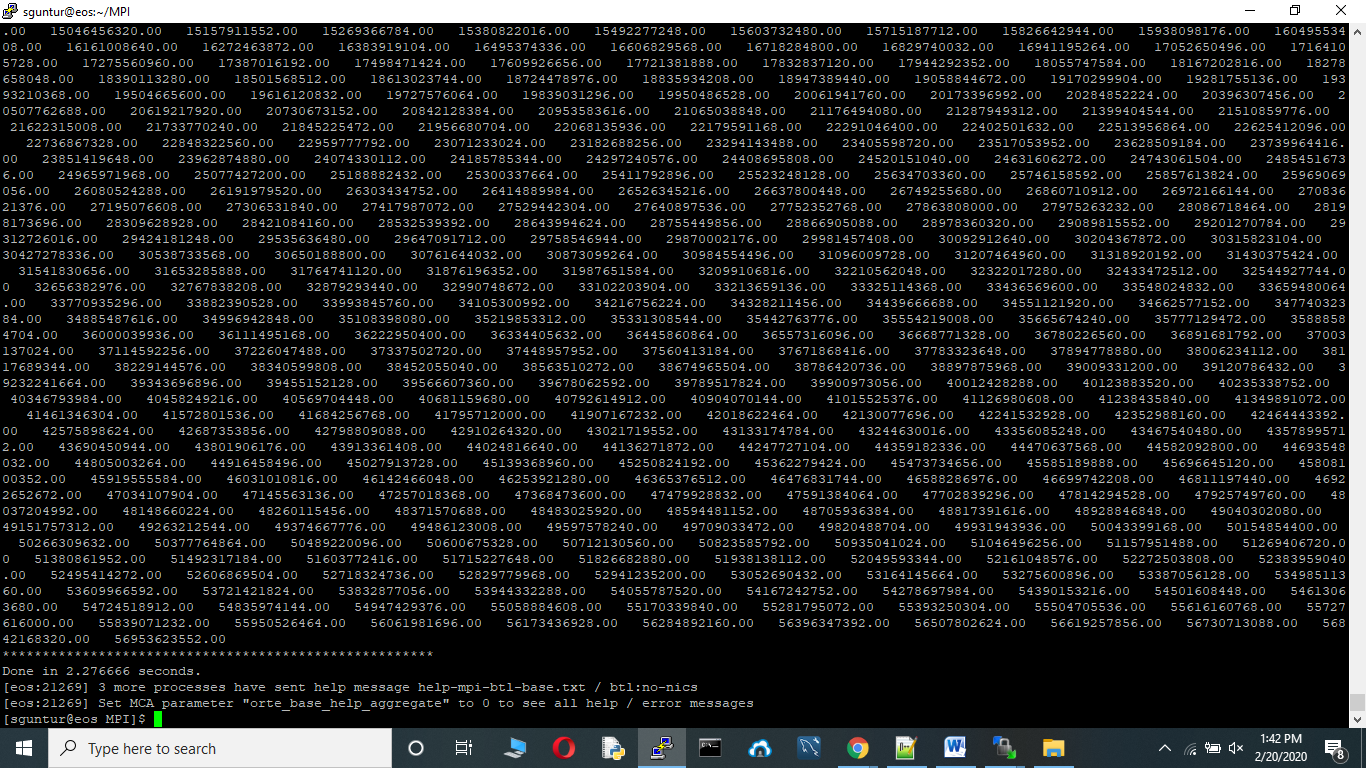


Other Implementation using Reduce:



Parallel Matrix multiplication Results for size 32 and 512:





Serial Matrix multiplication results for 32 and 1024:

