Name : Rakesh Baingolkar
UBID : 50097576
Course : Parallel and Distributed Processing

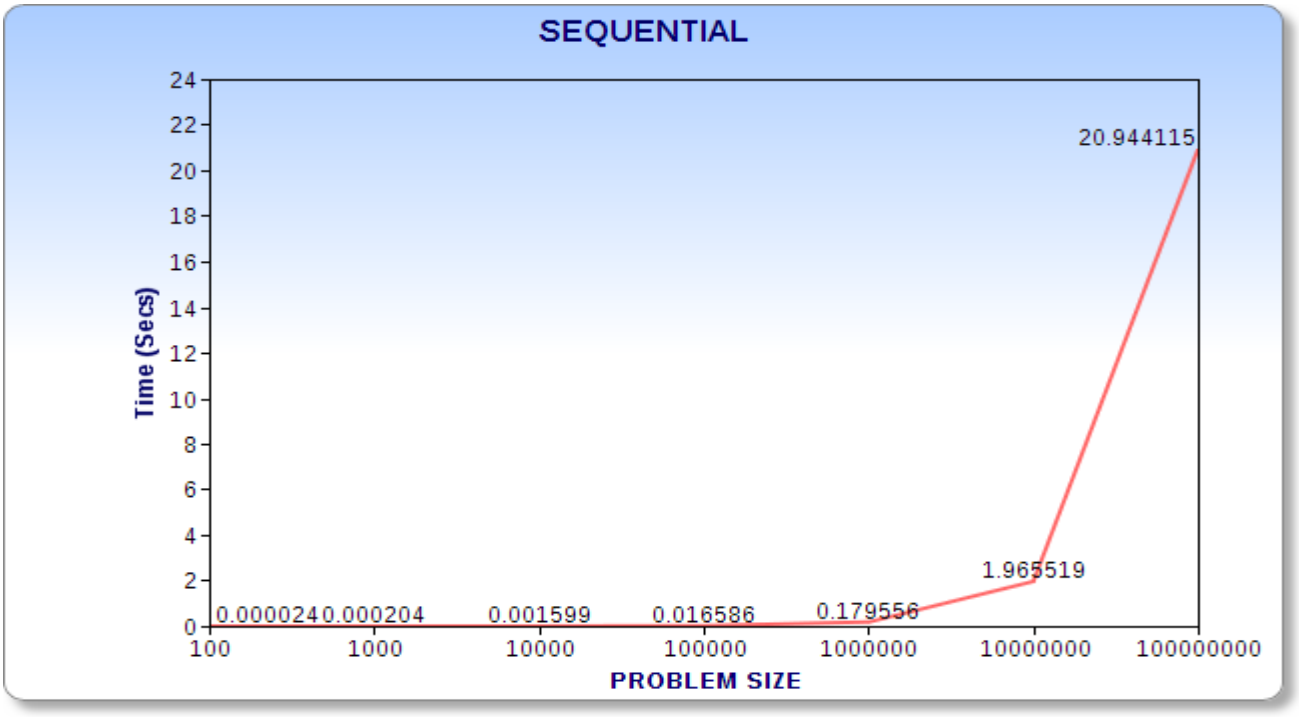
PROGRAMMING ASSIGNMENT 2

OBJECTIVE : Implement a Bucket sort algorithm in Sequential, OpenMP, MPI and OpenMP_MPI

SEQUENTIAL

Directory : Sequential
Filename : Sequential.c
Makefile : makefile
Command to run MakeFile : make
Object File: sequential
Command to Run Object File : ./sequential P N y
P->No_of_processors
N->Problem Size
y->To write on a file or n-> Do not write on a file

In the sequential approach I basically put the entire data in a single bucket for sorting. After the bucket is sorted I put the entire data back in the input. Insertions sort was used for sorting. For this I increased the Problem Size and the observations were as follows.

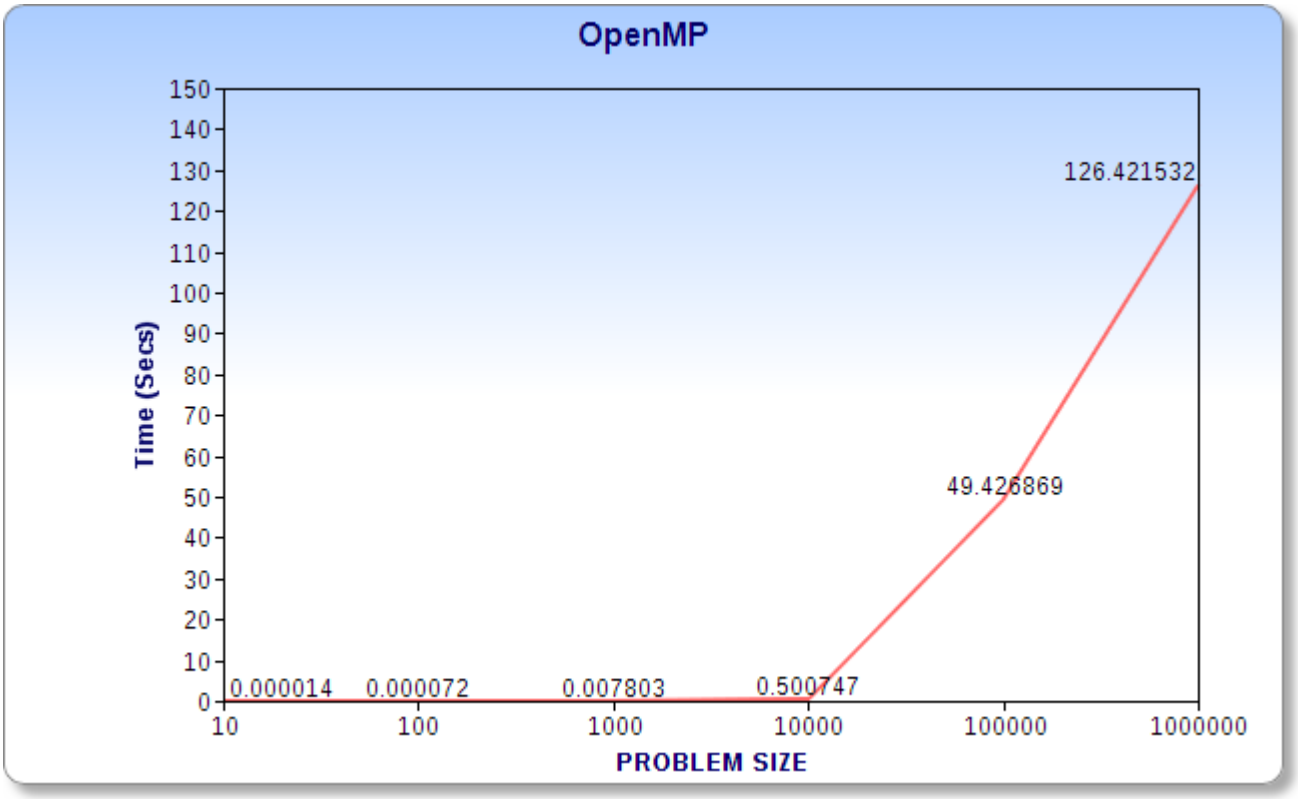


From the graph we can see that as we increase the Problem Size the Time also

increases.

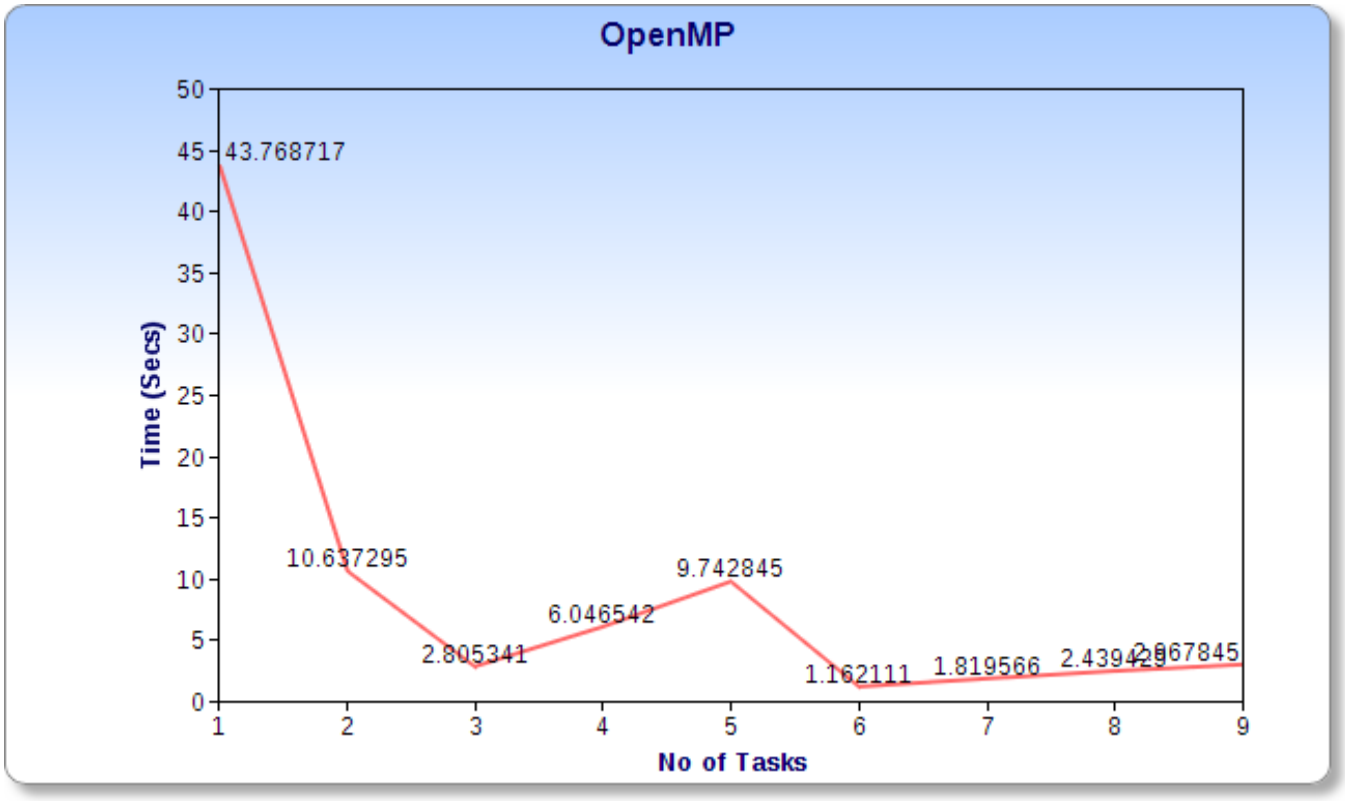
OPENMP
Directory : OpenMP
Filename : OpenMP.c
Makefile : makefile
Command to run MakeFile : make
Object File: OpenMP
Command to Run Object File : ./OpenMP P -t N y
P->No_of_cores
N->Problem Size
y->To write on a file or n-> Do not write on a file
In OpenMP as I have parallelized the process by which the input is assigned a particular bucket. The Bucket sorting is also done in parallel which is independent of each other and the sorting used is Insertion Sort. The process by which the buckets are concatenated to form the sorted input is also parallelized.

We try two methods to implement OpenMP.
First to increase the Problem Size.



The time increases with the problem size.

Next we increase the No of tasks per node i.e. no of threads the graph is as follows.

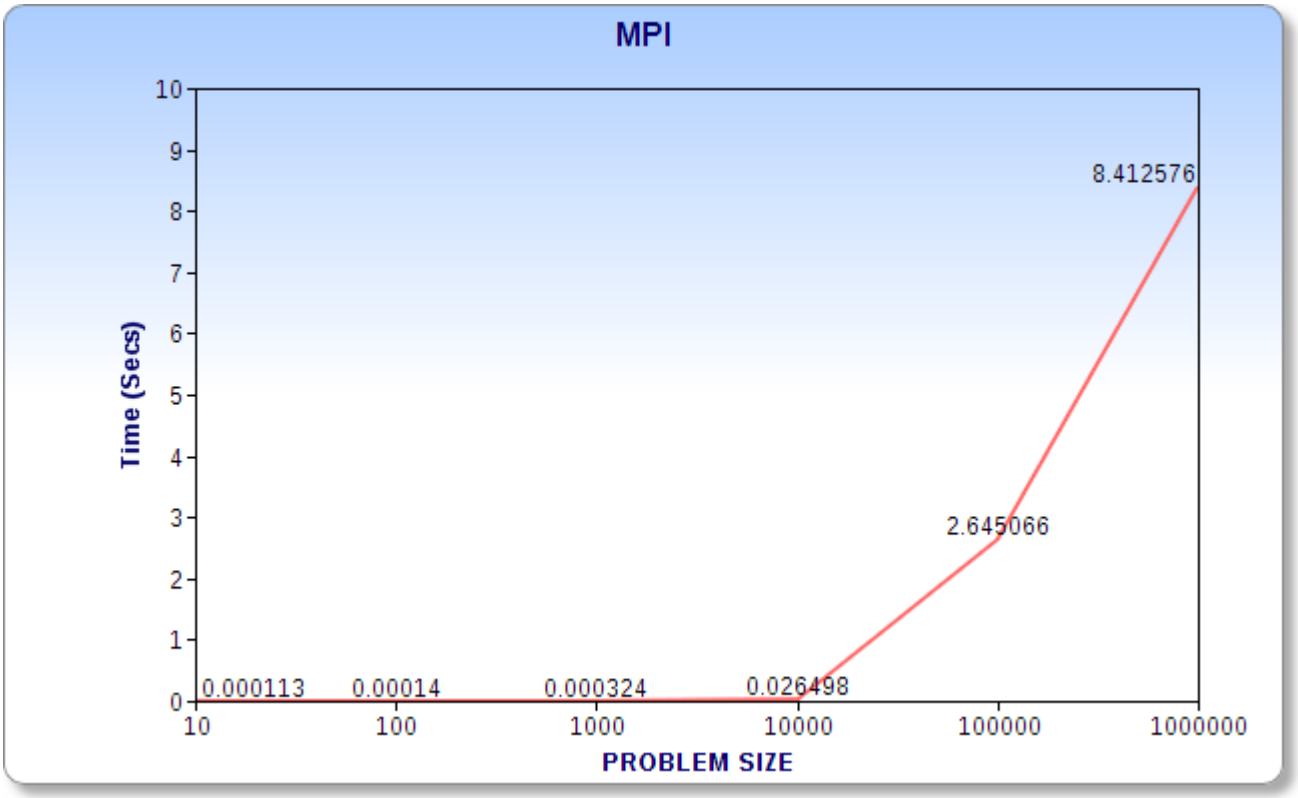


The time decreases and the there is a peak in betweek which may indicate overhead due to context switching.

MPI
Directory : MPI
Filename : MPI.c
Makefile : makefile
Command to run MakeFile : make
Object File: MPI
Command to Run Object File : ./MPI P N y
P->No_of_nodes
N->Problem Size
y->To write on a file or n-> Do not write on a file

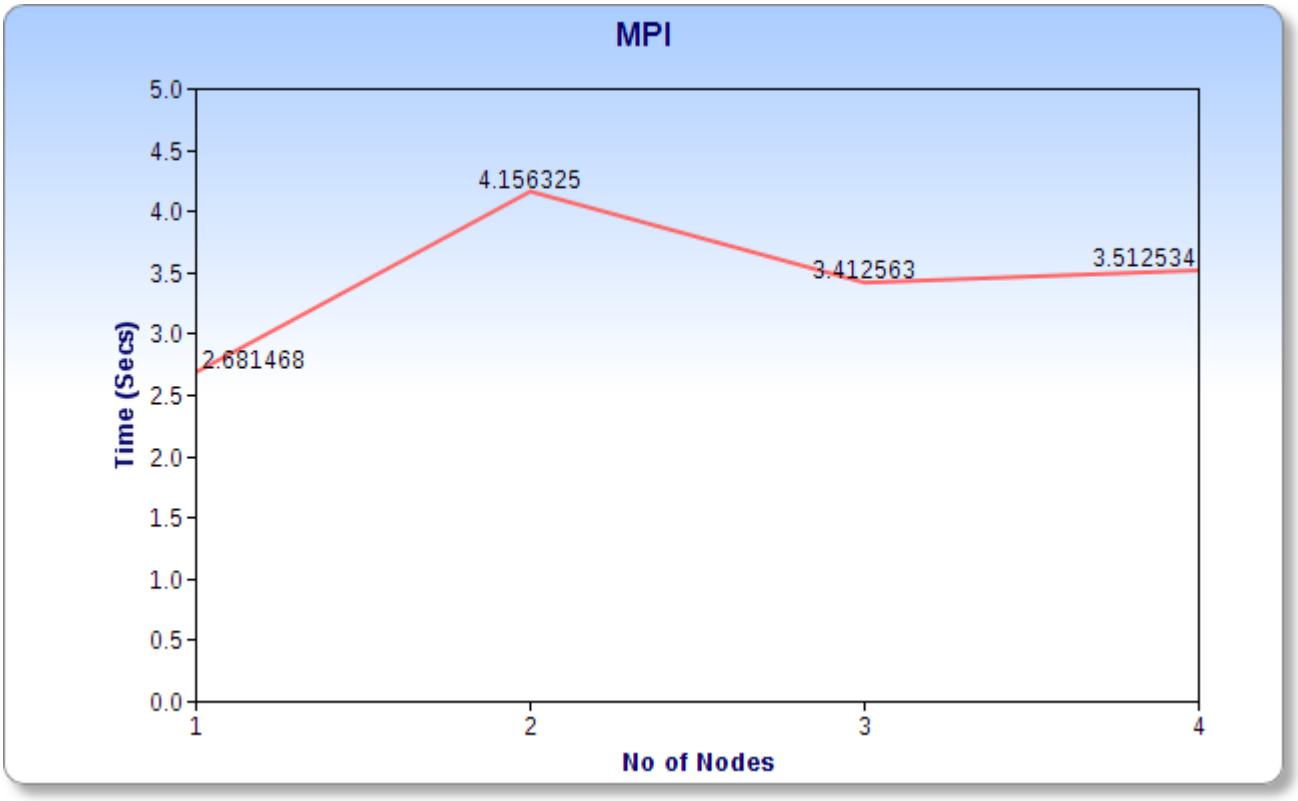
In MPI we take the input set then we divide the input set into corresponding buckets. Then thse buckets are given to the nodes for sorting and after sorting the buckets are returned and then they are concatenated to form the sorted output that we desire.

In MPI we take two observations
First we increase the size of the problem.



The time increases with incrase in problem size.

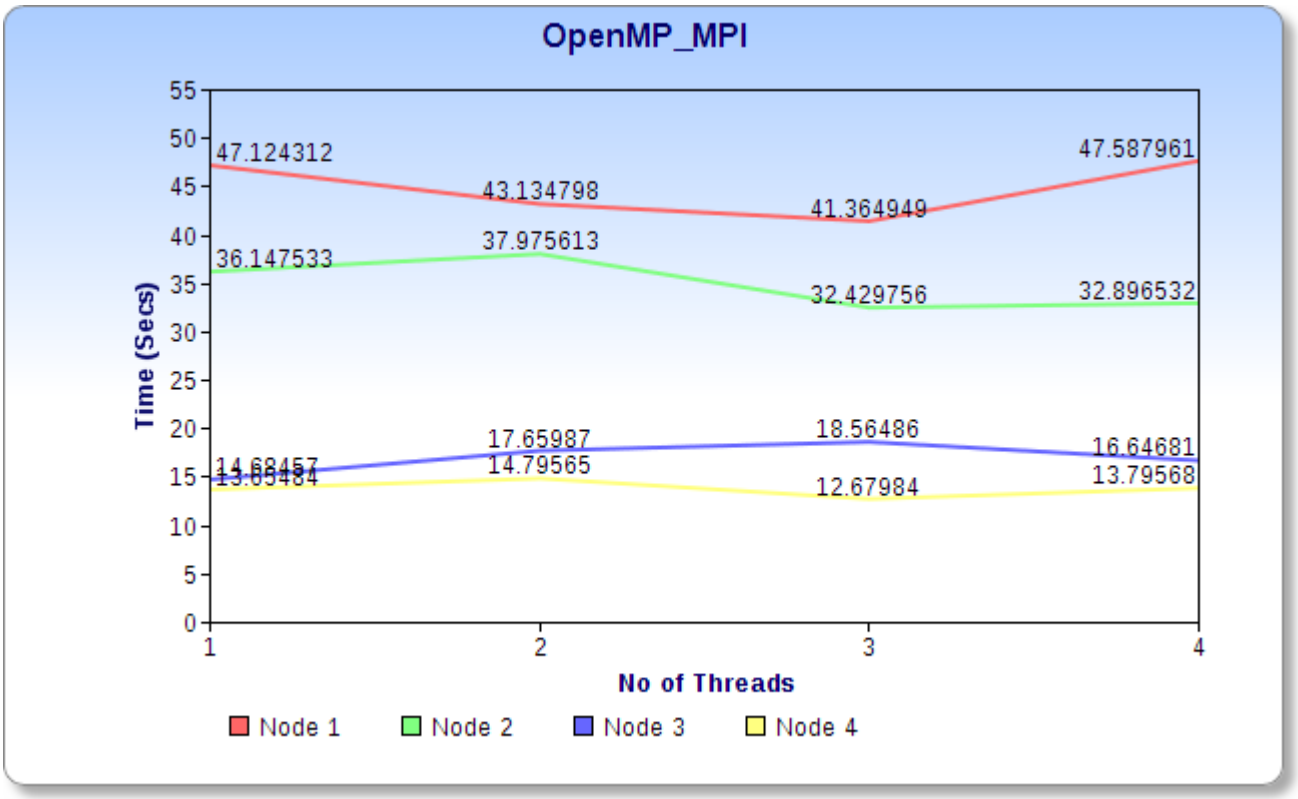
Next we increase the number of nodes.



The time increases with increase in node which may be because of the overhead of transferring data between two machines.

OPENMP_MPI
Directory : OpenMP_MPI
Filename : OpenMP_MPI.c
Makefile : makefile
Command to run MakeFile : make
Object File: OpenMP_MPI
Command to Run Object File : mpirun -n T./OpenMP_MPI P N y
T-> No of Nodes
P->No_of_threads
N->Problem Size
y->To write on a file or n-> Do not write on a file

In OpenMP_MPI we include take the part which is assigned to a particular node and then we parallellize that task into number of threads.



Program	No of cores	Problem Size	Time
OpenMP	1 node 2 task-per-node	100000	7.43928

OpenMP	1 node 2 task-per-node	1000000	10.637295
OpenMP	1 node 8task-per-node	1000000	2.439
MPI	1 node 2 task-per-node	100000	2.959305
MPI	1 node 2 task-per-node	1000000	6.781253
MPI	2 node 8 task-per-node	1000000	3.799325
OpenMP_MPI	1 node 2	100000	43.134798
OpenMP_MPI	1 node 2	1000000	48.698235
OpenMP_MPI	2 node 8	1000000	13.648712