**Parallel and Distributed Computing**

**PROJECT NAME:**

**INSERTION SORT ALGORITHM**

**BY OPENMP AND MPI**

# OBJECTIVE:

Our project mainly focuses on the shared and distributed memory using OpenMP and MPI respectively. The performance comparison will be observed by the implementation of Insertion Sort Algorithm with increasing size from 100 to 5000000.

# INTRODUCTION:

Our project focuses on the parallel computation by shared memory (OpenMP) and Distributed Memory (MPI).Insertion Sort is implemented in openMp and mpi and with different sizes of array the time is calculated to know which parallel environment work faster.

# METHODOLOGY (PROJECT DESCRIPTION ):

we are going to implement Sorting algorithm on OpenMp and MPI as follows:

# INSERTION SORT USING OPENMP:

For multithreading in shared memory, We have used OpenMp (#pragma omp) for doing Insertion sort with help of Merge function because insertion sort is not Adaptive so if we even divide the data and apply insertion sort separately and then applying at whole, it will take the same time as it took serially sometime maybe greater time than this. So for working parallel and quickly insertion sort at half of the data parallel which is independent of each other than after that merge function is called out to merge the arrays which is sorted which will take Log (n) time.

Time.h library is used to calculate the total time it took to sort the array of different size.

1. **INSETION SORT USING MPI:**

For multithreading in distributed memory, We have used MPI for Insertion sort in which parent process make the unsorted array which filled randomly generated element in the array and then send the slaves process some amount of element to sort, the slaves process which are running parallel is waiting to receive that array and whenever It get the array that process call the insertion sort on that size.

That’s how all the leaves process will be sorting the elements at the same time and when they complete their operation, they will send that array to parent function which is waiting for all the slaves to give response and then perform merge function which will take Log(n) time to sort

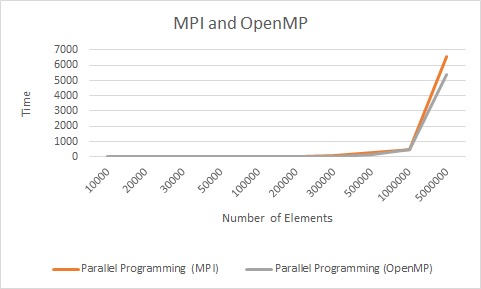
Time.h library is used to calculate the total time it took to sort the array of different size.

We are applying MPI in One machine that’s why we are using only shared memory processes.

**CHART:**

|  |  |  |  |
| --- | --- | --- | --- |
| Number of elements | Time (in seconds) taken by | | |
|  | Parallel Programming (MPI) | Parallel Programming (OpenMP) |
| 10000 |  | 0.1595584 | 0.153193 |
| 20000 |  | 0.472113 | 0.235819 |
| 30000 |  | 1.175424 | 0.403708 |
| 50000 |  | 2.98676 | 1.288482 |
| 100000 |  | 10.860088 | 5.976474 |
| 200000 |  | 45.195554 | 38.389737 |
| 300000 |  | 59.103418 | 42.154701 |
| 500000 |  | 272.883803 | 163.121111 |
| 1000000 |  | 467.126671 | 457.067291 |
| 5000000 |  | 6552.546928 | 5362.546928 |

# Graph:



# CONCLUSION:

The conclusion which can be drawn from the graphs/Comparison Chart is that:

Insertion Sort in OpenMp and MPI both take approximately equal time to compute and give sorted array at smaller size but as we are increasing the size of array MPI start taking more time than OpenMp which clearly shows that for Insertion Sort in OpenMp is best option in bigger size.

The maximum time taken by OPENMP to run Insertion sort is 1hr 48 minutes and maximum time taken by MPI is 1hr 82 minutes .

**CODE:**

***InsertionSort Using OpenMp:***

******



***InsertionSort Using MpI:***

******

******

******