Rohit Katariya

2020CSZ8844

Assignment 3

Parallel Sorting Using MPI

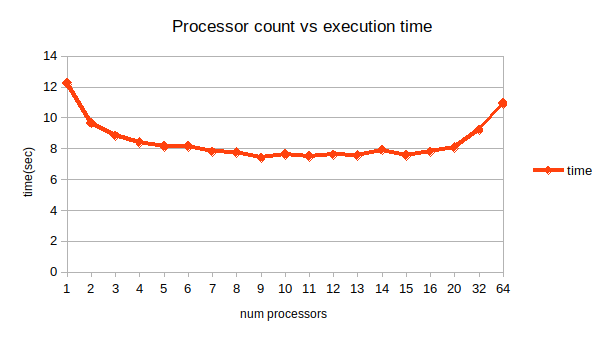
COL 730

2021

# Processor count vs Execution time graph

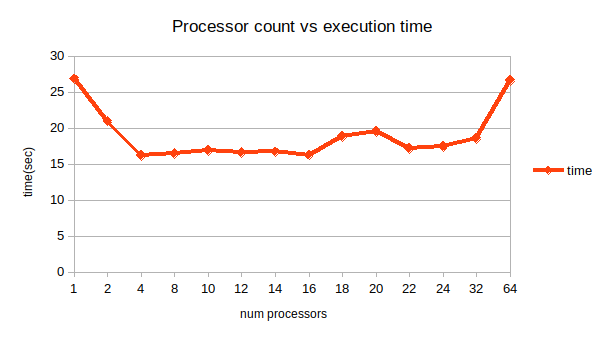
Experiment 1

|  |  |  |
| --- | --- | --- |
| size | nproc | time(sec) |
| 10000000 | 1 | 12.224 |
| 10000000 | 2 | 9.665 |
| 10000000 | 3 | 8.841 |
| 10000000 | 4 | 8.394 |
| 10000000 | 5 | 8.134 |
| 10000000 | 6 | 8.141 |
| 10000000 | 7 | 7.792 |
| 10000000 | 8 | 7.765 |
| 10000000 | 9 | 7.403 |
| 10000000 | 10 | 7.616 |
| 10000000 | 11 | 7.488 |
| 10000000 | 12 | 7.612 |
| 10000000 | 13 | 7.569 |
| 10000000 | 14 | 7.906 |
| 10000000 | 15 | 7.565 |
| 10000000 | 16 | 7.805 |
| 10000000 | 20 | 8.077 |
| 10000000 | 32 | 9.225 |
| 10000000 | 64 | 10.928 |



Experiment 2

|  |  |  |
| --- | --- | --- |
| size | nproc | time(sec) |
| 20000000 | 1 | 26.864 |
| 20000000 | 2 | 20.97 |
| 20000000 | 4 | 16.185 |
| 20000000 | 8 | 16.464 |
| 20000000 | 10 | 16.928 |
| 20000000 | 12 | 16.597 |
| 20000000 | 14 | 16.728 |
| 20000000 | 16 | 16.24 |
| 20000000 | 18 | 18.856 |
| 20000000 | 20 | 19.569 |
| 20000000 | 22 | 17.205 |
| 20000000 | 24 | 17.488 |
| 20000000 | 32 | 18.584 |
| 20000000 | 64 | 26.63 |



# Approach/ Algorithm used

* Designed parallelized version of quick sort
* **Why not merge sort**? Started with merge sort but thought it would require a lot of message passing and a lot of auxiliary memory creation required
* Parallelized quick sort:
  + Step 1: Message passing (all gather) between processors how many elements each has.
  + Step 2: Processor 1 sends pivot element to each processor(MPI\_Isend, MPI\_recv)
  + Step 3: Each processor divides itself into 2 parts left(containing items < pivot) and right(containing items >=pivot)
  + Step 4: Message passing to other processors that how many elements smaller than pivot are present with itself
  + Step 5: Processors compute whether they are left, right or pivot processors themselves; and start sending elements between each other swapping elements to get to a state where there are first all elements less than pivot then all with values more than pivot.
  + Step 6: pace pivot at the correct place by message passing to correct processor
  + Step 7: do recursive call on left and right arrays of pivot.

# Time analysis

* Time decreases with increase in number of processors , then becomes constant and after that it starts increasing again as the amount of communication between processors increases.
* This changes with amount of data with each processor as can be seen in the 2 experiments getting trough of minimum at different locations(9,4).