Sample Sort Using MPI

# Experimentation data for Serial computation

For serial computation we have chosen the serial code with 12 buckets and obtained the runtimes with size = 120000, 1200000, 12000000, 120000000. These results will be used as the baseline to compare with the parallel models.

|  |  |
| --- | --- |
| **(N)** | **Runtime (sec)** |
| 120000 | 0.058558 |
| 1200000 | 0.407464 |
| 12000000 | 3.881106 |
| 120000000 | 44.601656 |

# MPI Implementation

## Overview of the Algorithm and suggested optimization

1. Rank 0 creates the data set and scatters it to the P processes.
2. Each process does local sorting.
3. Each process select p-1 splitters.
4. All the p(p-1) splitters are gathered to rank 0.
5. Rank 0 selects a set of p-1 global splitters and sorting the sample of p(p-1) splitters and broadcasted that to P processes.
6. Each process put its local data set into local buckets based on the splitters set.
7. Each process rearranges its buckets using MPI\_Alltoall so that P0 has all the data < splitter[0] and soon.
8. Each process finally sorts their local buckets.

## Experimentation data

### Tabular Representation

**Note: The speedup is calculated using (serial runtime calculated above / runtime for P).**

|  |  |  |
| --- | --- | --- |
| **P** | **Runtime (seconds)** | **Speedup** |
| **N = 120000** | | |
| 2 | .032027 | 1.828349 |
| 6 | .010830 | 5.407017 |
| 12 | .021130 | 2.771294 |
| **N = 1200000** | | |
| 2 | .258872 | 1.573996 |
| 6 | .1213530 | 3.357675 |
| 12 | .074790 | 5.4480497 |
| **N = 12000000** | | |
| 2 | 2.206225 | 1.759161 |
| 6 | .796015 | 4.875664 |
| 12 | .499257 | 7.773751 |
| **N = 120000000** | | |
| 2 | 23.916551 | 1.864886 |
| 6 | 9.023845 | 4.942643 |
| 12 | 5.772662 | 7.726357 |

### Graphical Representation

### 

### Analysis

**Q.** Discuss the results you obtained. Were you able to gain a linear or near-linear speedup? Why or why not?

**Answer.**

#### Discussion on results

1. For N = 120000, the speedup drops down after P = 6.
   * Here the communication cost involved when P > 6 is much greater than the computation gain by sharing the workload among processors and as a result the performance degrades after P = 6.
2. For a given N, the speedup increases with value of P.
   * With more number of P, the workload per process (N/P) decreases leading to better overall performance (or lesser runtime).
3. With very large values of P, the performance curve breaks the linear behavior.
   * Let take the example of N=1200000. Had the curve been linear, then the expected value of speedup at P = 12 is 6.71535, but we got 5.448. Similarly, for N = 12000000, the obtained value of speedup is 7.73 at P = 12 which degrades from the expected value of 9.75 (which should be if the curve is linear). The same is true for N = 120000000.

In the examples mentioned above, the communication cost involved when P = 12 is much greater than the computation gain by sharing the workload among processors and as a result the performance degrades at P = 12.

#### Linear or near linear speedup

* The speedup that we obtain **is near linear.**
  + With initial incremental values of P, the speedup is increasing proportionately as the workload is distributed proportionately among the processes. But with larger values of P, the communication cost outweighs the gain with workload distribution and as a result the speedup curve started degrading at values where the computation/communication ratios are low. (This is substantiated in Analysis 3 above.)

Histogram Sort Using MPI

# Experimentation data for Serial computation

For serial computation we have chosen the serial code with 12 buckets and obtained the runtimes with size = 120000, 1200000, 12000000, 120000000. These results will be used as the baseline to compare with the parallel models.

|  |  |
| --- | --- |
| **(N)** | **Runtime (sec)** |
| 120000 | 0.060661 |
| 1200000 | 0.449810 |
| 12000000 | 4.006512 |
| 120000000 | 45.562630 |

# MPI Implementation

## Experimentation data

### Tabular Representation

**Note: The speedup is calculated using (serial runtime calculated above / runtime for P).**

|  |  |  |
| --- | --- | --- |
| **P** | **Runtime (seconds)** | **Speedup** |
| **N = 120000** | | |
| 2 | .034180 | 1.774720 |
| 6 | .013763 | 4.407349 |
| 12 | .034938 | 1.736207 |
| **N = 1200000** | | |
| 2 | .283378 | 1.587312 |
| 6 | .127599 | 3.525173 |
| 12 | .078755 | 5.711481 |
| **N = 12000000** | | |
| 2 | 2.280247 | 1.757051 |
| 6 | .873000 | 4.589360 |
| 12 | .5860612 | 6.836337 |
| **N = 120000000** | | |
| 2 | 25.159230 | 1.810970 |
| 6 | 9.624384 | 4.734082 |
| 12 | 6.537176 | 6.969772 |

### Graphical Representation

### 

### Analysis

**Q.** Discuss the results you obtained. Were you able to gain a linear or near-linear speedup? Why or why not?

**Answer.**

#### Discussion on results

1. For N = 120000, the speedup drops down after P = 6.
   * Here the communication cost involved when P > 6 is much greater than the computation gain by sharing the workload among processors and as a result the performance degrades after P = 6.
2. For a given N, the speedup increases with value of P.
   * With more number of P, the workload per process (N/P) decreases leading to better overall performance (or lesser runtime).
3. With very large values of P, the performance curve breaks the linear behavior.
   * Let take the example of N=1200000. Had the curve been linear, then the expected value of speedup at P = 12 is 7.05, but we got 5.711481. Similarly, for N = 12000000, the obtained value of speedup is 6.83 at P = 12 which degrades from the expected value of 9.17 (which should be if the curve is linear). The same is true for N = 120000000.

In the examples mentioned above, the communication cost involved when P = 12 is much greater than the computation gain by sharing the workload among processors and as a result the performance degrades at P = 12.

#### Linear or near linear speedup

* The speedup that we obtain **is near linear.**
  + With initial incremental values of P, the speedup is increasing proportionately as the workload is distributed proportionately among the processes. But with larger values of P, the communication cost outweighs the gain with workload distribution and as a result the speedup curve started degrading at values where the computation/communication ratios are low. (This is substantiated in Analysis 3 above.)