Parallel Merge Sort Using MPI

Instructor: Dr. Russ Miller

Prepared by Yihao Liu

University at Buffalo The State University of New York

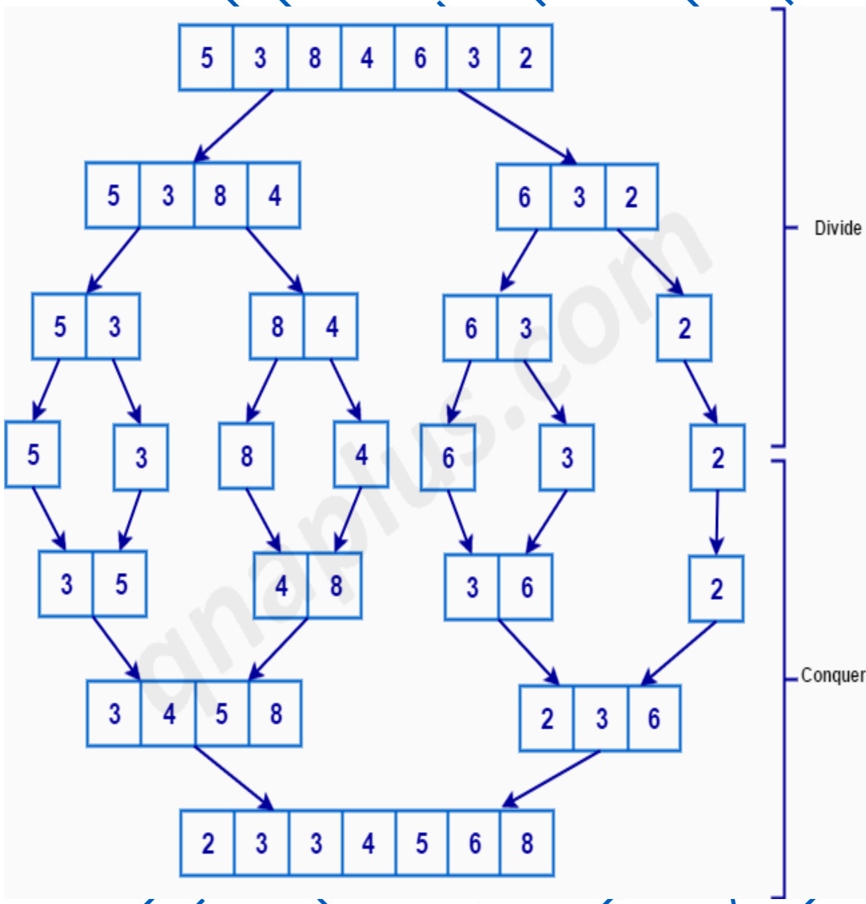
Agenda

- Sequential Merge Sort
- Parallel algorithm
- Experimentation in CCR
- Obtained results and analysis
- Conclusion

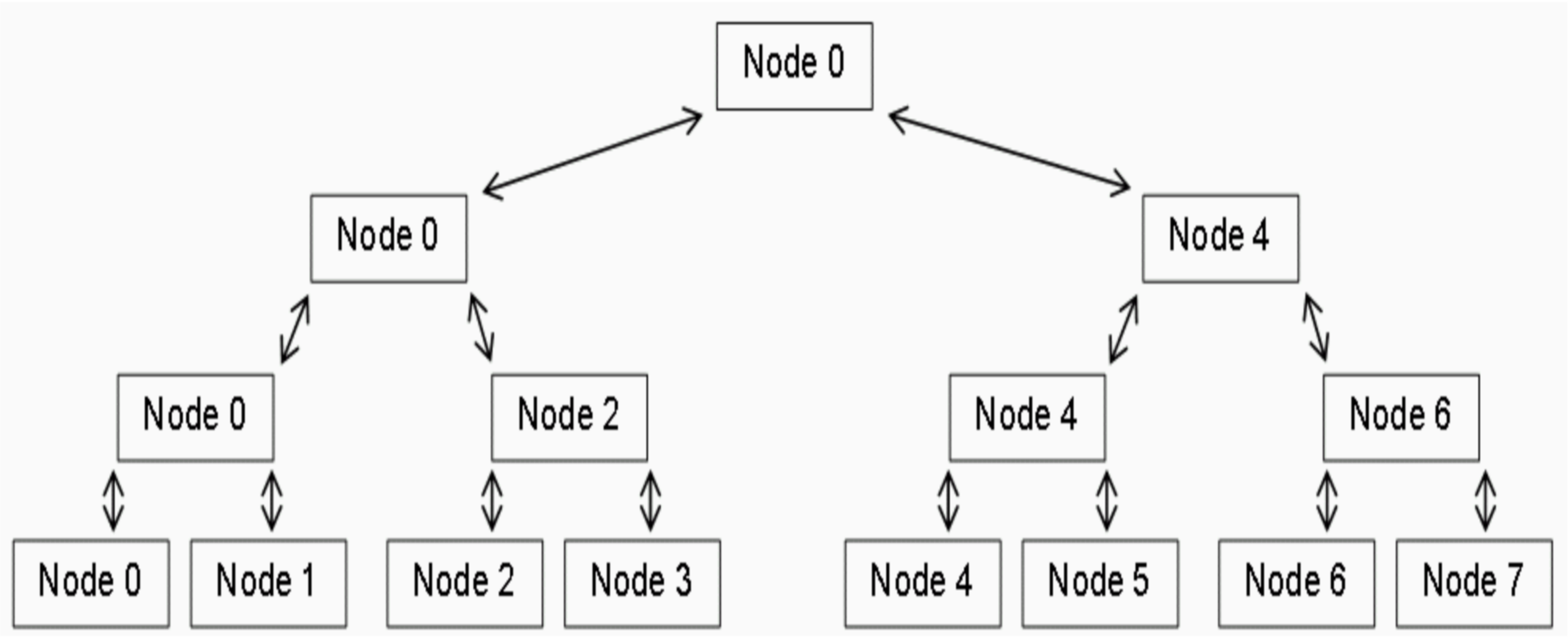
Sequential Merge Sort

mergesort(int[] a, int left, int right)

1. If the input sequence has fewer than two elements, return
2. Partition the input sequence into two halves: mid = (left + right)/ 2
3. Sort the two subsequences using the same algorithm:
mergesort(a, left, mid-1)
mergesort(a,mid,right)
4. Merge the two sorted subsequences to form the output sequence



Parallel merge sort



Parallel Mergesort Algorithm

1. Node with rank 0 is the host node. It computes the height of the node and get the entire dataset

2. Node 0 initiates the parallel merge operation

3. For internal nodes (height > 0), including node 0. Divide the data in half and send the right half to the right child as computed in previous slide. Recursively call parallel merge operation for the left half on the same node. Also, receive the sorted data from right child. Merge the sorted left and right halves.

4. If it is a leaf node, just do internal sorting

5. Send the sorted data to parent node

6. Finally, node 0 will have the entire sorted result

Experimentation in CCR: SBATCH script

```
[yihaoliu@vortex1:~]$ cat job.sh
#!/bin/bash
#SBATCH --qos=general-compute
#SBATCH --cluster=ub-hpc
#SBATCH --time=0:10:00
#SBATCH --partition=general-compute
#SBATCH --nodes=4
#SBATCH --constraint=CPU-E5645
#SBATCH --cpus-per-task=1
#SBATCH --tasks-per-node=1
#SBATCH --mem-per-cpu=4000
#SBATCH --mail-user=yihaoliu@buffalo.edu
#SBATCH --mail-type=END
#SBATCH --job-name=mpi_hello
#SBATCH --output=hello.out
#SBATCH --error=hello.err
#SBATCH --exclusive
```

mpirun –np 4 ./merge Arraysize

Result

1. For some data size, plot processing time vs number of nodes

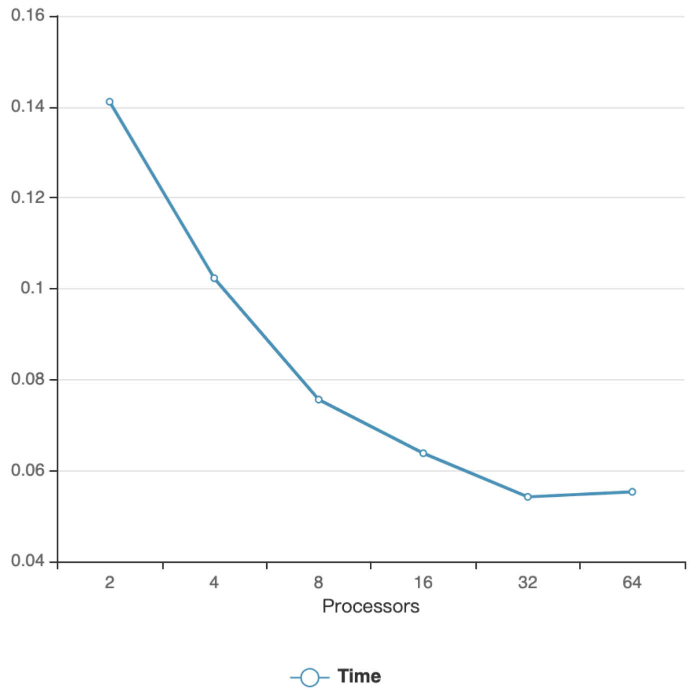
a. Tested on 3 different data sizes: 1M, 10M, 1 billion

b. Number of nodes: 2, 4, 8, 16, 32, 64

3. Plot graphs that depict for a particular number of processor and show how the runtime is affected with data size

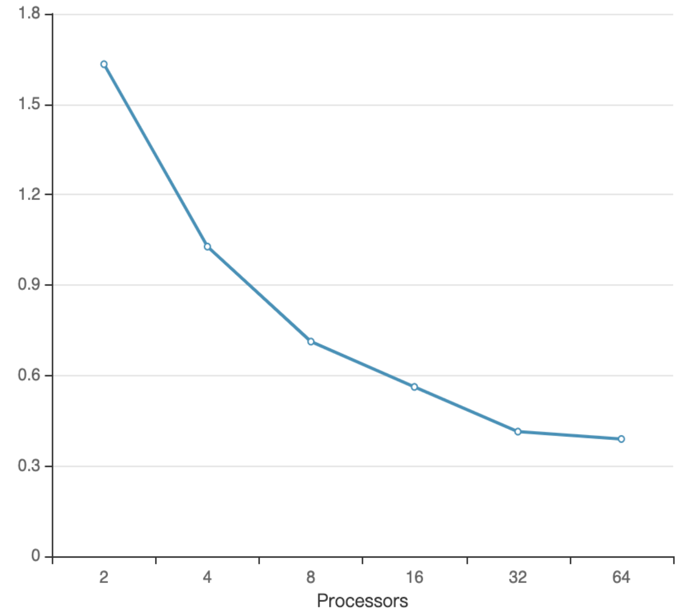
Runtime Vs Number of nodes for N = 1 million

Data size : 1 million	
Processors	Time
2	0.1411
4	0.1023
8	0.0756
16	0.0638
32	0.0542
64	0.0553



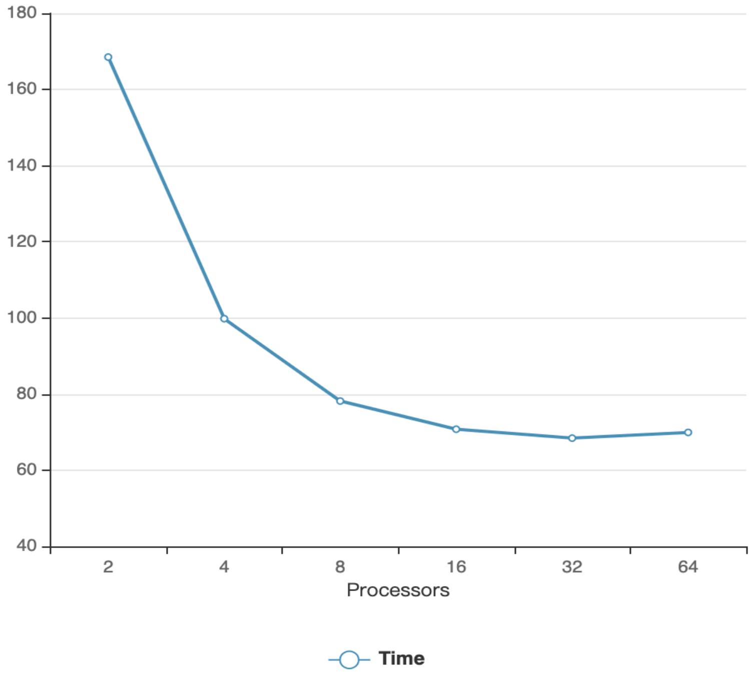
Runtime Vs Number of nodes for N = 10 million

Data size : 10 million	
Processors	Time
2	1.6326
4	1.0274
8	0.7123
16	0.5619
32	0.4136
64	0.3891



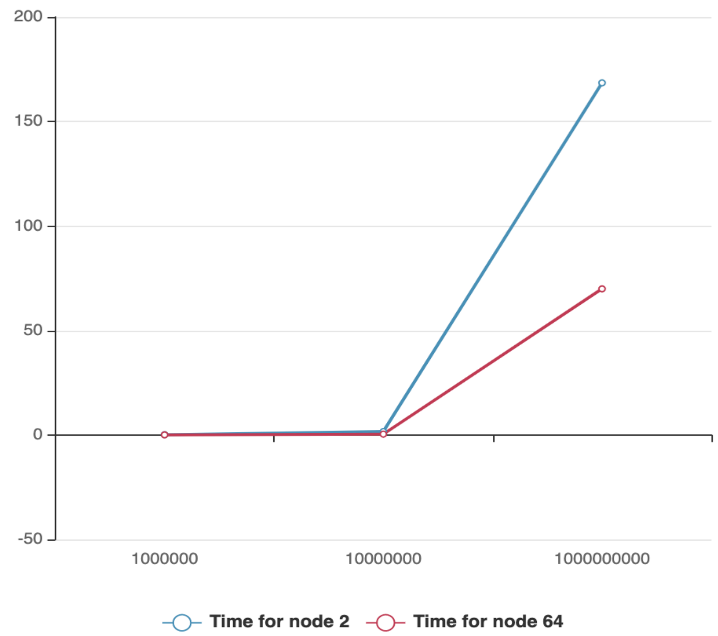
Runtime Vs Number of nodes for N = 1 billion

Data size : 1 billion	
Processors	Time
2	168.5064
4	95.7923
8	78.1659
16	70.7653
32	68.4249
64	69.9256



Runtime Vs Data size for P = 2 and 64

Data size	Time for node 2	Time for node 64
1000000	0.1411	0.0553
10000000	1.6326	0.3891
1000000000	168.5064	69.9256



Conclusion

1) One task was associated with one node. Thus, every physical server initiated one process only.

2) According to the results, parallelization can be efficient to a particular number of processors/nodes and reduce the time of sorting. In some

3) However, as the number of nodes increases, the cost from the communication between the nodes will also increase. Therefore, the efficiency will be hampered.

Thanks