Parallel Merge Sort
Using MPI

Instructor: Dr. Russ Miller

Prepared by Yihao Liu
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

```bash
[yinhaoliu@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=yinhaoliu@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**

<table>
<thead>
<tr>
<th>Processors</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.1411</td>
</tr>
<tr>
<td>4</td>
<td>0.1023</td>
</tr>
<tr>
<td>8</td>
<td>0.0756</td>
</tr>
<tr>
<td>16</td>
<td>0.0638</td>
</tr>
<tr>
<td>32</td>
<td>0.0542</td>
</tr>
<tr>
<td>64</td>
<td>0.0553</td>
</tr>
</tbody>
</table>
## Runtime Vs Number of nodes for N = 10 million

<table>
<thead>
<tr>
<th>Processors</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1.6326</td>
</tr>
<tr>
<td>4</td>
<td>1.0274</td>
</tr>
<tr>
<td>8</td>
<td>0.7123</td>
</tr>
<tr>
<td>16</td>
<td>0.5619</td>
</tr>
<tr>
<td>32</td>
<td>0.4136</td>
</tr>
<tr>
<td>64</td>
<td>0.3891</td>
</tr>
</tbody>
</table>

![Runtime Vs Number of nodes for N = 10 million](chart.png)
Runtime Vs Number of nodes for N = 1 billion

Data size : 1 billion

<table>
<thead>
<tr>
<th>Processors</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>168.5064</td>
</tr>
<tr>
<td>4</td>
<td>95.7923</td>
</tr>
<tr>
<td>8</td>
<td>78.1659</td>
</tr>
<tr>
<td>16</td>
<td>70.7653</td>
</tr>
<tr>
<td>32</td>
<td>68.4249</td>
</tr>
<tr>
<td>64</td>
<td>69.9256</td>
</tr>
</tbody>
</table>
## Runtime Vs Data size for P = 2 and 64

<table>
<thead>
<tr>
<th>Data size</th>
<th>Time for node 2</th>
<th>Time for node 64</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000000</td>
<td>0.1411</td>
<td>0.0553</td>
</tr>
<tr>
<td>10000000</td>
<td>1.6326</td>
<td>0.3891</td>
</tr>
<tr>
<td>1000000000</td>
<td>168.5064</td>
<td>69.9256</td>
</tr>
</tbody>
</table>

![Graph showing runtime vs data size](image-url)
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