PARALLEL COMPUTING
OF MAXIMUM SUM SUB-ARRAY
Using MPI
Instructor: Dr. Russ Miller
By,
Aditya Subramanian Muralidaran
Contents

• Objective
• Problem Statement
• Algorithm
• Slurm Script
• Analysis
• Observation
Objective

• To experience the power of parallel computing using MPI by implementing a parallel prefix operation.

Problem Statement

• Application of parallel prefix: Identifying the maximum sum that can be computed using contiguous elements in an array.
• Eg, Consider the Array = {-2, 1, -3, 4, -1, 2, 1, -5, 4}
• Maximum sum = 6
Parallel Prefix Algorithm

- Step 1: Perform a parallel prefix sum operation.
  
  Eg:
  
  \[
  \text{Array} = \{-2, 1, -3, 4, -1, 2, 1, -5, 4\} \Rightarrow \{-2, -1, -4, 0, -1, 1, 2, -3, 1\}
  \]

  \text{Parallel Prefix Sum}

- Step 2: Perform a parallel postfix max operation on the resultant array
  
  \[
  \text{Max Array} = \{-2, -1, -4, 0, -1, 1, 2, -3, 1\} \Rightarrow \{2, 2, 2, 2, 2, 2, 2, 1, 1\}
  \]

  \text{Parallel Postfix Max}
Parallel Prefix Algorithm

- Step 3: Compute the following formula in parallel for every element,
  \[ \text{Max}_\text{Array}[i] - \text{Sum}[i] + \text{Array}[i] \]
  \{2, 4, 3, 6, 2, 3, 1, -1, 4\}

- The maximum element in this array will be broadcasted to every processor.
Let the problem size be “n”.

Let the number of processors we have be “p”.

What if $n \gg p$ (very much greater)?

✓ We can divide the problem so that each processor get a chunk of data of size $n/p$.

✓ Consider,

Array = {-2, 1, -3, 4, -1, 2, 1, -5, 4, . . . . .} size = n.

| Size n/p | Size n/p | Size n/p | .......|

✓ Every processor will be responsible for a single chunk and will perform the parallel prefix/postfix operation in sequential manner within the $n/p$ chunk of data.
Parallel Prefix Sum Simulation

-2, 1, -3 } {4, -1, 2} {1, -5, 4}

-2, -1, -4 } {4, 3, 5} {1, -4, 0}

-2, -1, -4 } {4, 3, 1} {1, -4, 1}

-2, -1, -4 } {0, -1, 1} {2, -3, 1} => Solution.
#!/bin/sh
#SBATCH --nodes=16
#SBATCH --ntasks-per-node=1
#SBATCH --constraint=IB
#SBATCH --partition=general-compute --qos=general-compute
#SBATCH --time=12:00:00
#SBATCH --mail-type=END
#SBATCH --mail-user=adityasau@buffalo.edu
#SBATCH --output=64m_output_16.out
#SBATCH --job-name=TestingMaxSum64m_mpi16
module load intel/14.0
module load intel-mpi/4.1.3
module list
mpicc -lm -o obj1 max_subarray.c
ulimit -s unlimited
export I_MPI_PMI_LIBRARY=/usr/lib64/libpmi.so
mpicc -lm -o obj1 max_subarray.c
srun ./obj1 4000000
#
echo "All done!"
Analysis – Constant ‘n’ & Variable Node ‘p’

- Problem Size – 64,000,000 (64 Million)
- Sequential running time (in milliseconds) : 2780.9118 ms

<table>
<thead>
<tr>
<th>Number of Processors</th>
<th>Run Time (in milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1108.829 ms</td>
</tr>
<tr>
<td>8</td>
<td>629.524 ms</td>
</tr>
<tr>
<td>16</td>
<td>301.566 ms</td>
</tr>
<tr>
<td>32</td>
<td>186.051 ms</td>
</tr>
<tr>
<td>64</td>
<td>83.160 ms</td>
</tr>
<tr>
<td>128</td>
<td>48.400 ms</td>
</tr>
<tr>
<td>256</td>
<td>27.883 ms</td>
</tr>
</tbody>
</table>
Analysis – Constant ‘n’ & Variable Node ‘p’

RUN TIME (IN MILLISECONDS)
N = 64 MILLION

Number of Processors

Run Time (ms)
Analysis - Different Problem Size

ANALYSIS DIFFERENT PROBLEM SIZE

Run Time (ms)

- $n = 640,000$
- $n = 64,000$
- $n = 6,400,000$
Analysis – Increasing Running Time

RUN TIME (IN MILLISECONDS)
N = 64,000

Number of Processors

Run Time (ms)

1, 3.6047
4, 2.6064
8, 2.9211
16, 3.4263
32, 6.1203
64, 6.0324

0 10 20 30 40 50 60 70
0 1 2 3 4 5 6 7 8 9
Observation

• Using more processors decreases the run time of an algorithm (Not in all cases!).

• For smaller problem size, \((n = 64,000)\), lower number of processors lead to better performance. Why? – Because the time taken to communicate between the nodes is more than the time taken to run the actual algorithm.

• This invalidates the assumption that “Throw in more processors for better performance”. For any problem of size \(‘n’\), after a certain number of processors \(‘p’\), the run time of the algorithm begins to increase due to the overhead of communication between the processors as stated above.
THANK YOU