PARALLEL MATRIX MULTIPLICATION

Presented by: Prithvisagar Rao
Problem Statement

Given a matrix $A(n \times n)$ and a matrix $B(n \times n)$, the matrix $C$ resulting from the operation of multiplication of matrices $A$ and $B$, $C = A \times B$ is given as:

$$c_{i,j} = \sum_{k=1}^{n} a_{ik} \times b_{kj}$$
To calculate one value in matrix C we need to perform \( n \) multiplications and \( n-1 \) additions. For a matrix of size \( n^2 \) this results in \( n^3 \) calculations.
Sequential Algorithm

```c
for (i=0; i<n; i++) {
    for (j=0; j<n; j++) {
        c[i][j] = 0;
        for (k=0; k<n; k++) {
            c[i][j] = c[i][j] + a[i][k] * b[k][j];
        }
    }
}
```

As we can see, the sequential algorithm has 3 nested for loops which results in a $O(n^3)$ time complexity.
Parallel Algorithm

Parallel Algorithm for Matrix Multiplication

1. Partition $A$ and $B$ into P square blocks $A_{i,j}$ and $B_{i,j}$ where P is the number of processors available.

2. Ensure each process can maintain a block of $A$ and $B$ by creating a matrix of processes of size $P^{1/2} \times P^{1/2}$

3. The blocks are multiplied together and the results are added to the partial results in the C sub-blocks.

4. The sub-blocks of $A$ are shifted one step to the left and the sub-blocks of $B$ are shifted one step up.

5. Repeat this process for $P^{1/2}$ times
Parallel Algorithm

Divide the initial input matrix into P sub blocks and distribute the data to their processes.

Input matrix A

\[
\begin{align*}
A_1 & & A_2 \\ A_6 & & A_6 \\ A_9 & & A_{10} \\ A_{13} & & A_{14} \\ A_7 & & A_8 \\ A_{15} & & A_{16}
\end{align*}
\]

Input matrix B

\[
\begin{align*}
B_1 & & B_2 \\ B_5 & & B_6 \\ B_9 & & B_{10} \\ B_{13} & & B_{14} \\ B_3 & & B_4 \\ B_7 & & B_8 \\ B_11 & & B_{12} \\ B_{15} & & B_{16}
\end{align*}
\]
Parallel Algorithm

The processors perform the local multiplication based on the initial arrangement.
Parallel Algorithm

Shift matrix A to the left and matrix B upwards, perform the local multiplication and add it to the partial result
Parallel Algorithm

Add the partial answers
Results

Parameters used for running the parallel approach:

- Square matrices were used
- Matrix dimensions ranged from 2000 to 8000
- Number of processors used – 4, 9, 16, 25, 36, 49, 64
## Parallel Approach

<table>
<thead>
<tr>
<th>No of Processors</th>
<th>2000</th>
<th>3000</th>
<th>4000</th>
<th>5000</th>
<th>6000</th>
<th>7000</th>
<th>8000</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>7.963</td>
<td>36.769</td>
<td>97.568</td>
<td>269.82</td>
<td>394.163</td>
<td>779.479</td>
<td>1221.523</td>
</tr>
<tr>
<td>16</td>
<td>2.302</td>
<td>12.235</td>
<td>24.653</td>
<td>75.427</td>
<td>113.422</td>
<td>226.991</td>
<td>264.74</td>
</tr>
<tr>
<td>25</td>
<td>1.426</td>
<td>6.147</td>
<td>15.657</td>
<td>33.853</td>
<td>57.588</td>
<td>94.843</td>
<td>175.485</td>
</tr>
<tr>
<td>36</td>
<td>3.016</td>
<td>10.388</td>
<td>29.480</td>
<td>54.572</td>
<td>59.495</td>
<td>101.63</td>
<td>158.162</td>
</tr>
<tr>
<td>49</td>
<td>4.167</td>
<td>12.547</td>
<td>38.996</td>
<td>60.971</td>
<td>63.689</td>
<td>91.627</td>
<td>133.747</td>
</tr>
<tr>
<td>64</td>
<td>5.753</td>
<td>16.764</td>
<td>46.504</td>
<td>67.842</td>
<td>69.689</td>
<td>89.785</td>
<td>115.621</td>
</tr>
</tbody>
</table>
Observations

• Computation running time decreased as a result of parallelization.
• Increase in number of processors does not necessarily result in reduction in running time due to communication overhead.
• A good balance between number of processors and runtime was observed at 25 number of processors.
• Number of processors must be perfect squares.
• Data must be equally distributed among the processors.
• Got a good idea of parallelization.
Future work

• Ran the simple block matrix multiplication in parallel. Other algorithms such as Block-striped algorithm and Fox’s algorithm can be run and compared.

• Compare results with OpenMP implementation.
Thank you