PARALLELIZED
CONVOLUTION
Convolution

- Convolution is a mathematical operation on two functions.
- A function derived from two given functions by integration that expresses how the shape of one is modified by the other.
- The Mathematical expression for basic two dimensional convolution is

\[(f * g)(t) \overset{\text{def}}{=} \int_{-\infty}^{\infty} f(\tau) g(t - \tau) \, d\tau\]
Convolution
Applications

• Image Processing
• Electrical Engineering (Communication signal processing)
• Statistics
• Differential equations
Matrix Convolution

\[ O_{57} = I_{57}K_{11} + I_{58}K_{12} + I_{59}K_{13} + I_{67}K_{21} + I_{68}K_{22} + I_{69}K_{23} \]

\[ O(i, j) = \sum_{k=1}^{m} \sum_{l=1}^{n} I(i + k - 1, j + l - 1)K(k, l) \]
Sequential solution

- The kernel matrix is padded over the input matrix and the overlapping pixels are computed and this operation is continued for all the pixels of input matrix.
- The complexity is similar to matrix multiplication where the computation involves huge no of multiplications and additions.
Parallel solution

• Input Matrix is stored in a single node
• Kernel matrix is broadcasted to all the other processors
• Input Matrix node distributes chunks of input matrix to all the other processors
• All the processors sends the partially computed result to a single final node
• Because of independent convolutions, distributed parallelism can be implemented
The Input size

• 1024 x 1024 image – input
• 90 x 90 kernel
Running Time

No of Processors vs Running Time (seconds)

Running Time vs No of Processors

Running Time (90x90)
No of Processors vs Speedup
Running Time

No of Processors vs Running Time

- Running Time (90x90)
- Running Time (80x80)
## Running Time

<table>
<thead>
<tr>
<th>No of Processors allotted</th>
<th>Processors Requested</th>
<th>Running Time (seconds)</th>
<th>Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>80</td>
<td>160</td>
<td>92.12</td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>135</td>
<td>87.34</td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>115</td>
<td>83.98</td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>90</td>
<td>79.63</td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>80</td>
<td>76.38</td>
<td>Best Running Time</td>
</tr>
<tr>
<td>52</td>
<td>96</td>
<td>90.54</td>
<td></td>
</tr>
<tr>
<td>52</td>
<td>84</td>
<td>87.91</td>
<td></td>
</tr>
<tr>
<td>52</td>
<td>75</td>
<td>82.25</td>
<td></td>
</tr>
<tr>
<td>52</td>
<td>63</td>
<td>76.54</td>
<td></td>
</tr>
<tr>
<td>52</td>
<td>52</td>
<td>73.68</td>
<td>Best Running Time</td>
</tr>
</tbody>
</table>
Running Time
Choosing optimum N from the graph?

$xy = \text{constant}$

$x + y = \text{minimum}$

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>64</td>
<td>64</td>
</tr>
<tr>
<td>2</td>
<td>32</td>
<td>64</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>64</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>64</td>
</tr>
<tr>
<td>16</td>
<td>4</td>
<td>64</td>
</tr>
<tr>
<td>32</td>
<td>2</td>
<td>64</td>
</tr>
<tr>
<td>64</td>
<td>1</td>
<td>64</td>
</tr>
</tbody>
</table>

8+8=16 - which is the minimum among all x+y combinations. So choosing N (no of processors) at this point will give fairly best cost for a given input.
Optimum N?

• The optimum no of nodes for the approximate no of multiplications can be calculated.
• example: 1024 x 1024 image – input
  90 x 90 kernel
- 1024 x 1024 x 90 x 90 = ~ 8 billion operations
Difficulties

• Communication overhead
• Large no of multiplications
  - (~8 billion Multiplications and additions) for 90 x 90 kernel
  - (~6.5 billion Multiplications and additions) for 80 x 80 kernel
• Filtering the input matrix
References

Questions ?