PARALLELIZED CONVOLUTION

–Venkata Nanda Kishore Buddhiraju
3764–8659
CSE 633
Fall, 2011
Convolution

- Extent of overlap of 2 functions.
- Used to smooth the images and other functions.
- Also, causes Blurring.
Example: Convolution of an image at various levels

Original Image

Top left: level 1, Top right: level 2
Bottom left: level 3, Bottom right: level 4
Technically: Let $O$ be convolution matrix, $I$ is Image matrix

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

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

eg. $2048 \times 1536 = 3,145,728$ pixels or 3.1 megapixels
Difficulties

- Have to perform padding
- Huge number of multiplications
- Becomes much larger with Kernel size
Solution: Parallelize!

- Parallelism should reduce the time to compute
- Different approaches available
- Master node has the Input matrix and the kernel
- Master node decides the extent of padding and pads the input matrix
Parallelize(2)

- Master node then distributes chunks of input matrix to other nodes
- These nodes/processors compute local convolution and sends them to the Master node
- Master, then fixes the output matrix together
- Because of independent convolutions, distributed parallelism can be implemented
But–

- Communication between the nodes to align the right borders of local convolutions to facilitate merging
- This is communication overhead
- Or, append border pixels to the chunk of input matrix pixels for adjacent nodes
- This is redundancy of data!
Focus

- Levels of convolution
- Will be on the no. of chunks
- Time taken to compute
- Overhead vs. Redundancy
- Will be implementing on C and then using MPI
Implementation

Master Process
Rank=0

Child process 1

Child process n/2

Child process 2

Child process n
Problem Domain

- 1200 x 1200 image
- 80 x 80 kernel
- Around 8 billion Multiplication operations
- And a similar no. of Additions
- No. of Processors subscribed: 2 to 90

Note: The test cases mentioned here are changed after the presentation in the class, for a better throughput.
Results: Parallel Convolution

Parallelized Convolution

- Time in seconds vs. no. of processors
- Graph shows a decrease in time with an increase in the number of processors.
Speed Up vs. No. of Processors
Change of Kernel Size

![Graph showing the change of kernel size with varying number of processes. The graph indicates a decrease in time in seconds as the number of processes increases. Two lines represent different kernel sizes (85x85 and 90x90), showing distinct trends.]
## Effects of Under subscription
(level 2 convolution)

<table>
<thead>
<tr>
<th>Max. No. of processors (‘x’) (subscribed)</th>
<th>Processors requested(‘y’)</th>
<th>Time taken for ‘y’ processes on ‘x’ processors</th>
<th>Original time for ‘y’ processes on ‘y’ processors</th>
</tr>
</thead>
<tbody>
<tr>
<td>72</td>
<td>144</td>
<td>88.451</td>
<td>71.046</td>
</tr>
<tr>
<td>72</td>
<td>115</td>
<td>82.869</td>
<td>70.847</td>
</tr>
<tr>
<td>72</td>
<td>100</td>
<td>81.734</td>
<td>73.646</td>
</tr>
<tr>
<td>72</td>
<td>90</td>
<td>76.170</td>
<td>71.783</td>
</tr>
<tr>
<td>72</td>
<td>88</td>
<td>75.146</td>
<td>71.978</td>
</tr>
<tr>
<td>48</td>
<td>80</td>
<td>87.732</td>
<td>73.568</td>
</tr>
<tr>
<td>48</td>
<td>72</td>
<td>81.088</td>
<td>70.512</td>
</tr>
<tr>
<td>48</td>
<td>60</td>
<td>78.836</td>
<td>73.291</td>
</tr>
<tr>
<td>48</td>
<td>55</td>
<td>77.356</td>
<td>73.779</td>
</tr>
</tbody>
</table>
Observations

- Dealing with master process was tricky. It either is slacking off or overloaded.
- Significant increase in the size of data did not result in a proportionate increase in time taken. Only a small factor of increase in time was observed.
- Under subscribing enabled the usage of the processors with more efficiency.
Was I able to distribute the data equally among sub-processes?
Answer: No. It is not always possible to divide the data equally. I made the master process compute convolutions for these outliers.

Wont the sub-processes have a copy of the matrix as it is initialized in the program itself? If so, why distribute the data again?
Answer: Assuming that in a distributed environment, data is not known prior to the computation, I re-distributed the data through master process to other processes.
Future Works

- Implementation of separable convolution
- Implementing the same work on GPUs
- Implementing on OpenMP to see the execution time patterns on shared memory
Acknowledgements

- Dr. Russ Miller
- Dr. Matt Jones
- Cynthia D. Cornelius

References:

http://www.ccr.buffalo.edu/download/attachments/65681/Mpi-advanced-handout-2x2.pdf?version=2

http://www.ccr.buffalo.edu/download/attachments/65681/Mpi-intermed-handout-2x2.pdf?version=2

http://www.scribd.com/doc/58013724/10-MPI-Programmes

THANK YOU !!