Line Segment Visibility from Origin

-Uthish Balaji
CSE-633 Parallel Algorithms
Wednesday 23rd April 2014
Problem Definition

- Given a set of line segments which are perpendicular to the x axis, determine what piece of each line segment would be visible from origin.
- Input: A set of y coordinate values
- Output: A set of results which contain the piece of each line segment that is visible
Illustration
Assumptions

- One end of the line segment is at the x-axis.
- The lines are placed unit distance apart in x-axis.
- The data is arranged in ascending order with respect to the x axis values.
- One cannot see through the lines
Applications

- When creating a virtual reality, one would want the users to not be able to look through walls/buildings.
- Determining the length of a runway when building an airport.
What I’ve done

- Developed a serial algorithm
- Developed a parallel algorithm
- Compared them.
Serial Algorithm

- Read the data.
- \( \text{Currentslope} = \frac{y[0]}{x[0]} \)
- For \( i = 1 \) to \( n \)
  - If \( y[i] > \text{Currentslope} \times x[i] \)
    - \( \text{out}[i] = y[i] - \text{Currentslope} \times x[i] \)
    - \( \text{Currentslope} = \frac{y[i]}{x[i]} \)
  - Else
    - \( \text{out}[i] = 0; \)
- Print \( \text{out}[i] \).
Parallel Solution

- Parallel prefix on the maximum slope.
- Perform the serial algorithm to get the results within the particular processor.
- Pass on the maximum slope seen within the processor to the next processor.
- If the value obtained is greater than maximum local slope then
  pass on the value obtained to the next processor.
- Else pass on maximum local slope to next processor
  recalculate the local values with the value obtained.
Implementation - Input Data problem
Implementation - generation of input data

- Created dataset using random function.
- prevslope=0
- If rand()%4==3
  
  \[ \text{arr}[i]=\text{prevslope} \times i + 0.001 \]
  
  prevslope = arr[i]/i;

- Else arr[i]=0
Implementation - distribution of data

- Processor with rank=0 creates the dataset
- Rank0 processor then distributes the entire dataset to all other processors
- Since all processors know n(size of data) and p(number of processors) they’ll calculate the lower limit and upper limit of the entire data they are supposed to read
  - Lower limit = ((n/p)*rank)+1
  - Upper limit = (n/p)*(rank+1)
## Running Time Serial Implementation

<table>
<thead>
<tr>
<th>10,000</th>
<th>0.000131</th>
</tr>
</thead>
<tbody>
<tr>
<td>100,000</td>
<td>0.001348</td>
</tr>
<tr>
<td>1,000,000</td>
<td>0.013202</td>
</tr>
<tr>
<td>10,000,000</td>
<td>0.129934</td>
</tr>
<tr>
<td>100,000,000</td>
<td>1.300618</td>
</tr>
</tbody>
</table>
## Running Time - Parallel Implementation

<table>
<thead>
<tr>
<th></th>
<th>10,000</th>
<th>100,000</th>
<th>1,000,000</th>
<th>10,000,000</th>
<th>100,000,000</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.000125</td>
<td>0.001257</td>
<td>0.012603</td>
<td>0.09296</td>
<td>0.300813</td>
</tr>
<tr>
<td>4</td>
<td>0.000105</td>
<td>0.000663</td>
<td>0.006354</td>
<td>0.031587</td>
<td>0.275691</td>
</tr>
<tr>
<td>8</td>
<td>0.000149</td>
<td>0.000436</td>
<td>0.003248</td>
<td>0.016084</td>
<td>0.135961</td>
</tr>
<tr>
<td>16</td>
<td>0.000303</td>
<td>0.00049</td>
<td>0.002037</td>
<td>0.008422</td>
<td>0.074135</td>
</tr>
<tr>
<td>32</td>
<td>0.000617</td>
<td>0.000742</td>
<td>0.00176</td>
<td>0.005049</td>
<td>0.040772</td>
</tr>
<tr>
<td>64</td>
<td>0.001349</td>
<td>0.001536</td>
<td>0.001824</td>
<td>0.003948</td>
<td>0.021979</td>
</tr>
<tr>
<td>128</td>
<td>0.003168</td>
<td>0.002987</td>
<td>0.002502</td>
<td>0.002737</td>
<td>0.01576</td>
</tr>
</tbody>
</table>
Results - Serial Implementation

Serial Implementation

1=10k 2=100k 3=1m 4=10m 5=100m
Result - parallel implementation

Data = 10,000

Data = 100,000
Results - parallel implementation

Data = 1,000,000

Data = 10,000,000
Results - parallel implementation

Data = 100,000,000
Best Case vs Average/Worst Case

- Best case scenario: In the best scenario, the very first line would be visible and nothing beyond it will be. This is best case because all the local prefixes will be skipped because the local maximum slope will be less than the received slope.

- Average case: In the average case, there may or may not be a local maximum slope which is greater than the received slope.

- Worst case scenario: In the worst case scenario the local maximum is always greater than the prefix value that was received, so thus all local prefix calculations have to be done.
Best vs Average/worst case scenario

Comparison between best case and average case
Speedup (10m data)
Questions ?