Odd-Even Transposition Sort

Ned Nedialkov

McMaster University
Canada

CS/SE 4F03
February 2015
Outline

Bubble sort

Example: odd-even transposition sort

Example

Algorithm outline

Analysis
Bubble sort

Algorithm

Input: an array $A$ of size $n$
Output: sorted array $A$
Compute:
  for $i = n - 1$ down to 1
    for $j = 1 : i$
      compare-exchange($A[j], A[j + 1]$)

How to parallelize it?
### Example: odd-even transposition sort

#### Algorithm outline

<table>
<thead>
<tr>
<th>phase</th>
<th>2</th>
<th>10</th>
<th>5</th>
<th>3</th>
<th>7</th>
<th>9</th>
<th>4</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. odd</td>
<td>2</td>
<td>↔</td>
<td>10</td>
<td></td>
<td></td>
<td>5</td>
<td>↔</td>
<td>3</td>
</tr>
<tr>
<td>2. even</td>
<td>2</td>
<td></td>
<td>10</td>
<td>↔</td>
<td>3</td>
<td></td>
<td>5</td>
<td>↔</td>
</tr>
<tr>
<td>3. odd</td>
<td>2</td>
<td>↔</td>
<td>3</td>
<td></td>
<td>10</td>
<td>↔</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>4. even</td>
<td>2</td>
<td></td>
<td>3</td>
<td>↔</td>
<td>5</td>
<td></td>
<td>10</td>
<td>↔</td>
</tr>
<tr>
<td>5. odd</td>
<td>2</td>
<td>↔</td>
<td>3</td>
<td></td>
<td>5</td>
<td>↔</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>6. even</td>
<td>2</td>
<td></td>
<td>3</td>
<td>↔</td>
<td>4</td>
<td></td>
<td>5</td>
<td>↔</td>
</tr>
<tr>
<td>7. odd</td>
<td>2</td>
<td>↔</td>
<td>3</td>
<td></td>
<td>4</td>
<td>↔</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>8. even</td>
<td>2</td>
<td></td>
<td>3</td>
<td>↔</td>
<td>4</td>
<td></td>
<td>5</td>
<td>↔</td>
</tr>
</tbody>
</table>

#### Analysis

After $n$ phases the sequence is guaranteed to be sorted
## Example

Assume $n$ numbers and $p$ processes

Here $n = 16$, $p = 4$

<table>
<thead>
<tr>
<th></th>
<th>$P_0$</th>
<th>$P_1$</th>
<th>$P_2$</th>
<th>$P_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>input</td>
<td>35, 29, 42, 84</td>
<td>87, 28, 95, 79</td>
<td>54, 92, 64, 34</td>
<td>53, 14, 69, 70</td>
</tr>
<tr>
<td>local sort</td>
<td>29, 35, 42, 84</td>
<td>28, 79, 87, 95</td>
<td>34, 54, 64, 92</td>
<td>14, 53, 69, 70</td>
</tr>
<tr>
<td>odd phase</td>
<td>29, 35, 42, 84 ↔ 28, 79, 87, 95</td>
<td>34, 54, 64, 92 ↔ 14, 53, 69, 70</td>
<td></td>
<td></td>
</tr>
<tr>
<td>even phase</td>
<td>28, 29, 35, 42</td>
<td>79, 84, 87, 95 ↔ 14, 34, 53, 54</td>
<td>64, 69, 70, 92</td>
<td></td>
</tr>
<tr>
<td>odd phase</td>
<td>28, 29, 35, 42 ↔ 14, 34, 53, 54</td>
<td>79, 84, 87, 95 ↔ 64, 69, 70, 92</td>
<td></td>
<td></td>
</tr>
<tr>
<td>even phase</td>
<td>14, 34, 28, 29</td>
<td>35, 42, 53, 54 ↔ 64, 69, 70, 79</td>
<td>84, 87, 92, 95</td>
<td></td>
</tr>
<tr>
<td>sorted</td>
<td>14, 28, 29, 34</td>
<td>35, 42, 53, 54</td>
<td>64, 69, 70, 79</td>
<td>84, 87, 92, 95</td>
</tr>
</tbody>
</table>

After $p$ phases the sequence is guaranteed to be sorted
Algorithm outline

- Each process sorts its $n/p$ numbers
- Perform $p$ passes of odd-even interchanges
- After them, each process has its own sorted sequence

See e.g. http://en.wikipedia.org/wiki/Odd%E2%80%93even_sort#Algorithm
Analysis

Each process can sort in $O(n/p \log n/p)$

At each phase (odd/even)

- the communication is $O(n/p)$
- merging of two $n/p$ sequences is $O(n/p)$

Since we have $p$ phases, the total work for communication and merging is $O(n)$

Hence the parallel time is

$$T_p = O \left( \frac{n}{p} \log \frac{n}{p} \right) + O(n)$$

The serial time is (for the fastest sort)

$$T_s = O(n \log n)$$
\[ S = \frac{T_s}{T_p} = \frac{O(n \log n)}{O\left(\frac{n}{p} \log \frac{n}{p}\right) + O(n)} = \frac{1}{O\left(\frac{n}{p} \log \frac{n}{p}\right) + O\left(\frac{n}{n \log n}\right)} \]

\[ = \frac{1}{O\left(\frac{1}{p} \log \frac{n}{p}\right) + O\left(\frac{1}{\log n}\right)} \]

\[ E = \frac{1}{O\left(\log \frac{n}{p} \log n\right) + O\left(\frac{p}{\log n}\right)} = \frac{1}{O\left(1 - \frac{\log p}{\log n}\right) + O\left(\frac{p}{\log n}\right)} \]

\[ = \frac{1}{1 - O\left(\frac{\log p}{\log n}\right) + O\left(\frac{p}{\log n}\right)} \]

How would this algorithm scale?