Parallel Quicksort

Ned Nedialkov

McMaster University
Canada

CS/SE 4F03
March 2016
Outline

Quick sort

Performance

Parallel formulation

Example

Pivot selection

Combining blocks

MPI version
Quick sort

Algorithm

Quicksort$(A, p, r)$
if $p \geq r$

\[ x = A[r], \quad i = p - 1 \]

for $j = p$ to $r - 1$

\[ \text{if } A[j] \leq x \]

\[ i = i + 1 \]

swap$(A[i], A[j])$

swap$(A[i + 1], A[r])$

Quicksort$(A, p, i)$

Quicksort$(A, i + 2, r)$

After first partitioning

<table>
<thead>
<tr>
<th></th>
<th>10</th>
<th>2j</th>
<th>1</th>
<th>3</th>
<th>7</th>
<th>6</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>2i</td>
<td>2i</td>
<td>10j</td>
<td>1</td>
<td>3</td>
<td>7</td>
<td>6</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>10j</td>
<td>3</td>
<td>7</td>
<td>6</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td>10j</td>
<td>7</td>
<td>6</td>
<td>4</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td>10</td>
<td>7</td>
<td>6</td>
<td>4</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td>10</td>
<td>7</td>
<td>6</td>
<td>4</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>10j</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>10j</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>
Quick sort
Performance
Parallel formulation
Example
Pivot selection
Combining blocks
MPI version

- Initial array $A$
  
  \[
  \begin{array}{ccccccccc}
  10 & 2 & 1 & 3 & 7 & 6 & 4 & 5 \\
  \end{array}
  \]

- Quicksort($A,1,8$)
  - pivot 5, after partition, $i = 4$
  
  \[
  \begin{array}{cccccccc}
  2 & 1 & 3 & 4 & 5 & 6 & 10 & 7 \\
  \end{array}
  \]

  - Quicksort($A,1,4$)
  - Quicksort($A,6,8$)

- Quicksort($A,1,4$)
  - pivot 4, after partition, $i = 3$
  
  \[
  \begin{array}{cccccccc}
  2 & 1 & 3 & 4 & 5 & 6 & 10 & 7 \\
  \end{array}
  \]

  - Quicksort($A,1,3$)
  - Quicksort($A,5,4$) : return
Quick sort

- Quick sort(A, 1, 3)
  - pivot 3, after partition, \( i = 2 \)
  \[
  \begin{array}{cccccccc}
  2 & 1 & 3 & 4 & 5 & 6 & 10 & 7 \\
  \end{array}
  \]
  - Quick sort(A, 1, 2)
  - Quick sort(A, 4, 3) : return

- Quick sort(A, 1, 2)
  - pivot 1, after partition, \( i = 0 \)
  \[
  \begin{array}{cccccccc}
  1 & 2 & 3 & 4 & 5 & 6 & 10 & 7 \\
  \end{array}
  \]
  - Quick sort(A, 1, 0) : return
  - Quick sort(A, 2, 2) : return

- Quick sort(A, 6, 8)
  - pivot 7, after partition, \( i = 6 \)
  \[
  \begin{array}{cccccccc}
  1 & 2 & 3 & 4 & 5 & 6 & 7 & 10 \\
  \end{array}
  \]
  - Quick sort(A, 6, 6) : return
  - Quick sort(A, 8, 8) : return
Performance

- Worst case $O(n^2)$
- Average case $O(n \log n)$
- Crucial for the performance is how to select pivots
- One can select at random
Parallel formulation

- Assume \( n \) numbers and \( p \) processes
- Each process gets \( n/p \) consecutive elements
- Select a pivot
- Broadcast it to all processes
- Process \( i \) computes block of elements \( S_i \) and \( L_i \) such that
  \[
  S_i \leq \text{pivot} < L_i
  \]
- Rearrange the elements of the original array such that it is partitioned as
  \[
  S = \bigcup_i S_i \leq \text{pivot} < L = \bigcup_i L_i
  \]
- Assign \( p_1 \) processes to work on \( S \) and \( p_2 \) processes to work on \( L \), \( p_1 + p_2 = p \)
Apply the same scheme on $S$ with $p_1$ processes and $L$ with $p_2$ processes

$p_1 = \lfloor |S|p/n + 0.5 \rfloor$, $p_2 = p - p_1$

Partition until a block is assigned to a single process and then sort serially
## Example

<table>
<thead>
<tr>
<th></th>
<th>$P_0$</th>
<th></th>
<th>$P_1$</th>
<th></th>
<th>$P_2$</th>
<th></th>
<th>$P_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>86</td>
<td>79</td>
<td>52</td>
<td>18</td>
<td>40</td>
<td>14</td>
<td>4</td>
<td>94</td>
</tr>
<tr>
<td>31</td>
<td>30</td>
<td>34</td>
<td>47</td>
<td>65</td>
<td>3</td>
<td>85</td>
<td>56</td>
</tr>
<tr>
<td>65</td>
<td>3</td>
<td>86</td>
<td>35</td>
<td>6</td>
<td>45</td>
<td>86</td>
<td>56</td>
</tr>
</tbody>
</table>

After first partition with pivot 45

<table>
<thead>
<tr>
<th></th>
<th>$P_0$</th>
<th></th>
<th>$P_1$</th>
<th></th>
<th>$P_2$</th>
<th></th>
<th>$P_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td>40</td>
<td>52</td>
<td>86</td>
<td>79</td>
<td>14</td>
<td>4</td>
<td>31</td>
</tr>
<tr>
<td>30</td>
<td>34</td>
<td>3</td>
<td>85</td>
<td>47</td>
<td>65</td>
<td>35</td>
<td>6</td>
</tr>
<tr>
<td>45</td>
<td>86</td>
<td>56</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

After combining the lower and upper parts

<table>
<thead>
<tr>
<th></th>
<th>$P_0$</th>
<th></th>
<th>$P_1$</th>
<th></th>
<th>$P_2$</th>
<th></th>
<th>$P_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td>40</td>
<td>14</td>
<td>4</td>
<td>31</td>
<td>30</td>
<td>34</td>
<td>3</td>
</tr>
<tr>
<td>35</td>
<td>6</td>
<td>45</td>
<td>52</td>
<td>86</td>
<td>79</td>
<td>94</td>
<td>85</td>
</tr>
<tr>
<td>47</td>
<td>65</td>
<td>86</td>
<td>56</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

After second partition $P_0$, $P_1$ with pivot 6, $P_2$, $P_3$ with pivot 56

<table>
<thead>
<tr>
<th></th>
<th>$P_0$</th>
<th></th>
<th>$P_1$</th>
<th></th>
<th>$P_2$</th>
<th></th>
<th>$P_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>31</td>
<td>14</td>
<td>18</td>
<td>40</td>
<td>3</td>
<td>6</td>
<td>30</td>
</tr>
<tr>
<td>35</td>
<td>34</td>
<td>45</td>
<td>52</td>
<td>94</td>
<td>79</td>
<td>86</td>
<td>47</td>
</tr>
<tr>
<td>56</td>
<td>65</td>
<td>86</td>
<td>85</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

After combining the lower and upper parts
Now each process sorts sequentially

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>$P_1$</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_0$</td>
<td></td>
<td>4</td>
<td>3</td>
<td>6</td>
<td>31</td>
<td>14</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>$P_2$</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>$P_3$</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>4</td>
<td>6</td>
<td>14</td>
<td>18</td>
<td>30</td>
<td>31</td>
<td>34</td>
<td>35</td>
<td>40</td>
<td>45</td>
<td>47</td>
<td>53</td>
<td>56</td>
<td>65</td>
<td>79</td>
<td>85</td>
<td>86</td>
<td>86</td>
<td>94</td>
</tr>
</tbody>
</table>
Pivot selection

- Selecting a pivot at random works well in the sequential quick sort
- A process from a process group can select a pivot at random
- If a “bad” partition occurs, we may have load imbalance
- Assume uniform distribution of the elements
- If we assume uniform distribution of elements, $n/p$ can be considered as a representative sample
- A process can pick the median of these $n/p$ elements and round to the closest element
- The partitions are roughly in half
Combining blocks

- How to arrange the $S_i$ and $L_i$?
- We can concatenate them in process order
- Need to find where each block starts
- $S_0$ is at the beginning of the array
- $S_1$ starts at location $|S_0|$
- $S_2$ starts at location $|S_0| + |S_1|$
- $j$th element of $S_i$ is at location $\sum_{k=0}^{i-1} |S_k| + j$
- $j$th element of $L_i$ is at location $\sum_{k=i}^{p-1} |L_k| + j$
- We can maintain arrays

$$Q_i = \sum_{k=0}^{i-1} S_k, \quad R_i = \sum_{k=0}^{i-1} L_k$$

- $Q_i$ is the start of $S_i$
- $R_i$ is the start of $L_i$, where numbering is from the last element of the last $S_i$
The algorithm so far is suitable for a shared memory implementation. How to do a distributed implementation?

- Distribute the array to be sorted
- Broadcast a pivot among $p$ processes
- Each partitions $n/p$ elements in $O(n/p)$
- The re-arrangement of lower and upper parts involves communication
  - Need to know where a process should send its $S_i$ and $L_i$ parts
  - Once determined, data gets exchanged