CISC 320 Introduction to Algorithms
Fall 2003

Lecture 18
Parallel Algorithms

Other models

- A hypercube (dimension = 3)
- In a hypercube with p processors, each processor is connected to lg(p) other processors
- A bounded-degree network (degree = 4)

The NC class is a complexity class of all problems that are solved in polylogarithmic parallel time and polynomial processors.

- Input size = n
- Number of processors p(n) = O(n^k), where k is some constant
- Worst-case time T(n) = O(log^m n), where m is some constant. (This is called poly-log time.)

Note: The NC class is model independent, i.e., a problem that can be solved in poly-log time by PRAM should be solvable in a bounded-degree network in poly-log too.

The binary Fan-In technique
- Not all steps can be parallelized
- Tournament
  - Elements are paired off and compared in "rounds"

PRAM (Parallel Random Access Machine)

1. p general-purpose processors
2. Shared large random access memory
3. All processors run the same program
4. Processors use pid to index into the memory
5. Concurrent read
6. Write conflicts are resolved by variations
Input: Keys $x[0], ..., x[n-1]$, initially in memory cells $M[0], ..., M[n-1]$, and integer $n$.

Output: The largest key will be left in $M[0]$.

Remarks: cells $M[n]$ to $M[2n-1]$ are initialized to $-\infty$.

$\text{partTournamentMax}(M, n)$

\[ \text{inr} = 1; \]
\[ \text{while} \ (\text{inr} < n) \]
\[ \text{Key big, tmp0, tmp1}; \]
\[ \text{Read M[pid]} \text{ into tmp0}. \]
\[ \text{Read M[pid+inr]} \text{ into tmp1}. \]
\[ \text{big} = \max (\text{tmp0}, \text{tmp1}); \]
\[ \text{Write big into M[pid]}. \]
\[ \text{inr} = 2^*\text{inr}; \]

Correctness by induction
- At step $t$, incr = $2^t$. Cell $M[i]$ contains the maximum of $x[i],...,x[i+\text{incr}-1]$
  \[
  \text{tmp0} = \max(x[i],...,x[i+2^t-1])
  \]
  \[
  \text{tmp1} = \max(x[i+2^t],...,x[i+2^t+2^t-1])
  \]
  \[
  \text{big} = \max(x[i],...,x[i+2^t-1])
  \]

Handling Write Conflicts
- CREW (Concurrent Read, Exclusive Write): only one processor write in a particular cell at any one step: It is illegal to have more than one processor write to one cell at the same time.
- CRCW (Concurrent Read, Concurrent Write):
  - Common-Write model
  - Arbitrary-Write model
  - Priority-Write model

Problem: Boolean or on $n$ Bits
Input: Bits $x_0, ..., x_n$, as 0's and 1's, in $M[0], ..., M[n-1]$
Output: $x_0 \lor \cdots \lor x_n$, in $M[0]$.

Solution 1 (CREW)
Same as the binary Pan-In for finding the max of $n$ integers. It runs in $\Omega(\log n)$ time.
**Problem**  Boolean or on n Bits

Input: Bits $x_0, \ldots, x_{n-1}$ as 0's and 1's in $M[0], \ldots, M[n-1]$.

Output: $x_0 \lor \ldots \lor x_{n-1}$ in $M[0]$.

Solution 2 (CRCW)

commonWriteOn$(M, n)$

1. $P_i$ reads $x_i$ from $M[i]$.
   If $x_i$ is 1; then $P_i$ writes 1 in $M[0]$.

Correctness: Since all the processors that write into $M[0]$ write the same value, the program is a legal program under Common-Write model.

Running time: It takes just one step when $n$ processors are available.

---

**Input:** Keys $x[0], \ldots, x[n-1]$, initially in memory cells $M[0], \ldots, M[n-1]$, and integer $n > 2$.

**Output:** The largest key will be left in $M[0]$.

**Remarks:** Processors $P_i$ have index $i = \text{id}_i$ and $j = \text{id}_i - n$. If $i > j$, the processor does not work.

```
fastMaxOn$(M, n)$
1. Compute $i$ and $j$ from $\text{id}_i$.
   If $i > j$, return;
   $P_i$ reads $x_i$ (from $M[i]$).
2. $P_i$, $P_j$ compares $x_i$ and $x_j$.
   $P_i$ writes 1 if $x_i > x_j$.
   $P_i$ writes 1 if $x_i < x_j$.
   $P_i$ writes 0 if $x_i = x_j$.
```

---

**Initial memory contents (n = 4)**

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3</td>
<td>4</td>
<td>7</td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>P0</td>
<td>P1</td>
<td>P2</td>
<td>P3</td>
<td></td>
</tr>
</tbody>
</table>

**After Step 2**

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>P0</td>
<td>P1</td>
<td>P2</td>
<td>P3</td>
<td></td>
</tr>
</tbody>
</table>

**After Step 3**

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>P0</td>
<td>P1</td>
<td>P2</td>
<td>P3</td>
<td></td>
</tr>
</tbody>
</table>

---

**In p u t**

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>P0</td>
<td>P1</td>
<td>P2</td>
<td>P3</td>
<td></td>
</tr>
</tbody>
</table>

**In p u t L o s e r  a r r a y**

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>P0</td>
<td>P1</td>
<td>P2</td>
<td>P3</td>
<td></td>
</tr>
</tbody>
</table>