Randomized algorithms

Notes of lecture  21

On 11/8/2011

Taken By

Seema Munavalli
Theorem:

We can sort \( n \) integers in the range \([1, n(\log n)^C]\) in \( \tilde{O}(\log n) \) time using \( n/\log n \) Arbitrary CRCW PRAM processors, where \( c \) is any constant.

Proof: Here is an algorithm:

There are two phases:

- **Phase 1** - Sort the keys with respect to their first part -> Coarse Sort
- **Phase 2** - Stable sort the keys with respect to their second parts -> Fine Sort

**Fact:** If \( \exists \) a stable sort algorithm that sorts \( n \) integers in the range \([1,R]\) in time \( T \) using \( P \) processors, we can use the same algorithm to sort \( n \) integers in the range \([1, R^C]\) in \( O(T) \) time using \( P \) processors, for any constant \( C \).

Fine Sort:

Lemma – we can sort \( n \) integers in the range \([1, \log n]\) in \( O(\log n) \) time using \( n/\log n \) CREW PRAM processors.

Proof: Let \( K_1, K_2, \ldots, K_{\log n}, K_{\log n+1}, \ldots, K_{2 \log n}, \ldots, K_n \) be the input.

Number of processors, \( P = n/\log n \)
Assign log n keys per processor

**Step 1**

For 1≤i≤P in parallel do

Processor i performs a bucket sort of its keys and computes \(N_{ij} = \# \text{ of keys with a value } j \) 1≤j≤log n.

\[
\begin{array}{cccccc}
N_{11} & N_{12} & N_{13} & \ldots & \ldots & N_{1 \log n} \\
N_{21} & N_{22} & N_{23} & \ldots & \ldots & N_{2 \log n} \\
\vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\
N_{P1} & N_{P2} & N_{P3} & \ldots & \ldots & N_{P \log n}
\end{array}
\]

**This step takes O(log n) time.**

**Step 2**

All the P = n / log n processors do a prefix sum computation on \(N_{11}, N_{21}, N_{31}, N_{P1}, N_{12}, N_{22}, \ldots, N_{P2}, \ldots, N_{1 \log n}, N_{2 \log n}, \ldots, N_{P \log n}\)

**This step takes O(log n) time.**

**Step 3**

For 1≤i≤P in parallel do

processor i writes its keys with value j,

starting from memory cell

\[
\begin{align*}
N_{11} + N_{21} + \ldots + N_{P1} + N_{12} + N_{22} + \ldots + N_{P2} + \ldots + N_{1(j-1)} + N_{2(j-1)} + \ldots + N_{P(j-1)} + N_{1j} + N_{2j} + \ldots + N_{(i-1)j} + 1.
\end{align*}
\]

The above is done for each value of j, 1≤j≤log n.

**This step takes O(log n) time.**

**Total runtime for this algorithm = O(log n)**

Note: This algorithm is stable
Corollary: We can stable sort $n$ integers in the range $[1,(\log n)^C]$ in $O(\log n)$ time using $n/\log n$ CREW PRAM Processor, for any constant $C$. This takes care of the Fine Sort problem.

**An assignment problem**

Input is a sequence

$X = k_1, k_2, \ldots, k_n$

Each key $k_i$ belongs to a group $g_i$

Think of $g_i$ as an integer.

$1 \leq i \leq n; \quad 1 \leq g_i \leq Q.$

Let $n_j$ be the # of keys that belong to group $j$, $1 \leq j \leq Q$.

Let $N_i$ be such that

$N_i \geq n_i$ and $\sum N_i = O(n)$.

Given $X$ and the $N_i$’s the problem is to permute the sequence $X$, such that all the keys in group 1 appear first, followed by all the keys in group 2, …, followed by all the keys in group $Q$.

*LEMMA:* The above assignment problem can be solved in $\tilde{O}(\log n)$ time using $n/\log n$ Arbitrary CRCW PRAM processors.

<table>
<thead>
<tr>
<th>Bucket 1</th>
<th>Bucket 2</th>
<th>…………………………………</th>
<th>Bucket Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2N_1$</td>
<td>$2N_2$</td>
<td></td>
<td>(linear space)</td>
</tr>
</tbody>
</table>

Assume that the estimates $N_i$’s are in common memory (as a part of the input)

$N_1 \quad N_2 \quad \ldots \quad N_Q$

Assign $\log n$ keys per processor

**Step 1**

Perform a prefix computation on

$N_1, N_2, \ldots, N_Q$ and assign
2Nᵢ cells for group i, 1 ≤ i ≤ Q

**Step 2**

For 1 ≤ i ≤ P = n/log n in parallel do

Repeat

Processor i picks the next unassigned element and performs as many rounds as needed to place it.

Until all the log n elements are placed.

**A Round** – Let k be the element to be placed & let q be the group #.
Pick a random cell in bucket q
If this cell is occupied, wait for the next round; If not try to write k in this cell;
Read from this cell; if the cell has k, move on to next key; If not wait for the next round.

**Step 3**

Compress the buckets using a prefix sums computation.

**Analysis**

For any processor, probability of success in any round is ≥ ½

⇒ Expected # of keys successfully placed in one round ≥ ½
⇒ The # of keys successfully placed in d log n rounds is lower bounded by a binomial B(d log n, ½) using Chertoff bounds, this # is > log n with probability ≥ (1 - n⁻ᵃ) for some proper constant d.

A possible value for d is 16.