Introduction to Parallel and Distributed Processing

Parallel Prefix

Jaroslaw ‘Jaric’ Zola

http://www.jzola.org/

Suggested Reading

- Blelloch, G.E. "Prefix sums and their applications," 1990.

Prefix/Scan/Partial sum

- We are given a sequence of n elements $[x_0, x_1, \ldots, x_{n-1}]$
- And binary associative operator $\otimes$
- We want to generate a sequence $[s_0, s_1, \ldots, s_{n-1}]$ where:

$$s_i = x_0 \otimes x_1 \otimes \ldots \otimes x_i$$

Prefix Example

- Example with $n = 6$ integers and operator $+$

x_i	3	2	5	1	4	6
	3	3	3	3	3	3
		2	2	2	2	2
			5	5	5	5
				1	1	1
					4	4
						6
s_i	3	5	10	11	15	21

Sequential Algorithm

- Simple $O(n)$ algorithm:

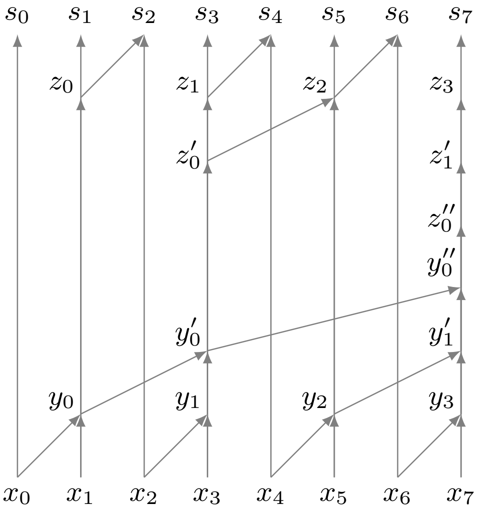
PREFIX_SUM

Input: $[x_0, x_1, \ldots, x_{n-1}]$
Output: $[s_0, s_1, \ldots, s_{n-1}]$

```
1: s_0 ← x_0
2: for i = 1 ... n − 1 do
3:     s_i ← s_{i−1} ⊗ x_i
```

- Nasty sequential dependency between s_i and s_{i-1}

Parallel Algorithm No 1



Parallel Algorithm No 1

PARALLEL_PREFIX_SUM

Input: $[x_0, x_1, \ldots, x_{n-1}], n = 2^d$

Output: $[s_0, s_1, \ldots, s_{n-1}]$

$s_0 \leftarrow x_0$
if $n = 1$ **then**
    return
for $i = 0 \ldots n/2 - 1$ **pardo**
    $y_i \leftarrow x_{2i} \otimes x_{2i+1}$
$[z_0, z_1, \ldots, z_{n/2}] \leftarrow$ PARALLEL_PREFIX_SUM($[y_0, y_1, \ldots, y_{n/2}]$)
for $i = 1 \ldots n - 1$ **pardo**
    if $i \bmod 2 = 0$ **then**
        $s_i \leftarrow z_{i/2-1} \otimes x_i$
    else
        $s_i \leftarrow z_{(i-1)/2}$

Parallel Algorithm No 1

- The algorithm can be implemented entirely in place
- We use balanced binary tree: traversing up we apply $\otimes$, traversing down we apply if statement
- Complexity analysis:

$$T_p(n) = T_p(n/2) + O(1)$$

$$T_p =$$

Parallel Algorithm No 1

- The algorithm can be implemented entirely in place
- We use balanced binary tree: traversing up we apply $\otimes$, traversing down we apply if statement
- Complexity analysis:

$$T_p(n) = T_p(n/2) + O(1)$$

$$T_p = O(\log(p))$$

Parallel Algorithm No 2

PARALLEL_PREFIX_SUM

Input: $rank, x_{rank}, n = 2^d$

Output: s_{rank}

$s_{rank} \leftarrow x_{rank}$
$S \leftarrow s_{rank}$
for $i = 0 \ldots d-1$ **do**
    SEND($\text{dest} = rank \oplus 2^i$, S)
    $S_{recv} \leftarrow$ RECV($\text{src} = rank \oplus 2^i$)
    $S \leftarrow S + S_{recv}$
    if $\text{dest} < rank$ **then**
        $s_{rank} \leftarrow S$

Parallel Algorithm No 2

- Complexity analysis:

  $T_p = O(\log(p))$ computation and communication steps

Question

- Propose efficient solution for case when $p \ll n$
- Something like: $T_p = O(\frac{n}{p} + \log(p))$

Example Application: Ranking Processors

- Each processor is “marked” or “unmarked” based on some predefined predicate
- We want to rank all r “marked” processors
  i.e. assign id from $0 \ldots r-1$
- We run parallel prefix, x_i is “mark”, s_i becomes rank

Example Application: Linear Recurrences

- We are given $x_i = ax_{i-1} + bx_{i-2}$ for some a, b
- We want to compute the sequence knowing x_0 and x_1
- We observe that:

$$\begin{bmatrix} x_i & x_{i-1} \end{bmatrix} = \begin{bmatrix} x_{i-1} & x_{i-2} \end{bmatrix} \begin{bmatrix} a & 1 \\ b & 0 \end{bmatrix}$$

hence:

$$\begin{bmatrix} x_i & x_{i-1} \end{bmatrix} = \begin{bmatrix} x_1 & x_0 \end{bmatrix} \begin{bmatrix} a & 1 \\ b & 0 \end{bmatrix}^{i-1}$$

Example Application: Linear Recurrences

- We can compute $\begin{bmatrix} a & 1 \\ b & 0 \end{bmatrix}^{i-1}$ using parallel prefix with operator 2-by-2-matrix-multiplication
- Practical use, linear congruential RNG:

$$x_{i+1} = (ax_i + b) \bmod m$$

- We have:

$$\left(\sum_{i=1}^{k} a_i\right) \bmod m = \left(\left(\sum_{i=1}^{k-1} a_i\right) \bmod m + a_k\right) \bmod m$$

$$\begin{bmatrix} x_i & 1 \end{bmatrix} = \begin{bmatrix} x_0 & 1 \end{bmatrix} \begin{bmatrix} a & 1 \\ b & 0 \end{bmatrix}^{i}$$

For Fun

The line-of-sight problem:

- Given a terrain map in the form of a grid of altitudes and an observation point X on the grid, find which points are visible along a ray originating at the observation point.