Self Simulation $3 = \lceil \frac{5}{2} \rceil$

PRAM of N processors $O(\frac{N}{n})$

PRAM of n processors steps

EREW PRAM P_1 P_2 P_3 P_4 P_5 Q_1 Q_2

M_1 M_2 M_3 M_4 M_5 1. $Q_1 Q_2 \to P_1 P_2$

M_1' M_2' M_3' M_4' M_5' 2. $Q_1 Q_2 \to P_3 P_4$

3. $Q_1 \to P_5$

10 processors

4	2 5 7 9
7	1 10
13	3 4 8
20	6

P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}
7	4	13	13	4	20	4	13	4	7
V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_8	V_9	V_{10}

PRIORITY : 1. $Q_1 Q_2 Q_3 Q_4 \to P_7$ P_8 P_9 P_{10}

write to 4 13 4 7 $P_7 P_8 P_{10}$ ✓

4	7	13	20
$\cancel{V_7}$	V_{10}	$\cancel{V_8}$	$\cancel{0_{20}}$
V_8		V_3	V_6

2. $Q_1 Q_2 Q_3 Q_4 \to P_3 - P_6$

13 13 4 20

10 processors

4	2 5 7 9
7	1 10
13	3 4 8
20	6

P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}
7	4	13	13	4	20	4	13	4	7
V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_8	V_9	V_{10}

PRIORITY : 1. $Q_1\ Q_2\ Q_3\ Q_4 \rightarrow P_7\ P_8\ P_9\ P_{10}$

write to 4 13 4 7 $\quad \frac{P_7}{P_9}\ P_8\ P_{10}$ ✓

4	7	13	20
~~V_7~~ ~~V_9~~ V_{10}		~~V_8~~	~~O_{20}~~
V_8		V_3 / V_4	V_6

2. $Q_1\ Q_2\ Q_1\ Q_4 \rightarrow P_3 - P_6$

13 13 4 20

3. $Q_1\ Q_2 \rightarrow P_1\ P_2$ write to 7 & 4 ✓

4	7	13	20
V_2	V_1	V_3 / V_4	V_6

$3 = \lceil 10/4 \rceil$ steps

N on n in $O(N/n)$ time

ARBITRARY

COLLISION harder!

\$: collision symbol

1: copy the old values into an auxiliary array

N O_7 O_4 O_{13} O_{13} O_4 O_{20} O_4 O_{13} O_4 O_7

2. Q_1 Q_2 Q_3 $Q_4 \to P_7 \cdots P_{10}$ 4 13 4 7

4 7 13 20

\$ V_{10} V_8 O_{20}

④ Q_1 $Q_2 \to P_1$ P_2 7 & 4

4 7 13 20

\$ \$ \$ V_6

Self-Simulation

- A PRAM model is said to be self-simulating, if for all $N \geq n \geq 1$, a PRAM of that model of size n can simulate a single step of another PRAM of the same model of size N in $O(N/n)$ time.
- All CRCW PRAMs we have seen except TOLERANT are self-simulating.
- TOLERANT is not known to be self-simulating.

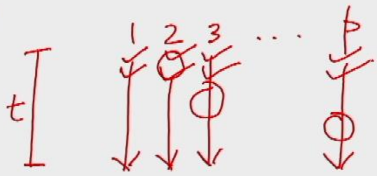
③ $Q_1 \cdots Q_4 \to P_3 \cdots P_6$ 13 13 4 20

4 7 13 20 \$ found, back off

\$ V_{10} \$ V_6

Cost

- The cost or work of an algorithm that runs in t time using p processors is the time-processor product pt
- Simulation on a one processor machine



Optimality

- If $\mathrm{Seq}(n)$ is the worst-case running time of the fastest known sequential algorithm for a problem of size n, an optimal parallel algorithm for the same problem runs in $O(\mathrm{Seq}(n)/p)$ time using p processors

cost = seq(n)

Merging $\Theta(n)$

			cost	
$O(\log n)$	$\frac{n}{\log n}$ prs	:	$O(n)$	✓
$O(\log^2 n)$	$\frac{n}{\log^2 n}$ prs	:	$O(n)$	✓
$O(\log n)$	n	:	$O(n \log n)$	/
time				

Sorting (comp. based)

$\Theta(n \log n)$

$O(\log^2 n)$	$\frac{n}{\log n}$	$O(n \log n)$	✓
$O(\log n)$	n	$O(n \log n)$	✓
$O(\log^2 n)$	n	$O(n \log^2 n)$	X
time	prcrs		

Degree of Parallelism

1	4 ins.
2	3 ins.
3	9 ins
4	1
5	

- The degree of parallelism of a parallel step is the number of instructions in it
- It is the same as the number of processors required to execute it in one clock cycle

1	w_1
2	w_2
3	w_3
$\vdots$	
T	w_T

$$\sum_{i=1}^{T} w_i = W$$ cost of PA

$$P = \max_i (w_i)$$

T steps

$PT = \omega(W)$ cost of exec^n

1	$n \log n$
2	n
3	n
$\vdots$	
$\log n$	n

$$W = n \log n + n(\log n - 1) = O(n \log n)$$

$$O(n \log^2 n)$$

1 $\quad w_1$

2 $\quad w_2$

3 $\quad w_3$

$\vdots$

$T \quad w_T$

$$\sum_{i=1}^{T} w_i = W \quad \text{cost of PA}$$

$$P = \max_i (w_i)$$

T steps

$PT = \omega(W) \longrightarrow$ cost of execn

1 $\quad n \log n$

2 $\quad n$

3 $\quad n$

$\vdots$

$\log n \quad n$

$$W = n \log n + n(\log n - 1) = O(n \log n)$$

$$O(n \log^2 n)$$

Model is self simulating

p processors are there

$$\left\lceil \frac{w_1}{p} \right\rceil + \left\lceil \frac{w_2}{p} \right\rceil + \cdots + \frac{\lceil w_T \rceil}{p} \leq$$

$$\left(\frac{w_1}{p} + 1\right) + \left(\frac{w_2}{p} + 1\right) + \cdots + \left(\frac{w_T}{p} + 1\right)$$

$$= \frac{W}{p} + T \longrightarrow O(T)$$

$$p = \left\lceil \frac{W}{T} \right\rceil \quad \text{cost} = O(W$$

Brent's Scheduling Principle

- Consider a parallel algorithm presented in T steps.
- Say the degree of parallelism of the i-th step is w_i
- So, the total number of instructions in the algorithm is $W = \sum_{i=1}^{n} w_i$
- If we use $P = \max_i(w_i)$ processors, the algorithm runs in exactly T steps
- But the cost of this execution may be $\omega(W)$
- For example, say $w_1 = n \log n$, while $w_i = n$, for $i > 1$, and $T = \log n$
- The above execution takes $T = \log n$ time with $w_1 = n \log n$ processors
- The cost is $O(n \log^2 n)$, whereas $W = O(n \log n)$