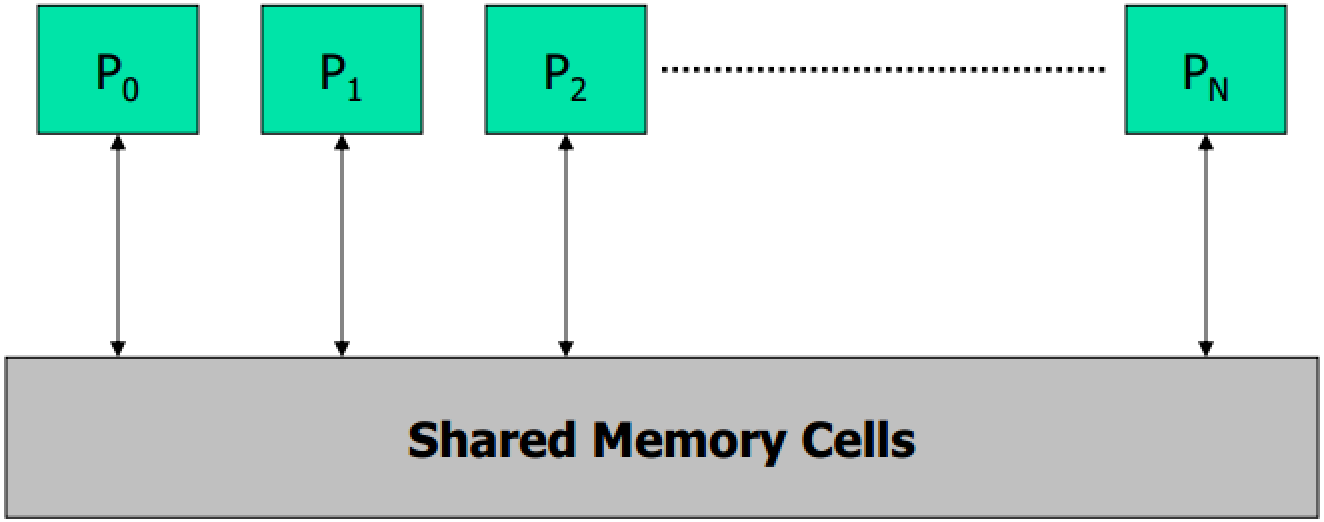
Tutorial 10 – PRAM

Arnamoy Bhattacharyya

SPCL

*Some slides are taken from http://cseweb.ucsd.edu/groups/csag/html/teaching/cse160s05/lectures/Lecture12.pdf

PRAM (Parallel RAM)



Strengths of PRAM

- Natural extension of RAM
- It is simple and easy to understand
  - Removes communication and synchronization issues
- Can be used as a benchmark
  - If algorithm fails on PRAM then too bad
- It is useful to reason (specially with increasing shared memory machines)

Memory Classifications

- PRAMs are classified based on their Read/Write abilities
  - Exclusive Read (ER) – all processors can simultaneously read from distinct memory locations
  - Exclusive Write (EW) – all processors can simultaneously write to distinct memory locations
  - Concurrent Read (CR) – All processors can simultaneously read from any memory location
  - Concurrent Write (CW) – All processors can write to any memory location

Concurrent Write (CW)

- What value gets written finally?
  - Priority CW – processors have priority based on which write value is decided
  - Common CW – multiple processors can simultaneously write only if values are same
  - Arbitrary/Random CW – any one of the values are randomly chosen

Strength of PRAM models

- Model A is computationally stronger (>=) than model B *iff* any algorithm written for B will run unchanged on A

Priority >= Arbitrary >= Common >= CREW >= EREW

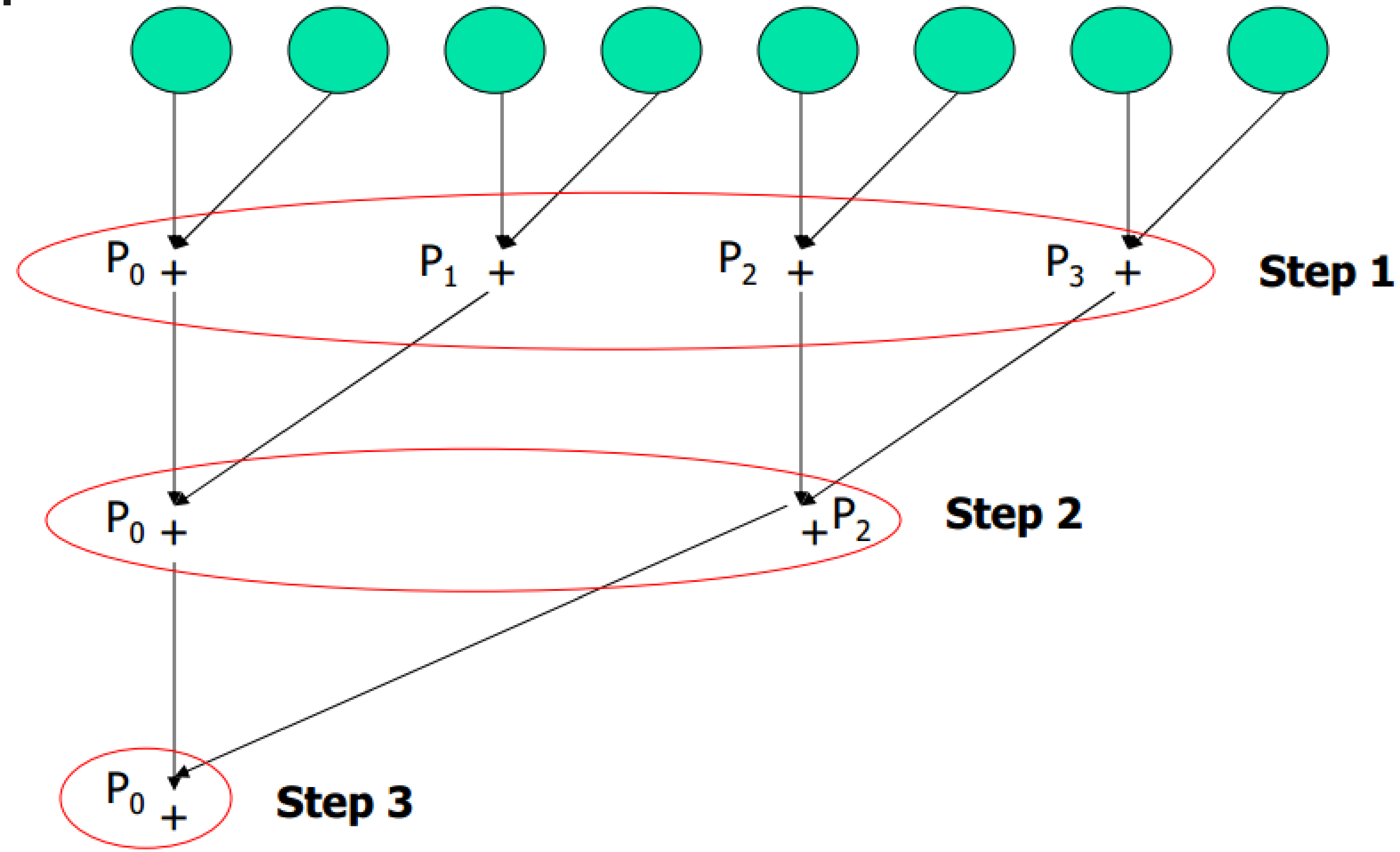
Most powerful → Least powerful

Least realistic ← Most realistic

An initial example

- How do you add N numbers residing in memory location M[0,1, …, N]
- Serial Algorithm = O(N)
- PRAM Algorithm using N processors P_0, P_1, P_2, …, P_N????

Parallel Addition



Parallel Addition

- log(n) steps=time needed
- n/2 processors needed
- Speed-up = n/log(n)
- Efficiency = 1/log(n)
- Applicable for other operations too
  - +, *, <, >, == etc.

Example 2 (complicating things)

- p processor PRAM with n numbers ($p \leq n$)
- Does x exist within the n numbers?
- P_0 contains x and finally P_0 has to know
- Algorithm
  - Inform everyone what x is
  - Every processor checks [n/p] numbers and sets a flag
  - Check if any of the flags are set to 1

Example 2 (complicating things)

- Inform everyone what *x* is
- Every processor checks [n/p] numbers and sets a flag
- Check if any of the flags are set to 1

EREW	CREW	CRCW (common)
log(p)	1	1
n/p	n/p	n/p
log(p)	log(p)	1

Example 3 Compute OR

- Initially
  - table **A** contains values 0 and 1
  - **output** contains value 0

$$\text{for each } 1 \leq i \leq 5 \text{ do in parallel}$$
$$\text{if } A[i] = 1 \text{ then output=1;}$$

- The program computes the "Boolean OR" of A[1], A[2], A[3], A[4], A[5]

Example 4 Pascal's Triangle

- Assume initially table **A** contains [0,0,0,0,0,1] and we have the parallel program

for each $1 \le i \le 5$ do in parallel
$A[i] := A[i] + A[i+1]$

then the consecutive values of the tables A (in parallel step 0, 1, 2, 3, 4, 5) correspond to the Pascal triangle, the nonzero elements in the n-th row are

$$\binom{n}{0}, \binom{n}{1}, \binom{n}{2}, \dots \binom{n}{n}$$

for $n = 0, 1, 2, 3, 4, 5, 6.$

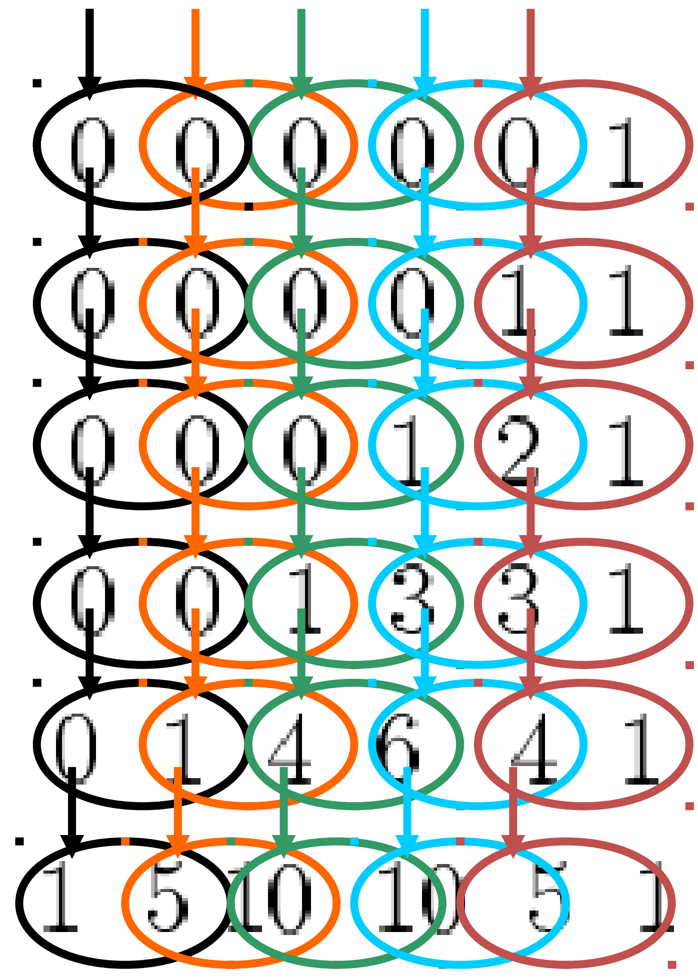
Example 4 Pascal's Triangle

$$\binom{n}{0},\ \binom{n}{1},\ \binom{n}{2},\ \ldots \binom{n}{n}$$

for $n = 0, 1, 2, 3, 4, 5, 6.$

PRAM CREW

for each $1 \le i \le 5$ do in parallel
$A[i]; = A[i] + A[i+1]$



Finding Maximum: CRCW Algorithm

Given n elements A[0, n-1], find the maximum.
With n^2 processors, each processor (i,j) compare A[i] and A[j], for 0 i, j n-1.

FAST-MAX(A):

```
n   length[A]
for i   0 to n-1, in parallel
    do m[i]   true
for i   0 to n-1 and j   0 to n-1, in parallel
    do if A[i] < A[j]
        then m[i]   false
for i   0 to n-1, in parallel
    do if m[i] =true
        then max   A[i]
return max
```

$A[i]$ \ $A[j]$	5	6	9	2	9	m
5	F	T	T	F	T	F
6	F	F	T	F	T	F
9	F	F	F	F	F	T
2	T	T	T	F	T	F
9	F	F	F	F	F	T

max=9

The running time is $O(1)$.

Note: there may be multiple maximum values, so their processors
Will write to max concurrently. Its *work* = $n^2 \times O(1) = O(n^2)$.