CSE 6220 INTRODUCTION TO HIGH PERFORMANCE COMPUTING

A PARALLEL ALGORITHM EXAMPLE & MODEL OF PARALLEL COMPUTATION

Ümit V. Çatalyürek

School of Computational Science and Engineering

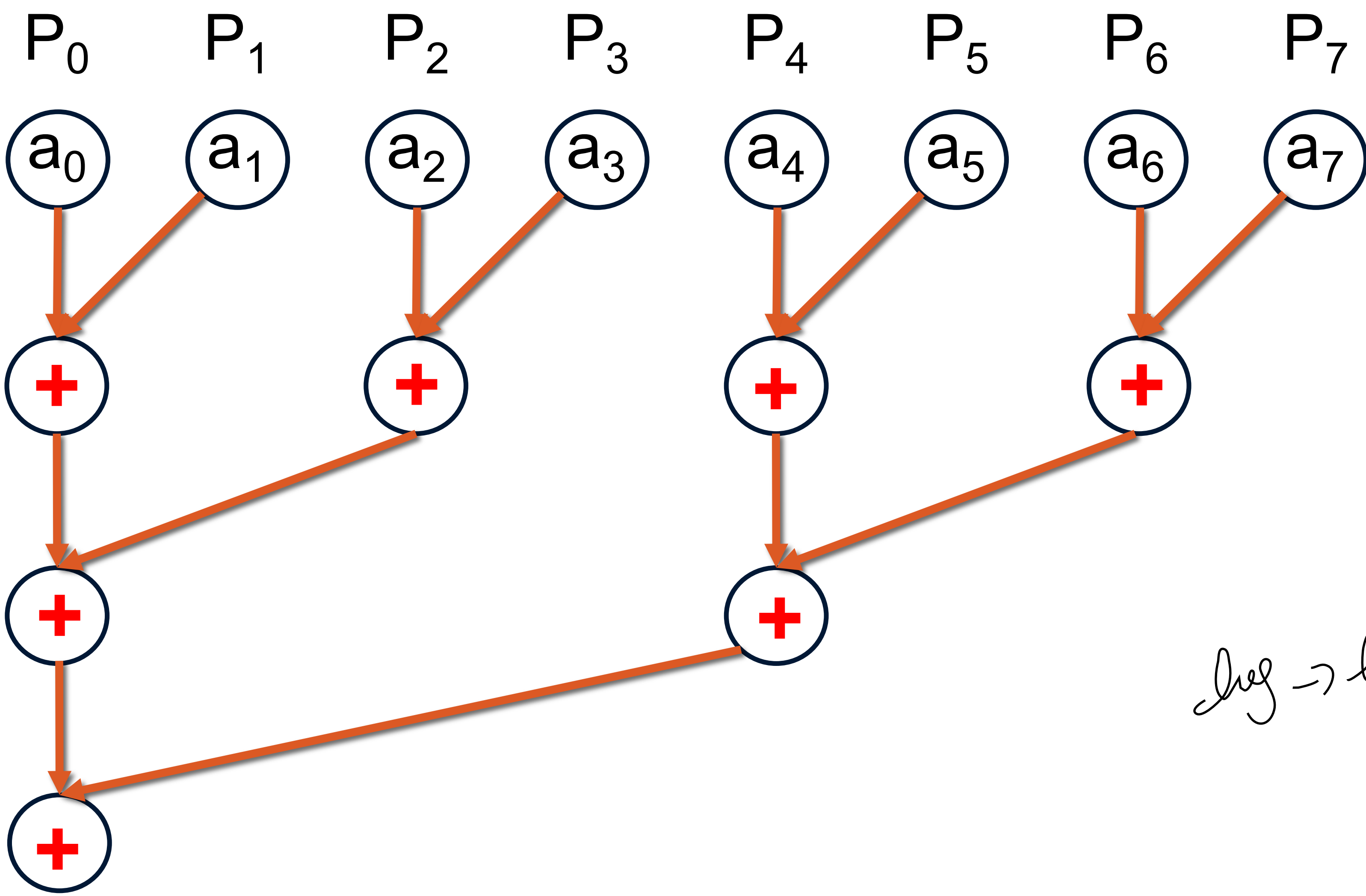
Georgia Institute of Technology

Finding the Sum

- Given n numbers, compute their sum.

- Serial runtime: $T(n, 1) \approx n \;\Theta(n)$

Finding the Sum: A Parallel Algorithm



log → log 2

Finding the Sum

- Given n numbers, compute their sum.
- Serial runtime: $T(n,1) \approx c_1 \cdot n$
- Parallel runtime: $T(n,n) \approx c_2 \cdot \log n$
- Speedup: $S(n) \approx \dfrac{c_1 \cdot n}{c_2 \cdot \log n} \approx \dfrac{c_1}{c_2} \cdot \dfrac{n}{\log n}$
- c_1 ? c_2

$$S(n) \le \frac{n}{\log n}$$

Finding the Sum

- Suppose n =1,000,000.

$$S(n) \leq \frac{n}{\log n}$$

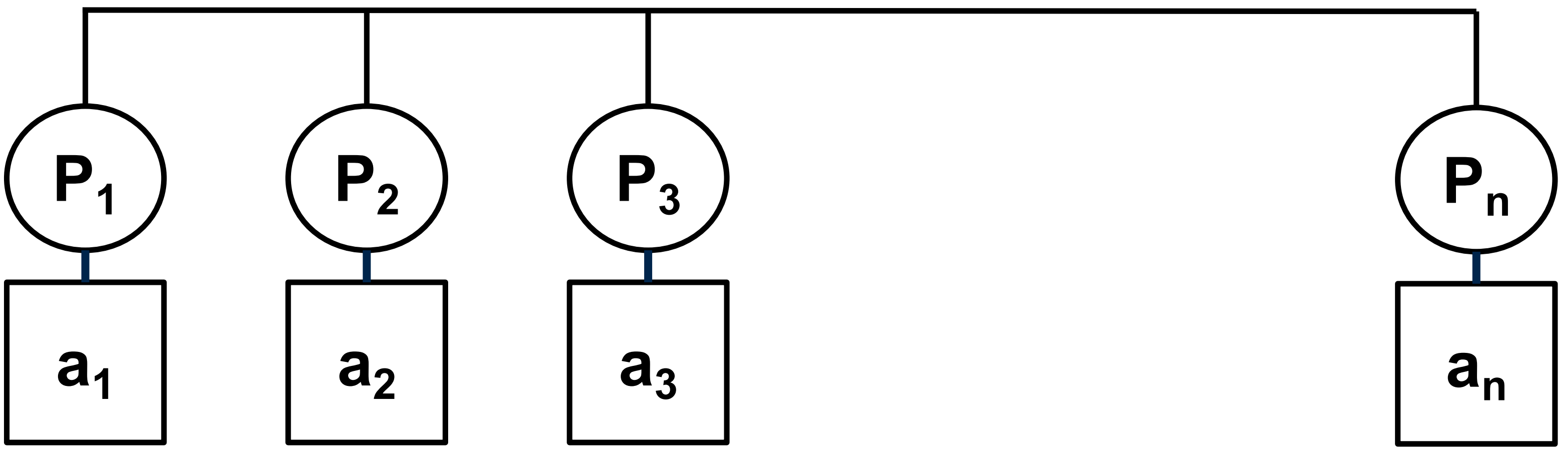
$$S(n) \leq \frac{1{,}000{,}000}{20}$$

$$S(n) \leq 50{,}000$$

(5%) ← 5% score

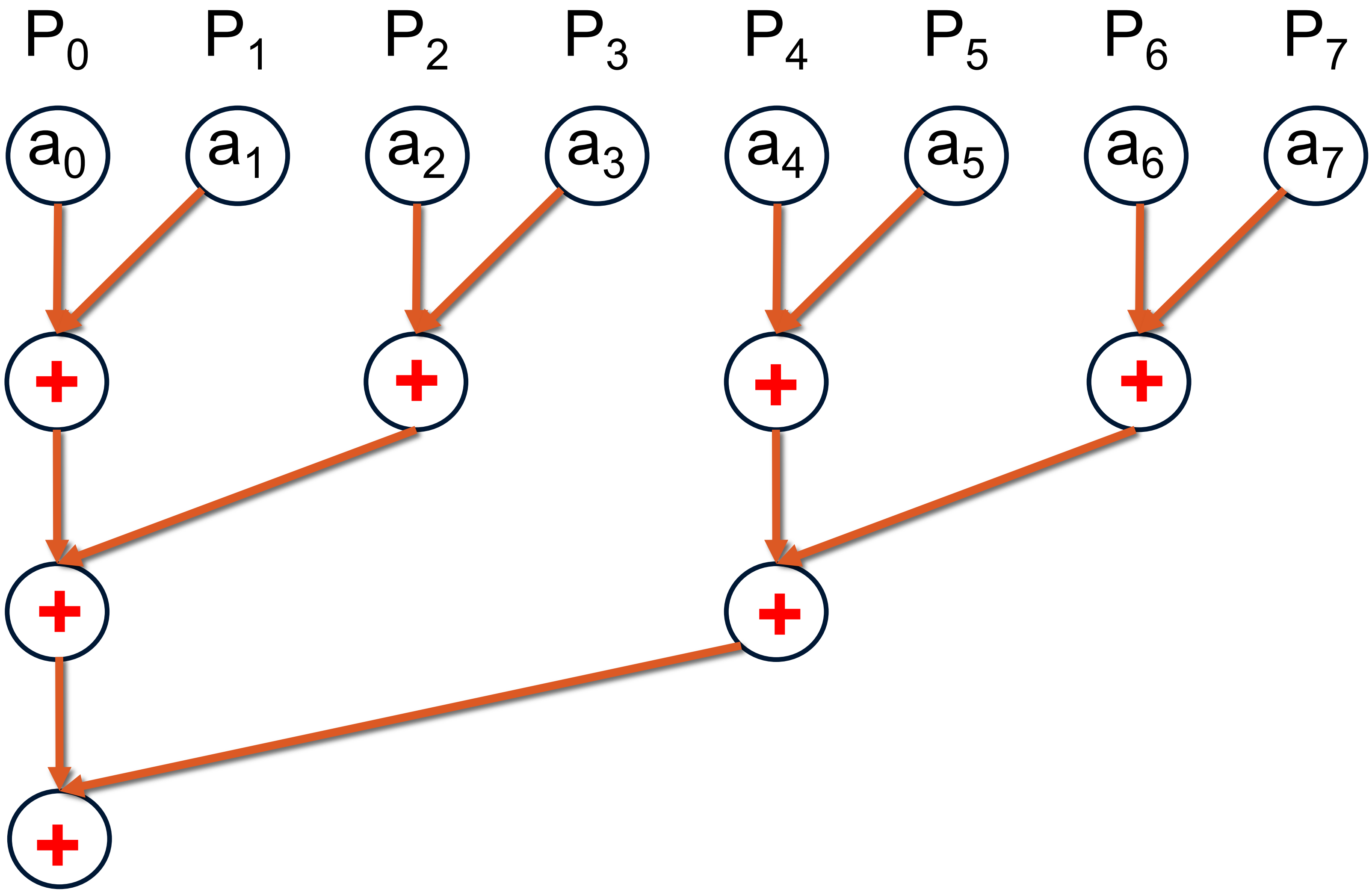
Interconnection Network is Important

1. Bus



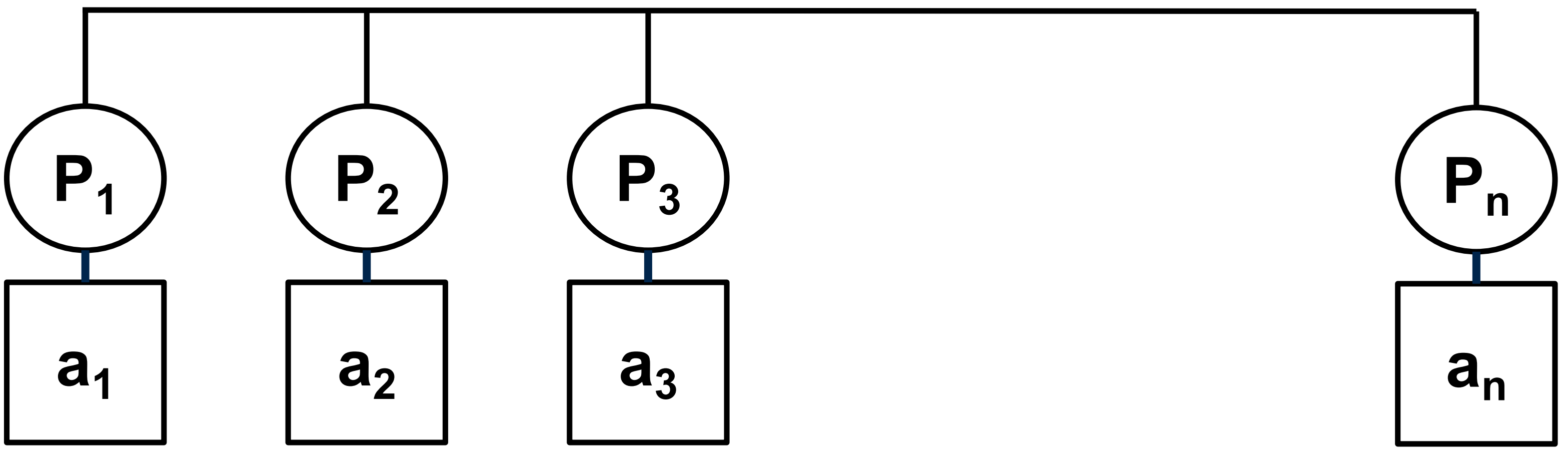
one at a time

Finding the Sum: A Parallel Algorithm



Interconnection Network **is** Important

1. Bus



$$T(n,n) = c_2 \cdot \left[\frac{n}{2} + \frac{n}{4} + \cdots + 1\right]$$

$$T(n,n) = c_2 \cdot (n-1)$$

$$S(n) \approx \frac{c_1 \cdot n}{c_2 \cdot n} = \frac{c_1}{c_2} < 1$$

Worse: c_2 is latency = τ

$$\frac{c_1}{c_2} \approx \frac{1}{10{,}000}$$

Interconnection Network **is** Important

2. All-Connected Network

there is no global memory.

$$T(n,1) \approx 1 \cdot n$$

$$T(n,n) \approx \tau \cdot \log n$$

$$S(n) \approx \frac{1{,}000{,}000}{10{,}000 \times 20} = 5$$

Another Parallel Algorithm

- Use $p \ll n$ processors.
- Assign $\frac{n}{p}$ numbers per processor.
- **Step 1:** Serial — Sum local $\frac{n}{p}$ numbers.
- **Step 2:** Parallel — Add p numbers using p processors.

Interconnection Network **is** Important

2. All-Connected Network

$$T(n,1) \approx n$$

$$T(n,p) \approx \frac{n}{p} + \tau \cdot \log p$$

$$S(p) \approx \frac{n}{\frac{n}{p} + \tau \cdot \log p}$$

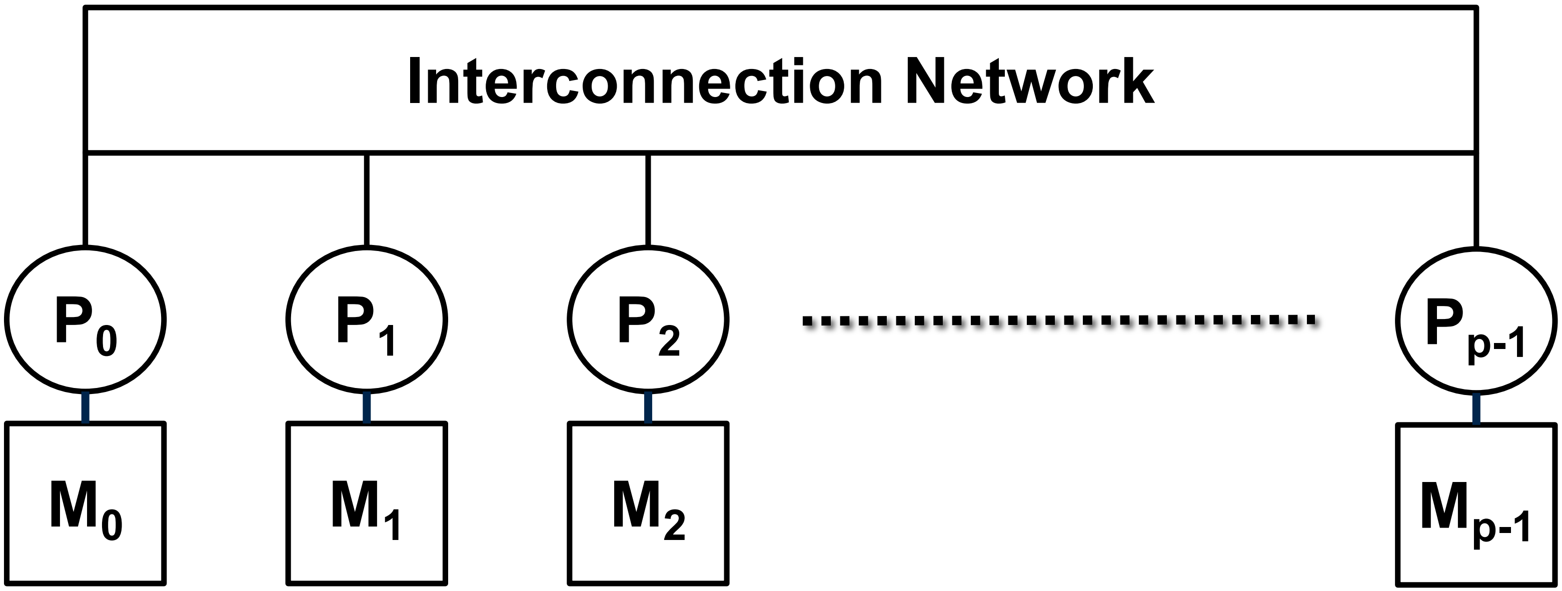
$$S(p) \geq \frac{p}{2} \qquad \text{if} \qquad \tau p \log p \leq n$$

$$n = 1{,}000{,}000$$

$$p \log p \leq \frac{n}{\tau} = 100$$

$$p \leq 22$$

Model of Parallel Computation



Modeling Communication

- P_i is sending message to P_j of size m
- Transfer time: $t_c = \tau + \mu \cdot m$ network cost

  latency τ network

- $\frac{1}{\mu} =$ network bandwidth
- 10Gbps = 1.25GBytes/s
- μ (per byte) = $\frac{1}{1.25\times10^9} = 0.8$ ns

Modeling Communication

- 10Gbps = 1.25GBytes/s
- μ (per byte) = $\frac{1}{1.25\times10^9} = 0.8$ ns
- μ (per word) = 3.2 ns
- 3GHz $\Longrightarrow$ 1 clock cycle = $\frac{1}{3}$ ns
- $\tau : \mu : 1 \quad \Longrightarrow \quad 10^3 - 10^5 : 10 - 20 : 1$

Modeling Communication

- In a parallel communication step
  - Each processor can send at most 1 message
  - Each processor can receive at most 1 message

- Example suppose p=8
  - $0 \rightarrow 5$
  - $1 \rightarrow 0$
  - $2 \rightarrow 1$
  - $3 \rightarrow 7$
  - $4 \rightarrow 2$
  - $5 \rightarrow 4$
  - $6 \rightarrow 6$
  - $7 \rightarrow 3$

$\rightarrow$ (5 0 1 7 2 4 6 3)

Permutation Network Model

- Communication that corresponds to any permutation can be realized in parallel.
- p! communication patterns!
- Not everyone has to participate.
- Each message size can be different.
- Not everyone communicates at “exactly” the same time.

Permutation Network Model

- Communication that corresponds to any permutation can be realized in parallel.
- Ideal network?
- Benes → $\sim 2\, p\ \log p$ ← using serial algorithm, itself.
- Does not work in practice, requires centralized planner.

Permutation Network Model

1. Shift Permutations:
    1. Left shift: $i \to (i - 1 + p) \bmod p$
    2. Right shift: $i \to (i + 1) \bmod p$

2. Hypercubic Permutations:
    - $p = 2^d$ (d is an integer)
    - Represent processor ranks with using d-bit binary number
    - Example $p = 16$ and $d = 4$
    - Consider P_{11} rank = 1011. d positions (0, 1, 2, 3), starting from least significant bit, e.g., position-2 is 0.
    - Hypercubic permutation on fixed position j:
        - Two processors communicate if their ranks differ **only** in position j

Hypercubic Permutations

- Example p = 8, d = 3

j = 0	j = 1	j = 2
$0 \leftrightarrow 1$	$0 \leftrightarrow 2$	$0 \leftrightarrow 4$
$2 \leftrightarrow 3$	$1 \leftrightarrow 3$	$1 \leftrightarrow 5$
$4 \leftrightarrow 5$	$4 \leftrightarrow 6$	$2 \leftrightarrow 6$
$6 \leftrightarrow 7$	$5 \leftrightarrow 7$	$3 \leftrightarrow 7$

Toy Problem using Hypercubic Permutations

- Given n numbers, compute their sum using p processors.
- Serial runtime: $T(n,1) = \Theta(n)$
- Parallel runtime: $T(n,p) = \Theta\left(\frac{n}{p} + \log p\right)$

Finding the sum of n numbers

Algorithm (for P_i)

sum ← add local n/p numbers

for j=0 to d-1 do

  if ((rank AND 2^j) ≠ 0)

    send sum to (rank XOR 2^j)

  else

    receive sum' from (rank XOR 2^j)

    sum = sum + sum'

endfor

if (rank=0)

  print sum

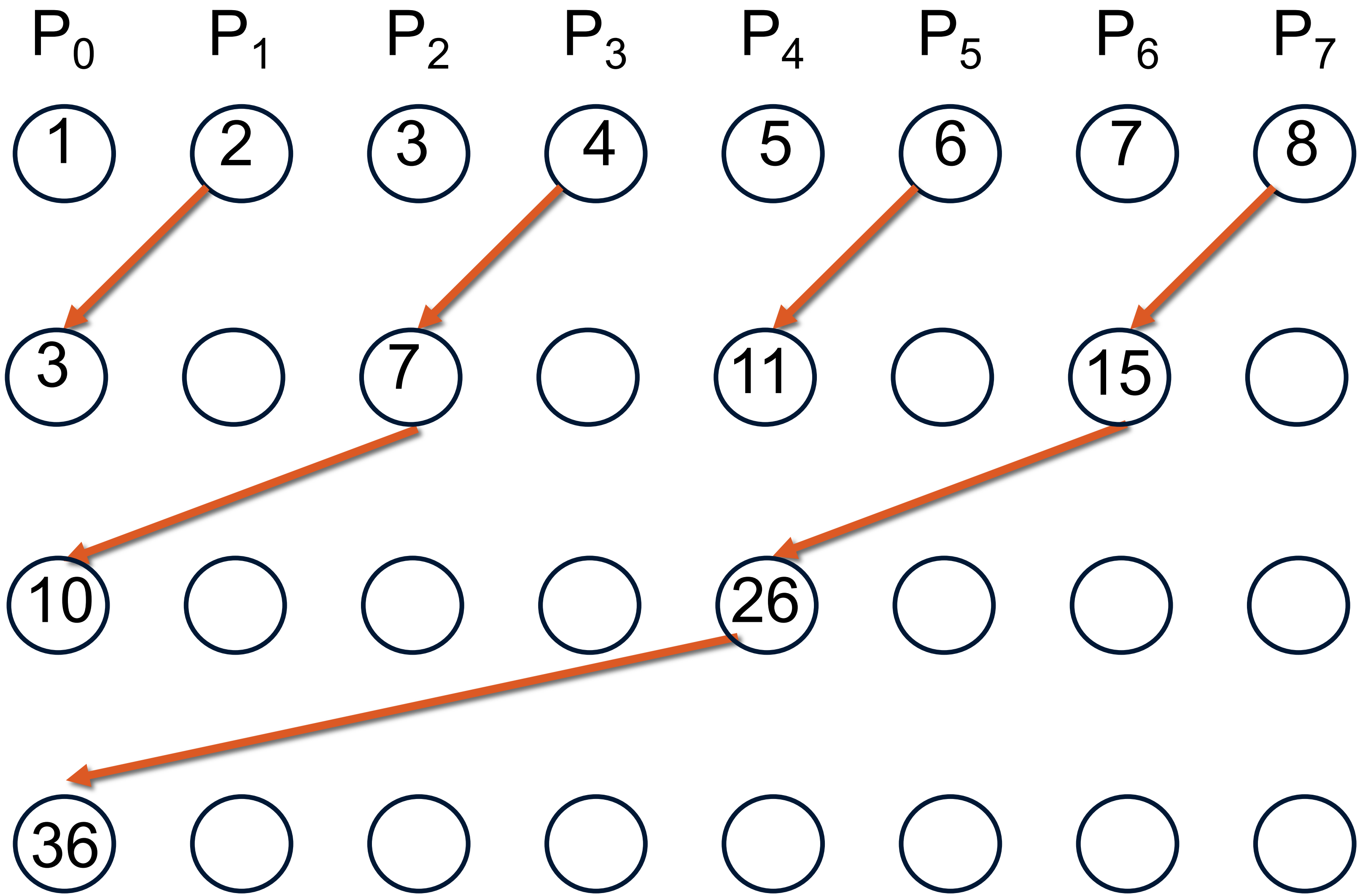
```
- - - - - - - -
        ^
        j
0 0 0 0 1 0 0 0
```

AND, XOR : bitwise operators

log P

one for.

Finding the sum of n numbers



Finding the sum of n numbers

```
Algorithm (for Pi)
sum ← add local n/p numbers
for j=0 to d-1 do

    if ((rank AND 2^j) ≠ 0)
        { send sum to (rank XOR 2^j);  exit; }
    else
        receive sum' from (rank XOR 2^j)
        sum = sum + sum'
endfor
if (rank=0)
    print sum
```