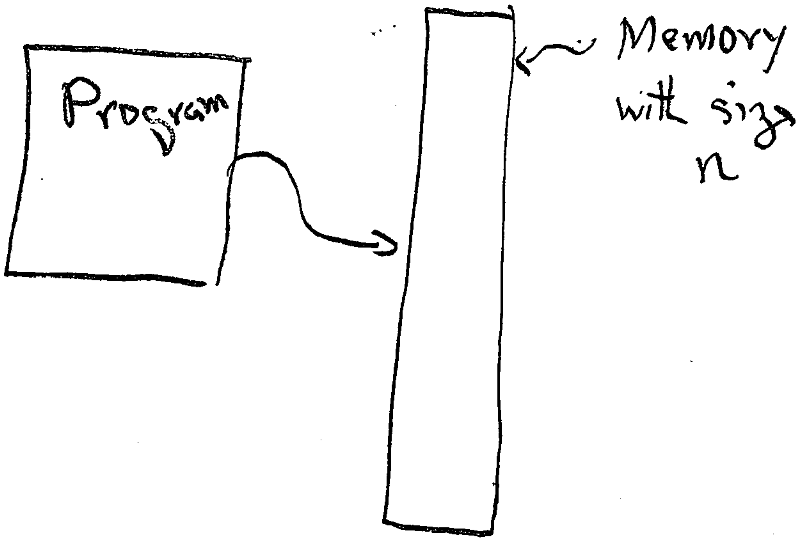
Parallel Algorithms

15-750
3/30/16

So far we have assumed the following model: RAM model



unit time ops

($+, \times, \div$ (log n bits))

(read/write into memory)

A central model to describe Graph Algorithms.

Other models:

1) Agents / ants 2) pointer machines

RAM is unrealistic as n goes to infinity.

1) speed of light (large size machines)
2) quantum effects (small size machines)

Bottom Line: RAM

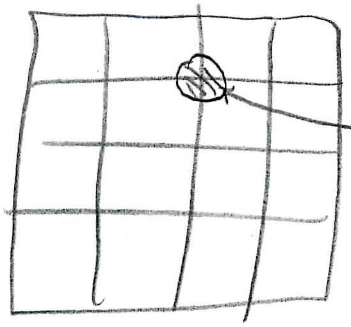
1) Many important algorithms were found using this model.
2) Most algorithms are coded in a RAM like language. eg C

Parallel Models

Fixed connection machines

machines ≡ 1) finite state machine
2) RAM

A) Cellular Arrays 1D, 2D, 3D



processor 2D

1940's von Neumann
60's, 70's algorithms for CA.
Alvy Ray Smith 1974
Wolfram
80's Klaus Sutner
Conway Game of Life

60's Edgar Codd
80's HT Kung

Highly connected models

1) Hypercube $\equiv (V, E)$ 1980's

$$V \equiv \{(a_1 \ldots a_m) \mid a_i \in \{0,1\}\} \quad m = \log_2 n$$

$$((a_1 \cdots a_m), (a_1 \cdots \bar{a}_i \cdots a_m)) \in E$$

2) Shuffle-exchange graph 1980's

$$V = \{(a_1 \cdots a_m) \mid a_i \in \{0,1\}\}$$

$$((a_1 \cdots a_m), (\bar{a}_1 a_2 \cdots a_m)) \in E$$

$$((a_1 \cdots a_m), (a_m a_1 \cdots a_{m-1})) \in E$$

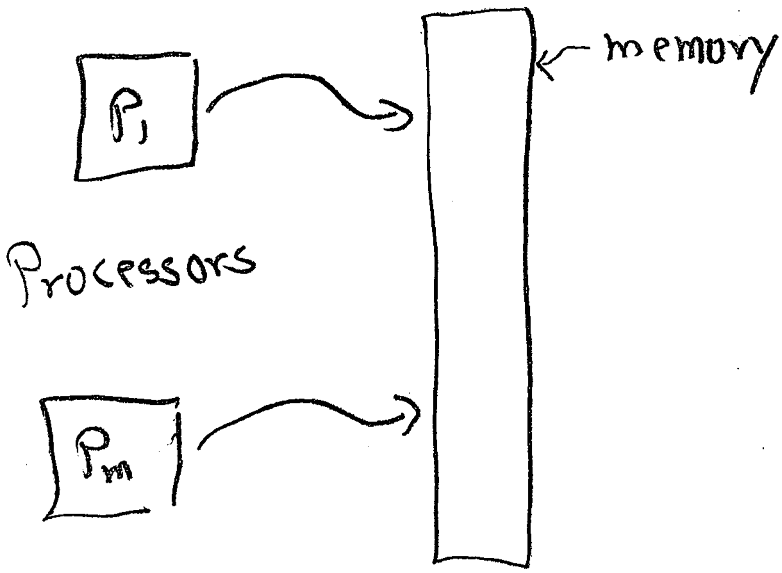
3) Randomly connected graphs

possible models of the brain!

(Valiant)

Shared memory models

1) PRAM (Parallel Random Access Machine)



	Exclusive	Concurrent
Read		
Write		

Unit time ops

$+, \times, \div$

read / write

ER EW

CR CW

PRAM issues:

Penalty for CR on an ER machine?

1) machine crashes!

2) garbage read!

How is synchronization handled?

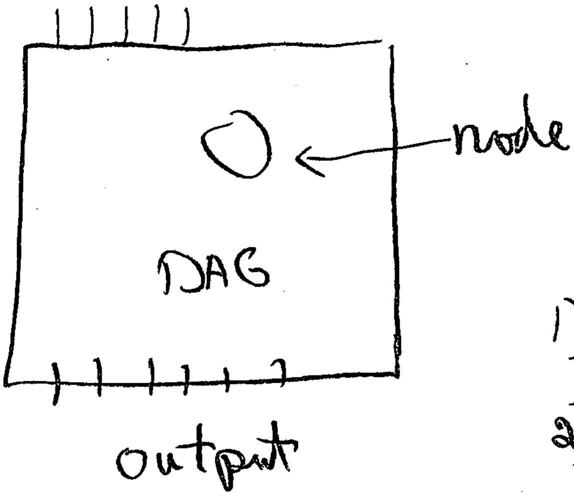
1) After each unit of time!

2) ~~Bulk~~ Synchronous Parallel BSP Valiant

3) none!

Circuit Model

inputs in either bits or words



nodes 1) $\wedge, \vee, \neg$ gates
2) Arithmetic ops

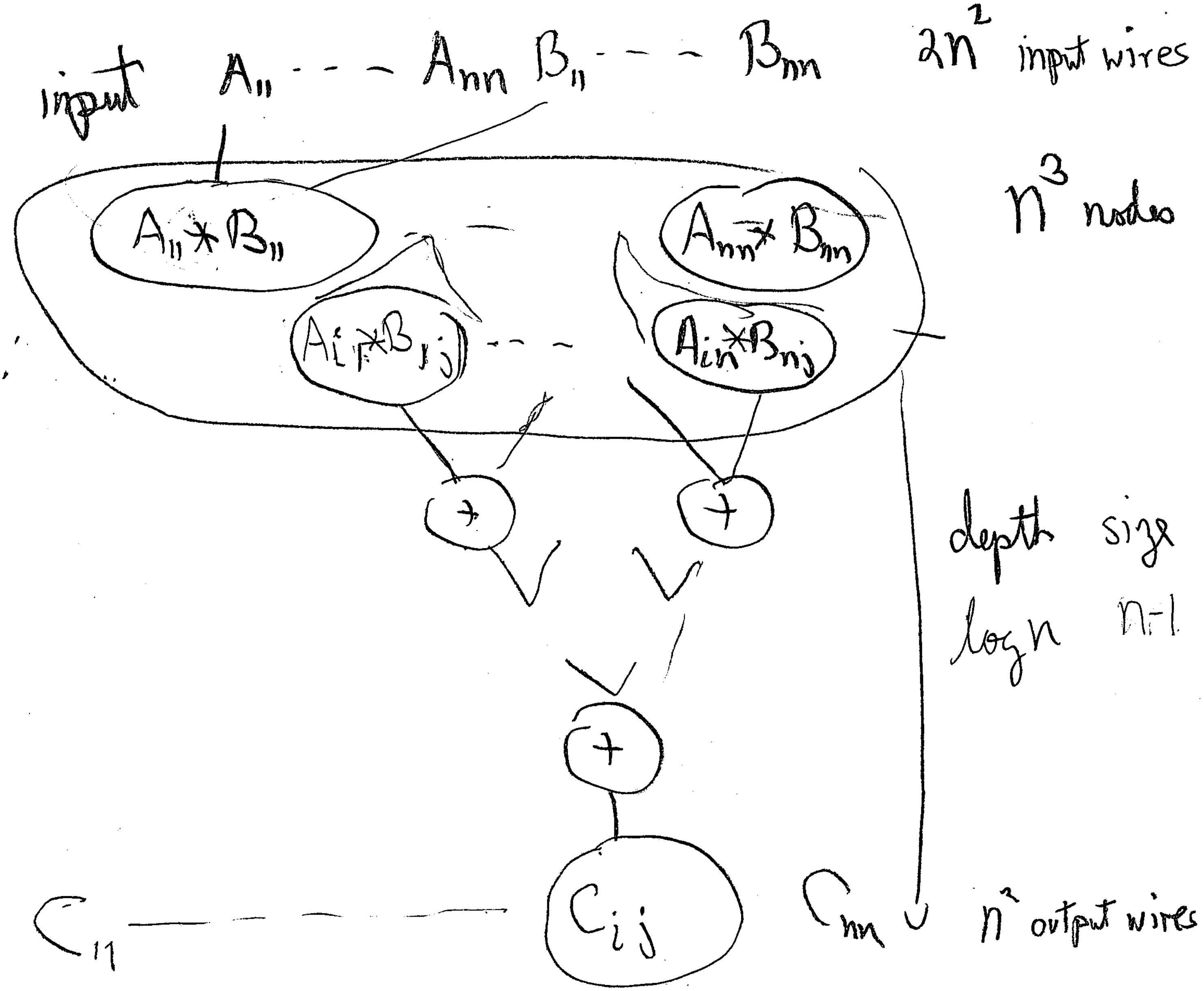
1) Constant fan in.
2) Arbitrary fan out.

Work $\equiv$ # nodes
Time $\equiv$ Longest path from input to output
Span (15-210)
Depth

Neural Nets

Naive Matrix Multiply in the Circuit Model

$$C_{ij} = \sum_{k=1}^{n} A_{ik} B_{kj} \qquad C = A \cdot B$$



Circuit Totals:

Work: $O(n^3)$

Time: $O(\log n)$

Naive MM on PRAM

P = #processors $T \equiv$ Parallel Time

1) 1-processor/node CRCW $P = O(n^3)$

$T = O(\log n)$

Note: Fan-out $\equiv$ reads

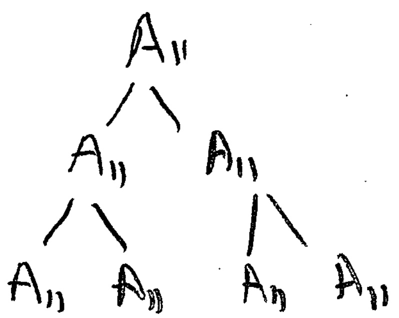
Fan-in $\equiv$ writes

---

2) Easy CREW (each processor reads its arguments)

3) EREW (Use binary tree to make copies)

eg



This increases depth by additive $\log n$.

PRAM Work $\equiv P \cdot T$

Pay for each processor for life of run.

So far $O(n^3 \log n)$ work Alg.

Claim: $O(n^3/\log n)$ processor $O(\log n)$ time

for Naive MM Alg.

Start with n^3 mult:

$\underbrace{A_{11} * B_{11} \text{ --- }}_{P_1} \quad \underbrace{\quad}_{} \quad \text{--- ---} \quad \underbrace{A_{nn} * B_{nn}}_{P_{n^3/\log n}}$

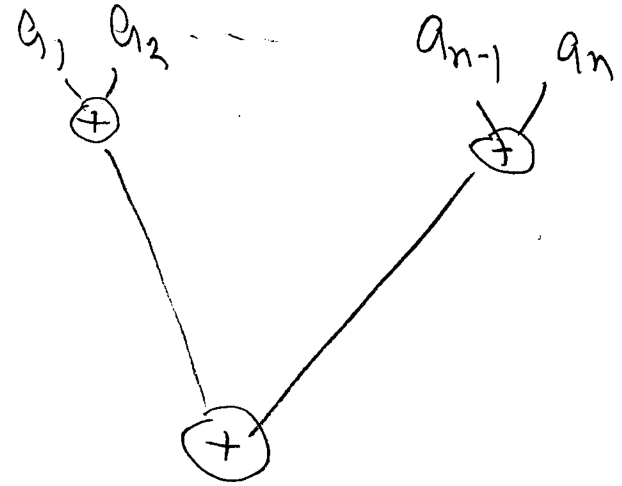
$\log n$ size blocks

Each P_i computes $\log n$ mult in $O(\log n)$ time.

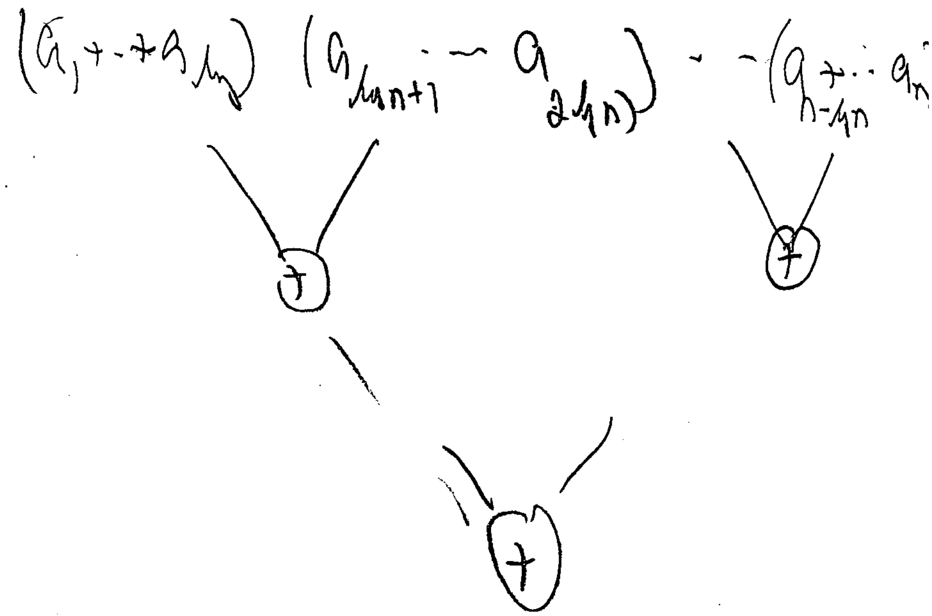
mult: $O(n^3/\log n) \equiv P$

$O(\log n) \equiv T$

Additions: eg



replace with:



Add n-numbers with $O(n/\log n)$ processors

$O(\log n)$ time.

<u>Finished Claim</u>

The Slow Down Principle:

Give parallel alg Processor P & Time T

$\forall p' \leq P$ run alg time $(P/p')T$ using p' processors.

Each processor simulates P/p' virtual processors

Strassen's Alg

Recall: Recurrance

$$MM(n) = \underbrace{7\,MM(n/2)}_{\text{7 recursive calls}} + \underbrace{cn^2}_{\text{matrix additions}}$$

Note Matrix addition

$O(n^2)$ work

$O(1)$ time

Time: $T(n) = T(n/2) + O(1)$

parallel calls

$O(\log n)$

Processors: $P(n) = 7P(n/2) + cn^2$

we must pay for each call!

$O(n^{2.81\cdots})$

Work: $O(n^{\log_2 7} \log n)$