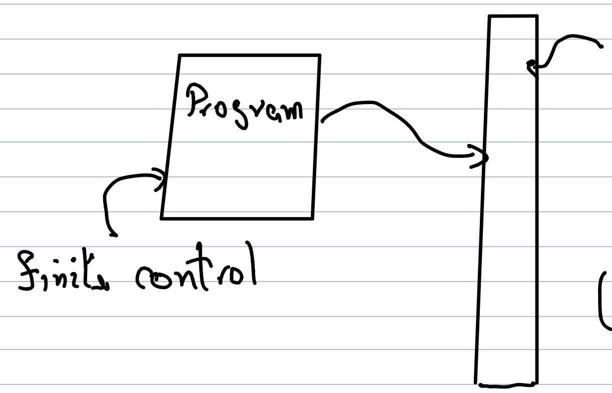
Parallel Algorithms

15-750
4/24/19

So far we have assumed the following model:

Random Access Model RAM



Unit time ops
(+, ×, ÷ log n bits)
(read/write in memory)

A central model to describe:
Graph Algorithms.
Algs that ignore locality

Other Models:
1) Agents/Ants on the graph.
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

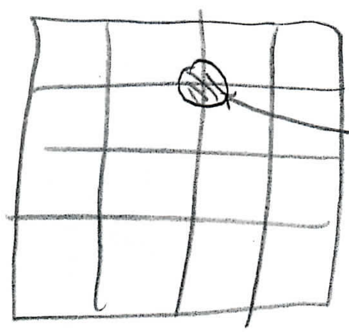
Parallel Models

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

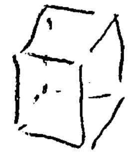
80's Klaus Sutner

Conway Game of Life

60's Edgar Codd

80's HT Kung

Highly connected models



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

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

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

2) Shuffle-exchange graph 1980's

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

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

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

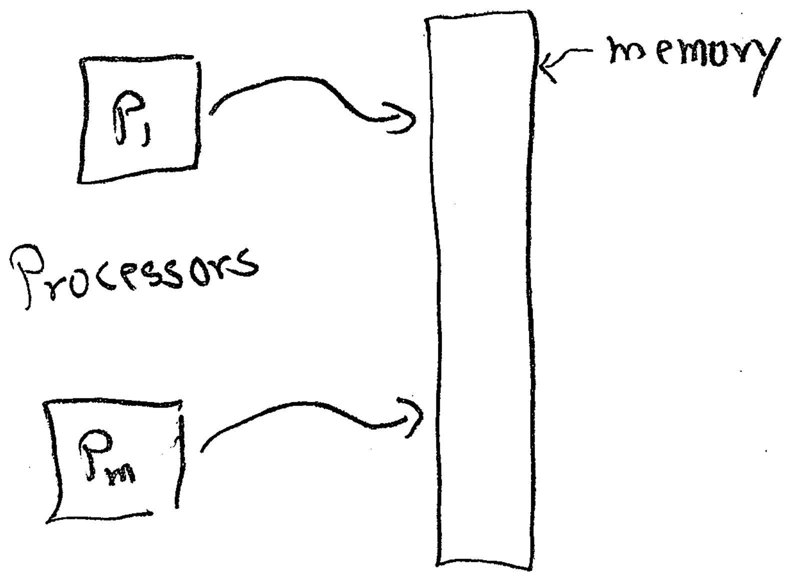
3) Randomly connected graphs

possible models of the brain!

(Valiant)

Shared memory models

1) PRAM (Parallel Random Access Machine)



Unit time ops

$+, \times, \div$

read / write

	Exclusive	Concurrent
Read		
Write		

ER EW

CR CW

Wyllie 1979

PRAM Issues:

What should be the effect of a CW on a EW machine?

1) Machine crashes!
2) Garbage Reads!

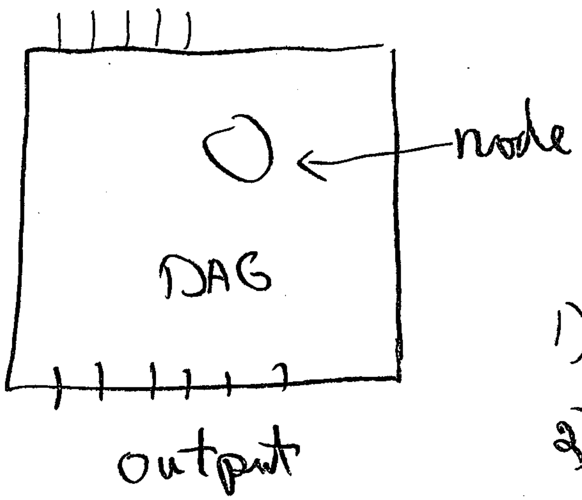
How is synchronization handled?

1) Synch after each unit of time!
2) Bulk Synchronous Parallel (BSP) Valiant.
3) None!

We will mostly use 1).

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.

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

Neural Nets

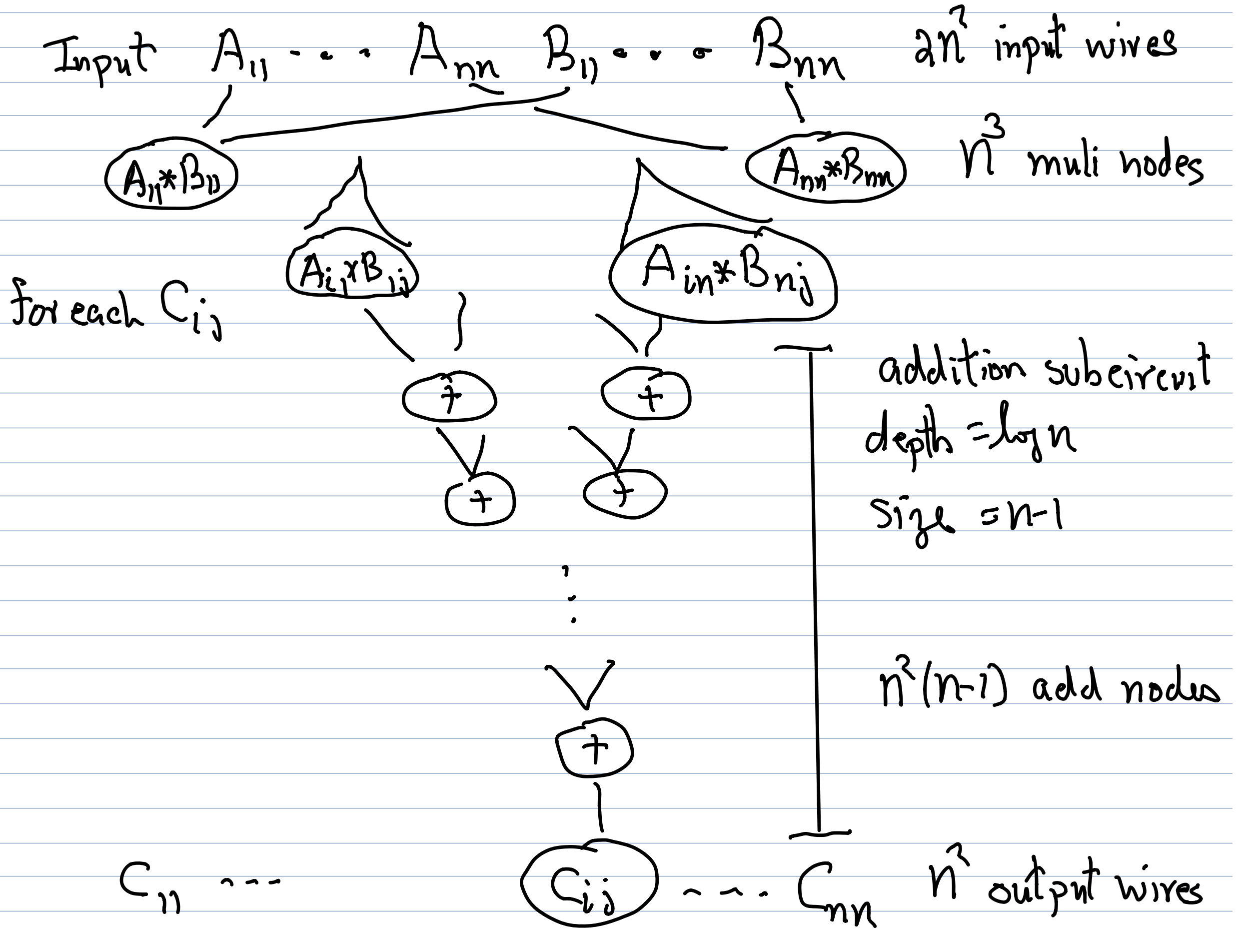
Deep Learning!!

Naive Matrix Multiply in the Circuit Model

Input $A^{n\times n}$ & $B^{n\times n}$

Output $C^{n\times n}$ where $C_{ij} = \sum_{k=1}^{n} A_{ik} B_{kj}$

Circuit



Circuit Total: work: $O(n^3)$

Time: $O(\log n)$

Lets simulate Naive Matrix-Multi on PRAM

Let P = # processors we request.
Let T = Parallel time they need.

First implementation

1) Request: 1 processor / node of circuit.
request a CRCW PRAM.

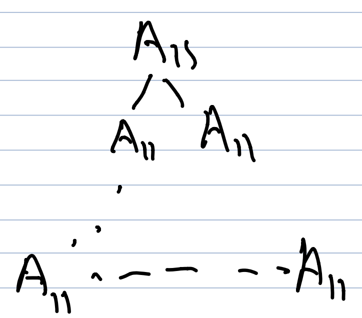
We get $P = O(n^3)$ $T = O(\log n)$ CRCW Alg.

---

2) If each processor reads it's arguments
we only need a CREW machine.

3) To get an EREW machine
We use binary tree to make copies
(These are new processors)

eg: We need n copies of A_{11}



Increase in depth by additive $\log n$

Processor increase $n^2(n-1)$ new write processors

PRAM Work

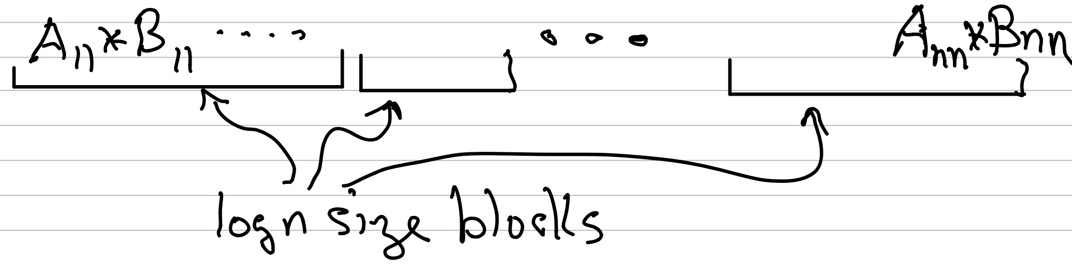
Def $P \equiv$ # of processor used for life of run
$T \equiv$ Total time.

Matrix Mult

So far $O(n^3) \equiv P$ & $T = O(\log n)$ or
$P \cdot T = O(n^3 \log n)$

Claim $O(n^3/\log n)$ processor
$O(\log n)$ time for Naive MM.

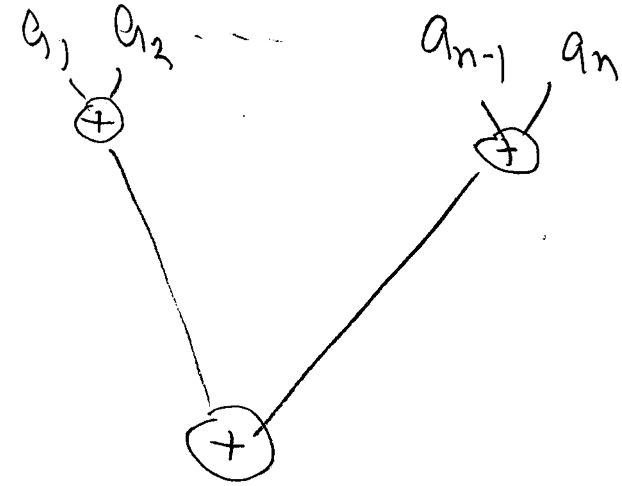
pf: Let's start with n^3 mult:



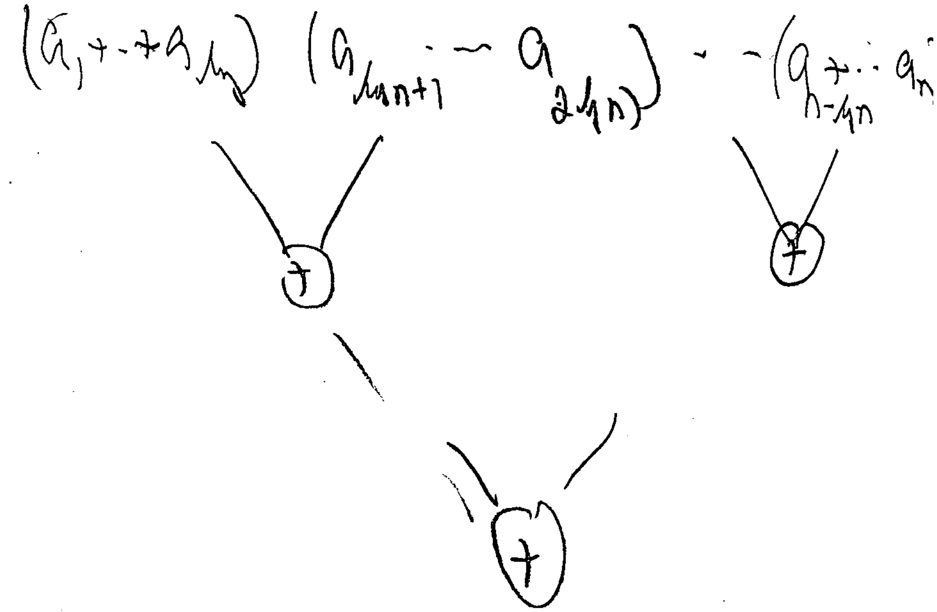
Each P_i computes $\log n$ mult in $O(\log n)$ time.

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

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:

Given an Parallel alg using T time & P processors.
$\forall P' \leq P$ $\exists$ parallel alg using
$(P/P')T$ time and only P' processors

pf Each real processor simulates P/P' virtual proc.

Lets implement Strassen's Alg on a PRAM

Let SMM = Strassen's Alg

Recall Recurrance

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

Parallel Strassen Analysis

Note: Matrix addition is
$O(n^2)$ processors
$O(1)$ time

Timing Recurrance:

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

we do parallel calls

Processor Recurrance:

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

we must hire processor for each call

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

$P \cdot T = \text{Work} = O(n^{\log_2 7} \log_2 n)$