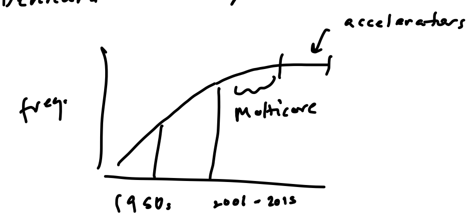
"How to have a bad research career"

- 3 homework - 30%
- project - 40%
- midterm - 25%
- 5% participation

Why study parallel algorithms and d.s.?

- Moore's law ending "Dennard scaling"



- parallelism & concurrency fundamental ideas for how to do work (eg. producing a complex object)

- superset of serial algorithms; understand new aspects of a problem by studying in parallel.

- Amount of data to analyze not slowing down.
  - genomics data
  - video/image data

(1) Multicore machines exist!

- p cores, $\underline{p\times \text{ faster}}$

  factor p speedup.

$$\frac{t_1}{t_p} \qquad\qquad \frac{t^*}{t_p} \leftarrow \text{best seq. running time}$$

$O(n^2)$ operations (work) $\qquad$ $O(n \log n)$ operations

- Early apps were large-scale simulation

1961 $\quad$ IBM Stretch:
1 MB memory

1972 $\quad$ Illiac IV: $\quad$ 4 CPUs $\quad$ 256 FPUs
512k memory $\quad$ 1 CPU $\quad$ 64 FPUs

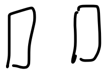
$\longrightarrow$ SIMD (single-instruction, multiple-data)

1973 Eispack: matrix eigenvalues

→ Lapack (Jack Dongarra)

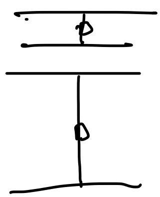
1976 CRAY-1

- vector processing capabilities

 $\oplus$ $\otimes$

<u>1912</u>: MSF algorithm (Börukke)

1968: Batcher Sort / Bitonic Sort



1 step

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

$O(\log^2 n)$ <u>delay</u> (# <u>steps</u>)

rounds

1975 : Valiant: Max, Merging and Sorting

- $O(\log\log n)$ rounds $\underbrace{\Omega(\log\log n)}_{n \text{ processors}}$ round

$\Omega(\log n)$

1978 : PRAM (Wylie)

1979 : Circuit models (P-completeness , NC)

1980s: Golden age.

90-95: other model (Asynch PRAM)

(BSP - Valiant)

predecessor of MapReduce

1995 Work / depth or Work / Span model.

Parallelism Winter

2010

Resurgance. Lot of interesting work.

Present

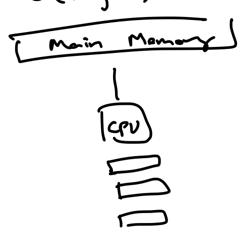
RAM: Random Access Machine;

- how does an algorithm behave when ~~data~~ data grows?

  $O(n^2)$ time vs $O(n \log n)$ time.

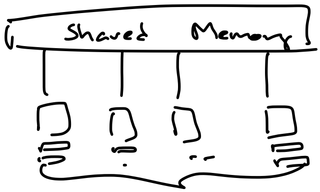
- +, x, ÷ unit time → $O(1)$
- load / store unit time
- Goal: compare algorithms asymptotically

- $O(\log n)$ bit words

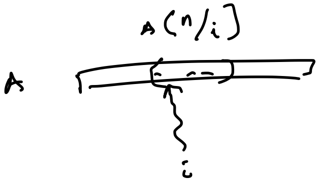


Parallel Models

PRAM Parallel Random Access Machine



p CPUs



- synchronous
  - works in lockstep
- not particularly realistic
  - fixed # processors
  - schedule tasks → processors?

Cost : - Running Time T (p processors)
- Processors p

total # instructions = PT = work

Sorting in $O\left(\underbrace{\frac{n\log n}{p}} + \underbrace{\log^2 n}\right)$ time

PRAM models: ER : can't read same location at the same time

EW : " write "

CR

CW

EREW , CREW , CRCW

↳ Common

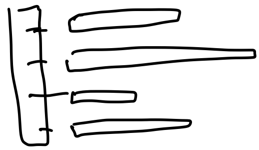
Arb

Priority

Connection Machine

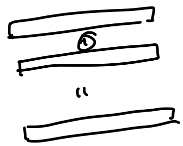
$\boxed{\text{Vector Models}}$

Guy Blelloch's thesis



vector memory

- instructions operate on vectors



merge, prefix-sum

Cost: Work (element complexity) $= \sum_{s_i \in \text{Steps}} \text{length}(v_i)$

Step complexity = # steps executed by the program

E.g. sorting in $O(\log^2 n)$ steps

$O(n \log n)$ work

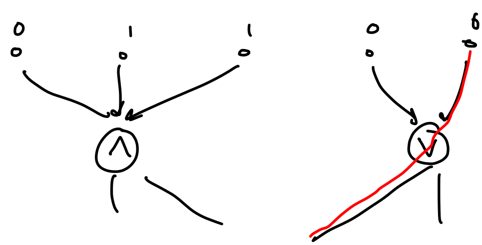
"Connection Machine" Book

NESL

$\boxed{\text{Circuit Model}}$ 1979 (Pippenger)

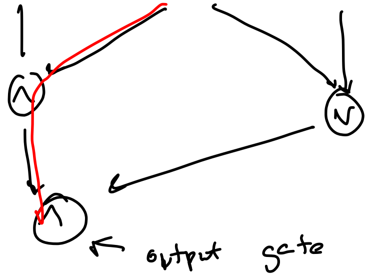
- view parallel comp. as a circuit (DAG)

$\wedge$ (and) and $\vee$ (or) gates



Input

$C_{\{n\}}$



Cost: - size = # gates

- depth

Sorting: (AKS network)

$O(n \log n)$ size, $O(\log n)$ depth

CRCW PRAM $\longrightarrow$ EREW PRAM

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

$O(\log n)$ depth $\longrightarrow$ AC circuits in same depth

$O(\log n)$ depth $\xrightarrow{O(\log n) \text{ depth loss}}$ NC circuits

---

NC^k : circuits with $O(\log^k n)$ depth (polynomial size)

$$NC = \bigcup_k NC^k$$