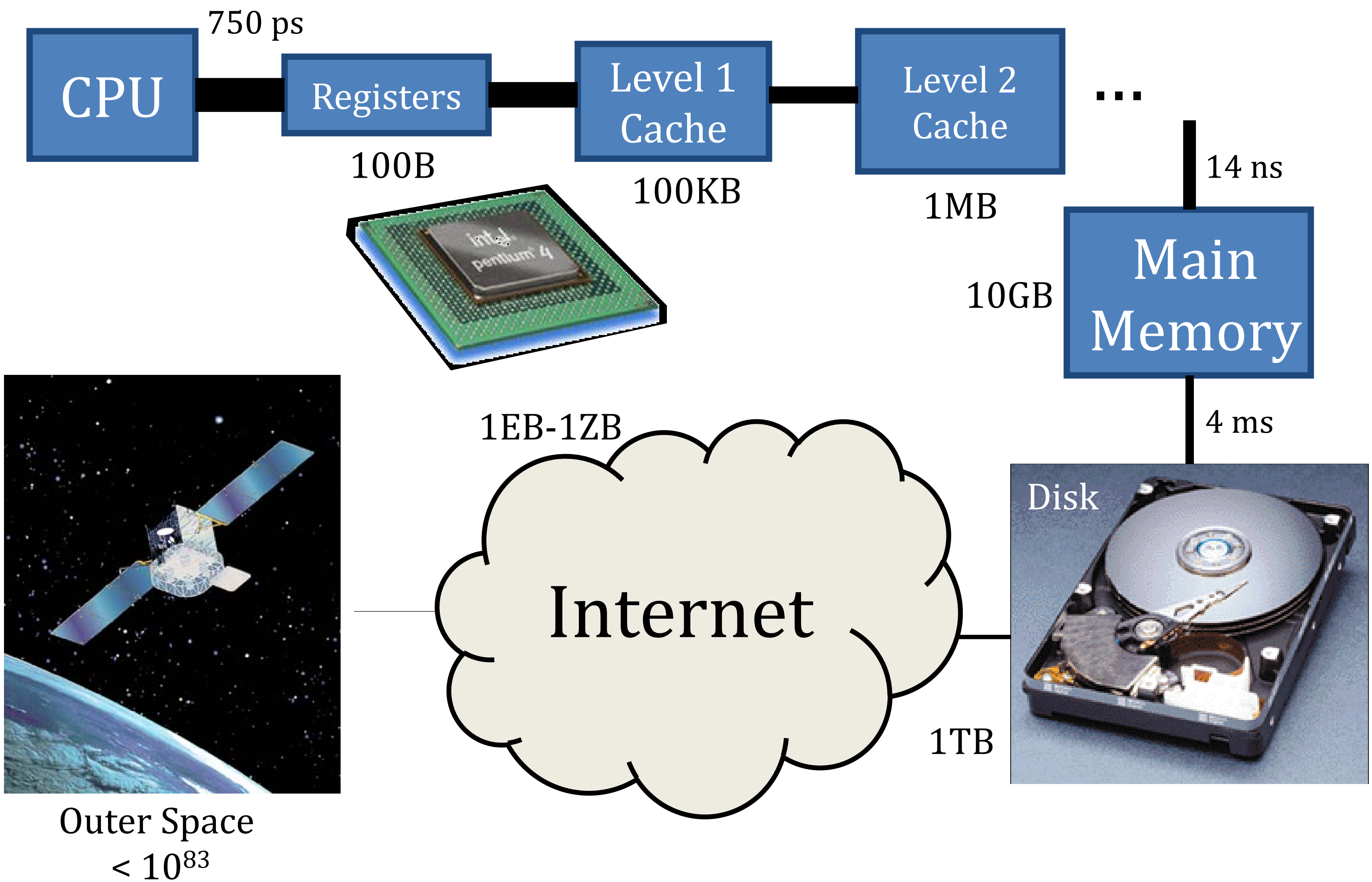
The History of I/O Models

Erik Demaine

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

madalgo
CENTER FOR MASSIVE DATA ALGORITHMICS

Memory Hierarchies in Practice

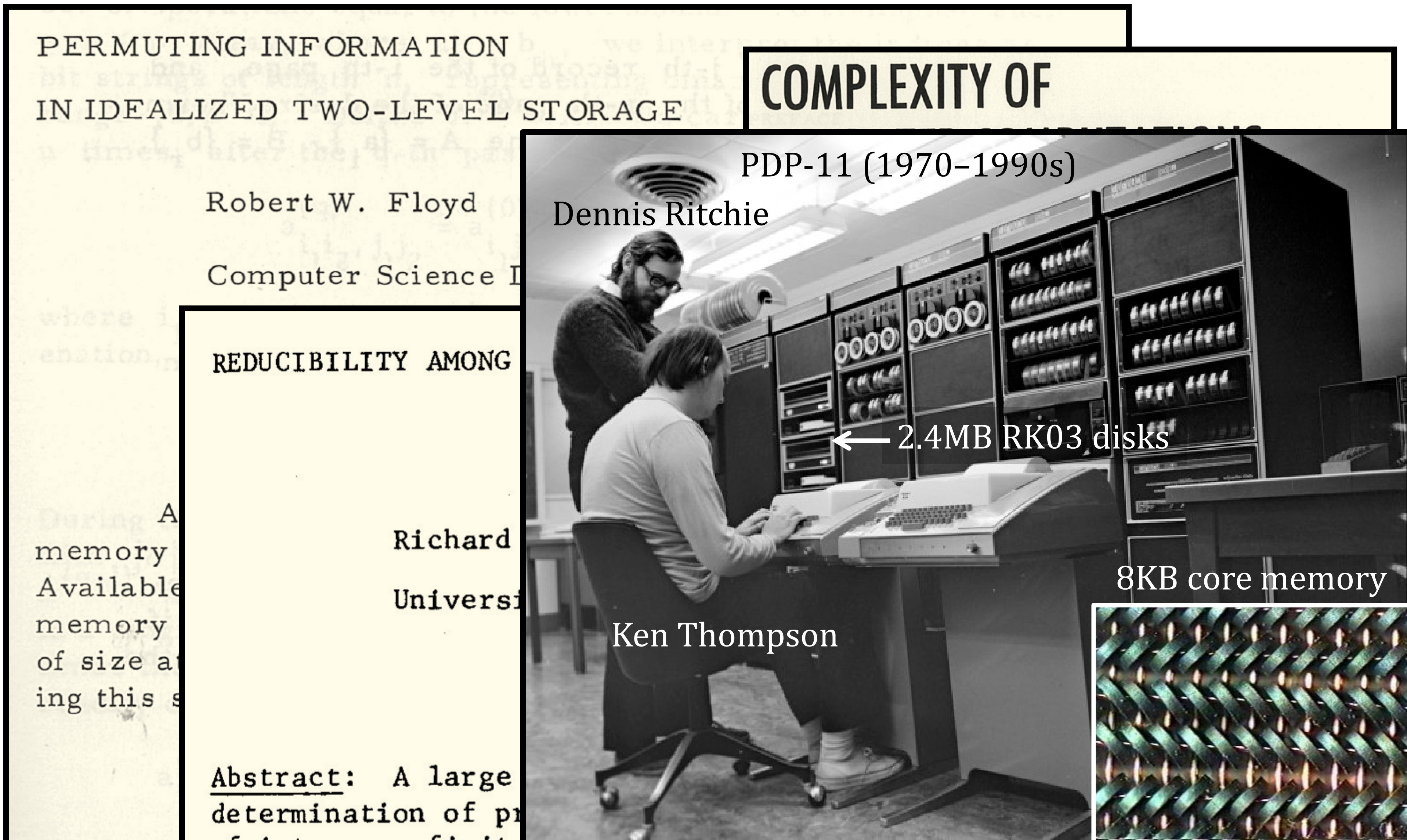


Models, Models, Models

Model	Year	Blocking	Caching	Levels	Simple
Idealized 2-level	1972	✓	✗	2	✓
Red-blue pebble	1981	✗	✓	2	✓−
External memory	1987	✓	✓	2	✓
HMM	1987	✗	✓	∞	✓
BT	1987	~	✓	∞	✓−
(U)MH	1990	✓	✓	∞	✗
Cache oblivious	1999	✓	✓	2–∞	✓+

Idealized Two-Level Storage

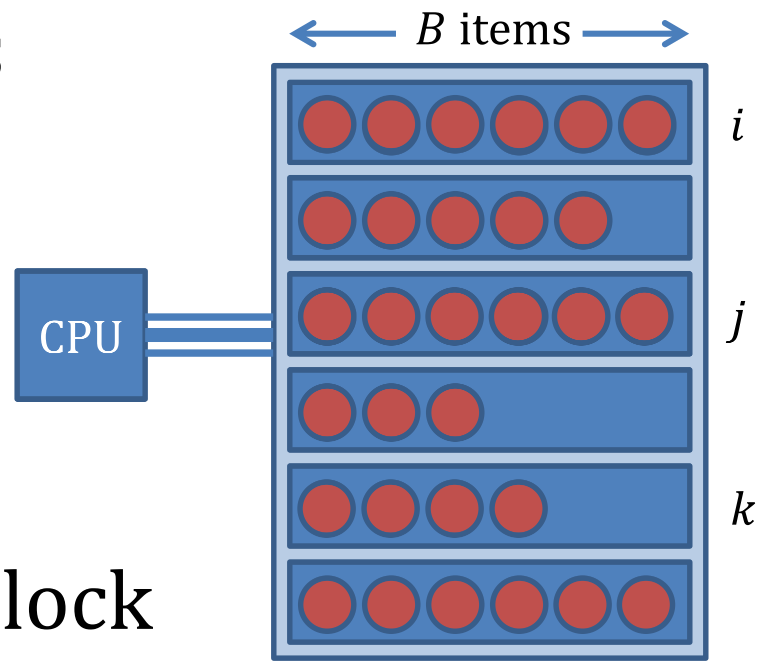
[Floyd — Complexity of Computer Computations 1972]


PERMUTING INFORMATION
IN IDEALIZED TWO-LEVEL STORAGE
Robert W. Floyd
Computer Science D
A
memory
Available
memory
of size a
ing this s
COMPLEXITY OF
REDUCIBILITY AMONG
Richard
Universi
Abstract: A large
determination of pr

Idealized Two-Level Storage

[Floyd — Complexity of Computer Computations 1972]

- RAM = blocks of $\leq B$ items
- **Block operation:**
  - Read up to B items from two blocks i, j
  - Write to third block k
- Ignore item order within block
  - CPU operations considered free
- Items are **indivisible**



Permutation Lower Bound

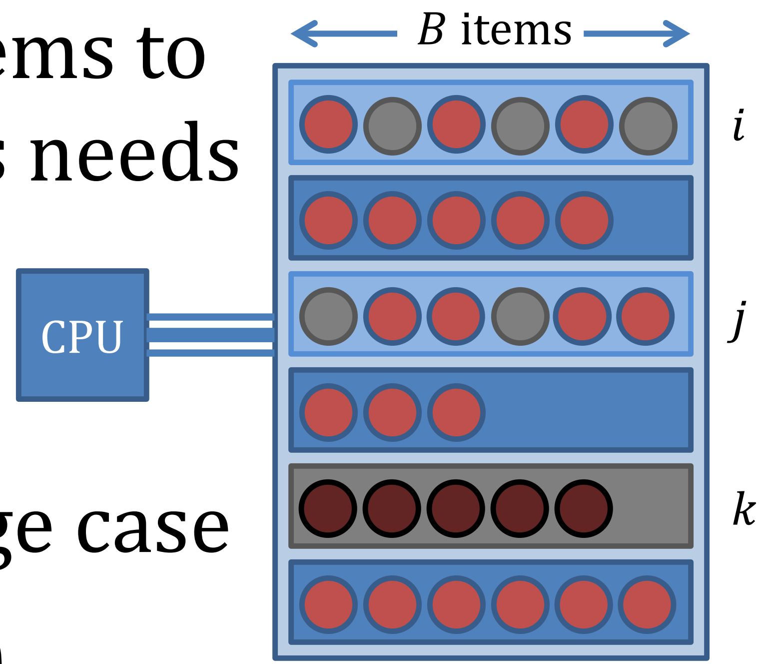
[Floyd — Complexity of Computer Computations 1972]

- Theorem: Permuting N items to N/B (full) specified blocks needs

$$\Omega\left(\frac{N}{B}\log B\right)$$

block operations, in average case

  - Assuming $\frac{N}{B} > B$ (**tall disk**)

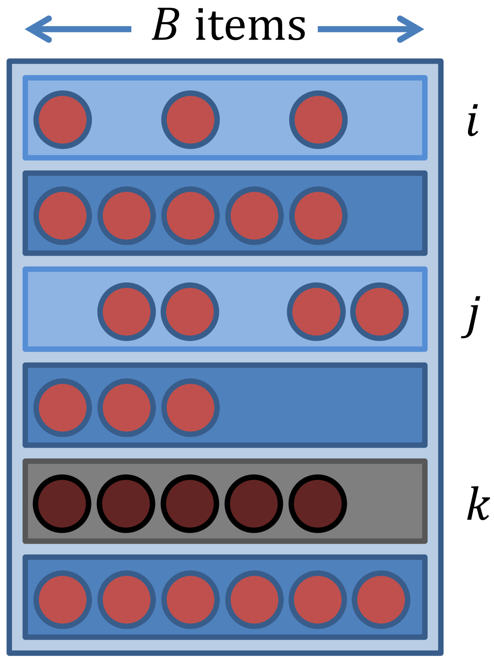


- **Simplified model:** Move items instead of copy
  - Equivalence: Follow item's path from start to finish

Permutation Lower Bound

[Floyd — Complexity of Computer Computations 1972]

- Potential: $\Phi = \sum_{i,j} \underbrace{n_{ij}}_{\text{\# items in block } i \text{ destined for block } j} \log n_{ij}$
  - Maximized in target configuration of full blocks ($n_{ii}=B$): $\Phi = N \log B$
  - Random configuration with $\frac{N}{B} > B$ has $\mathrm{E}[n_{ij}] = O(1) \Rightarrow \mathrm{E}[\Phi] = O(N)$
  - Claim: Block operation increases Φ by $\leq B$
  - $\Rightarrow$ Number of block operations $\geq \frac{N \log B - O(N)}{B}$



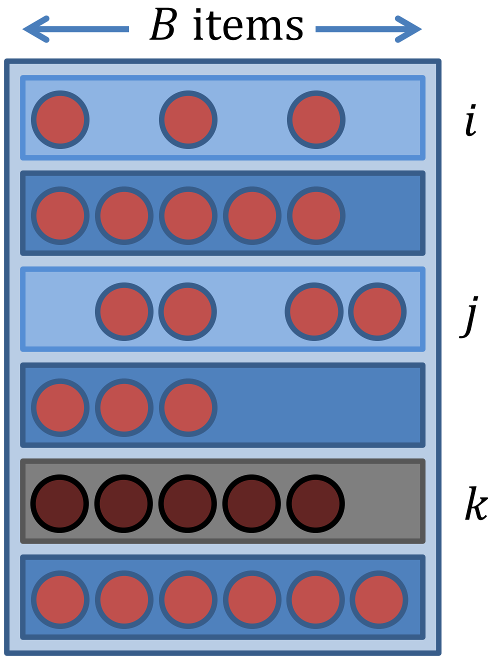
Permutation Lower Bound

[Floyd — Complexity of Computer Computations 1972]

- Potential: $\Phi = \sum_{i,j} \underbrace{n_{ij}} \log n_{ij}$

  # items in block i destined for block j

  - Maximized in target configuration of full blocks ($n_{ii}=B$): $\Phi = N \log B$
  - Random configuration with $\frac{N}{B} > B$ has $\mathrm{E}[n_{ij}] = O(1) \Rightarrow \mathrm{E}[\Phi] = O(N)$
  - Claim: Block operation increases Φ by $\leq B$
    - Fact: $(x+y)\log(x+y) \leq x \log x + y \log y + x + y$
    - So combining groups x & y increases Φ by $\leq x + y$



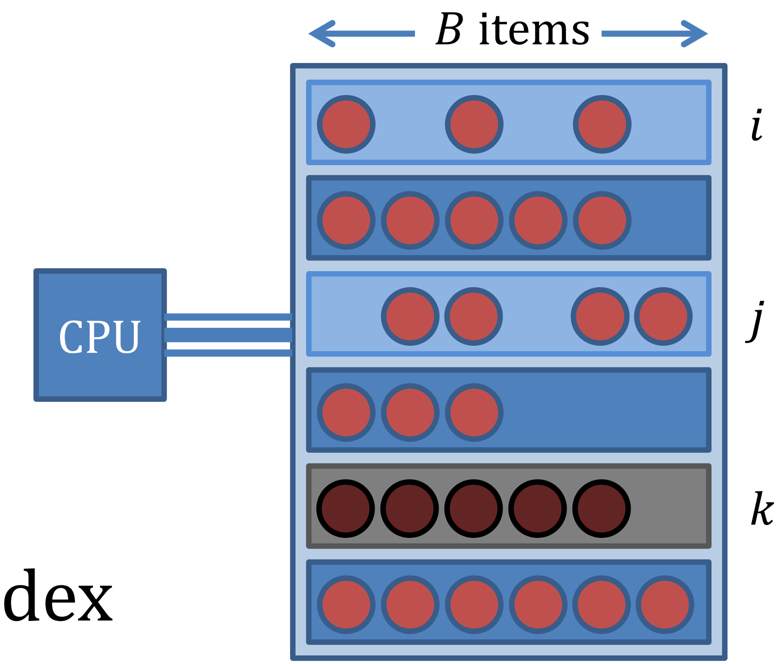
Permutation Bounds

[Floyd — Complexity of Computer Computations 1972]

- <u>Theorem:</u> $\Omega\left(\frac{N}{B}\log B\right)$
  - Tight for $B = O(1)$
- <u>Theorem:</u> $O\left(\frac{N}{B}\log\frac{N}{B}\right)$
  - Similar to radix sort,
    where key = target block index
  - <u>Accidental claim:</u> tight for all B $\left(< \frac{N}{B}\right)$

    > By information theoretic considerations, most permutations with $w > p$ require $O(w(\log_2 p + \log_2 w))$ operations.

  - <u>We will see:</u> tight for $B > \log\frac{N}{B}$

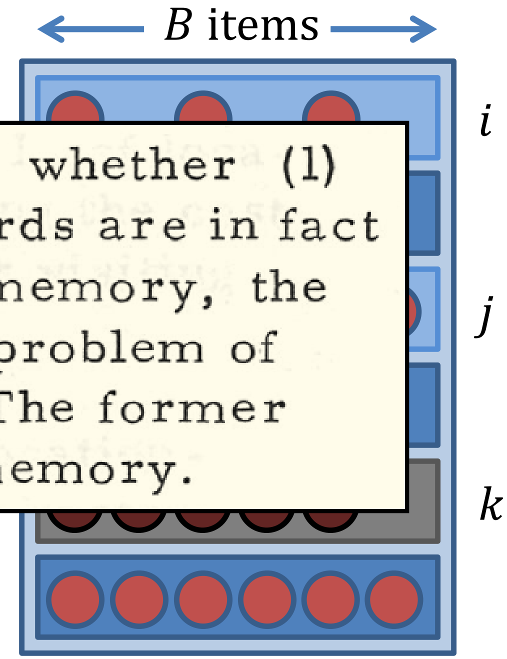


Idealized Two-Level Storage

[Floyd — Complexity of Computer Computations 1972]

- External memory & word RAM:

> Obviously the above results apply equally, whether (1) the pages are blocks on a disc or drum, the records are in fact records, or (2) the pages are words of internal memory, the records are bits. The latter corresponds to the problem of transposing a Boolean matrix in core memory. The former corresponds to tag sorting of records on a disc memory.

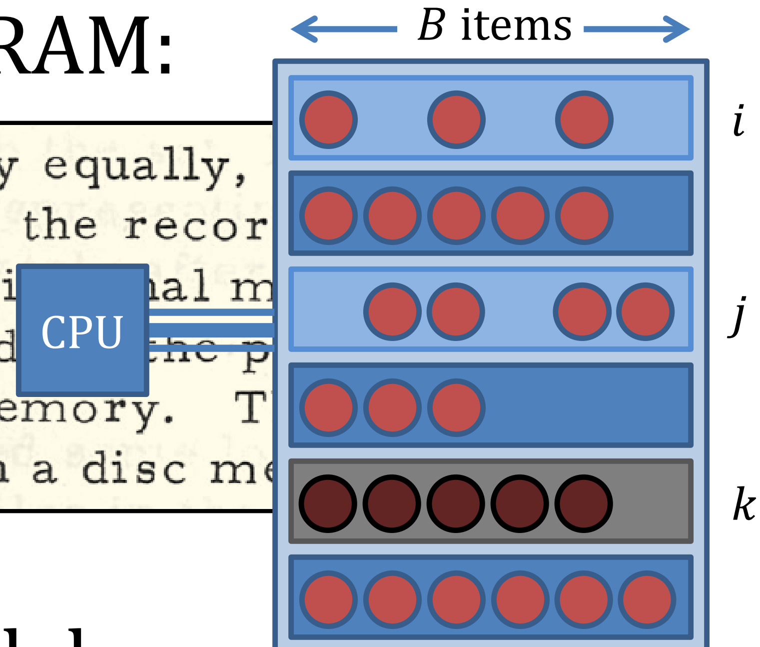


Idealized Two-Level Storage

[Floyd — Complexity of Computer Computations 1972]

- External memory & word RAM:

Obviously the above results apply equally,
the pages are blocks on a disc or drum, the recor
records, or (2) the pages are words of i al m
records are bits. The latter correspond the p
transposing a Boolean matrix in core memory. T
corresponds to tag sorting of records on a disc me

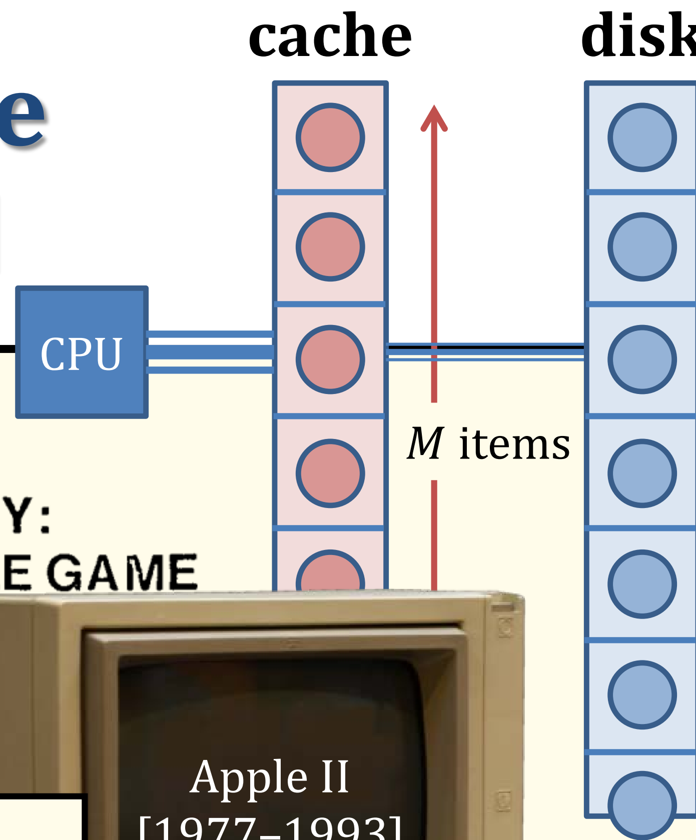


- Foreshadowing future models:

The above results apply to an idealized three-address machine. Work is in progress attempting to apply a similar analysis to idealized single-address machines with fast memories capable of holding two or more pages.

Red-Blue Pebble Game

[Hong & Kung — STOC 1981]



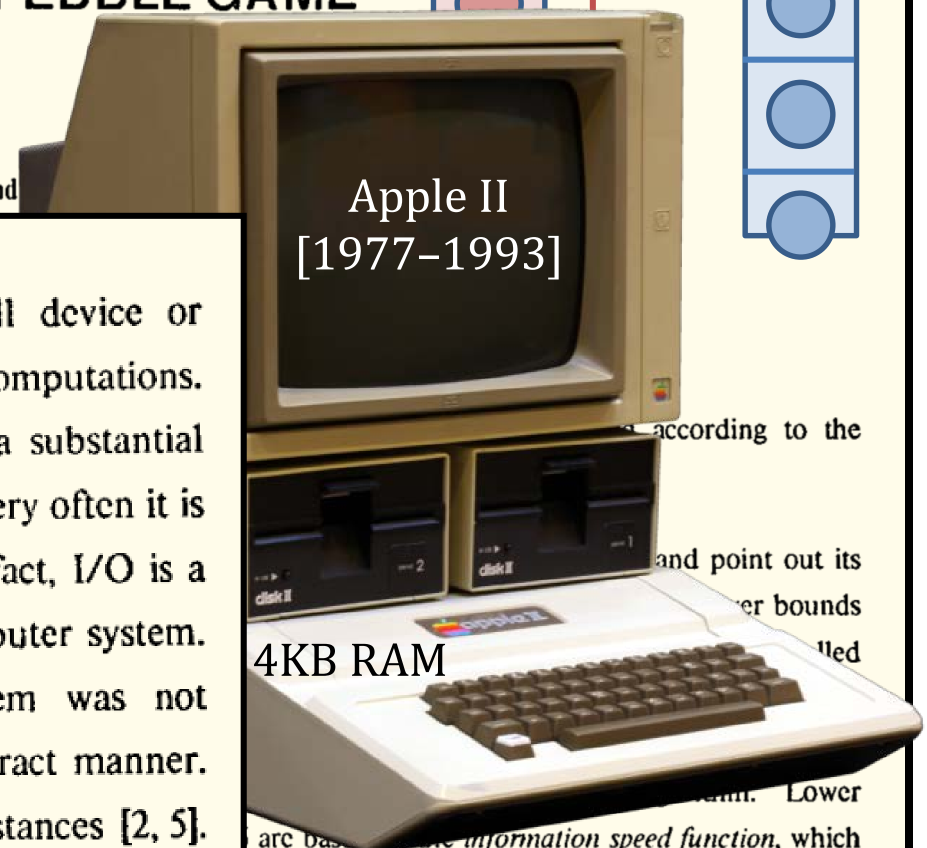
6-333.

**I/O COMPLEXITY:
THE RED-BLUE PEBBLE GAME**

Hong, Jia-Wei and

1. Introduction

When a large computation is performed on a small device or memory, the computation must be decomposed into subcomputations. Executing subcomputations one at a time may require a substantial amount of I/O to store or retrieve intermediate results. Very often it is the I/O that dominates the speed of a computation. In fact, I/O is a typical bottleneck for performance at all levels of a computer system. However, to the authors' knowledge the I/O problem was not previously modelled or studied in any systematic or abstract manner. Similar problems were studied only in a few isolated instances [2, 5].



according to the

and point out its

er bounds

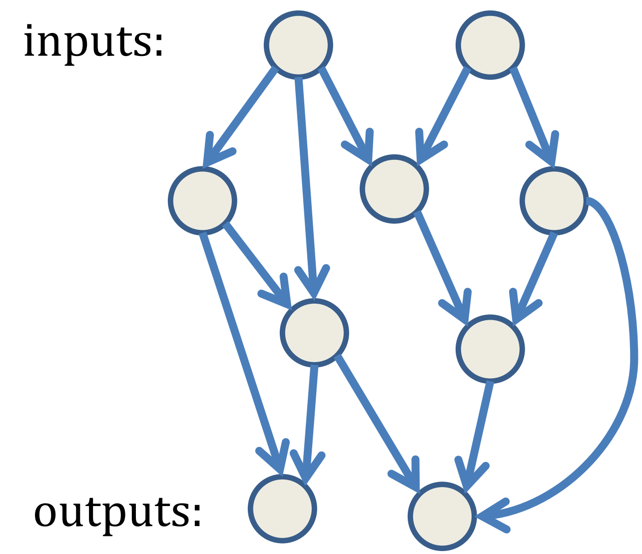
lled

Lower

are ba... *information speed function*, which

Pebble Game
[Hopcroft, Paul, Valiant — J. ACM 1977]

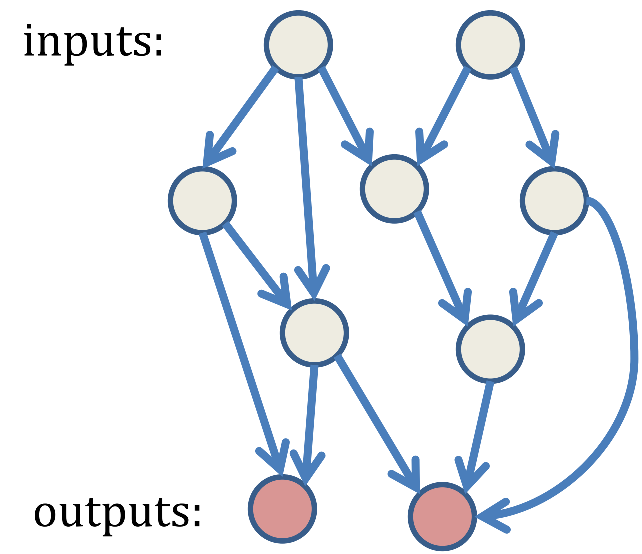
- View computation as DAG of data dependencies
- **Pebble** = "in memory"
- <u>Moves:</u>
  - Place pebble on node if all predecessors have a pebble
  - Remove pebble from node
- <u>Goal:</u> Pebbles on all output nodes
  - Minimize maximum number of pebbles over time



Pebble Game

[Hopcroft, Paul, Valiant — J. ACM 1977]

- Theorem: Any DAG can be "executed" using $O\left(\frac{n}{\log n}\right)$ maximum pebbles



- Corollary:

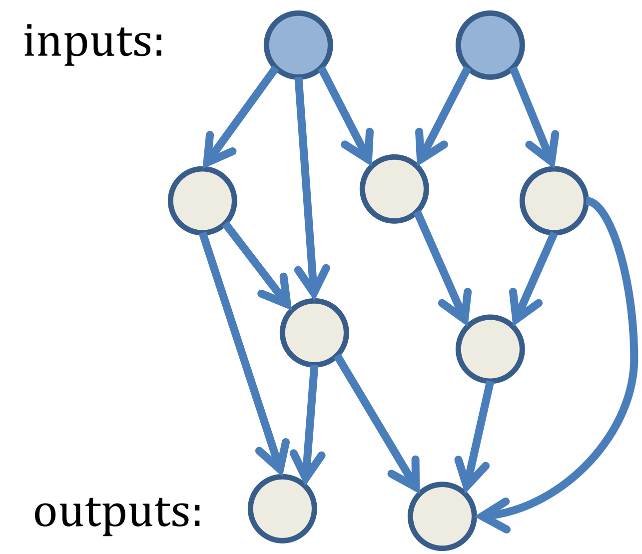
$$\mathrm{DTIME}(t) \subseteq \mathrm{DSPACE}\left(\frac{t}{\log t}\right)$$

Red-Blue Pebble Game

[Hong & Kung — STOC 1981]

- **Red pebble** = "in cache"
- **Blue pebble** = "on disk"
- Moves:
  - Place *red* pebble on node if all predecessors have red pebble
  - Remove pebble from node
  - **Write:** Red pebble → blue pebble
  - **Read:** Blue pebble → red pebble

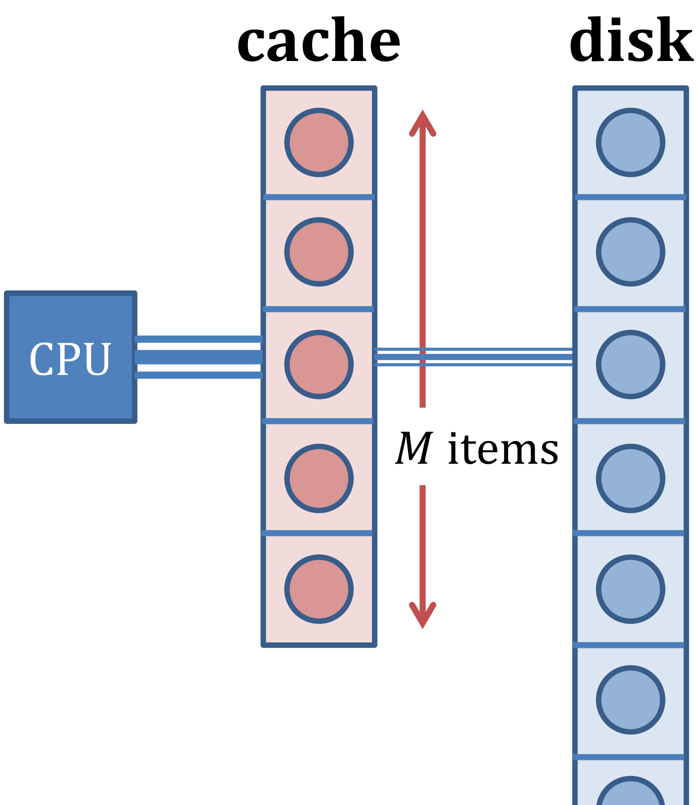
  minimize
- Goal: Blue inputs to blue outputs
  - $\leq M$ red pebbles at any time

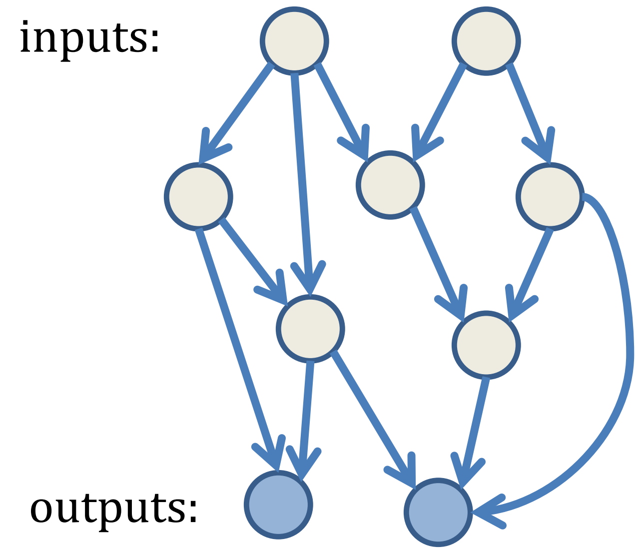


Red-Blue Pebble Game

[Hong & Kung — STOC 1981]

- **Red pebble** = "in cache"
- **Blue pebble** = "on disk"

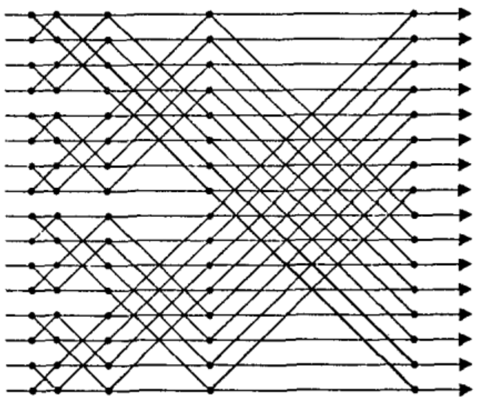


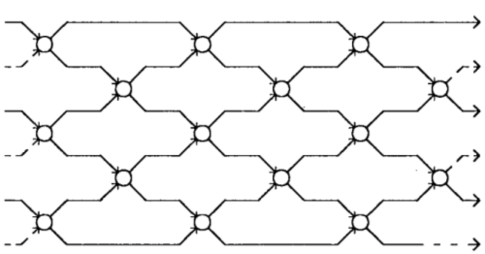


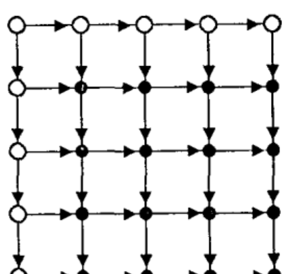
minimize number of cache ↔ disk **I/Os** (**memory transfers**)

Red-Blue Pebble Game Results

[Hong & Kung — STOC 1981]







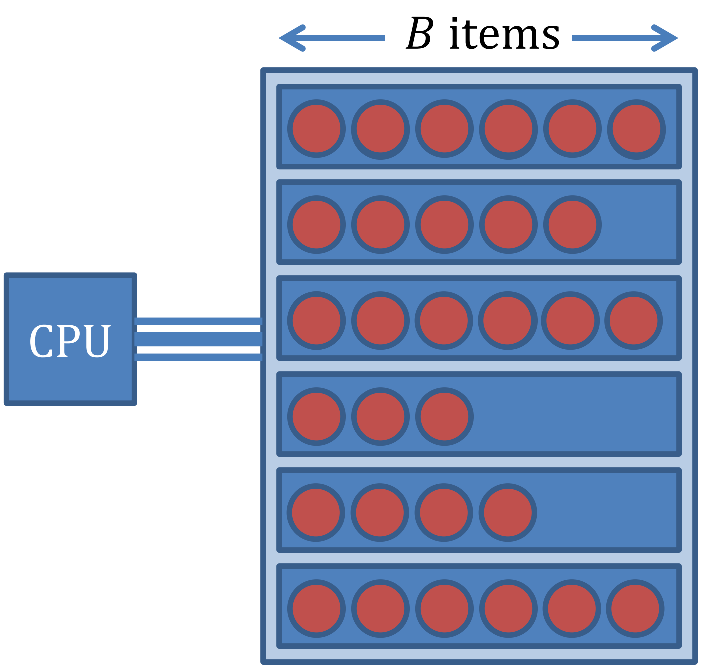
Computation DAG	Memory Transfers	Speedup
Fast Fourier Transform (FFT)	$\Theta(N \log_M N)$	$\Theta(\log M)$
Ordinary matrix-vector multiplication	$\Theta\left(\frac{N^2}{M}\right)$	$\Theta(M)$
Ordinary matrix-matrix multiplication	$\Theta\left(\frac{N^3}{\sqrt{M}}\right)$	$\Theta(\sqrt{M})$
Odd-even transposition sort	$\Theta\left(\frac{N^2}{M}\right)$	$\Theta(M)$
$\underbrace{N \times N \times \cdots \times N}_{d}$ grid	$\Omega\left(\frac{N^d}{M^{1/(d-1)}}\right)$	$\Theta\left(M^{1/(d-1)}\right)$

Comparison

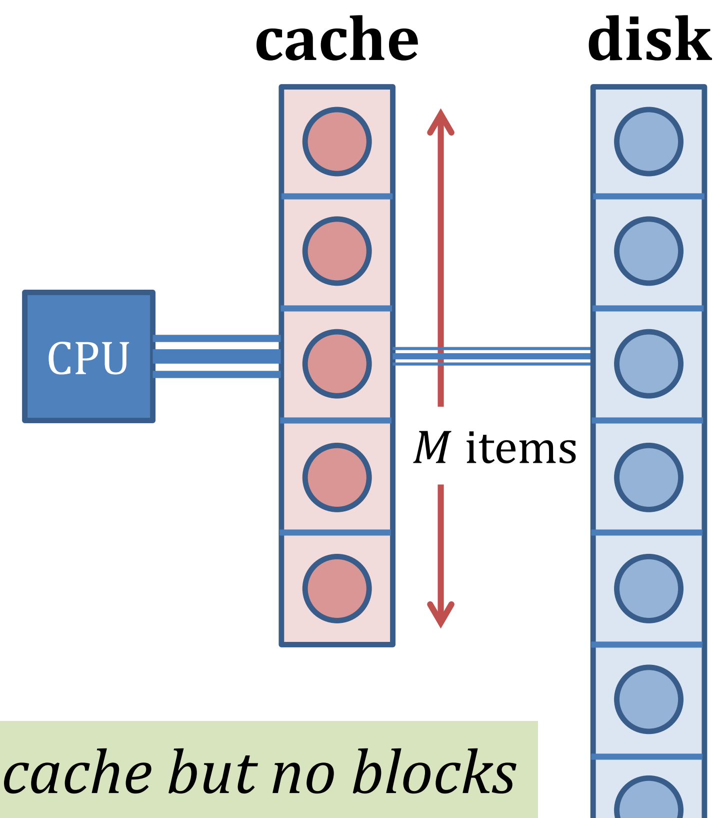
Idealized two-level storage
[Floyd 1972]

VS

Red-blue pebble game
[Hong & Kung 1981]



blocks but no cache

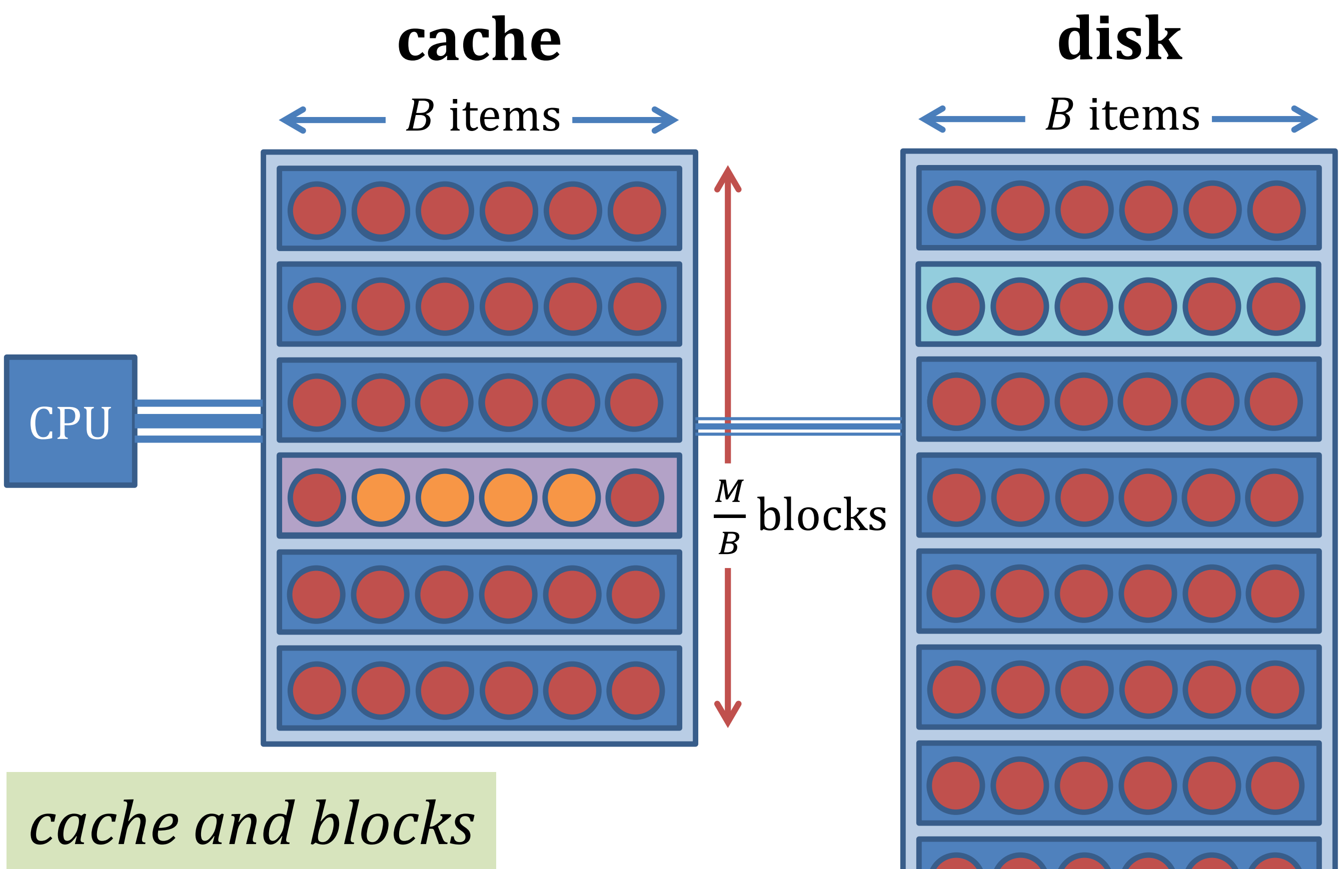


cache but no blocks

I/O Model

[Aggarwal & Vitter — ICALP 1987, C. ACM 1988]

- AKA: **External Memory Model**, Disk Access Model
- Goal: Minimize number of **I/Os (memory transfers)**

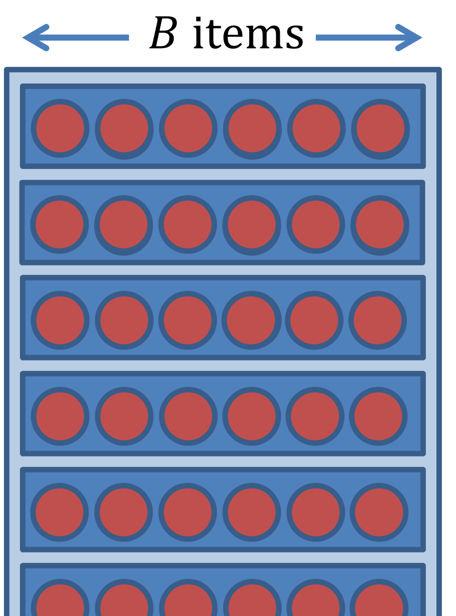


cache and blocks

Scanning

[Aggarwal & Vitter — ICALP 1987, C. ACM 1988]

- Visiting N elements in order costs $O\left(1+\frac{N}{B}\right)$ memory transfers
- More generally, can run $\leq \frac{M}{B}$ parallel scans, keeping 1 block per scan in cache
  - E.g., merge $O\left(\frac{M}{B}\right)$ lists of total size N in $O\left(1+\frac{N}{B}\right)$ memory transfers



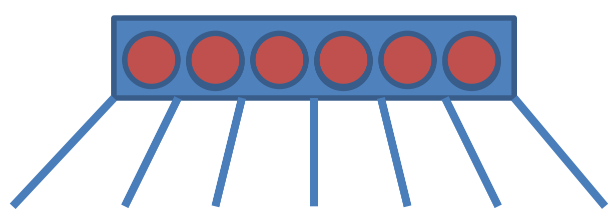
Practical Scanning [Arge]

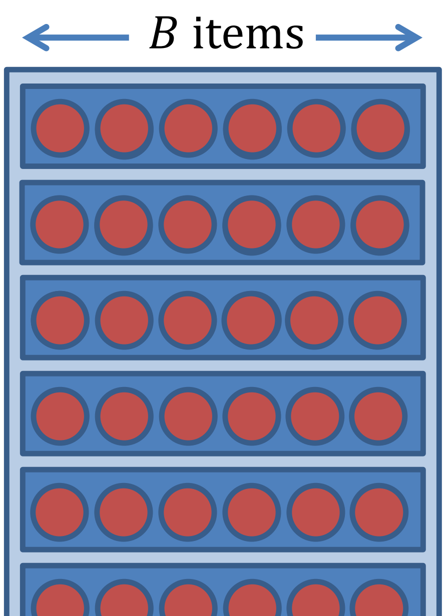
- Does the B factor matter?
  - Should I presort my linked list before traversal?
- Example:
  - $N = 256{,}000{,}000 \sim 1\text{GB}$
  - $B = 8{,}000 \sim 32\text{KB}$ (small)
  - 1ms disk access time (small)

  - N memory transfers take $256{,}000$ sec $\approx$ **71 hours**
  - $\frac{N}{B}$ memory transfers take $256/8 =$ **32 seconds**

Searching
[Aggarwal & Vitter — ICALP 1987, C. ACM 1988]

- Finding an element x among N items requires $\Theta(\log_{B+1} N)$ memory transfers
- Lower bound: **(comparison model)**
  - Each block reveals where x fits among B items
  - $\Rightarrow$ Learn $\leq \log(B+1)$ bits per read
  - Need $\log(N+1)$ bits
- Upper bound:
  - B-tree
  - Insert & delete in $O(\log_{B+1} N)$

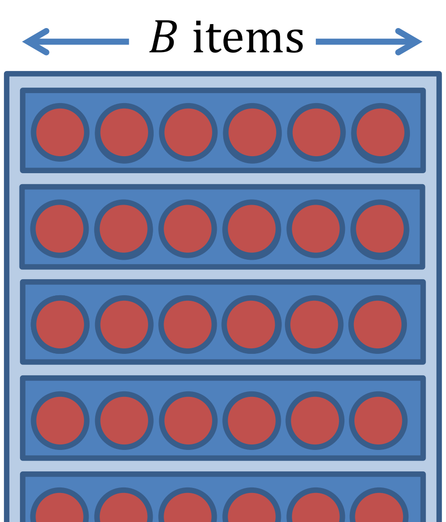




Sorting and Permutation

[Aggarwal & Vitter — ICALP 1987, C. ACM 1988]

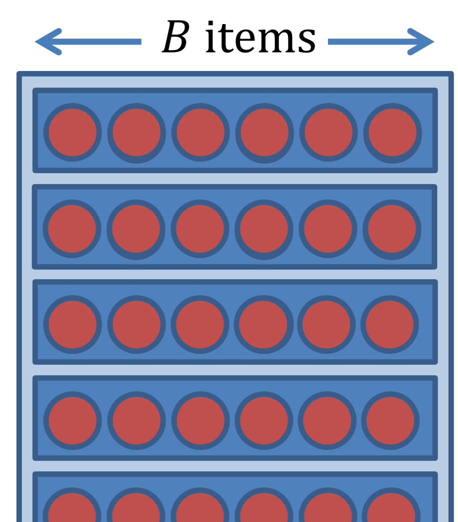
- **Sorting bound:** $\Theta\left(\frac{N}{B}\log_{M/B}\frac{N}{B}\right)$
- **Permutation bound:** $\Theta\left(\min\left\{N, \frac{N}{B}\log_{M/B}\frac{N}{B}\right\}\right)$
  - Either sort or use naïve RAM algorithm
  - Solves Floyd's two-level storage problem ($M = 3B$)



Sorting Lower Bound

[Aggarwal & Vitter — ICALP 1987, C. ACM 1988]

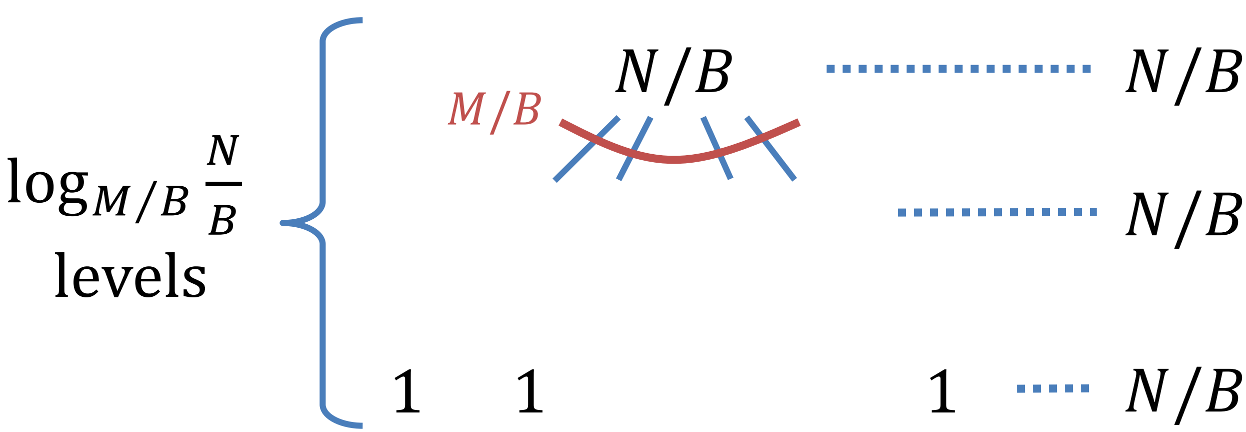
- **Sorting bound:** $\Omega\left(\frac{N}{B}\log_{M/B}\frac{N}{B}\right)$
  - Always keep cache sorted (free)
  - Might as well presort each block
  - Upon reading a block, learn how those B items fit amongst M items in cache
  - $\Rightarrow$ Learn $\lg\binom{M+B}{B} \sim B\lg\frac{M}{B}$ bits
  - Need $\lg N! \sim N\lg N$ bits
  - Know $N\lg B$ bits from block presort

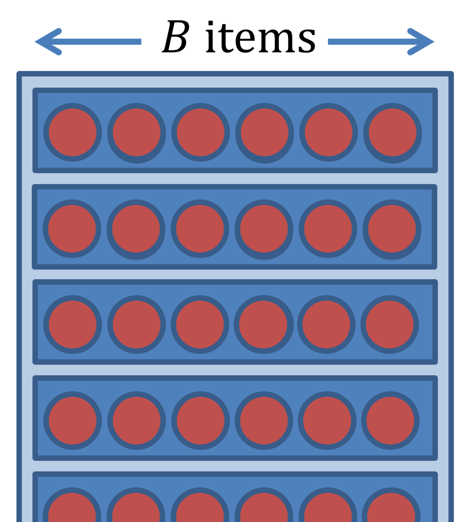


Sorting Upper Bound

[Aggarwal & Vitter — ICALP 1987, C. ACM 1988]

- **Sorting bound:** $O\left(\frac{N}{B}\log_{M/B}\frac{N}{B}\right)$
  - $O\left(\frac{M}{B}\right)$-way **mergesort**
  - $T(N) = \frac{M}{B} T\left(N/\frac{M}{B}\right) + O\left(1 + \frac{N}{B}\right)$
  - $T(B) = O(1)$



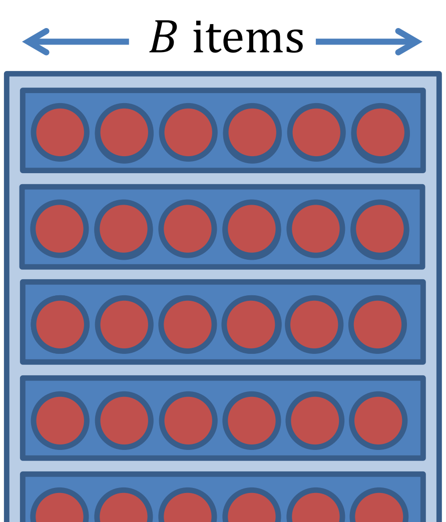


Distribution Sort

[Aggarwal & Vitter — ICALP 1987, C. ACM 1988]

- $\sqrt{M/B}$-way quicksort

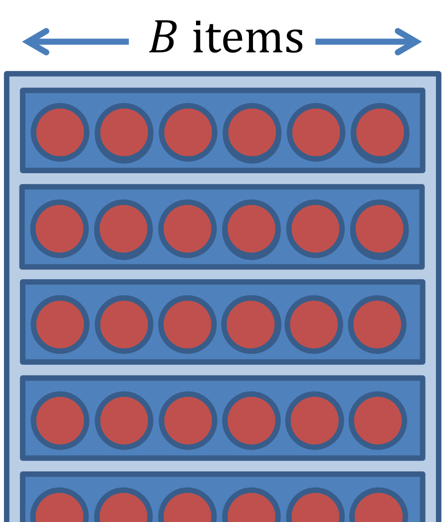
1. Find $\sqrt{M/B}$ partition elements, roughly evenly spaced
2. Partition array into $\sqrt{M/B}+1$ pieces
   - Scan: $O\left(\frac{N}{B}\right)$ memory transfers
3. Recurse
   - Same recurrence as mergesort



Distribution Sort Partitioning

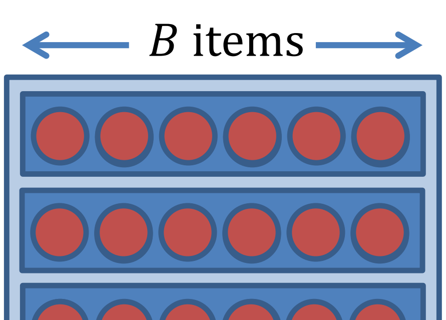
[Aggarwal & Vitter — ICALP 1987, C. ACM 1988]

1. For first, second, … interval of M items:
   - Sort in $O(M/B)$ memory transfers
   - Sample every $\frac{1}{4}\sqrt{M/B}$ th item
   - Total sample: $4N/\sqrt{M/B}$ items
2. For $i = 1,2, \ldots, \sqrt{M/B}$:
   - Run **linear-time selection** to find sample element at $i/\sqrt{M/B}$ fraction
   - Cost: $O\left(\left(\frac{N}{\sqrt{M/B}}\right)/B\right)$ each
   - Total: $O(N/B)$ memory transf.



Random vs. Sequential I/Os [Farach, Ferragina, Muthukrishnan — FOCS 1998]

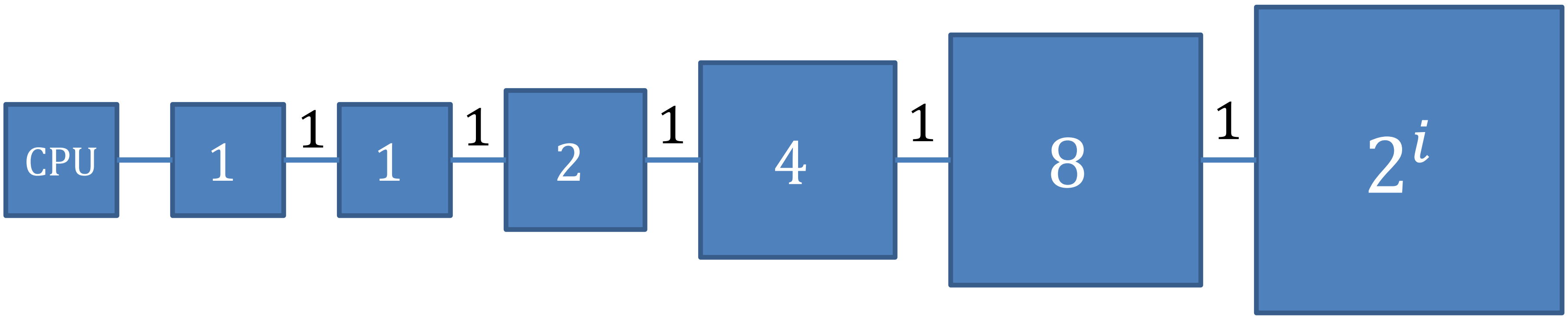
- **Sequential** memory transfers are part of bulk read/write of $\Theta(M)$ items
- **Random** memory transfer otherwise
- **Sorting:**
  - 2-way mergesort achieves $O\left(\frac{N}{B}\log\frac{N}{B}\right)$ sequential
  - $o\left(\frac{N}{B}\log_{M/B}\frac{N}{B}\right)$ random implies $\Omega\left(\frac{N}{B}\log\frac{N}{B}\right)$ total
- Same trade-off for suffix-tree construction



Hierarchical Memory Model (HMM)

[Aggarwal, Alpern, Chandra, Snir — STOC 1987]

- Nonuniform-cost RAM:
  - Accessing memory location x costs $f(x) = \lceil \log x \rceil$



"particularly simple model of computation that mimics the behavior of a memory hierarchy consisting of increasingly larger amounts of slower memory"

Why $f(x) = \log x$? [Mead & Conway 1980]

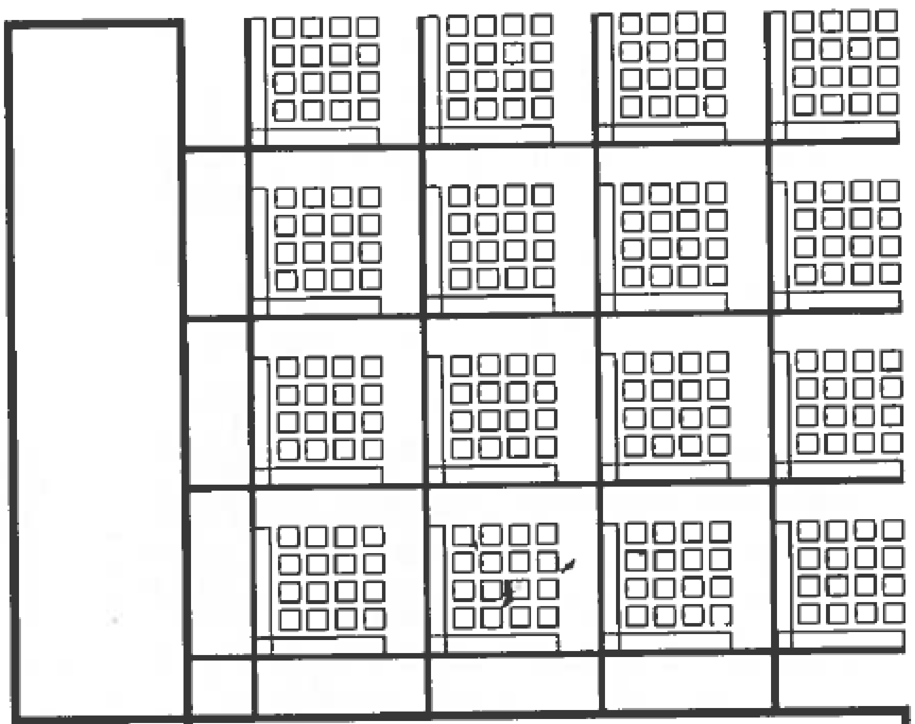
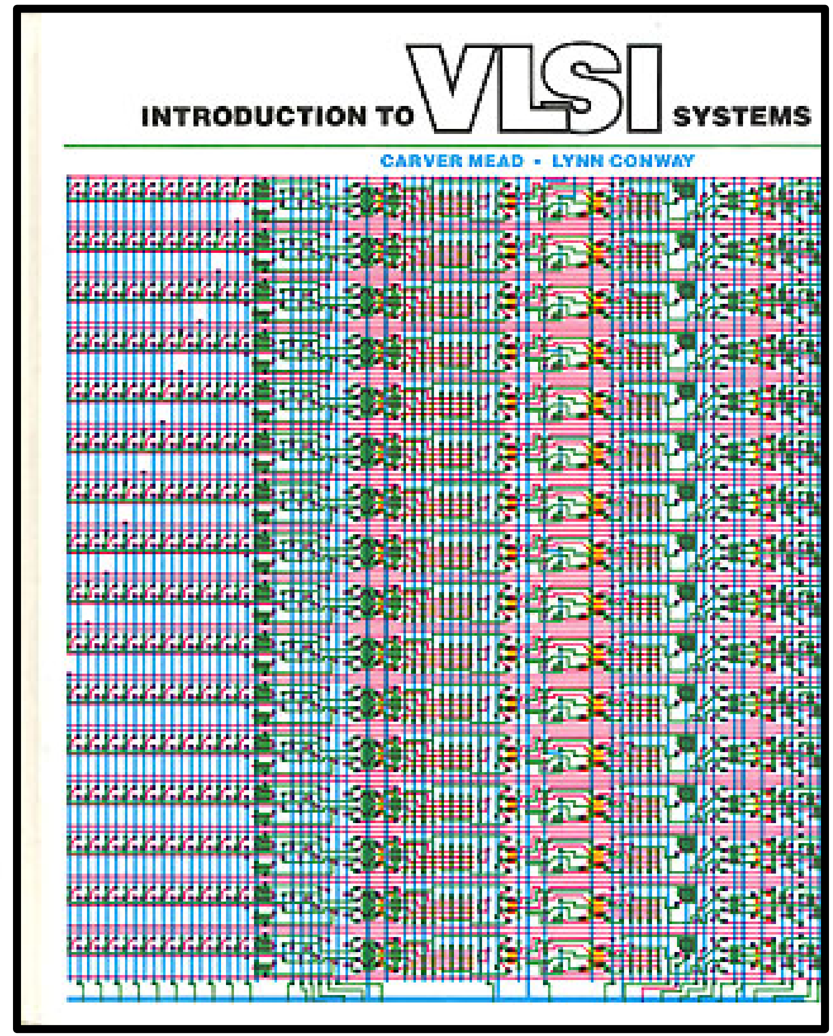


Fig. 8.31 Three levels of a memory hierarchy with alpha = 4.



8.5.2.3 Access Time of the RAM

For a RAM of S words, the access time in units of τ is then $\alpha b_0(\log S/2\log\alpha)$.

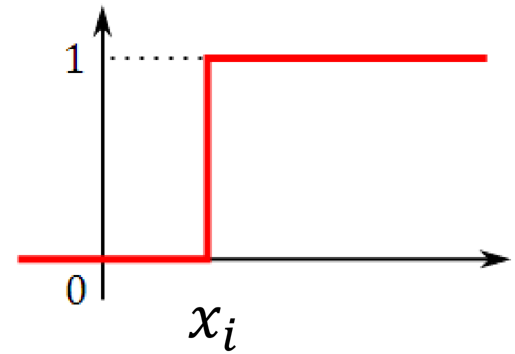
HMM Upper & Lower Bounds

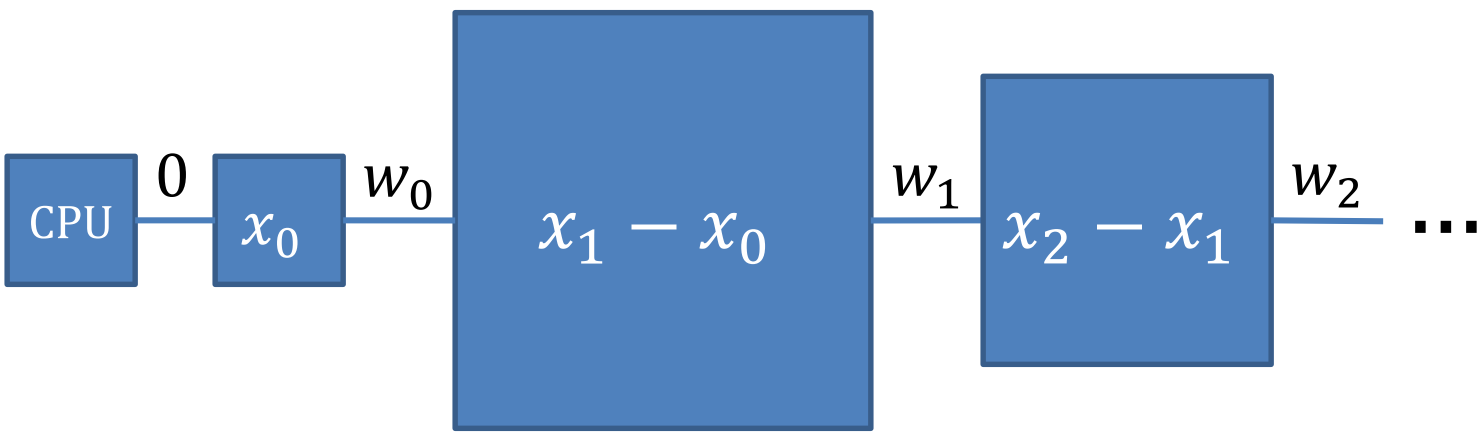
[Aggarwal, Alpern, Chandra, Snir — STOC 1987]

Problem	Time	Slowdown
Semiring matrix multiplication	$\Theta(N^3)$	$\Theta(1)$
Fast Fourier Transform	$\Theta(N \log N \log\log N)$	$\Theta(\log\log N)$
Sorting	$\Theta(N \log N \log\log N)$	$\Theta(\log\log N)$
Scanning input (sum, max, DFS, planarity, etc.)	$\Theta(N \log N)$	$\Theta(\log N)$
Binary search	$\Theta(\log^2 N)$	$\Theta(\log N)$

HMM$_{f(x)}$ [Aggarwal, Alpern, Chandra, Snir — STOC 1987]

- Say accessing memory location x costs $f(x)$
- Assume $f(2x) \le c\, f(x)$ for a constant $c > 0$ ("**polynomially bounded**")
- Write $f(x) = \sum_i w_i \cdot [x \ge x_i?]$ (weighted sum of threshold functions)

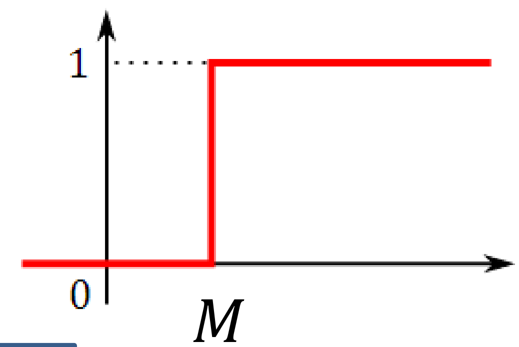


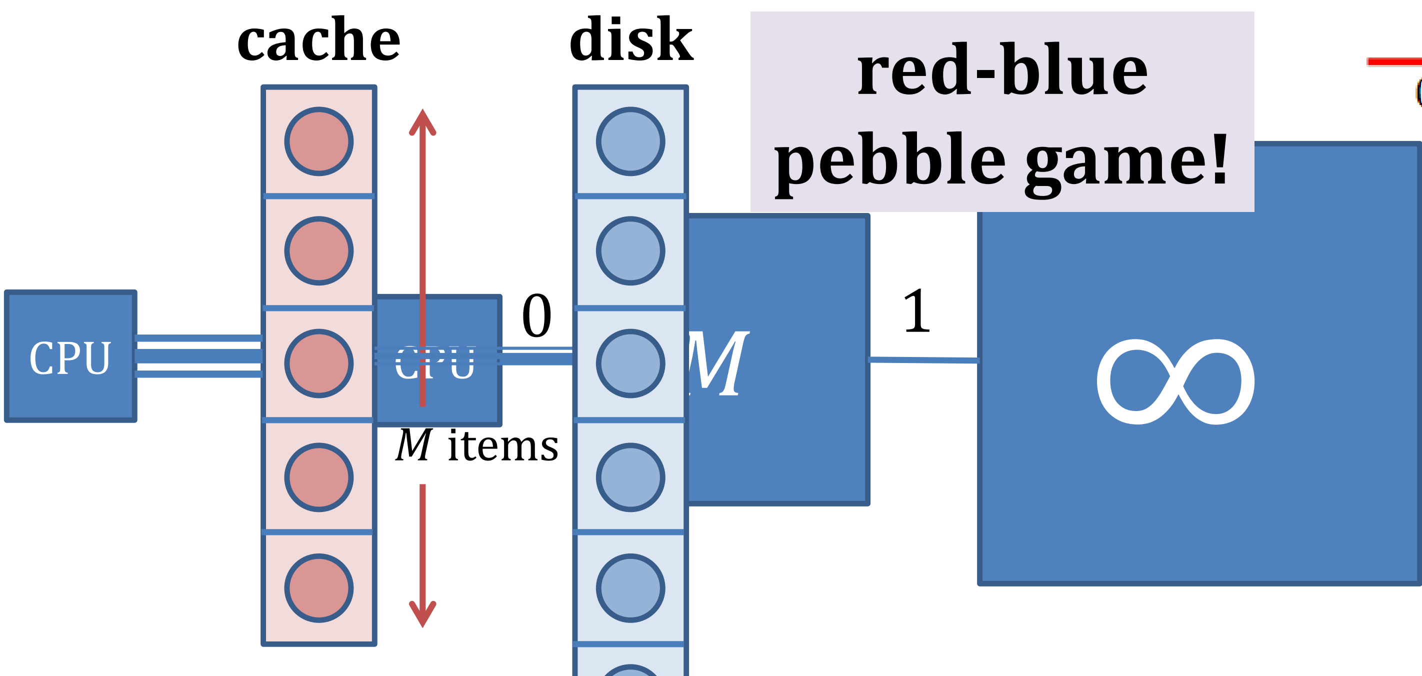


Uniform Optimality

[Aggarwal, Alpern, Chandra, Snir — STOC 1987]

- Consider *one* term $f_M(x) = [x \geq M?]$
- Algorithm is **uniformly optimal** if optimal on $\mathrm{HMM}_{f_M(x)}$ for *all* M
- Implies optimality for all $f(x)$





$\text{HMM}_{f_M(x)}$ Upper & Lower Bounds

[Aggarwal, Alpern, Chandra, Snir — STOC 1987]

Problem	Time	Speedup
Semiring matrix multiplication	$\Theta\left(\frac{N^3}{\sqrt{M}}\right)$	$\Theta(\sqrt{M})$
Fast Fourier Transform	$\Theta(N \log_M N)$	$\Theta(\log M)$
Sorting	$\Theta(N \log_M N)$	
Scanning input (sum, max, DFS, planarity, etc.)	$\Theta(N - M)$	$1 + 1/M$
Binary search	$\Theta(\log N - \log M)$	$1 + 1/\log M$

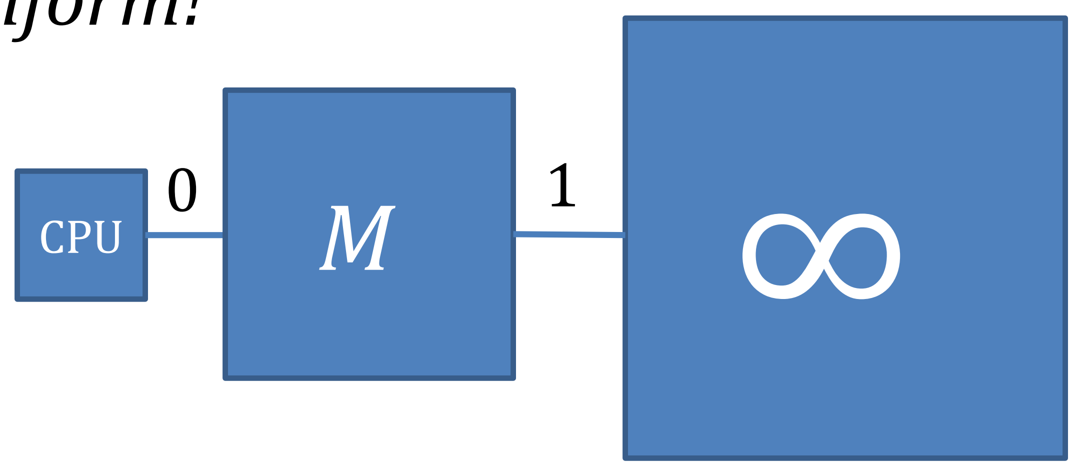
upper bounds known by Hong & Kung 1981

other bounds follow from Aggarwal & Vitter 1987

Implicit HMM Memory Management

[Aggarwal, Alpern, Chandra, Snir — STOC 1987]

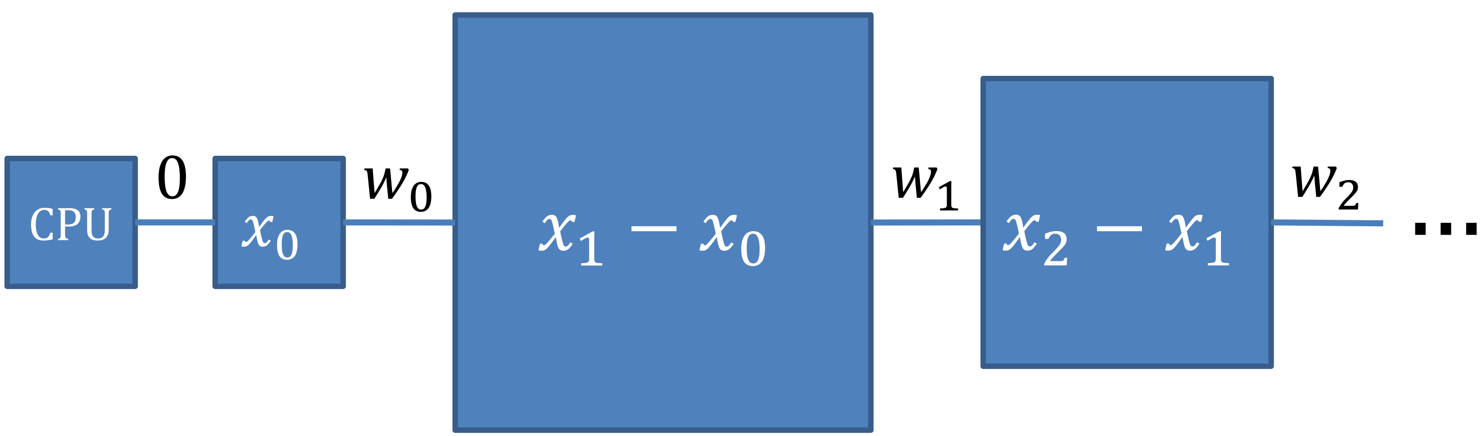
- Instead of algorithm explicitly moving data, use any **conservative replacement strategy** (e.g., FIFO or LRU) to evict from cache
  [Sleator & Tarjan — C. ACM 1985]
- $T_{\text{LRU}}(N, M) \leq 2 \cdot T_{\text{OPT}}(N, M/2)$
  $= O\big(T_{\text{OPT}}(N, M)\big)$ assuming $f(2x) \leq c\, f(x)$
- *Not uniform!*



Implicit HMM Memory Management

[Aggarwal, Alpern, Chandra, Snir — STOC 1987]

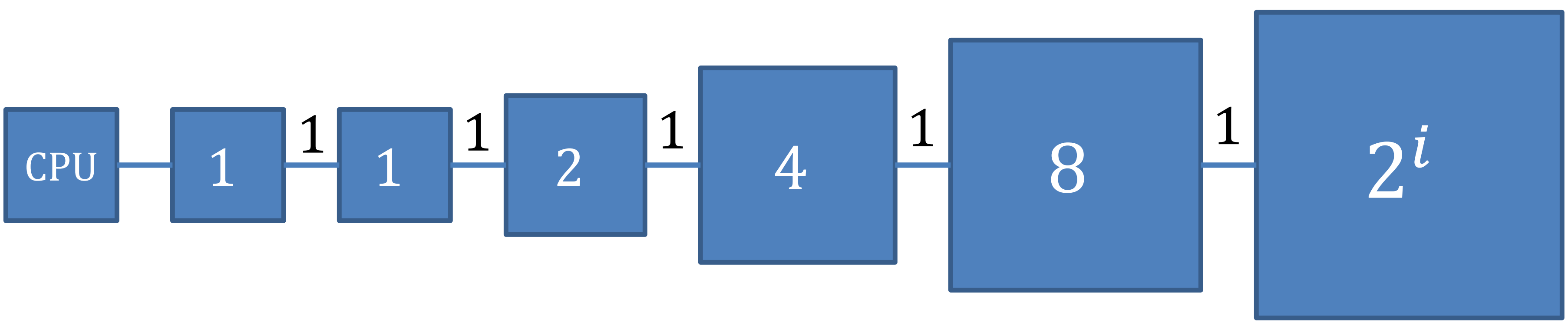
- For general f, split memory into **chunks** at x where $f(x)$ doubles (up to rounding)



Implicit HMM Memory Management

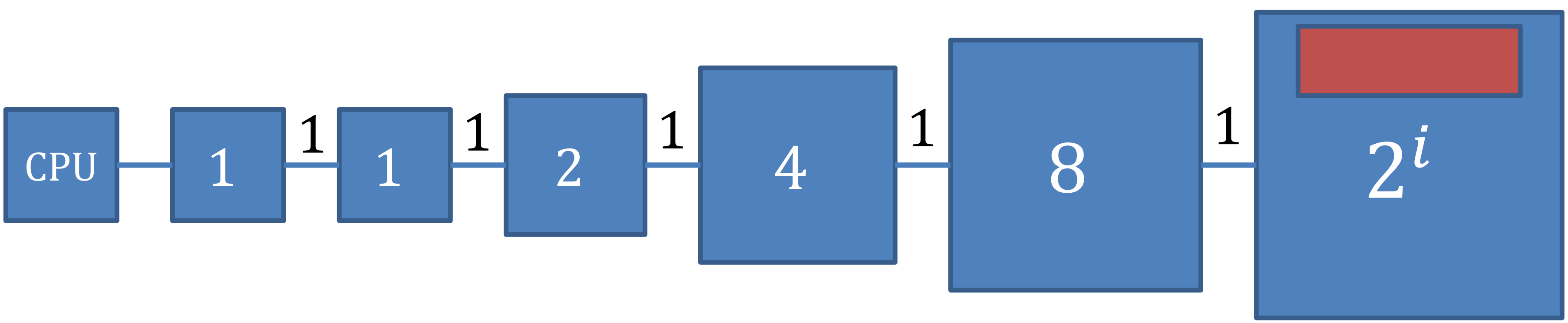
[Aggarwal, Alpern, Chandra, Snir — STOC 1987]

- For general f, split memory into **chunks** at x where $f(x)$ doubles (up to rounding)
- LRU eviction from first chunk into second; LRU eviction from second chunk into third; etc.
- $T_{LRU}(N) = O\big(T_{OPT}(N) + N \cdot f(N)\big)$
  - Like MTF



HMM with Block Transfer (BT)
[Aggarwal, Chandra, Snir — FOCS 1987]

- Accessing memory location x costs $f(x)$
- Copying memory interval from $x - \delta \ldots x$ to $y - \delta \ldots y$ costs $f(\max\{x, y\}) + \delta$
  - Models memory pipelining ~ block transfer
  - Ignores block alignment, explicit levels, etc.



BT Results

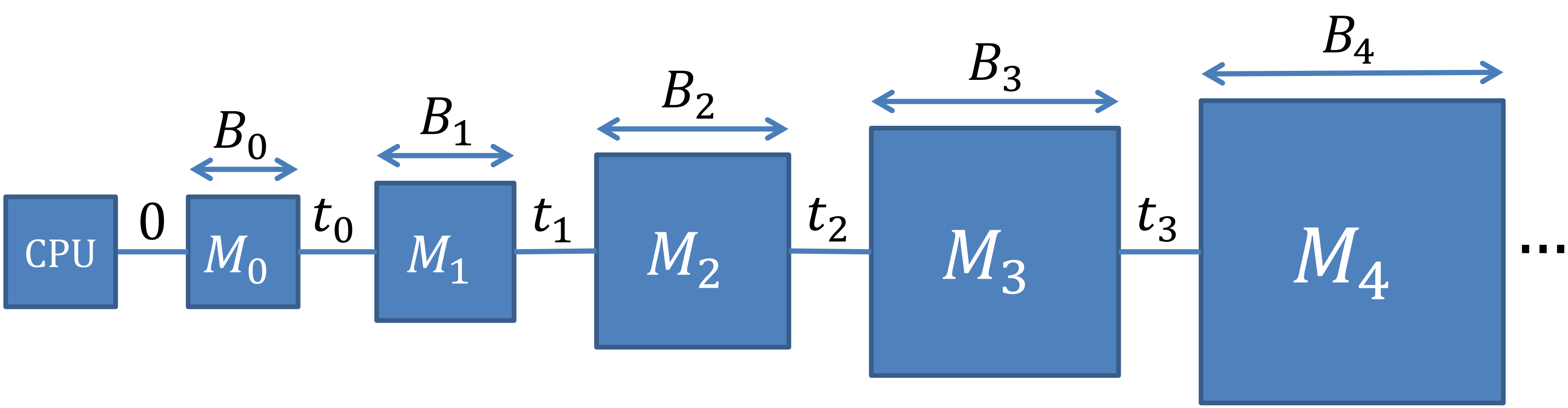
[Aggarwal, Chandra, Snir — FOCS 1987]

Problem	$f(x) = \log x$	$f(x) = x^\alpha$, $0 < \alpha < 1$	$f(x) = x$	$f(x) = x^\alpha$, $\alpha > 1$
Dot product, merging lists	$\Theta(N \log^* N)$	$\Theta(N \log\log N)$	$\Theta(N \log N)$	$\Theta(N^\alpha)$
Matrix mult.	$\Theta(N^3)$	$\Theta(N^3)$	$\Theta(N^3)$	$\Theta(N^\alpha)$ if $\alpha > 1.5$
Fast Fourier Transform	$\Theta(N \log N)$	$\Theta(N \log N)$	$\Theta(N \log^2 N)$	$\Theta(N^\alpha)$
Sorting	$\Theta(N \log N)$	$\Theta(N \log N)$	$\Theta(N \log^2 N)$	$\Theta(N^\alpha)$
Binary search	$\Theta\left(\frac{\log^2 N}{\log\log N}\right)$	$\Theta(N^\alpha)$	$\Theta(N)$	$\Theta(N^\alpha)$

Memory Hierarchy Model (MH)

[Alpern, Carter, Feig, Selker — FOCS 1990]

- Multilevel version of external-memory model
- $M_i \leftrightarrow M_{i+1}$ transfers happen in blocks of size B_i (subblocks of M_{i+1}), and take t_i time
- All levels can be actively transferring at once

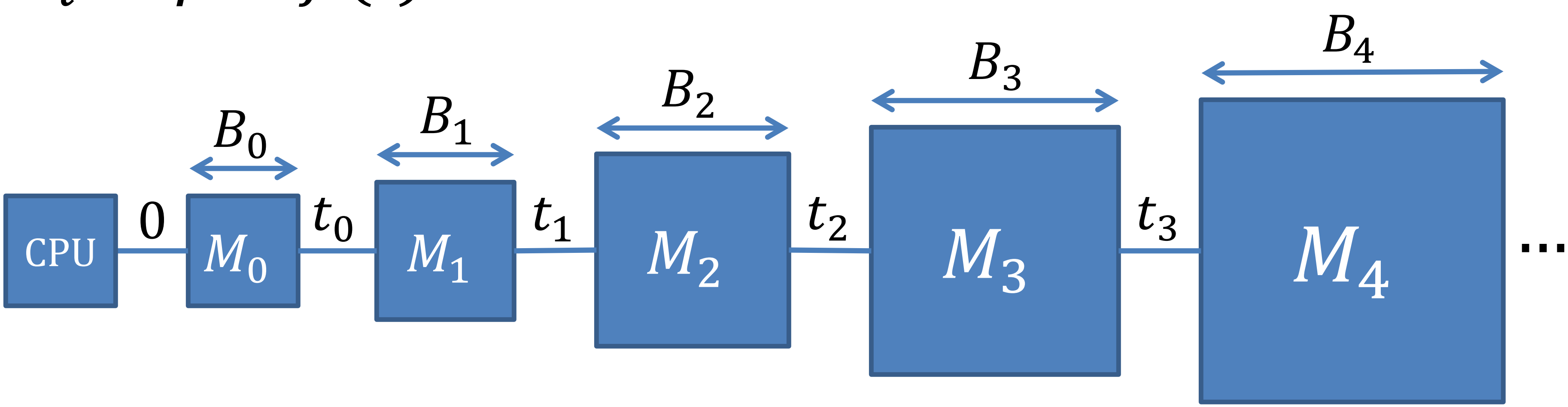


Uniform Memory Hierarchy (UMH)

[Alpern, Carter, Feig, Selker — FOCS 1990]

- Fix **aspect ratio** $\alpha = \frac{M/B}{B}$, **block growth** $\beta = \frac{B_{i+1}}{B_i}$
- $B_i = \beta^i$
- $\frac{M_i}{B_i} = \alpha \cdot \beta^i$
- $t_i = \beta^i \cdot f(i)$

2 parameters
1 function



UMH Results

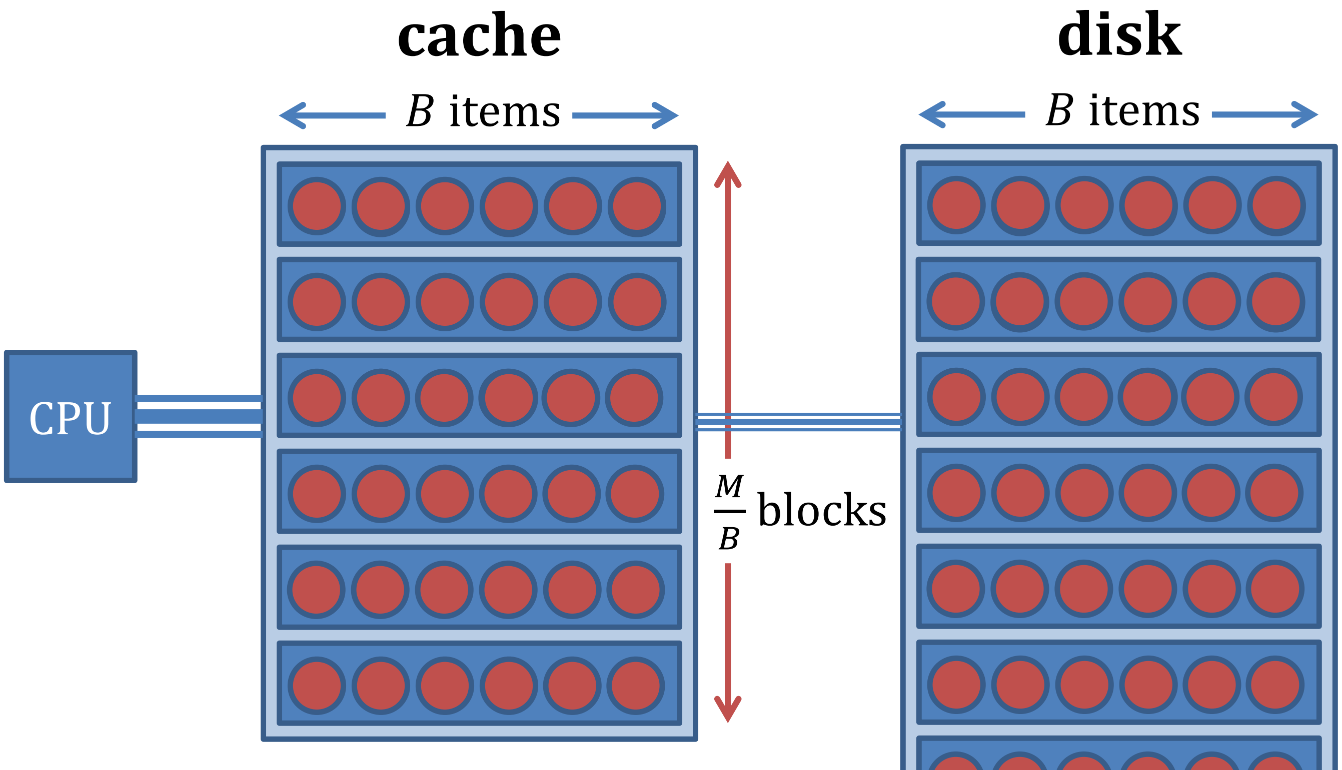
[Alpern, Carter, Feig, Selker — FOCS 1990]

Problem	Upper Bound	Lower Bound
Matrix transpose $f(i) = 1$	$O\left(\left(1+\frac{1}{\beta^2}\right)N^2\right)$	$\Omega\left(\left(1+\frac{1}{\alpha\beta^4}\right)N^2\right)$
Matrix mult. $f(i) = O(\beta^i)$	$O\left(\left(1+\frac{1}{\beta^3}\right)N^3\right)$	$\Omega\left(\left(1+\frac{1}{\beta^3}\right)N^3\right)$
FFT $f(i) \leq i$	$O(1)$	$\Omega(1)$

General approach: Divide & conquer

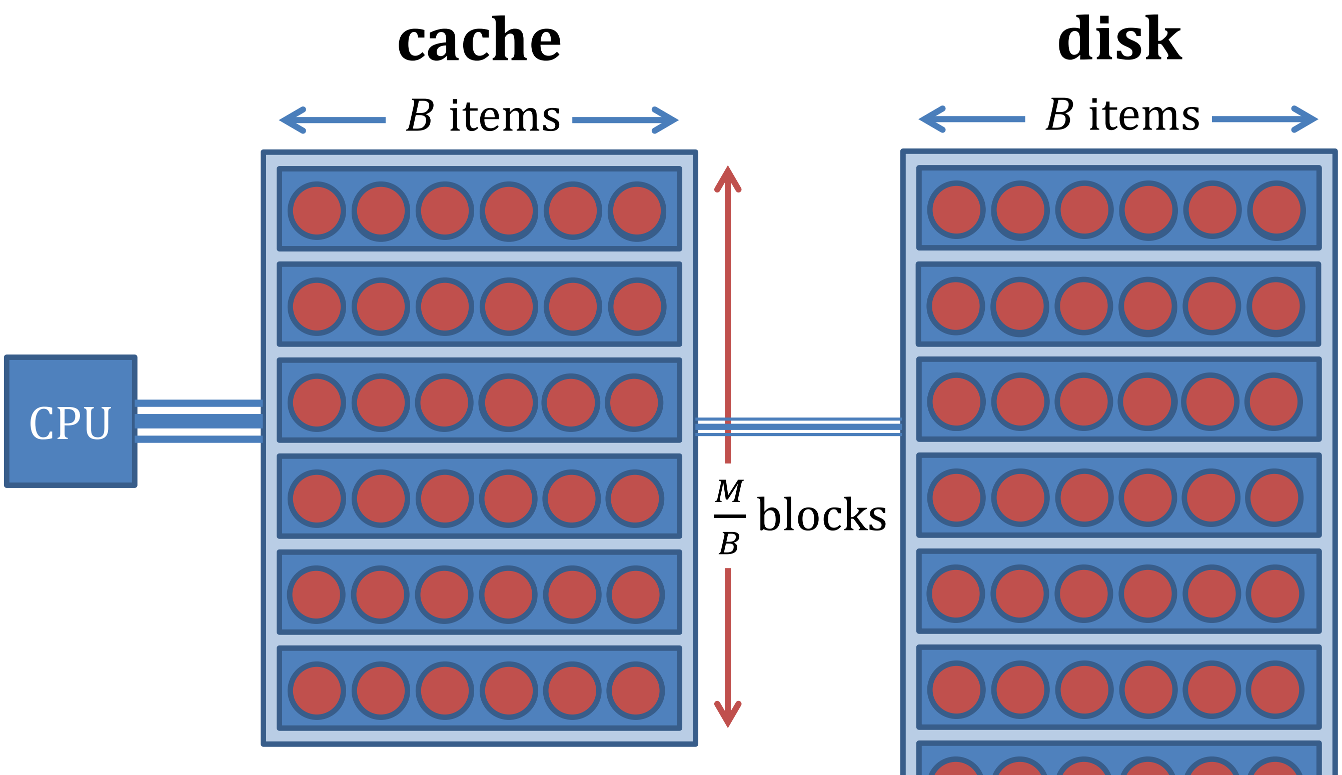
Cache-Oblivious Model [Frigo, Leiserson, Prokop, Ramachandran — FOCS 1999]

- Analyze RAM algorithm (not knowing B or M) on external-memory model
  - Must work well for *all* B and M



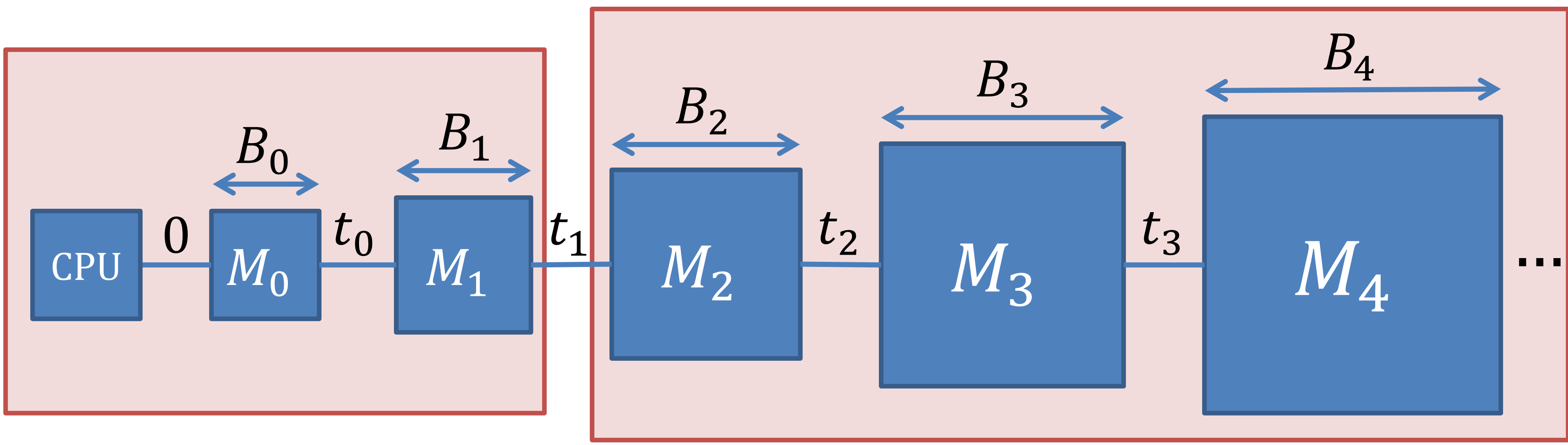
Cache-Oblivious Model [Frigo, Leiserson, Prokop, Ramachandran — FOCS 1999]

- Automatic block transfers via LRU or FIFO
- Lose factor of 2 in M and number of transfers
  - Assume $T(B, 2M) \leq c\, T(B, M)$



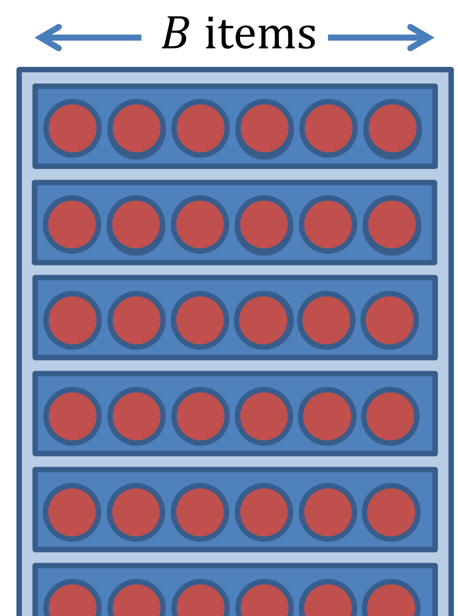
Cache-Oblivious Model [Frigo, Leiserson, Prokop, Ramachandran — FOCS 1999]

- Clean model
- Adapts to changing B (e.g., disk tracks) and changing M (e.g., competing processes)
- Adapts to multilevel memory hierarchy (MH)
  - Assuming inclusion

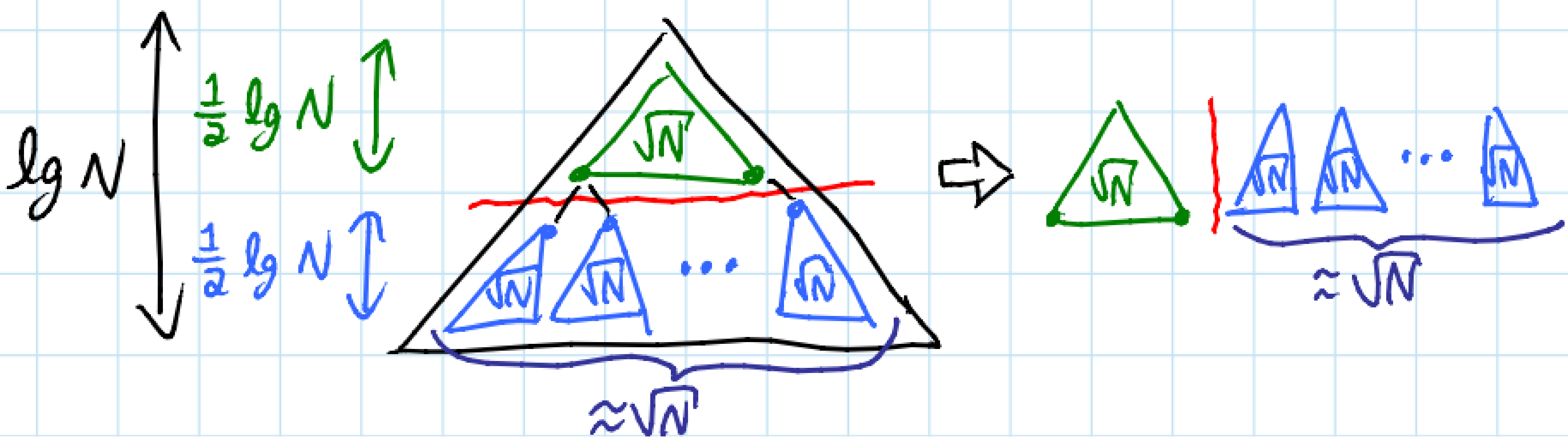


Scanning [Frigo, Leiserson, Prokop, Ramachandran — FOCS 1999]

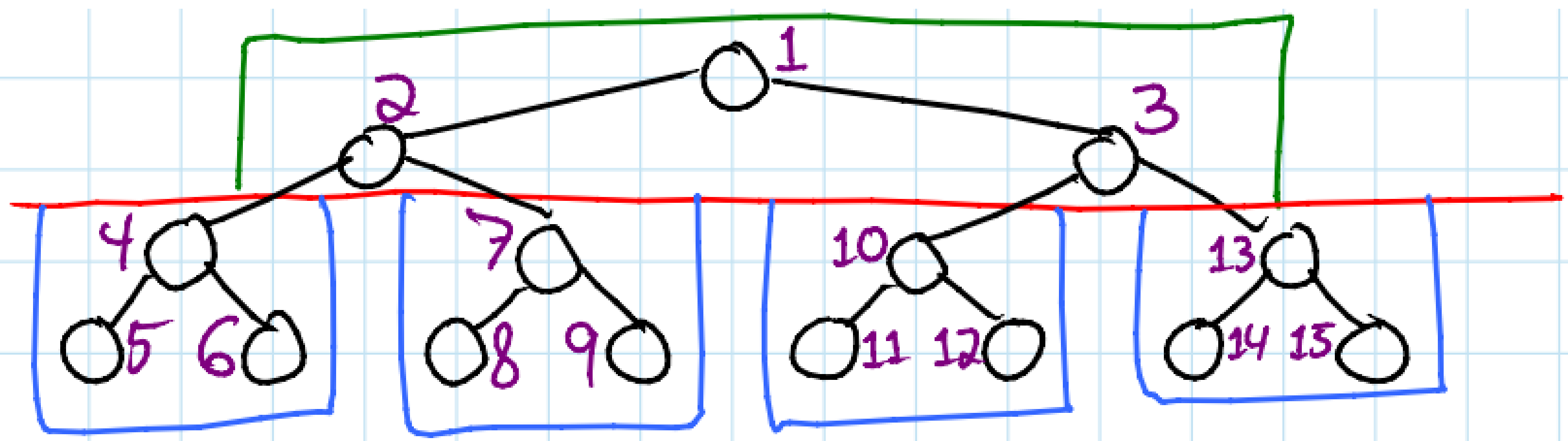
- Visiting N elements in order costs $O\left(1+\frac{N}{B}\right)$ memory transfers
- More generally, can run $O(1)$ parallel scans
  - Assume $M \geq c\, B$ for appropriate constant $c > 0$
- E.g., merge two lists in $O\left(\frac{N}{B}\right)$



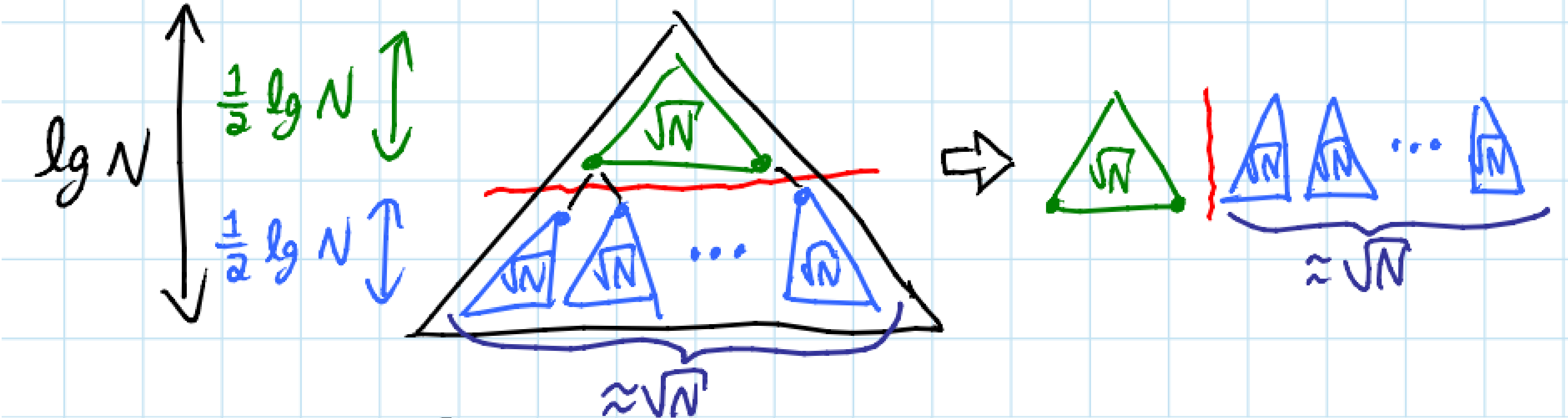
Searching [Prokop — Meng 1999]

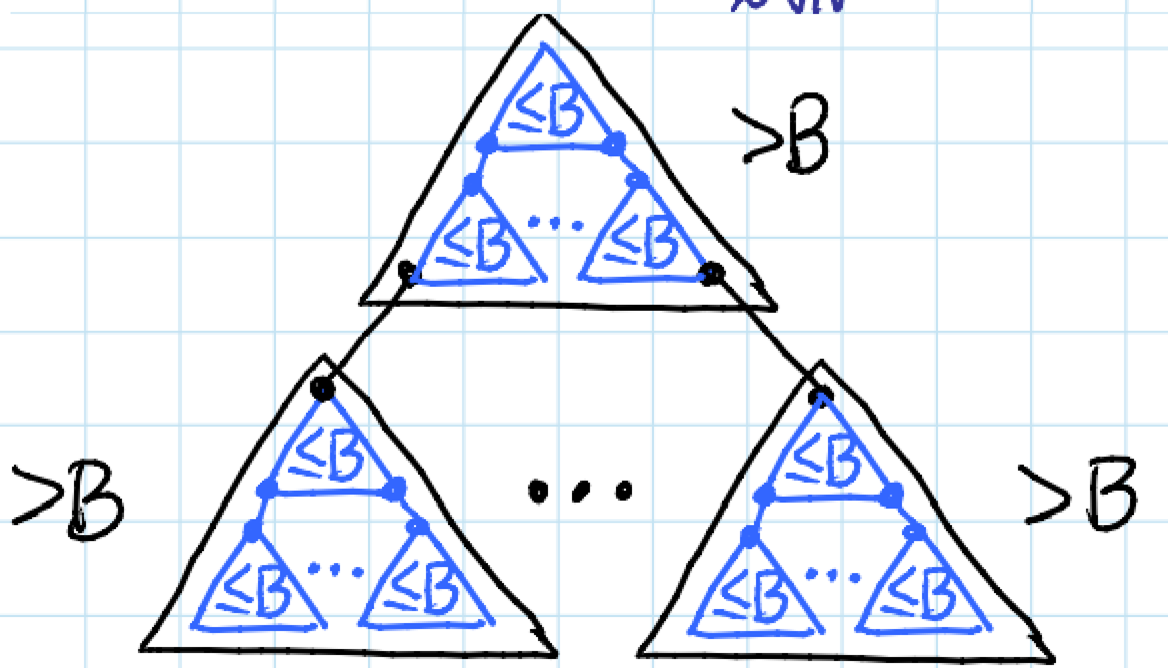


"van Emde Boas layout"



Searching [Prokop — Meng 1999]

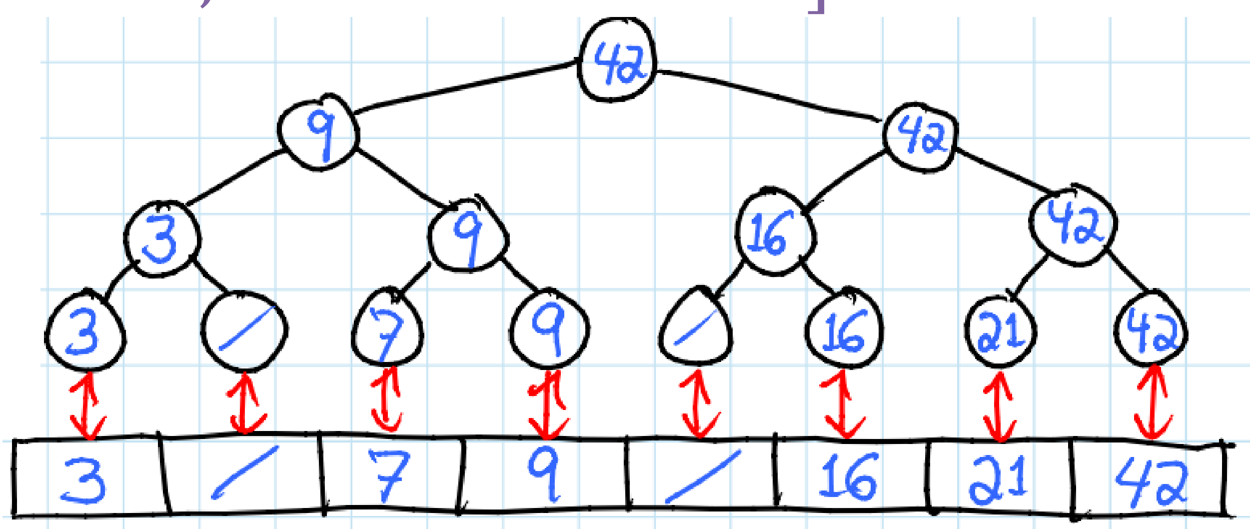




- height(△) $\geq \frac{1}{2} \lg B$
- ≤ 2 memory transfers per △
- $\leq 4 \log_B N$ total

Cache-Oblivious Searching

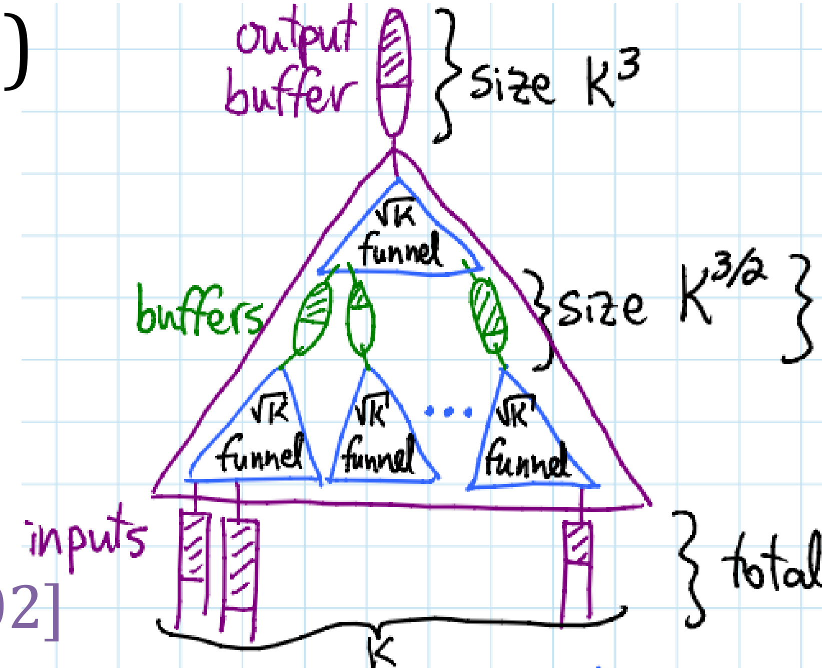
- $\left(\lg e + o(1)\right) \log_B N$ is optimal
  [Bender, Brodal, Fagerberg, Ge, He, Hu, Iacono, López-Ortiz — FOCS 2003]
- Dynamic B-tree in $O(\log_B N)$ per operation
  [Bender, Demaine, Farach-Colton — FOCS 2000]
  [Bender, Duan, Iacono, Wu — SODA 2002]
  [Brodal, Fagerberg, Jacob — SODA 2002]



Cache-Oblivious Sorting

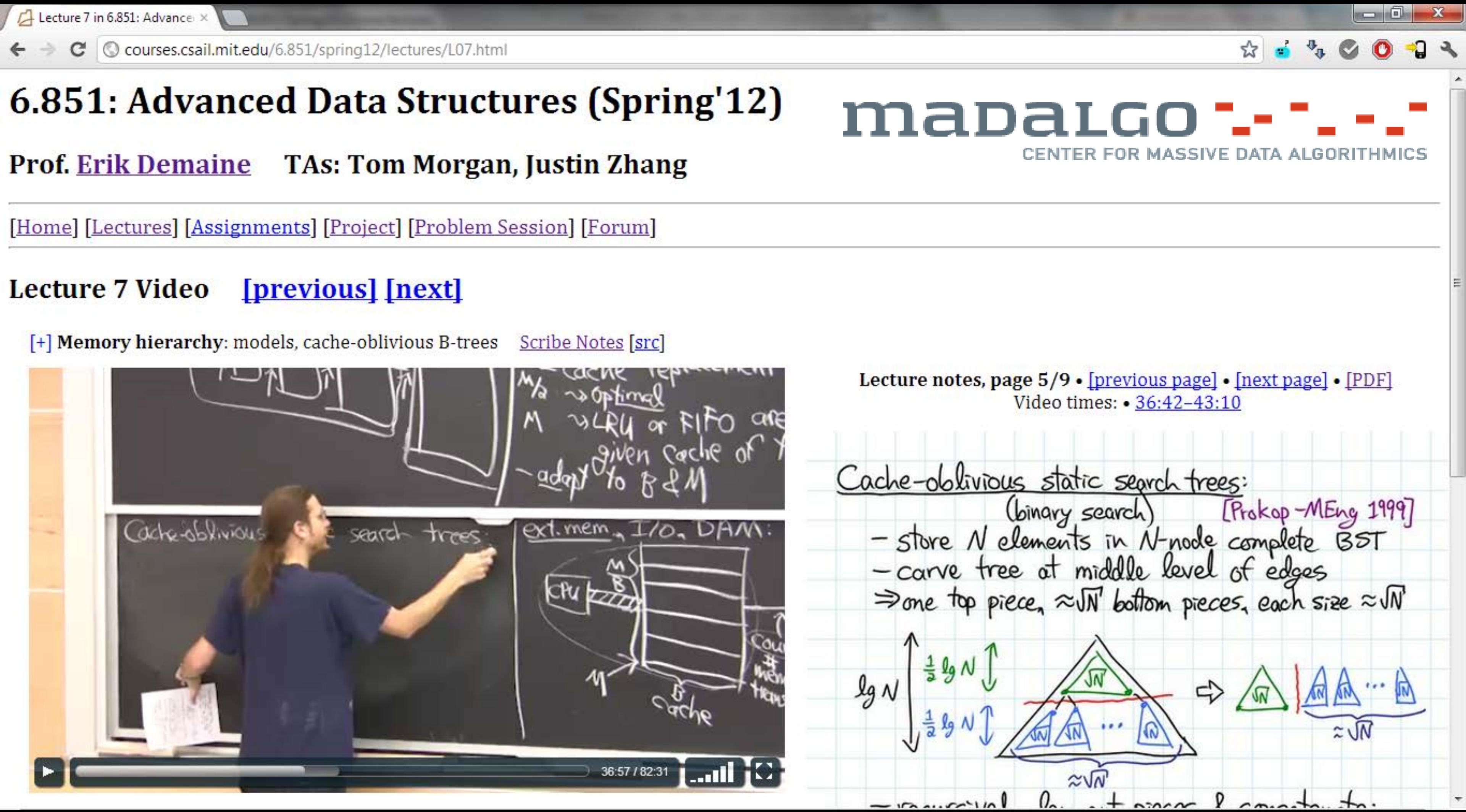
- $O\left(\frac{N}{B}\log_{M/B}\frac{N}{B}\right)$ possible, assuming $M \geq \Omega(B^{1+\varepsilon})$ (**tall cache**)
  - Funnel sort: mergesort analog
  - Distribution sort

  [Frigo, Leiserson, Prokop, Ramachandran — FOCS 1999; Brodal & Fagerberg — ICALP 2002]



- Impossible without tall-cache assumption [Brodal & Fagerberg — STOC 2003]

http://courses.csail.mit.edu/6.851/



Models, Models, Models

Model	Year	Blocking	Caching	Levels	Simple
Idealized 2-level	1972	✓	✗	2	✓
Red-blue pebble	1981	✗	✓	2	✓–
External memory	1987	✓	✓	2	✓
HMM	1987	✗	✓	∞	✓
BT	1987	~	✓	∞	✓–
(U)MH	1990	✓	✓	∞	✗
Cache oblivious	1999	✓	✓	2–∞	✓+