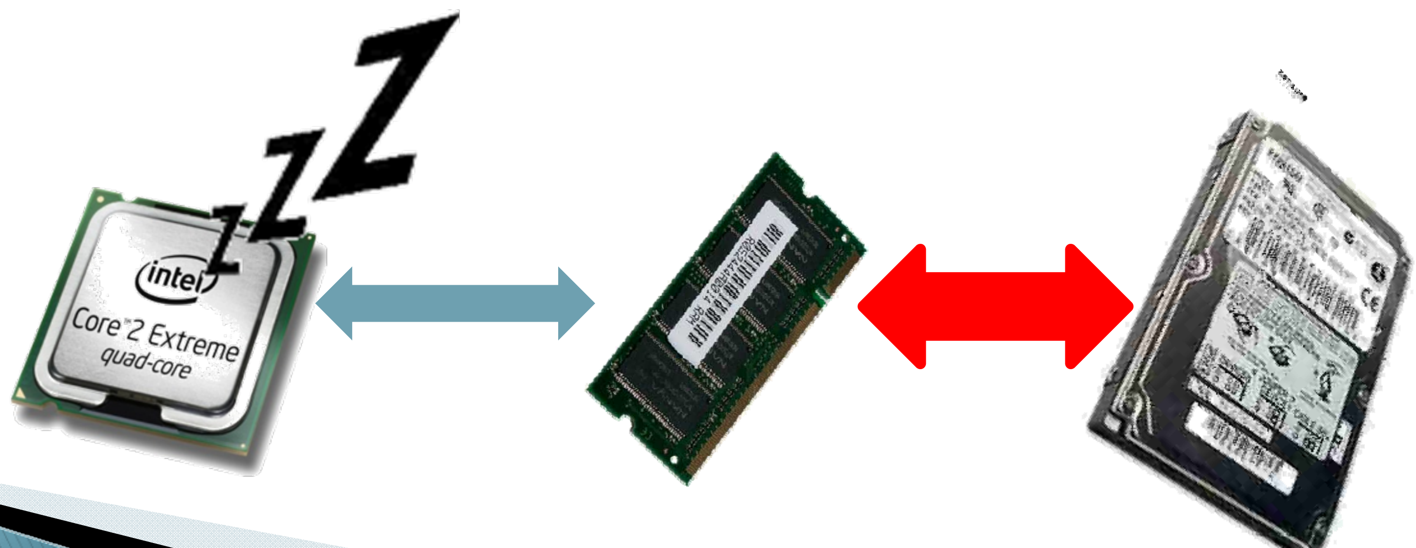
Cache-Oblivious Algorithms

Marc Tchiboukdjian
MOAIS Project
Grenoble Informatics Laboratory

Motivation

- Data sets are often too massive to fit completely inside the computer's internal memory

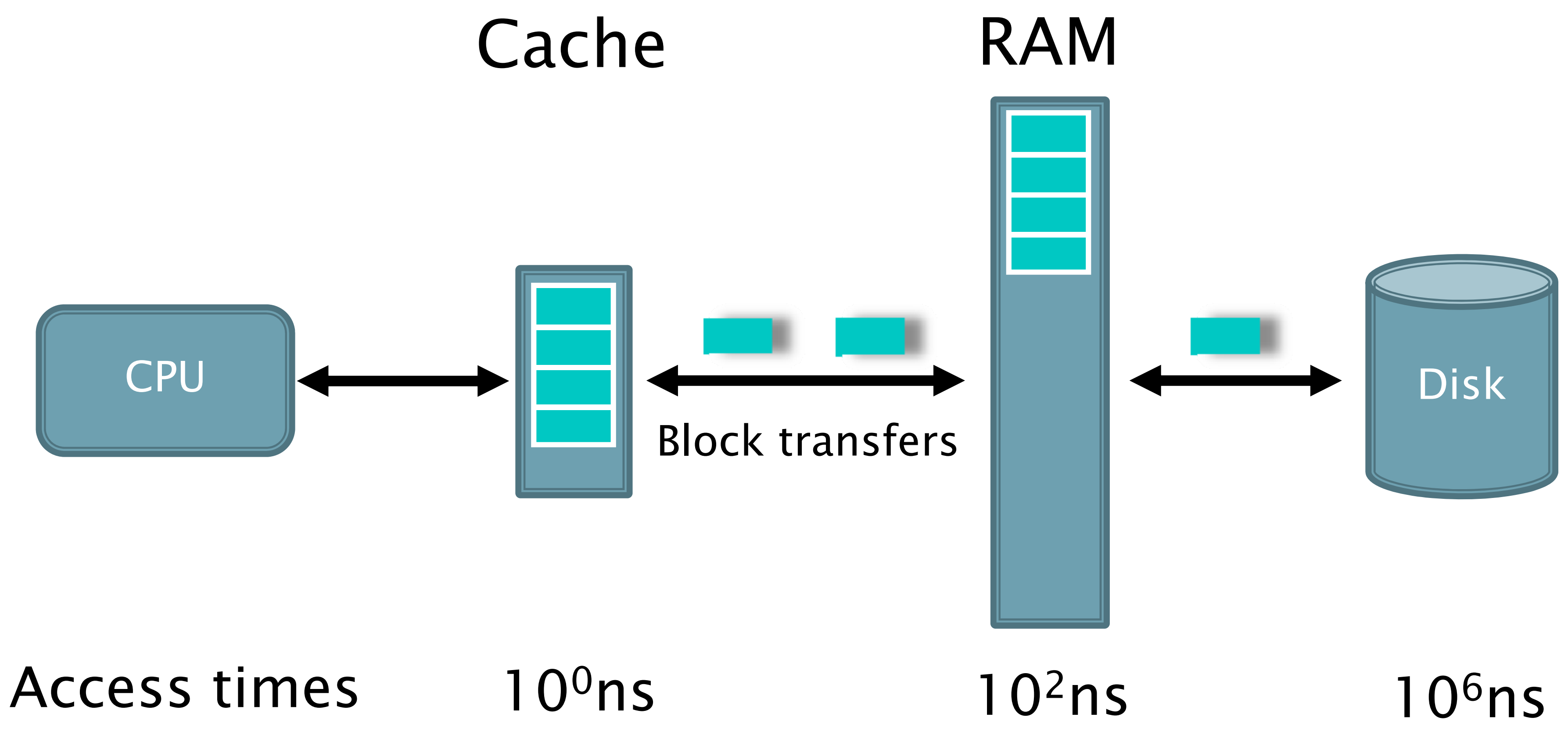
- I/Os between internal and external memory is the bottleneck



Outline

- Memory Hierarchy
- Disk Access Model
- Cache Oblivious Model

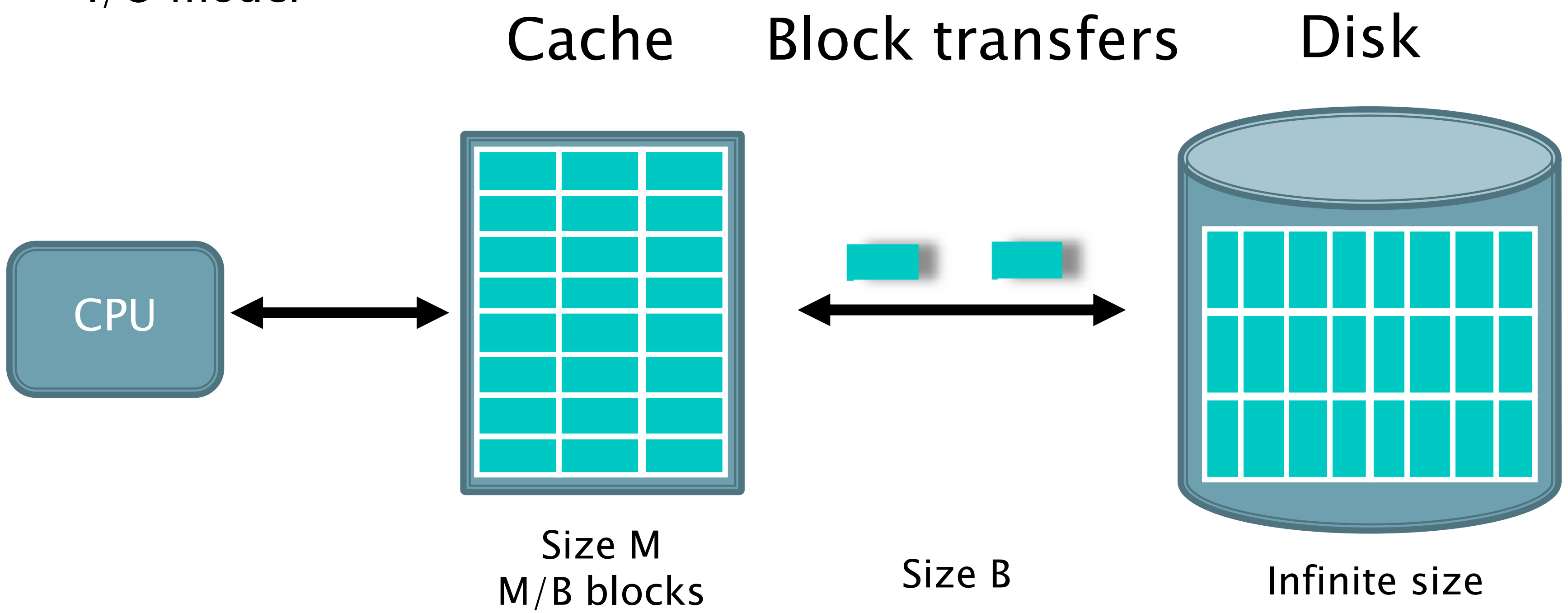
Memory Hierarchy



Disk Access Model (DAM)

or external memory
out-of-core
cache-aware
I/O model

[Aggarwal and Vitter 1988]

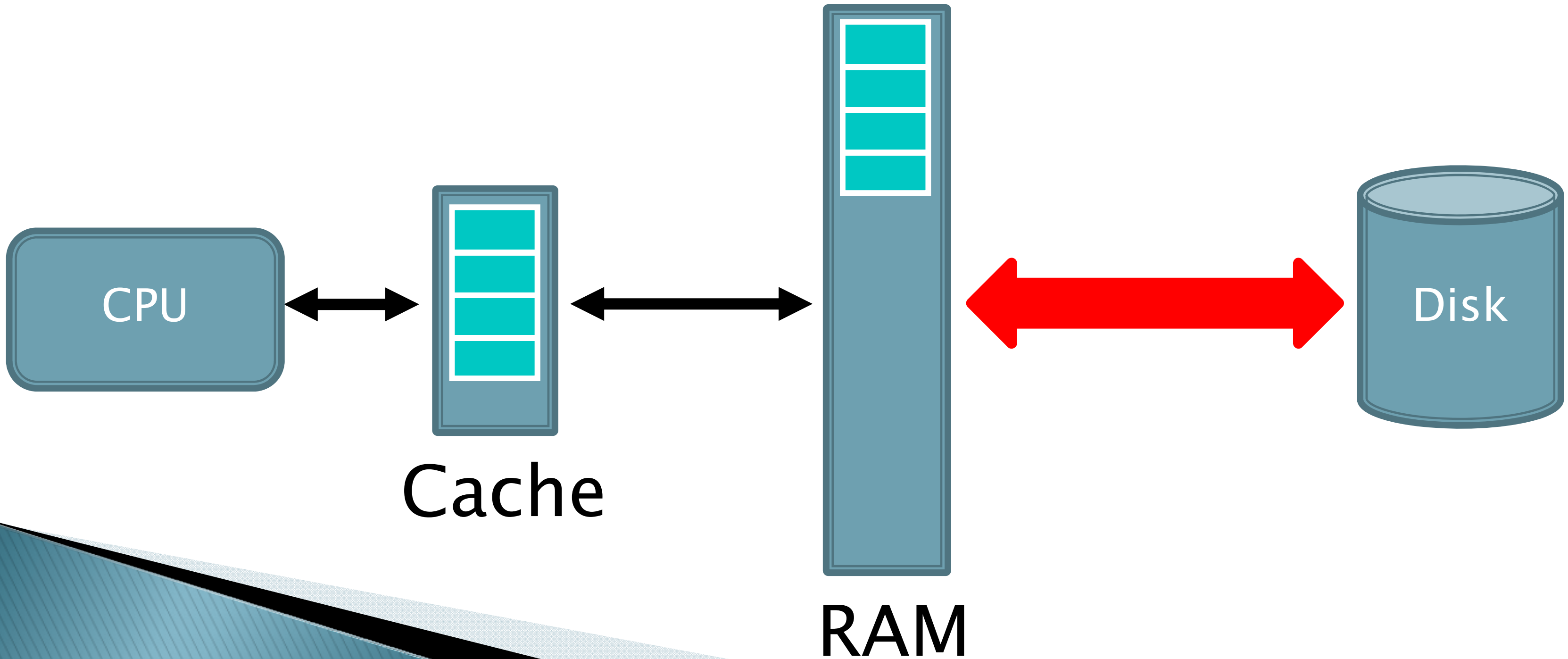


W: #operations CPU

Q: #block transfers

Advantages of the DAM model

- Simple: only two levels
- Good when the bottleneck is between two specific levels

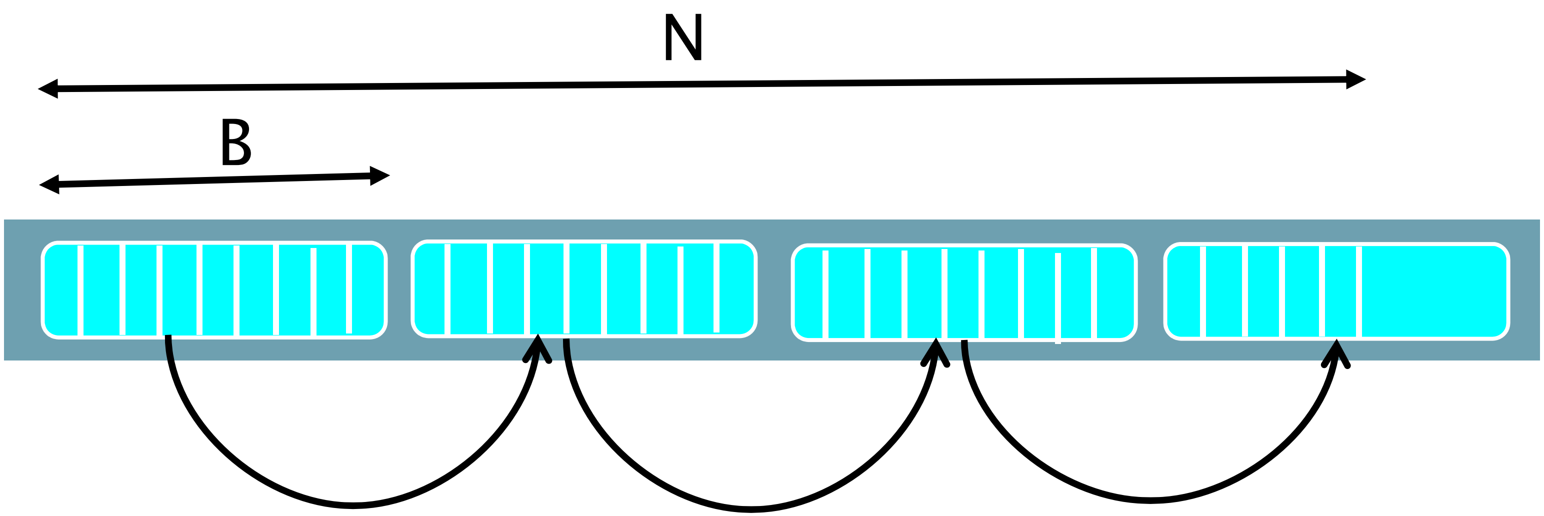


Principles of external-memory algorithm design

- *Internal efficiency:* work is comparable to the best internal memory algorithms
- *Spatial locality:* a block should contain as much useful data as possible
- *Temporal locality:* as much useful work as possible before the block is ejected

Scanning in the DAM model

Read an N-elements array: the naive algorithm is optimal



$$W(N) = N$$

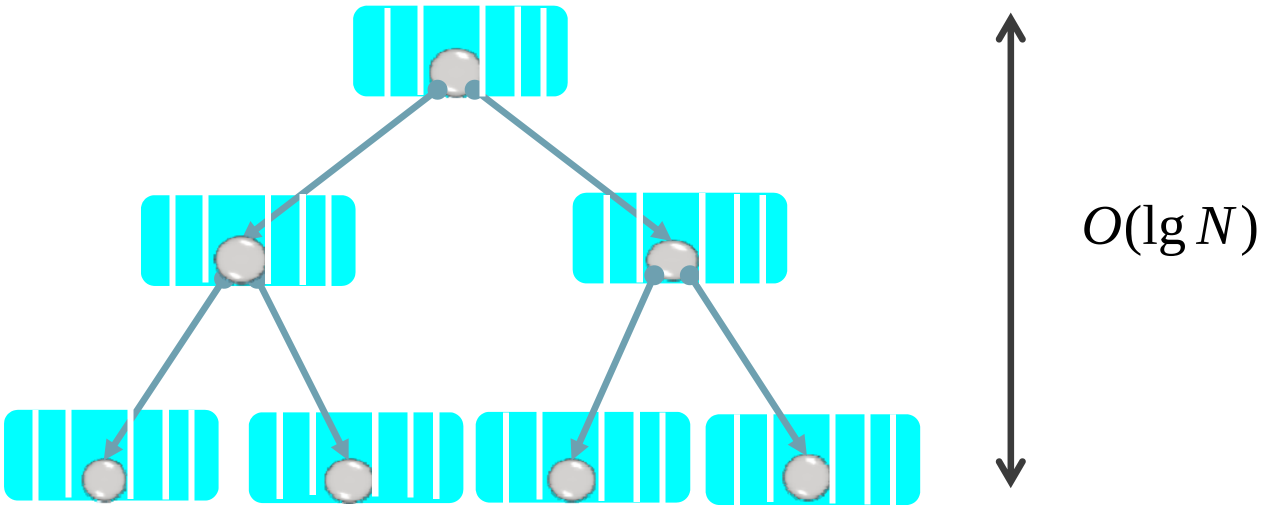
$$Q(N) = \lceil N / B \rceil$$

$$scan(N) = \lceil N / B \rceil$$

this bound is optimal

Searching in the DAM model

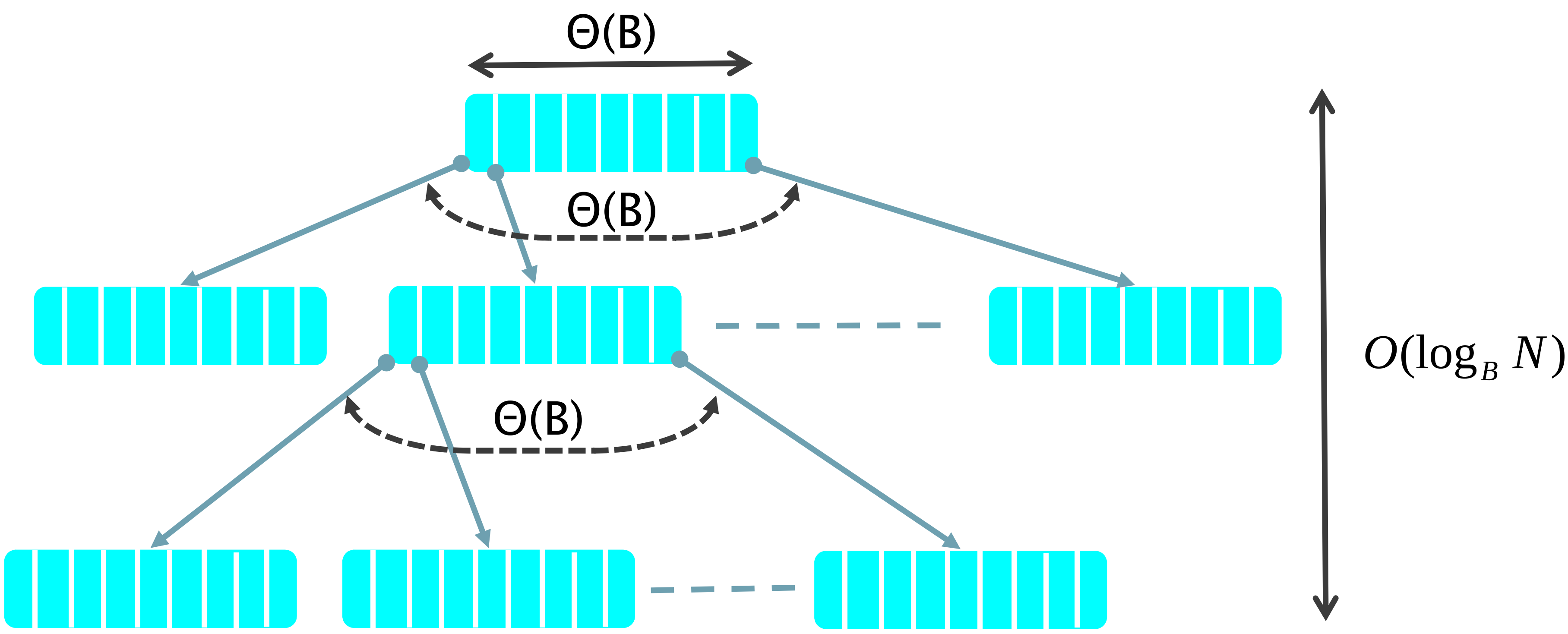
Searching a key in an N-nodes balanced binary tree: naive doesn't work



$$W(N) = 1.O(\lg N) = O(\mathbf{l}gN)$$
$$Q(N) = 1.O(\lg N) = O(\lg N)$$

Searching in the DAM model

Searching a key in an N-elements B-tree [Bayer and McCreight 1972]



$$W(N) = \lg B . O(\log_B N) = O(\lg N)$$

$$Q(N) = 1 . O(\log_B N) = O(\log_B N)$$

Multiplying in the DAM model

NxN matrices in row-major order: naive doesn't work

Using the naive N^3 algorithm:

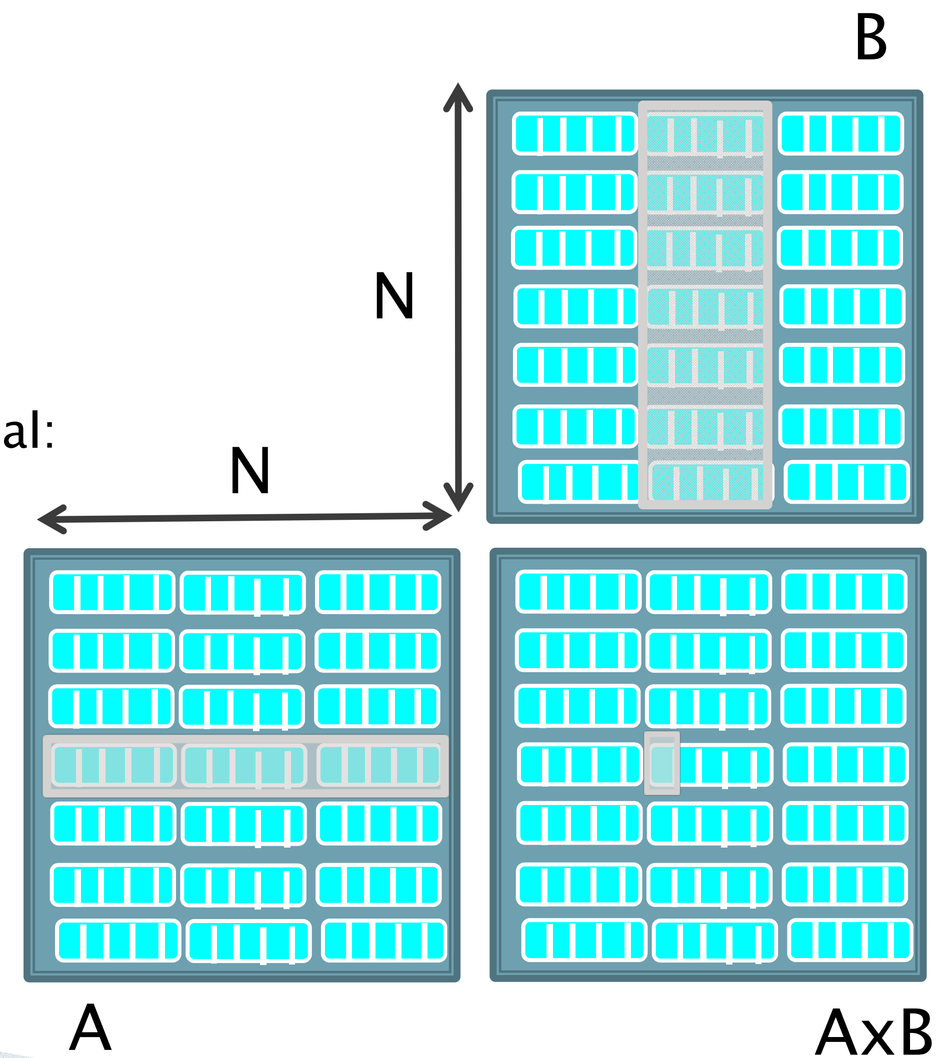
$$W(N) = O(N).N^2$$

$$W(N) = O(N^3)$$

Memory accesses in B are suboptimal:

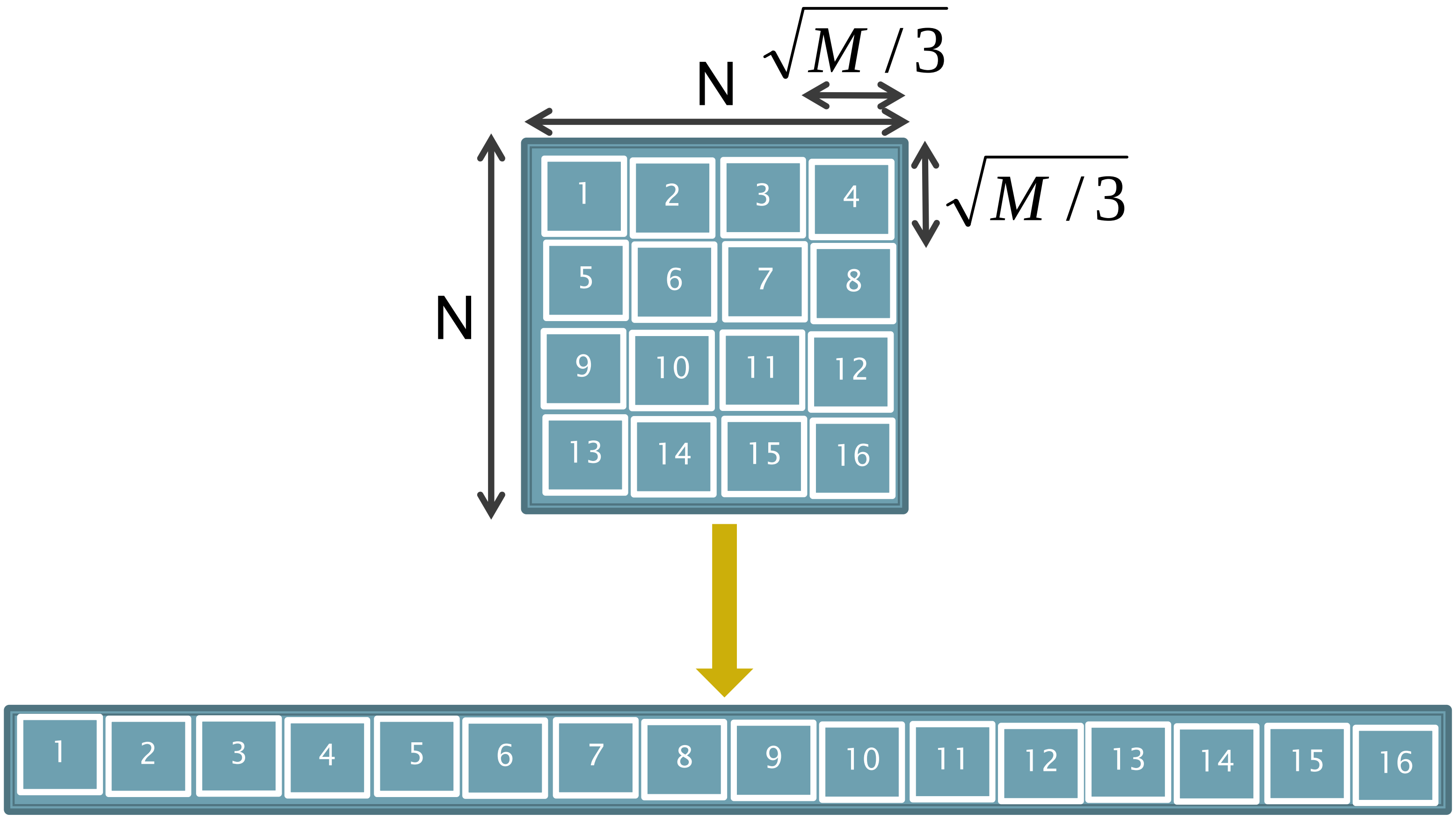
$$Q(N) = O\left(\frac{N}{B} + N\right).N^2$$

$$Q(N) = O(N^3)$$



Multiplying in the DAM model

NxN matrices in submatrices



Multiplying in the DAM model

NxN matrices in submatrices

- Cost for two sub-matrices

$$W(N) = O\left(\sqrt{M}^3\right) \quad Q(N) = O\left(\frac{M}{B}\right)$$

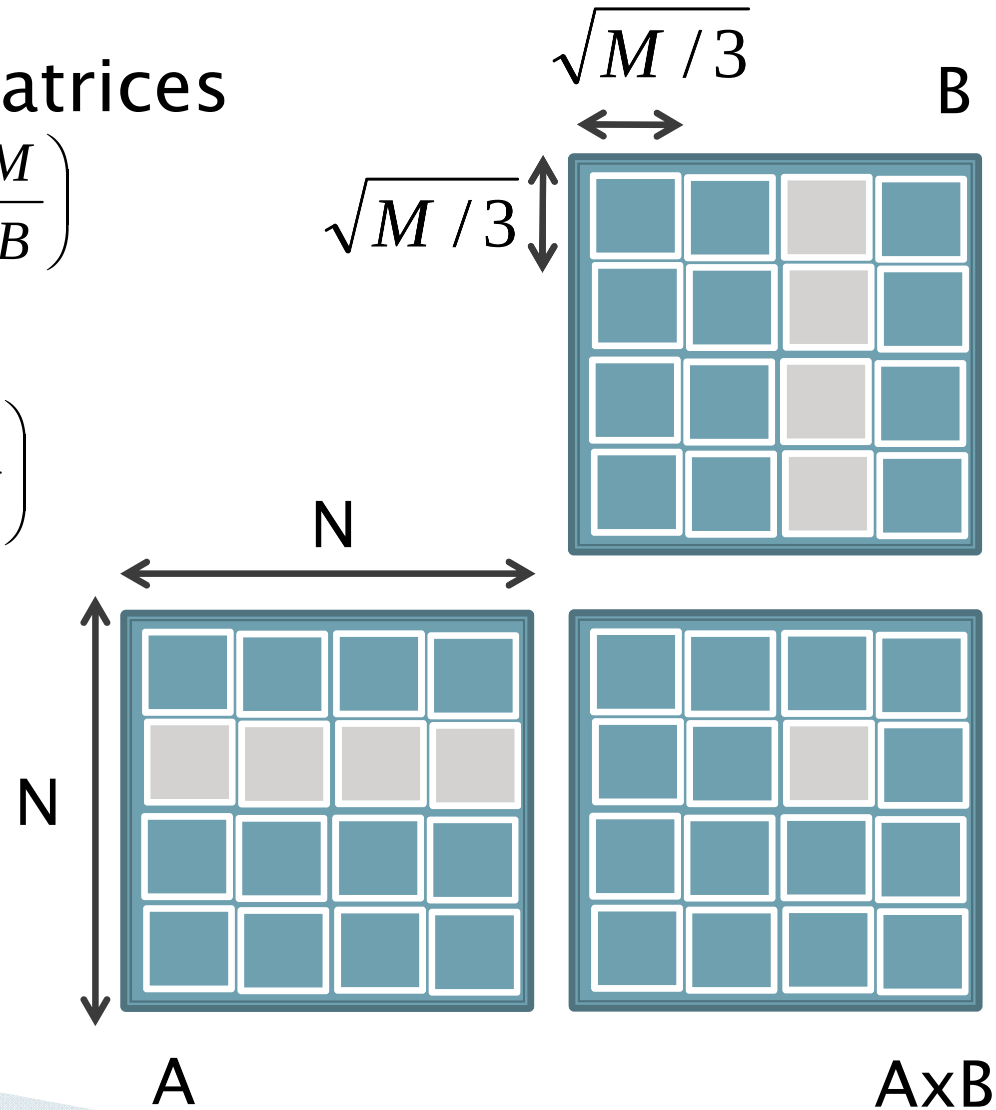
- Total cost

$$W(N) = O\left(\sqrt{M}^3\right) . O\left(\frac{N}{\sqrt{M}}\right) . O\left(\frac{N^2}{M}\right)$$

$$W(N) = O\left(N^3\right)$$

$$Q(N) = O\left(\frac{M}{B}\right) . O\left(\frac{N}{\sqrt{M}}\right) . O\left(\frac{N^2}{M}\right)$$

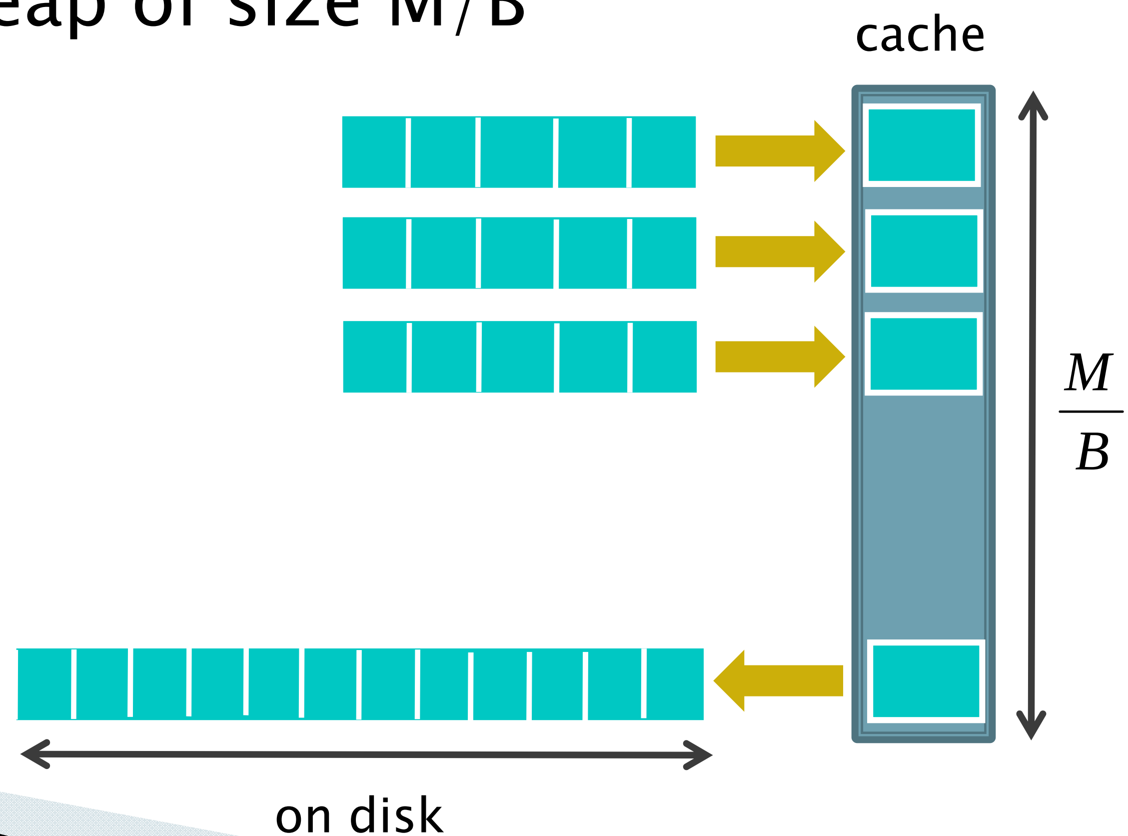
$$Q(N) = O\left(\frac{N^3}{B\sqrt{M}}\right)$$



Sorting in the DAM model

M/B-way merge sort of an N-elements array

- Cut into M/B sublists
- Recursively sort them
- Merge using a heap of size M/B



Sorting in the DAM model

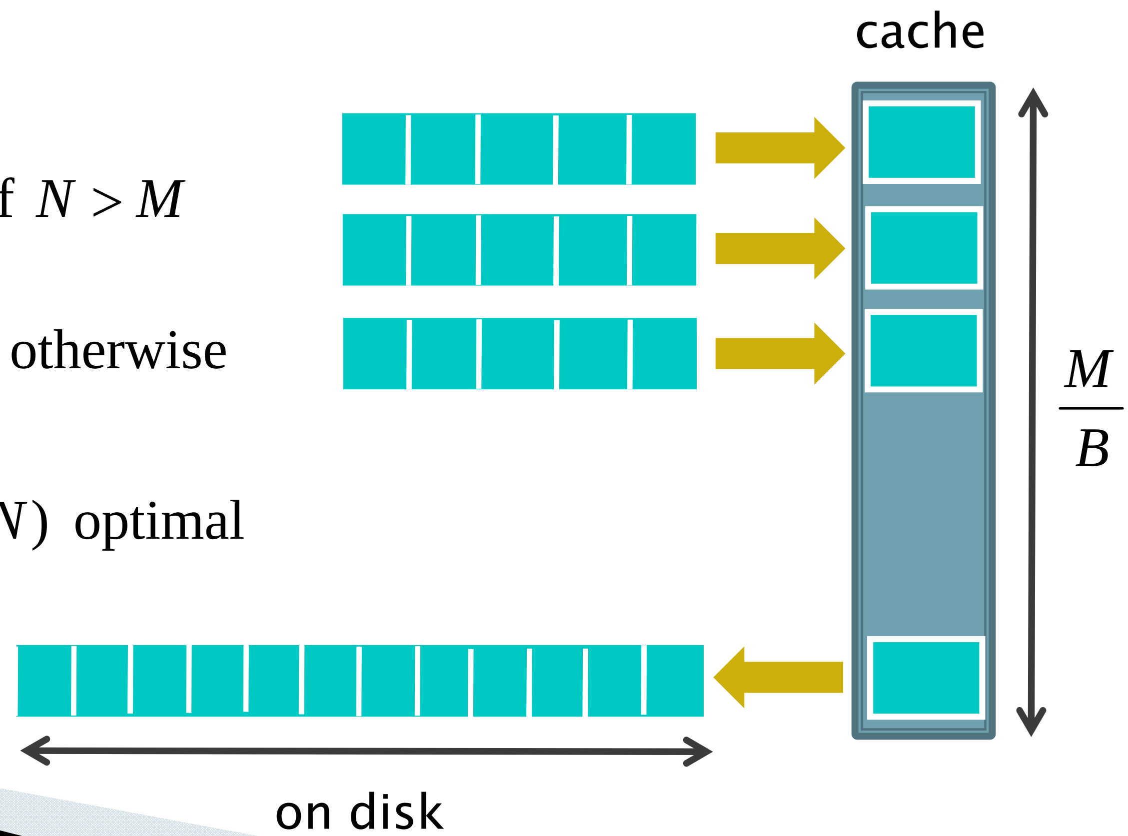
M/B-way merge sort of an N-elements array

$$W(N) = \begin{cases} \frac{M}{B} W\left(\frac{N}{M/B}\right) + N.O\left(\log \frac{M}{B}\right) & \text{if } N > 1 \\ O(1) & \text{otherwise} \end{cases}$$

$$W(N) = O(N \log N)$$

$$Q(N) = \begin{cases} \frac{M}{B} Q\left(\frac{N}{M/B}\right) + O\left(\frac{N}{B}\right) & \text{if } N > M \\ O\left(\frac{N}{B}\right) & \text{otherwise} \end{cases}$$

$$Q(N) = O\left(\frac{N}{B} \log_{M/B} \frac{N}{B}\right) = sort(N) \text{ optimal}$$

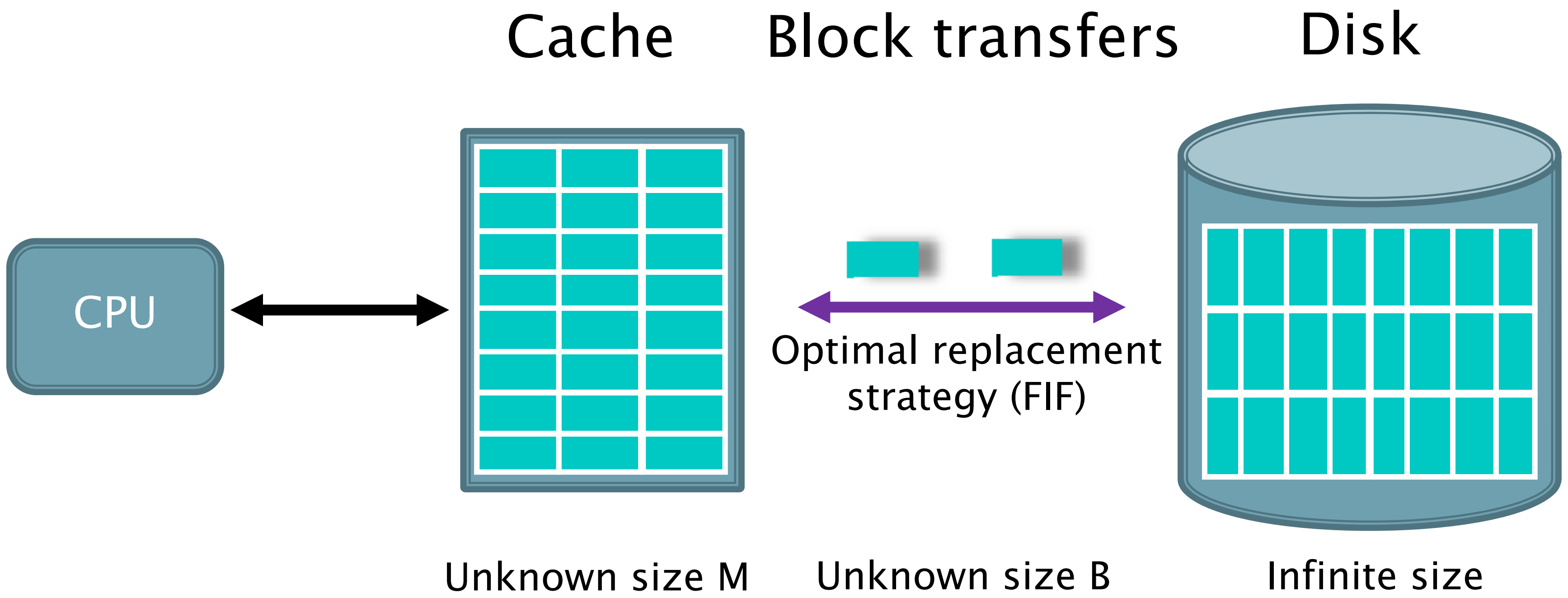


Limitations of the DAM model

- B and M are needed to design the algorithm
- Only two levels of the hierarchy
- B and M can vary
  - e.g. multi-process scheduling
- Block transfer cost is not uniform
  - disk seek time

Cache-Oblivious Model (CO)

[Frigo et al 1999]



Advantages of the CO Model

- Simple
- Parameters are unknown (block and cache size)
- Machine-independent
- Efficient with all levels of the memory hierarchy

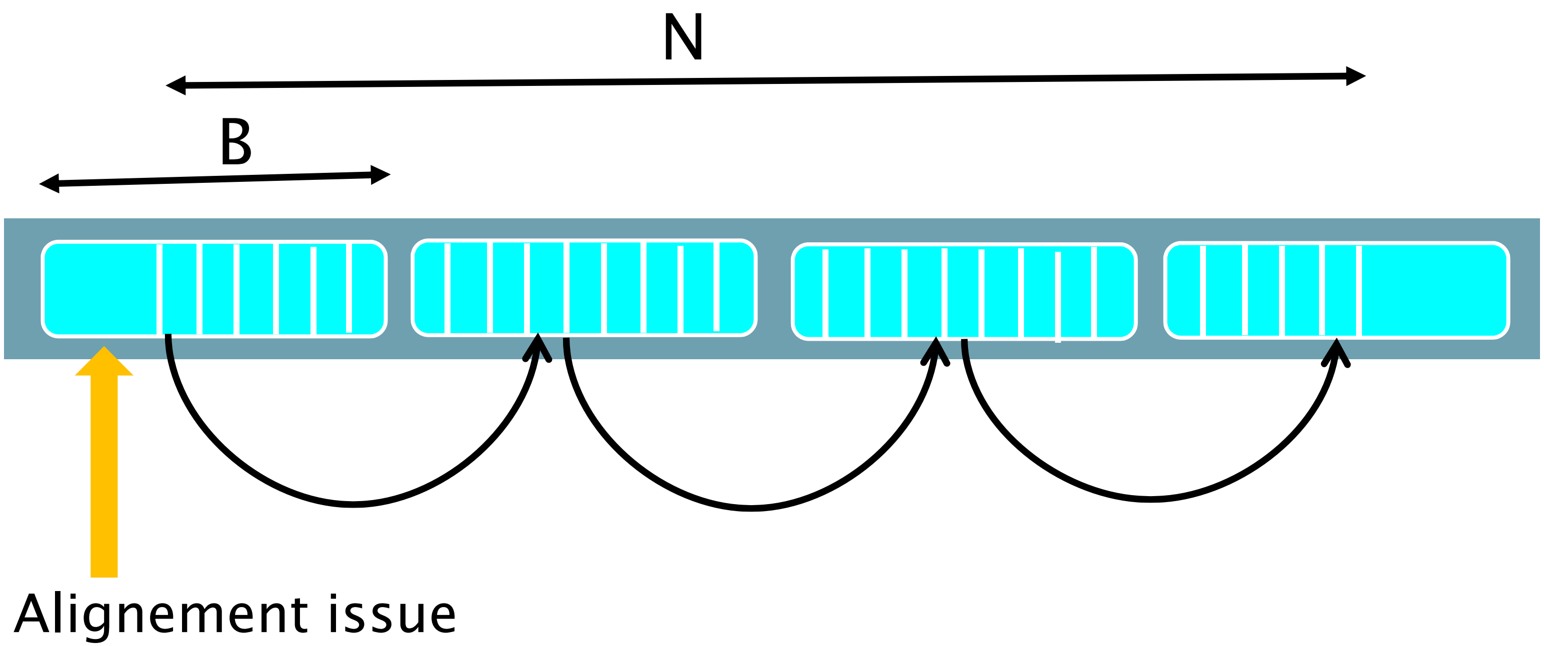
Assumptions

- Optimal replacement
- Only two levels of memory
- Full associativity
- Tall-cache assumption

$$M = \Omega(B^2)$$

$$M = \Omega(B^{1+\varepsilon})$$

Scanning in the CO model

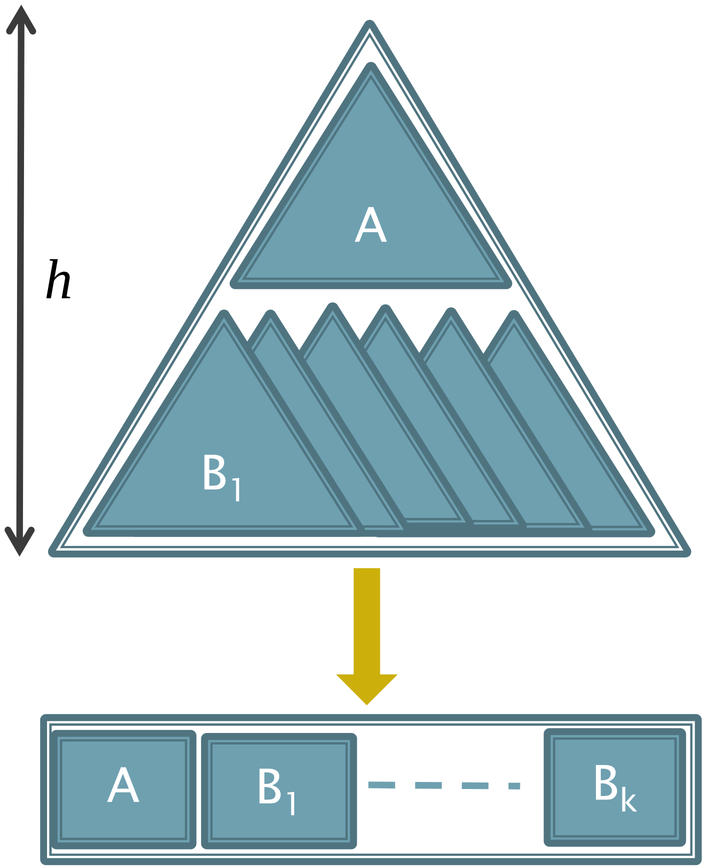


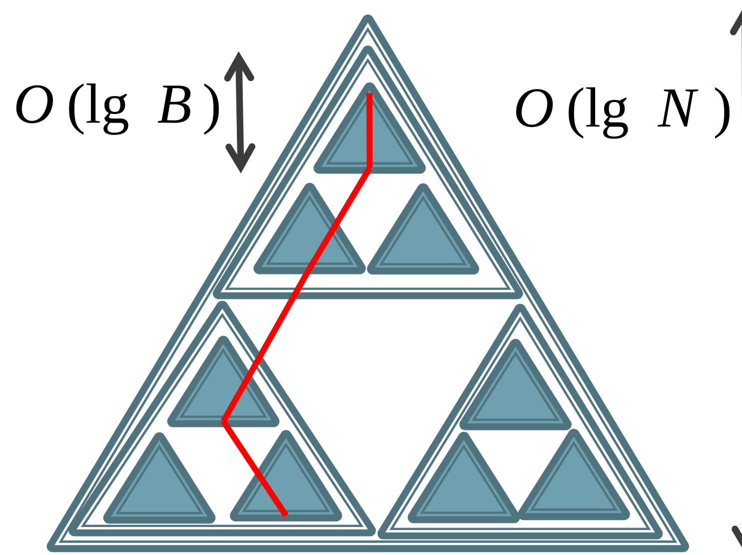
$$W(N) = N$$

$$Q(N) = \lceil N / B \rceil + 1$$

Searching in the CO model

Binary tree mapped in memory using a recursive layout [Bender et al 2000]



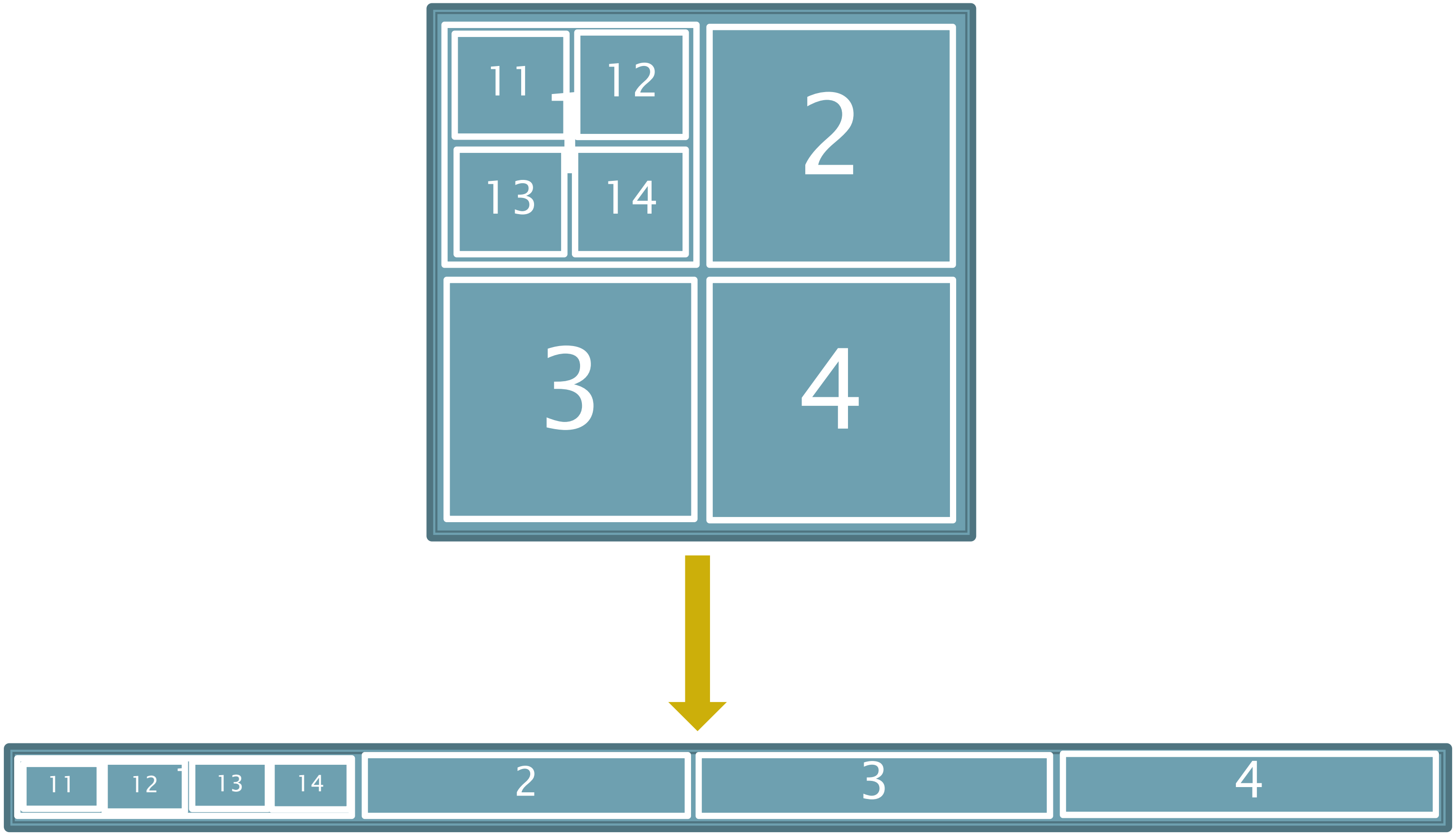


$$W(N) = O(\lg N)$$

$$Q(N) = O(1).\frac{O(\lg N)}{O(\lg B)} = O(\log_B N)$$

Multiplying in the CO model

D&C matrix multiplication using a recursive layout



Multiplying in the CO model

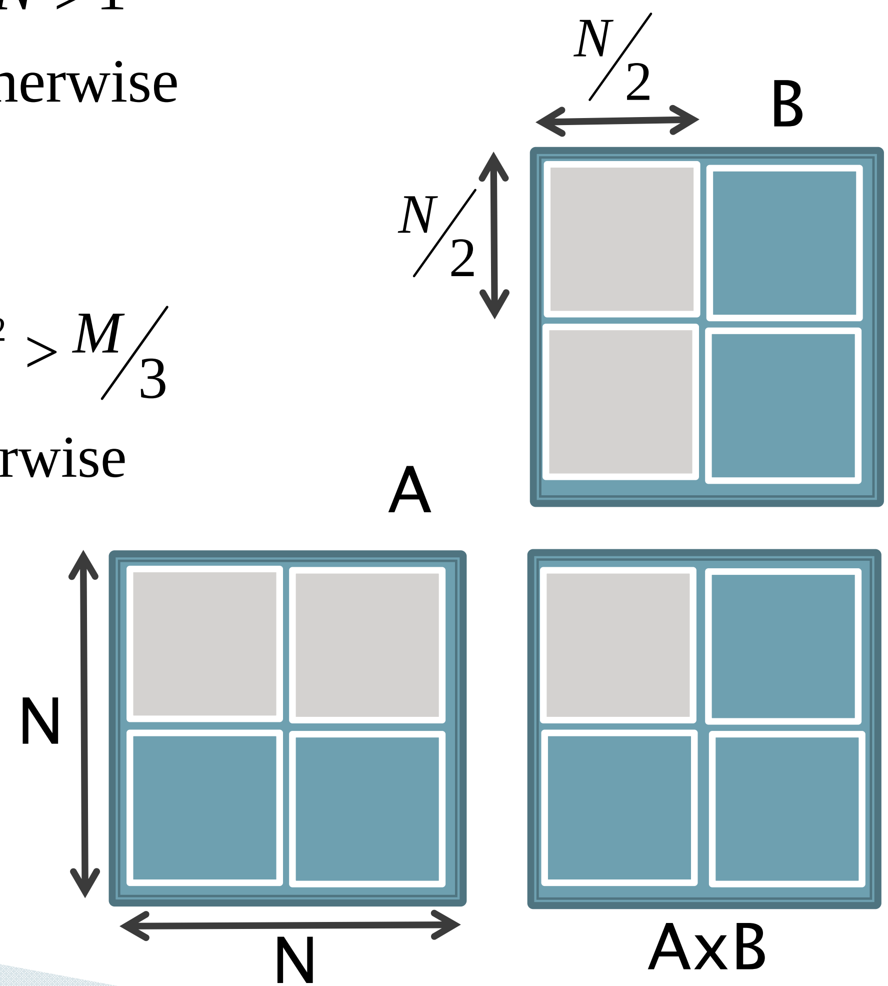
D&C matrix multiplication using a recursive layout

$$W(N) = \begin{cases} 8W\left(N/2\right) + O\left(N^2\right) & \text{if } N > 1 \\ O(1) & \text{otherwise} \end{cases}$$

$$W(N) = O\left(N^3\right)$$

$$Q(N) = \begin{cases} 8Q\left(N/2\right) + O\left(N^2/B\right) & \text{if } N^2 > M/3 \\ O\left(N^2/B\right) & \text{otherwise} \end{cases}$$

$$Q(N) = O\left(N^3 / B\sqrt{M}\right)$$



Packed Memory Array

[Itai et al 81]

[Bender et al 00,05]

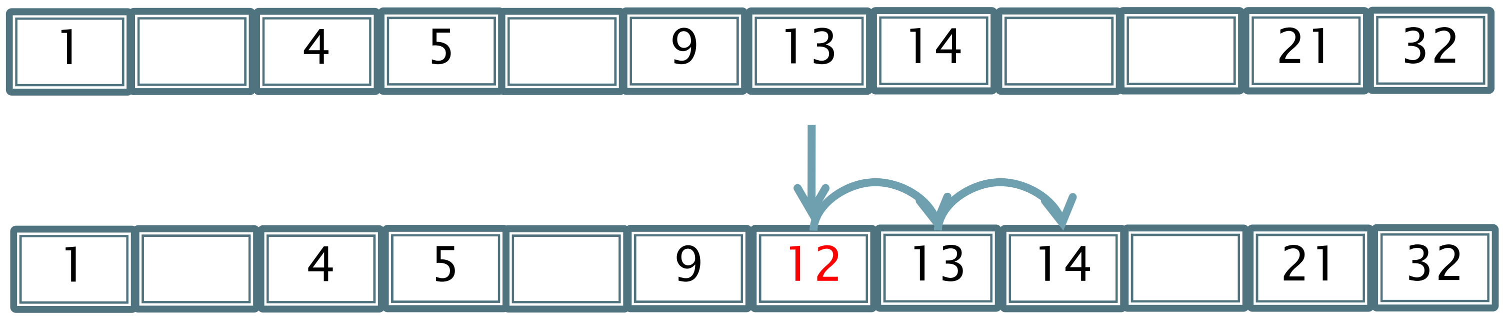
- Dynamically maintains N elements in order in a $\Theta(N)$-sized array with gaps

- Motivation: keep data in order on disk
  - Sequential block accesses are faster
  - Take advantage of prefetching
  - Range query

1		4	5		9	13	14			21	32

Packed Memory Array

- Idea: rearrange elements & gaps to accommodate future insertions

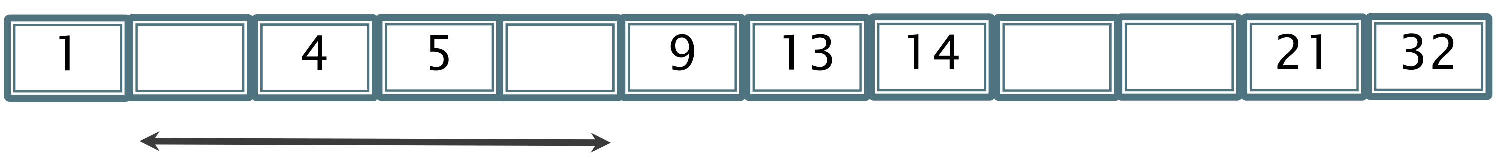


- Objective: minimize amortized number of elements moved per update

Packed Memory Array

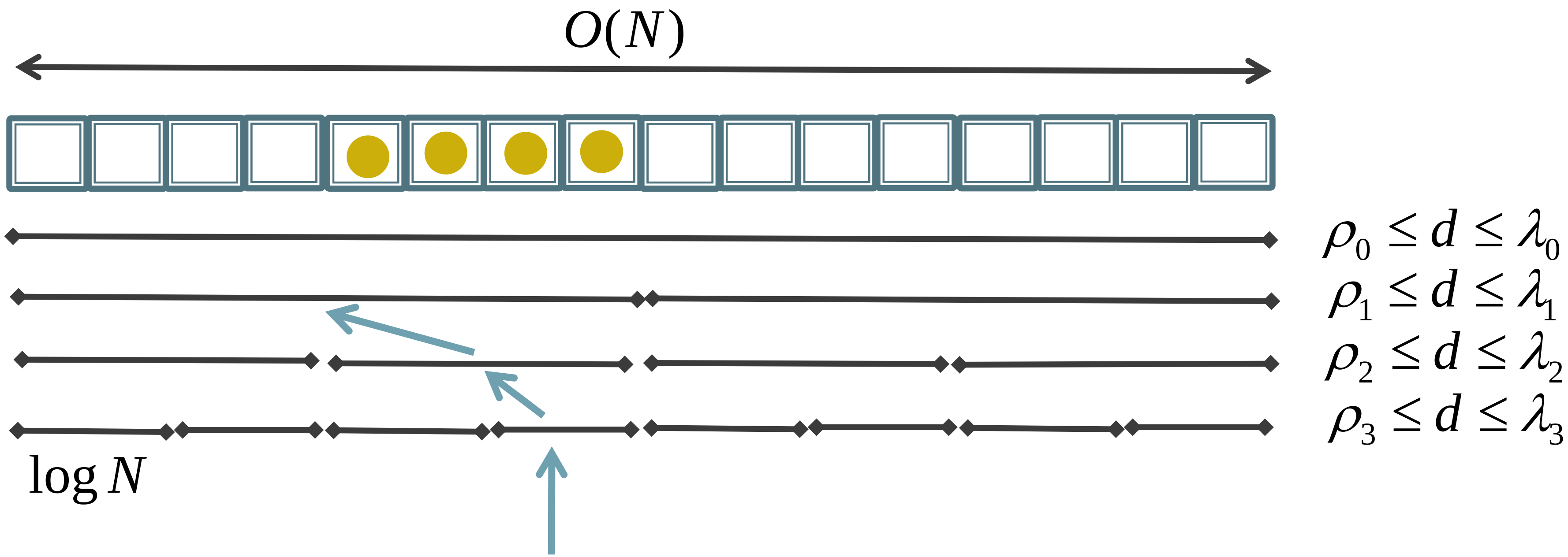
- Insertions/Deletions:
  - $O(\log^2 N)$ amortized moves per insert
  - $O\left(\log^2 N / B\right)$ amortized memory transfers

- Scans of k elements
  - $O(k / B)$ memory transfers



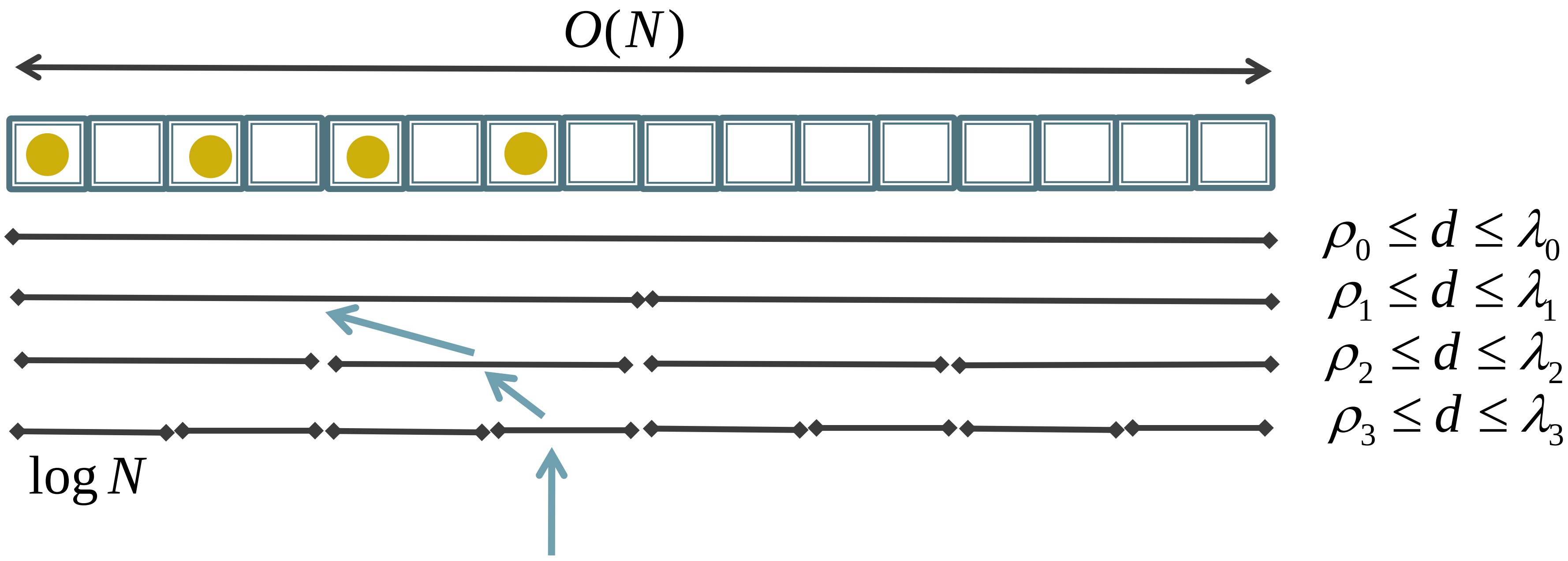
Θ(k) elements in an
interval of size k

Packed Memory Array



- Try to insert in a leaf interval
- If full, find the closest ancestor within threshold

Packed Memory Array

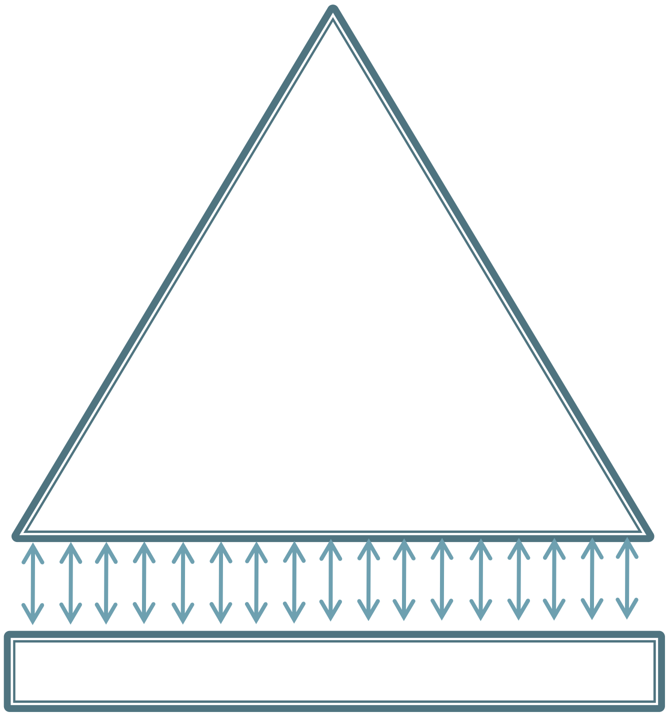


- Try to insert in a leaf interval
- If full, find the closest ancestor within threshold
- Rebalance elements uniformly in this interval

Dynamic CO B-tree

Static CO B-tree on top of the PMA [Bender et al 00]

- Search $O(\log_B N)$
- Update $O\left(\log_B N + \log^2 N / B\right)$
- Range query $O(\log_B N + k / B)$



Back to the assumptions [Frigo et al 1999]

- Optimal replacement [Sleator & Tarjan 1985]

cache misses on a (M,B) LRU-cache is at most twice the number of misses on a (M/2,B) ideal-cache

- Only two levels of memory

LRU + $\text{cache}_i \subset \text{cache}_{i+1}$ ⇨ optimal on all levels

- Full associativity

universal hash function

Related Work

- Matrix transposition
- FFT
- Search tree
- Sorting
- Priority queue
- Graph algorithms
- Computational Geometry
- Mesh layouts

CO technics

- Scanning
- Sorting
- Divide and Conquer
- Recursive layout

CA vs CO?

[Brodal and Fagerberg 03]

- CO sorting requires the tall cache assumption
- CO permuting cannot match the CA bound
- Experimental comparison [Gunnels et al 07]

+ Efficient with all levels of the hierarchy
+ Machine independent
+ Keep data in order on disk
  - On disk sequential block transfers are much faster
  - Prefetching