Small Refinements to the DAM Can Have Big Consequences for Data-Structure Design

Michael A. Bender, Alex Conway, Martín Farach-Colton, William Jannen, Yizheng Jiao, Rob Johnson, Eric Knorr, Sara McAllister, Nirjhar Mukherjee, **Prashant Pandey**, Donald E. Porter, Jun Yuan, Yang Zhan

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL
RUTGERS
Stony Brook University
Williams
vmware
HARVEY MUDD COLLEGE
Carnegie Mellon University
PACE UNIVERSITY

I/O models → predict performance

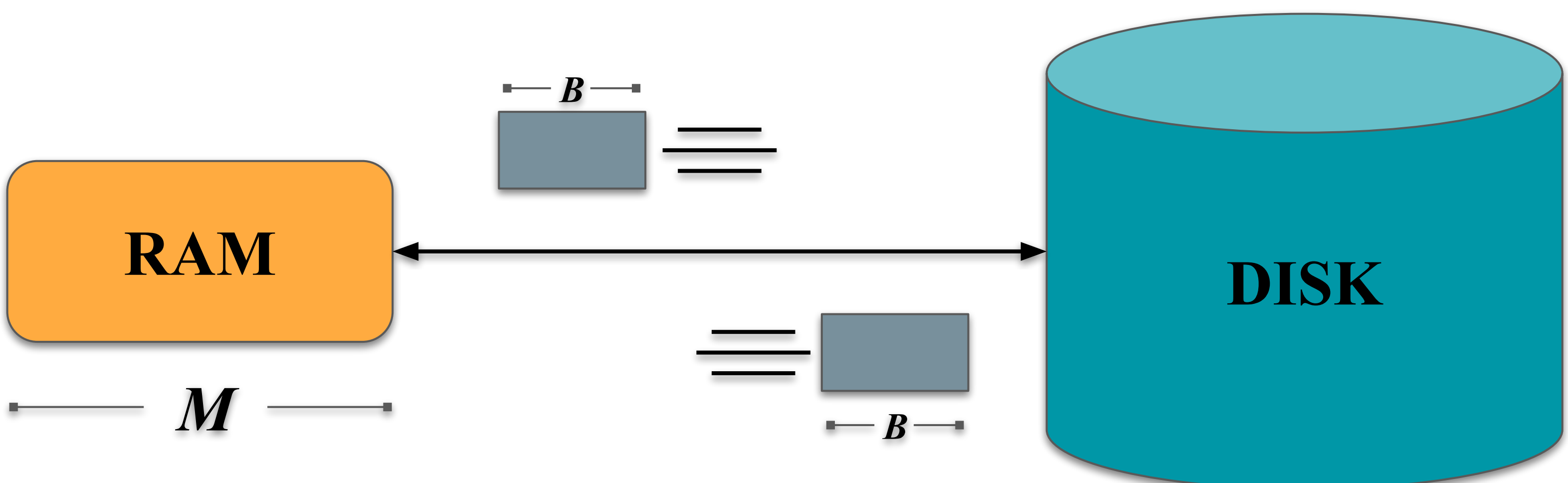
I/Os are slow! Even fast I/Os!

Common to all storage: block I/Os

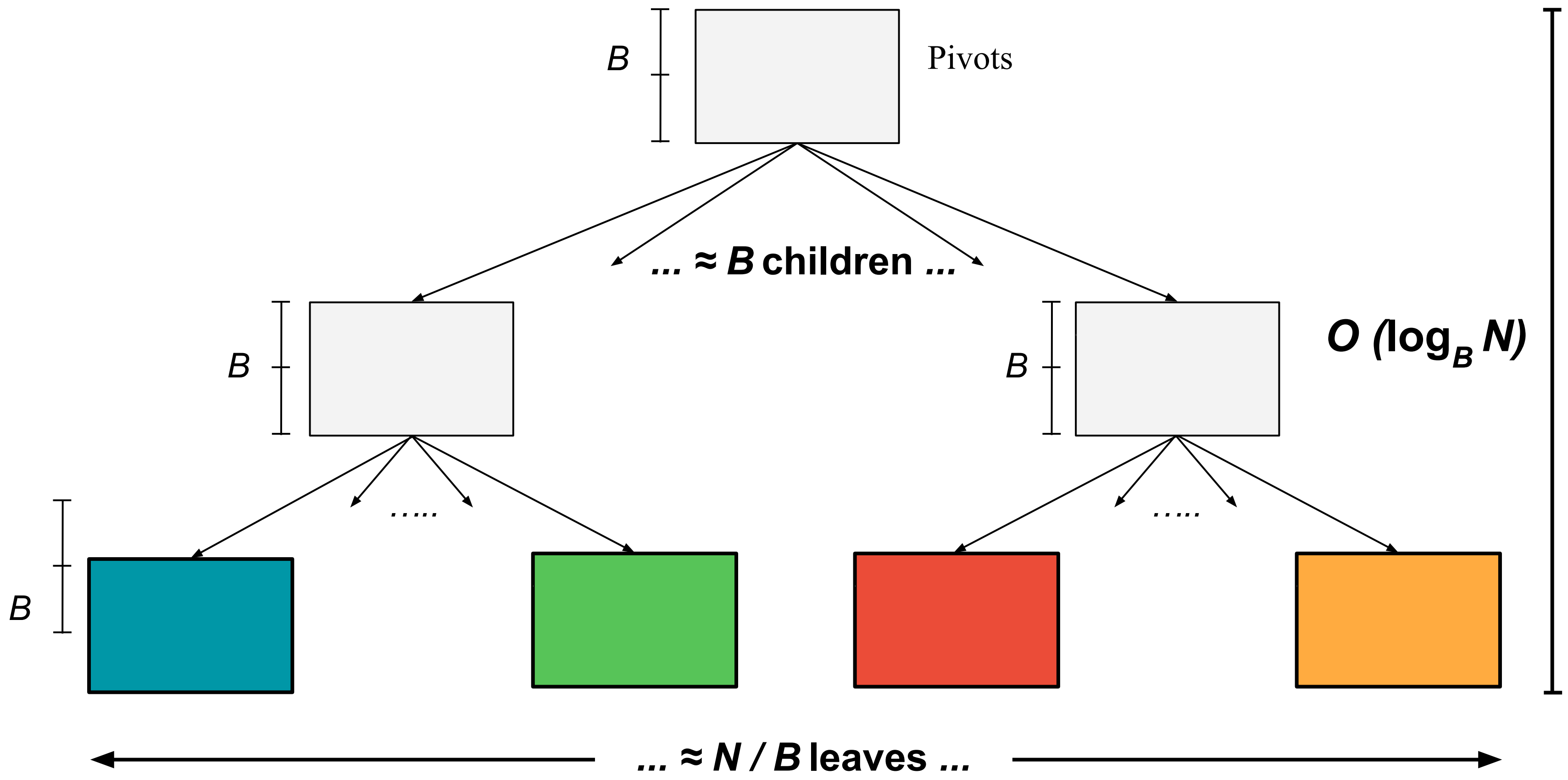
The DAM model: de facto for external memory

Aggarwal+Vitter '88

- **How computations work:**
  - Data is transferred in blocks between RAM and disk.
  - The number of block transfers dominate the running time.
- **Goal: Minimize number of block transfers**
  - Performance bounds are parameterized by block size B, memory size M, data size N.



B-tree: a classic external memory data structure



B-trees in the DAM model

	Insert	Point query
B-tree (DAM)	$O\left(\log_B N\right)$	$O\left(\log_B N\right)$

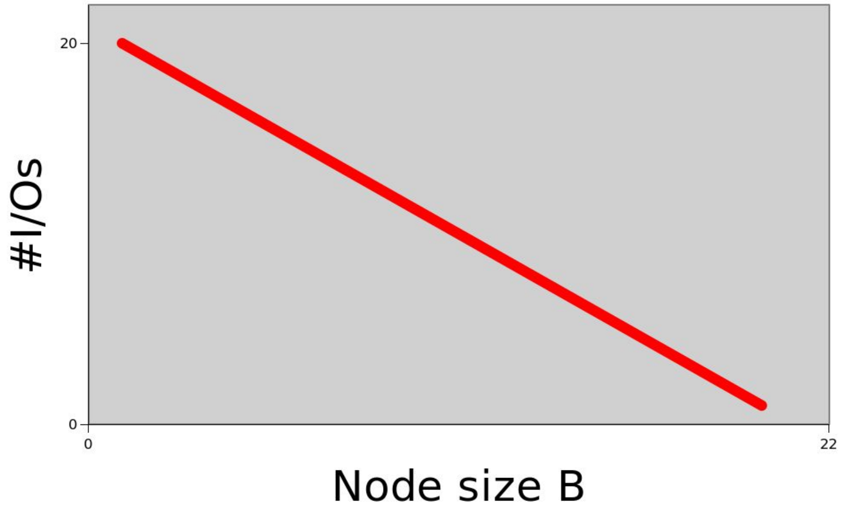
B-trees in the DAM model

	Insert	Point query
B-tree (DAM)	$O\left(\log_B \frac{N}{M}\right)$	$O\left(\log_B \frac{N}{M}\right)$

B-trees in the DAM model

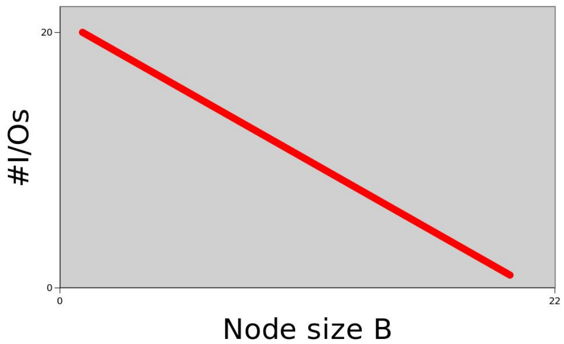
	Insert	Point query
B-tree (DAM)	$O\left(\frac{1}{\log B}\log\frac{N}{M}\right)$	$O\left(\frac{1}{\log B}\log\frac{N}{M}\right)$

DAM analysis

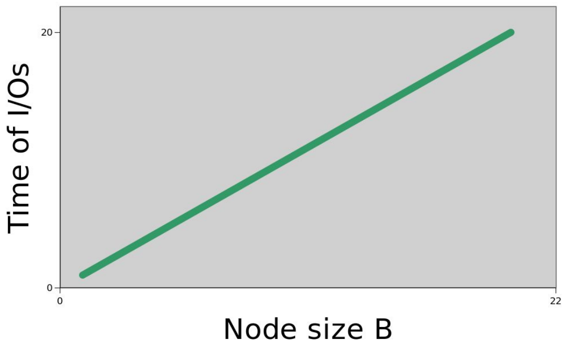


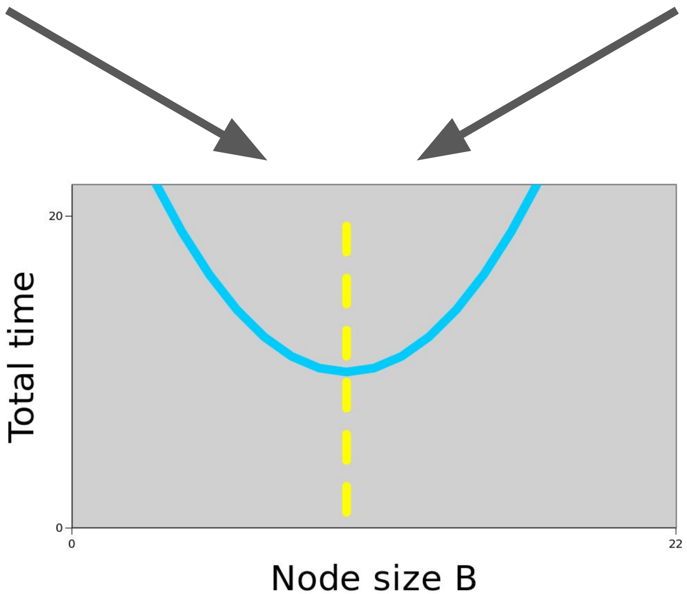
DAM ❌ Hardware = Profit

DAM analysis



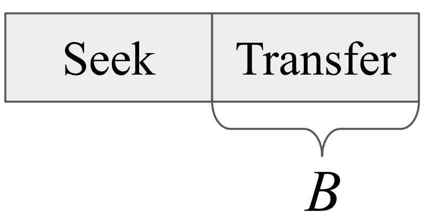
Hardware measurements

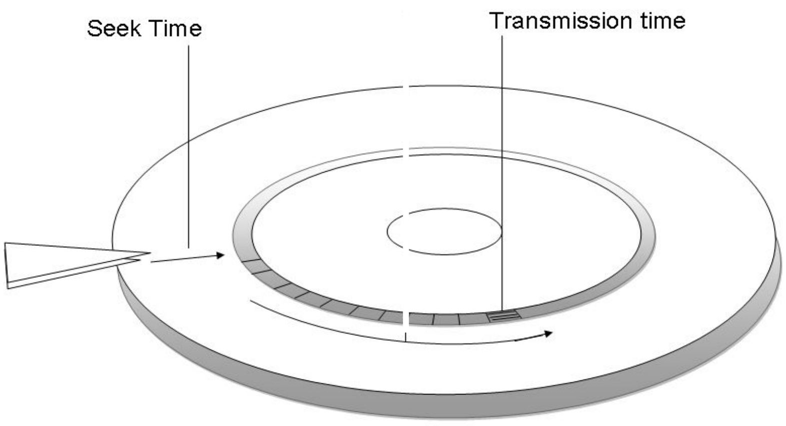




Why is one B better than another B?

- Latency $\rightarrow$ seek time
- Bandwidth $\rightarrow$ transfer cost
- ***Half-bandwidth point***, B where latency = bandwidth

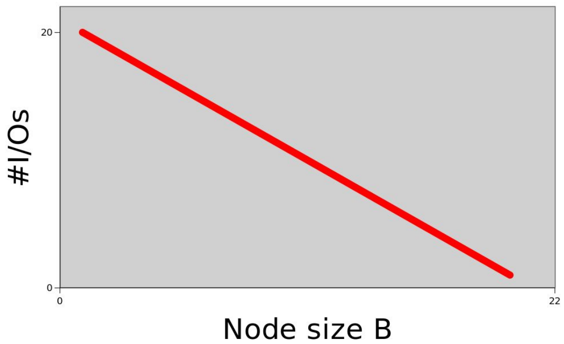




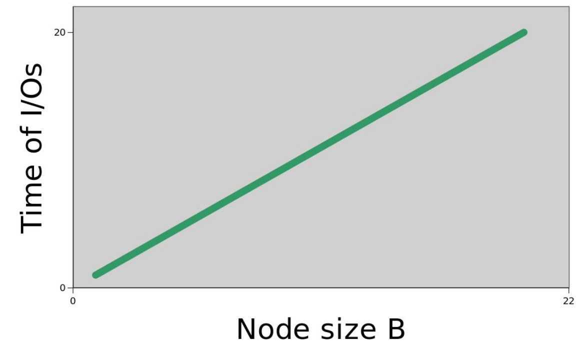
When B = half-bandwidth point,
the DAM 2-approximates the I/O cost

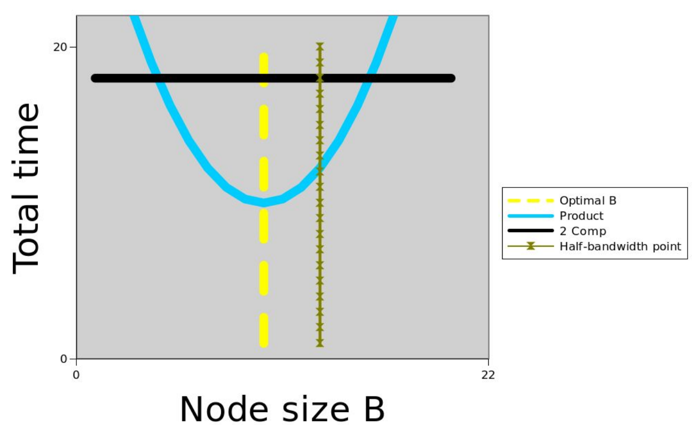
DAM ❌ Hardware = Profit

DAM analysis



Hardware measurements





An example of the half-bandwidth point

- HDD: Bandwidth = **300MB/sec** and Seek time = **5 millisec**
  - Half-bandwidth point = **1.5MB**

An example of the half-bandwidth point

- HDD: Bandwidth = **300MB/sec** and Seek time = **5 millisec**
  - Half-bandwidth point = **1.5MB**
- In practice,
  - B-trees node sizes = **4KB-64KB**
  - B^{ε}-trees (explained later) node sizes = **4MB**

An example of the half-bandwidth point

- HDD: Bandwidth = **300MB/sec** and Seek time = **5 millisec**
  - Half-bandwidth point = **1.5MB**
- In practice,
  - B-trees node sizes = **4KB-64KB**
  - B^{ε}-trees (explained later) node sizes = **4MB**
- **Why deviate from the half-bandwidth point?**

An example of the half-bandwidth point

- HDD: Bandwidth = **300MB/sec** and Seek time = **5 millisec**
  - Half-bandwidth point = **1.5MB**
- In practice,
  - B-trees node sizes = **4KB-64KB**
  - B$^{\varepsilon}$-trees (explained later) node sizes = **4MB**
- **Why deviate from the half-bandwidth point?**

The DAM model cannot answer these questions because
B is a parameter of the model.

This paper: using refined models for HDDs and SSDs

- Affine model[1, 2] : for HDDs

- PDAM model[3]: for SSDs

[1]. Matthew Andrews, Michael A. Bender, and Lisa Zhang. 2002. New Algorithms for Disk Scheduling. Algorithmica 32, 2 (2002)
[2]. C Ruemmler and J. Wilkes. 1994. An introduction to disk drive modeling. IEEE Computer
[3]. Alok Aggarwal and Jeffrey Scott Vitter. 1988. The Input/Output Complexity of Sorting and Related Problems. Commun. ACM

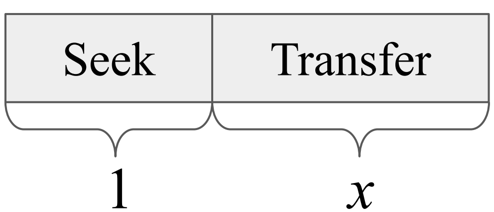
This paper: using refined models for HDDs and SSDs

- Affine model[1, 2] : for HDDs
  - **How to optimize *B*** for B-trees and B^{ε}-trees on hard drives.
  - **Explains node-size variability** in B-trees and B^{ε}-trees.
  - Reveals new **intra-node optimizations for** B^{ε}-trees.
- PDAM model[3]: for SSDs
  - Allows **parallelism-oblivious optimizations**.

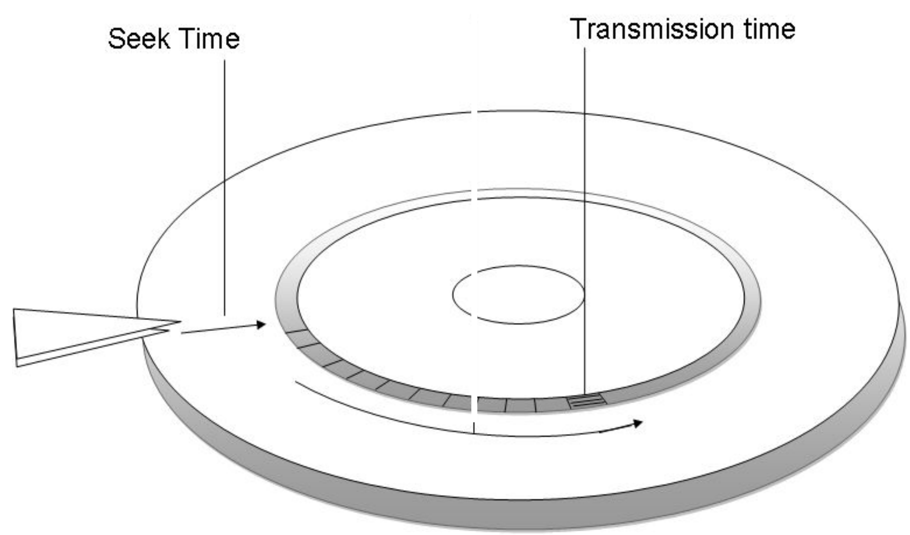
[1]. Matthew Andrews, Michael A. Bender, and Lisa Zhang. 2002. New Algorithms for Disk Scheduling. Algorithmica 32, 2 (2002)
[2]. C Ruemmler and J. Wilkes. 1994. An introduction to disk drive modeling. IEEE Computer
[3]. Alok Aggarwal and Jeffrey Scott Vitter. 1988. The Input/Output Complexity of Sorting and Related Problems. Commun. ACM

The affine model: for hard drives

- An I/O of size x words costs $1 + \alpha x$,
    - 1 is the normalized setup cost
    - $\alpha \leq 1$ is the normalized bandwidth cost
        - for spinning disks, α = *transfer time/seek time*

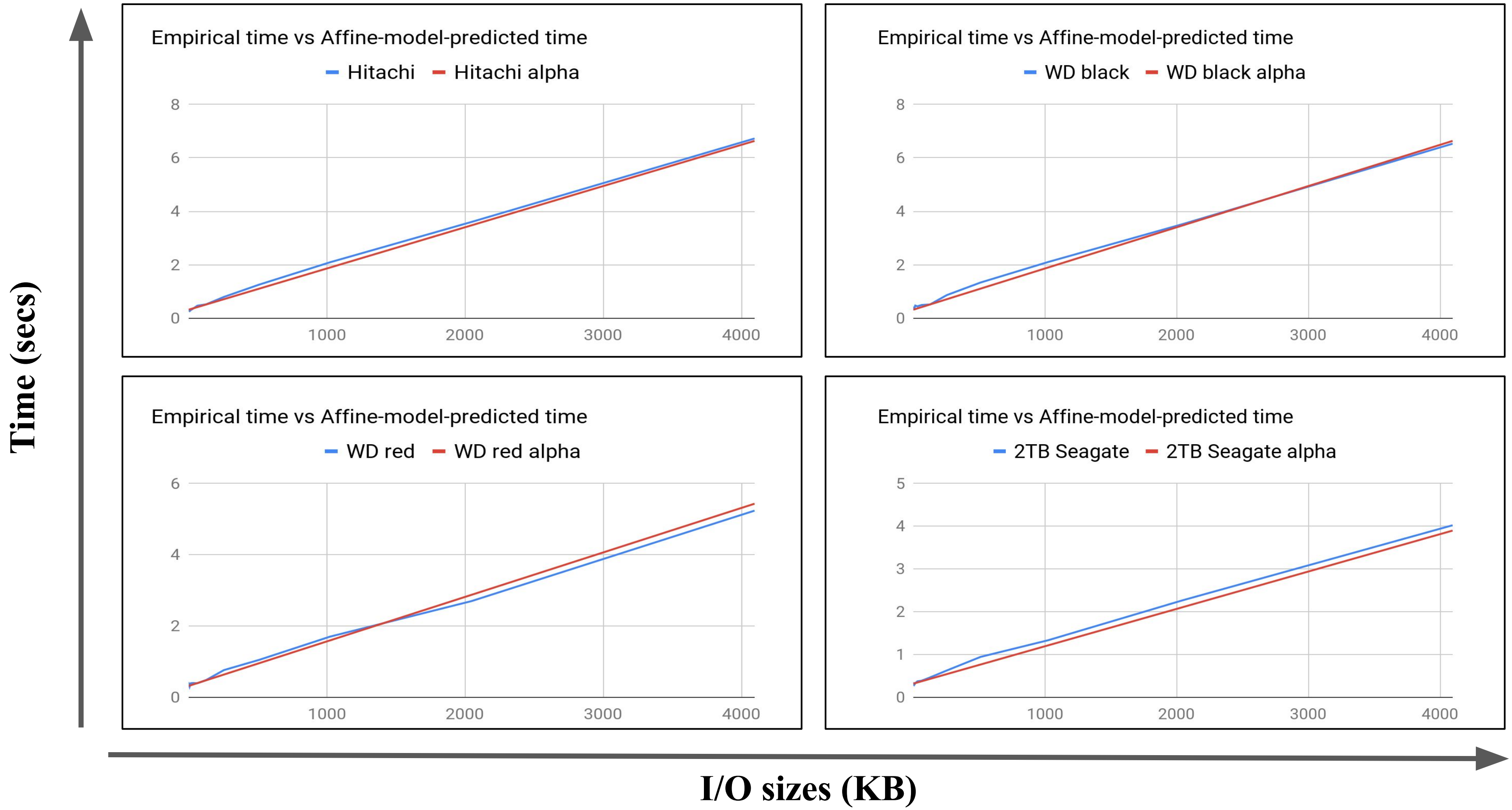


Half-bandwidth point $= \dfrac{1}{\alpha}$



http://web.cs.ucla.edu/classes/winter12/cs111/scribe/10b/

Experimental validation of the affine model



Empirical I/O cost is almost exactly same as the cost predicted by the affine model.

B-trees in the affine model

	Insert	Point query
B-tree (DAM)	$O\left(\frac{1}{\log B}\log\frac{N}{M}\right)$	$O\left(\frac{1}{\log B}\log\frac{N}{M}\right)$
B-tree (Affine)	$O\left(\frac{1+\alpha B}{\log B}\log\frac{N}{M}\right)$	$O\left(\frac{1+\alpha B}{\log B}\log\frac{N}{M}\right)$

Point queries and inserts are optimized when the node size is:

$$B = \Theta\left(\frac{1}{\alpha}\frac{1}{\ln(1/\alpha)}\right)$$

B-trees are optimized by making nodes much smaller than the half-bandwidth point

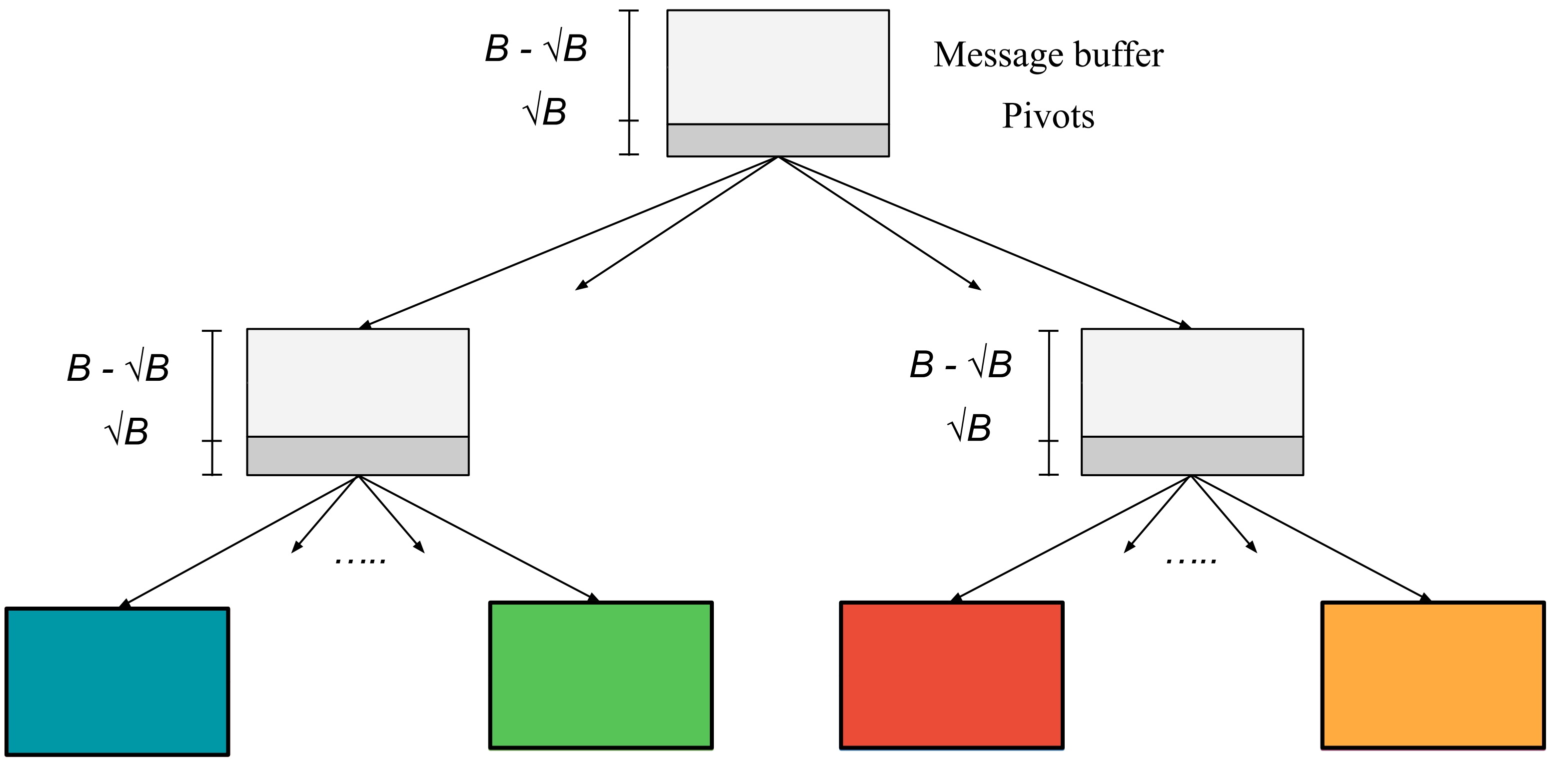
Example: optimal B-tree node size in the affine model

Seagate 2TB Hard Drive

Seek time	5 msec
bandwidth	300 MB/sec
Key-value pair size	16 bytes
ln (1/α)	11.45
Optimal node size *B*	≈130KB

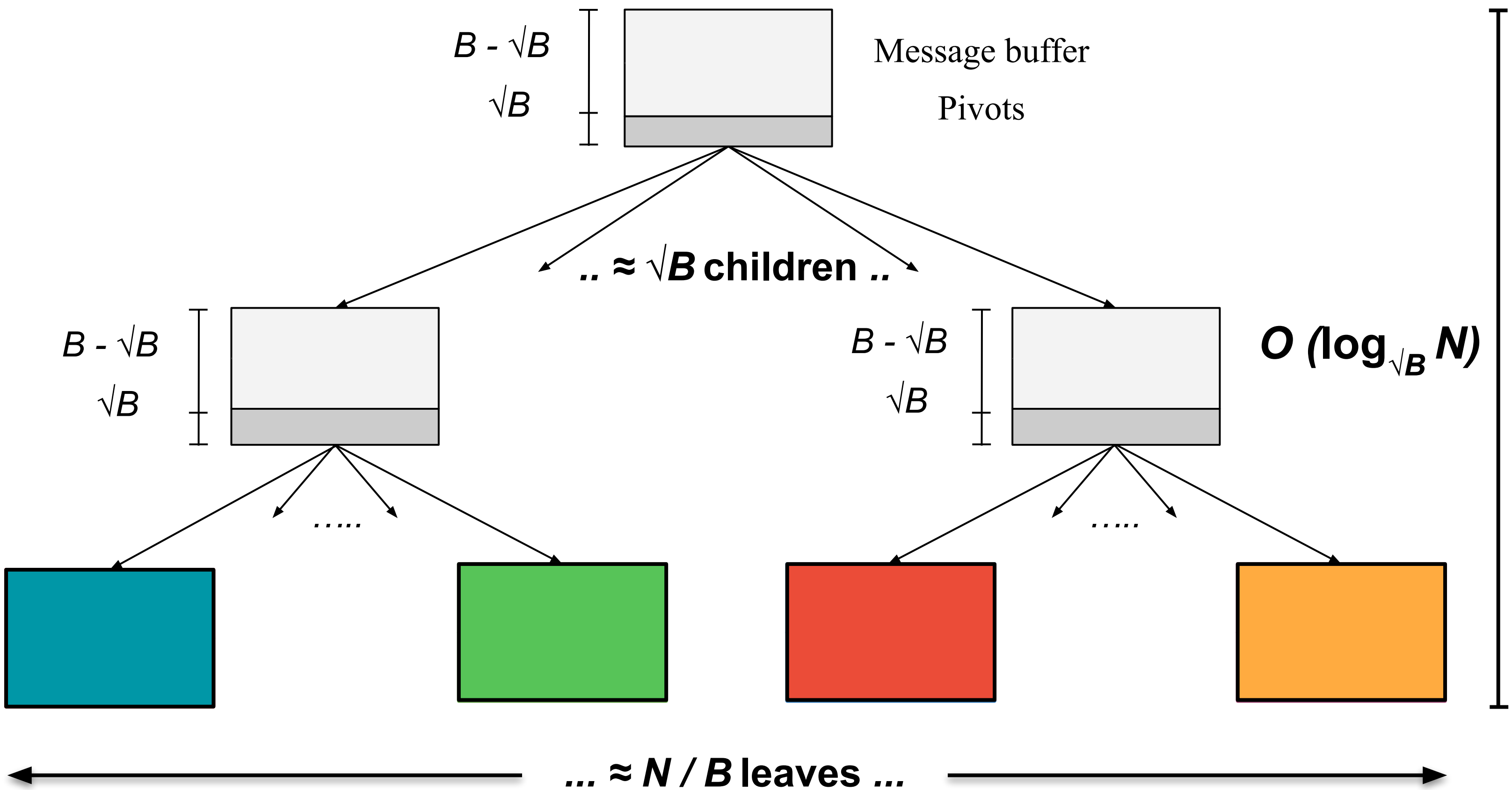
The affine model explains most of the discrepancy between the node sizes used in practice and those that optimize the DAM model.

$B^{1/2}$-tree: using nodes as buffers



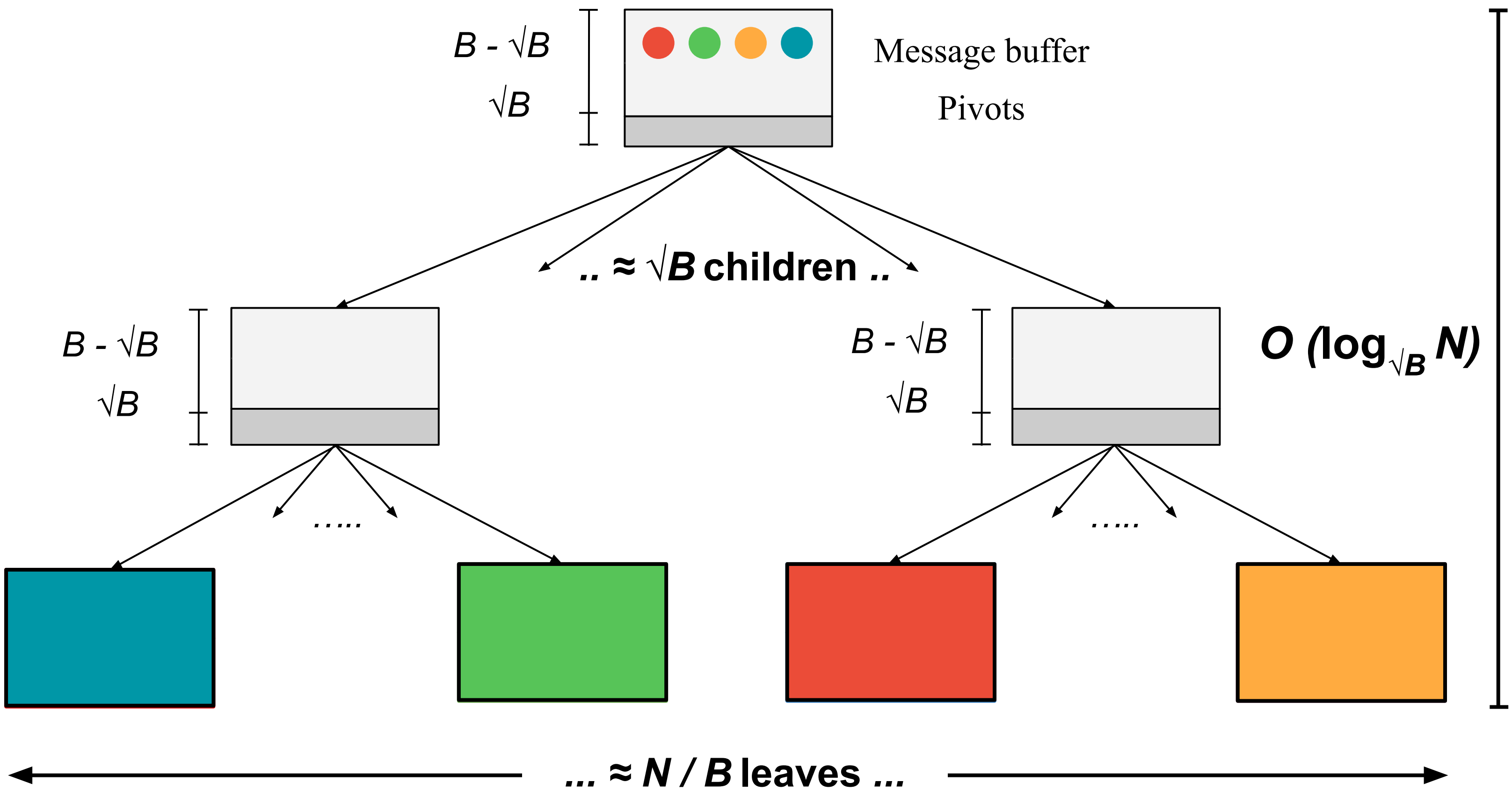
Use the node space ($B - \sqrt{B}$) as buffer for inserts and delete messages.

$B^{1/2}$-tree: buffering reduces the fanout



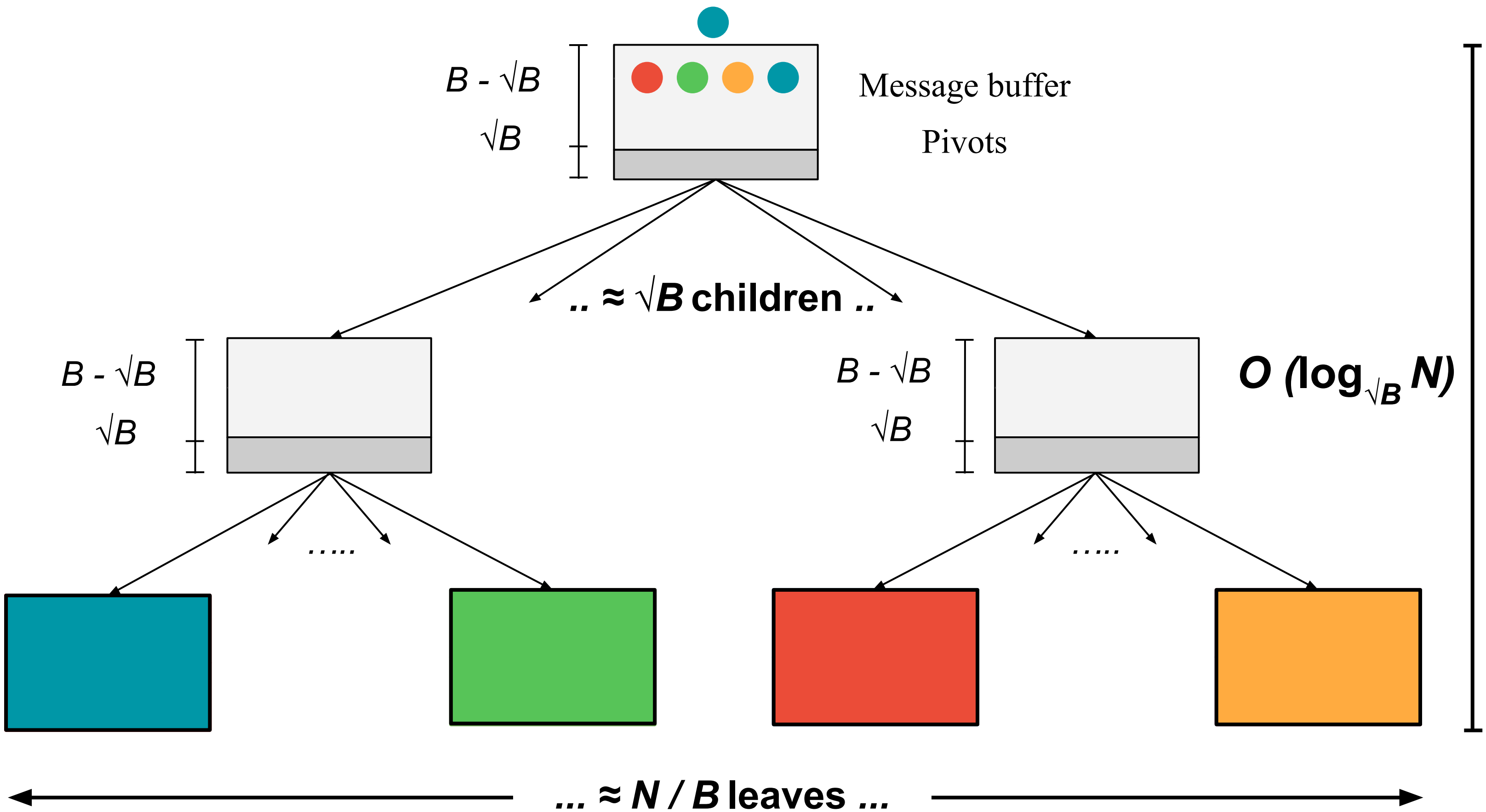
Reduce the fanout in the B-tree to $\sqrt{B}$.

$B^{1/2}$-tree: store insert/delete messages at root



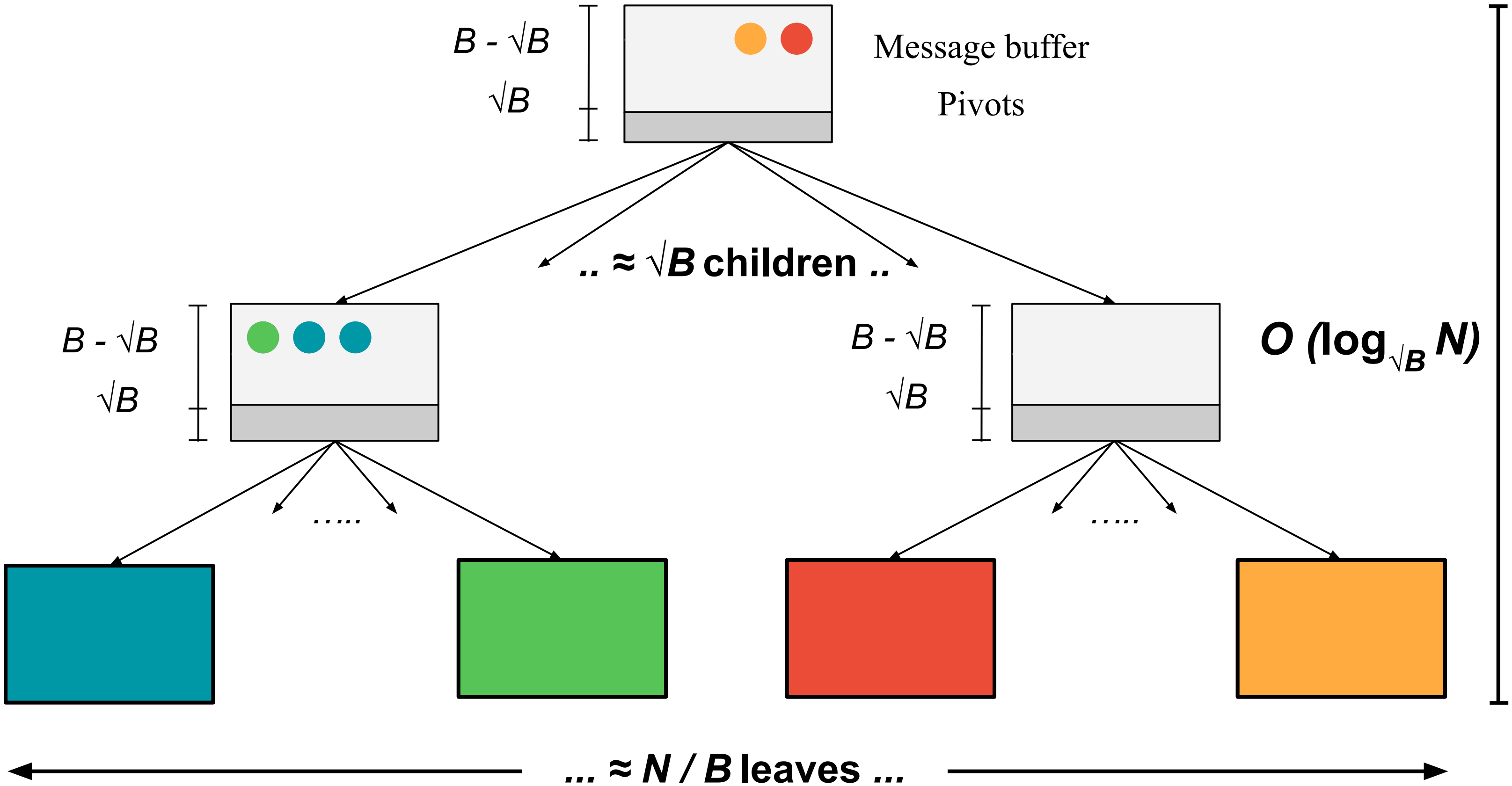
Inject insert and delete messages at the root.

$B^{1/2}$-tree: flush when root buffer is full



When buffer fills up, flush to child nodes.

$B^{1/2}$-tree: move all messages destined for a child

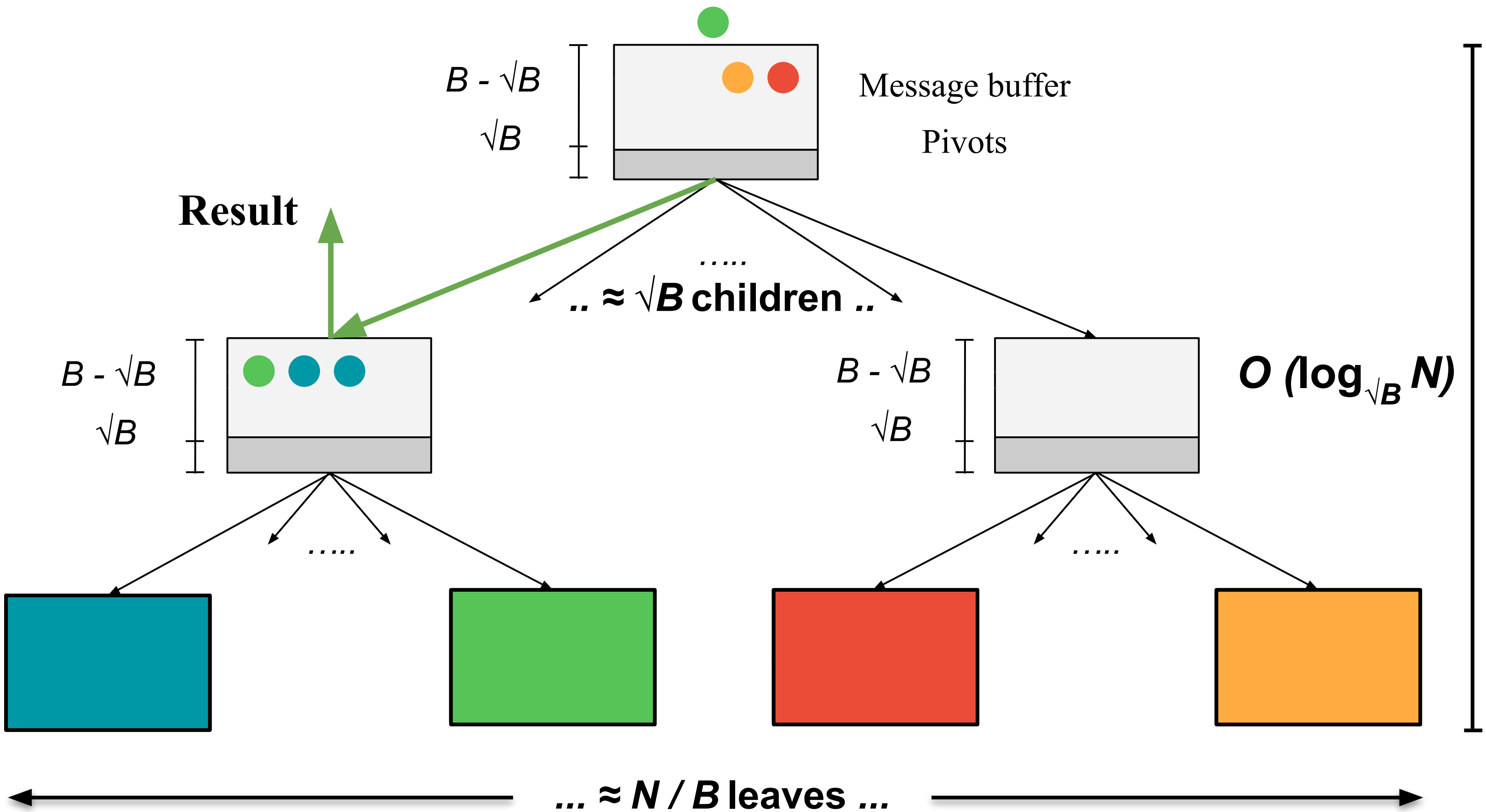


≈ $\sqrt{B}$ elements move down the tree in 1 I/O.

$B^{1/2}$-tree in the DAM model

	Insert	Point query
$B^{1/2}$-tree	$O\left(\frac{1}{\sqrt{B}}\log_{\sqrt{B}}\frac{N}{M}\right)$	

$B^{1/2}$-tree: searches cost the same as the B-tree



Examine each buffer on root-to-leaf path. 1 I/O per node.

$B^{1/2}$-tree in the DAM model

	Insert	Point query
$B^{1/2}$-tree	$O\left(\frac{1}{\sqrt{B}}\log_{\sqrt{B}}\frac{N}{M}\right)$	$O\left(\log_{\sqrt{B}}\frac{N}{M}\right)$

$B^{1/2}$-tree in the DAM model

	Insert	Point query
$B^{1/2}$-tree	$O\left(\frac{1}{\sqrt{B}\log B}\log\frac{N}{M}\right)$	$O\left(\frac{1}{\log B}\log\frac{N}{M}\right)$

$B^{1/2}$-tree in the affine model

	Insert	Point query
$B^{1/2}$-tree	$O\left(\frac{1+\alpha B}{\sqrt{B}\log B}\log\frac{N}{M}\right)$	$O\left(\frac{1+\alpha B}{\log B}\log\frac{N}{M}\right)$

$B^{1/2}$-tree in the affine model

	Insert	Point query
$B^{1/2}$-tree	$O\left(\frac{1+\alpha B}{\sqrt{B}\log B}\log\frac{N}{M}\right)$	$O\left(\frac{1+\alpha B}{\log B}\log\frac{N}{M}\right)$

Insert cost increases more slowly in $B^{1/2}$-trees than in B-trees as the node size increases.

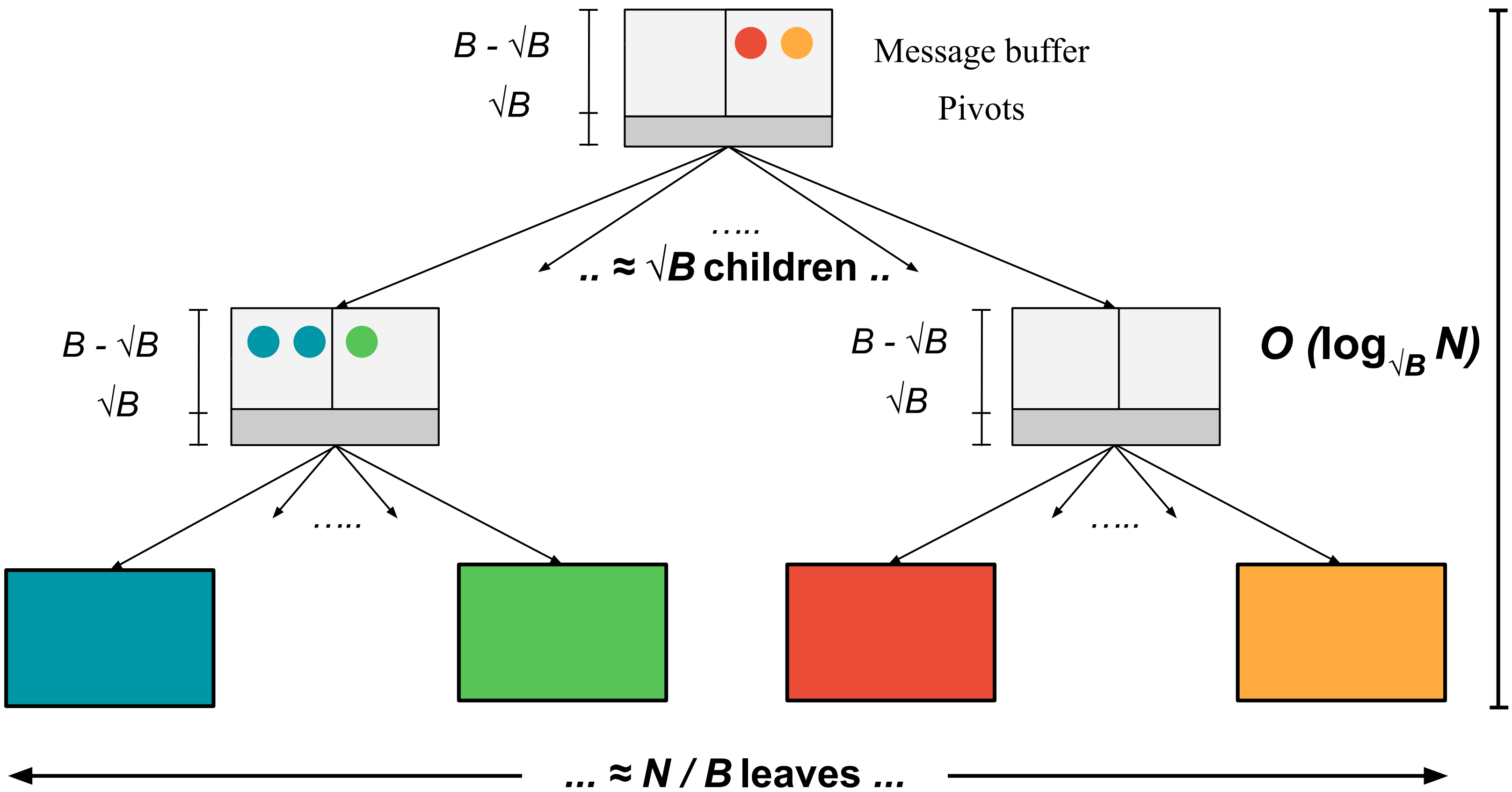
$B^{1/2}$-tree in the affine model

	Insert	Point query
$B^{1/2}$-tree	$O\left(\frac{1+\alpha B}{\sqrt{B}\log B}\log\frac{N}{M}\right)$	$O\left(\frac{1+\alpha B}{\log B}\log\frac{N}{M}\right)$

Insert cost increases more slowly in $B^{1/2}$-trees than in B-trees as the node size increases.

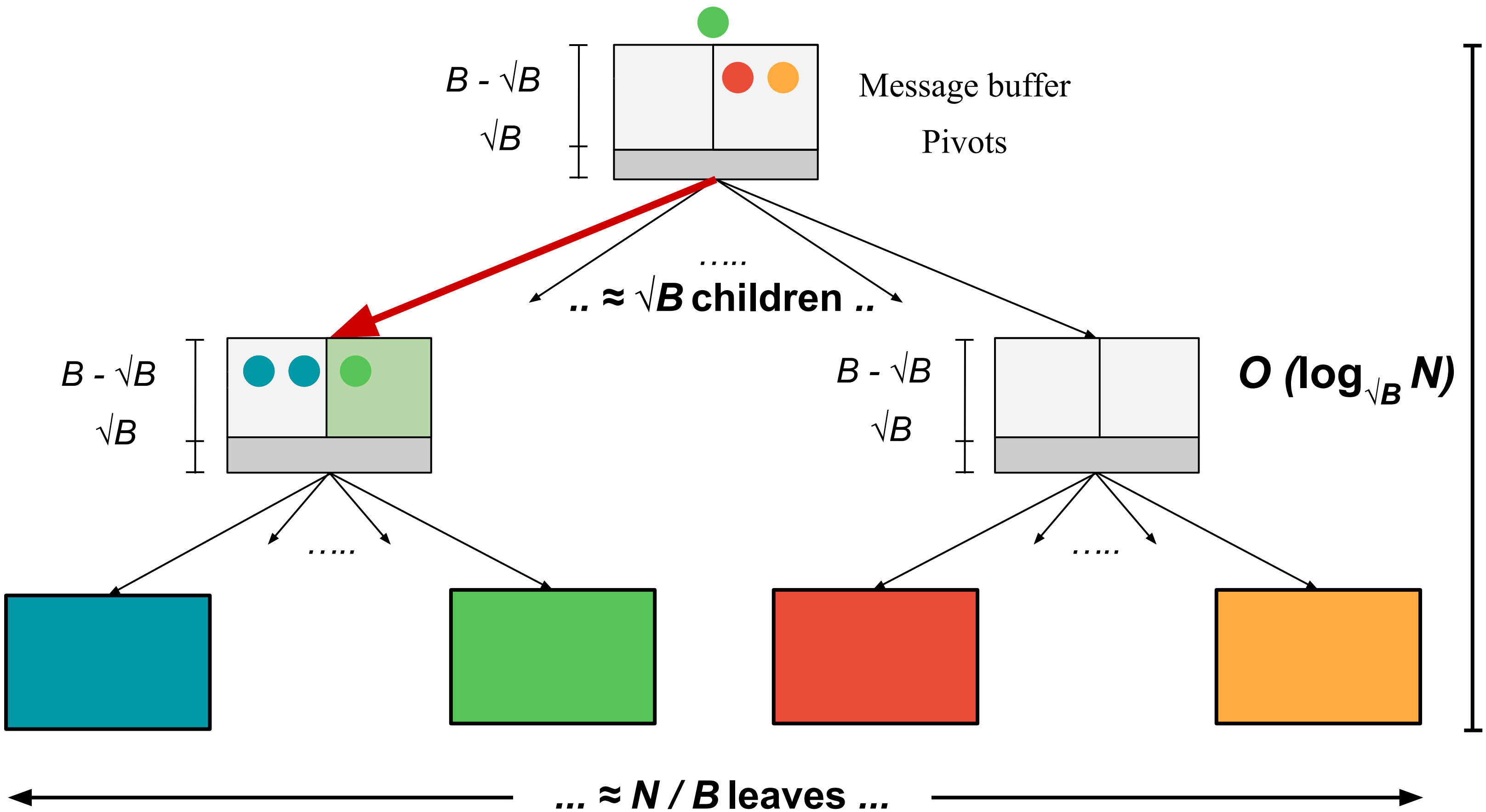
But the bandwidth cost of queries is still linear in B.

Organizing messages → 2 I/Os per node



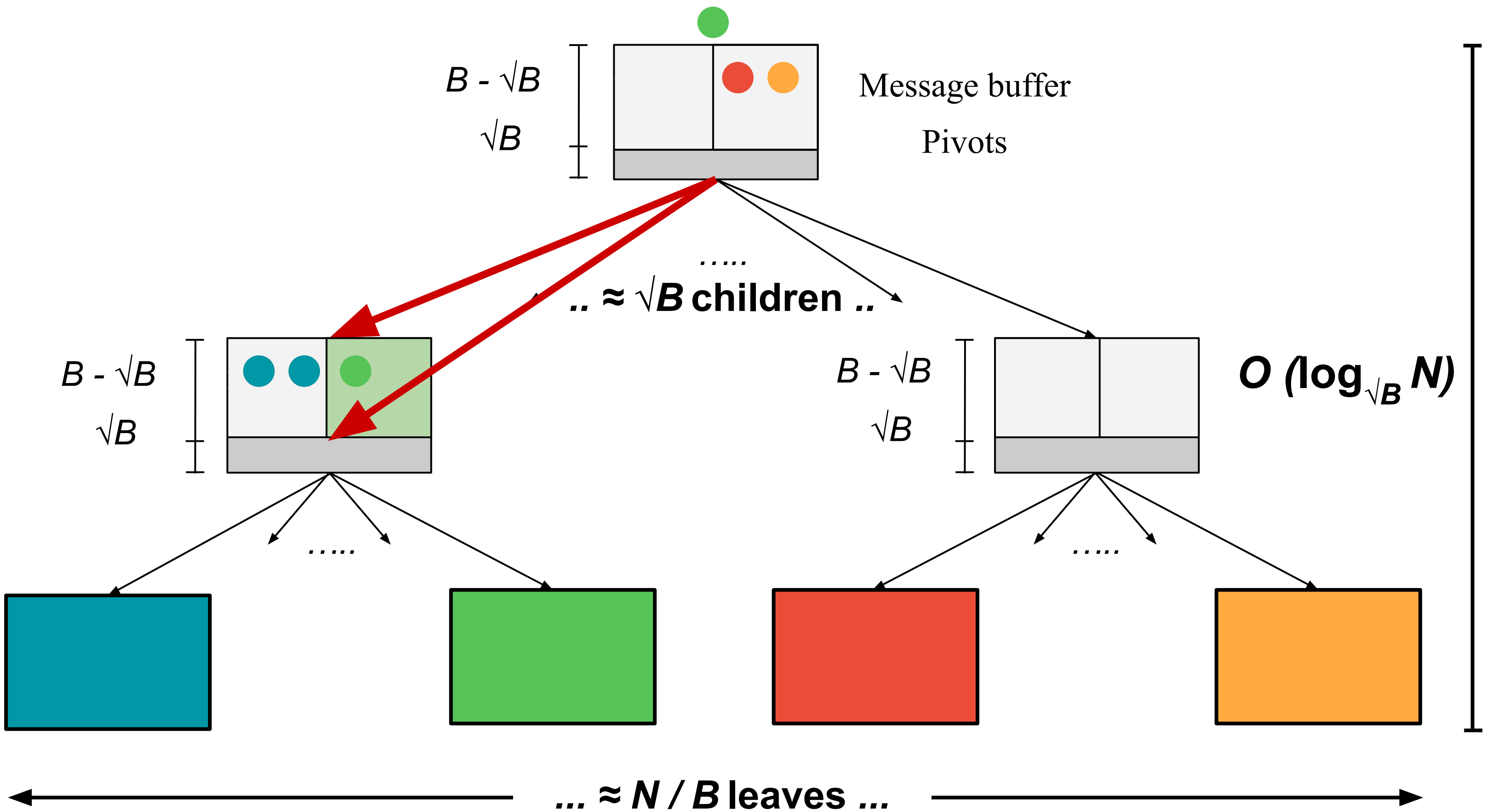
Elements destined for a particular node are stored together.

Organizing messages → 2 I/Os per node



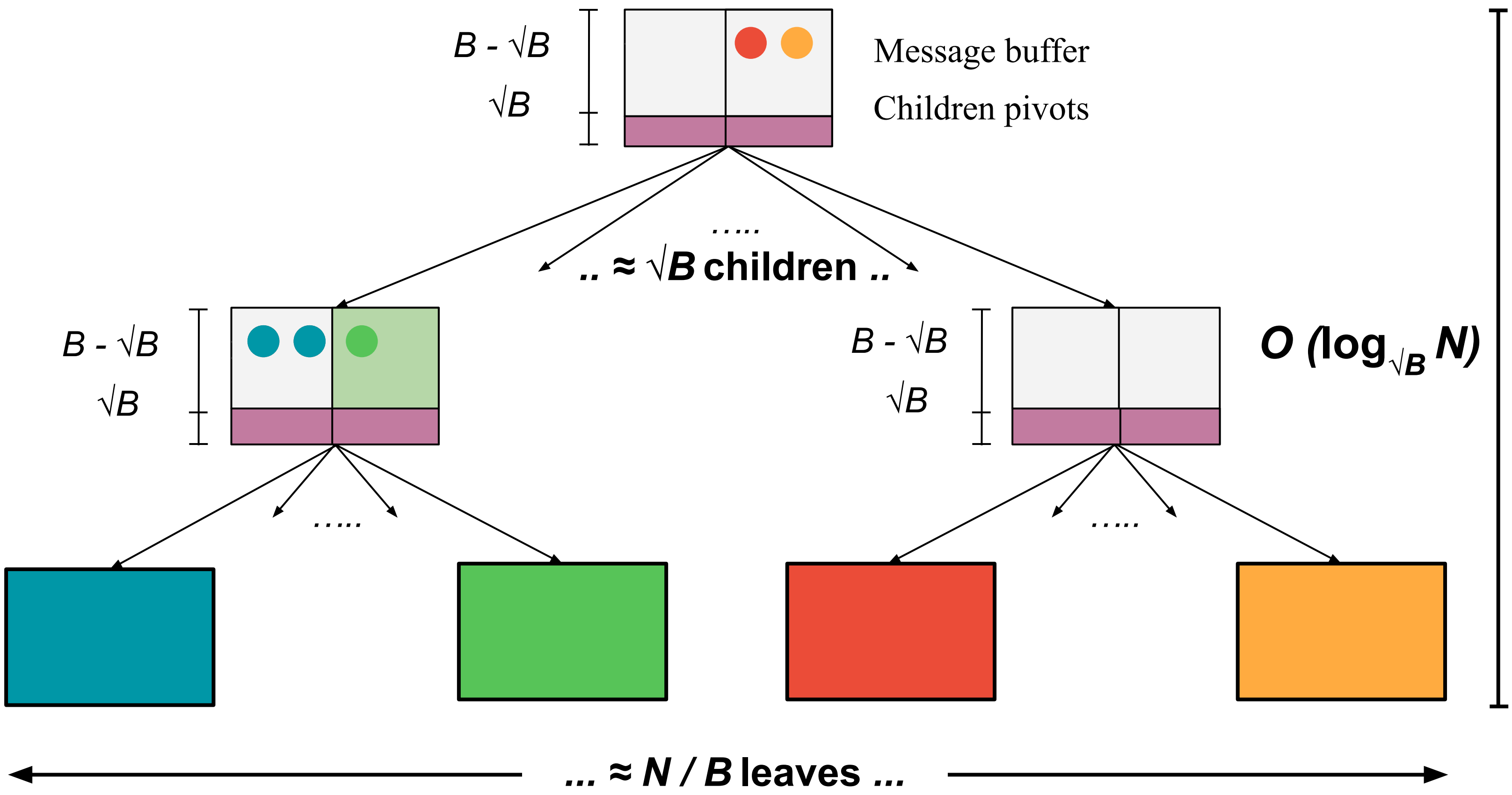
The cost to read all these elements is $1 + \alpha\sqrt{B}$.

Organizing messages → 2 I/Os per node



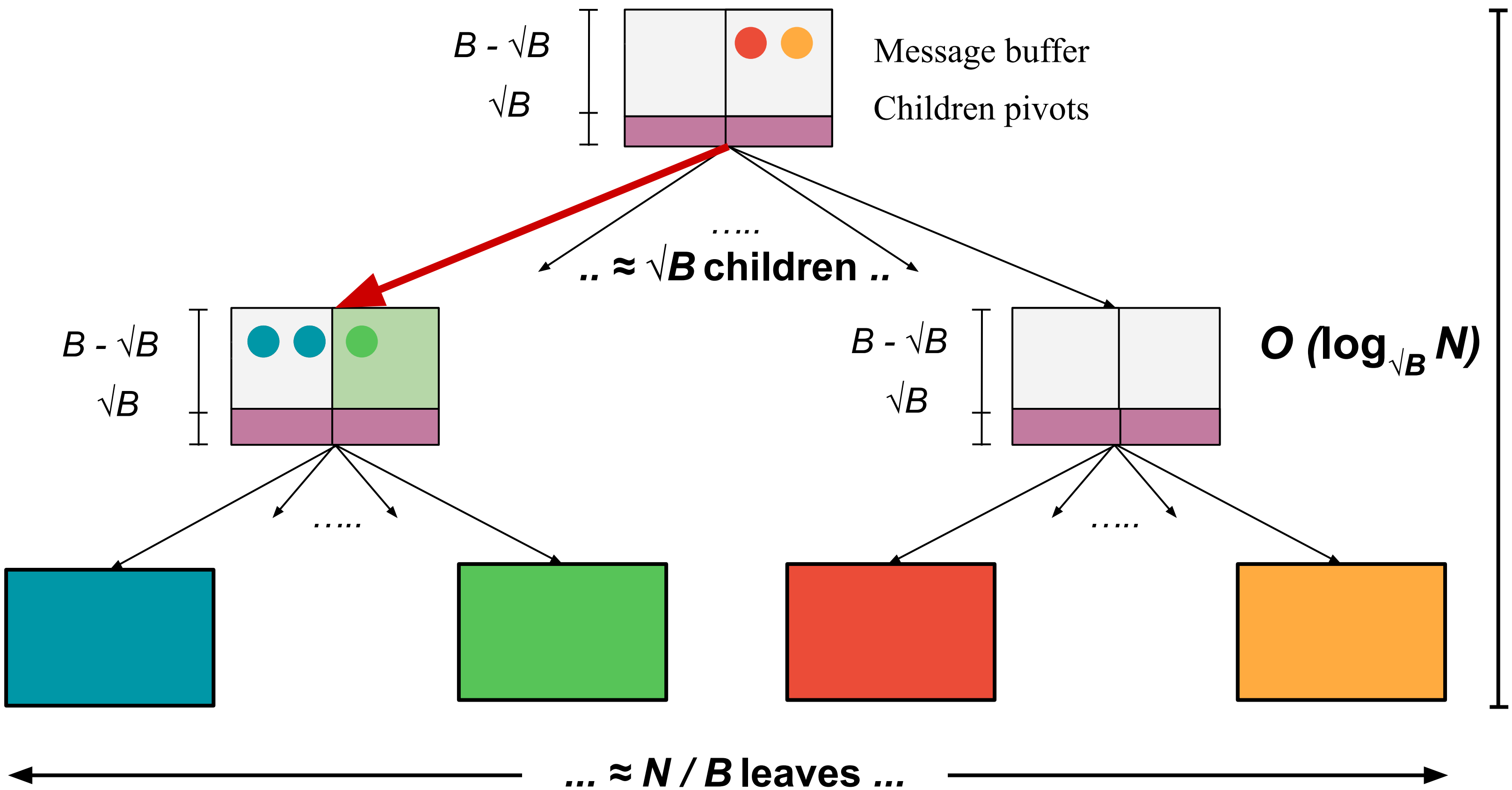
The cost to read all these elements is $1 + \alpha\sqrt{B}$.

Store child's pivot in parent → 1 I/O per node



Keeping pivots of a node in its parent avoids the extra I/O.

Store child's pivot in parent → 1 I/O per node

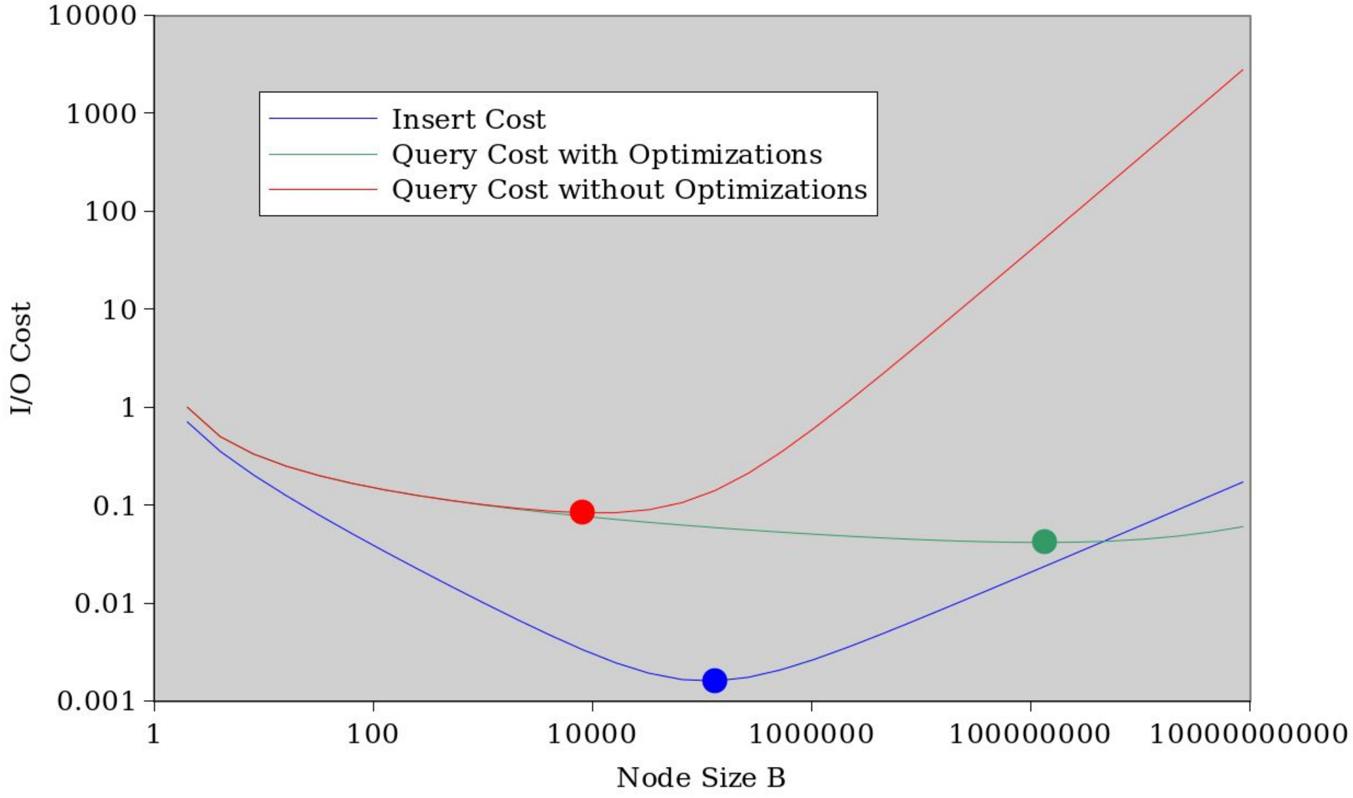


If fanout is $\approx \sqrt{B}$ the node size increases by at most constant factor.

Optimized query costs in $B^{1/2}$-trees

	Insert	Point query
$B^{1/2}$-tree	$O\left(\frac{1+\alpha B}{\sqrt{B}\log B}\log\frac{N}{M}\right)$	$O\left(\frac{1+\alpha\sqrt{B}}{\log B}\log\frac{N}{M}\right)$

Optimized query cost grows slowly



The query cost now grows more slowly with increasing node size.

Example: optimal $B^{1/2}$-tree node size in the affine model

Seagate 2TB Hard Drive

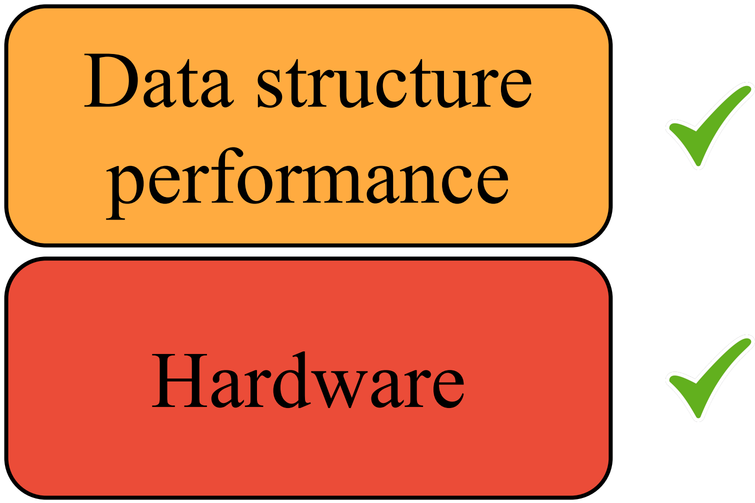
Seek time	5 msec
bandwidth	300 MB/sec
Key-value pair size	16 bytes
Optimal node size *B*	≈1.8MB

The affine model explains the discrepancy between the node sizes used in practice and those that optimize the DAM model.

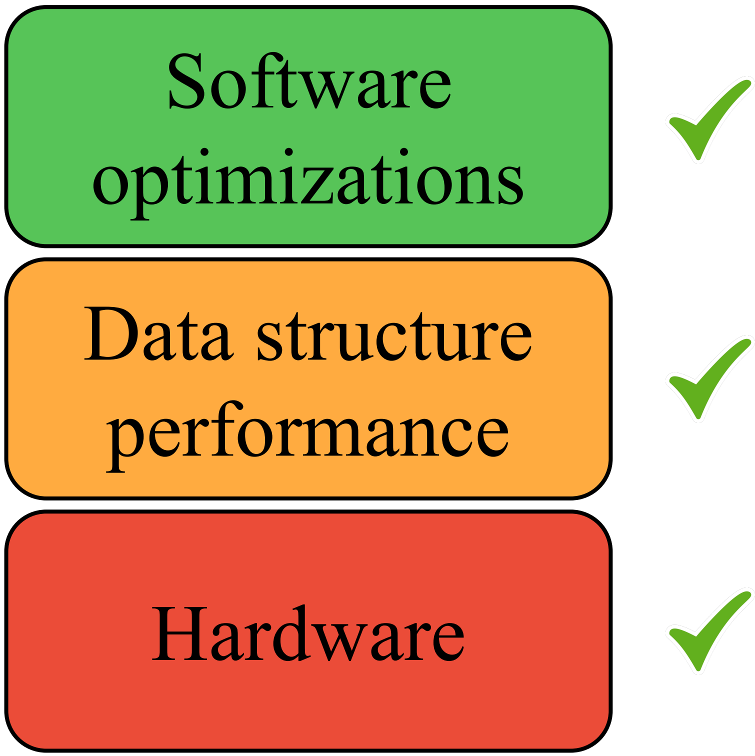
Conclusion: Design in the DAM, Refine in the Affine

Hardware ✓

Conclusion: Design in the DAM, Refine in the Affine



Conclusion: Design in the DAM, Refine in the Affine



Conclusion: Design in the DAM, Refine in the Affine

Software optimizations

Data structure performance

Hardware

New data structure ideas


Thank You