Better Metadata Management Through Data Structures

Michael A. Bender
Stony Brook & Tokutek

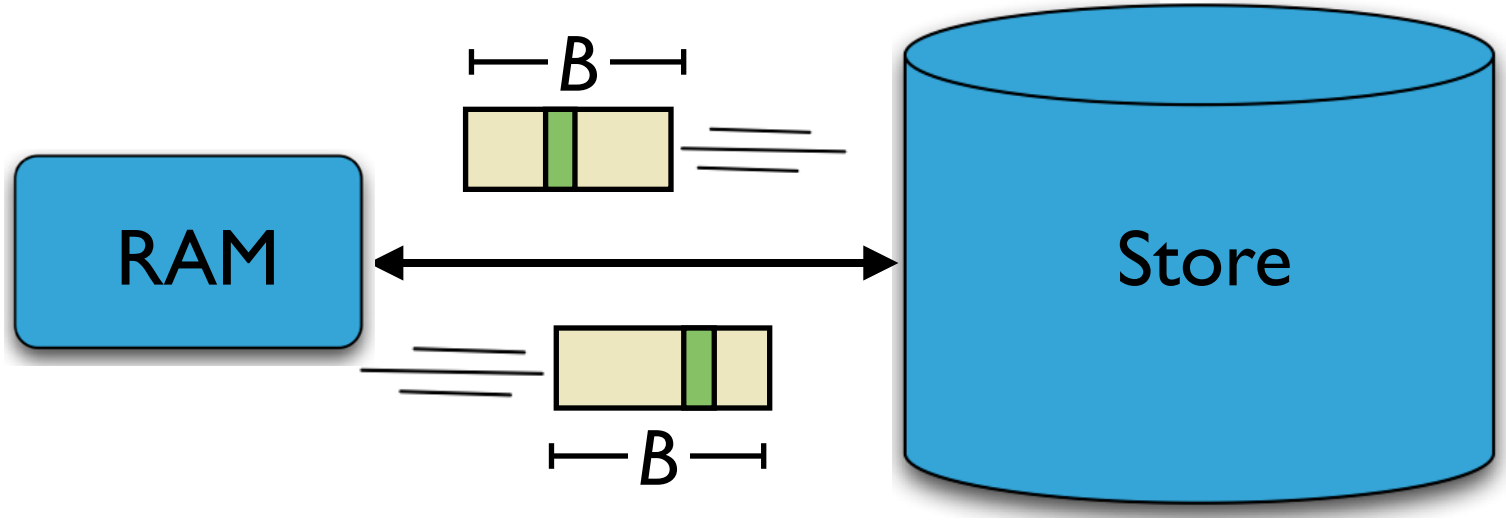
Martin Farach-Colton
Rutgers & Tokutek

Bradley C. Kuszmaul
MIT & Tokutek

Metadata ≈ Microdata

Microdata requires careful use of disks.

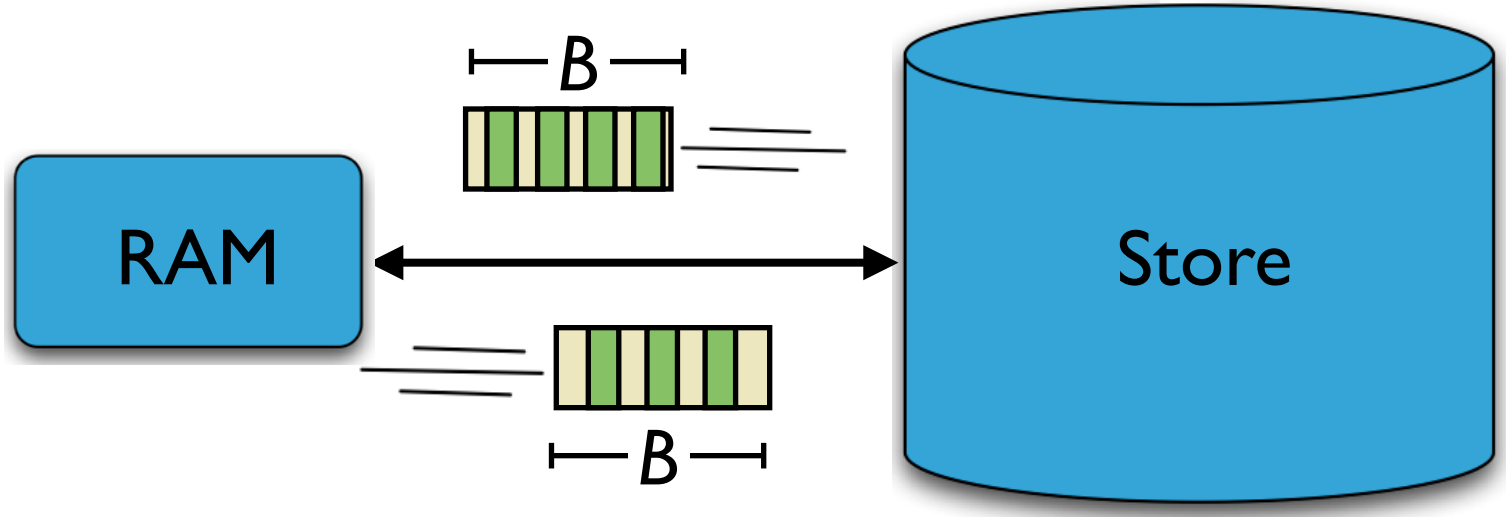
- A file system inserts/queries/deletes tiny pieces of data into its indexes (sorted data structures).
- Challenge: find ways to transfer tiny pieces via a mechanism that relies on big blocks.



Metadata ≈ Microdata

Microdata requires careful use of disks.

- A file system inserts/queries/deletes tiny pieces of data into its indexes (sorted data structures).
- Challenge: find ways to transfer tiny pieces via a mechanism that relies on big blocks.



Gain performance by transferring lots of data per I/O.

Why Microdata Management is Increasingly Important

Example: Time to fill a disk in 1973, 2010, 2018.

- log data sequentially versus index data in B-tree.

Year	Size	Bandwidth	Access Time	Time to log data on disk	Time to fill disk using a B-tree (row size 1K)
1973	35MB	835KB/s	25ms	39s	975s
2010	3TB	150MB/s	10ms	5.5h	347d
2018	220TB	1.05GB/s	10ms	2.4d	70y

Better data structures may be a luxury now, but they will be essential by the decade's end.

Why Microdata Isn't as Hard as We Think

Write-optimized structures mitigate some traditional I/O bottlenecks.

Data structures: [O'Neil,Cheng, Gawlick, O'Neil 96], [Buchsbaum, Goldwasser, Venkatasubramanian, Westbrook 00], [Argel 03], [Graefe 03], [Brodal, Fagerberg 03], [Bender, Farach,Fineman,Fogel, Kuszmaul, Nelson'07], [Brodal, Demaine, Fineman, Iacono, Langerman, Munro 10], [Spillane, Shetty, Zadok, Archak, Dixit 11].
Systems: BigTable, Cassandra, H-Base, TokuDB.

	B-tree	Streaming B-tree log-structured merge tree
Insert/delete	$O(\log_B N) = O\left(\frac{\log N}{\log B}\right)$	$O\left(\frac{\log N}{B}\right)$

- If B=1024, then speedup is $B/\log B \approx 100$.
- Hardware trends mean bigger B, bigger speedup.

Why Microdata Isn't as Hard as We Think

Write-optimized structures mitigate some traditional I/O bottlenecks.

Data structures: [O'Neil,Cheng, Gawlick, O'Neil 96], [Buchsbaum, Goldwasser, Venkatasubramanian, Westbrook 00], [Argel 03], [Graefe 03], [Brodal, Fagerberg 03], [Bender, Farach,Fineman,Fogel, Kuszmaul, Nelson'07], [Brodal, Demaine, Fineman, Iacono, Langerman, Munro 10], [Spillane, Shetty, Zadok, Archak, Dixit 11].
Systems: BigTable, Cassandra, H-Base, TokuDB.

	B-tree	Streaming B-tree log-structured merge tree
Insert/delete	$O(\log_B N)=O\left(\frac{\log N}{\log B}\right)$	$O\left(\frac{\log N}{B}\right)$

- If B=1024, then speedup is $B/\log B \approx 100$.
- Hardware trends mean bigger B, bigger speedup.

We need data structures that scale with disk bandwidth. That's what Tokutek is commercializing.

Insert/Point Query Tradeoff

	B-tree	log structured merge tree	Streaming B-tree
Insert & delete	$O(\log_B N) = O\left(\frac{\log N}{\log B}\right)$	$O\left(\frac{\log N}{B}\right)$	$O\left(\frac{\log N}{B}\right)$
		$O\left(\frac{\log_B N}{\varepsilon B^{1-\varepsilon}}\right)$	$O\left(\frac{\log_B N}{\varepsilon B^{1-\varepsilon}}\right)$
point query	$O(\log_B N)$	$O(\log^2 N)$	$O(\log N)$
		$O\left(\frac{\log^2 N}{\varepsilon \log B}\right)$	$O(\log_B N)/\varepsilon)$

Streaming B-trees insert orders of magnitude faster than B-trees with no big loss to point queries.

Insert/Point Query Tradeoff

	B-tree	log structured merge tree	Streaming B-tree
Insert & delete	$O(\log_B N) = O\left(\frac{\log N}{\log B}\right)$	$O\left(\frac{\log N}{B}\right)$	$O\left(\frac{\log N}{B}\right)$
		$O\left(\frac{\log_B N}{\varepsilon B^{1-\varepsilon}}\right)$	$O\left(\frac{\log_B N}{\varepsilon B^{1-\varepsilon}}\right)$ $2\log_B N/\sqrt{B}$
point query	$O(\log_B N)$	$O(\log^2 N)$	$O(\log N)$
		$O\left(\frac{\log^2 N}{\varepsilon \log B}\right)$	$O(\log_B N/\varepsilon)$ $2\log_B N$

Streaming B-trees insert orders of magnitude faster than B-trees with no big loss to point queries.

Insert/Point Query Asymmetry

	B-tree	log structured merge tree	Streaming B-tree
Insert & delete	$O(\log_B N) = O\left(\frac{\log N}{\log B}\right)$	$O\left(\frac{\log N}{B}\right)$	$O\left(\frac{\log N}{B}\right)$
		$O\left(\frac{\log_B N}{\varepsilon B^{1-\varepsilon}}\right)$	$O\left(\frac{\log_B N}{\varepsilon B^{1-\varepsilon}}\right)$ $2\log_B N/\sqrt{B}$
point query	$O(\log_B N)$	$O(\log^2 N)$	$O(\log N)$
		$O\left(\frac{\log^2 N}{\varepsilon \log B}\right)$	$O(\log_B N/\varepsilon)$ $2\log_B N$

Insert/point query asymmetry

- Indexing can be fast (>100K high-entropy writes/sec/disk).
- Point queries are necessarily slow (<200 high-entropy reads/sec/disk).

Fancy Indexing Helps Microdata Management

Time to fill a disk in 1973, 2010, 2018.

- log data sequentially, index data in B-tree, index in streaming B-tree.

Year	Size	Bandwidth	Access Time	Time to log data on disk	Time to fill disk using a B-tree (row size 1K)	Time to fill using Streaming B-tree* (row size 1K)
1973	35MB	835KB/s	25ms	39s	975s	200s
2010	3TB	150MB/s	10ms	5.5h	347d	36h
2018	220TB	1.05GB/s	10ms	2.4d	70y	23.3d

Streaming B-trees transform random into seq I/O.

- Support indexing performance that scales with bandwidth.

Fancy indexing structures may be a luxury now, but they will be essential by the decade's end.

* Projected times for fully multi-threaded version

Point Queries Permit Limited Improvements

Bloom filters & alternatives ("Negative cache")

- Determine whether data is present without I/O.
- **To get the data, you must perform the I/O.**

Flash

- Faster but more expensive.

Flash + Bloom filter alternatives

- "Don't thrash: cache your hash on flash"
  [Canim, Mihaila, Bhattacharhee, Lang, Ross 10]
  [Bender, Kuszmaul, Shetty, Johnson, Montes, Zadok 11]

We Can Accelerate Sophisticated Queries

The right index can make queries run fast.

- E.g., *find* or *spotlight* on a file system can be fast if the data is indexed correctly.

Streaming B-trees maintain indexes efficiently.

Fast writes/indexing is the currency that we can use to make queries go faster.

When Are Fast Inserts Not Fast?

Some inserts/deletes have hidden searches.

Ex:

- return error when a duplicate key is inserted.
- return # elements removed on a delete.
- file creation.

These "cryptosearches" throttle insertions.

Research:

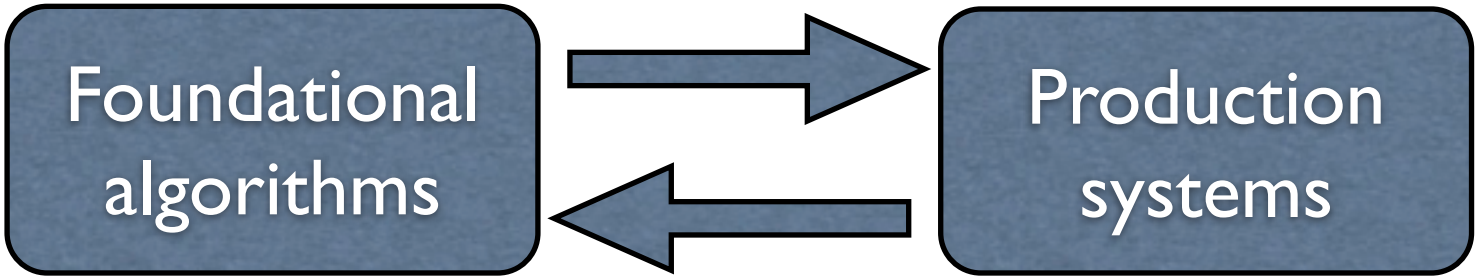
- *Make cryptosearches fast via Bloom filters & quotient filters.*
- *Change APIs to remove cryptosearches.*

Tokutek is Commercializing Streaming B-trees

TokuDB is a storage engine that delivers indexing-based query acceleration

- API: MySQL, Berkeley DB
- *Insertion Speed*: 10x-80x faster index inserts
- *Query Performance*: Fast ad-hoc and rich queries
- *Hot Schema Changes*: Hot indexing and hot column addition
- *Compression*: 5 – 15x at customer sites
- *Scalability*: Predictable performance at 10+ billion rows
- *Fast Dataset Loading*: 1+ million rows / second
- Full ACID semantics

Better Metadata Management Through Data Structures

With the right data structures, maintaining metadata becomes orders of magnitude cheaper.

Core data structures to develop (scaleup and scaleout).