

Fractal Tree[®] Indexes

Theoretical Overview
TokuDB[®] Features
Customer Use Cases

Tim Callaghan
tim@tokutek.com

About me

“Mark Callaghan’s lesser-known but nonetheless smart brother.”

[C. Monash, May 2010]

<http://www.dbms2.com/2010/05/25/voltdb-finally-launches>

About me

- Internal development (Financial Services, Oracle), 89-99
- SaaS development (Hospitality Software, Oracle), 99-09
- Field engineering (VoltDB), 09-11
- Field engineering (Tokutek), 11-now
- I've always been most interested in databases but consider myself an "IT Toolbox"
 - development, administration, management, infrastructure, testing, benchmarking, product management, support, whatever

Fractal Tree[®] Indexes

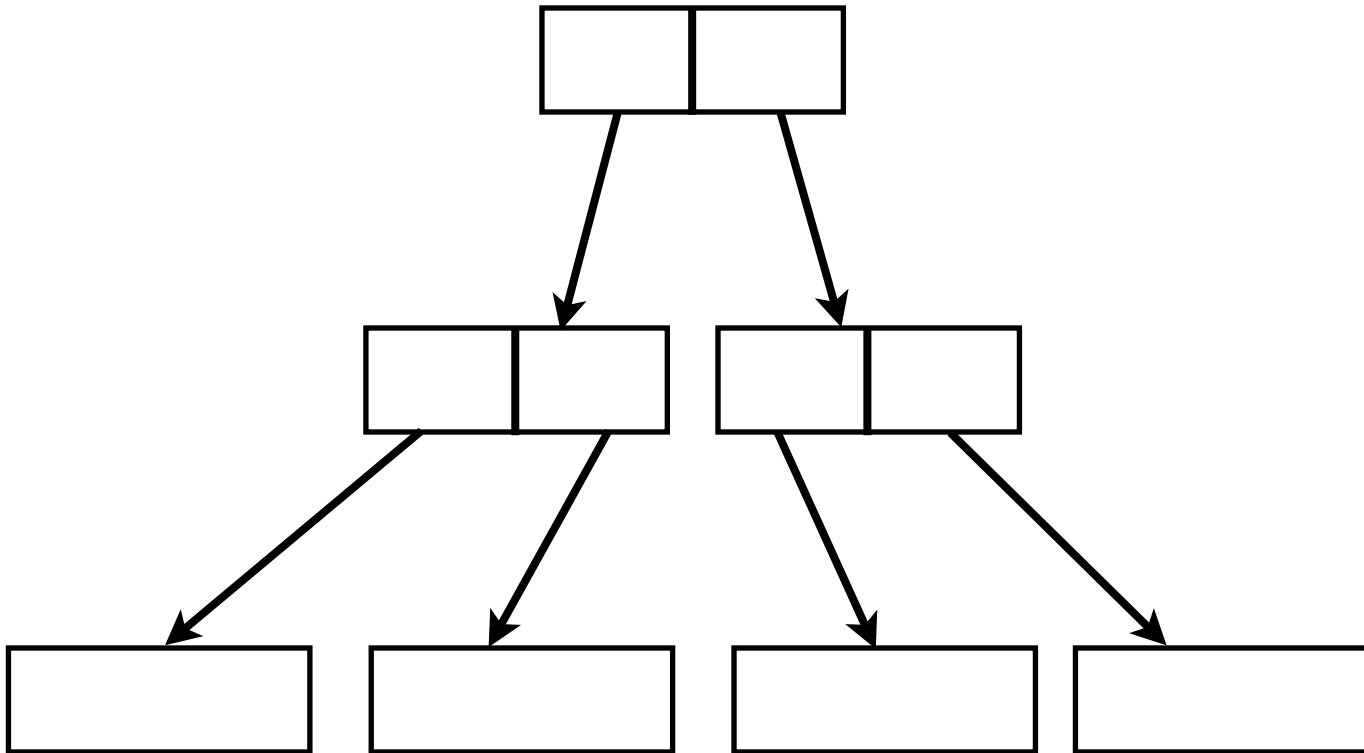
Theoretical Overview

- At a high level
 - B-trees
 - InnoDB storage
 - TokuDB® storage

I'm going to cover very simple scenarios (no splitting or merging)

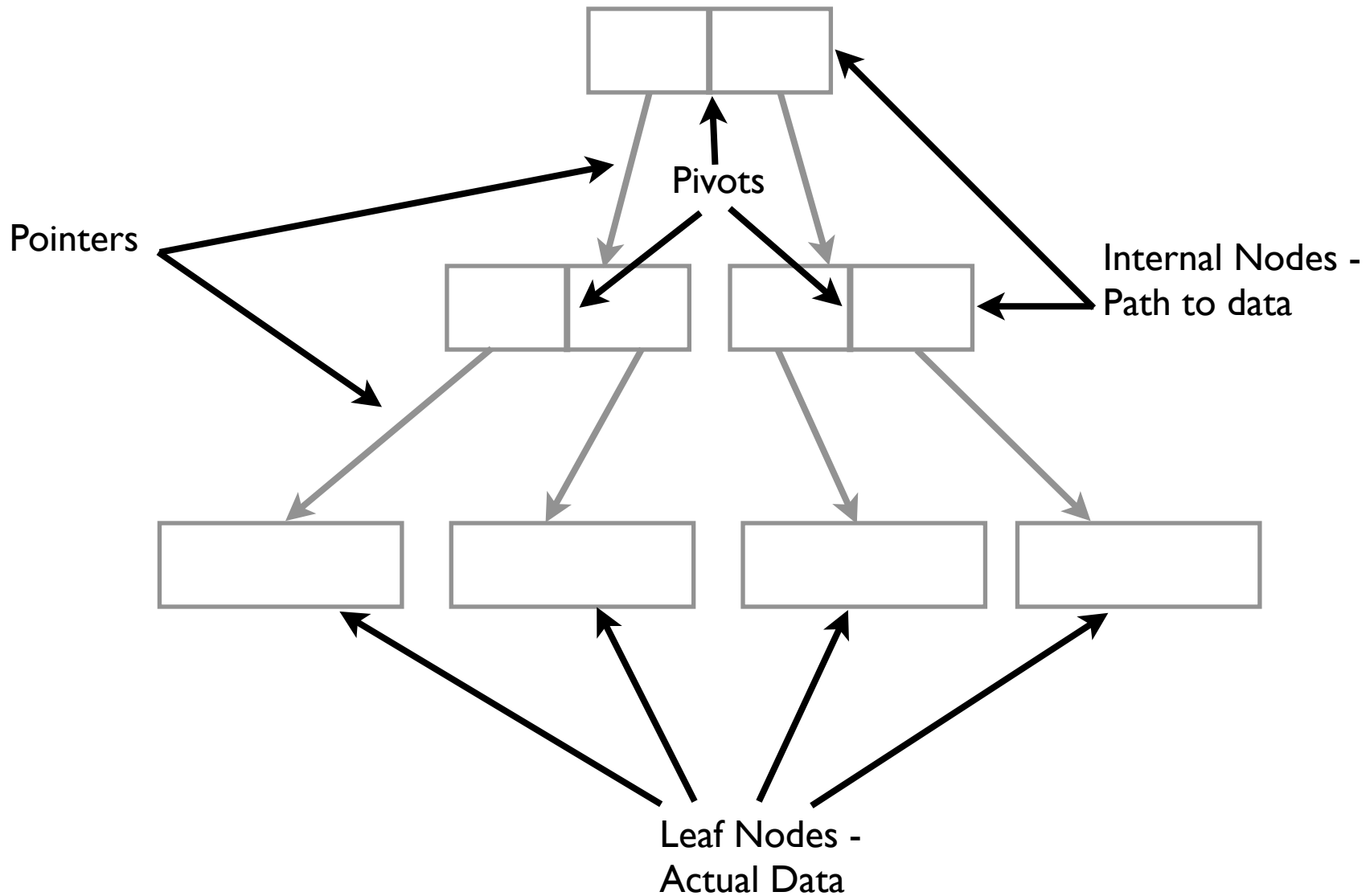
B-trees

B-tree Overview

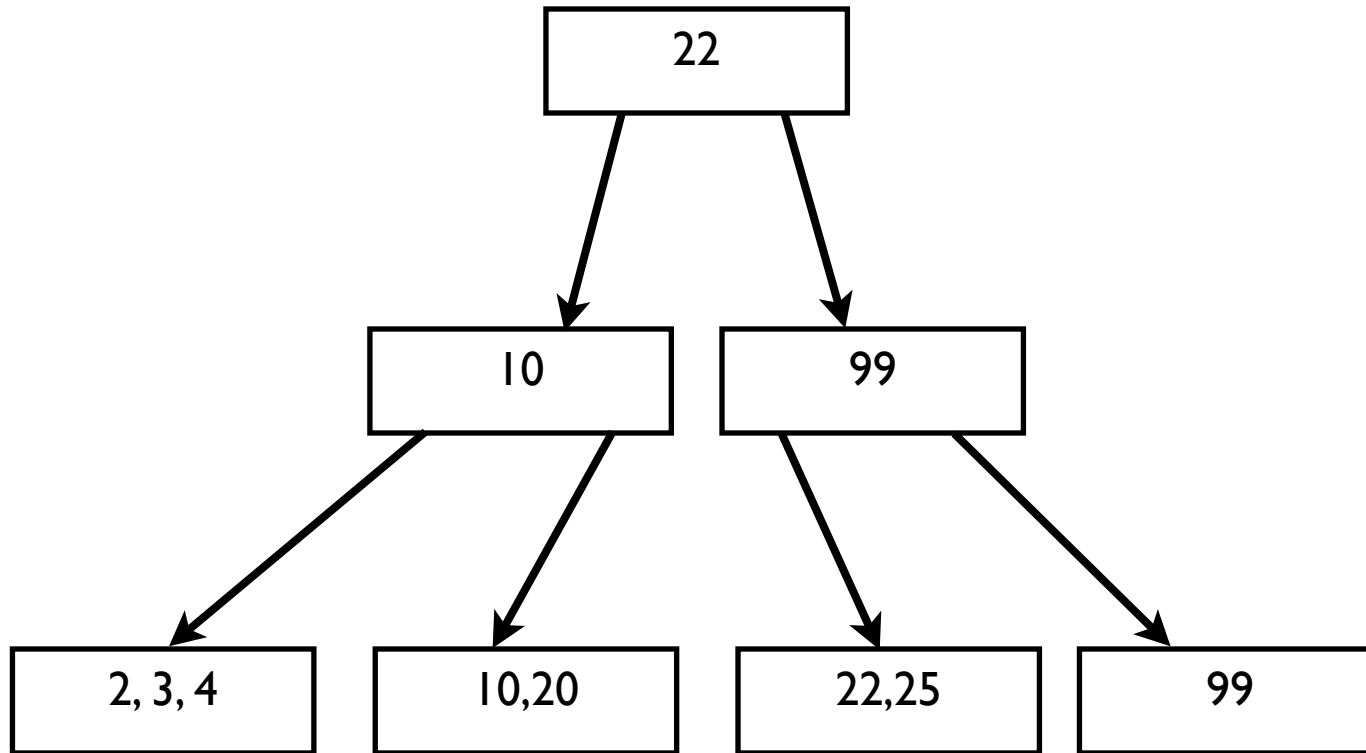


I will use a simple single-pivot example throughout this presentation

B-tree Overview - vocabulary



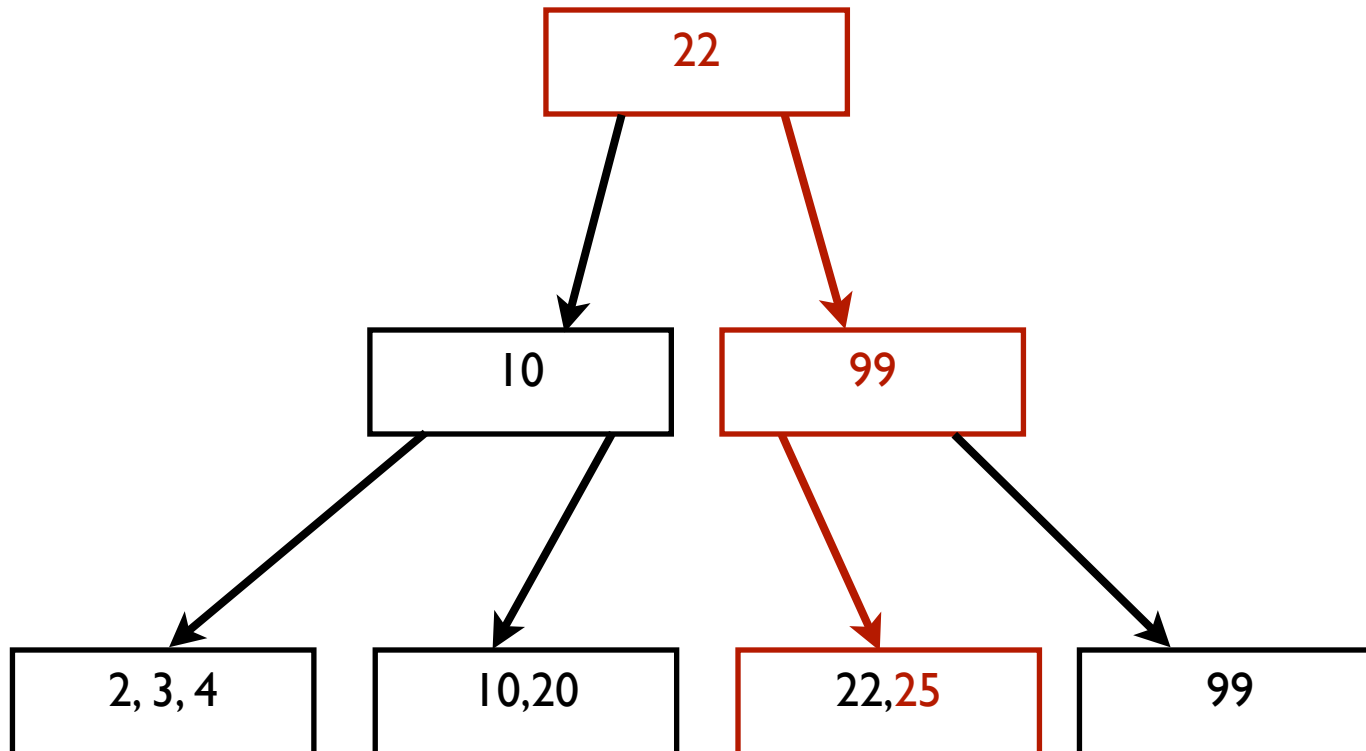
B-tree Overview - example



* Pivot Rule is \geq

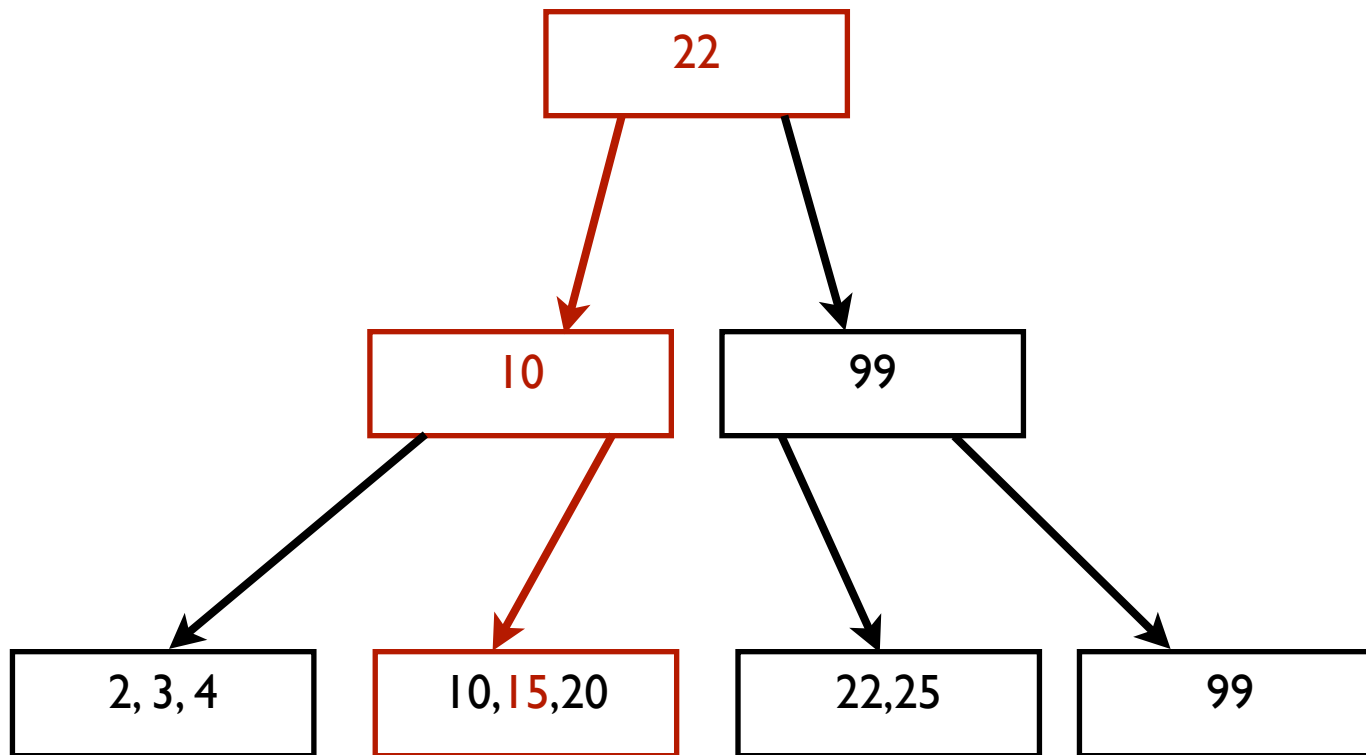
B-tree Overview - search

“Find 25”



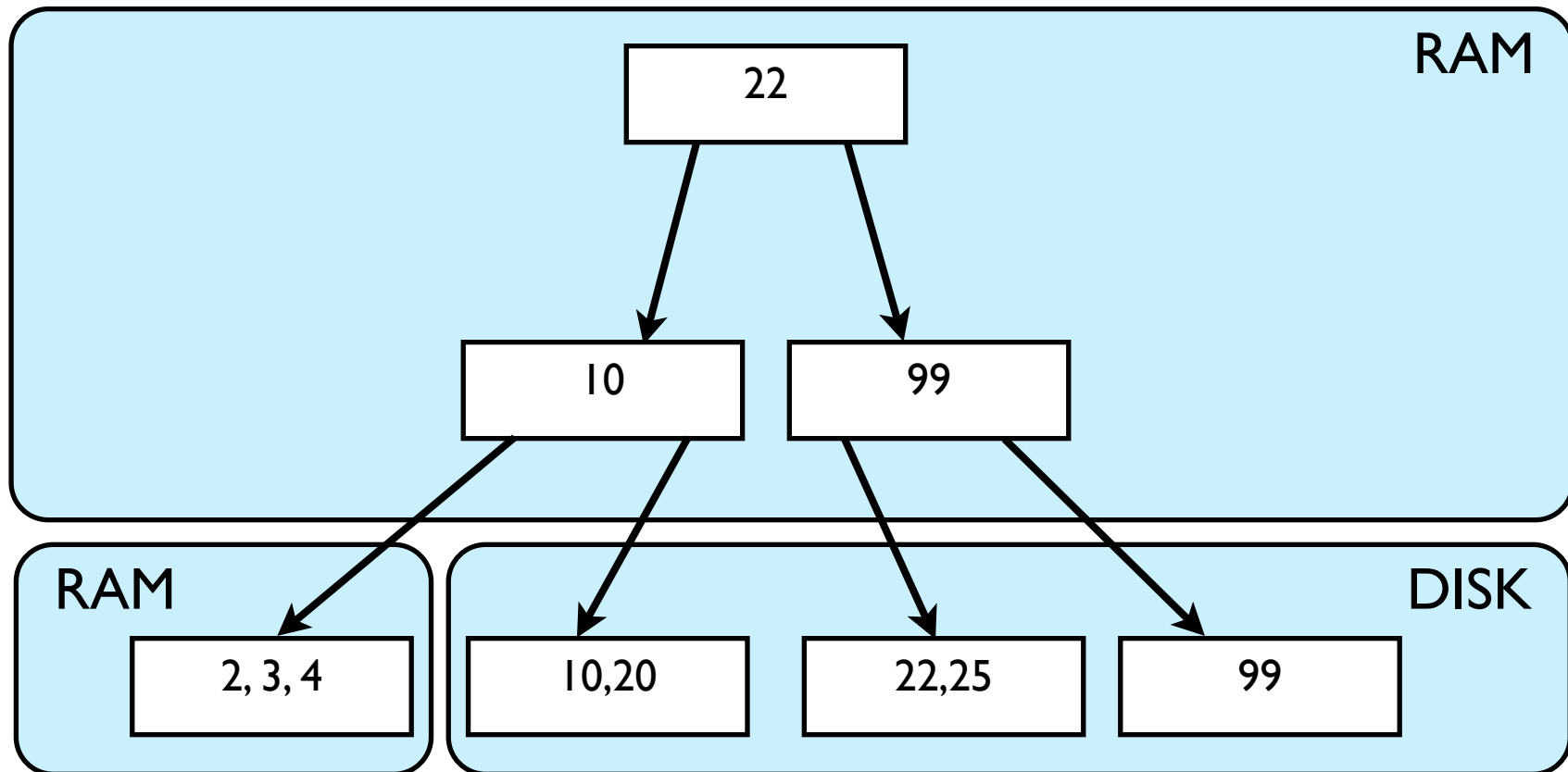
B-tree Overview - insert

“Insert 15”



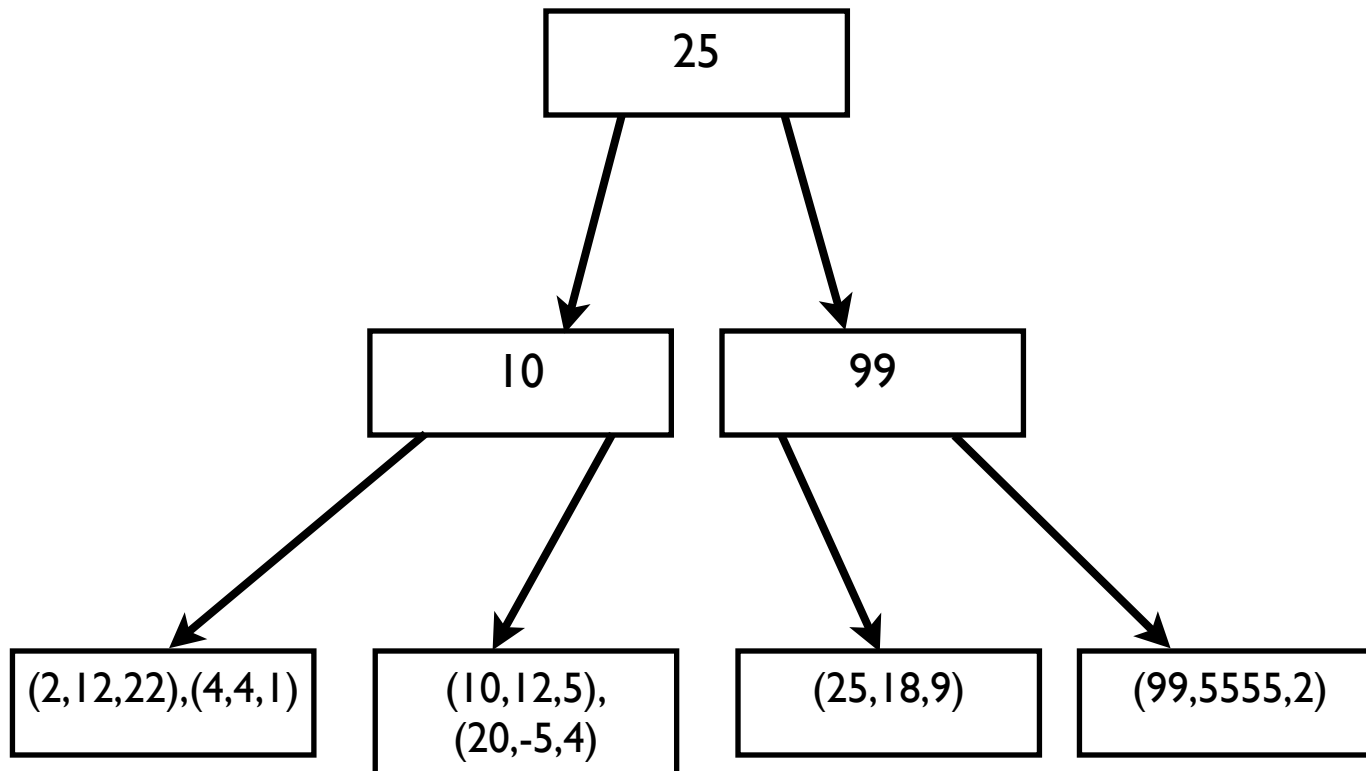
B-tree Overview - storage

Performance is IO limited when bigger than RAM:
try to fit all internal nodes and some leaf nodes



InnoDB

InnoDB B-trees - primary key

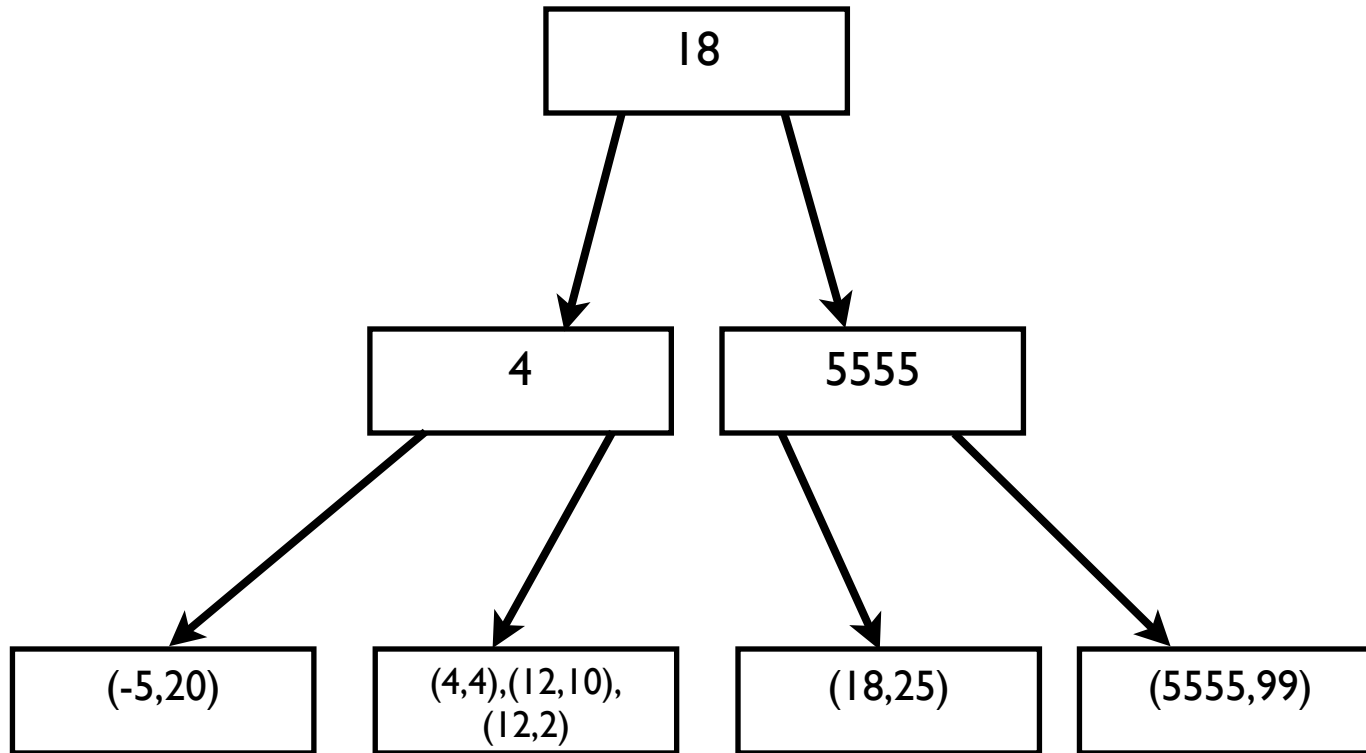


store full rows in leaf nodes

use primary key for ordering

example: table t (a int primary key, b int, c int)

InnoDB B-trees - secondary keys

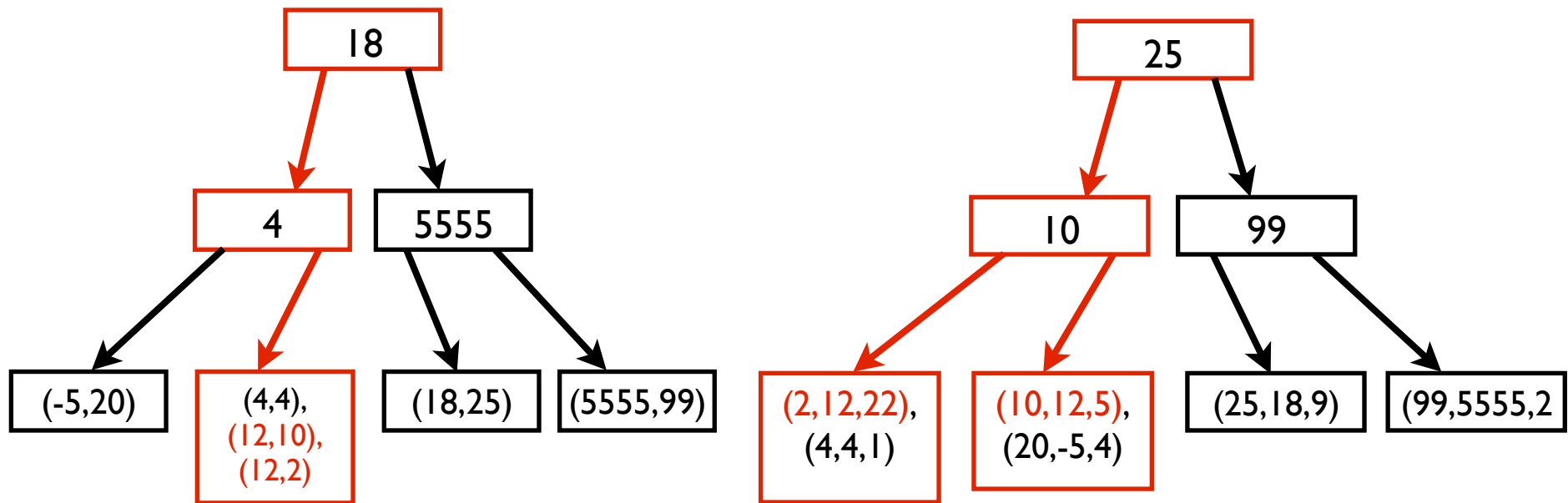


each secondary key creates another B-tree
use secondary key for ordering
PK is stored as well

InnoDB B-trees - search

select * from t where b=12;

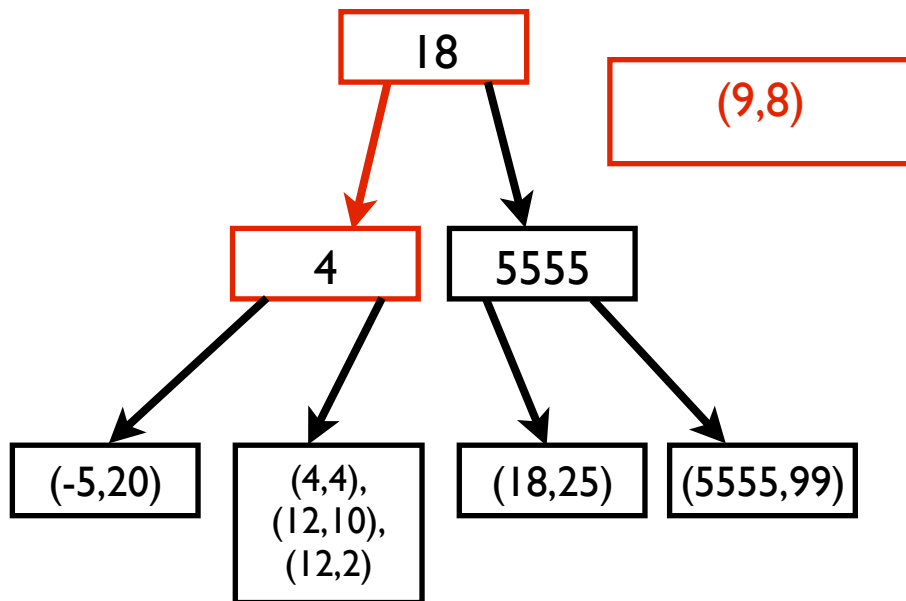
(a) secondary index - find rows for b=12 (get PKs) (b) PK index - get * for PK=10 and 2



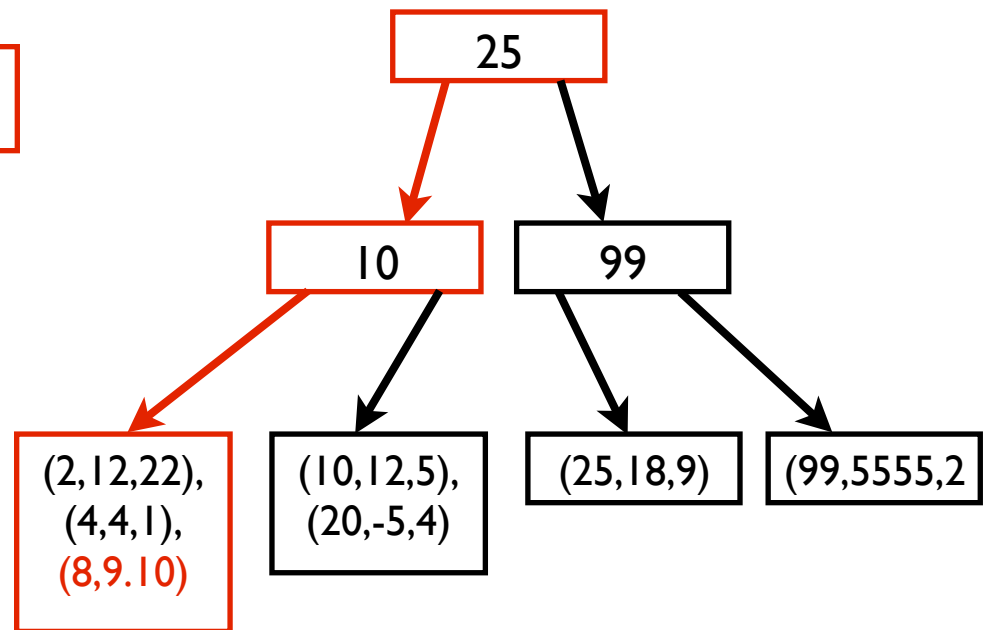
InnoDB B-trees - insert

insert into t values (8,9,10);

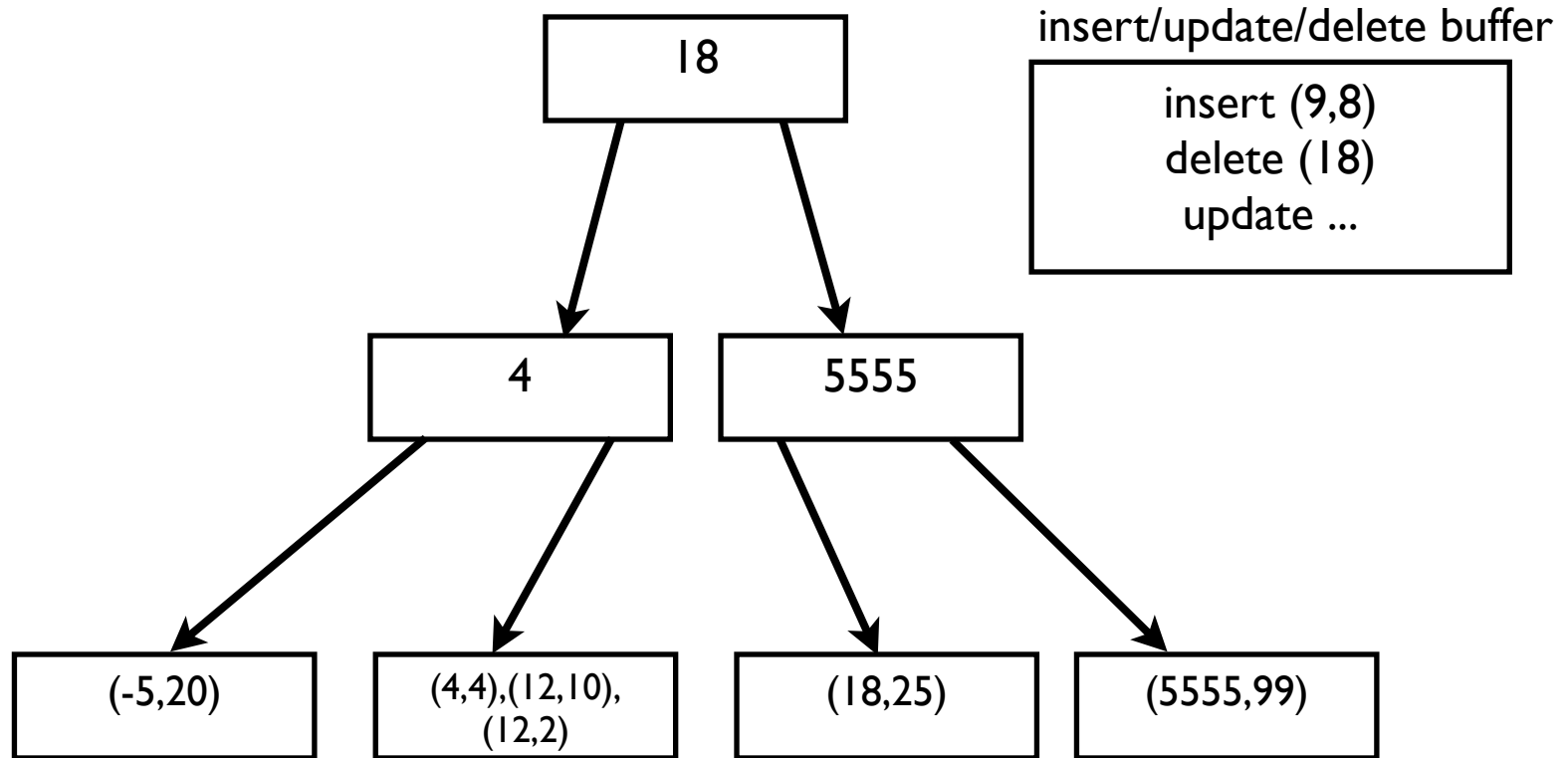
(a) insert into leaf if in memory, buffer if not



(b) insert into leaf node in PK index



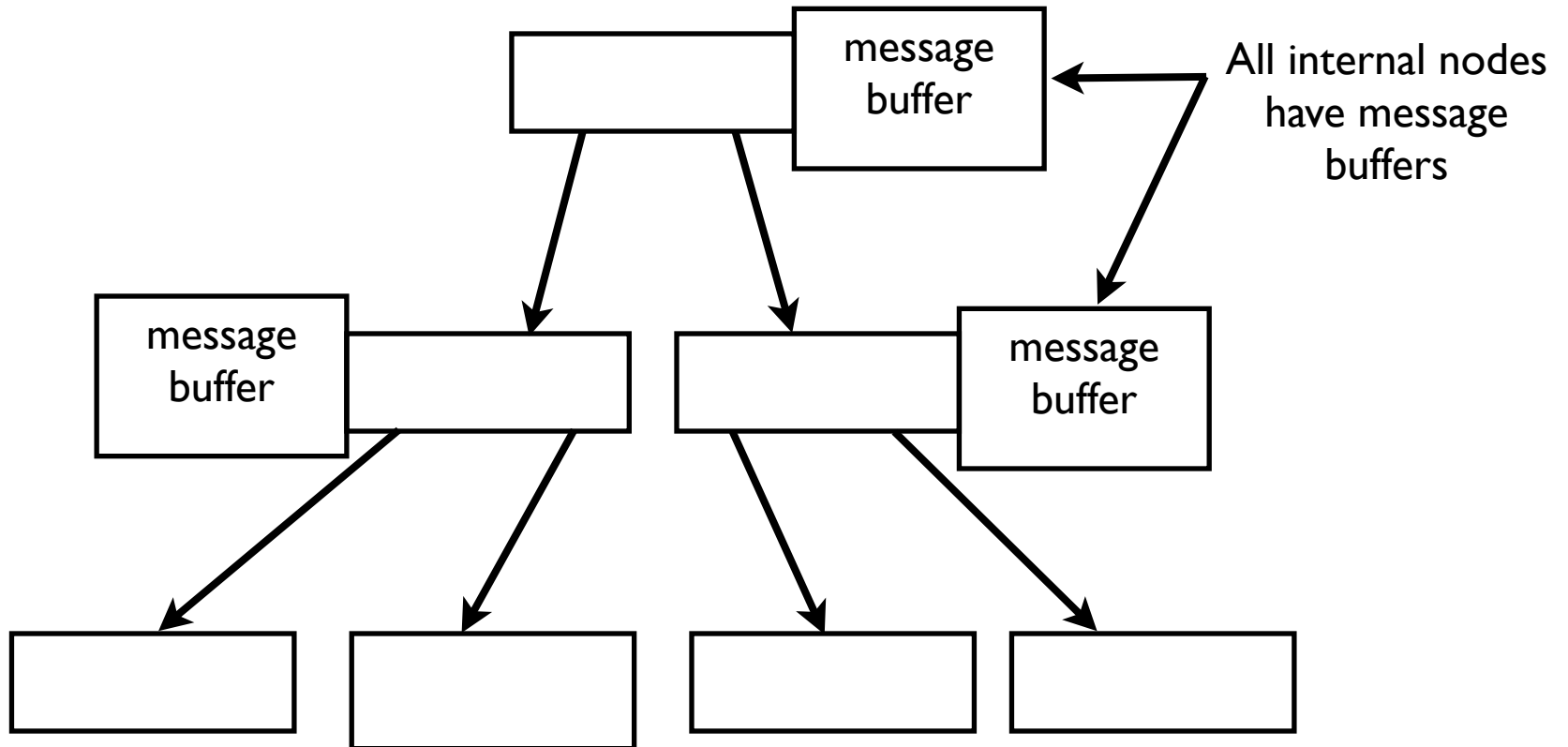
InnoDB B-trees - secondary key buffer



- Buffer inserts, deletes, and updates if the needed leaf node is not in memory
- Flushing occurs when the buffer is full or the leaf node comes into memory

Fractal Tree[®] Indexes

Fractal Tree[®] Indexes



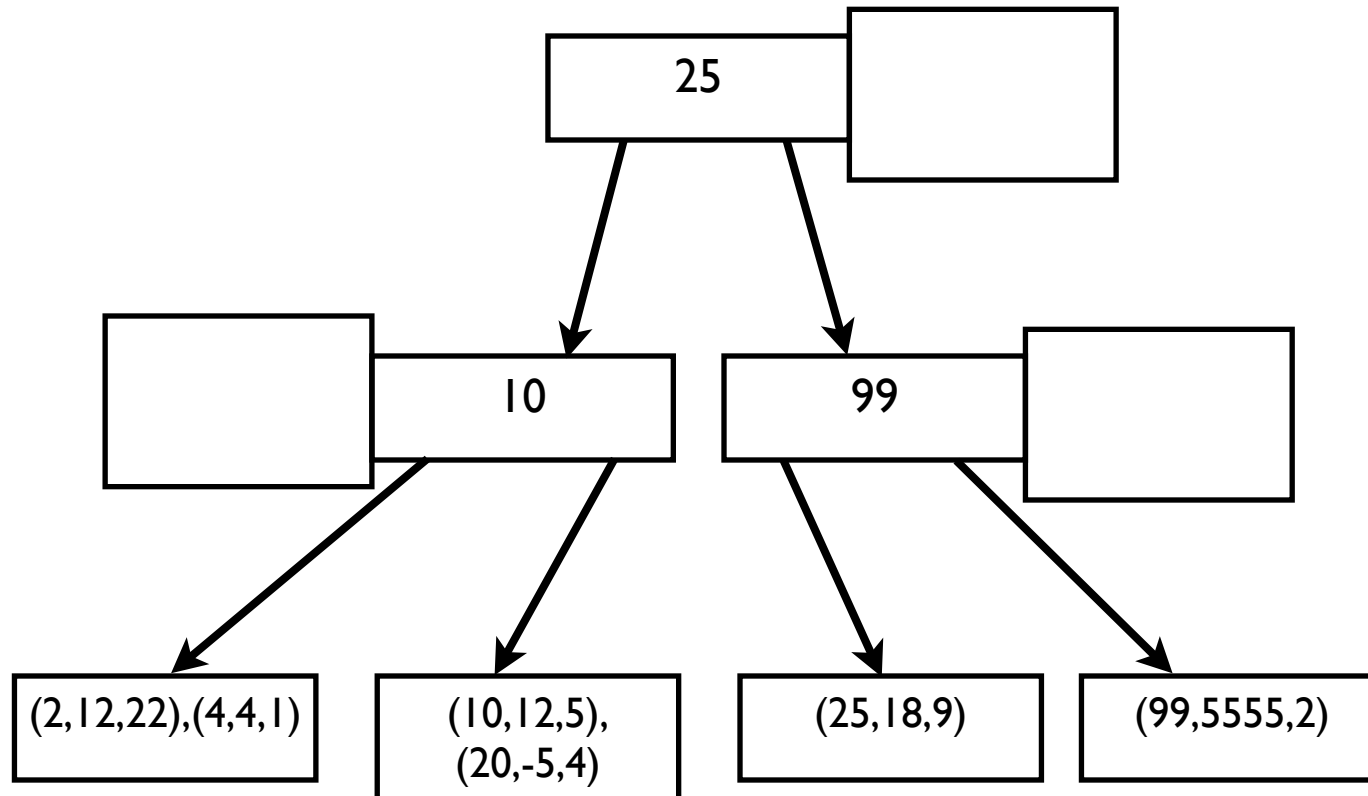
similar: store data in leaf nodes

similar: use PK for ordering

different: message buffer within all internal nodes

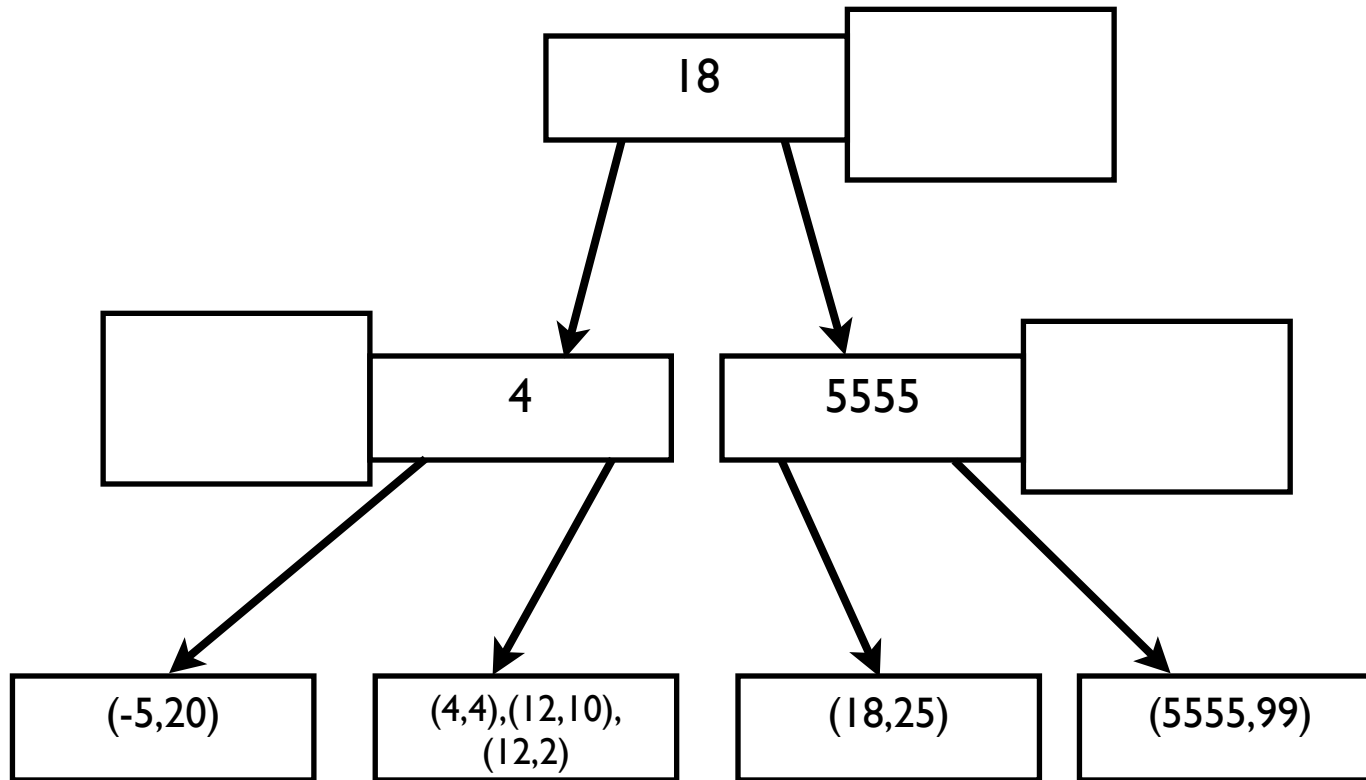
different: much larger nodes (4MB vs. 16KB)

Fractal Tree[®] Indexes - primary key



example: table t (a int primary key, b int, c int)

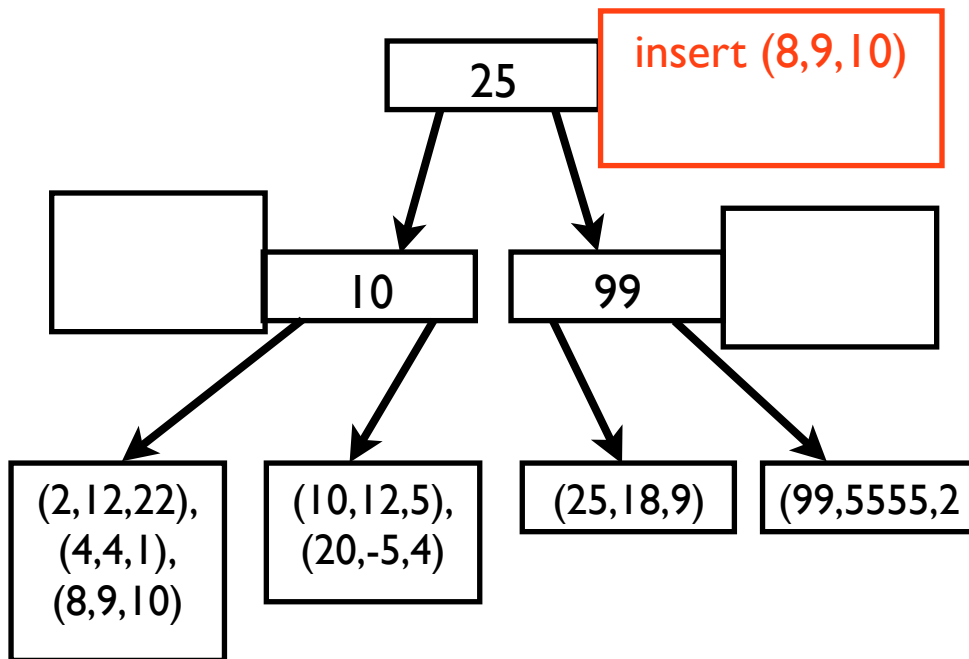
Fractal Tree[®] Indexes - secondary keys



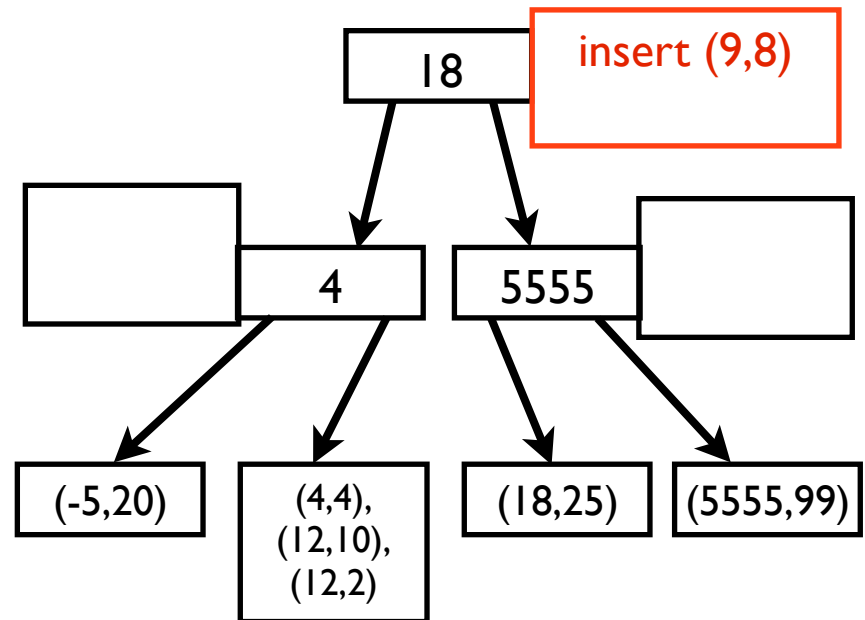
Fractal Tree[®] Indexes - insert

insert into t values (8,9,10);

(a) insert into leaf node in PK index

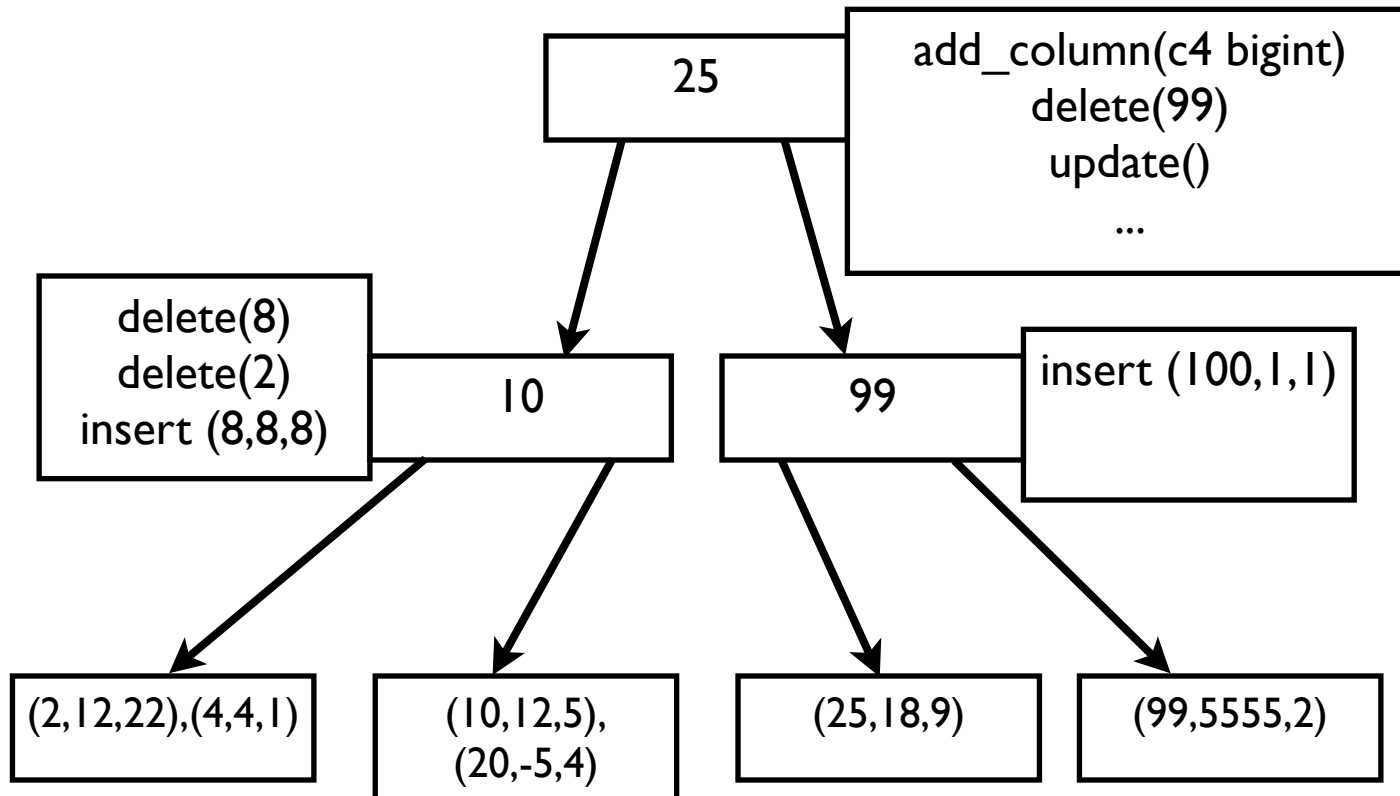


(b) insert into leaf if in memory, buffer if not



- messages cascade down the tree as buffers fill up
- they are eventually applied to the leaf nodes

Fractal Tree[®] Indexes - other operations



Lots of operations can be messages!

TokuDB[®] Features

What is TokuDB®?

- MySQL Storage Engine
- Available for MySQL 5.1 and MariaDB 5.2
- Provides ACID and MVCC
- 64-bit Linux

A Few Myths about Big* Databases

Big* = significantly larger than RAM

Myth: Big Databases are Inflexible

- Applications running on large transactional and operational databases often require maintenance windows
 - Adding indexes
 - Adding columns
 - Defragmenting tables and indexes

Important Notice

The system will be unavailable on
Saturday, November 19
from 1:00am to 3:00am
for maintenance.

Myth: Big Databases are Slow

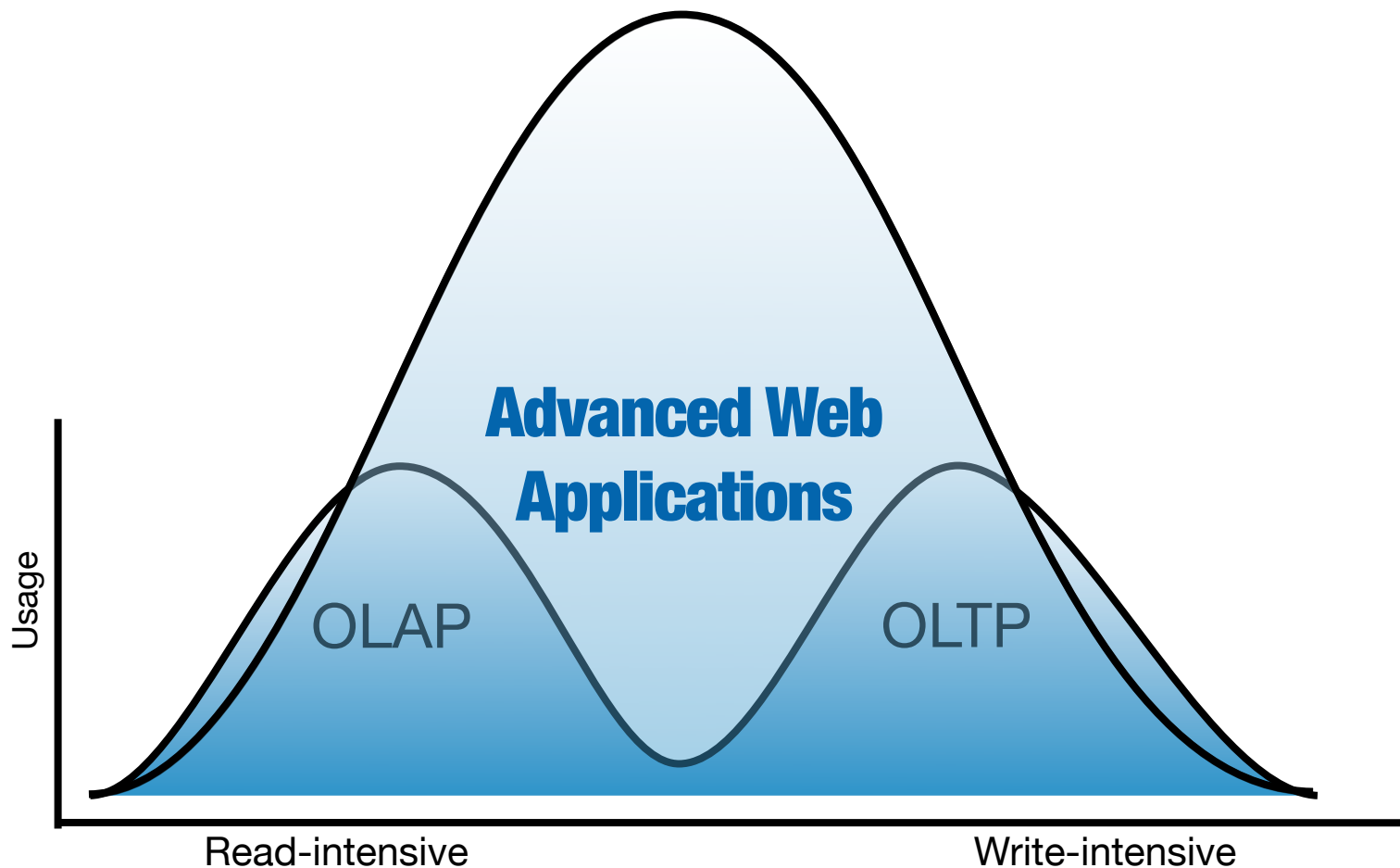
- Existing technologies provide a no-win situation

	Insert Performance	Query Performance
More Indexes	Down	Up
Less Indexes	Up	Down

An End to the Either-Or

– Matt Aslett / The 451 Group:

- “While TokuDB® is effectively an operational database technology, it does blur the lines between operations and analytics.”

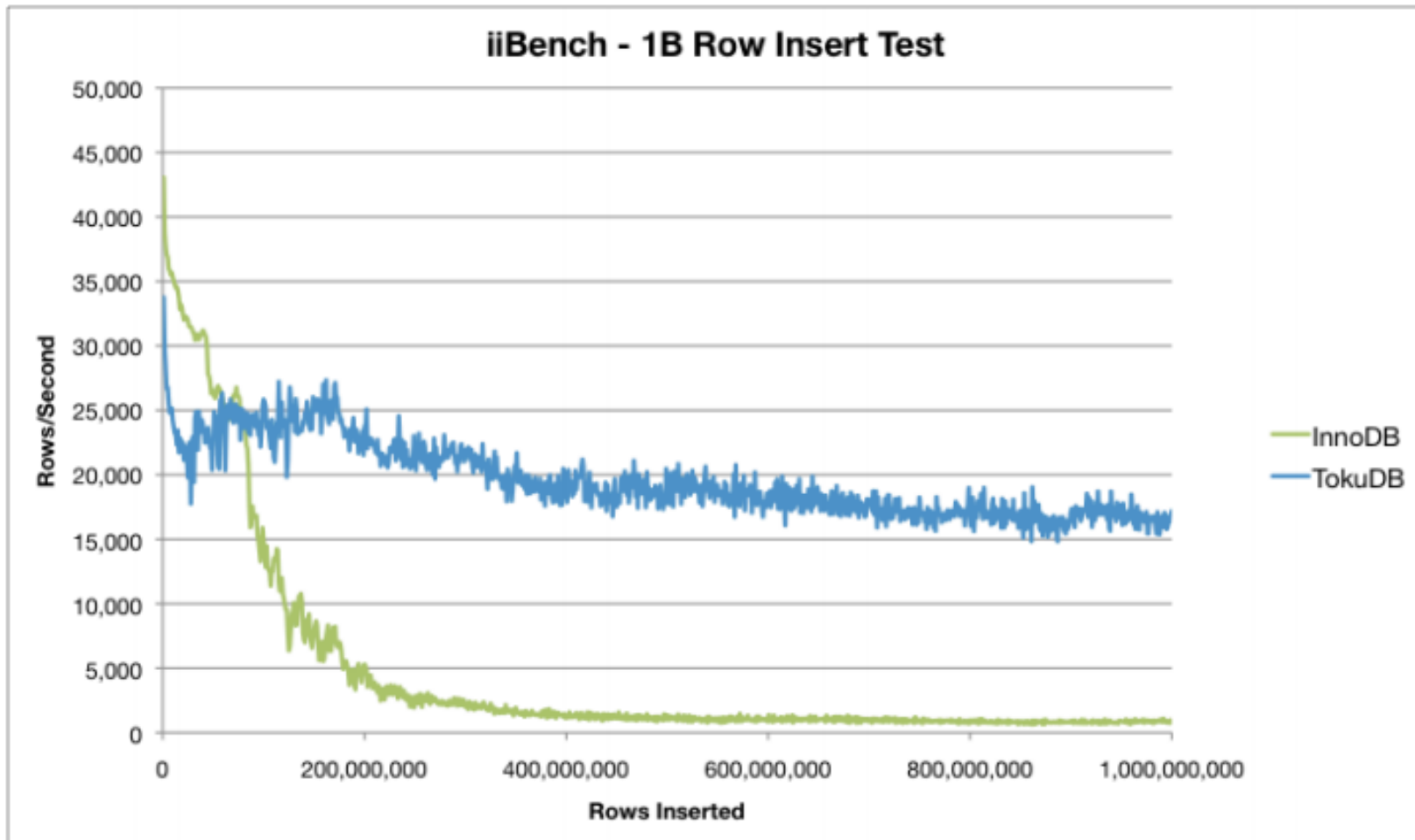


TokuDB[®] Features

- Performance
 - Storage engine capabilities
 - Multiple clustering keys
 - Data loader
- Agility
 - Online operations
- Management at Scale
 - OLTP and OLAP
 - Compression
 - No fragmentation
 - Checkpoints and recovery
 - Progress tracking

Performance: Storage engine

- High-performance insert/update/delete for large databases while maintaining indexes



Performance: Storage engine

- TokuDB[®] uses Tokutek's Fractal Tree[®] technology
 - Large block size enables high compression and excellent range query performance
 - Internal nodes are similar to B-trees (keys and pointers) but also contain message buffers

Performance: Multiple clustered keys

- Like InnoDB
 - All row data is stored by primary key (clustered)
 - Secondary keys store primary key
 - Lookups by secondary key actually require a “join” to retrieve the rest of the row
- Unlike InnoDB, TokuDB[®] allows secondary keys to be clustered
 - Compression saves space, indexes are small
 - All columns of the table are immediately available on secondary key lookup

Performance: Data loader

- Traditional MySQL loader is single threaded and loads data via MySQL.
- TokuDB[®] loader creates Fractal Tree[®] Indexes directly and uses all available cores
- Index creation uses same technology
- No more “create table...”, load data, “create index1”, “create index 2”
 - Building secondary indexes after data is loaded to improves load performance
 - In TokuDB, secondary indexes are built after all data is is loaded

Management: Online operations

- Common schema changes can take hours in MySQL
 - Adding or dropping a column
 - Adding an index
 - More on the way...
- Also, the table is unavailable to all other operations during the process
- As a workaround, people generally
 - Create the new index on a replication slave
 - Once complete, allow it to catch up to the master
 - then promote the slave to master (swap)
- Many are considering NoSQL (schema-less) technologies to overcome these limitations

Management: Hot column addition/deletion

- “alter table t1 add column c4 bigint;”
- InnoDB
 - Locks the table and performs a “select into ...” to the new table structure
 - Indexes get rebuilt as well
 - No access to the table allowed during the process
- TokuDB[®]
 - Creates an addcolumn() message and returns
 - Over time, the column is physically added to the actual rows

Management: Hot column addition/deletion

- 122mm row "air traffic data" table, add column
 - InnoDB 5.1 plugin took 17 hours, 44 minutes
 - table was locked the entire time
 - TokuDB[®] 5.0 took 3 seconds
- details at <http://goo.gl/sEzx1>

Management: Hot indexing

- `“create index c4_idx on t1(c4);”`
- Traditional MySQL
 - Locks the table and creates the index
 - No access to the table allowed during the process
- TokuDB®
 - Begins creating the index in the background
 - Uses TokuDB parallelized loader technology
 - Index is available to MySQL when finished
 - Accurate progress via `“show processlist;”`

Management: Hot indexing

- 122mm row "air traffic data" table, create new index
 - InnoDB 5.1 plugin took 31 minutes, 34 seconds
 - table was locked the entire time
 - TokuDB[®] 5.0 took 9 minutes, 30 seconds
 - table was locked for under 2 seconds
- details at <http://goo.gl/ffW4E>

Management: OLTP and OLAP

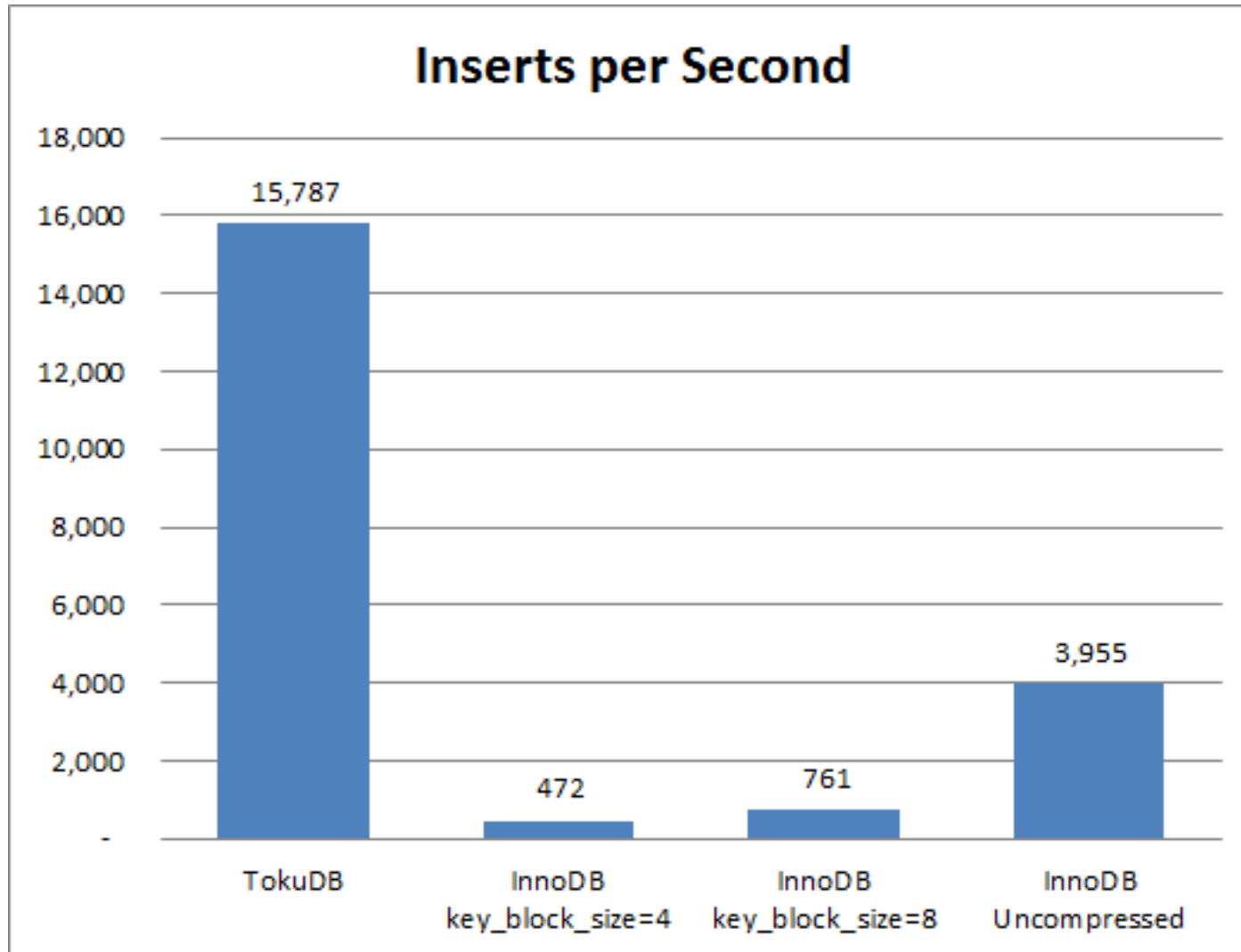
- “Hybrid” data implementations are becoming the norm
 - 1 database for OLTP, another for OLAP
- Usually because performance degrades as the database gets large
- Running a single TokuDB[®] database means more timely analytical information and less moving parts

Management: Compression

- TokuDB[®] has a larger block size than InnoDB which often leads to higher compression ratios
- 5x to 15x can be achieved
- TokuDB[®] compression is always enabled, no knobs, no tuning, no performance penalties

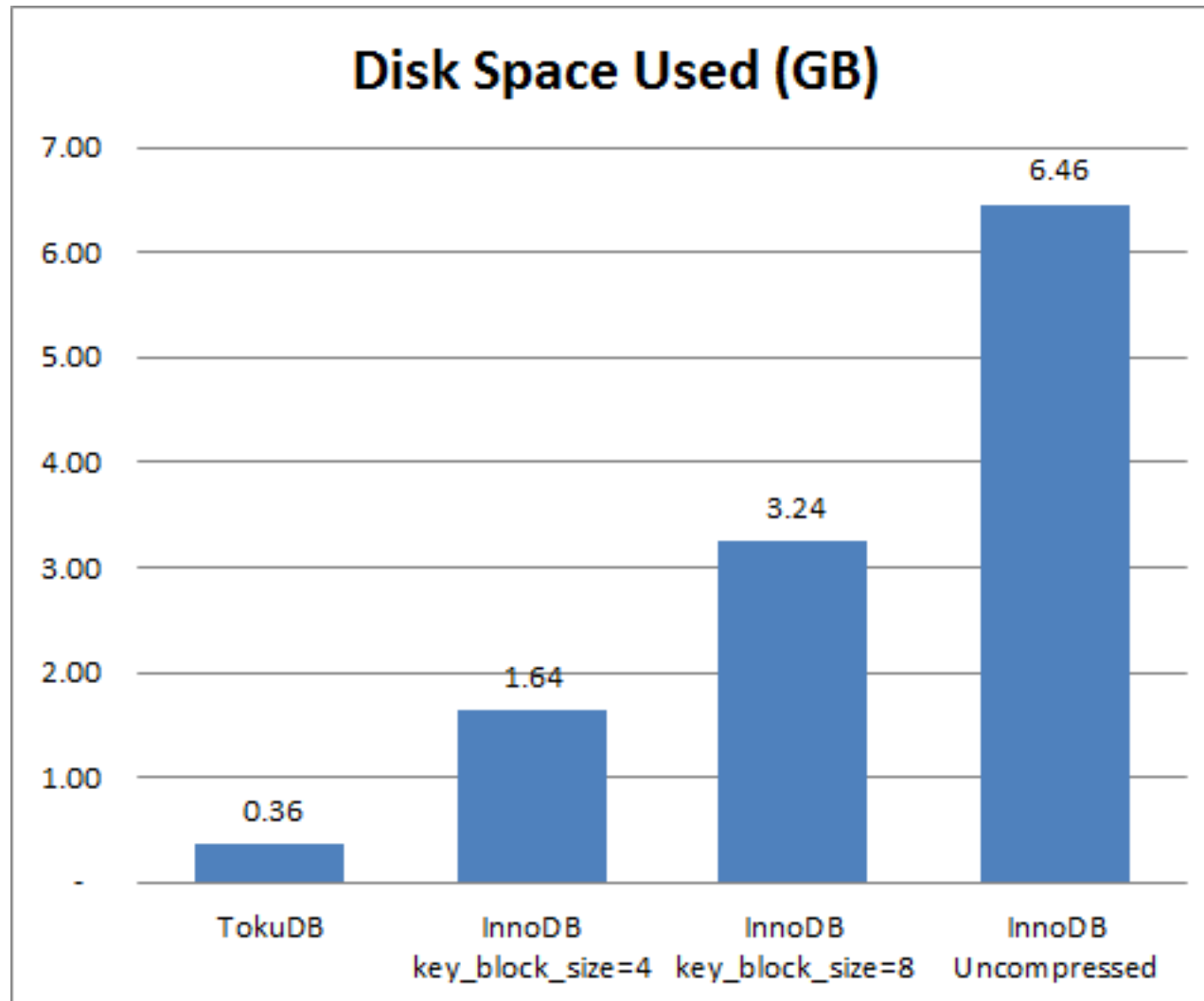
Management: Compression

- Compression performance penalties, iiBench



Management: Compression

- Compression space savings, log data



Management: Fragmentation

- Over time, B-trees with small block sizes suffer from fragmentation
- Best practices are dump/reload or “optimize table”
- Requires maintenance windows during which table is unavailable
- Fractal Tree[®] Indexes do not fragment
 - 4MB compressed blocks vs. 16KB uncompressed
 - Far less random IO for range queries

Management: Checkpoints and recovery

- TokuDB[®] performs a checkpoint every 60 seconds
 - Cost of checkpoint work is reduced
 - Also enables fast recovery time (less than 1 minute)
 - 60 second time period is user definable
- Be careful, I've run benchmarks on large servers (> 48GB) with other storage engines that show long periods of inactivity when checkpointing
 - see Vadim Tkachenko's blog at <http://www.mysqlperformanceblog.com/2011/09/18/disaster-mysql-5-5-flushing/>

Management: Progress tracking

- “show processlist”
- long running processes show accurate % complete
 - Data loader
 - Index creation

Deployment and Other Details

- 64-bit Linux only
 - Developed and tested on CentOS 5
 - Other Linux distros in production
- Physical hardware or virtualized
 - Well suited for Amazon EC2/EBS
- Windows/Mac development via Virtual Machines
- MySQL 5.1 and MariaDB 5.2
- Commercial usage free for data < 50GB
- Also free for academic, research, evaluation

TokuDB[®] Customer Use-Cases

Use-case #1 - The Problem

- Want to perform ad-hoc analytics on clickstream data
 - Daily “canned” reports helpful
 - Data loaded daily into HDFS
 - Reports generated using Hadoop Map/Reduce jobs
- Needs
 - Ingest constant flow of data while maintaining indexes
 - Ad-hoc reporting capabilities

Use-case #1 - The Solution

- Why TokuDB[®]?
 - High insertion rates, billions of rows
 - InnoDB was not sufficient
 - Query performance
 - Didn't remove indexes to maintain insert performance
 - SQL Interface
 - Business analysts already know SQL, don't want to learn Map/Reduce or NoSQL access methods
 - Schema flexibility
 - Adding indexes and columns while still accessing table data
 - Bonus: Compression
 - Clickstream data is highly compressible

Customer feedback - Intent Media:

"Column additions in the past simply were not practical, taking days to complete. They now take a matter of seconds, and can be accomplished in a non-disruptive fashion. This has dramatically improved our ability to adapt to the changing needs of our business, without fear that a schema change would lock up a table for a week or more, blocking other time-sensitive analyses."

Use-case #2 - The Problem

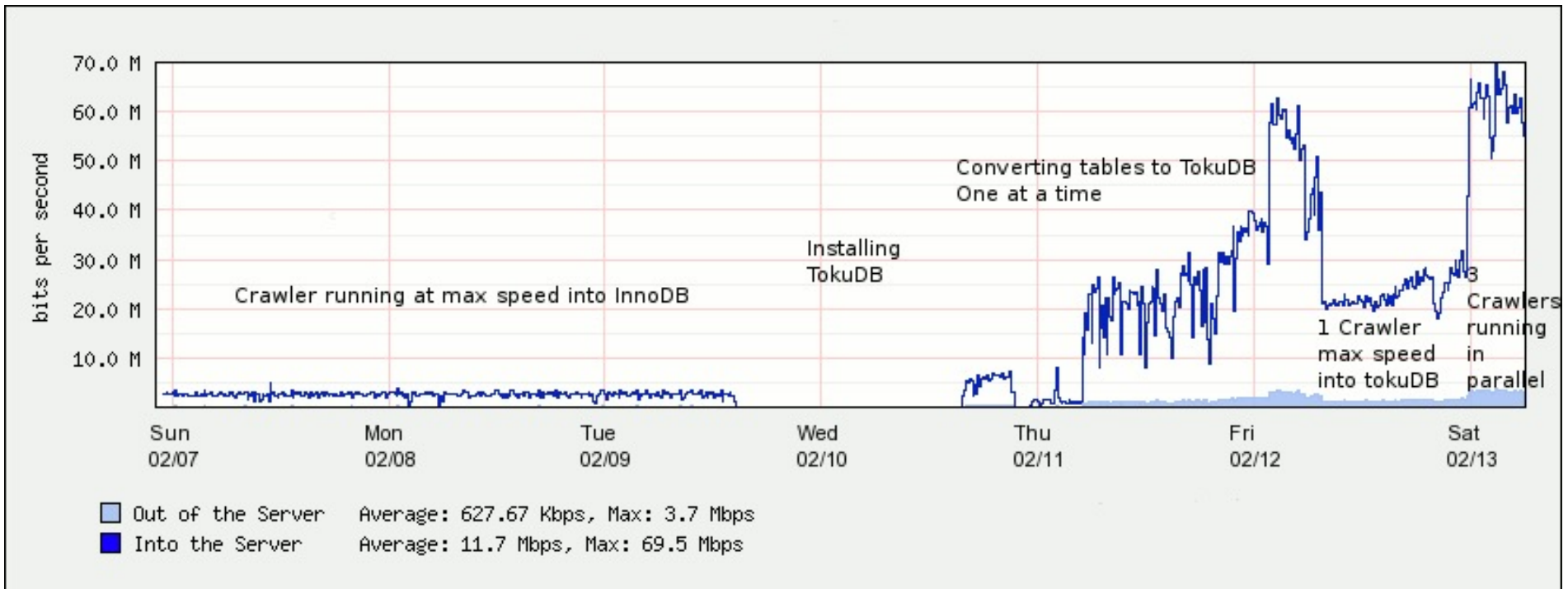
- Web crawler speed severely limited by existing database
 - Random insert performance of InnoDB insufficient
 - Estimated between 3 and 6 months to do a full crawl
 - Experienced 30 hour crash-recovery times
- Needs
 - High speed indexed inserts
 - Improved recovery time

Use-case #2 - The Solution

- Why TokuDB[®]?
 - Measured 80x improved insertion rate
 - Shifted focus to improving crawler application performance
 - Full crawls now complete in under 2 weeks
 - Improved recovery time
 - Recovery now measured in minutes
 - Bonus: Compression
 - Crawl data highly compressible
 - Saving on expensive RAID storage
 - Bonus: Schema flexibility
 - Just added an index on the largest table, 24 hours to complete, table was fully available

Use-case #2 - The Picture

- Migration from InnoDB to TokuDB®



Customer feedback - Profile Technology:

"I measured an 80x improvement on our crawler's insertion rate, boosting our overall performance by 20x, and we expect our 2 billion row database will now quickly grow to 8 billion or more. Insertion speed is no longer the bottleneck and we can now complete a full crawl in just 1-2 weeks."

Use-case #3 - The Problem

- Size of database > RAM (12GB server also running data analysis jobs)
- Need to continuously ingest sensor data
- At 700GB InnoDB insert performance couldn't keep up with rate of inbound data
 - Archive the database and restart from empty
- Large database was also difficult to manage (backup and schema changes)

Use-case #2 - The Solution

- **Why TokuDB[®]?**
 - Significantly improved insertion rate
 - Far exceeds requirement, even at 700GB
 - Reduced disk footprint
 - Size of data files down from 650GB to 50GB due to compression
 - Single database
 - No need to search multiple databases for answers
 - Schema flexibility
 - Can add columns and indexes without downtime

Questions?

- We are looking for a Field Engineer and a Storage Engine Developer
- My contact information
 - tim@tokutek.com, @tmcallaghan on Twitter
- Downloads and Support
 - tokutek.com/products/downloads
 - Will need to register to get a login
 - Tarballs offer either MySQL or MariaDB with TokuDB
 - Commercial use free for < 50GB. Also free for academic, research, eval
 - support@tokutek.com, @tokutek on Twitter
- Technical Info
 - tokutek.com/technology
 - Understanding Indexing (Zardosht Kasheff)
 - » <http://vimeo.com/26454091>
 - How TokuDB[®] Fractal Tree[®] Indexes Work (Bradley Kuszmaul)
 - » <http://goo.gl/Smu6H>