mongoDB

Thanks for joining:

A Technical Introduction to WiredTiger

David Hows

Software Engineer (Storage), MongoDB

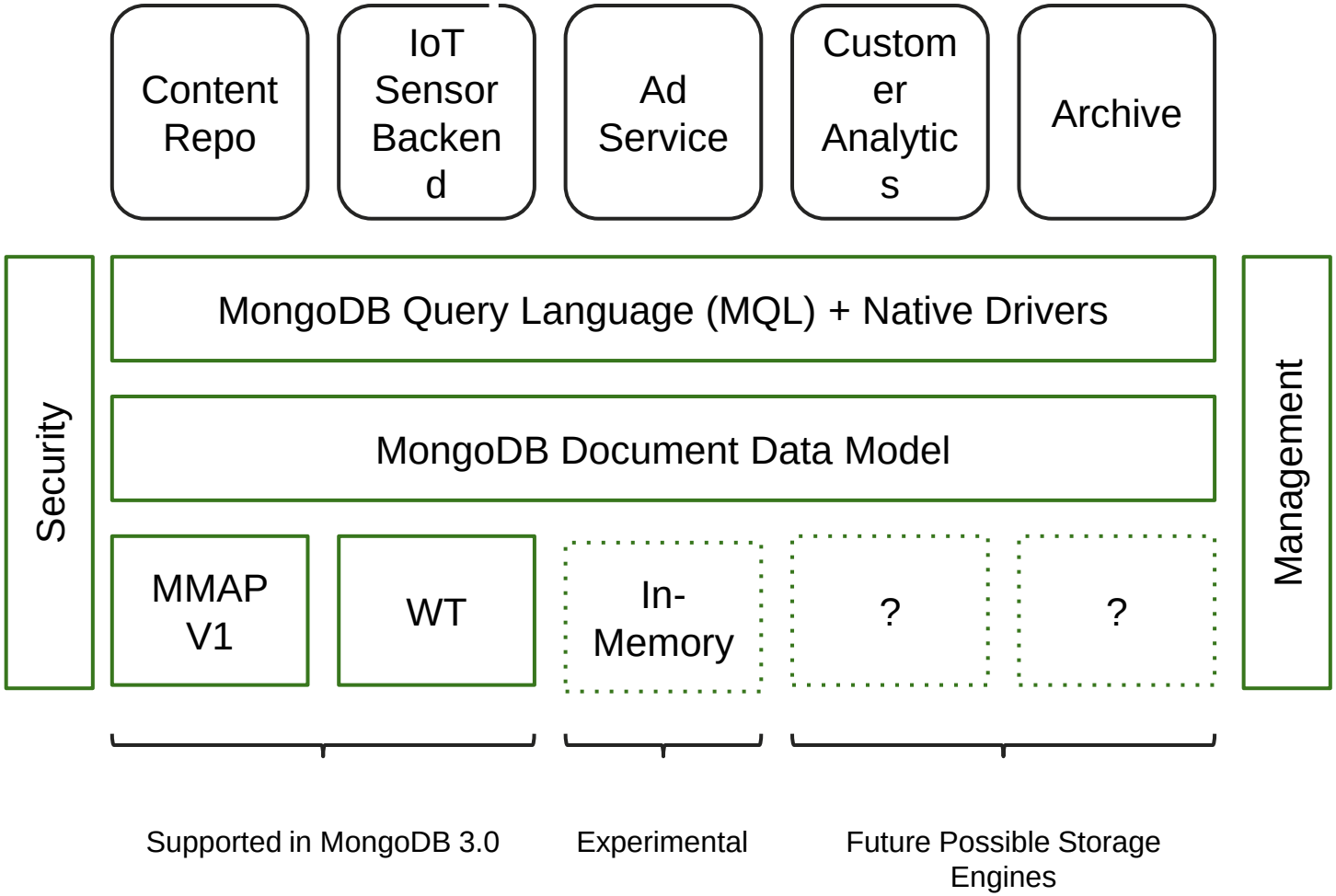
What You Will Learn

- MongoDB + WiredTiger Architecture
- In-memory performance
- Document-level concurrency
- Compression and checksums
- What’s next?

MongoDB's Storage Engine API

- Allows different storage engines to "plug-in"
  - different workloads have different performance characteristics
  - mmap is not ideal for all workloads
  - more flexibility
    - mix storage engines on same replica set/sharded cluster
    - MongoDB can innovate faster
- Opportunity to integrate further (HDFS, native encrypted, hardware optimized …)

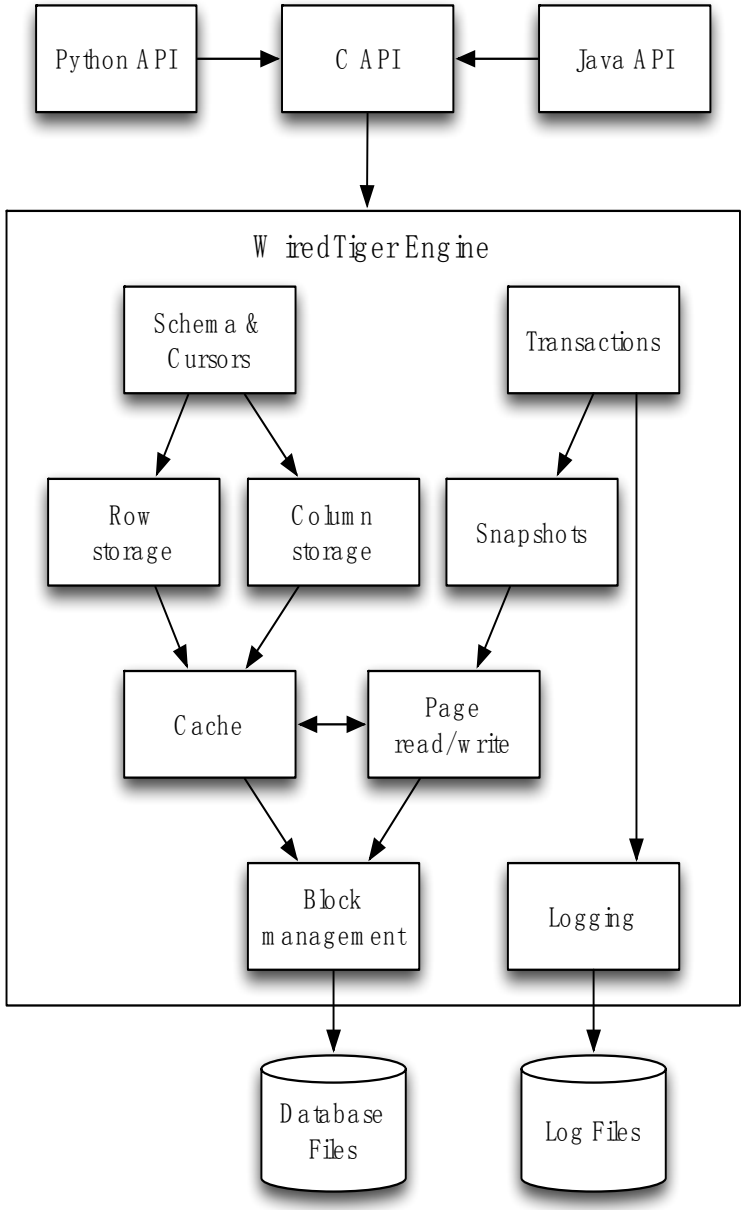
Storage Engine Layer



Motivation for WiredTiger

- Take advantage of modern hardware:
  - many CPU cores
  - lots of RAM
- Minimize contention between threads
  - lock-free algorithms, e.g., hazard pointers
  - eliminate blocking due to concurrency control
- Hotter cache and more work per I/O
  - compact file formats
  - compression

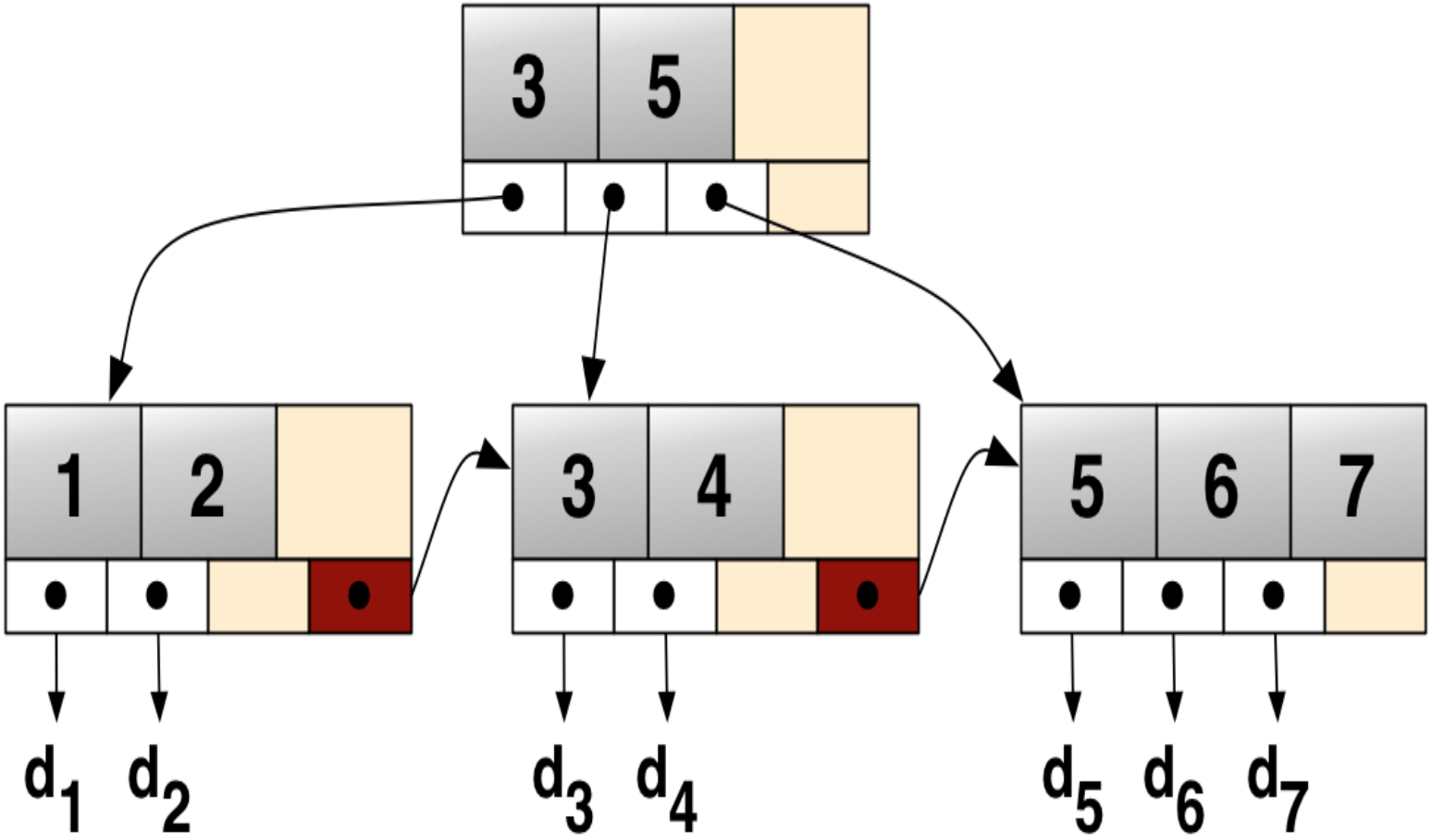
WiredTiger Architecture



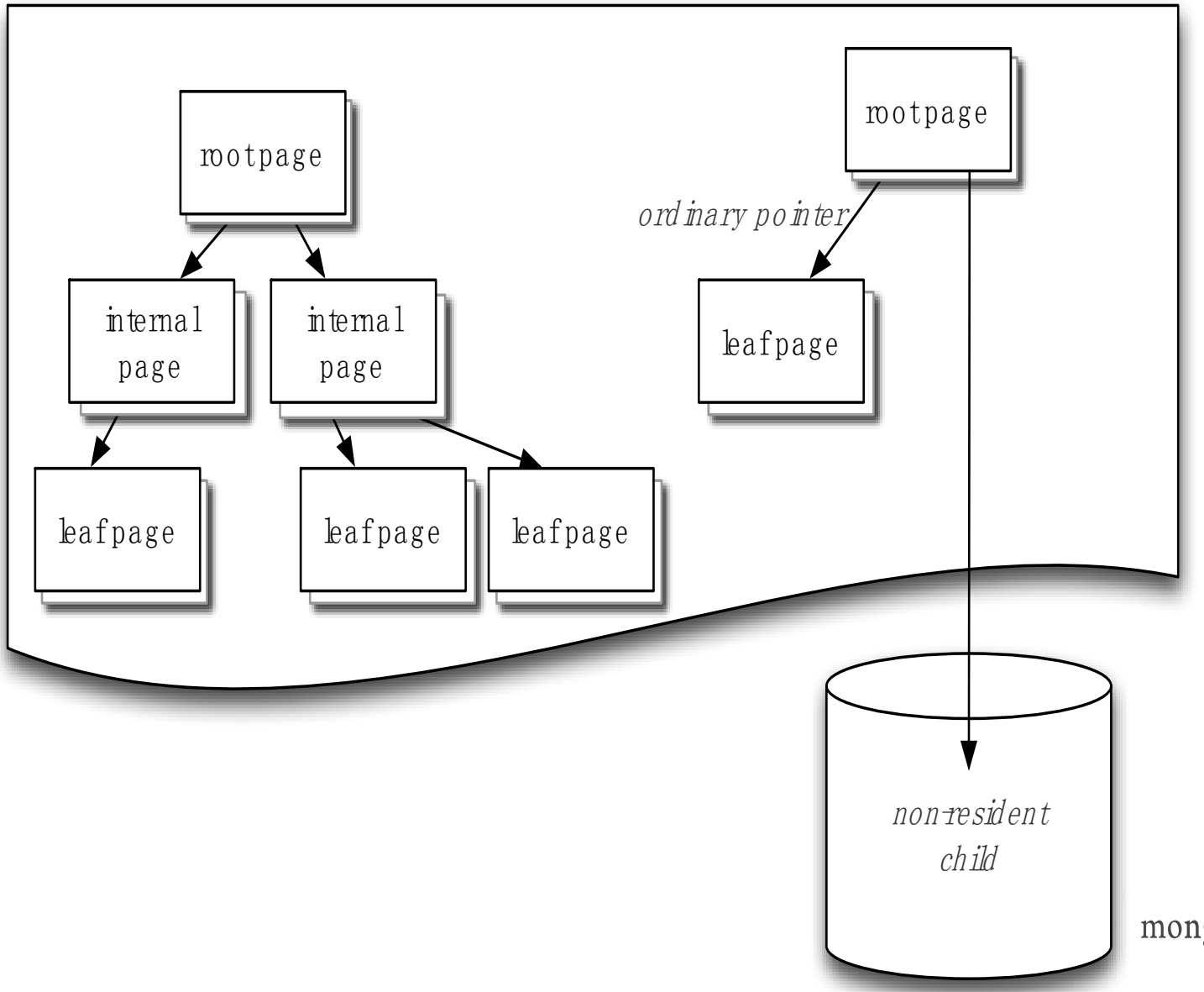
What You Will Learn

- ✓ MongoDB + WiredTiger Architecture
- ➢ In-memory performance
- Document-level concurrency
- Compression and checksums
- What’s next?

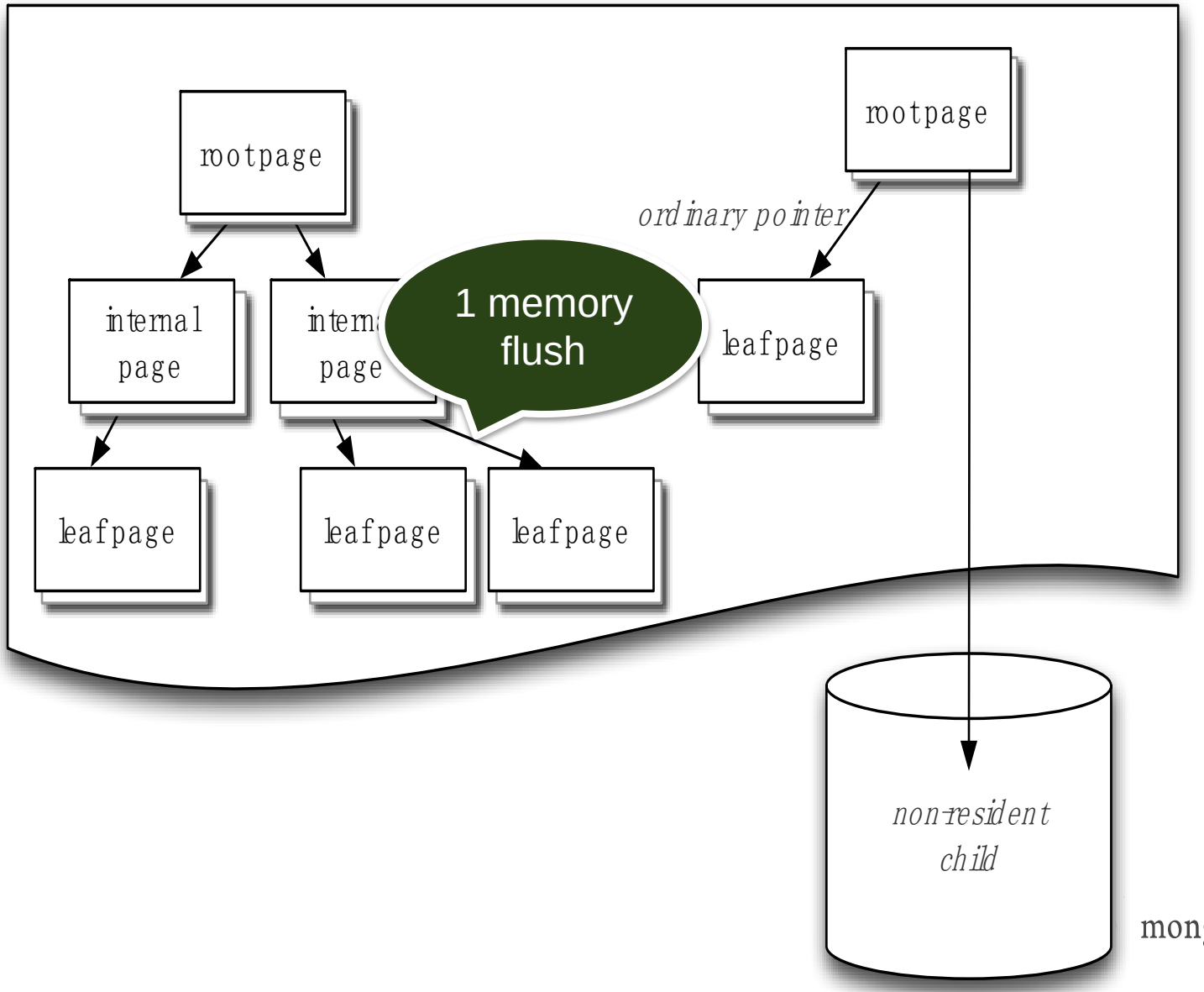
Traditional B+tree (ht wikipedia)



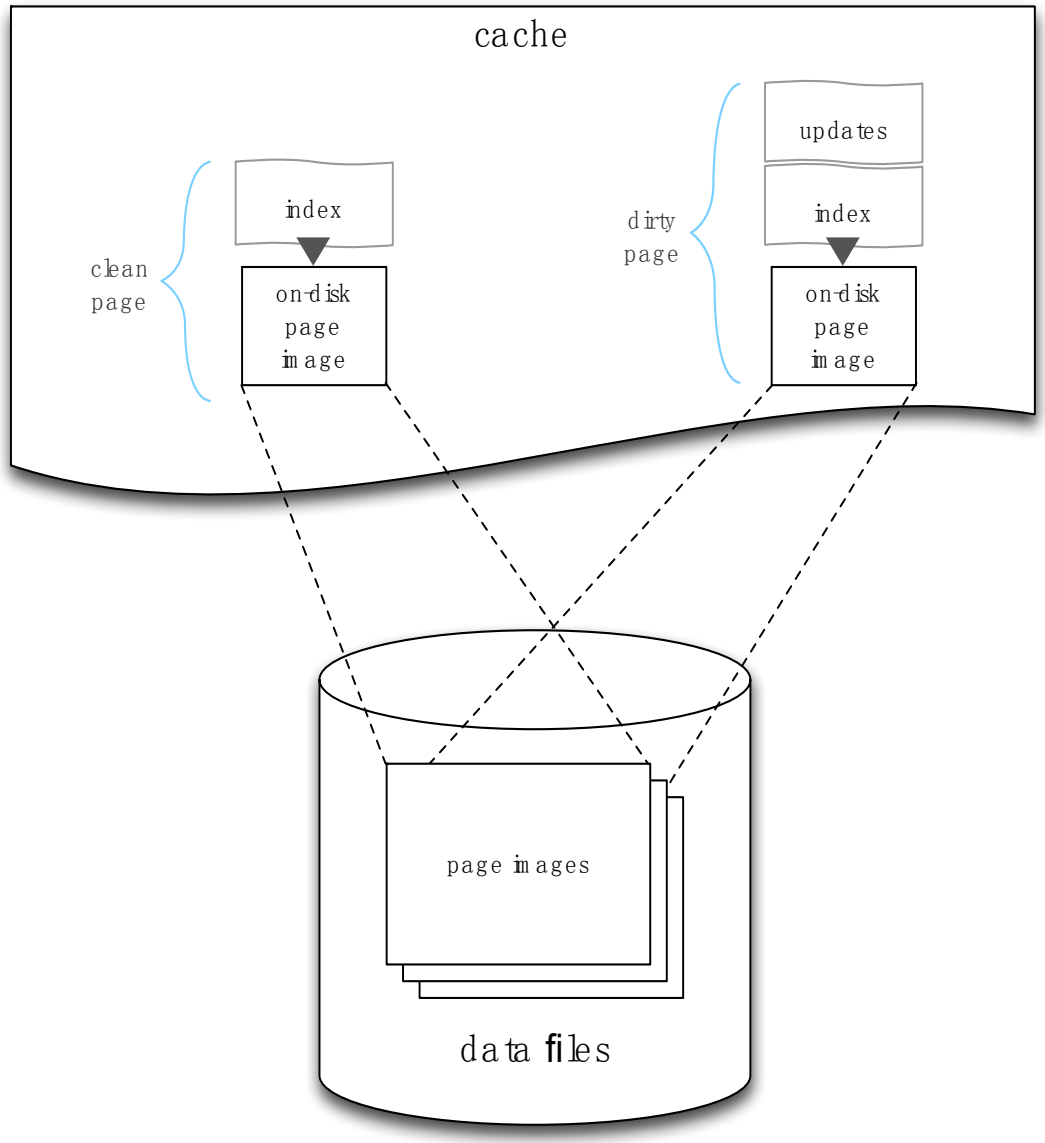
WiredTiger B+Trees in cache



Hazard Pointers



Pages in cache



In-memory performance

- Trees in cache are optimized for in-memory access
- Follow pointers to traverse a tree
  - no locking to access pages in cache
- Keep updates separate from clean data
- Do structural changes (eviction, splits) in background threads

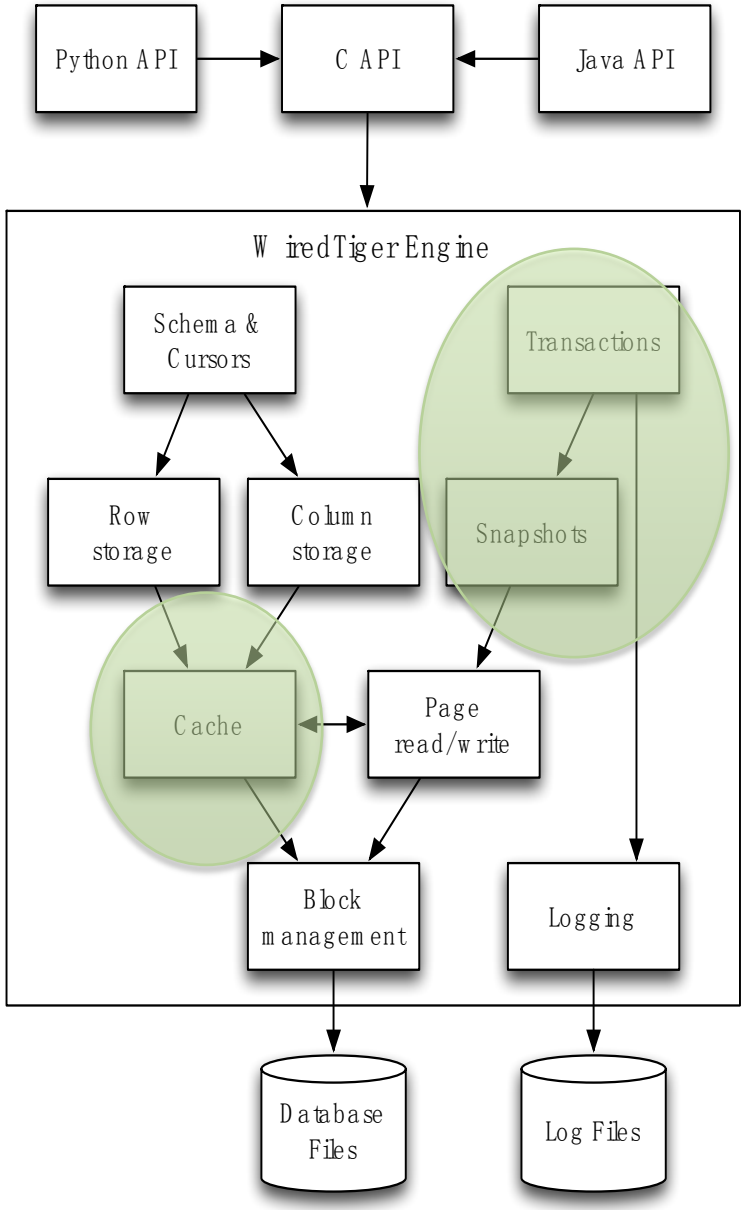
What You Will Learn

- ✓ MongoDB + WiredTiger Architecture
- ✓ In-memory performance
- ➢ Document-level concurrency
- • Compression and checksums
- • What’s next?

What is Concurrency Control?

- Computers have
  - multiple CPU cores
  - multiple I/O paths
- To make the most of the hardware, software has to execute multiple operations in parallel
- **Concurrency control has to keep data consistent**
- Common approaches:
  - locking
  - keeping multiple versions of data (MVCC)

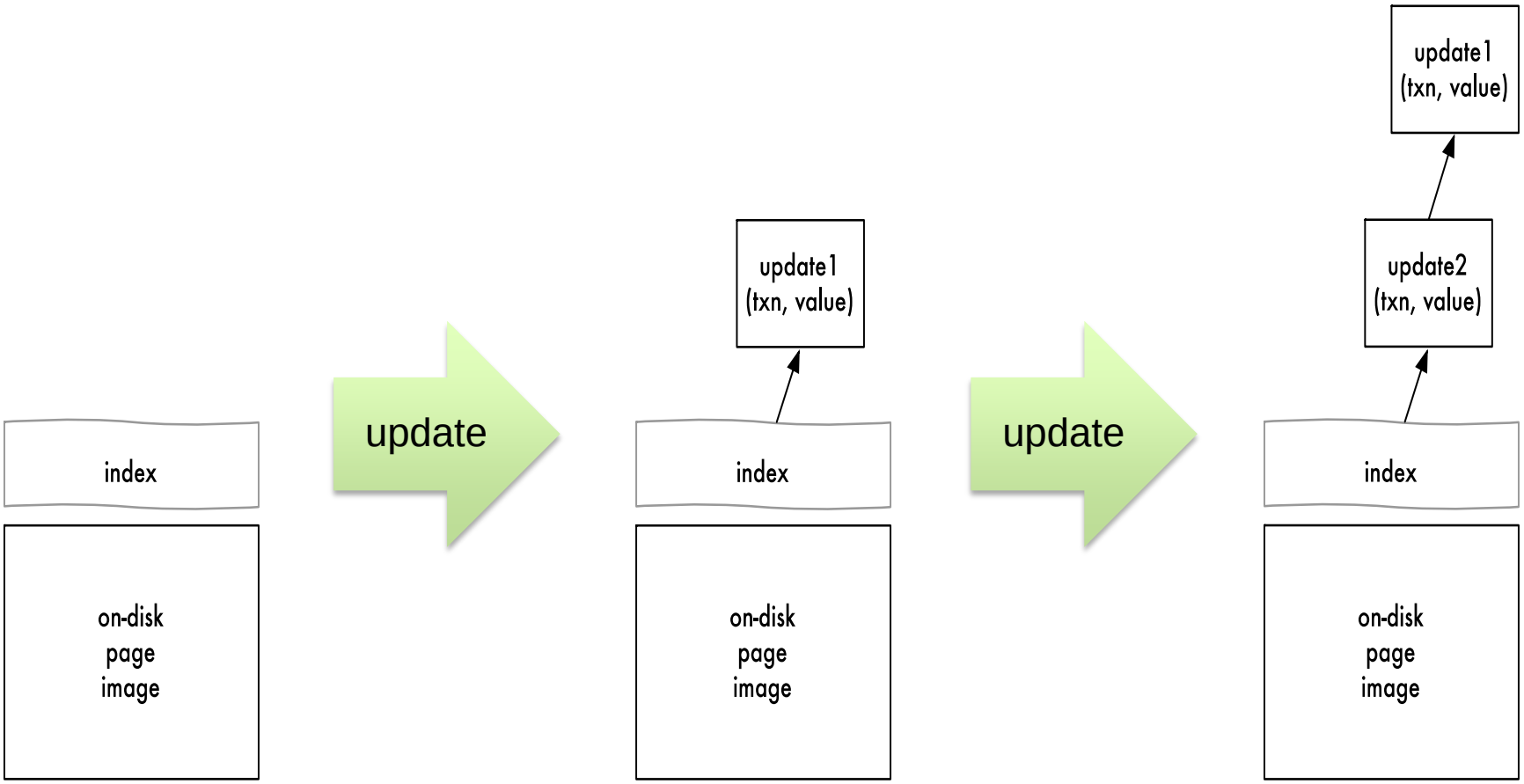
WiredTiger Concurrency Control



Multiversion Concurrency Control (MVCC)

- Multiple versions of records kept in cache
- Readers see the committed version before the transaction started
  - MongoDB “yields” turn large operations into small transactions
- Writers can create new versions concurrent with readers
- Concurrent updates to a single record cause write conflicts
  - MongoDB retries with back-off

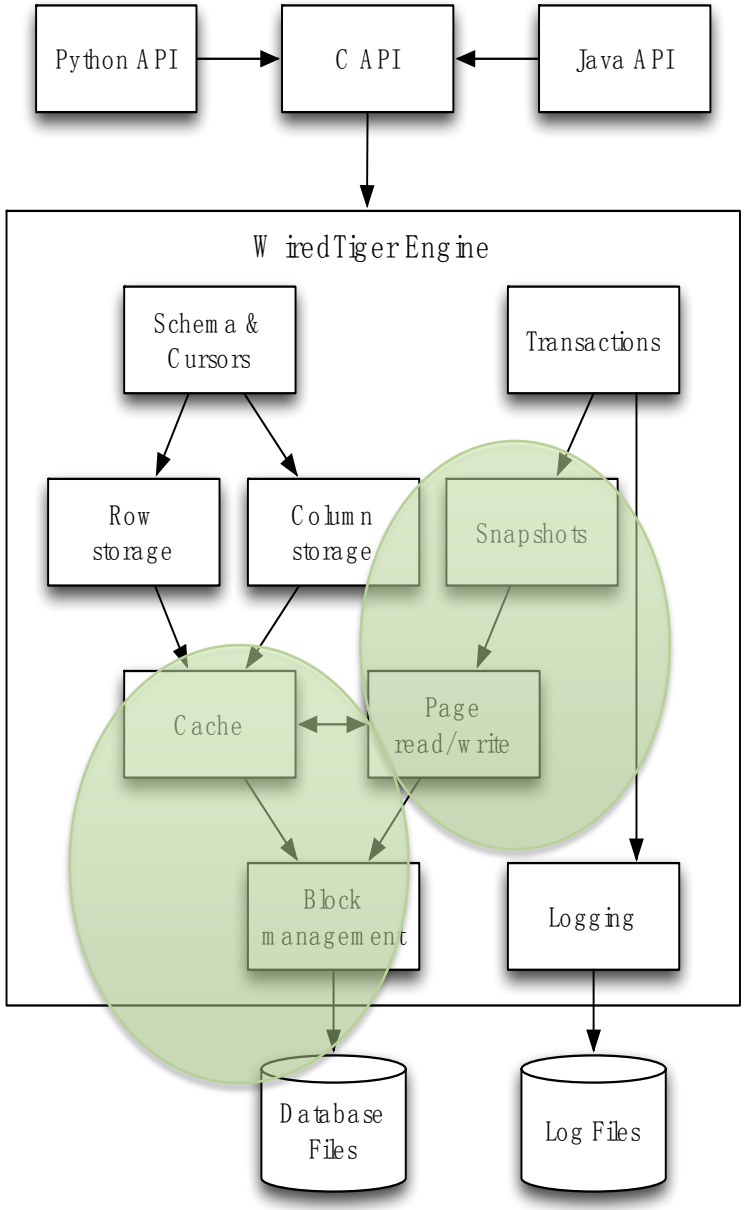
MVCC In Action



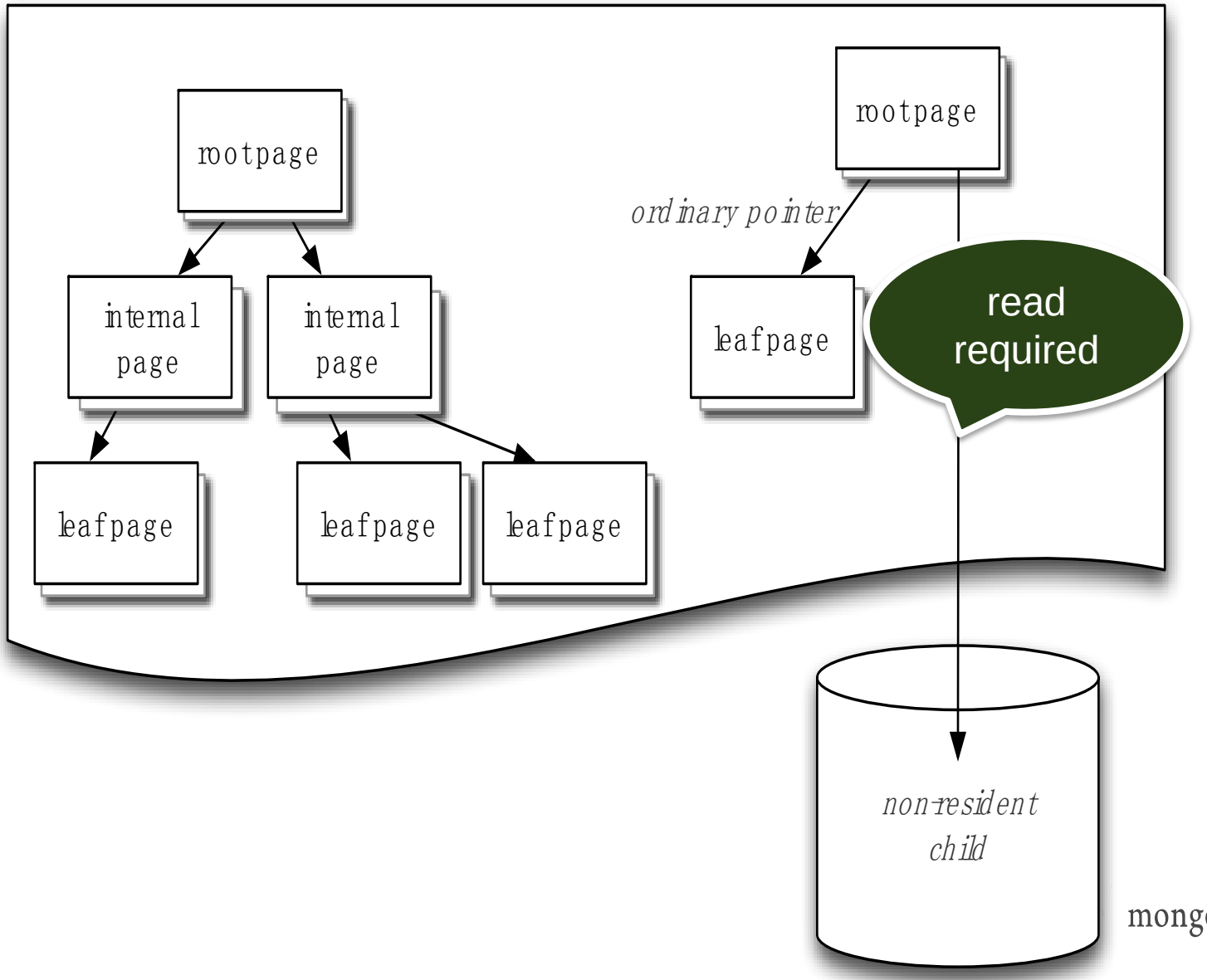
What You Will Learn

- ✓ MongoDB + WiredTiger Architecture
- ✓ In-memory performance
- ✓ Document-level concurrency
- ➢ Compression and checksums
- • What’s next?

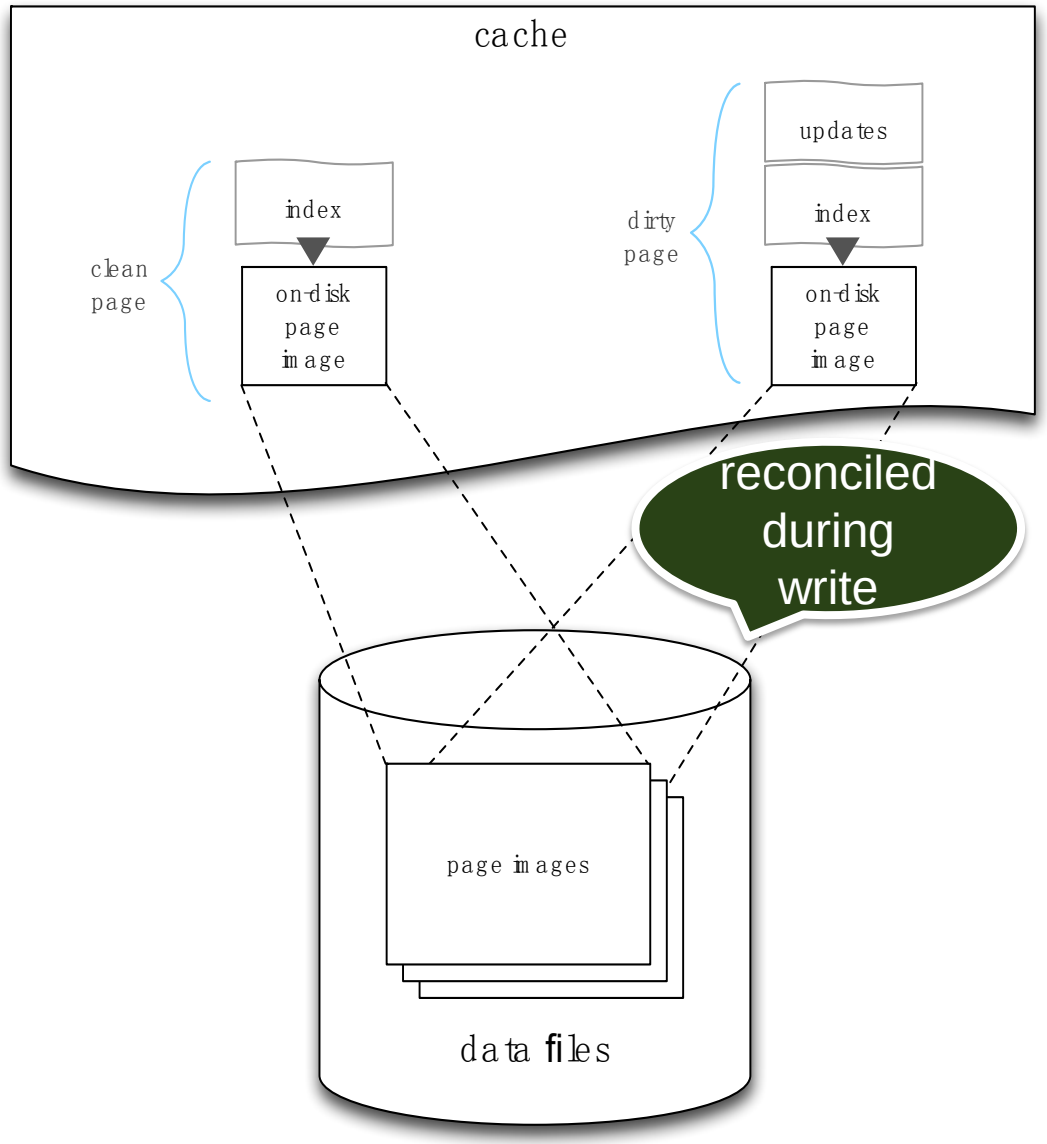
WiredTiger Disk Reads/Writes

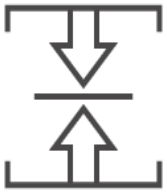


Trees in cache



Pages in cache

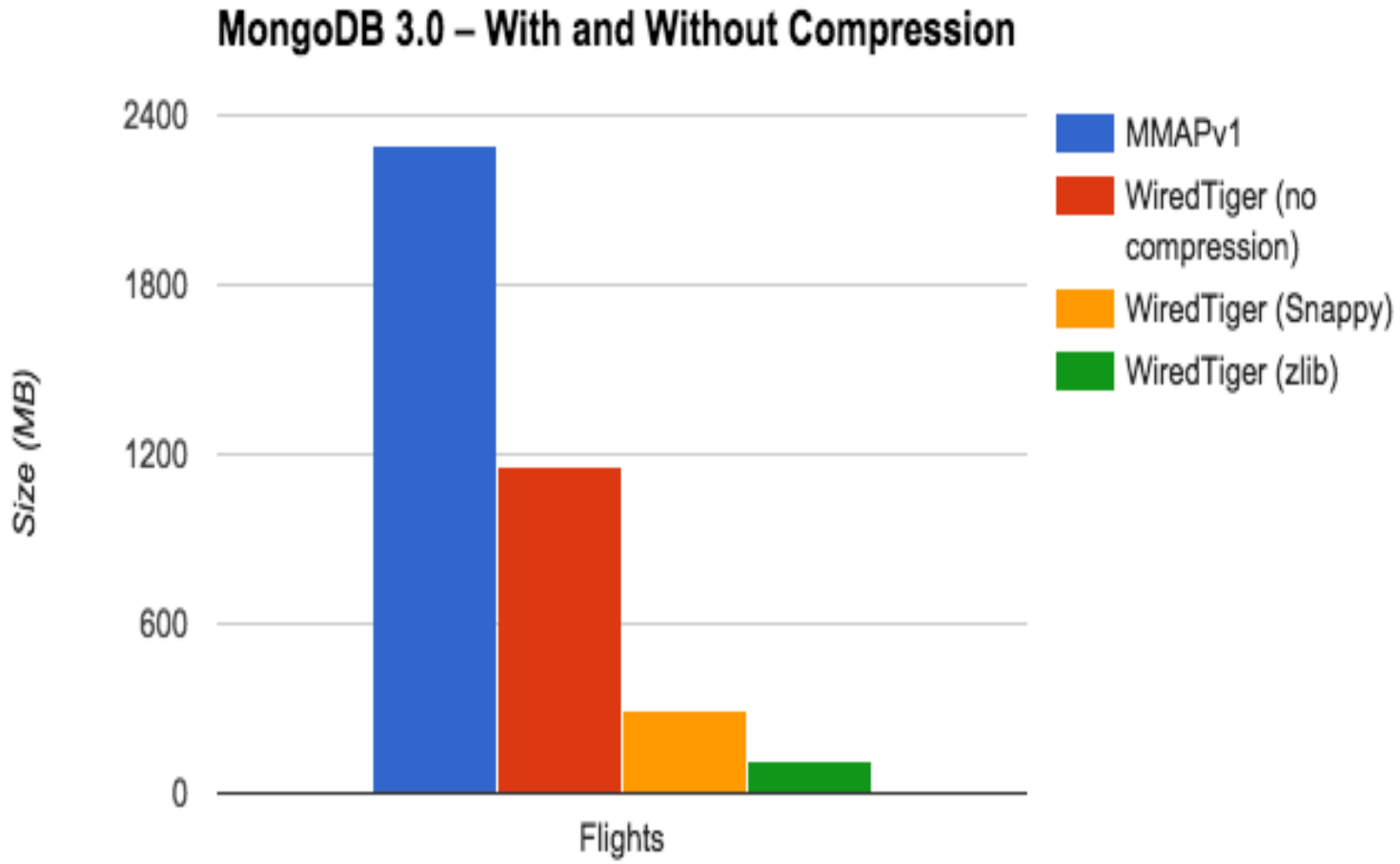




Compression

- WiredTiger uses snappy compression by default in MongoDB
- Supported compression algorithms:
  - snappy [default]: good compression, low overhead
  - zlib: better compression, more CPU
  - none
- Indexes also use prefix compression
  - stays compressed in memory

Compression in Action



(Flights database, ht Asya)

What You Will Learn

- ✓ WiredTiger Architecture
- ✓ In-memory performance
- ✓ Document-level concurrency
- ✓ Compression and checksums
- ➢ What’s next?

Adam Midvidy
@amidvidy

Follow

WiredTiger just became the default storage engine in the @MongoDB source tree. Congrats @WiredTigerInc @m_j_cahill github.com/mongodb/mongo/...

GitHub

SERVER-17861 Change the default storage engine to wiredTiger. ·...

WiredTiger is used as the default storage engine if the dbpath does not contain any data files. Otherwise, the storage engine specified in the storage.bson metadata file is used when the --storageEngi

View on web

RETWEETS	FAVORITES
16	10

7:04 PM - 21 May 2015

What's next for WiredTiger?

- Tune for (many) more workloads
  - avoid stalls during checkpoints with 100GB+ caches
  - make capped collections (including oplog) more efficient
  - Mark Callaghan seeing ~2x speedup in 3.2 snapshots: http://smalldatum.blogspot.com/
- Improving WiredTiger for out-of-cache, write-heavy workloads
- Adding encryption (in MongoDB 3.2)
- More advanced transactional semantics in the storage engine API

Thanks!

Questions?

David Hows
david.hows@mongodb.com