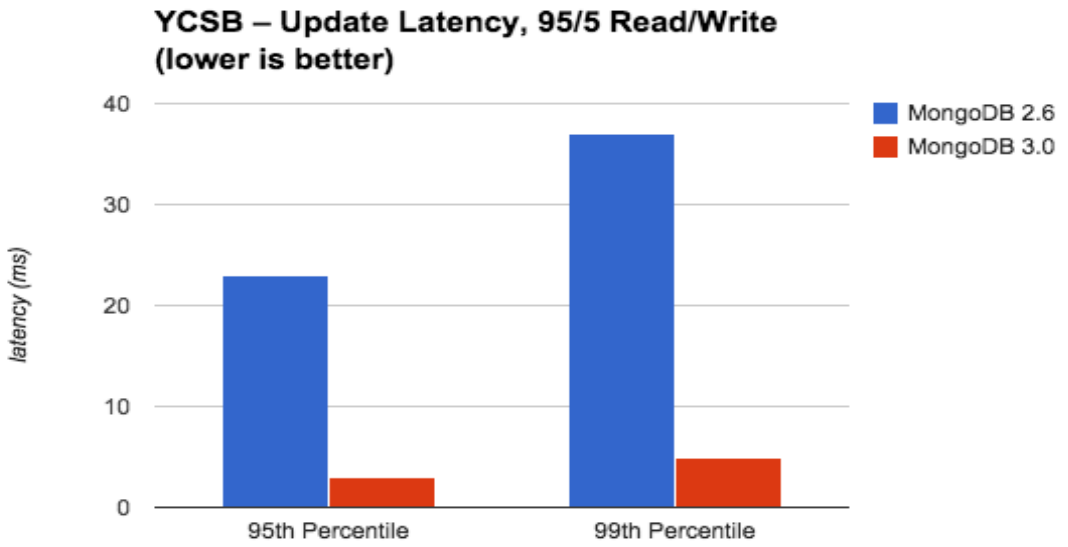
WiredTiger Internals

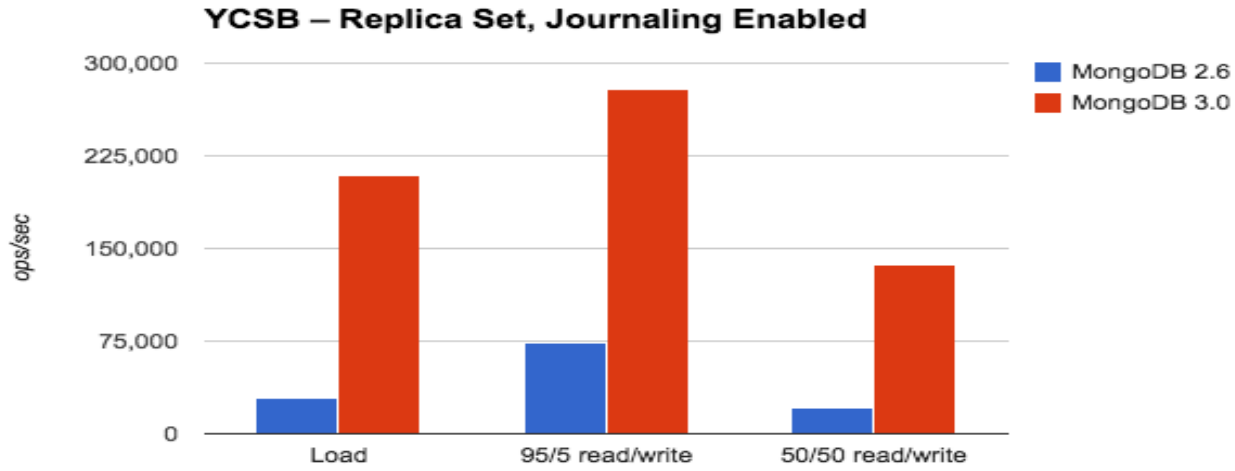
I'm Excited

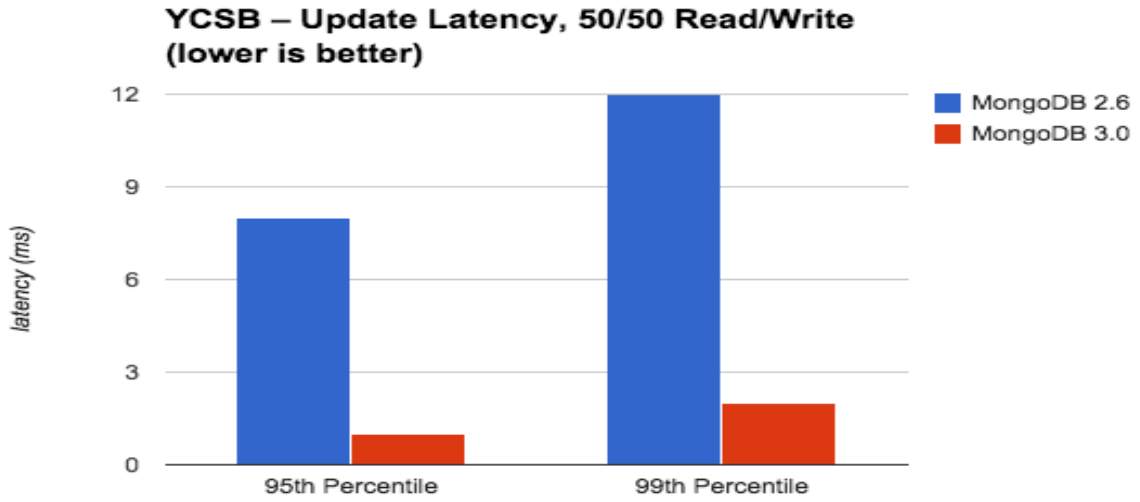
Benefits


Reduces operational overhead up to
95%
Global distribution with up to
50
replica set members
MongoDB
3.0
The world is changing.
Your database should too.
New flexible storage architecture including
WiredTiger
storage engine
WT
7-10x
better write performance
Up to
80%
less storage with compression

Performance!

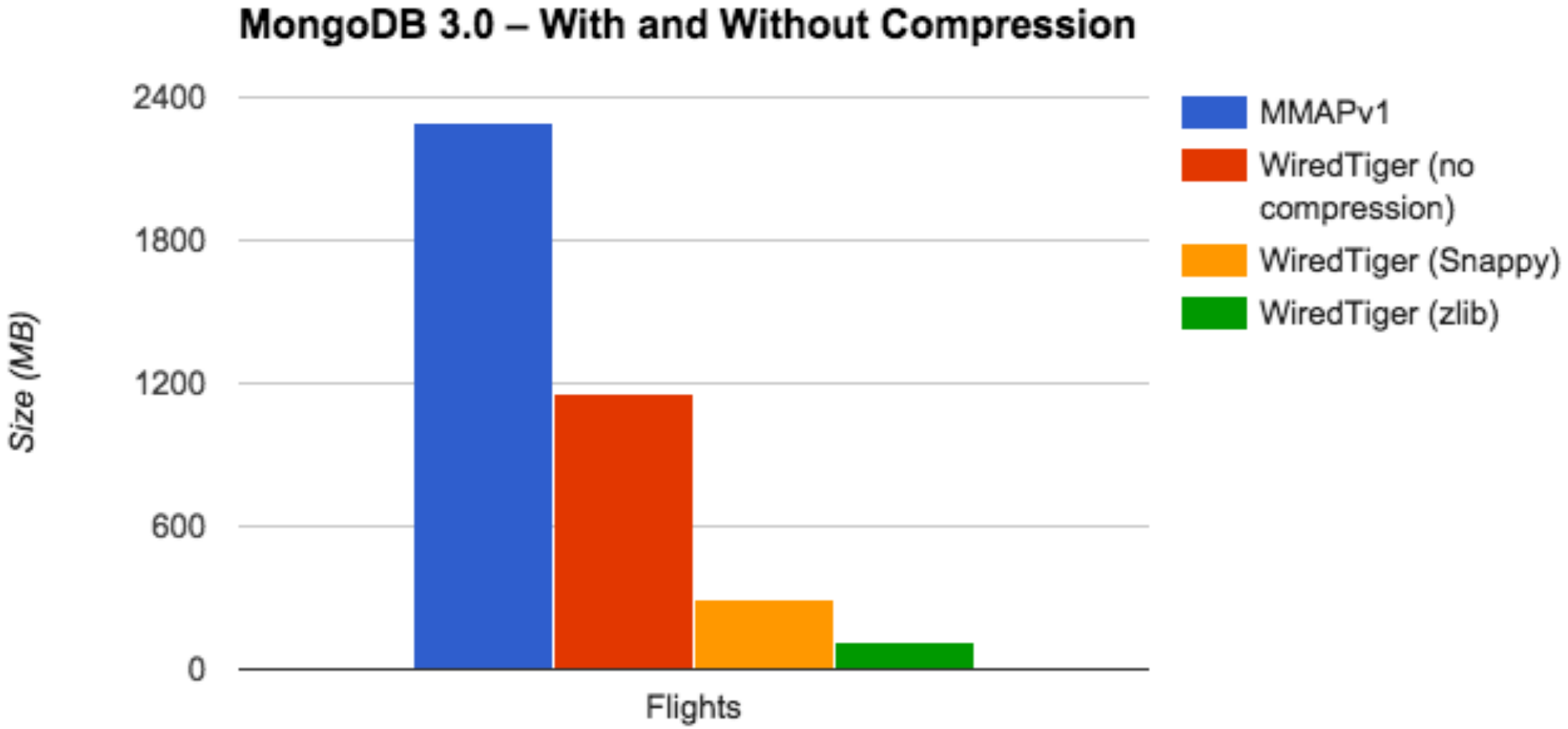






mongoDB

Compression in Action



(Flights database, ht Asya)

Agenda

Storage Engine API

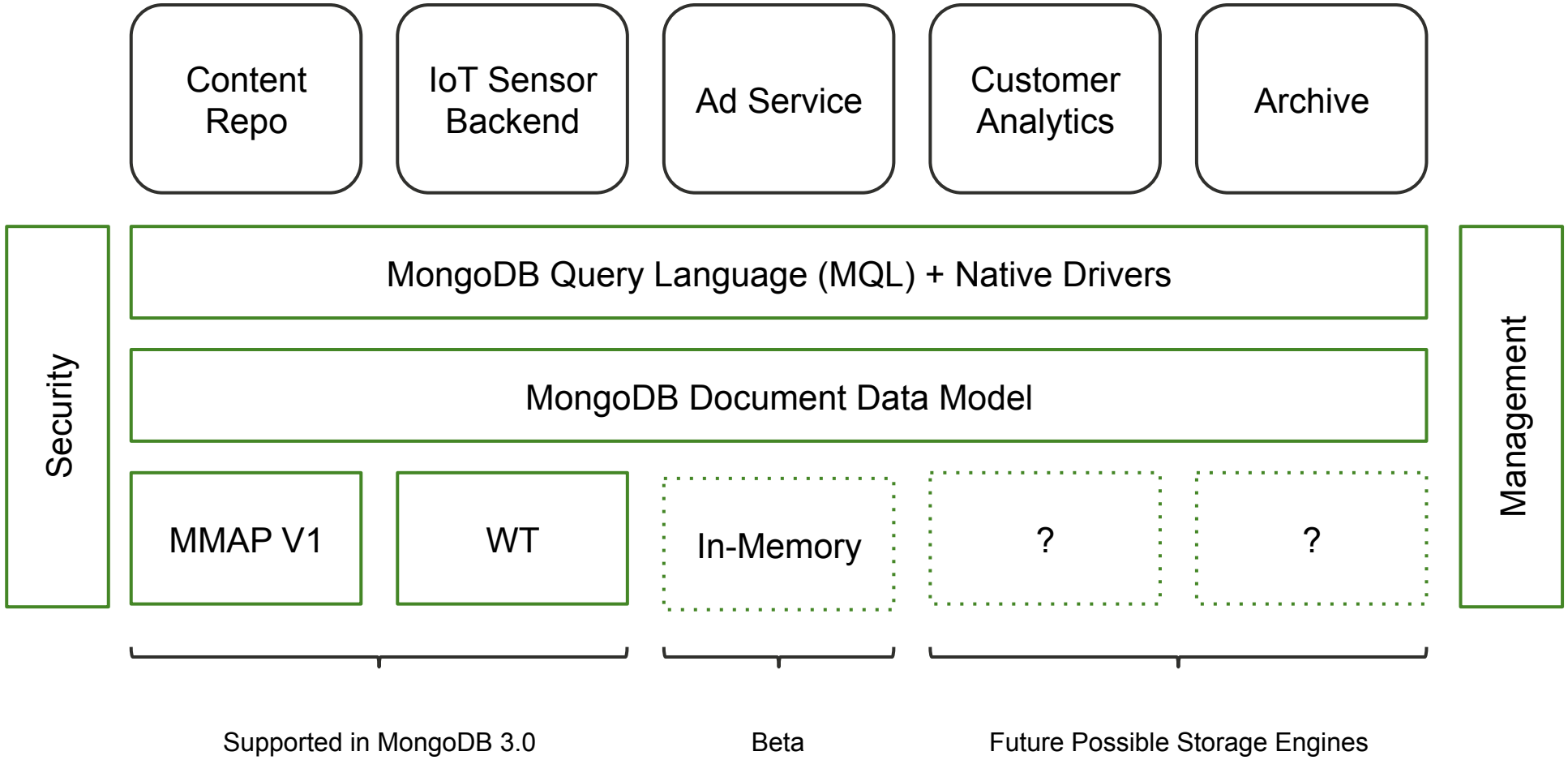
WiredTiger Architecture

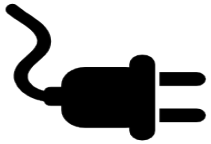
Overall Benefits

Pluggable Storage Engine API

http://www.livingincebuforums.com/ipb/uploads/monthly_10_2011/post-198-0-67871200-1318223706.jpg

MongoDB Architecture

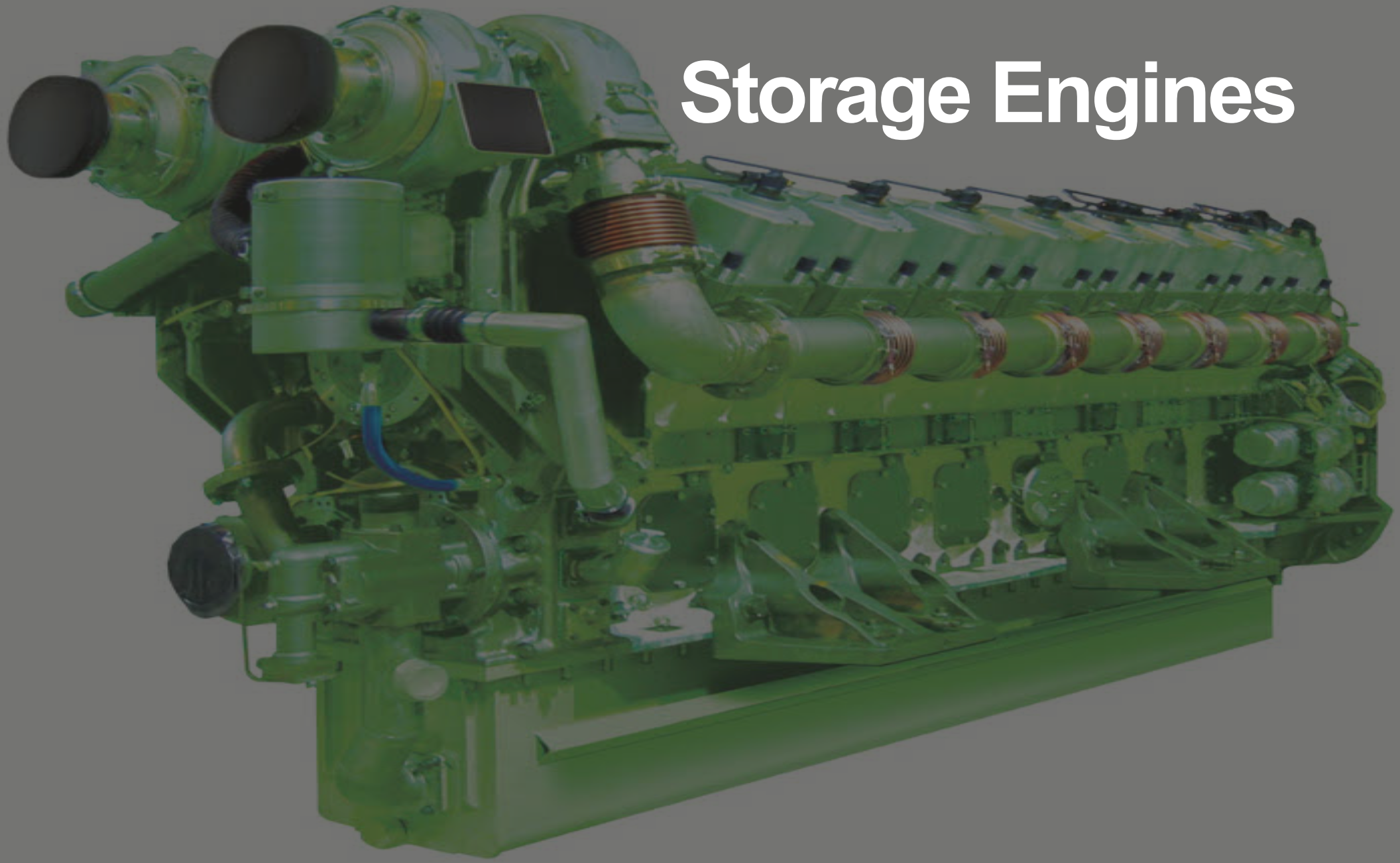




Storage Engine API

- Allows to "plug-in" different storage engines
  - Different use cases require different performance characteristics
  - mmapv1 is not ideal for all workloads
  - More flexibility
    - Can mix storage engines on same replica set/sharded cluster
- Opportunity to integrate further (HDFS, native encrypted, hardware optimized …)

Storage Engines



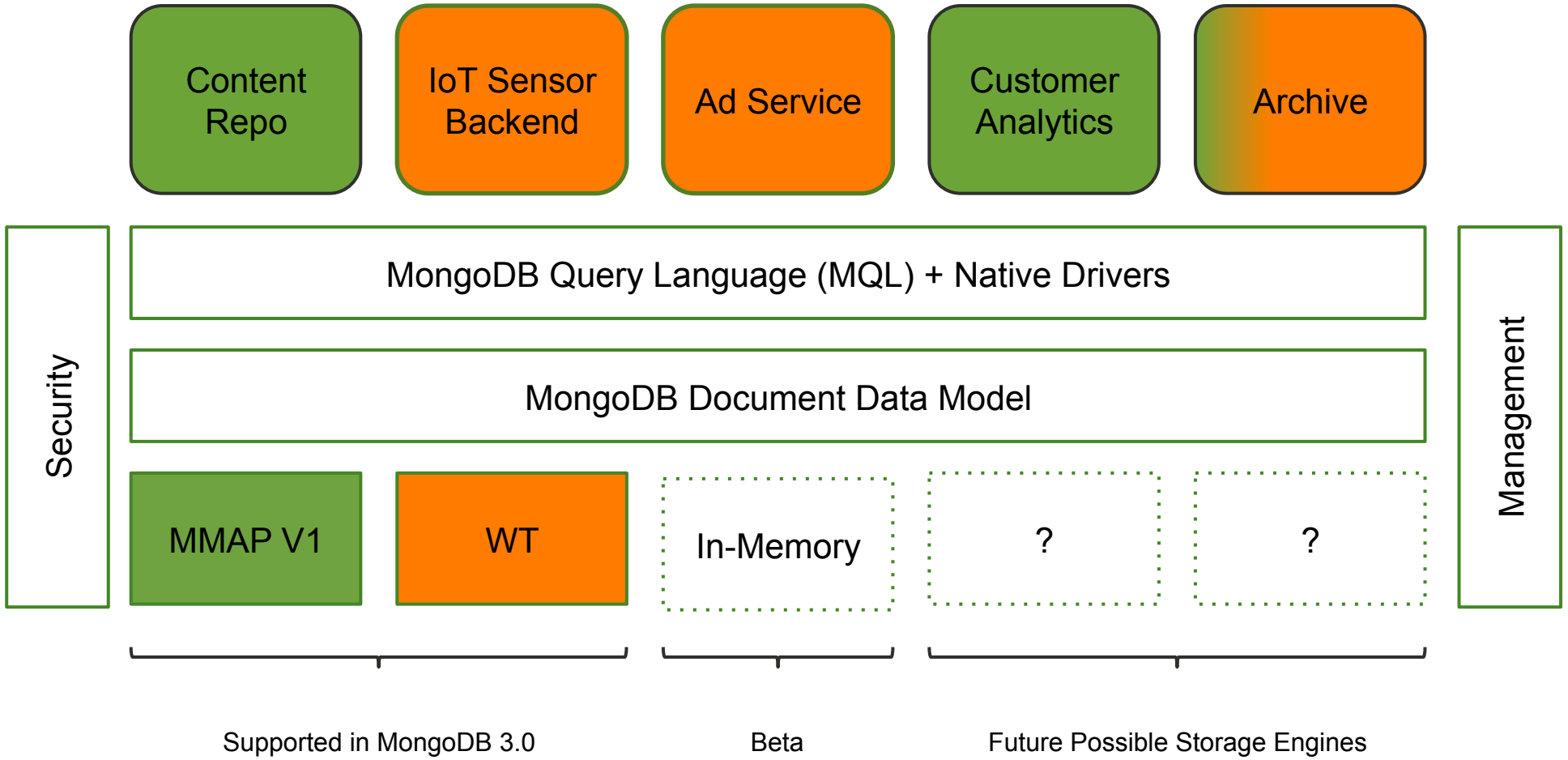
2 Storage Engines Available

MMAPv1

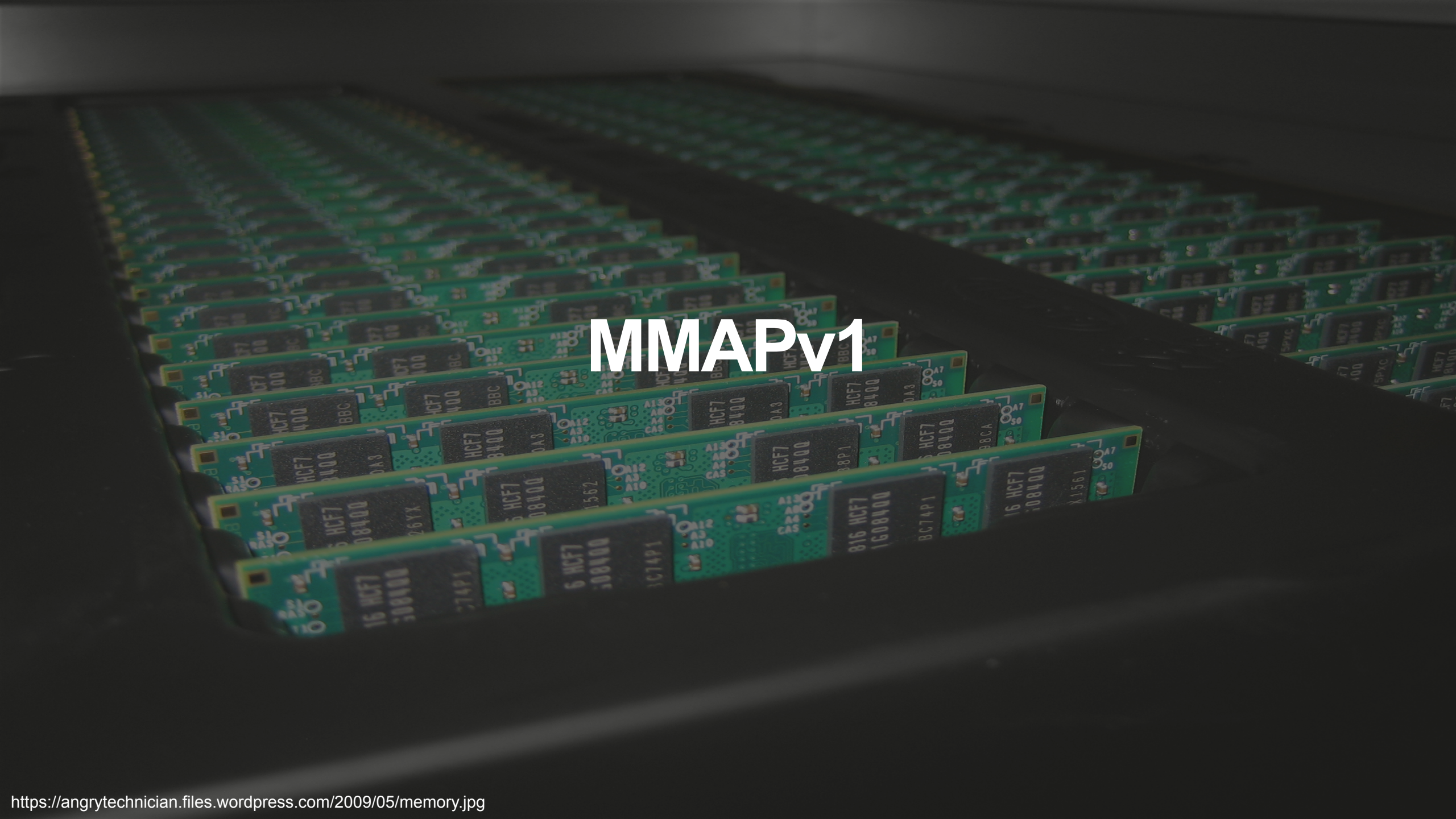
… and more in the making!

WiredTiger

MongoDB Architecture

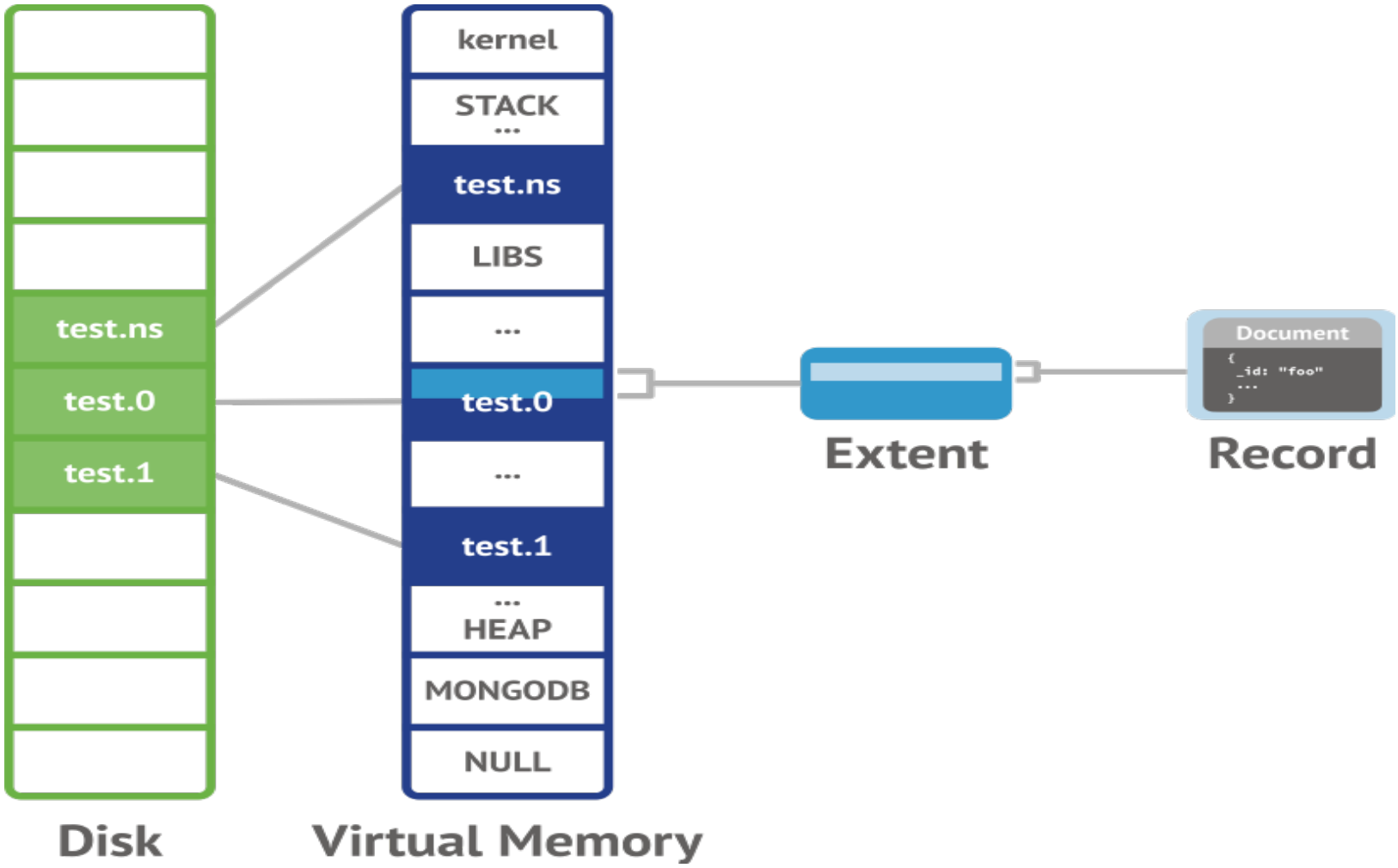


mongoDB



https://angrytechnician.files.wordpress.com/2009/05/memory.jpg

MMAPv1


test.ns
test.0
test.1
kernel
STACK
...
test.ns
LIBS
...
test.0
...
test.1
...
HEAP
MONGODB
NULL
Extent
Document
{
_id: "foo"
...
}
Record
Disk
Virtual Memory

MMAPv1

3.0 Default

- Improved concurrency control
- Great performance on read-heavy workloads
- Data & Indexes memory mapped into virtual address space
- Data access is paged into RAM
- OS evicts using LRU
- More frequently used pages stay in RAM

WiredTiger

http://cdn.theatlantic.com/static/infocus/ngt051713/n10_00203194.jpg

mongod **--storageEngine** **wiredTiger**

What is WiredTiger?

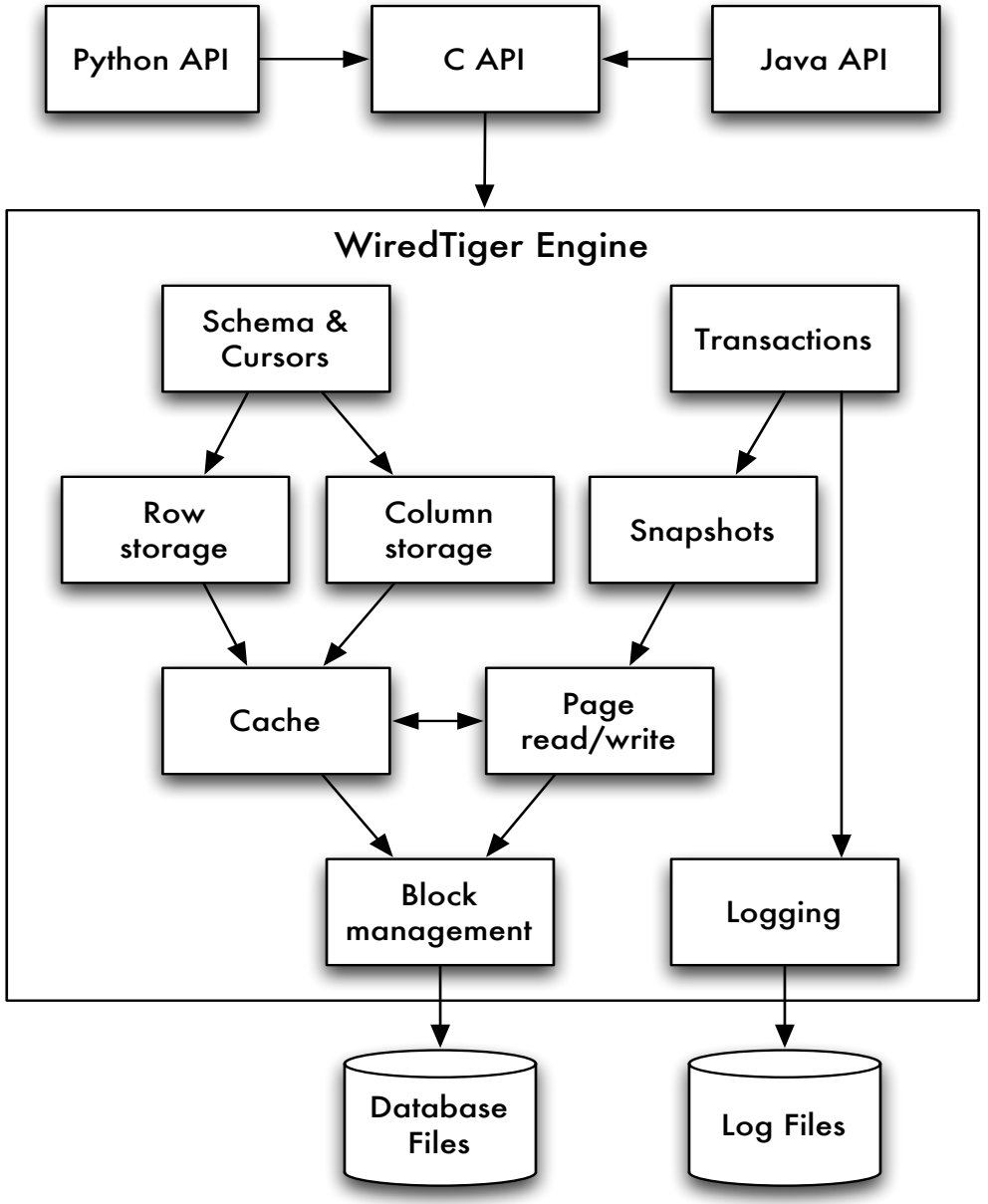
- Storage engine company founded by BerkeleyDB alums
- Recently acquired by MongoDB
- Available as a storage engine option in MongoDB 3.0

Motivation for WiredTiger

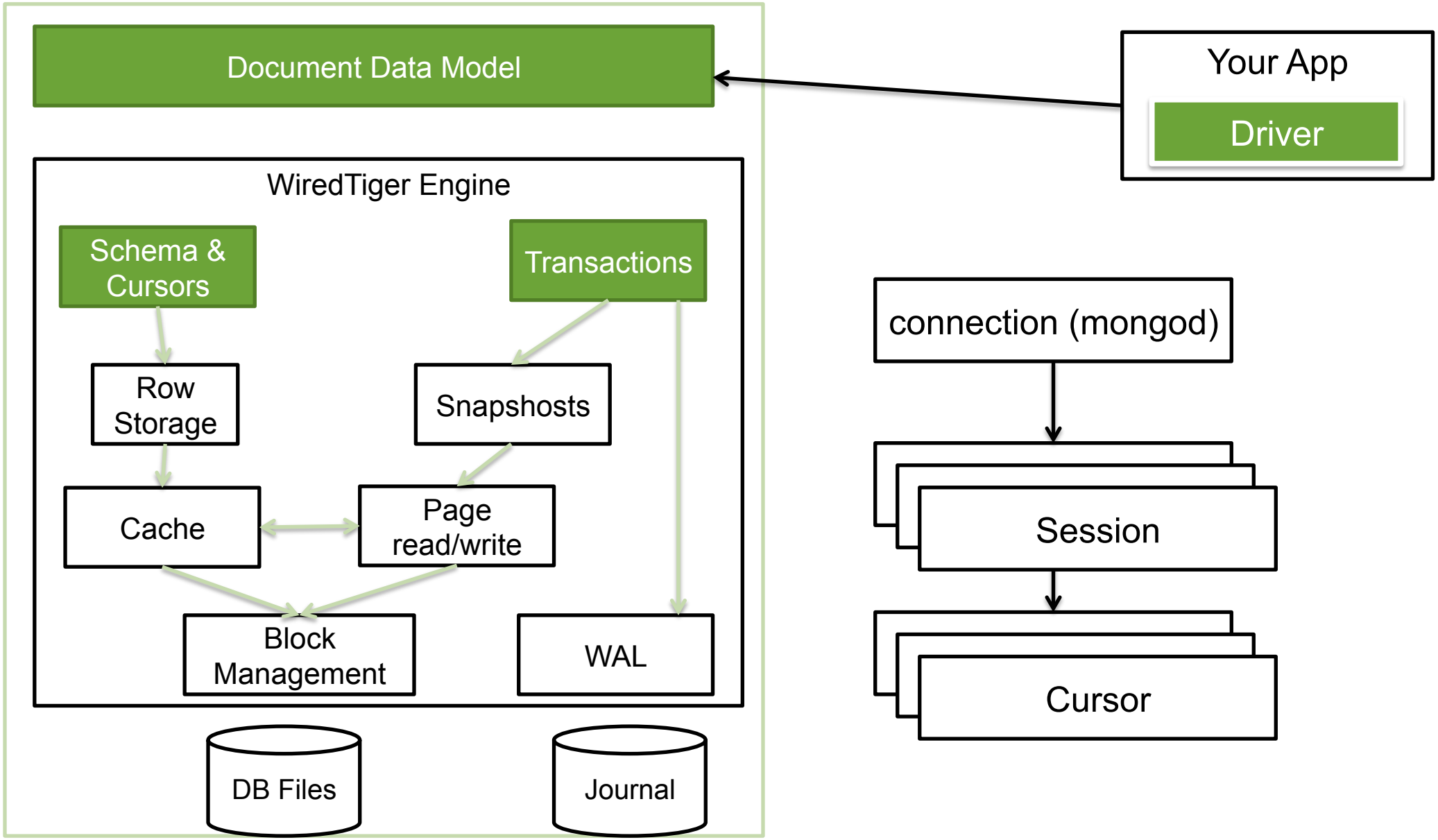
- Take advantage of modern hardware:
  - many CPU cores
  - lots of RAM
- Minimize contention between threads
  - lock-free algorithms, e.g., hazard pointers
  - eliminate blocking due to concurrency control
- Hotter cache and more work per I/O
  - compact file formats
  - compression

WiredTiger Architecture

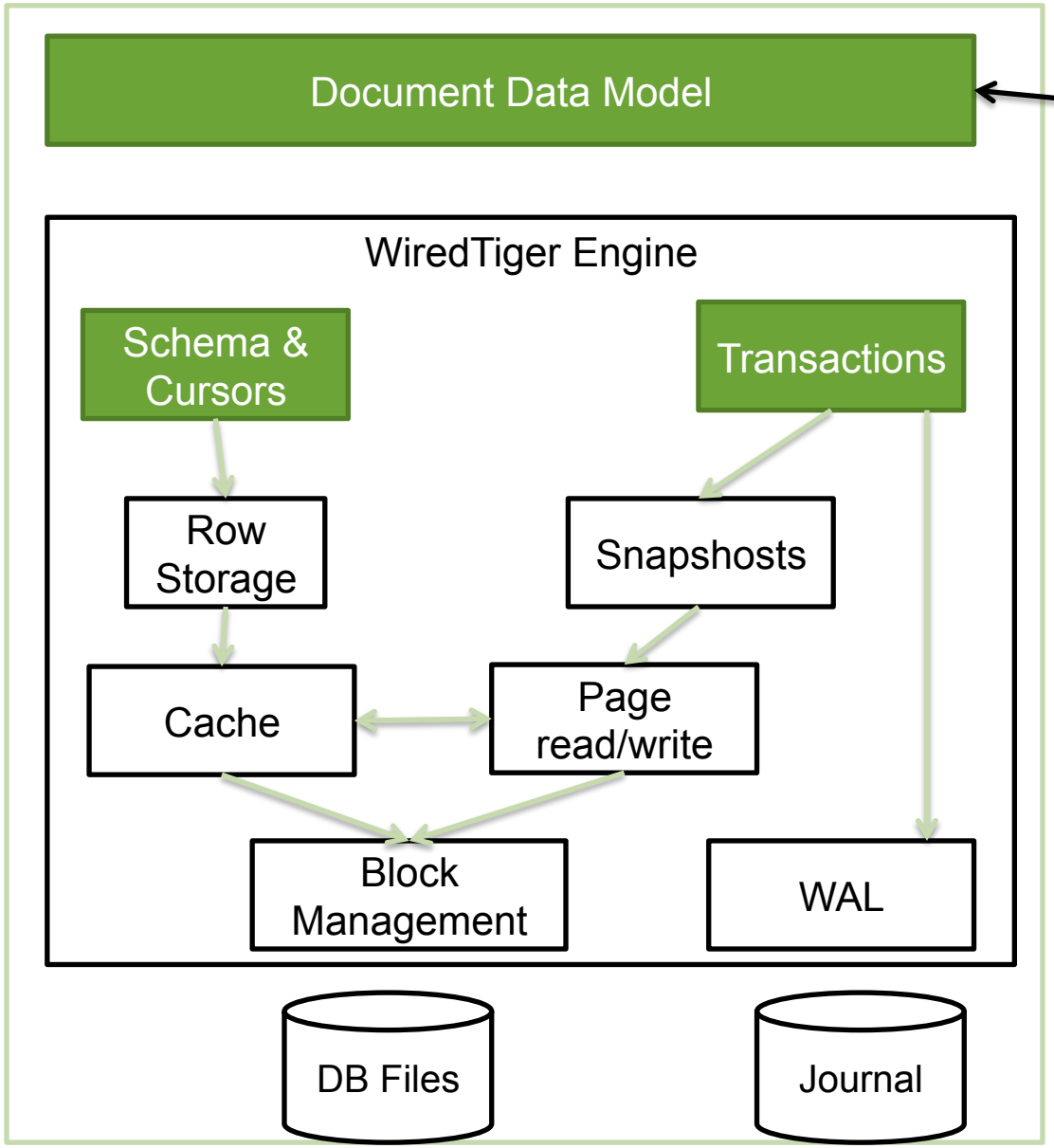
WiredTiger Architecture

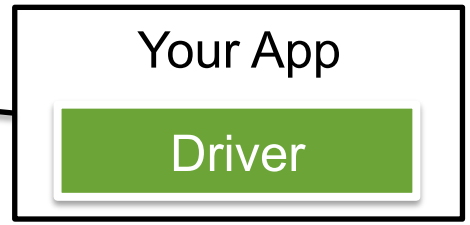


WiredTiger Sessions



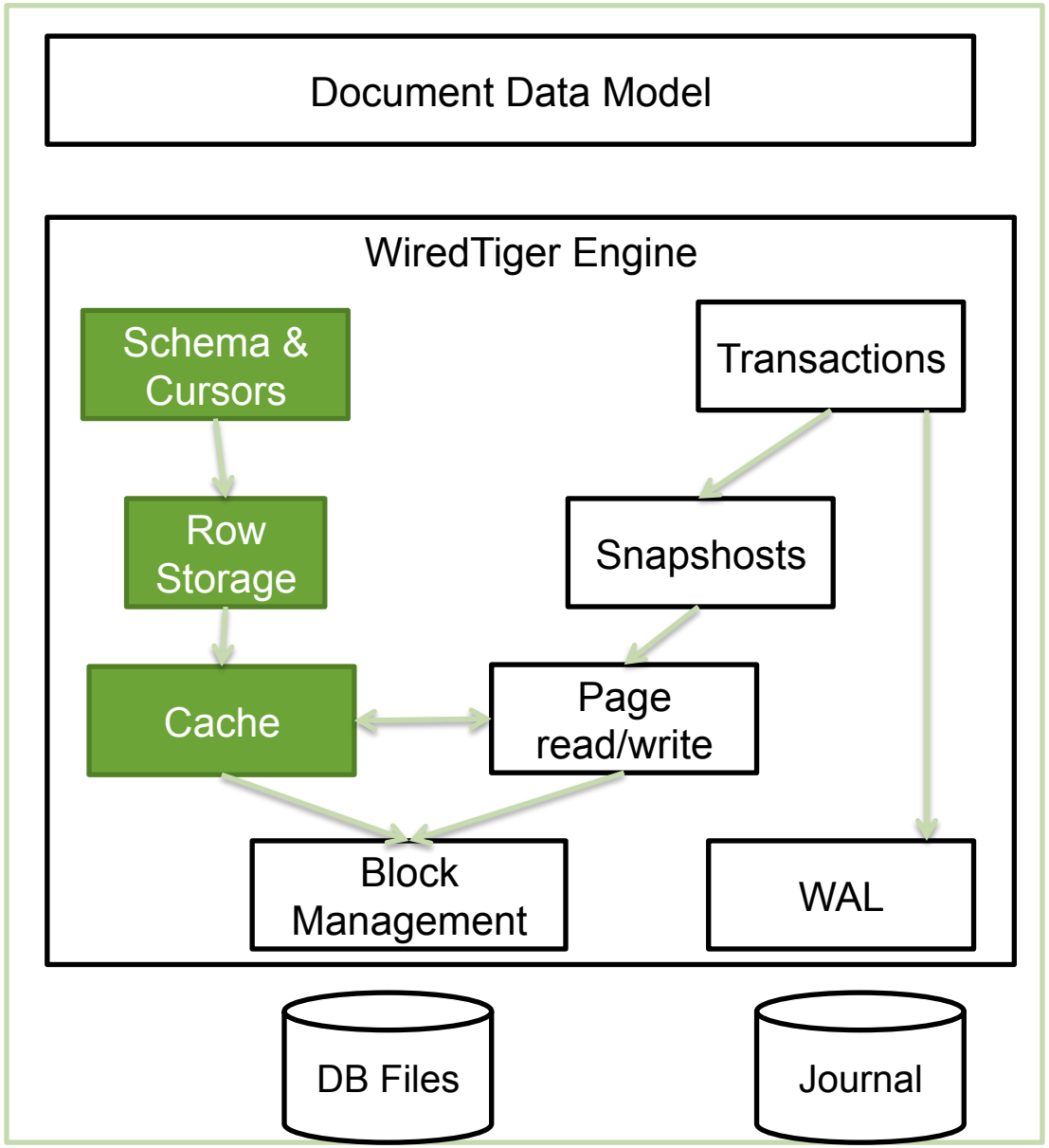
WiredTiger Sessions





- 3 main purposes
  - Schema Operations
  - Transaction Management
  - Cursor Creation

WiredTiger Schema & Cursors




```
db.createCollection("somecollection")
```

```
db.somecollection.drop()
```

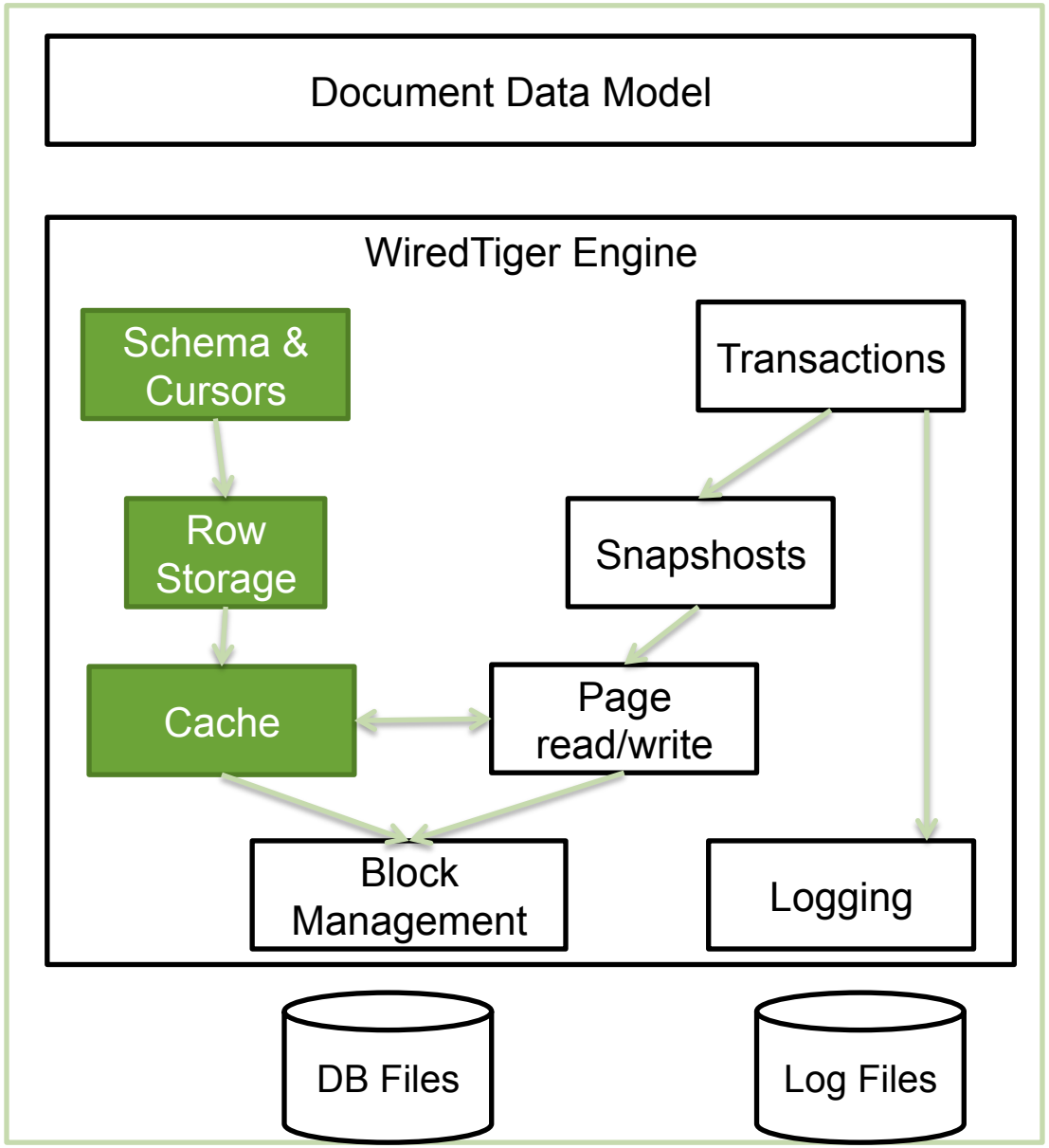
```
db.somecollection.find({...})
```

```
db.somecollection.remove({_id:111})
```

```
db.somecollection.stats()
```

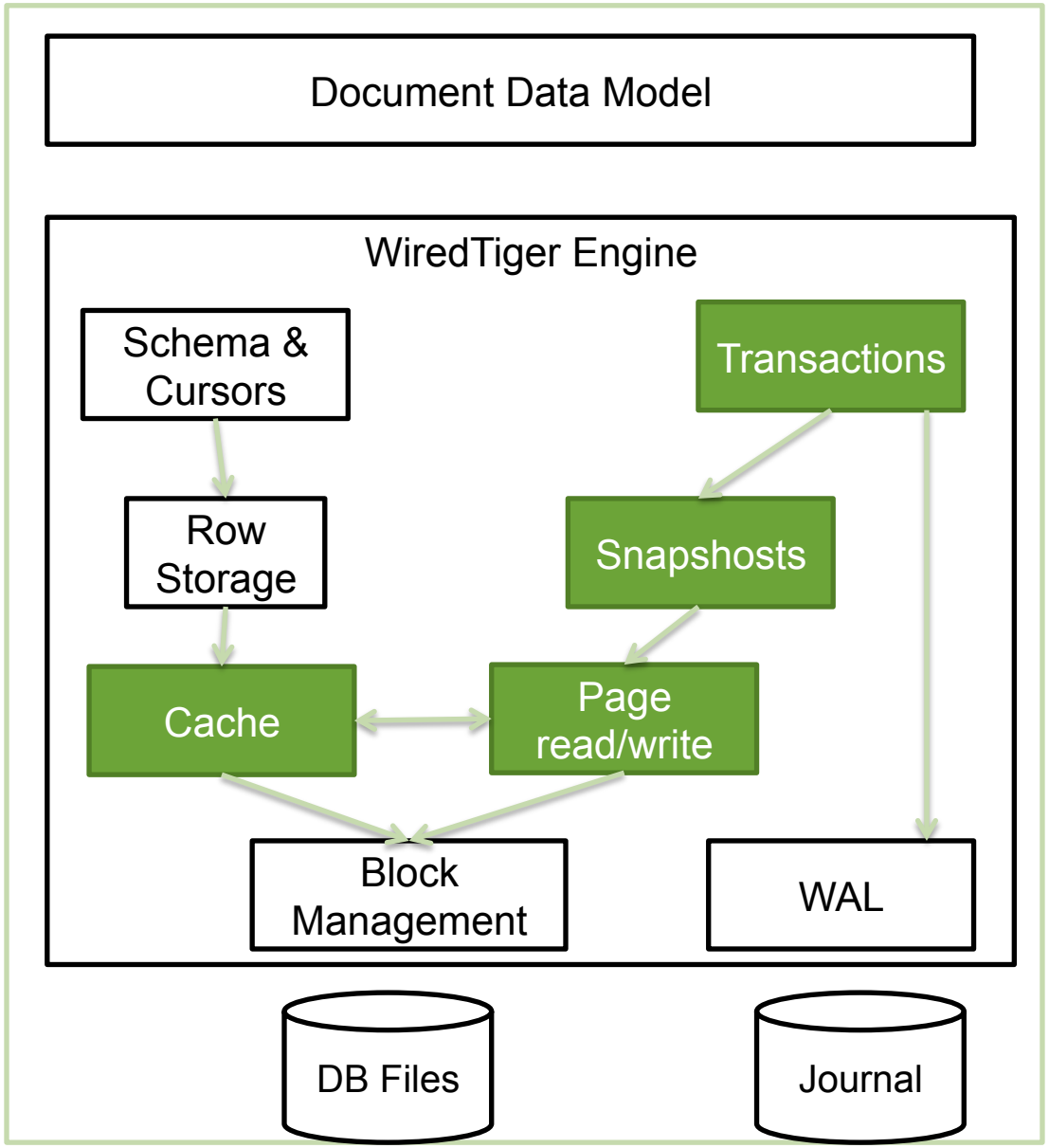
```
db.somecollection.explain().find({_id:111})
```

WiredTiger Schema & Cursors



- Schema Operations
  - create, drop, truncate, verify …
- Cursors
  - CRUD
  - Data iteration
  - Statistics

WiredTiger Transactions



COMMAND INSERT

```
{
  "_id": 111,
  "name": "Spock",
  "message": "Live Long and Prosper!"
}
```

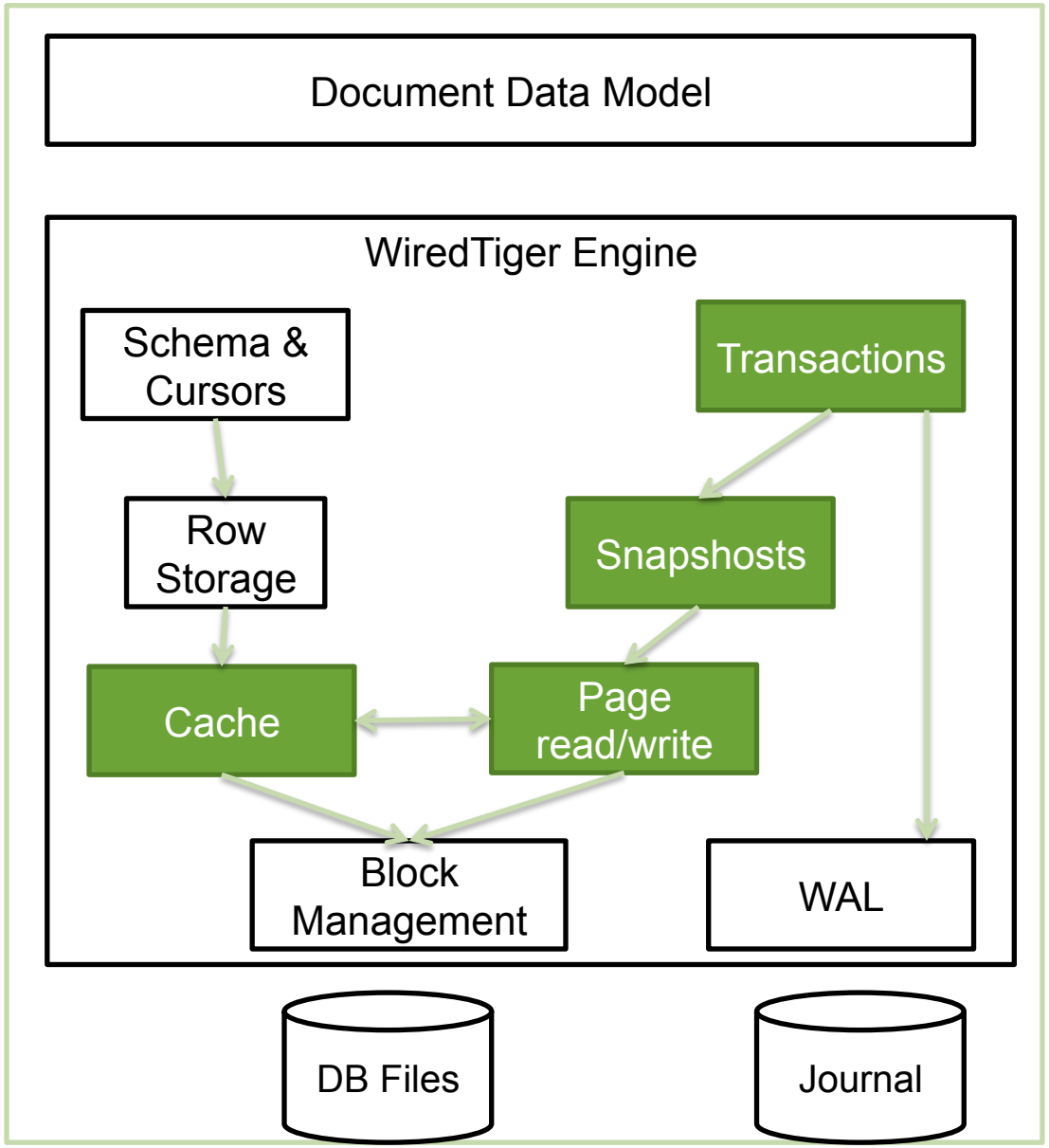
Collection

Index
{_id: 1}

Index
{name: 1}

Index
{message: "text"}

WiredTiger Transactions

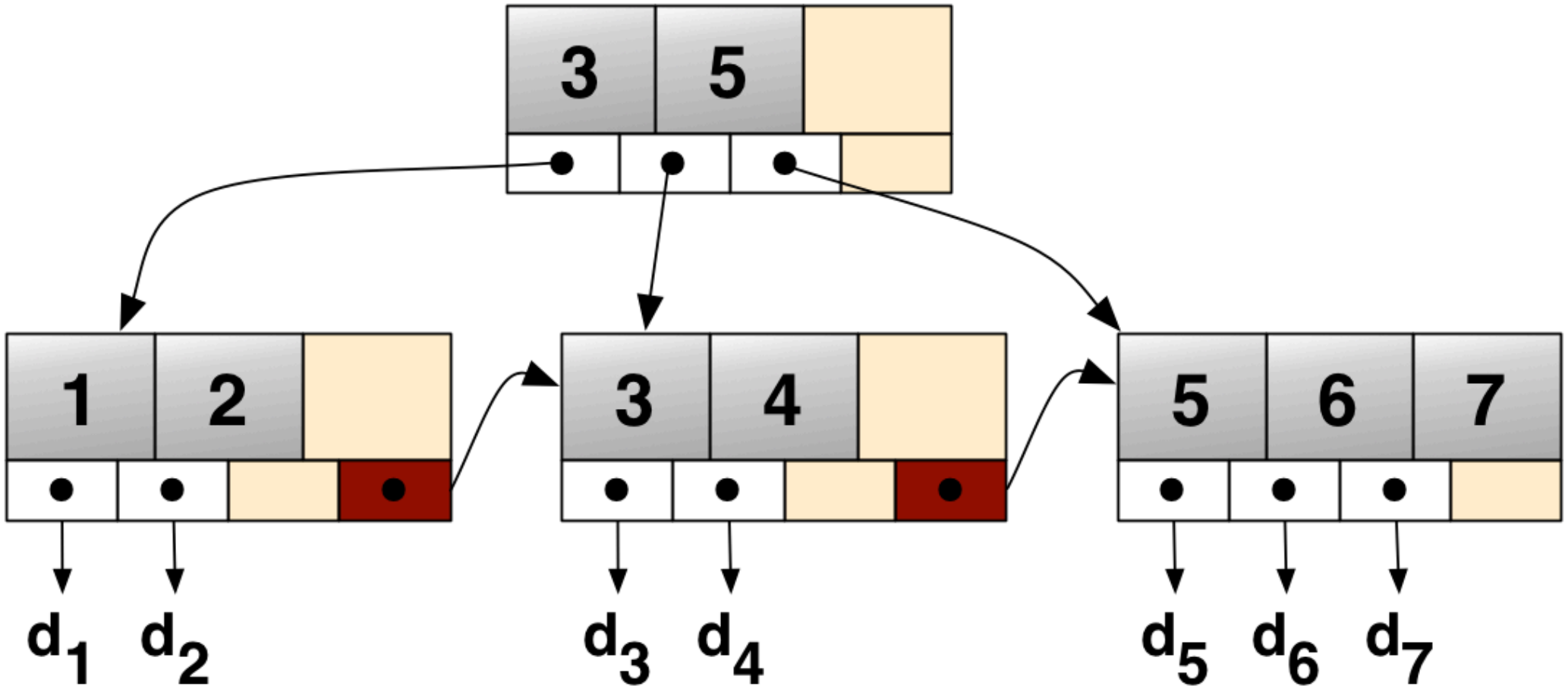


- Transaction per Session
  close, open, commit, rollback
- Ensures the atomicity/consistency of MongoDB operations
  - Updates / Writes documents
  - Updates indexes

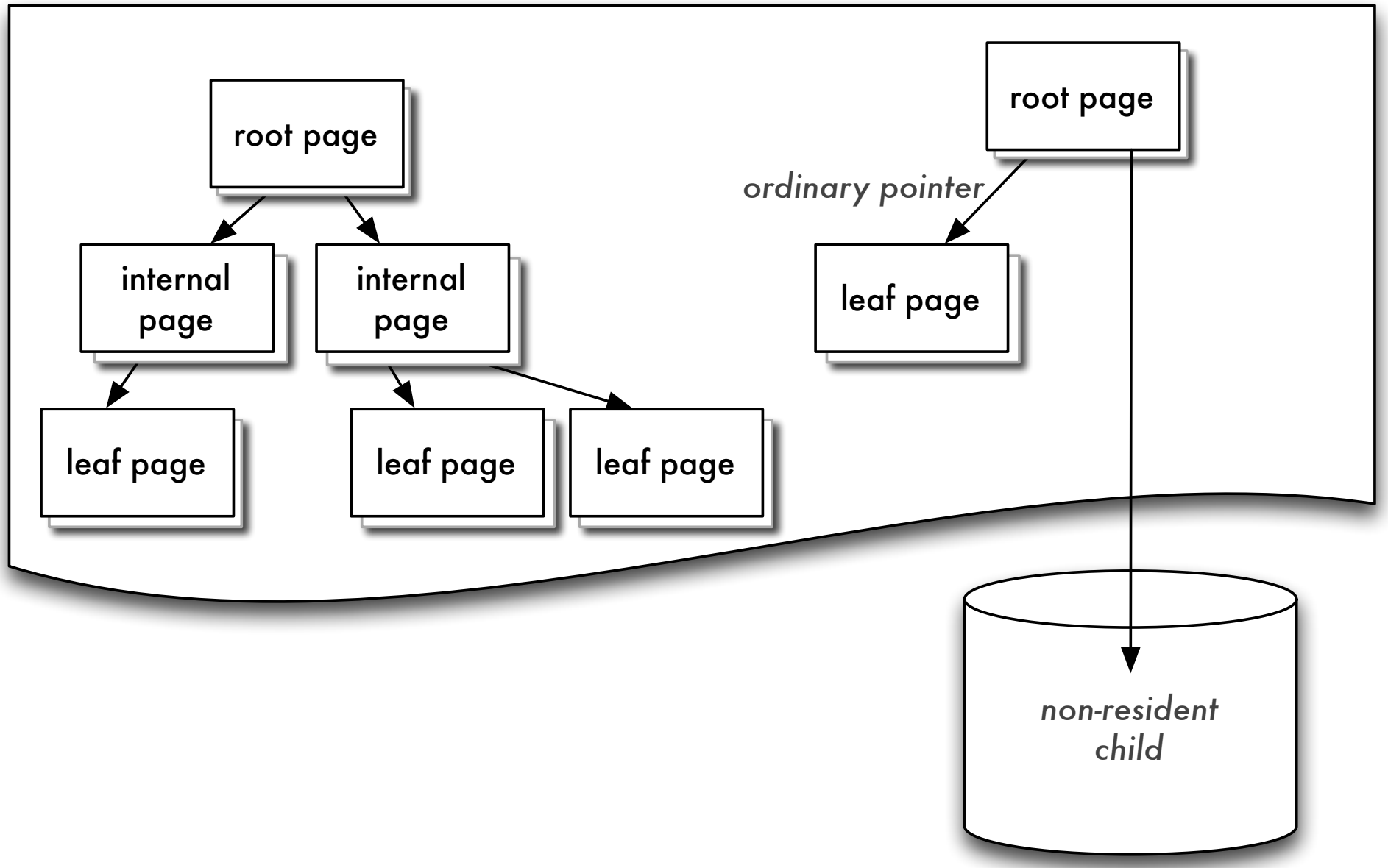
This is not multi-document transaction support!

In-memory Performance

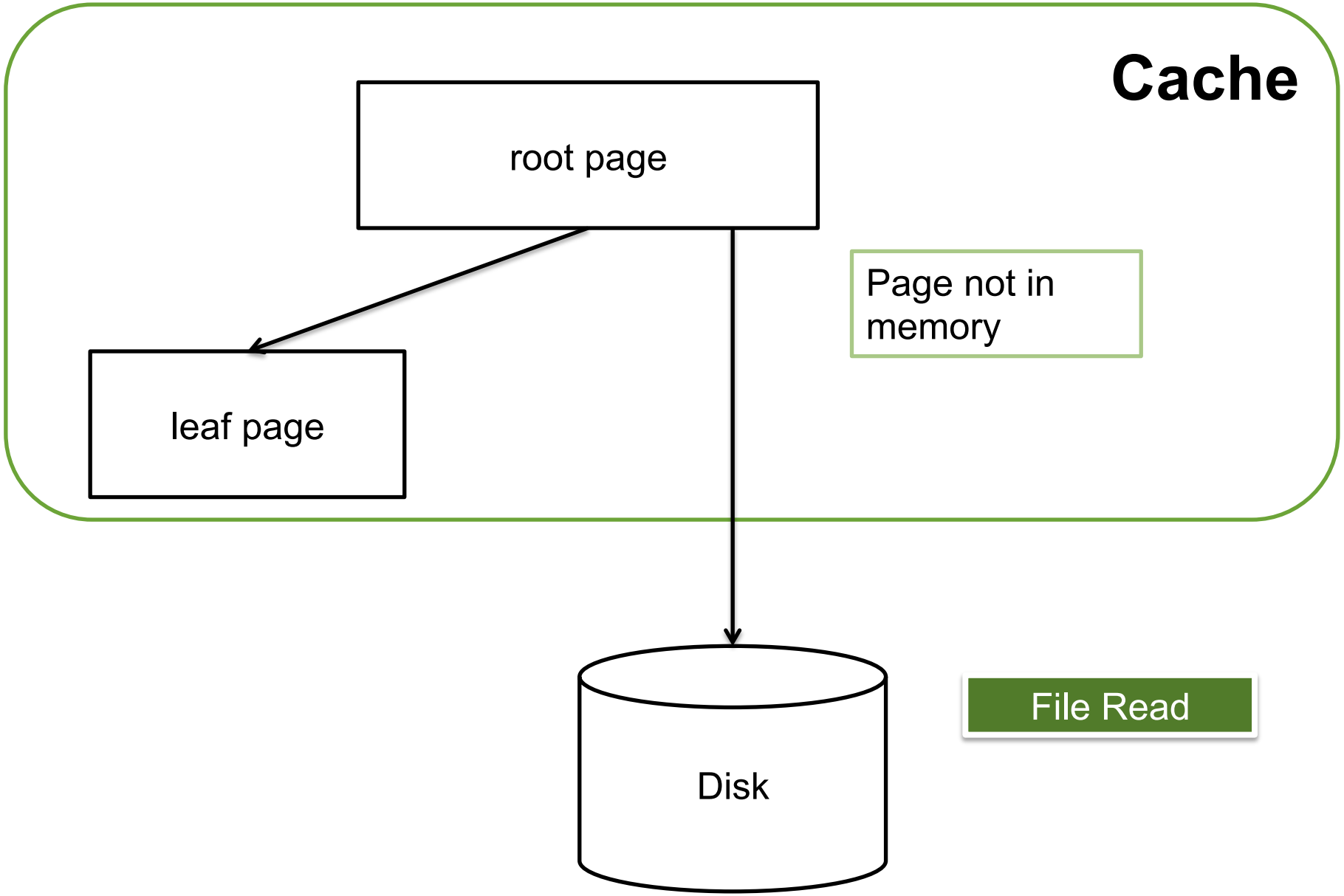
Traditional B+tree (ht wikipedia)



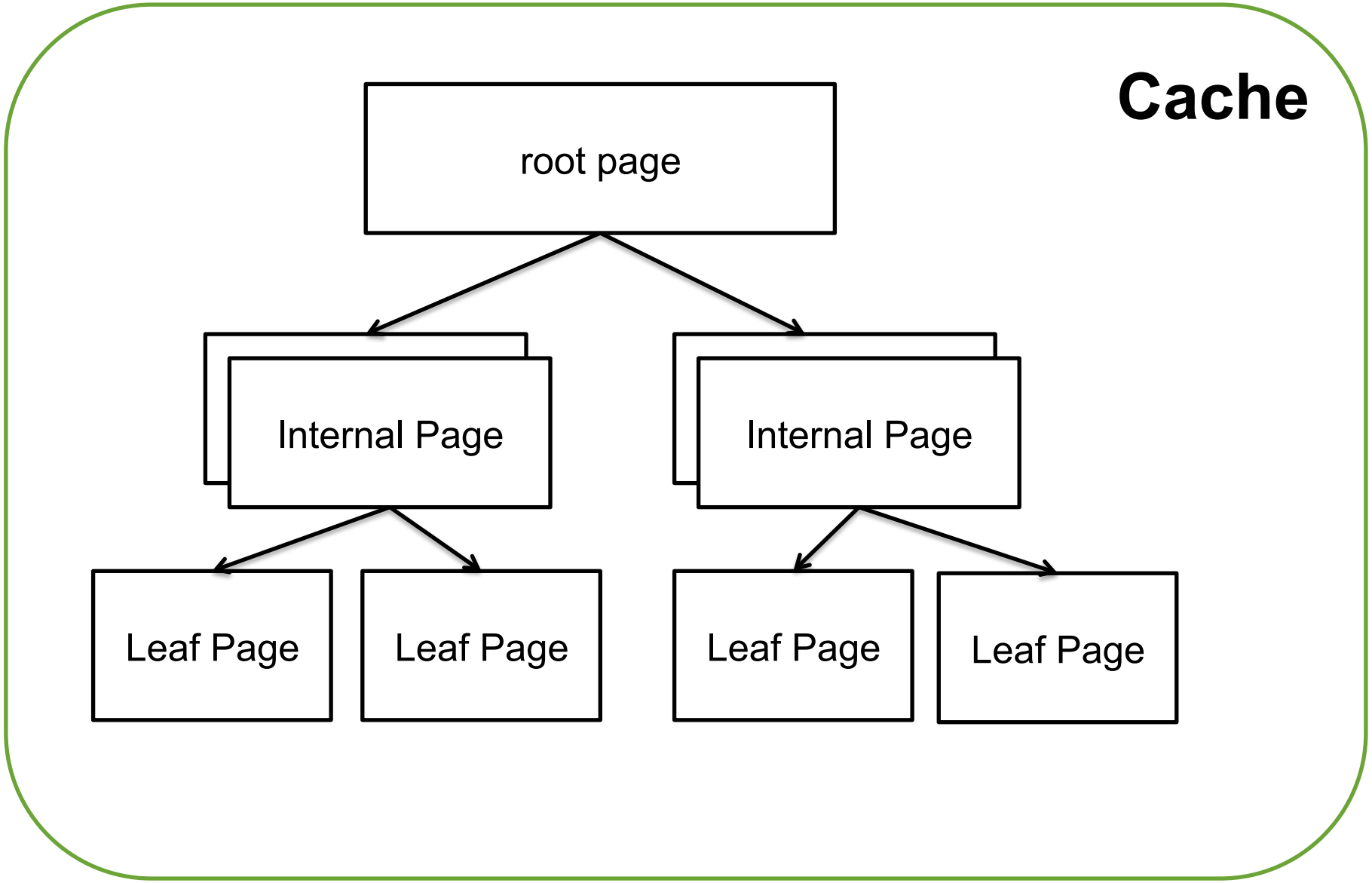
Trees in cache



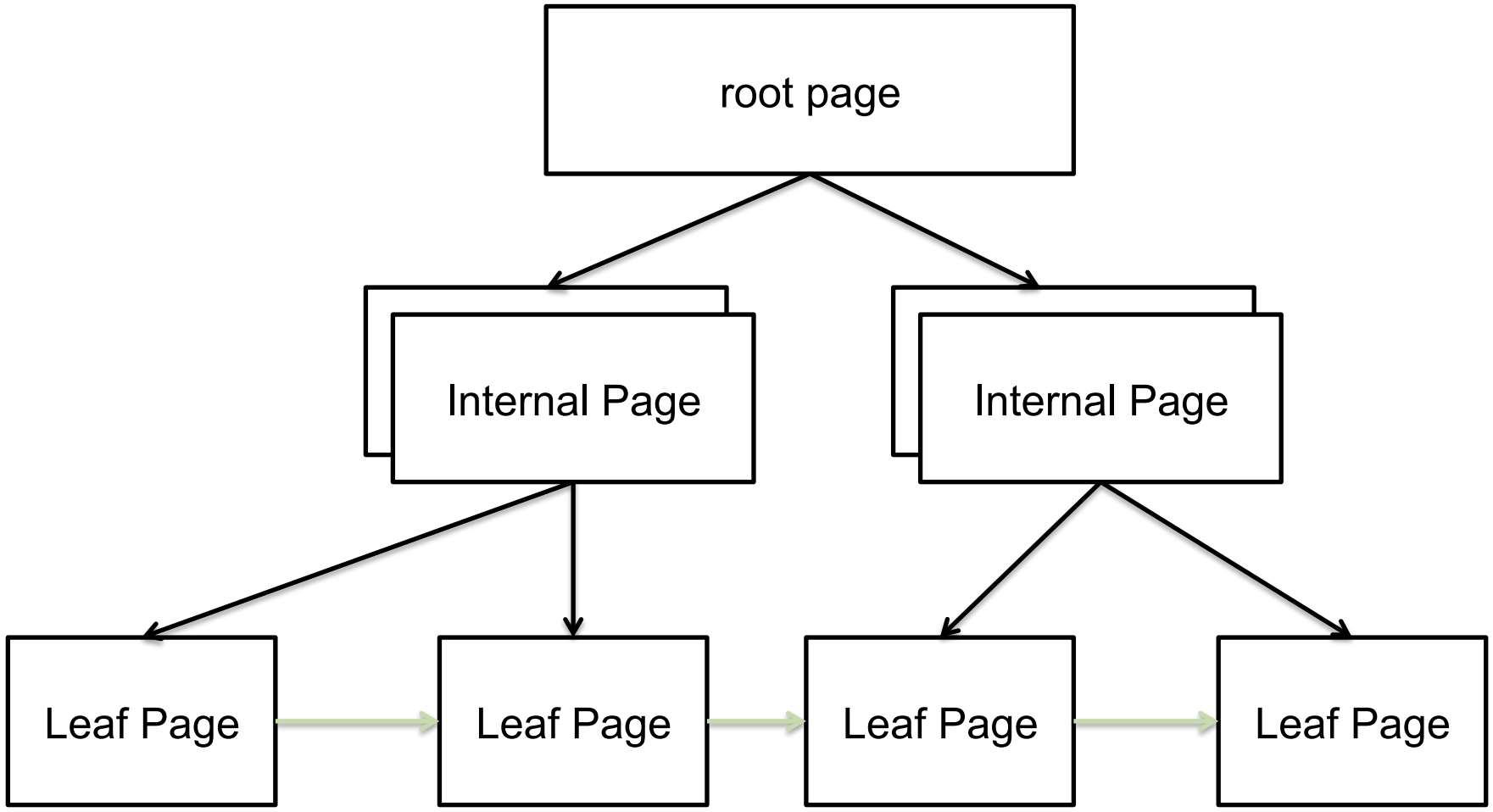
Page in Disk



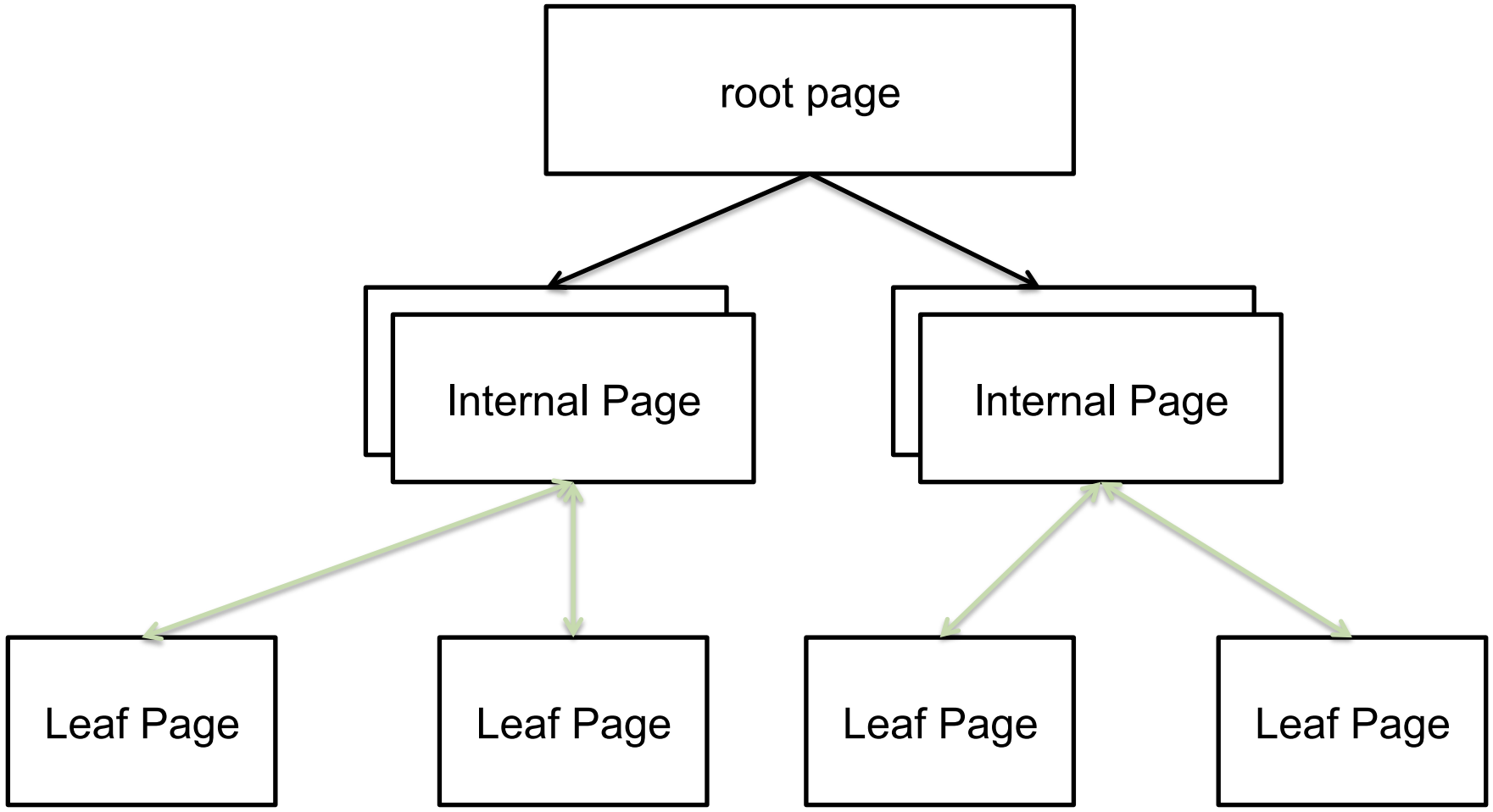
Page in Memory



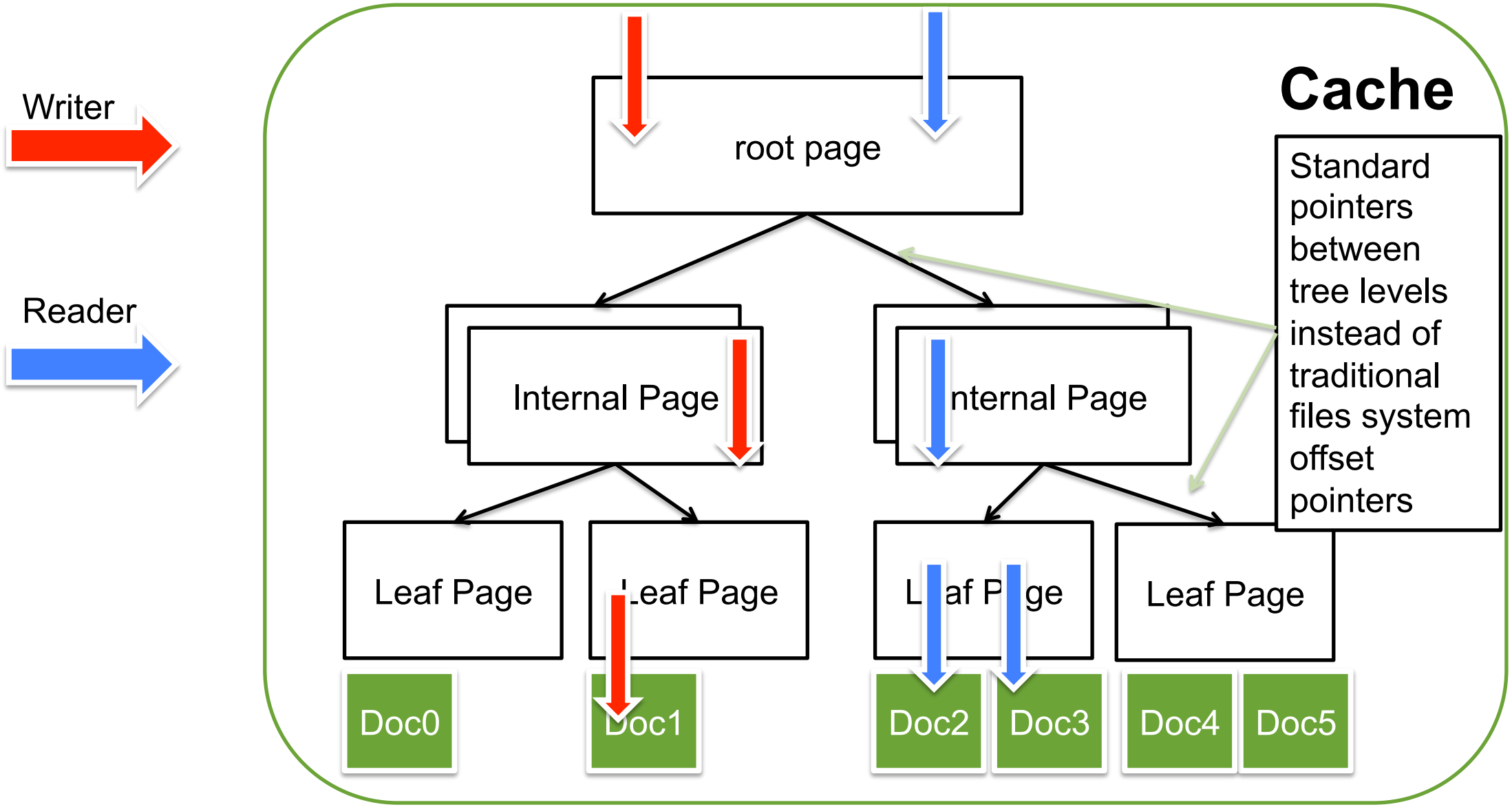
Traditional B-tree



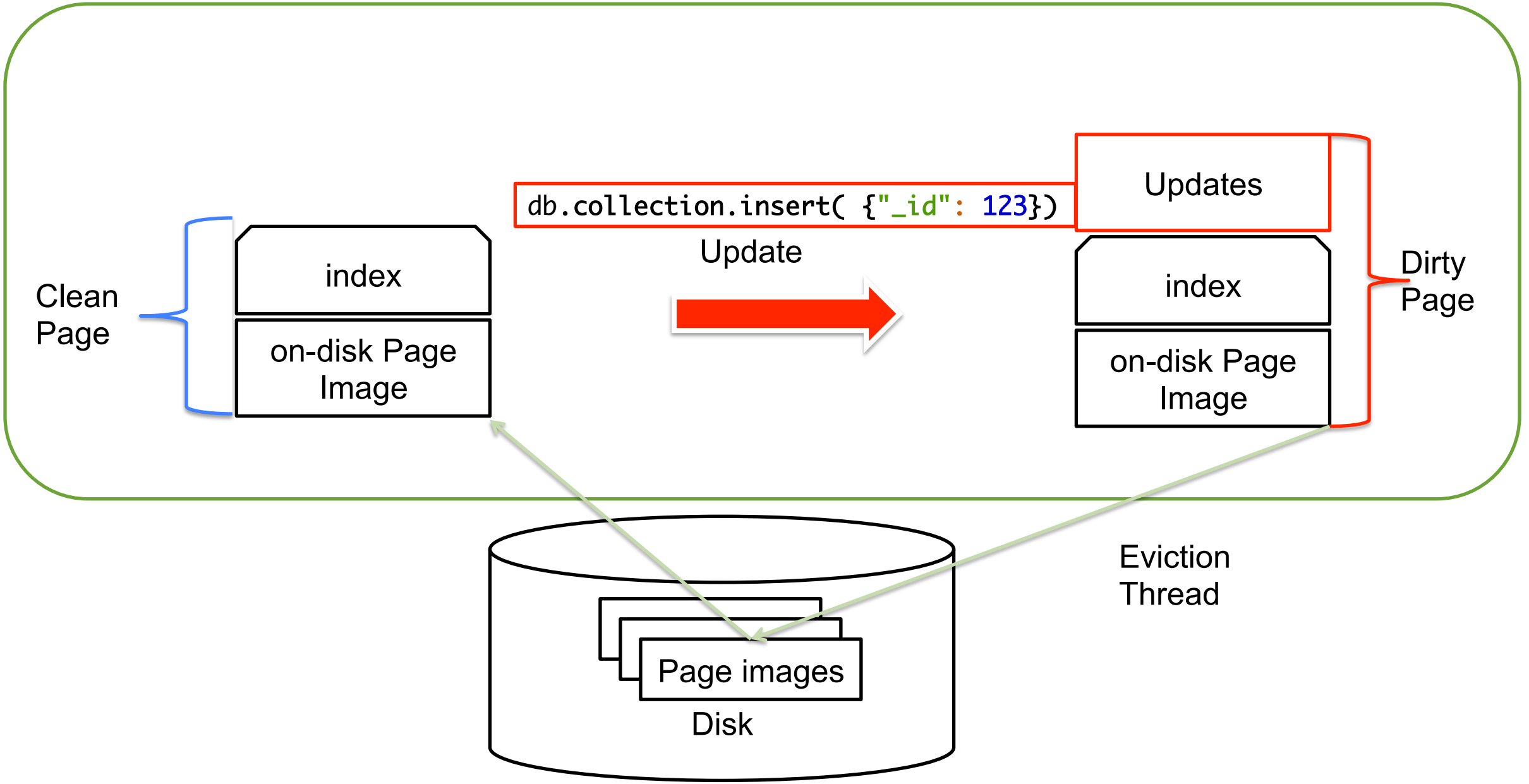
Deadlock Avoidance



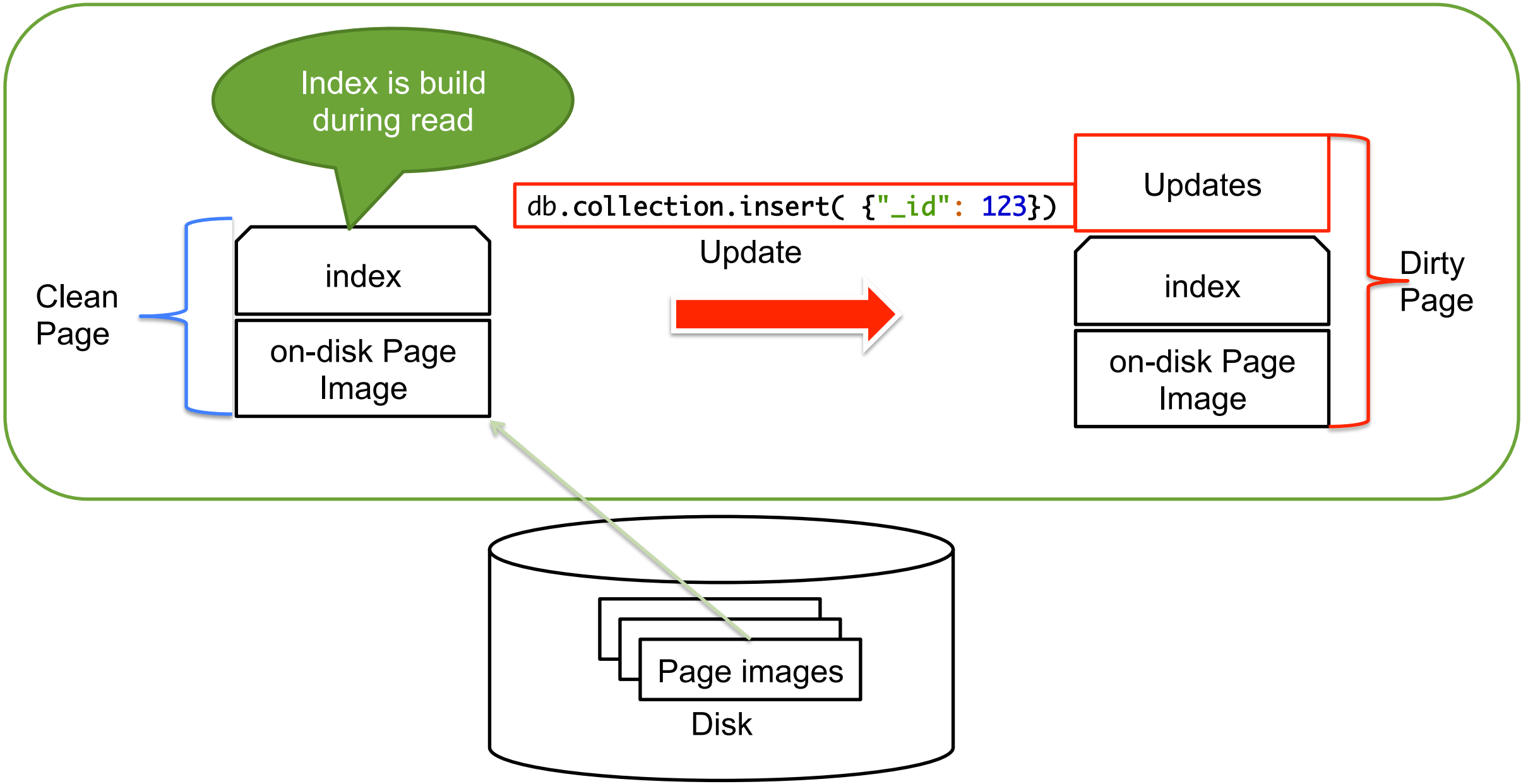
Page in Memory



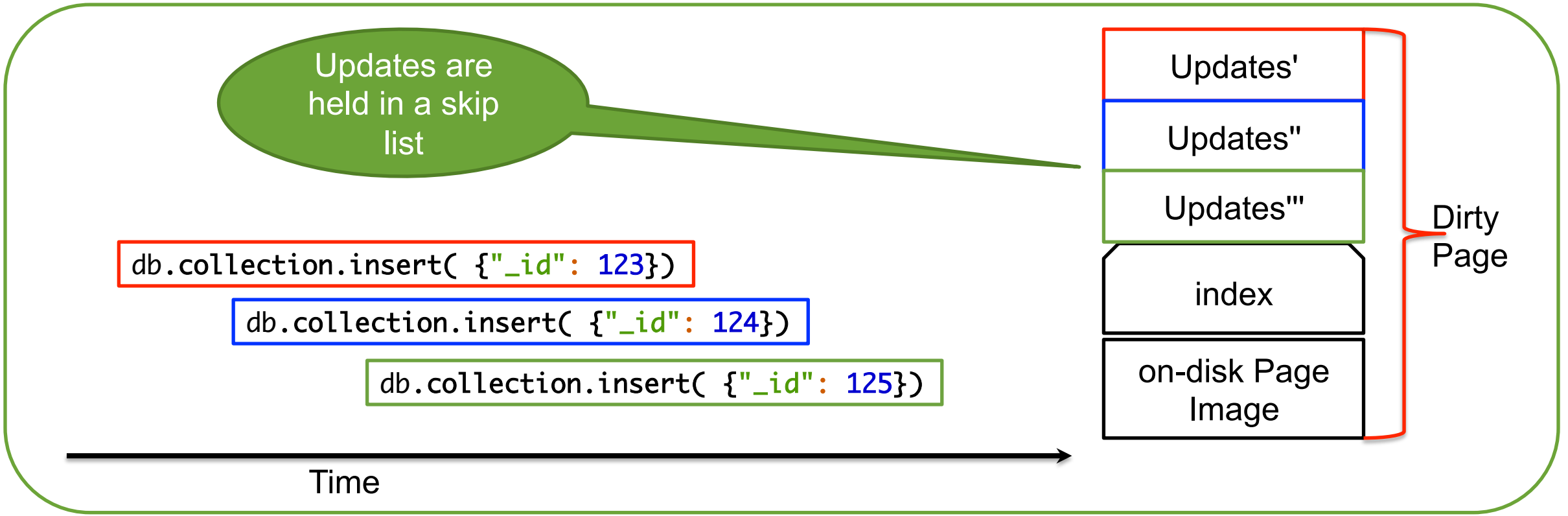
Page in Cache

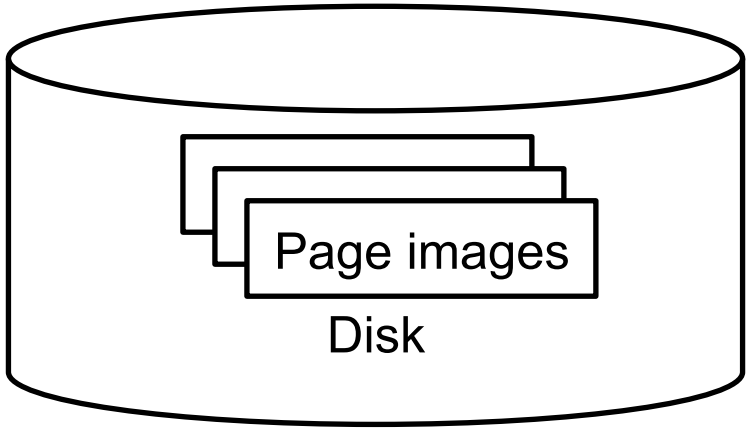


Page in Cache



Page in Cache

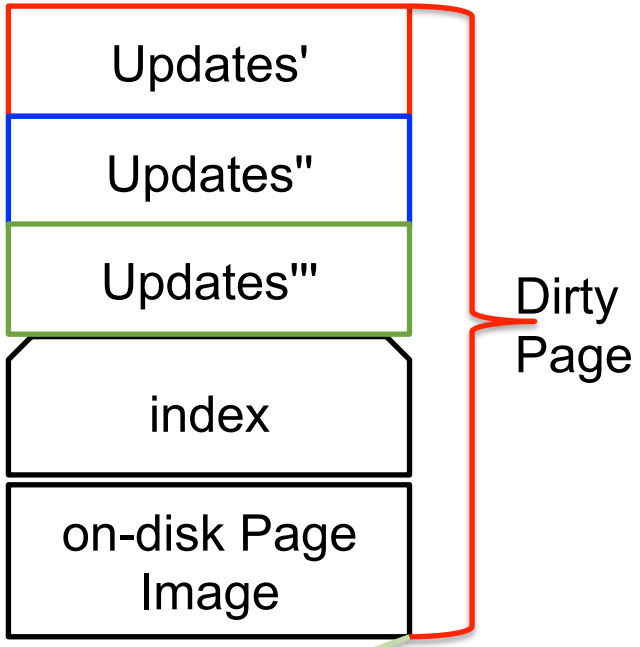


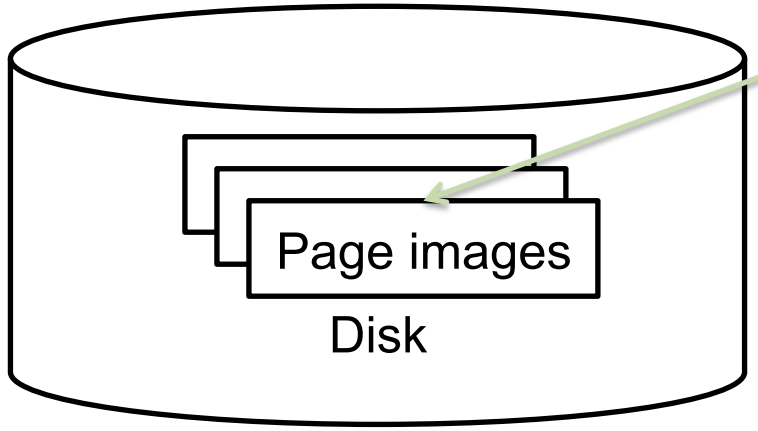


Reconciliation

Reconciliation

Cache full
After X operations
On a checkpoint





Eviction Thread

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

Takeaways

- Page and tree access is optimized
- Allows concurrent operations – updates skip lists
- Tuning options that you should be aware of:
  - storage.wiredTiger.engineConfig.**cacheSizeGB**
    - You can determine the space allocated for your cache
    - Makes your deployment and sizing more predictable

Wiredtiger cache! MongoDB uses more memory than just the Storage Engine needs!

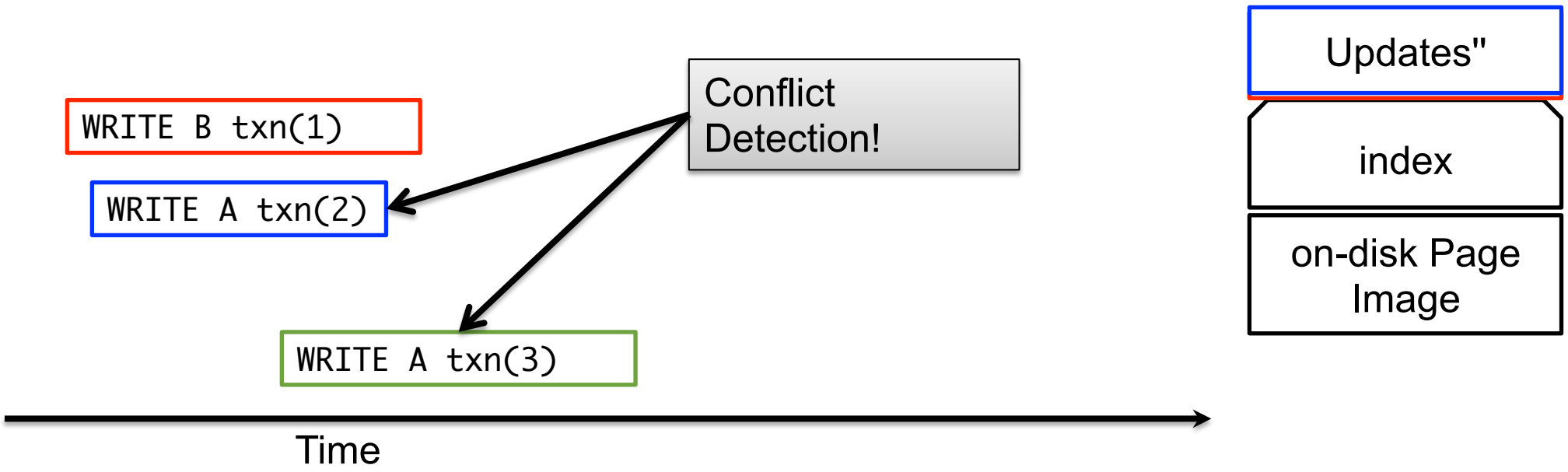
https://docs.mongodb.org/manual/reference/configuration-options/#storage.wiredTiger.engineConfig.cacheSizeGB

Document-level Concurrency

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)

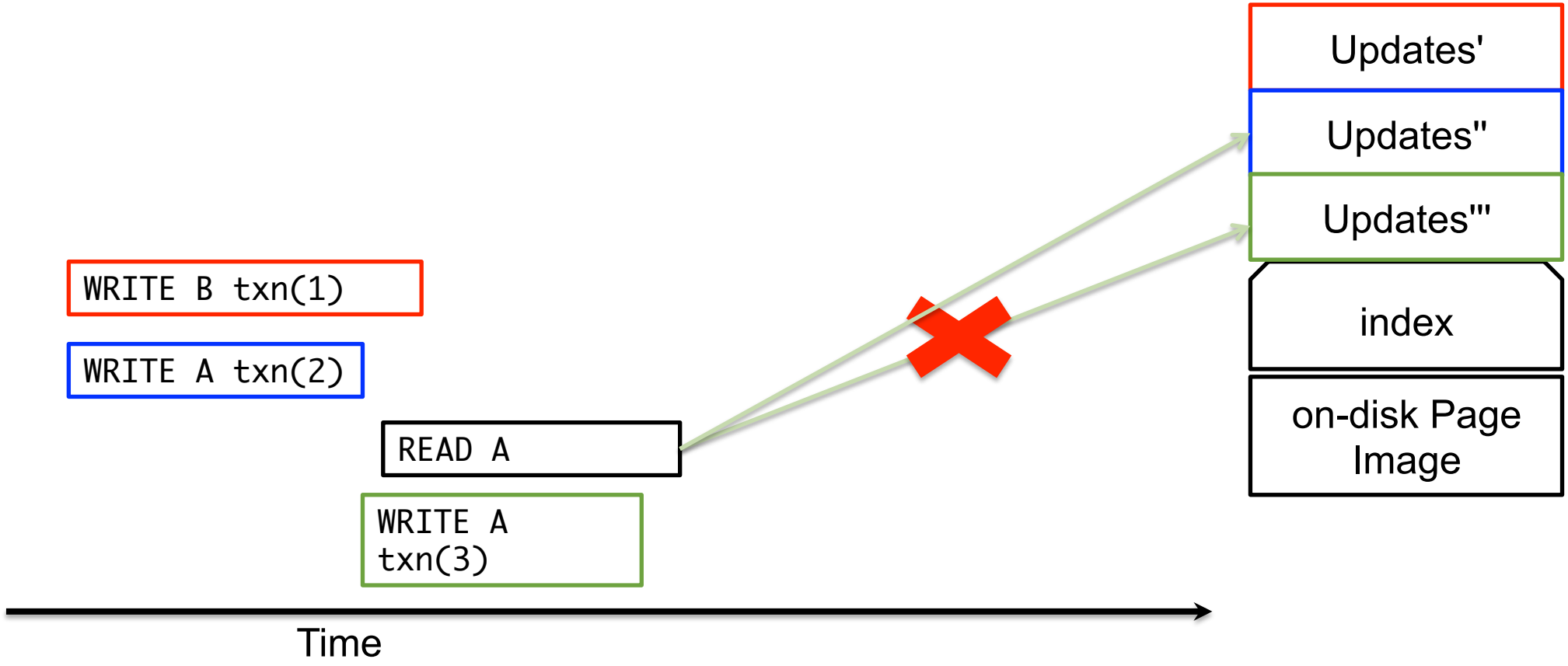
MVCC


WRITE B txn(1)
WRITE A txn(2)
WRITE A txn(3)
Conflict Detection!
Time
Updates"
index
on-disk Page Image

Multiversion Concurrency Control (MVCC)

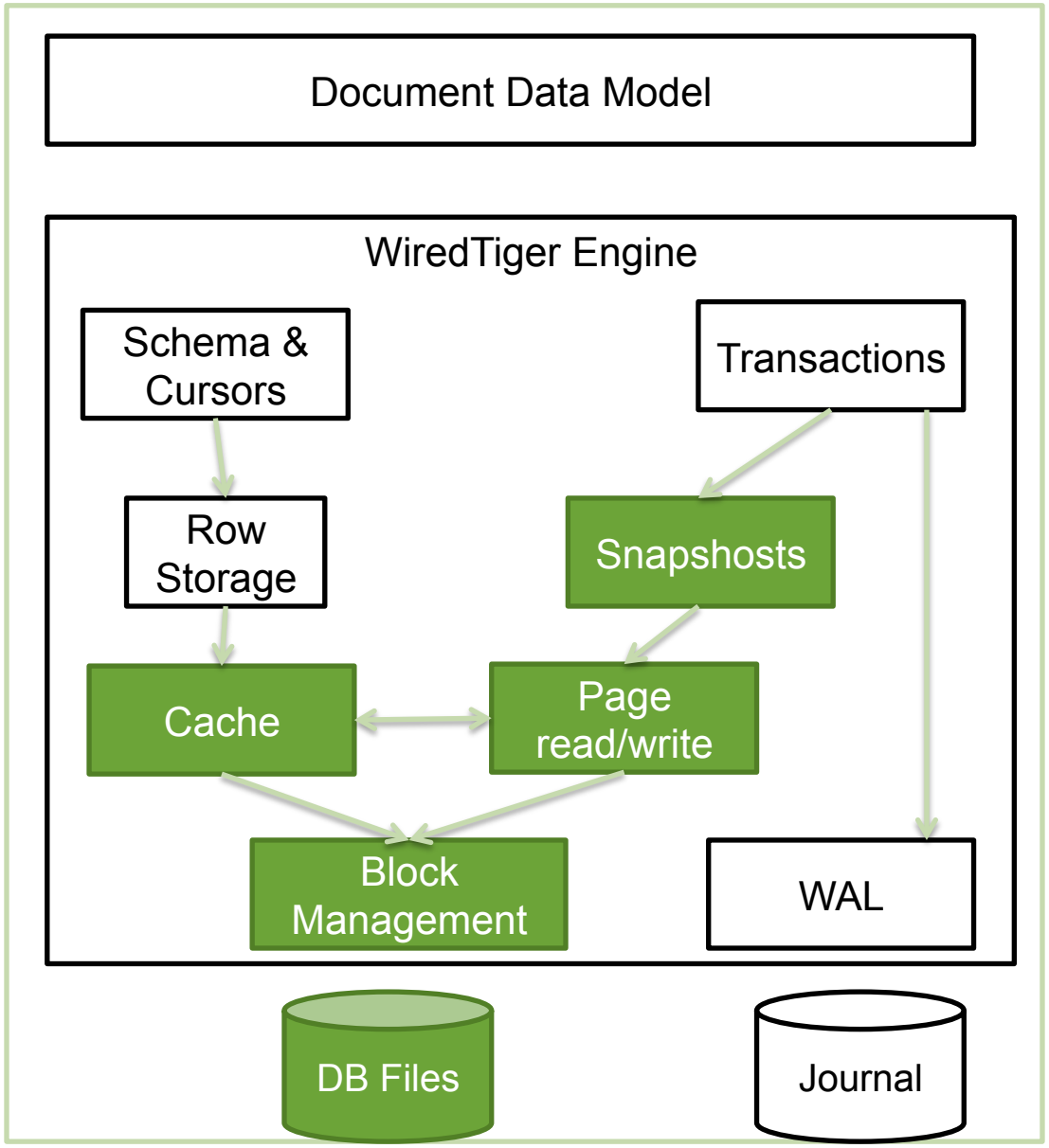
- Multiple versions of records kept in cache
- Readers see the committed version before the transaction started
  - MongoDB “yields” turn large operations into small transactions
- Writers can create new versions concurrent with readers
- Concurrent updates to a single record cause write conflicts
 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

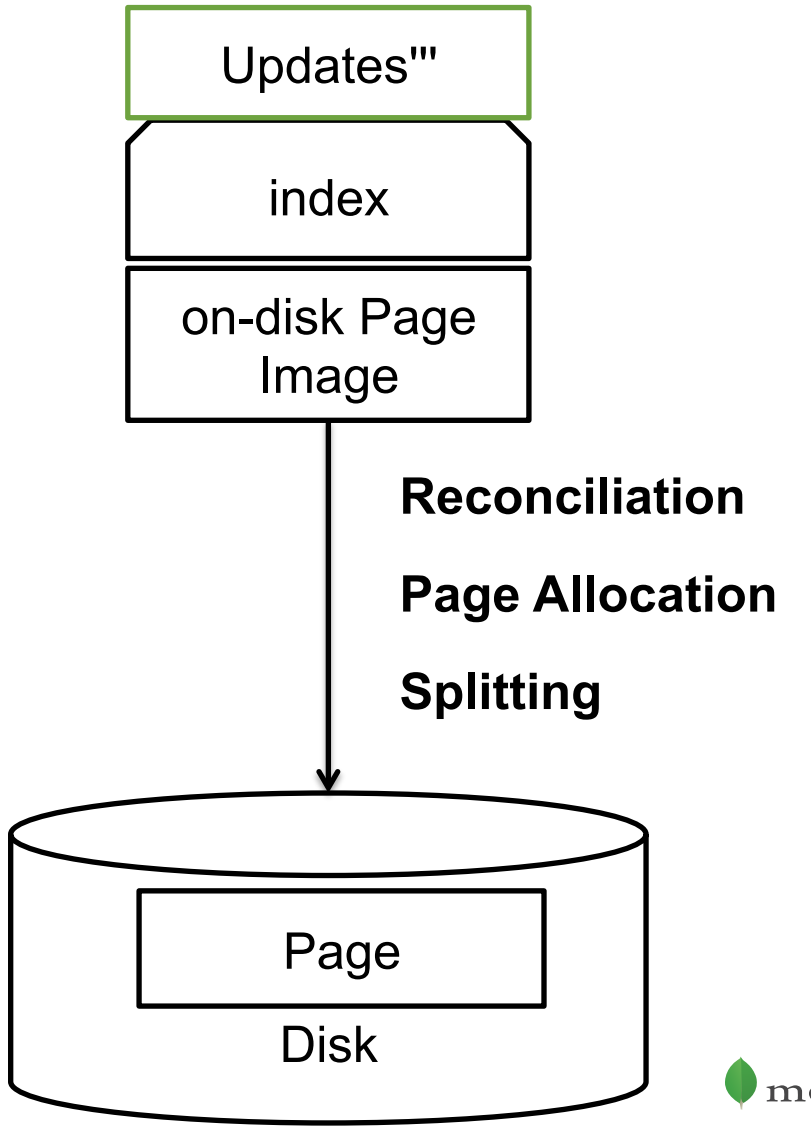
MVCC


Updates'
Updates''
Updates'''
index
on-disk Page Image
WRITE B txn(1)
WRITE A txn(2)
READ A
WRITE A txn(3)
Time

Compression

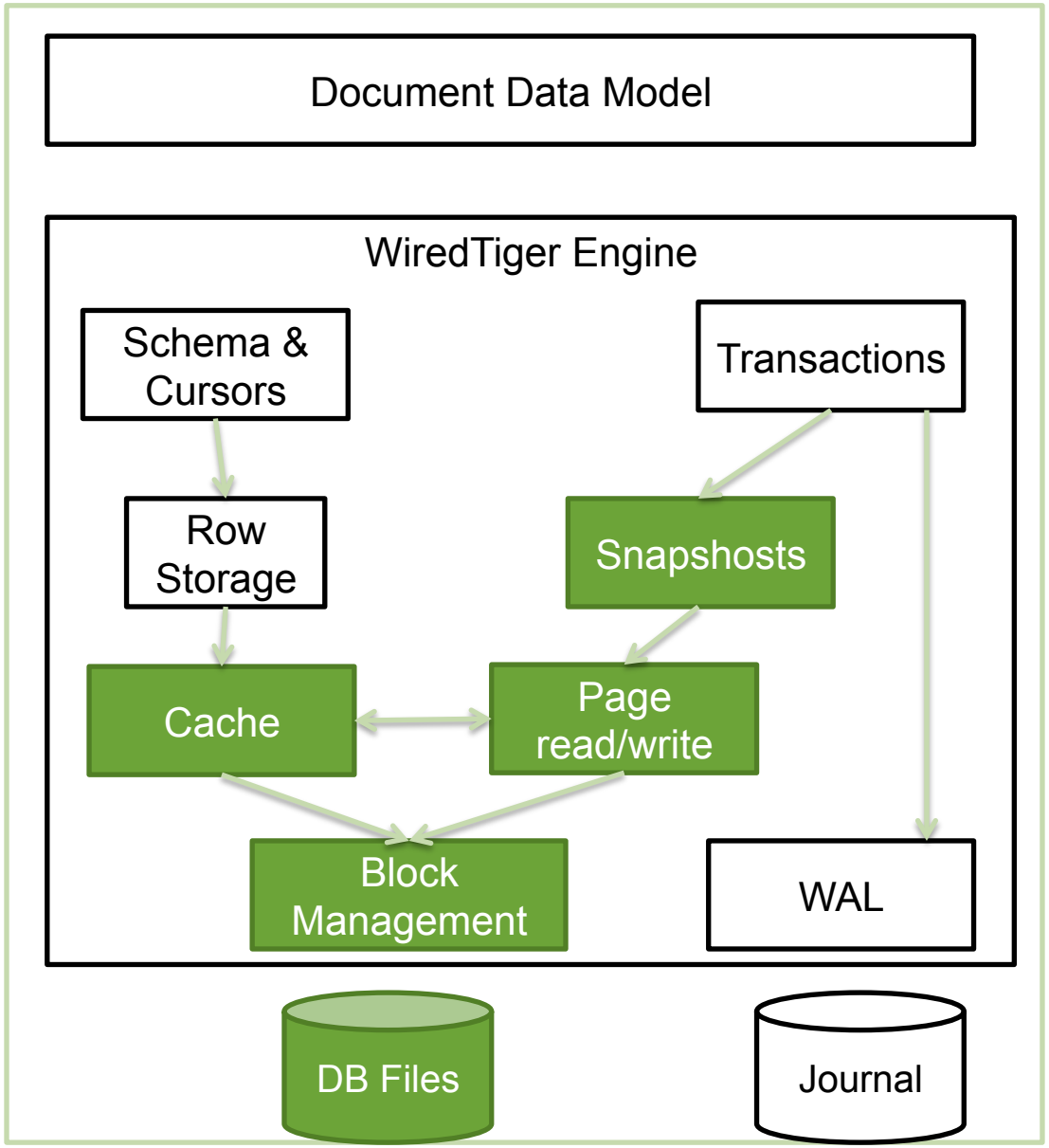
WiredTiger Page IO

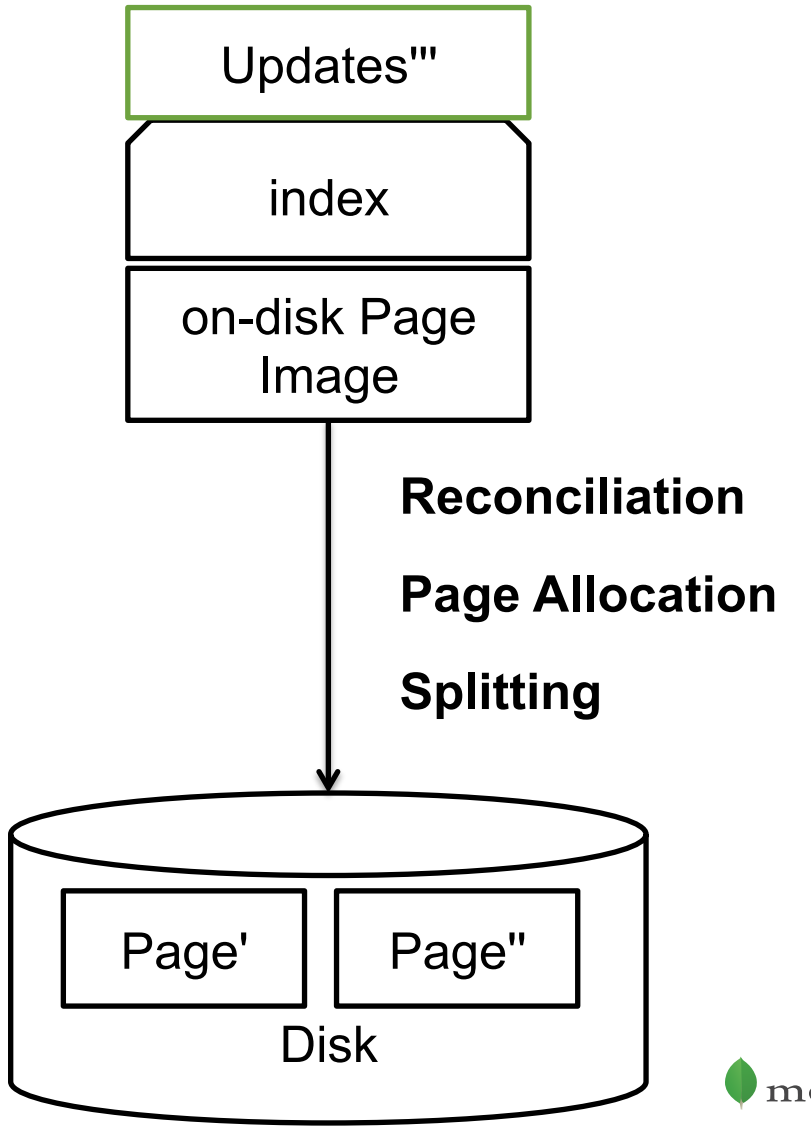




mongoDB

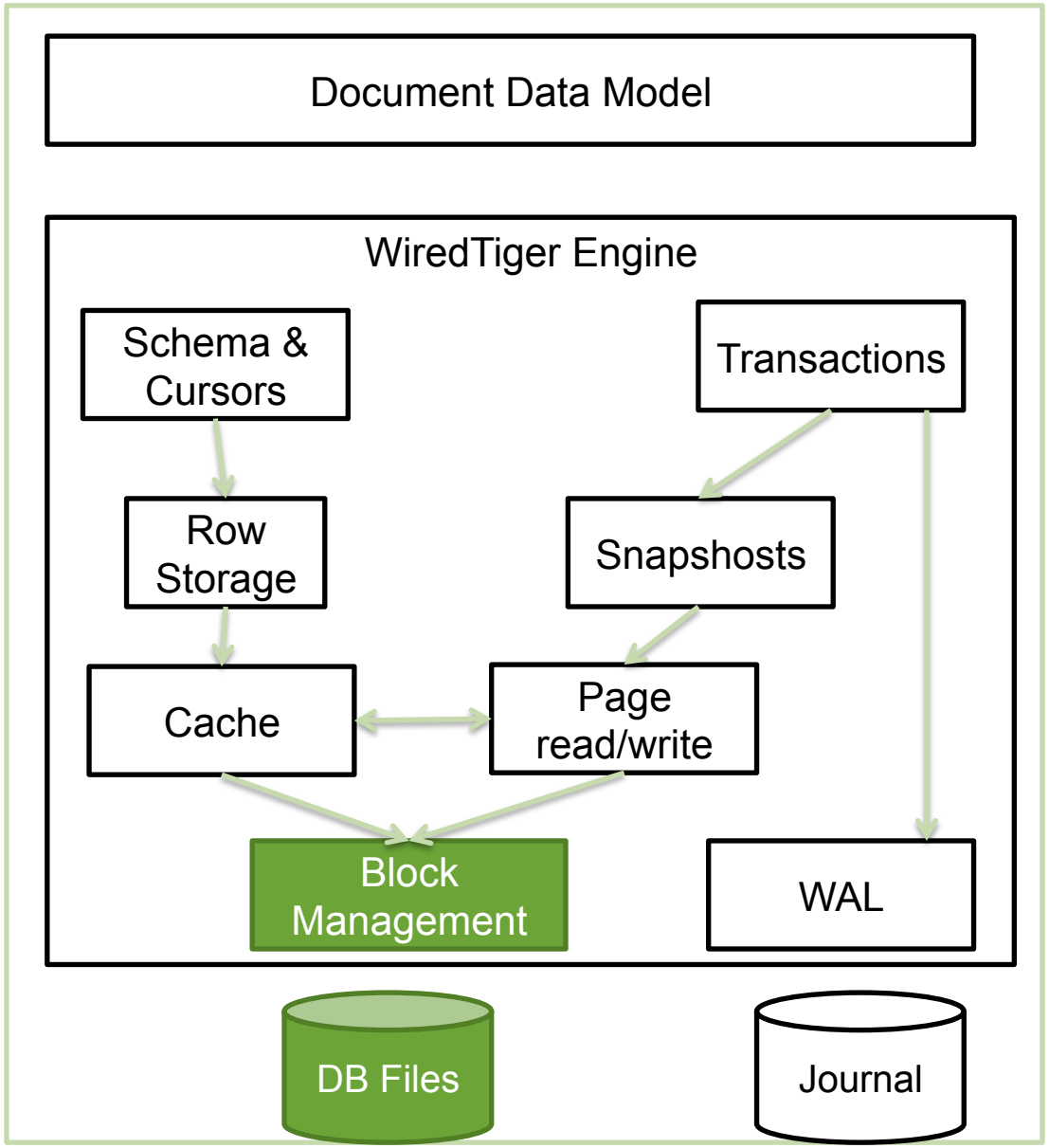
WiredTiger Page IO





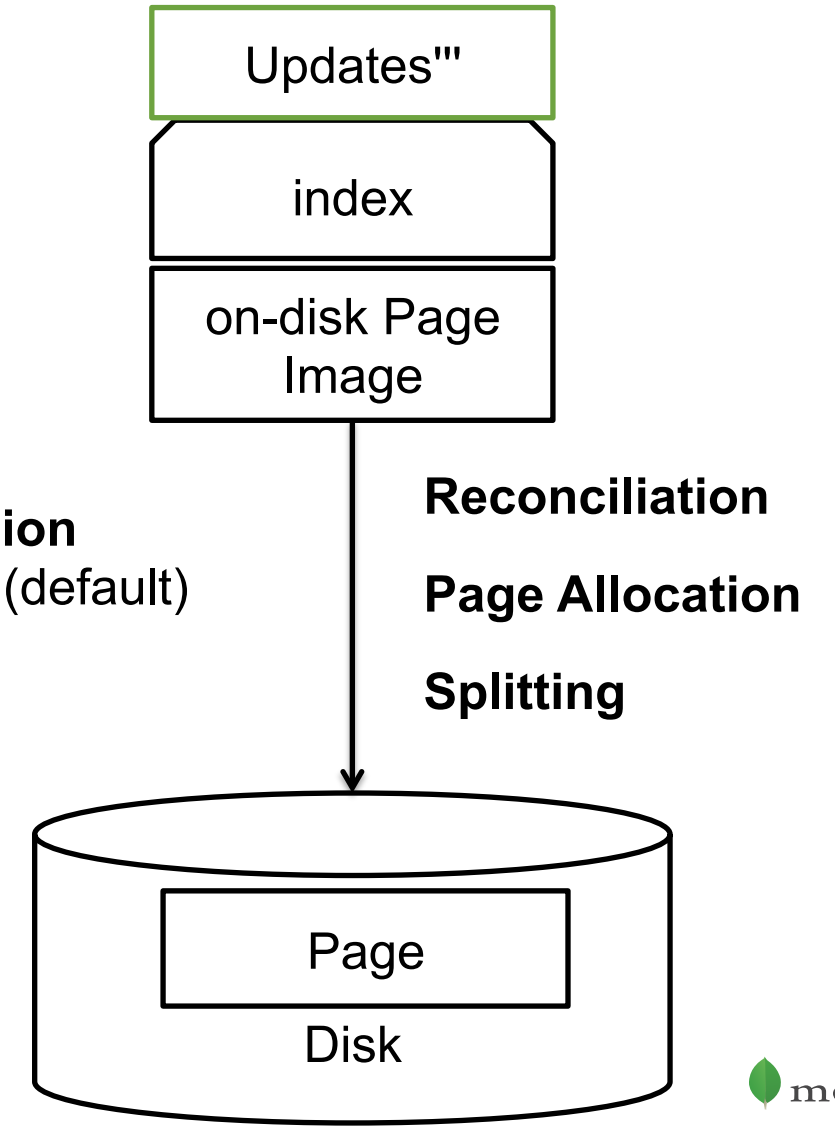
mongoDB

WiredTiger Page IO

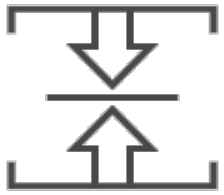


Compression

- snappy (default)
- zlib
- none



mongoDB



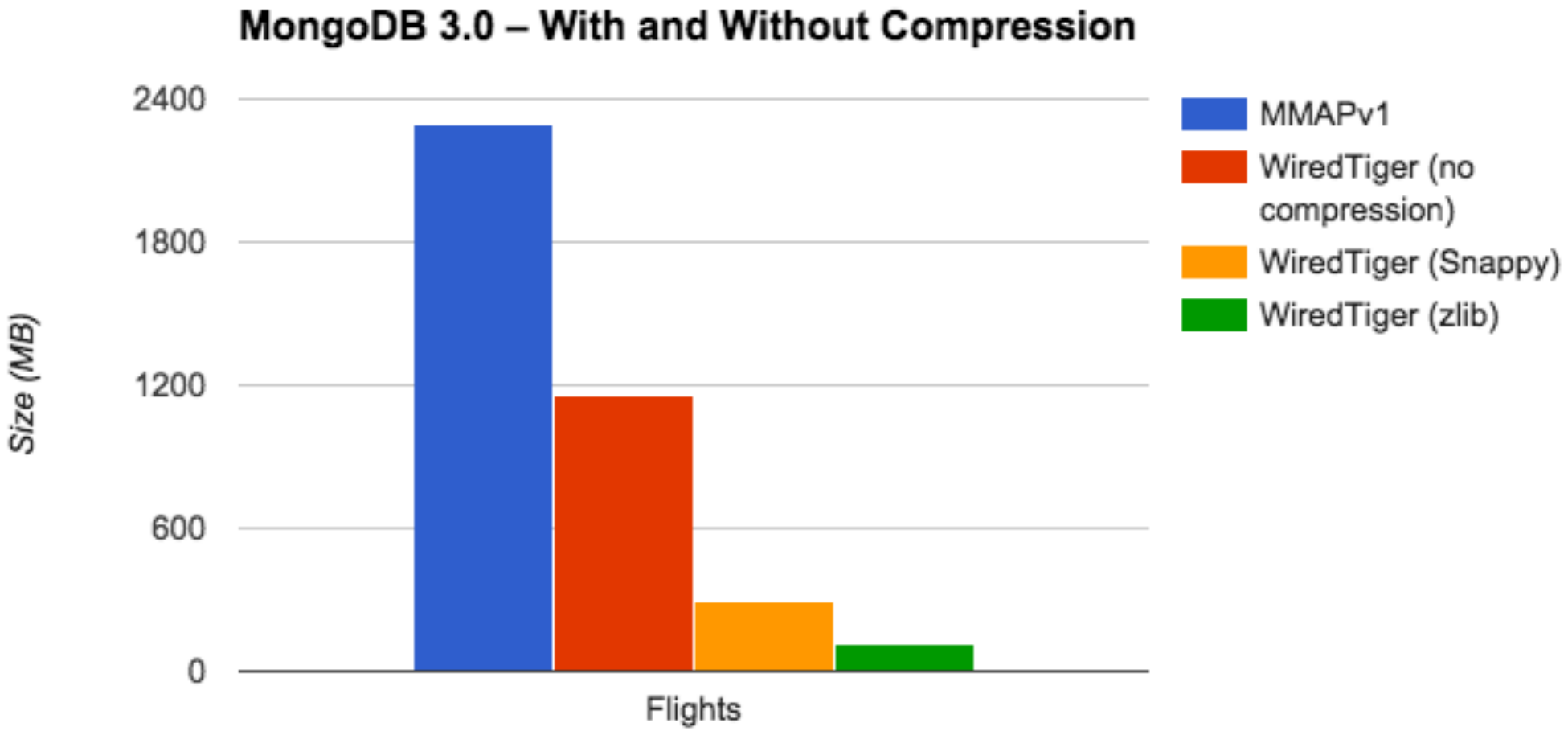
Compression

- 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

Checksums

- A checksum is stored with every uncompressed page
- Checksums are validated during page read
  - detects filesystem corruption, random bitflips
- WiredTiger stores the checksum with the page address (typically in a parent page)
  - extra safety against reading a stale page image

Compression in Action



(Flights database, ht Asya)

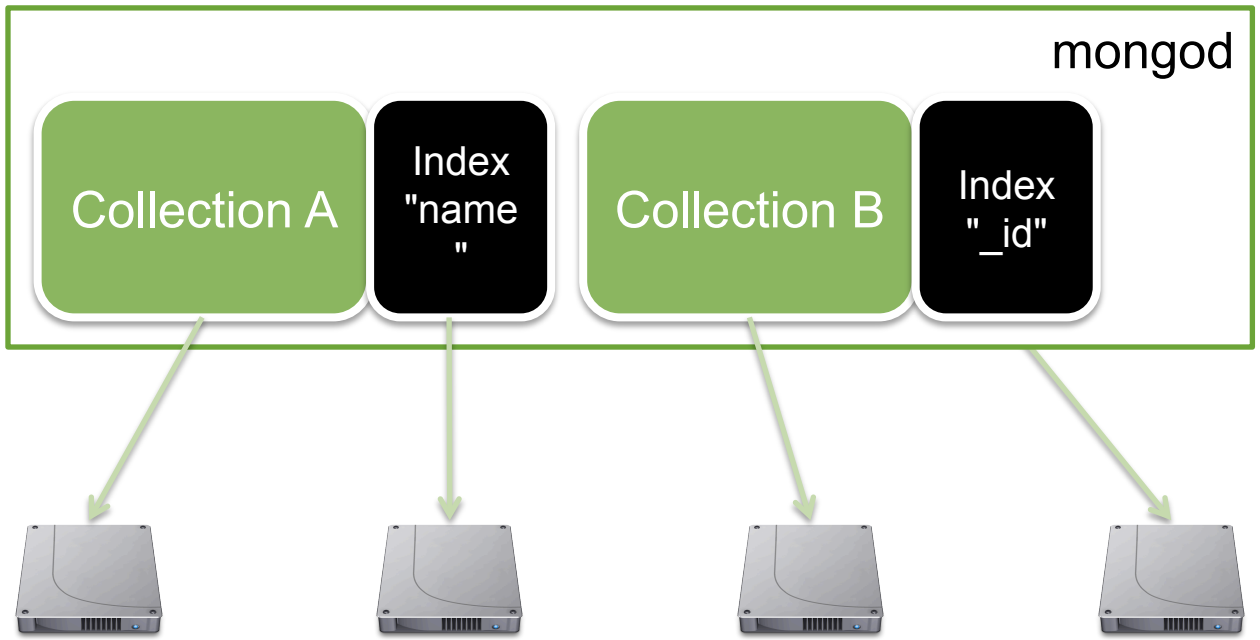
Takeaway

- Main feature that impacts the vast majority of MongoDB projects / use cases
- CPU bound
- Different algorithms for different workloads
  - Collection level compression tuning
- Smaller Data Faster IO
  - Not only improves the disk space footprint
  - also IO
  - and index traversing

Other Gems of WiredTiger

Index per Directory

- Use different drives for collections and indexes
  - Parallelization of write operations
  - MMAPv1 only allows *directoryPerDatabase*



Consistency without Journaling

- MMAPv1 uses write-ahead log (journal) to guarantee consistency
- WT doesn't have this need: no in-place updates
  - Write-ahead log committed at checkpoints and with j:true
  - Better for insert-heavy workloads
  - By default journaling is enabled!
- Replication guarantees the durability

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

mongoDB

What's next for WiredTiger?

- Tune for (many) more workloads
  - avoid stalls during checkpoints with 100GB+ caches
  - make capped collections (including oplog) more efficient
- Adding encryption
- More advanced transactional semantics in the storage engine API

Updates and Upgrades

- Can not
  - Can't copy database files
  - Can't just restart w/ same dbpath
- Yes we can!
  - Initial sync from replica set works perfectly!
  - mongodump/restore
- Rolling upgrade of replica set to WT:
  - Shutdown secondary
  - Delete dbpath
  - Relaunch w/ --storageEngine=wiredTiger
  - Wait for resync
  - Rollover

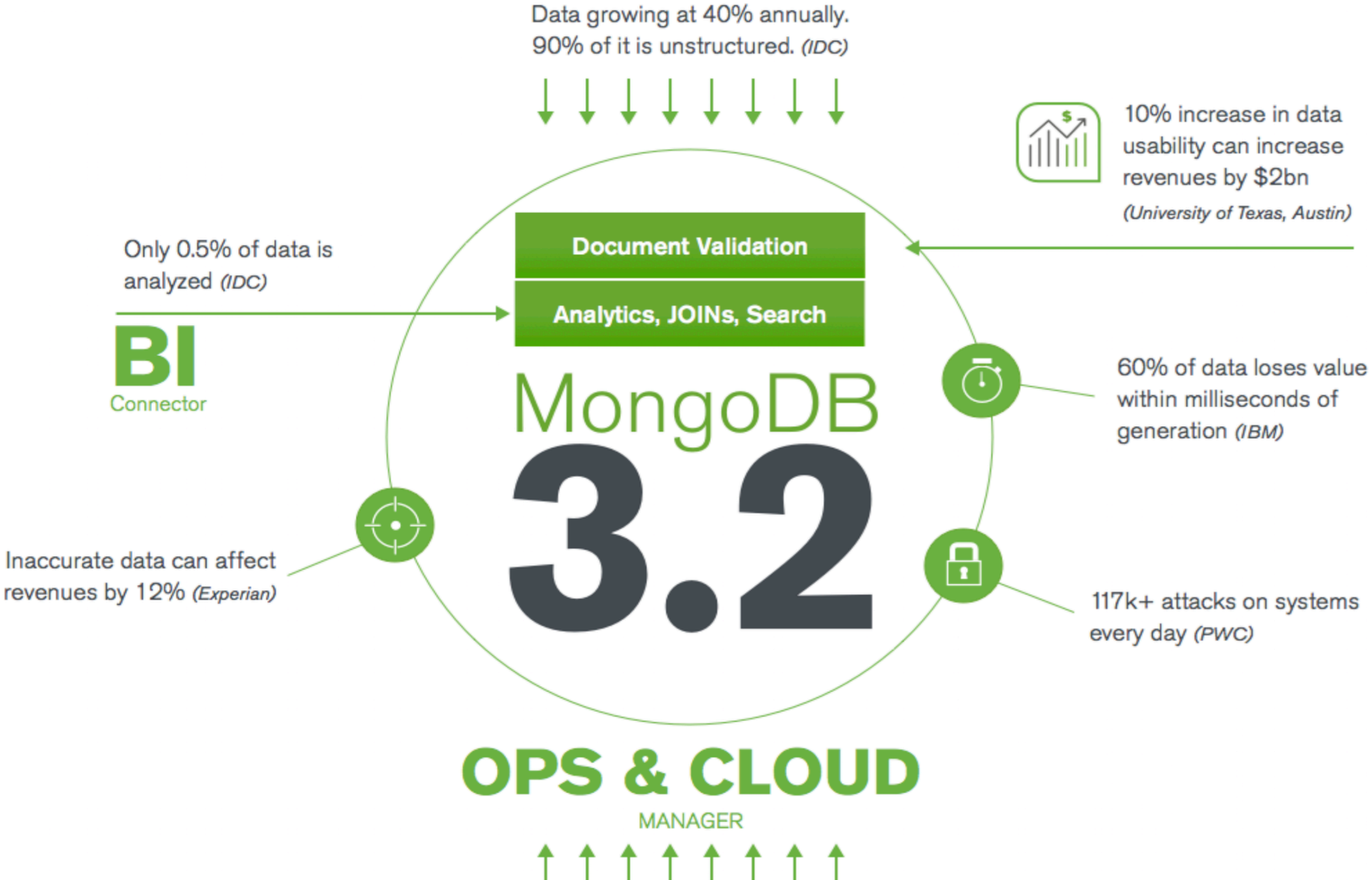
Summary

https://tingbudongchine.files.wordpress.com/2012/08/lemonde1.jpeg

MongoDB 3.0

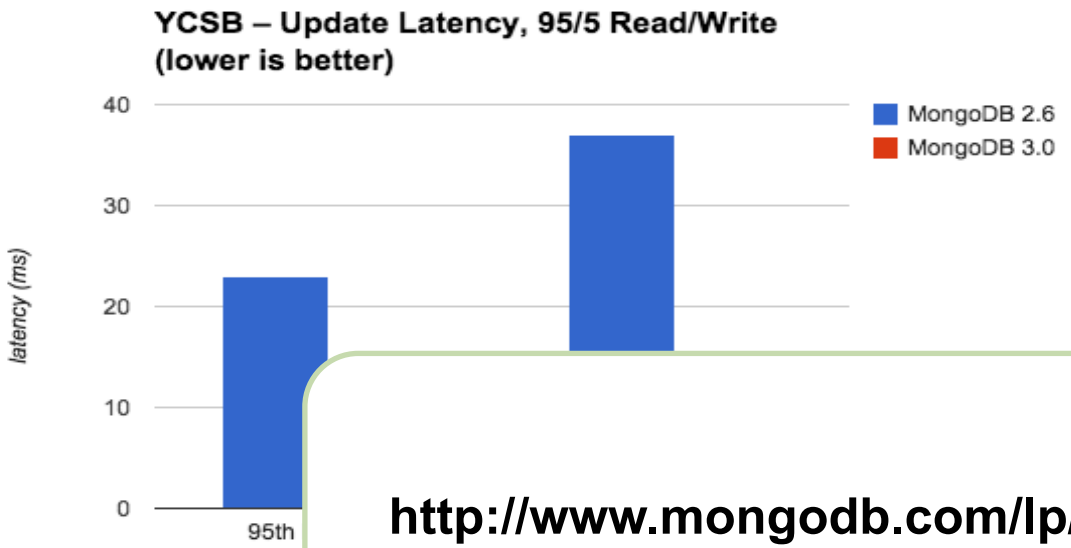
- Pluggable Storage Engine API
- Storage Engines
- Large Replica Sets
- Big Polygon
- Security Enhancements – SCRAM
- Audit Trail
- Simplified Operations – Ops Manager
- Tools Rewrite

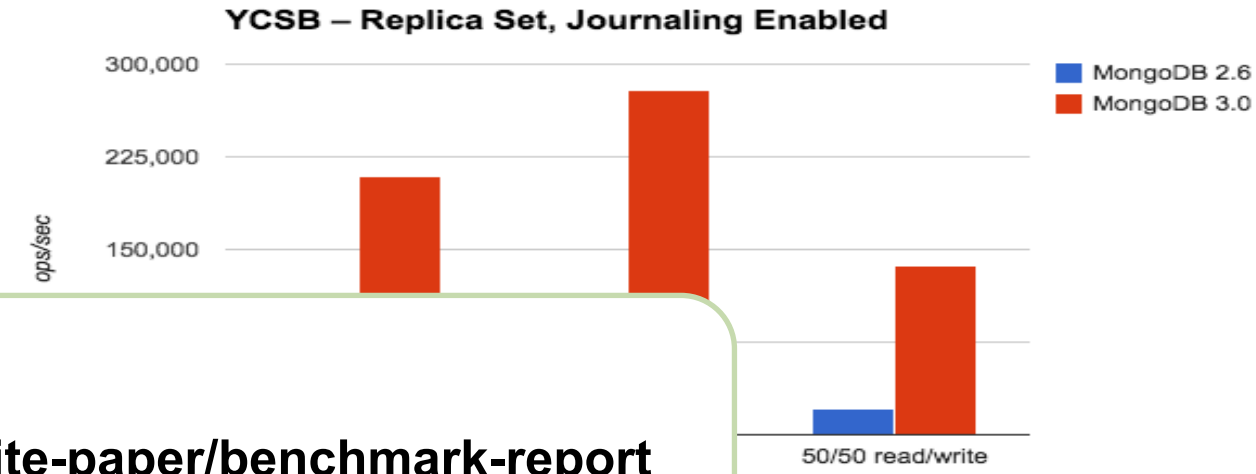
3.2 is coming!



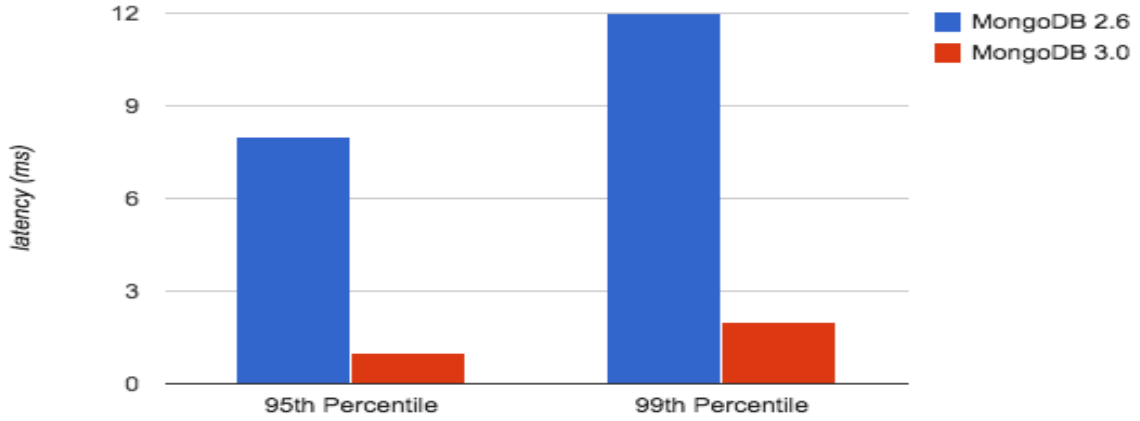
ongoDB

Performance!





http://www.mongodb.com/lp/white-paper/benchmark-report



mongoDB

Obrigado!

Norberto Leite
Technical Evangelist
norberto@mongodb.com
@nleite

http://cl.jroo.me/z3/v/D/C/e/a.baa-Too-many-bicycles-on-the-van.jpg