mongoDB®

FOR GIANT IDEAS

Beyond The Basics : Part 1
Storage Engines

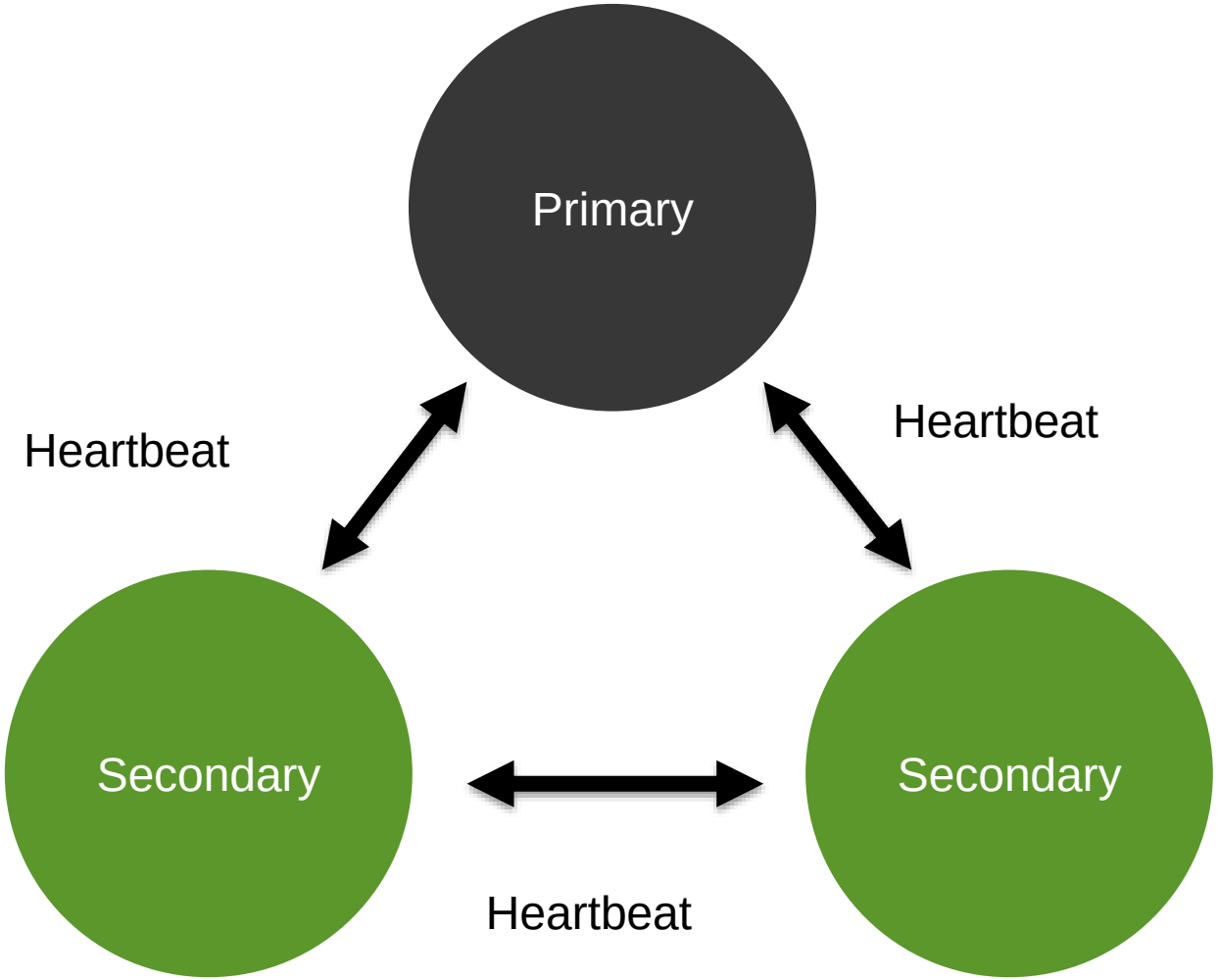
Joe Drumgoole
Director of Developer Advocacy, EMEA
@jdrumgoole

v1.2.1

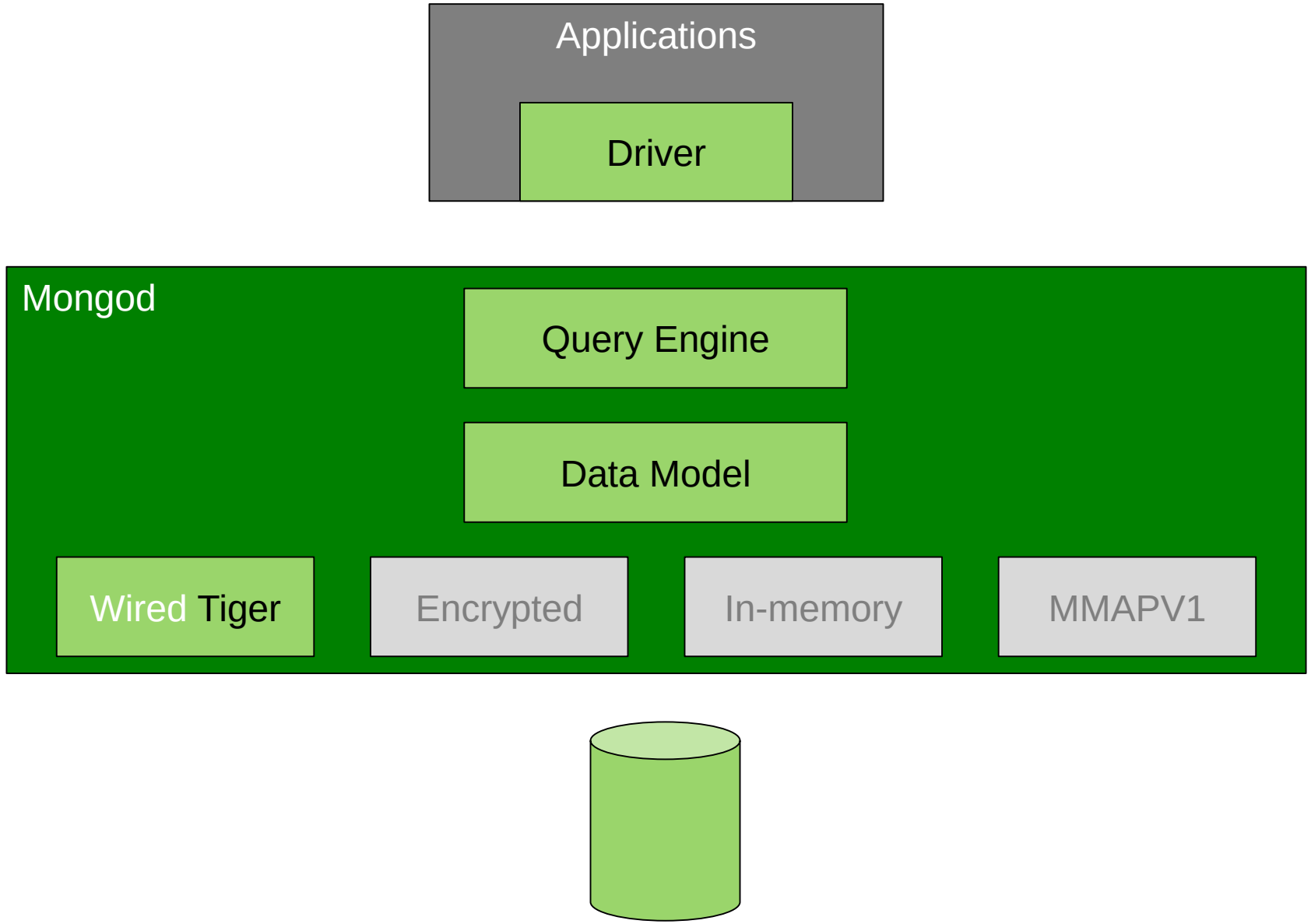
Beyond The Basics

- Follow on from Back to Basics : An Introduction to NoSQL and MongoDB
- Covers more advanced topics:
  - **Storage Engines**
    - What storage engines are and how to pick them
  - **Aggregation Framework**
    - How to deploy advanced analytics processing right inside the database
  - **The BI Connector**
    - How to create visualisations and dashboards from your MongoDB data
  - **Authentication and Authorisation**
    - How to secure MongoDB, both on-premise and in the cloud

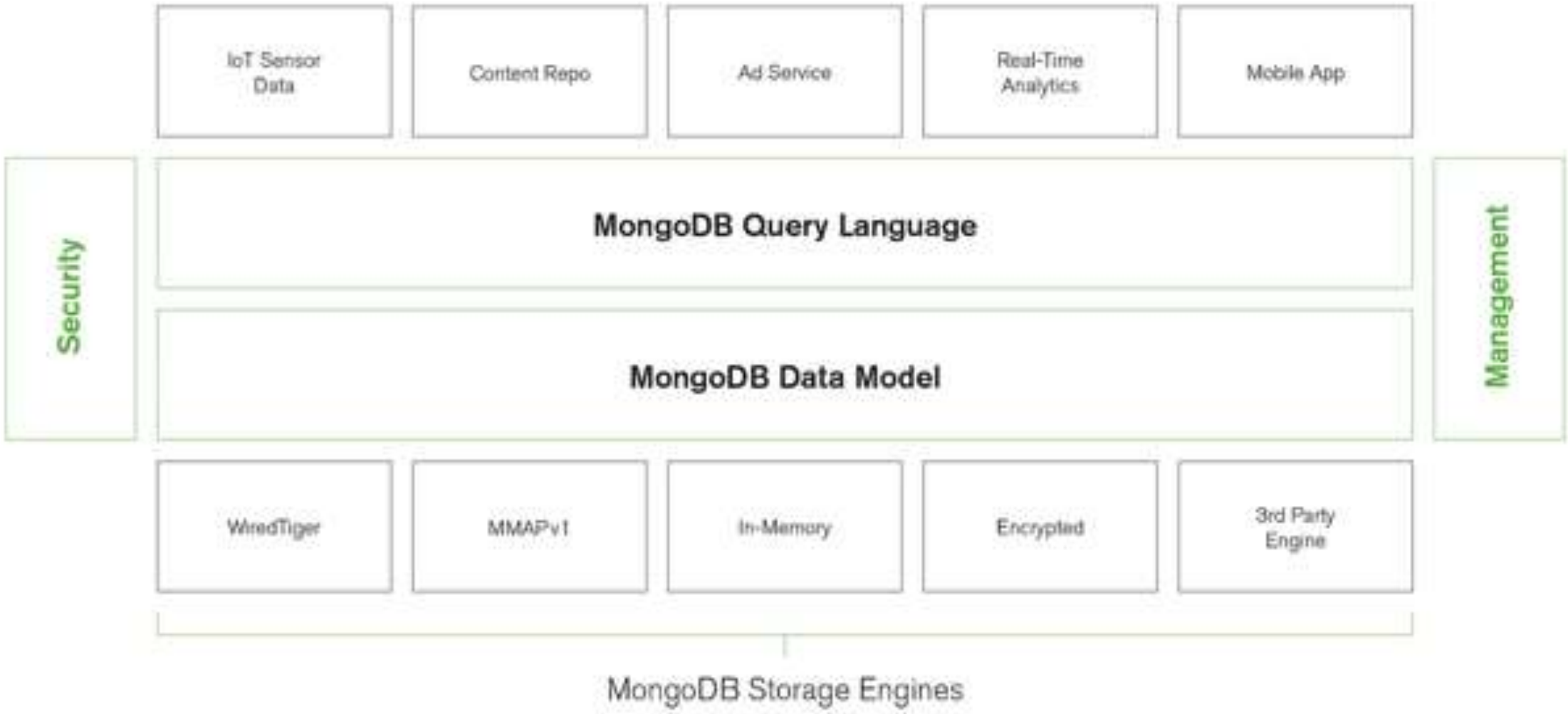
Replica Set Refresher



Storage Engines



Storage Engines



Comparison

	MongoDB WiredTiger	MongoDB In-Memory	MongoDB Encrypted	MongoDB MMAPv1
Write Performance	Excellent Document-Level Concurrency Control	Excellent Document-Level Concurrency Control	Good Document-Level Concurrency Control	Good Collection-Level Concurrency Control
Read Performance	Excellent	Excellent	Good	Good
Predictable Latency (99%)	Good	Excellent	Good	Good
Disk Compression Support	Yes	N/A	Yes	No
MongoDB Query Language Support	Yes	Yes	Yes	Yes
Secondary Index Support	Yes	Yes	Yes	Yes
Replication Support	Yes	Yes	Yes	Yes
Sharding Support	Yes	Yes	Yes	Yes
Ops Manager & Cloud Manager Support	Yes	Yes	Yes	Yes
Native Encryption-At-Rest	No	N/A	Yes	No
Read Concern	Yes	Yes	Yes	No
Security Controls	Yes	Yes	Yes	Yes
Larger than RAM Datasets	Yes	No	Yes	Yes
Platform Availability	Linux, Windows, Mac OS X	Linux, Windows, Mac OS X	Linux, Windows, Mac OS X	Linux, Windows, Mac OS X, Solaris (x86)

Storage Engine Support

- Arrived with MongoDB 3.0 (March 2015)
- Two storage engines intially supported
  - Wired Tiger
  - MMAPV1
- MongoDB 3.2 (Dec 2015)
  - made Wired Tiger the default storage engine
  - added the encrypted storage engine
- MongoDB 3.2.6 – In memory storage engine

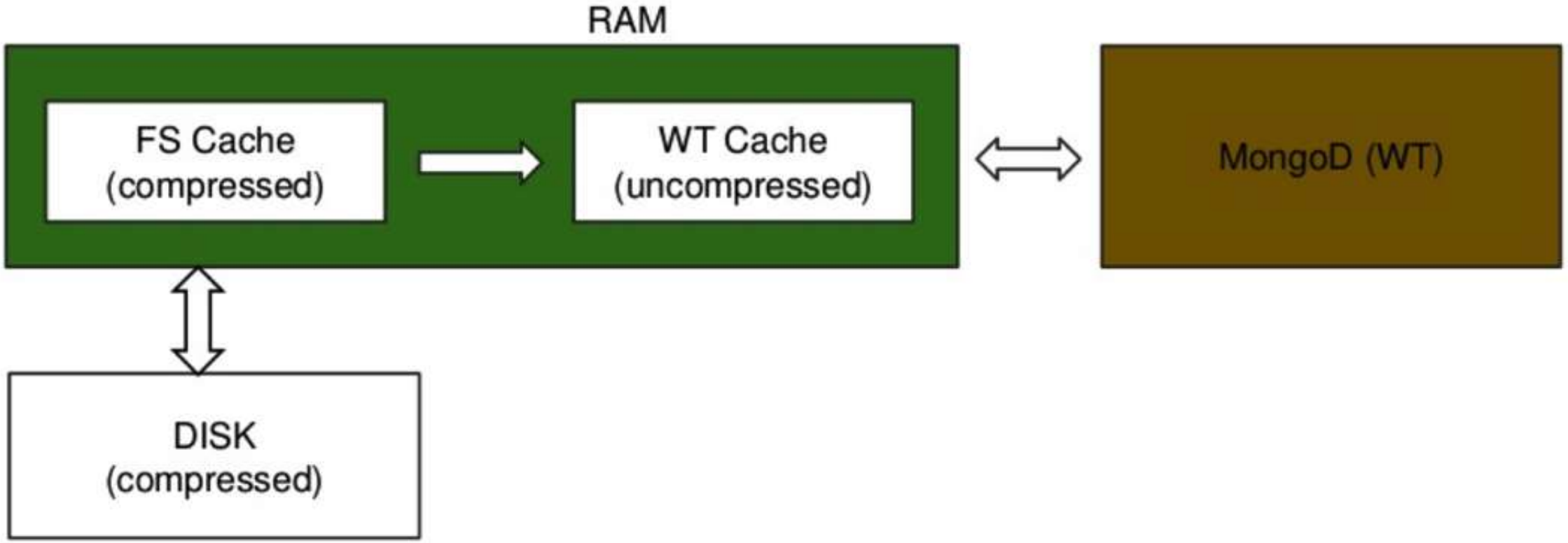
MMAPV1

- Memory mapped, used MMAP to map pages to memory
- The original MongoDB storage engine
- Collection level locking
- Refactored MongoDB 2.6 code base
- Fast for reads, slow for writes
- Can't utilise the memory and core footprint of modern server architectures
- Was still the default storage engine in 3.0

Wired Tiger

- Created by founders of BerkeleyDB and subsequently Oracle's NoSQL database
- Designed for modern server architectures (high memory, high core count)
- Uses a "lock free" architecture to minimise locks that at held
- Use MVCC (multi-version concurrency control) to maximise the ability for readers to access data consistently
- Designed to be a "database toolkit"
- Has internal support for row stores, column stores, LSM trees, transactions (not all these are used by MongoDB)
- Is the basis of the encrypted storage engine

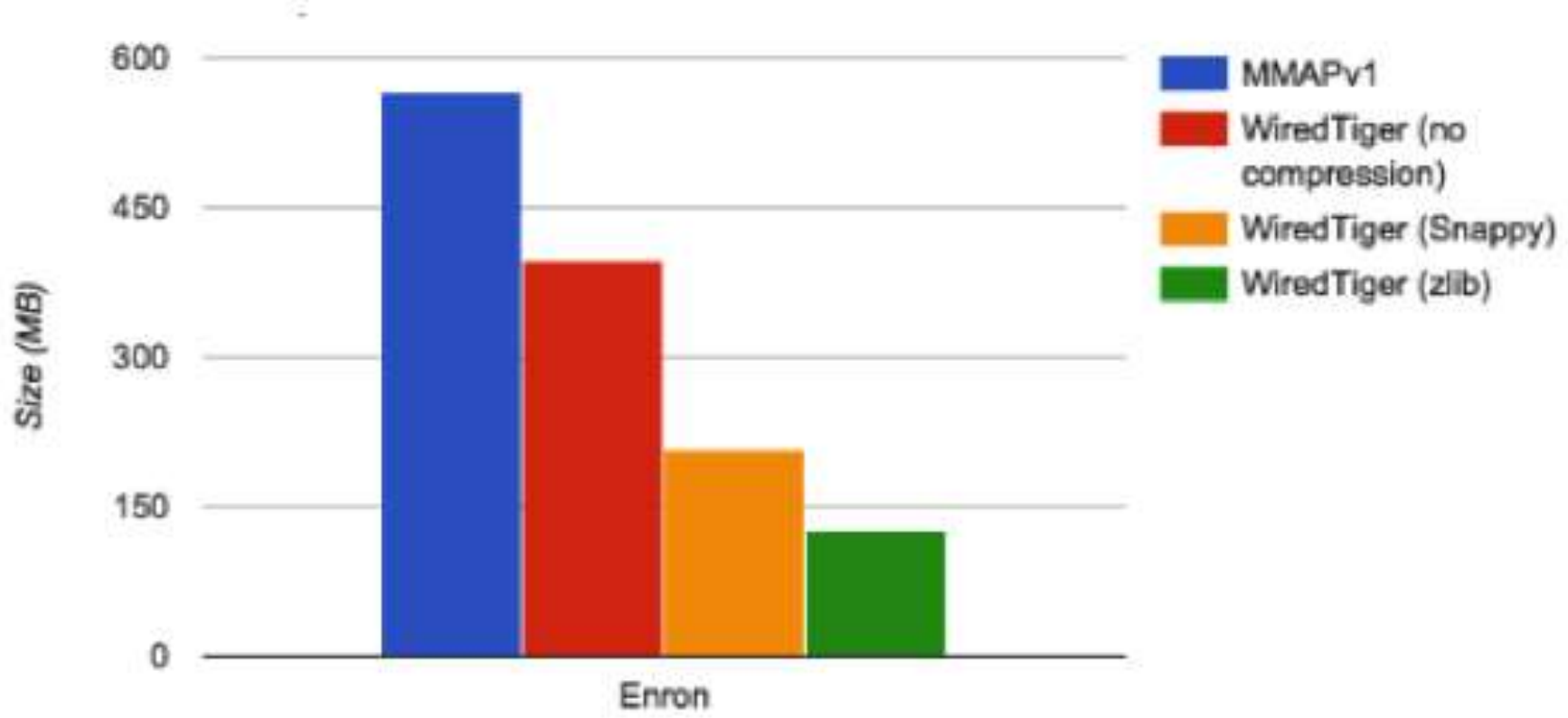
Wired Tiger Performance



Wired Tiger Cache

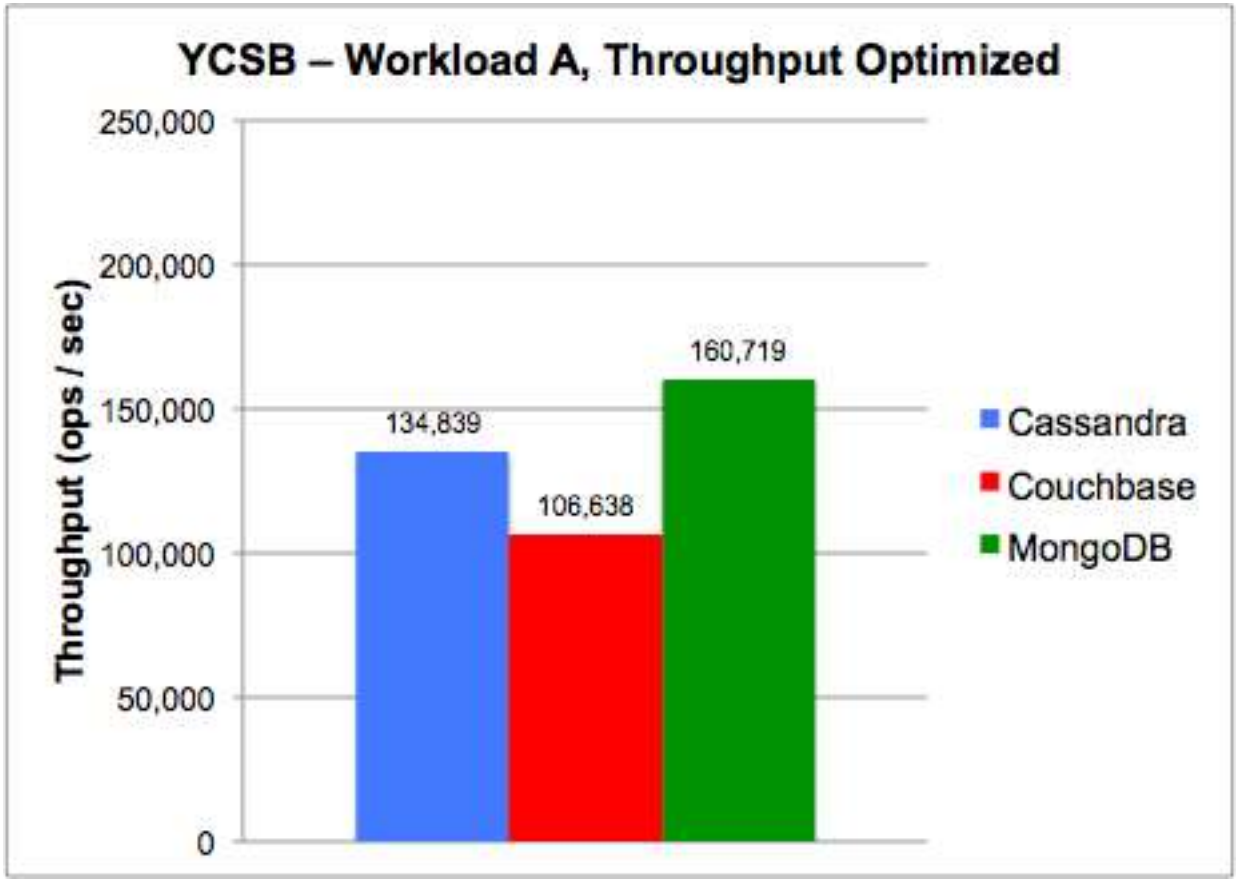
- Starting in 3.4, the WiredTiger internal cache, by default, will use the larger of either:
  - 50% of RAM minus 1 GB, or
  - 256 MB
- Cache does not represent the total memory footprint of the server
- Keeping data in the cache really improves performance

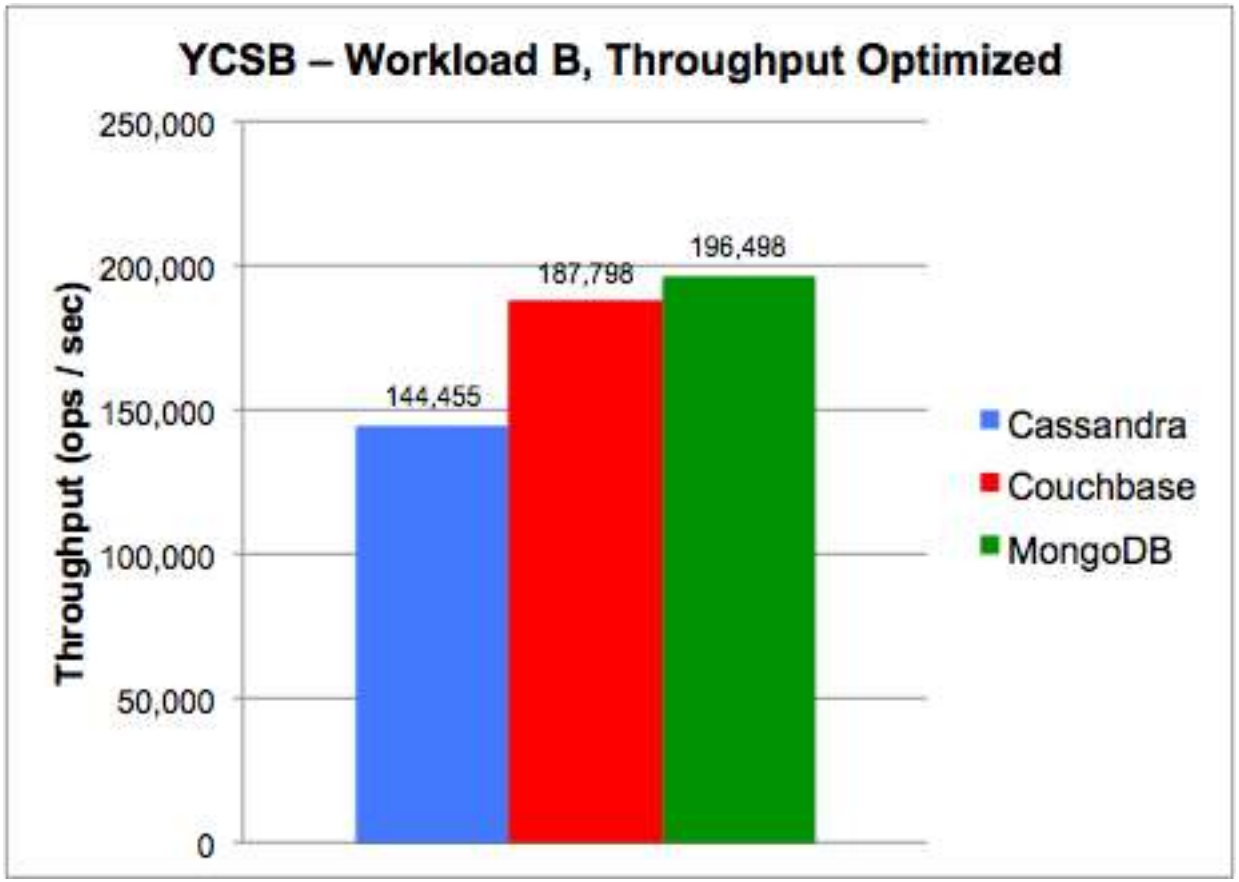
Wired Tiger Compression



Enron Email Corpus
http://mongodb-enron-email.s3-website-us-east-1.amazonaws.com/
About 0.5m messages

Wired Tiger Write Performance





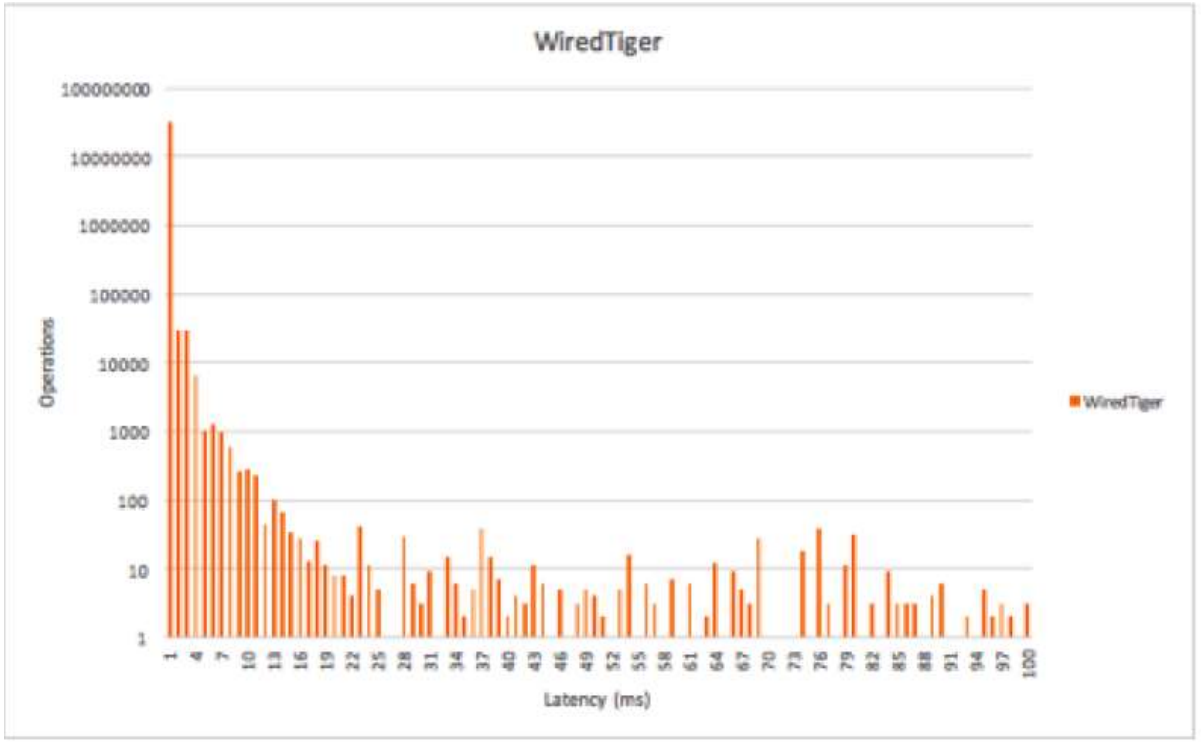
Report
https://www.mongodb.com/collateral/comparative-benchmarks-mongodb-vs-couchbase-vs-cassandra

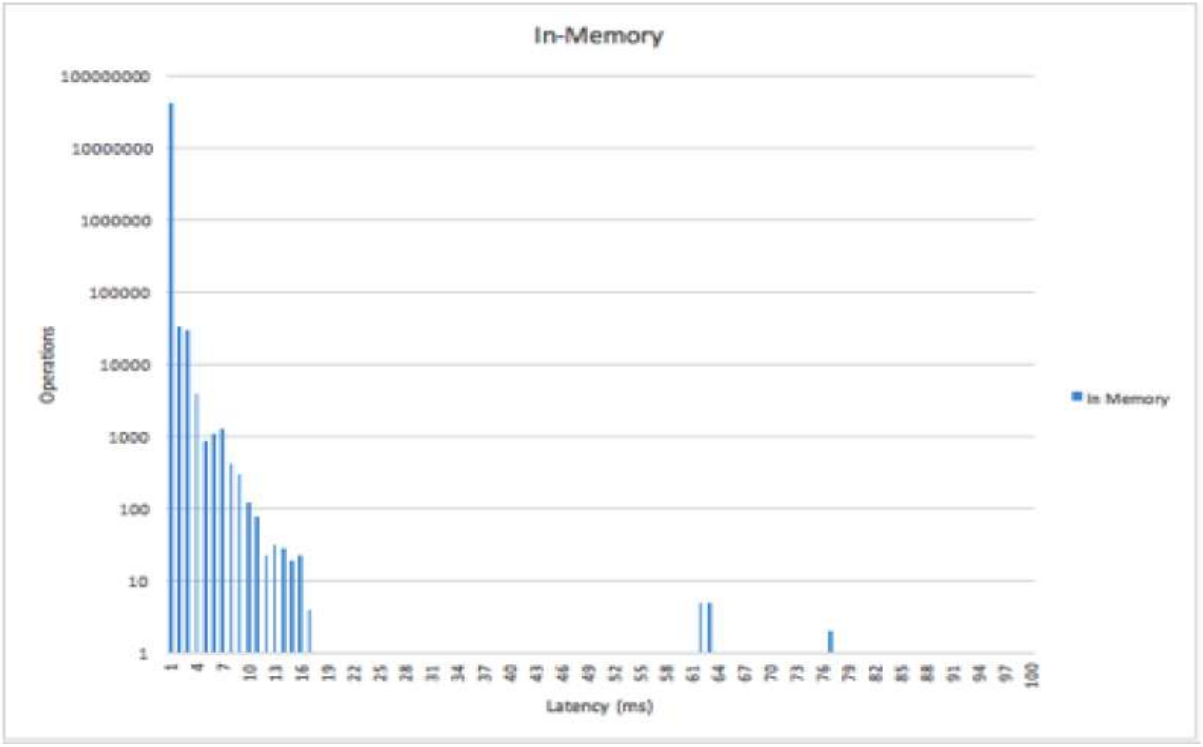
In Memory

- `--storageEngine inMemory`
- Design for low, consistent latency
- Dataset must fit in memory
- By default will allocate 50% of RAM minus 1GB
- Can configure with `–inMemorySizeGB` to reduce on increase
- Exceed memory error:

```
"WT_CACHE_FULL: operation would overflow cache"
```

In Memory Performance

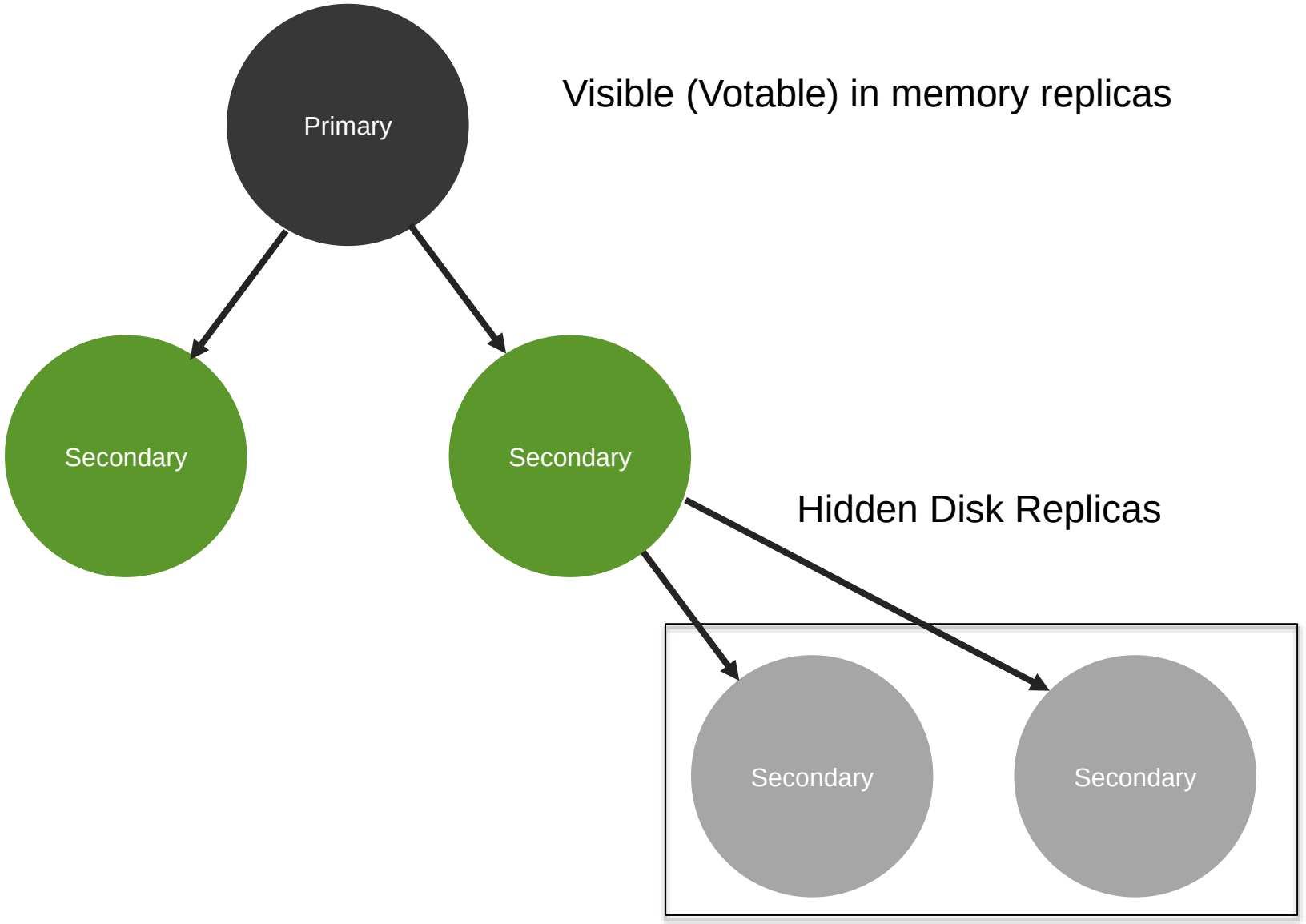




Encrypted Storage Engine

- Built on the Wired Tiger Storage engine
- Supports KMIP - Key Management Interoperability Protocol
  - https://en.wikipedia.org/wiki/Key_Management_Interoperability_Protocol
- Start server with `–enableEncryption`
- Essentially a variant on the WT storage engine as opposed to a storage engine in its own right
- The KMIP key is used to generate internal keys to encrypt the data

Example Mixed Deployment



When To Use

- Default is Wired Tiger : Use this every where
- MMAPV1 : Use for migrations from 2.6 and older versions of MongoDB
- InMemory : For low latency read/write operations where long term durability is less of an issue
- Can mix and match
- Easy to switch
- But – if you are building mixed mode cluster please talk to us

Q&A