MongoDB 3.0, Wired Tiger, and the era of pluggable storage engines

Kenny Gorman
Chief Technologist; Data at Rackspace
Co-Founder, ObjectRocket

@rackspace @kennygorman

rackspace®
the #1 managed cloud company

ObjectRocket
by Rackspace

Whats up with MongoDB 3.0+

- Storage Engines
  - Pluggable storage engine API
  - Wired Tiger, mmapV1, TokuSE, …
  - MongoDB ships with mmapV1, and Wired Tiger
- Why do I care?
- It's early

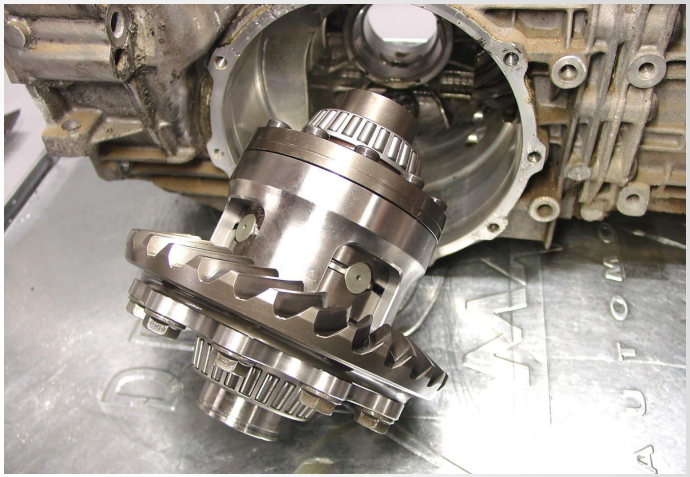
Wired Tiger

- Acquisition of Wired Tiger Inc
- Technology plus Architects
  - Sleepycat Software
  - Berkeley DB

- Proper DB Engine
  - MVCC
  - Durability
  - Compression
  - Performance
  - Full docs
  - LSM and Btree indexing

Wired Tiger and MongoDB

- New configuration **options**
- Different file format and layout
- Allocates files at write time.
- Write ahead log (WAL)
- Wire protocol stays same
- Oplog format changes
- Memory allocated for page cache
- Subset of WT features implemented
- Traditional tools are aware of different engines



New WT stuff to consider

- files allocated at write time, no preallocation.
- no padding factor, no powerof2sizes
- indexes and data locations may be separate
- no huge pages, keep it off (mostly like before)
- NUMA configs: zone_reclaim == 0
- SSD or spinning disk, you have choices now
- A whole new set of configurables
- oh %^$! faults gone in mongostat!
- no LSM indexing yet
- driver updates

Wired Tiger configuration

```
--storageEngine wiredtiger

--wiredTigerCacheSizeGB 100

--wiredTigerEngineConfig "<option>=<setting>,<option>=<setting>"

--wiredTigerCollectionConfig "<option>=<setting>,<option>=<setting>"

--wiredTigerIndexConfig "<option>=<setting>,<option>=<setting>"
```

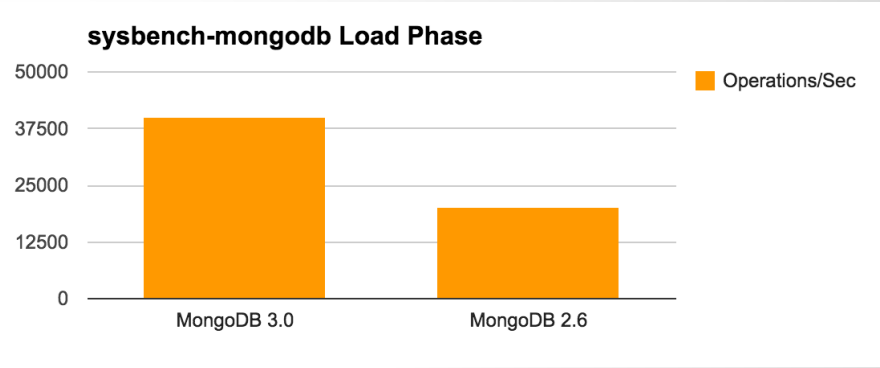
```
db.createCollection("<collectionName>",

       {storageEngine: {wiredtiger: {configString:"<option>=<setting>,<option>=<setting>"}}});
```

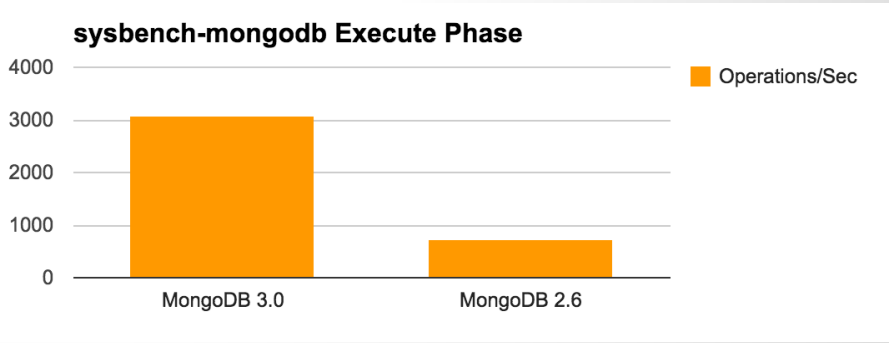
Wired Tiger configuration

```
storage:
   dbPath: "/data/mongodb"
   journal:
       enabled: true
   engine: "wiredTiger"
   wiredTiger:
       engineConfig:
           cacheSizeGB: 99
           journalCompressor: none
           directoryForIndexes: "/indexes/mongodb/"
       collectionConfig:
           blockCompressor: snappy
       indexConfig:
           prefixCompression: true
systemLog:
   destination: file
   path: "/tmp/mongodb.log"
   logAppend: true
processManagement:
   fork: true
net:
   port: 9005
   unixDomainSocket:
       enabled : true
```

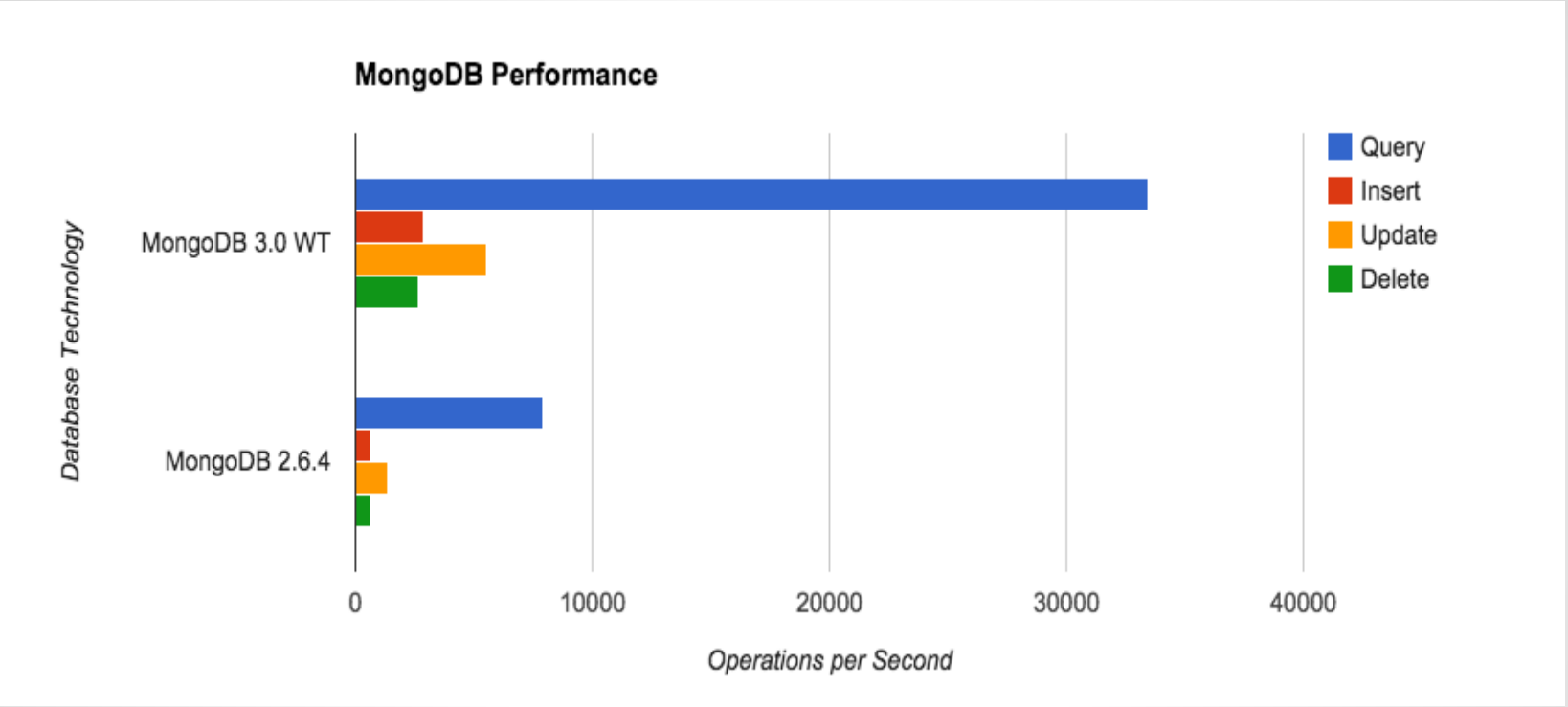
Wired Tiger Performance

- sysbench-mongodb
- 2.6.8 vs 3.0.0
- Mix I/U/D/Q
- Concurrency == Awesome



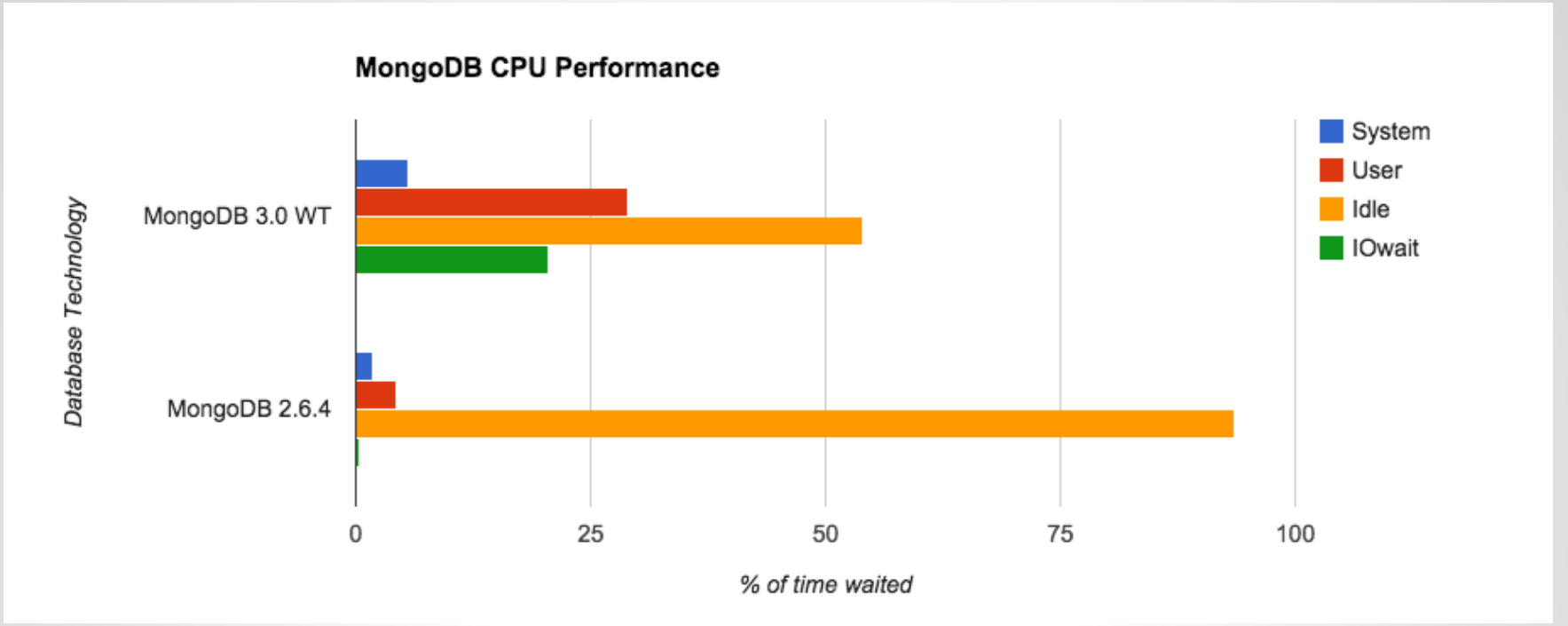


Wired Tiger Performance



sysbench-mongodb benchmark.
https://gist.github.com/kgorman/13bd5613a2c45e131edf
~120GB data, 99GB cache, PCIe Flash, Bare Metal

Wired Tiger Performance



sysbench-mongodb benchmark.
https://gist.github.com/kgorman/13bd5613a2c45e131edf
~120GB data, 99GB cache, PCIe Flash, Bare Metal

rackspace®
the #1 managed cloud company

ObjectRocket
by Rackspace

Wired Tiger Compression

- snappy / zlib for data (page/block level data compression)
- copy on write … ish
- prefix for indexes
- no dictionary compression nor Huffman encoding
- memory implications, expansion at read and CPU

The cache generally stores uncompressed changes (the exception is for very large documents). The default snappy compression is fairly straightforward: it gathers data up to a maximum of 32KB, compresses it, and if compression is successful, writes the block rounded up to the nearest 4KB.

The alternative zlib compression works a little differently: it will gather more data and compress enough to fill a 32KB block on disk. This is more CPU intensive but generally results in better compression ratios (independent of the inherent differences between snappy and zlib).

-**Michael Cahill**

Compression Tradeoffs

- CPU vs on disk compression vs performance
- memory usage profile, expansion of data in memory
- SSD vs spinning disk

YMMV

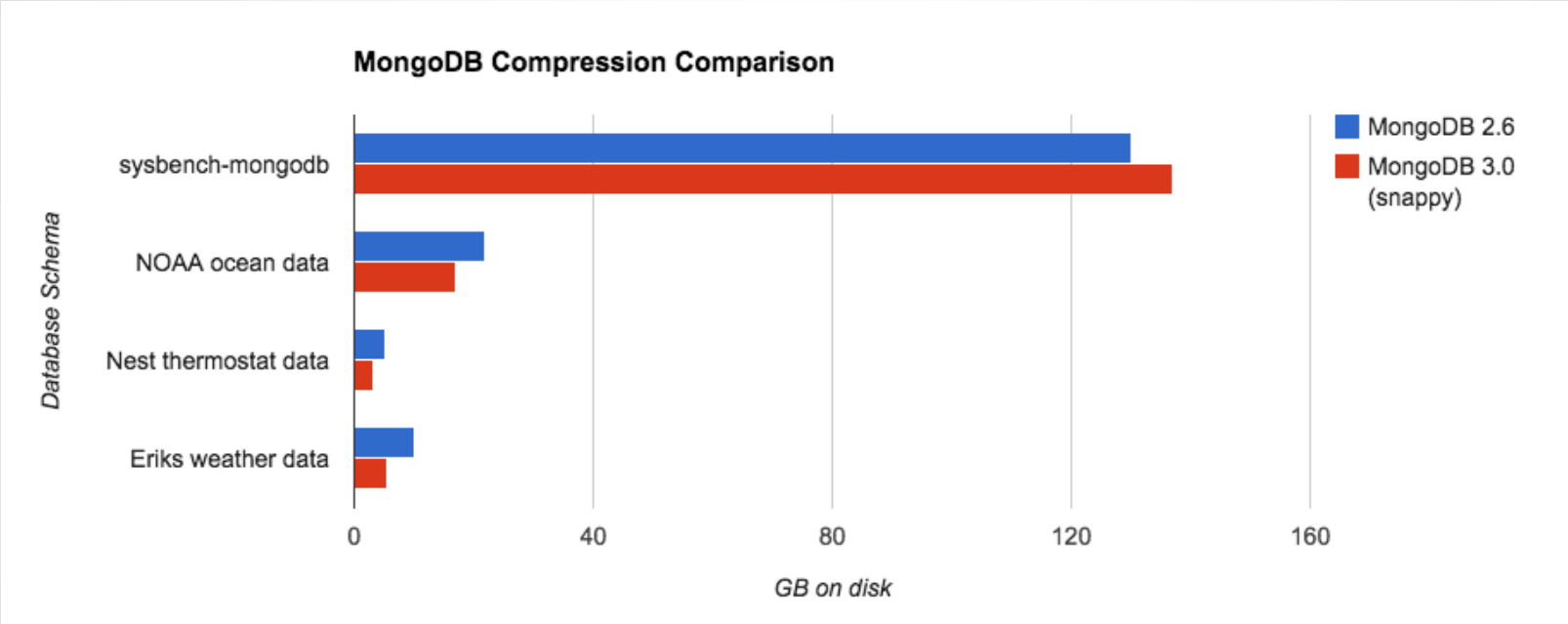
- Test yer shit

Compression Realities?


```
{
        "_id" : ObjectId("53e4fcc42239c23dce3cb7bc"),
        "station_id" : 8461490,
        "loc" : {"type" : "Point","coordinates" : [-72.09,41.3614]},
        "name" : "New London",
        "products" : [
                {"v" : 69.4,"t" : ISODate("2014-08-08T16:24:00Z"),"name" : "water_temperature","f" : "0,0,0"},
                {"v" : 77,"t" : ISODate("2014-08-08T16:24:00Z"),"name" : "air_temperature","f" : "0,0,0"},
                {"d" : "360.00","g" : "8.75","f" : "0,0","s" : "4.08","t" : ISODate("2014-08-08T16:24:00Z"),"dr" : "N","name" : "wind"},
                {"v" : 1015.8,"t" : ISODate("2014-08-08T16:24:00Z"),"name" : "air_pressure","f" : "0,0,0"}
        ],
        "fetch_date" : ISODate("2014-08-08T16:37:22.640Z"),
        "id" : 8461490
}
```


- Sensor data, typical type of workload
- Compresses just 20% using snappy
- https://gist.github.com/kgorman/7d48f37ec5b23efb1475

Various compression results



as measured with: db.<collection>.stats[“size”]

rackspace®
the #1 managed cloud company

ObjectRocket
by Rackspace

Wired Tiger Fragmentation

- Say it isn't so!
- block_allocation and file_extend tunables
- in-place updates don't exist in WT, so is fragmentation worse?
- Compaction w/o blocking?
  http://source.wiredtiger.com/2.5.0/compact.html

Demo

Getting there from here

- Play today; *but wait a while for gods sake*
- 2.4 -> 2.6 -> 3.0
- Use replica sets to migrate seamlessly
- Can fallback if needed using replica sets
- http://docs.mongodb.org/manual/release-notes/3.0-upgrade/
- Go slow, be methodical

Key take-aways

- It's a whole new world
- Start testing now()
- Understand the tradeoffs of compression, performance, and CPU usage for *your* workloads and datasets
- Performance likely easy win
- Compression likely not
- YMMV

ObjectRocket/Rackspace + MongoDB 3.0

- Play today on cloud servers:

https://data.rackspace.com/blog/mongodb-3.0-getting-started/

- ObjectRocket is evaluating 3.0 just like everyone else
- Test in dev, move to prod

More reading

- http://docs.mongodb.org/v3.0/release-notes/3.0/
- http://www.wiredtiger.org
- https://data.rackspace.com/blog/mongodb-3.0-getting-started/

Contact

@kennygorman

@rackspace

kenny.gorman@rackspace.com

http://data.rackspace.com/databases/