MongoDB and Sharding for Scalability: Tips & Tricks

Tokutek, Inc.
57 Bedford Road, Suite 101
Lexington, MA 02420

www.tokutek.com

Tokutek: Performance Databases

- What is Tokutek?
  - TokuMX: high-performance distribution of MongoDB.
  - TokuDB: high-performance storage engine for MySQL and MariaDB.
  - Open source, same core storage code in both products.

- Tokutek Removes Limitations
  - Improve insertion performance by 20X.
  - Reduce HDD and flash storage requirements up to 90%.
  - No need to rewrite code.

Tokutek Mission:

Empower your database to handle the Big Data requirements of today's applications.

Tokutek Customers

Webinar Housekeeping

- This webinar is being recorded.
- A link to the recording and to a copy of the slides will be posted on tokutek.com.
- We welcome questions: enter questions into the chat box and we will respond at the end of the presentation.
- Think of something later?
  - Email us at contact@tokutek.com
  - Visit tokutek.com/contact

Agenda

- Reasons for sharding in MongoDB
- Problems that arise with MongoDB sharding
- Common resolutions
- What is TokuMX?
- TokuMX resolutions for sharding problems
- A new sharding architecture for scaling out writes with fewer machines
- Q&A

Reasons for Sharding in MongoDB

1. Scale out write-induced I/O (spindles).
2. Scale out write locking (processes).
3. Scale out disk space.

Problems with MongoDB Sharding

- Chunk balancer is intrusive.
  - Migrations are slow.
  - Migrations induce a lot of I/O.
  - Hurts write performance.
  - Hurts concurrency.
- Shard key selection is hard.
  - Writes want higher entropy.
  - Range queries want lower entropy.
    - Range queries still slow after index search.

Common Resolutions

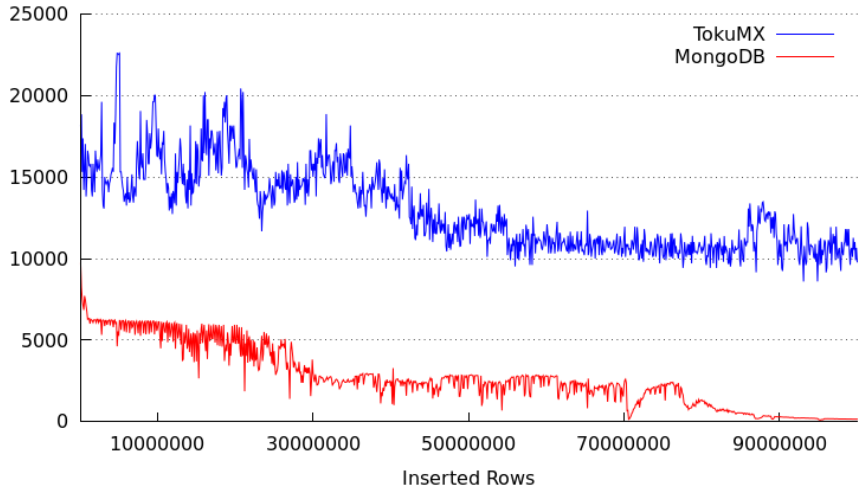
- Don’t shard:
  - Write/keep less data.
  - Use fewer indexes: don’t run interesting queries.
- Schedule balancing window:
  - Avoid peak traffic times.
- Hardware:
  - Buy faster SSDs.
- Hashed shard key:
  - No range queries.
  - But, no balancer!

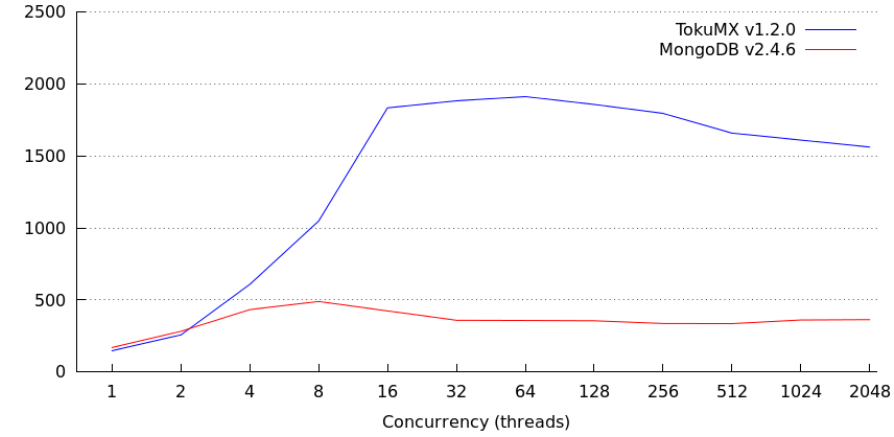
What is TokuMX?

- TokuMX: MongoDB with improved storage.
- Drop-in replacement for MongoDB apps:
  - Including replication and sharding.
  - Same data model.
  - Same query language.
  - Drivers just work.
  - No text indexes or geospatial.
- Open source:
  - http://github.com/Tokutek/mongo
- Same storage core as TokuDB for MySQL.

TokuMX Resolutions

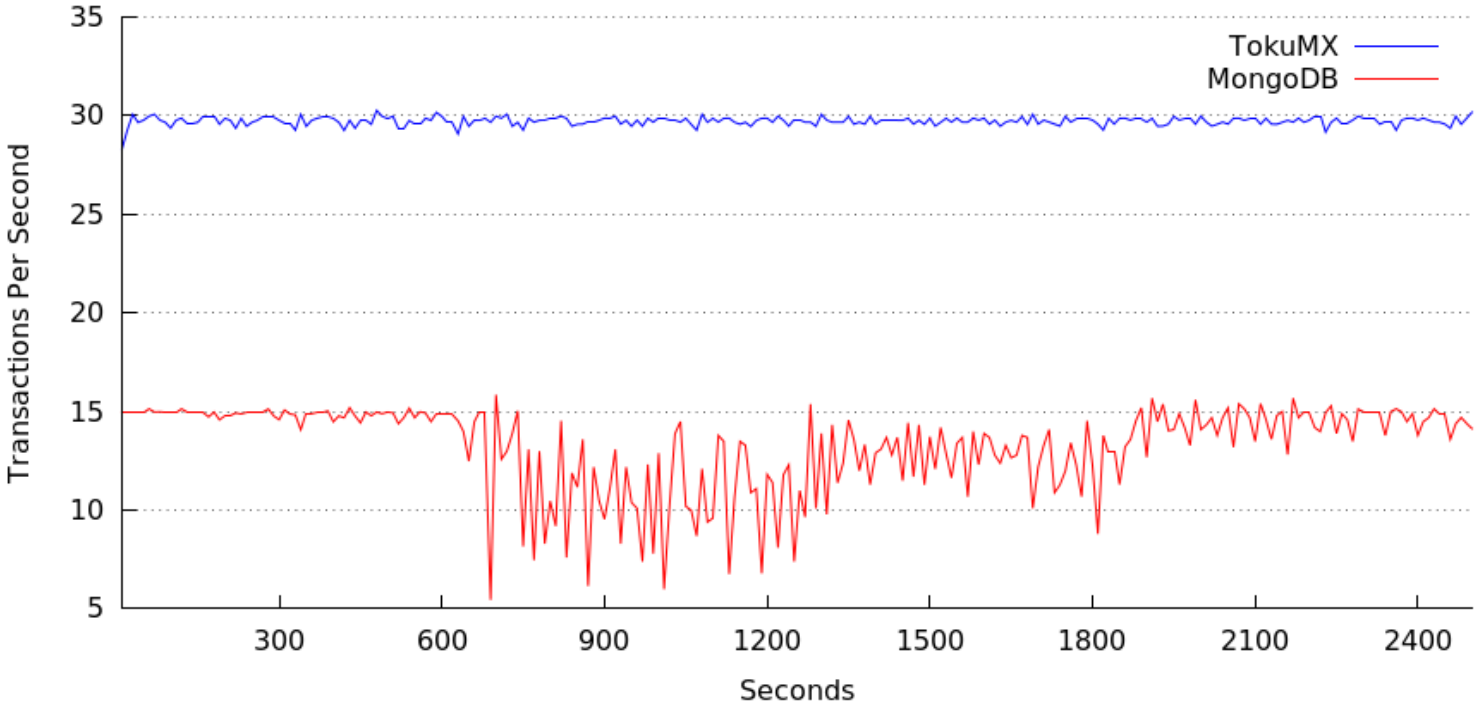
- Don’t shard:
  - Higher write throughput, no need to scale I/O.
  - No DB-level write lock, no need to have more processes.
  - Compression: no need to scale disk space.
  - Bonus: fewer operational costs, fewer machine failures, less complexity.





TokuMX Resolutions: Range-based Sharding

- Migrations in TokuMX are fast.
  - Clustering shard key.
  - High write throughput with little I/O.



TokuMX Resolutions: Range-based Sharding

- Clustering shard key means range queries are fast.
- Tip #1: Disable `migrateUniqueChecks`.
  - Even less I/O on recipient shard: can migrate chunks with **zero** random I/O.
  - Dangerous if **both** of these are true:
    - Using `_id` as primary key (not default for collections created with `shardCollection`).
    - Application is generating its own non-unique `_id` values (this is also dangerous in stock MongoDB).
  - Moral: create the collection with `sh.shardCollection()`, then load data.

TokuMX Resolutions: Hash-based Sharding

- Perfectly balanced write distribution.
- Tip #2: De-cluster the shard key.
  - Don’t need clustering because we won’t be doing range queries or chunk migrations.
  - `sh.shardCollection(‘db.coll’, {_id: ‘hashed’}, false, false)`
    - On empty collection, to pre-split.
  - `noAutoSplit=true`
    - Auto-split is slow on non-clustering shard key, but make sure collection is pre-split and balanced before loading.
  - Adding/removing shards is expensive, be careful.

TokuMX Resolutions: Hash-based Sharding

- Tip #3: Beware of write batching.
  - `mongos` divides write batches into contiguous chunks for each shard.
    - When using a hashed shard key, it's unlikely that many documents next to each other in a batch will go to the same shard.
  - With normal write batches, you lose the benefit of batching, still have many round trips between `mongos` and `mongod`.
  - Try to batch docs that all have the same shard key.
  - https://github.com/Tokutek/mongo/issues/912 so eventually you won't need to do this on TokuMX.

TokuMX Sharding Tricks!

- Trick #1: Scale out without buying more hardware.
- Setup:
  - Replica set of 3 nodes:
    - Redundancy: 1 nodes can partition, still have quorum.
    - Redundancy: 2 nodes can explode, still have data.
    - Read scaling: 3x IOPS available for (secondary) reads.
  - Want 3x write scaling too.
    - With MongoDB, need to buy 6 more machines to maintain above advantages.
      - Also, 3x operations and maintenance cost of replica set.

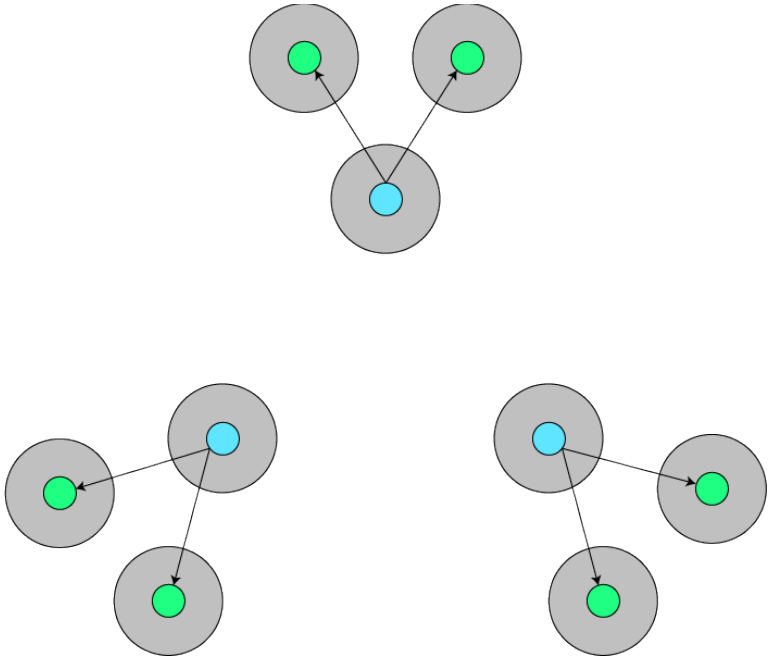
TokuMX Sharding Tricks!

RAID level	Disk meaning	MongoDB analog
RAID 0: striping	Faster, no redundancy	Sharding with single-server shards, no replication
RAID 1: mirroring	Redundant, not faster	Replica set, no sharding
RAID 10: striping over mirroring	Faster and redundant, many disks	Sharding over replica sets

Other RAID architectures?

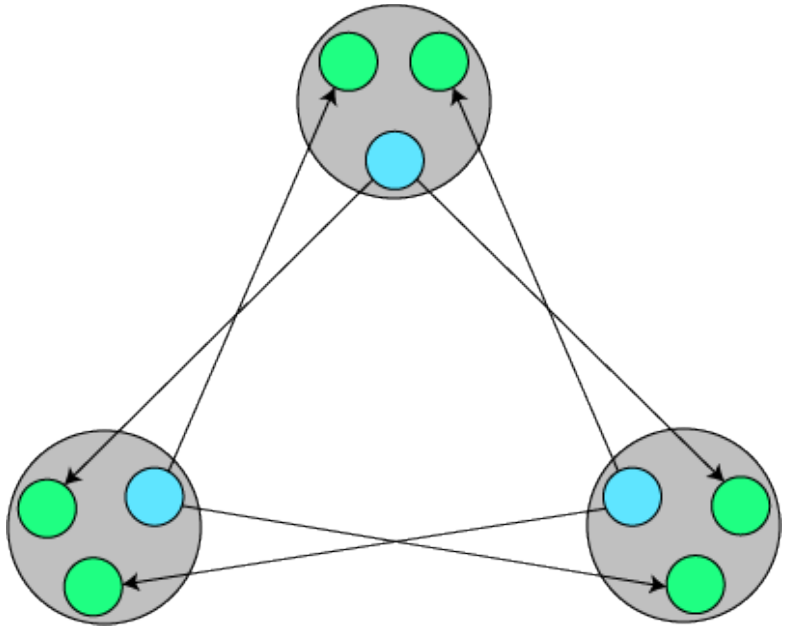
TokuMX Sharding Tricks!

- Traditional sharded cluster:
  - 3 replica sets.
    - Primary: blue
    - Secondary: green
  - 3 shards.
  - Extra stuff (routers, config servers).
  - General: R replication factor & S sharding: R*S machines.
- RAID 10



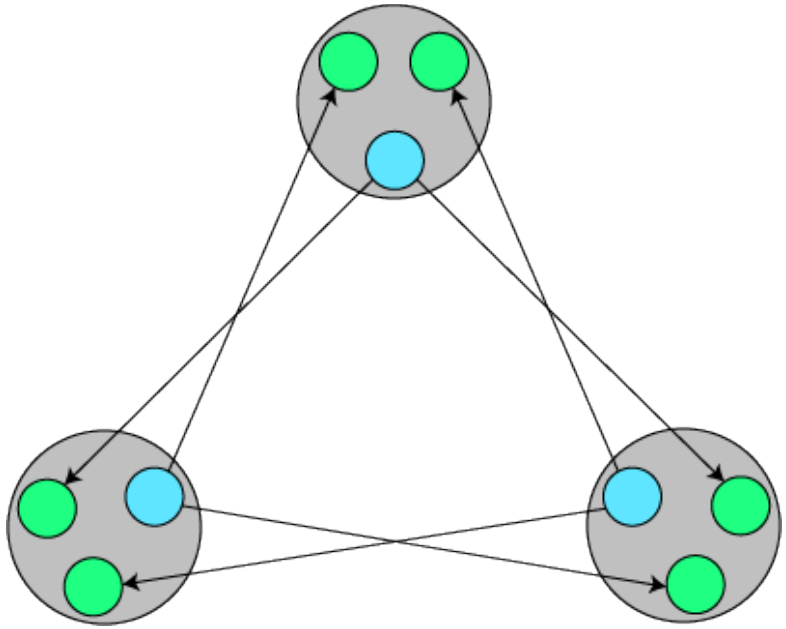
TokuMX Sharding Tricks!

- TokuMX permits new sharded cluster architecture:
  - `cacheSize` makes multi-tenancy easier.
  - Virtually no I/O needed for secondaries.
- RAID 5 (ish): parity mode
  - Lose any one disk, others compensate.



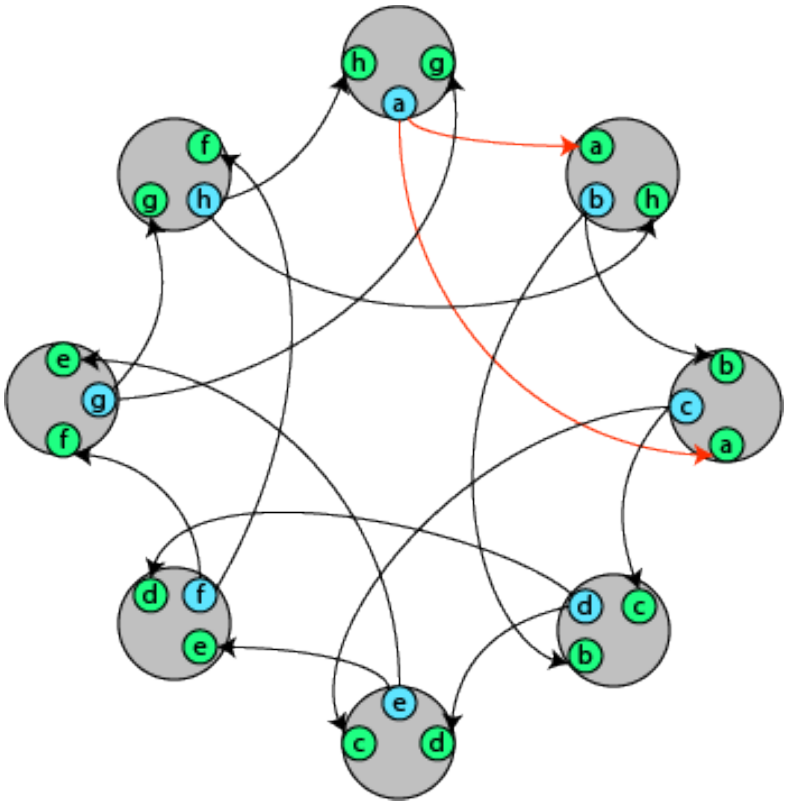
TokuMX Sharding Tricks!

- 3 primaries:
  - 3x write scaling.
- Full copy of data on each node:
  - 1 node partitions:
    - Quorum on all shards, can still process all writes after failover.
  - 2 nodes explode:
    - All shards copied on each node.
  - 3x read scaling.



TokuMX Sharding Tricks!

- Generalize:
  - R replication factor.
  - S write scaling.
  - Need max(R, S) machines in a ring.
  - One shard primary on each of first S machines.
  - For each shard, put R-1 secondaries on each of the consecutive machines after the primary.



Wrap-up

- Avoid balancer stalls by using hashed sharding or switching to TokuMX.
- With TokuMX range sharding, turn off `migrateUniqueChecks`, and don't generate your own `_id` values.
- With TokuMX hashed sharding, de-cluster the shard key, pre-split, and use `noAutoSplit`.
- Always try to use `shardCollection` on a non-existent collection, for optimal indexes and chunk distribution.

Any Questions?

Thank you for attending!
Enter questions into the chat box.

- Download TokuMX free: www.tokutek.com/downloads
- Full write-up on zero-impact chunk migrations here:
  - http://www.tokutek.com/2014/02/tokumx-vs-mongodb-sharding-balancer-performance/
- Contact us: contact@tokutek.com
- Free 30-day enterprise eval: support@tokutek.com
- Follow us on twitter: @tokutek @leifwalsh