MongoDB Readiness from DBRE & Ops Viewpoints

David Murphy
MongoDB Practice Manager
@ Percona

Agenda

- What is DBRE anyhow?
  - Past and Future
  - Why
  - Core
- MongoDB architecture
  - High availability
- Config Management
  - Cloud provisioning
  - Effective monitoring solutions
- Mongo Specifics
  - How to make sure you have consistency?
  - Mongo Zones explained
  - Common Mongo Challenges

What is DBRE anyhow - Past/Now

Past - DBA

- Custodian of Data
- Protectors
- Special Forces / Heroes
- Siloed
- Specialized
- Non-Empathetic to other groups

Now - DBRE

- Custodian of Data
- Educators & Mentors
- Expert Feedback
- T shaped Knowledge
  - Wide breadth
  - Deep knowledge on data
- Empathic and Team building

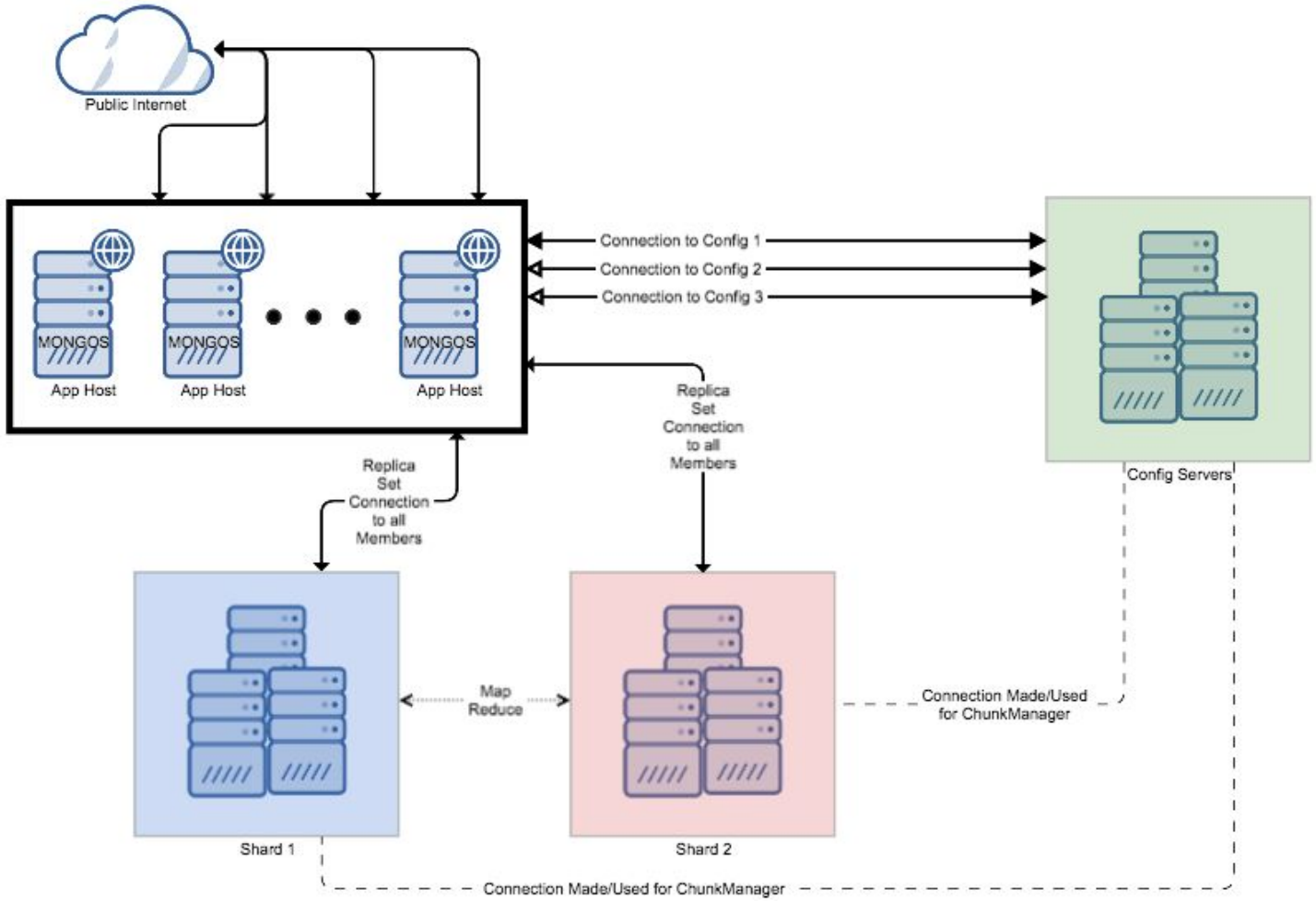
Why?

- The classic DBA was about response not prevention at the core
- The database being healthy doesn't mean the stack is.
- The closer we are to the data the more reliability matters

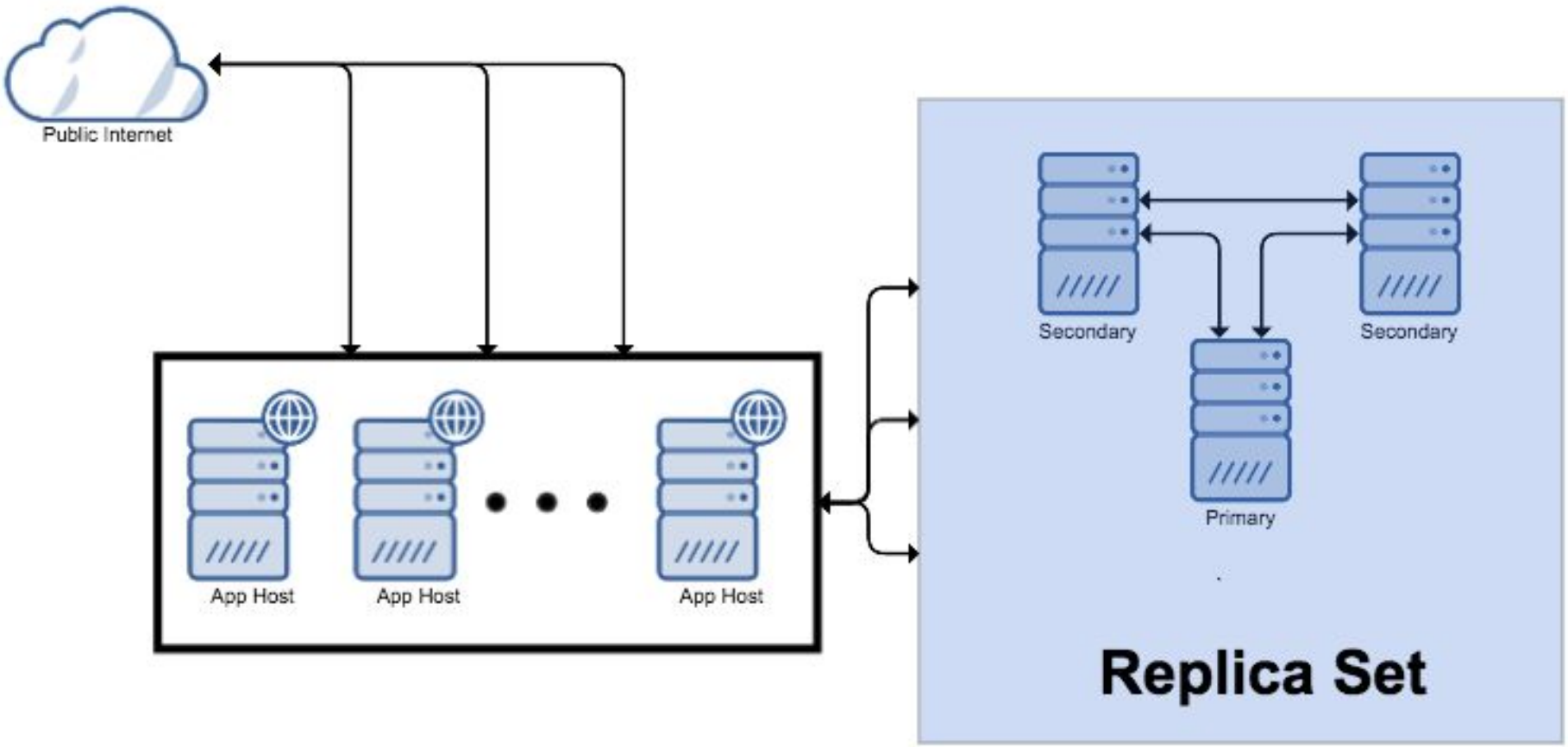
Due to this we needed to re-examine what our professionals do

  - Troubleshoot issues
  - Prevent issues via education and empowerment of devs
  - DBA are just another member of Operations not a snowflake

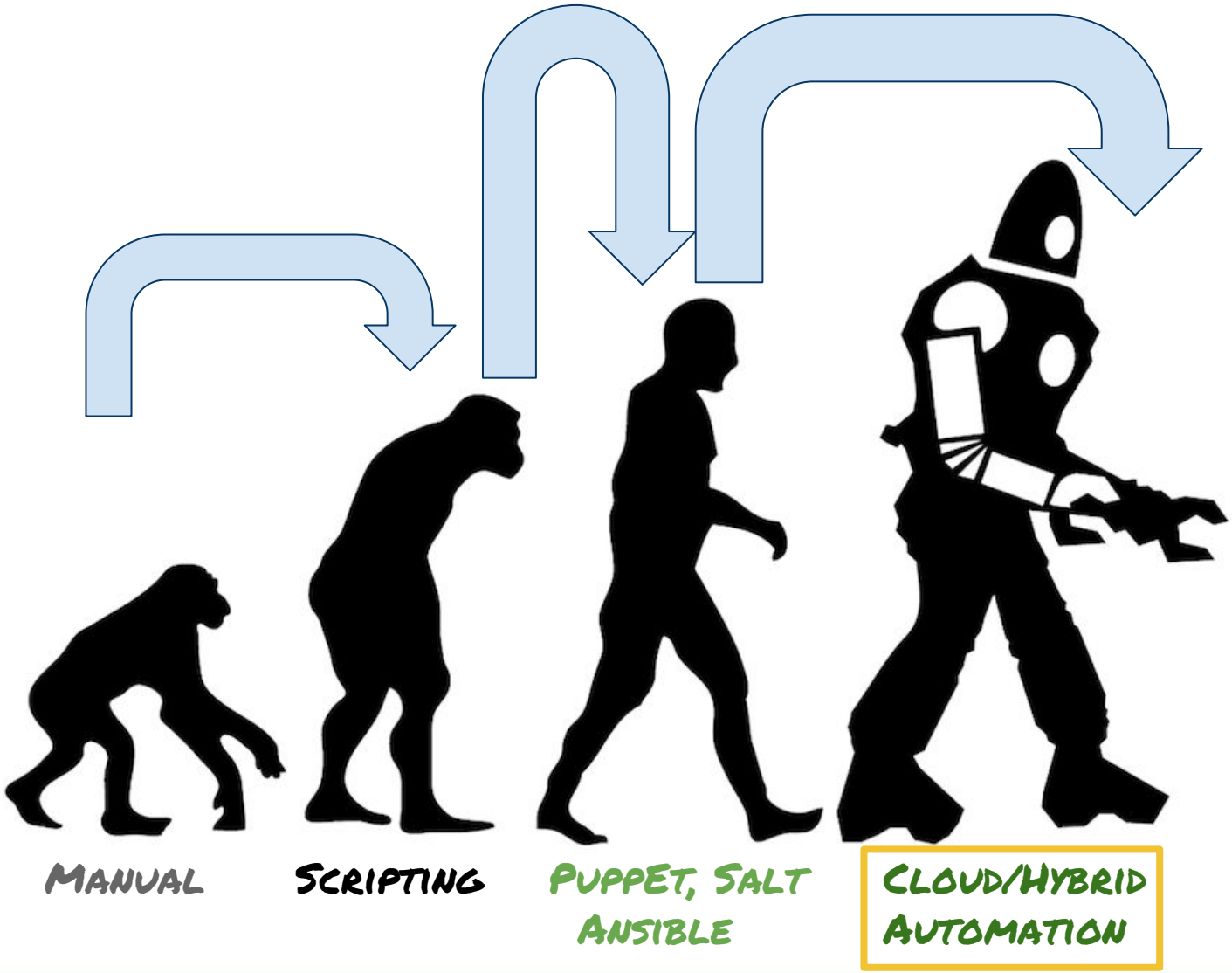
MongoDB Architecture



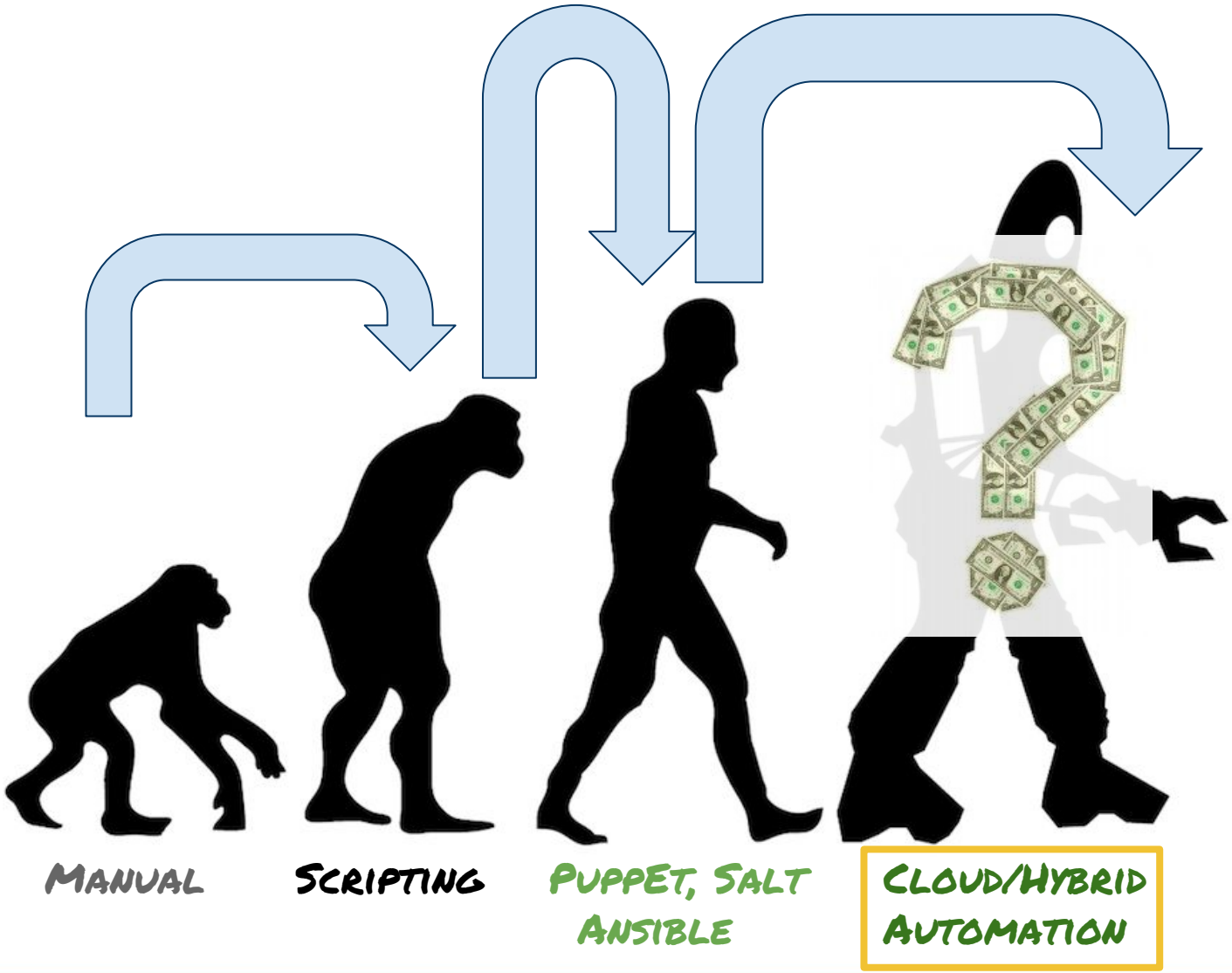
High Availablity



Config Management



Config Management



Config Management vs Automation

The reality is config management are the **EXACT** same except one it more polished and typical has a GUI

Vendors will always push you toward what will get them more income, in the case of Mongo:

- Ops Manager
- Cloud Manage
- MongoDB Atlas
- Severalnines Cluster Control

THIS IS GOOD, but if you have SRE/DevOps you tie your teams hands!

Cloud Provisioning Options

Some example places where there is

- Docker / Kubernetes
- Ansible / Ansible Tower
- Cloud Foundry on AWS

Plenty can be found on GitHub, however MongoDB Atlas it very useful for starter teams, or if you have not customize a solution to meet your needs.

Atlas and Cluster Control both help you deploy and manage a cluster, however they won't manage the OS, or MongoD directly. Additionally **MongoDB Atlas upgrades mongo for you**, so there is risk if you're on an older driver you could suddenly have issues in your stack.

Effective Monitoring

More of a philosophy than a design here

1. You should only alert on actually issues
2. If your alert does not in itself identify the issue it's the wrong measure.
3. Will this issue take production down? If not why is this not a ticket?
4. Alert on prediction only when necessary
    a. Oplog Length == Bad Backup restoration
5. What is the acceptable degradation of a system that can auto heal before I should wake someone up
    a. Mongo auto fails over, so only alert if there is no primary for an extended amount of time, not on all elections.

Managing MongoDB Consistency

Consistency is a real challenge for DBRE and Operations groups

At the core the issue is we want to give developers choice, but to do so we must educate them.
Some options are:

- To low of a write consistency and data might not be safe (maybe ok though)
- To high and things are slow (majority) or even fragile
- Linear reads are possible with mongo but it's very expensive on the write:majority + reading from multiple nodes. With more in 3.6.

Lets help them understand but not say no to build more teamwork.

Mongo Zones - What is a zone?

With MongoDB you might have heard about tags before, which are now called zones. But what is it?

Sharding

- Ability to tell some ranges of shard keys to only be on some shards
- Ability to balance a given range of data to a limited number of shards
- Ability to tell some collections to only be on a given shard such as session data

ReplSet

- Ability to tell write concern ensure 1 memory + 1 WT node has the date

Mongo Zones - Data locality

Privacy is HUGE

We must think about a global cluster, how can I keep data in it's own regions

1) Shard key must have country or region field.

   This will prevent normal hashed indexes, and you will need to make your own hash.

2) Add Shard Tags for each shard for the region it's in

```
sh.addShardTag("shard_uk", "EU");sh.addShardTag("shard_us", "NA"); sh.addShardTag("shard_ap_jap", "AP");
```

3) Tell the system what countries go where

```
sh.addTagRange("records.docs_col", { country: "uk" }, { country:"uk"}, "EU");
sh.addTagRange("records.docs_col", { country: "DE" }, { country:"DE"}, "EU");
sh.addTagRange("records.docs_col", { country: "us" }, { country:"us"}, "NA");
sh.addTagRange("records.docs_col", { country: "canada" }, { country:"canada"}, "NA");
sh.addTagRange("records.docs_col", { country: "china" }, { country:"china"}, "AP");
```

Mongo Zones - Even Data spreading

Taking the previous example:

What if we had 80% of our users in the US. So we are not just talking about locality but also evenly spreading the users inside of a country.

Calculate a hash makeHash(Hexidecima2Interger(md5(**raw_value**)))

Shard key {country:XX,hash: make_hash(doc._id)}

You could have:

```
sh.addShardTag("shard_us_east", "US");sh.addShardTag("shard_us_west", "US");
sh.addShardTag("shard_canada", "NA-misc");
sh.addTagRange("records.docs_col", { country: "us" }, { country:"us"}, "NA");
sh.addTagRange("records.docs_col", { country: "canada" }, { country:"canada"}, "NA-misc");
sh.addTagRange("records.docs_col", { country: "mexico" }, { country:"mexico"}, "NA-misc");
```

We now have 2 shards for US data evenly split, while all canada and mexico go to the misc shard(s) allow scaling either one we need to!

Mongo Zones - Multiple Engines

This is a much easier design.

We have data coming in but shopping carts and sessions don't matter much and we don't want it slowing down orders and inventory

1) Add Shard Tags for each shard for the type it's in

```
sh.addShardTag("shard_prod1", "Persistent");sh.addShardTag("shard_prod2", "Persistent");sh.addShardTag("shard_mem", "MEM");
```

2) Tell the system what countries go where

```
sh.addTagRange("prod.shopping_cart", { _id: MinKey  }, { _id: MaxKey }, "MEM");
sh.addTagRange("prod.sessions", { _id: MinKey }, { _id: MaxKey }, "MEM");
sh.addTagRange("prod.orders", { order_id: MinKey }, { order_id:MaxKey}, "Persistent");
```

Mongo Challenge #1

Backups!

Especially consistent backups, which are easy for a replica set, but harder when sharded

Enter Mongo-consistent-backup on github.com/Percona-lab

However taking a backup is only your first step:

- How do you recover to a point in time? We have a blog for that :)
- When should you use a logical vs binary backup?
- What can we do today for a shard consistent binary backup?

How often should you test your backups?

Mongo Challenge #2

Identifying slow queries

This is likely your number ticket from your internal teams.

By default you need to look at the log.It's not as helpful youd assume

```
101)  --log--> 2017-09-19T06:00:03.403+0000 I COMMAND  [conn199265] query
proctor.testSessionDocument query: { ia: true } planSummary: COLLSCAN cursorid:66366665758
ntoreturn:0 ntoskip:0 nscanned:0 nscannedObjects:290975 keyUpdates:0 writeConflicts:0
numYields:2273 nreturned:101 reslen:11130 locks:{ Global: { acquireCount: { r: 4548 } },
MMAPV1Journal: { acquireCount: { r: 2277 } }, Database: { acquireCount: { r: 2274 } },
Collection: { acquireCount: { R: 2274 }, acquireWaitCount: { R: 3 }, timeAcquiringMicros:
{ R: 1460 } } } user_key_comparison_count:0 block_cache_hit_count:0 block_read_count:0
block_read_byte:0 internal_key_skipped_count:0 internal_delete_skipped_count:0
get_from_memtable_count:0 seek_on_memtable_count:0 seek_child_seek_count:0 1592ms
```

Always use operationProfiling.slowOpThresholdMs & operationProfiling.mode in your config!

Mongo Challenge #3

The database just … stops…

Really there are two cases here and they both make a lot of sense

- MongoDB is actively evicting from memory and creates a global lock when it does so. Shard often and fast is a good approach here but each version improves on how it can do it more efficiently
- MongoDB is running out of work slots, 128 read & 128 write by default. Sometimes for busy systems you need to increase this, but do so with care the bigger the queue, the slower it will decide on its next action!

Mongo Challenge #4

- Mongo has had SSL support for a long time but its rarely used
  - Legacy with no checking
    - --ssl in config
    - ssl:true in drivers
- Modern
  - Checks for valid hosts (optional)
  - Setup Key file to us
  - Custom CA for self-signed or alternative signers
  - allowSSL, preferSSL, requireSSL modes
  - Can make replication use x509 also for intra-cluster comms

PERCONA

Thanks for joining!
Be sure to checkout the Percona Blog for more technical blogs and topics!