# MongoDB

Scope: Query Analysis

Ramu RC

1

## Query Analysis

2

## To work out why a query is slow

- .explain()
- db.collection.explain()
- cursor.explain()

3

## Profile

- db.setProfilingLevel(?)
- **Level 0** - The profiler is off and does not collect any data. This is the default profiler level.
- **Level 1** - The profiler collects data for operations that take longer than the value of slowms, which you can set.
- **Level 2** - The profiler collects data for all operations.

4

## current profiling status

- db.getProfilingStatus()

5

## What we achieve?

- when the query was run
- how fast it ran
- how many documents were examined
- the type of plan it used
- whether it was able to fully-use and index
- what sort of locking happened

6

## Sort queries by when they were recorded, showing just commands

- db.system.profile.find({ "command.pipeline": { $exists: true } }, { "command.pipeline":1 }).sort({$natural:-1}).pretty();

7

## Find all queries doing a COLLSCAN because there is no suitable index

- db.system.profile.find({"planSummary":{$eq:"COLLSCAN"}, "op" : {$eq:"query"}}).sort({millis:-1})

- **COLLSCAN** - Collection scan
- **IXSCAN** - Scan of data in index keys
- **FETCH** - Retrieving documents
- **SHARD_MERGE** - Merging results from shards
- **SORT** - Explicit sort rather than using index order

8

## Find any query or command doing range or full scans

- db.system.profile.find({"nreturned":{$gt:1}})

9

## Find the source of the top ten slowest queries

- db.system.profile.find({"op" : {$eq:"query"}}, { "query" : NumberInt(1), "millis": NumberInt(1) } ).sort({millis:-1},{$limit:10}).pretty()

10

## Find the source of the top ten slowest aggregations

- db.system.profile.find({"op" : {$eq:"command"}}, { "command.pipeline" : NumberInt(1), "millis": NumberInt(1) } ).sort({millis:-1},{$limit:10}).pretty()

11

## Find all queries that take more than ten milliseconds

- db.system.profile.find({"millis":{$gt:10}}) .sort({millis:-1})
- displaying both queries and aggregations

12

## Long-running queries and aggregations

- db.system.profile.find({ "millis": { $gt: 10 } }, { millis: NumberInt(1), "query": NumberInt(1), "command.pipeline": 1 }).sort({ millis: -1 })

13

## Find all queries that took more than 20 ms

- db.system.profile.find({"millis":{$gt:20}})

14

## criterion value of slowms

- db.setProfilingLevel(1, { slowms: 30 })
- db.getProfilingStatus()
  - MongoDB confirms that it has set the criterion. { "was" : 0, "slowms" : 100, "ok" : 1 } { "was" : 1, "slowms" : 30 }
- to collect just the data for operations that take longer than the criterion value of slowms

15

## For more info:

- Todo

16