MongoDB 在盛大大数据量下的应用

郭理靖@snda; guolijing@gmail.com;

Sections

- Basic introduction to MongoDB
- Monitor MongoDB
- Backup and Rollback of MongoDB
- Case Study

What's MongoDB

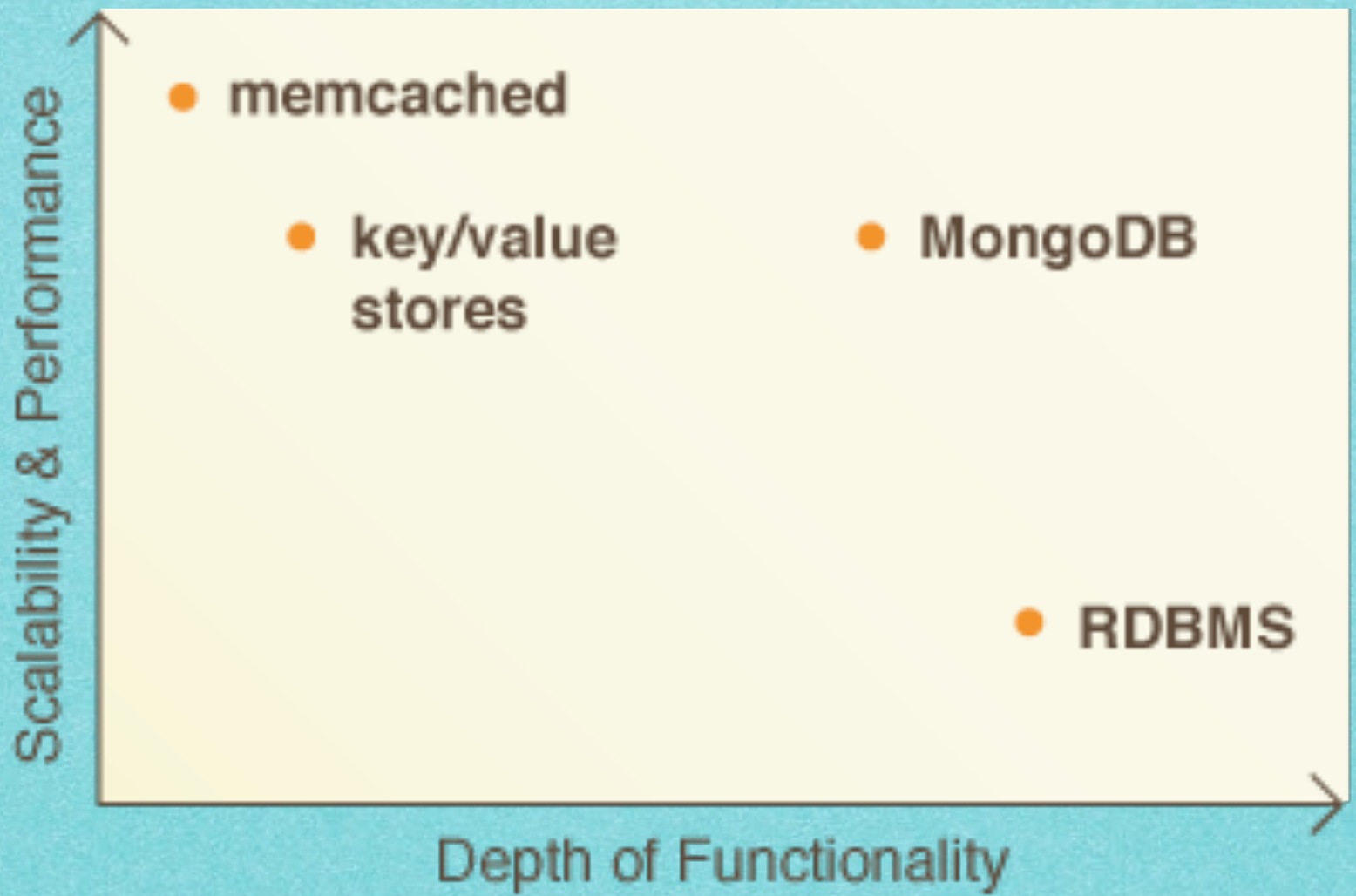
- MongoDB (from "humongous") is a scalable, high-performance, **open source**, **document-oriented** database.
- Current version:2.0.0

What's MongoDB

- **MongoDB=**

JSON + Indexes

Philosophy Of MongoDB



Features

- Document-oriented storage （JSON）
- Full Index Support （Indexes）
- Replication & High availability
- Rich Document-based Queries and Updates
- Map/Reduce
- Auto-Sharding and GridFS

Terminology

RDBMS	MongoDB
Table	Collection
View/Row(s)	JSON Document
Column name	Field name
Index	Index
Join	Embedding & Linking
Partition	Shard
Partition Key	Shard Key

mySQL

```
SELECT
    Dim1, Dim2,
    SUM(Measure1) AS MSum,
    COUNT(*) AS RecordCount,
    AVG(Measure2) AS MAvg,
    MIN(Measure1) AS MMin
    MAX(CASE
      WHEN Measure2 < 100
      THEN Measure2
    END) AS MMax
FROM DenormAggTable
WHERE (Filter1 IN ('A','B'))
    AND (Filter2 = 'C')
    AND (Filter3 > 123)
GROUP BY Dim1, Dim2
HAVING (MMin > 0)
ORDER BY RecordCount DESC
LIMIT 4, 8
```

MongoDB

```
db.runCommand({
mapreduce: "DenormAggCollection",
query: {
    filter1: { '$in': [ 'A', 'B' ] },
    filter2: 'C',
    filter3: { '$gt': 123 }
  },
map: function() { emit(
    { d1: this.Dim1, d2: this.Dim2 },
    { msum: this.measure1, recs: 1, mmin: this.measure1,
      mmax: this.measure2 < 100 ? this.measure2 : 0 }
  );},
reduce: function(key, vals) {
    var ret = { msum: 0, recs: 0, mmin: 0, mmax: 0 };
    for(var i = 0; i < vals.length; i++) {
      ret.msum += vals[i].msum;
      ret.recs += vals[i].recs;
      if(vals[i].mmin < ret.mmin) ret.mmin = vals[i].mmin;
      if((vals[i].mmax < 100) && (vals[i].mmax > ret.mmax))
        ret.mmax = vals[i].mmax;
    }
    return ret;
  },
finalize: function(key, val) {
    val.mavg = val.msum / val.recs;
    return val;
  },
out: 'result1',
verbose: true
});
db.result1.
  find({ mmin: { '$gt': 0 } }).
  sort({ recs: -1 }).
  skip(4).
  limit(8);
```

1. Grouped dimension columns are pulled out as keys in the map function, reducing the size of the working set.
2. Measures must be manually aggregated.
3. Aggregates depending on record counts must wait until finalization.
4. Measures can use procedural logic.
5. Filters have an ORM/ActiveRecord-looking style.
6. Aggregate filtering must be applied to the result set, not in the map/reduce.
7. Ascending: 1; Descending: -1

Revision 4, Created 2010-03-06
Rick Osborne, rickosborne.org

Indexes

- Unique Indexes or Duplicate Values Indexes
- Index of Single Key(Embedded key, Document key) .Default index, ‘_id’: MongoID(GUID)
- Index of Compound Keys(db.user.ensureIndex({credit : -1, name: 1}))
- Sparse Index. (A sparse index can only have one field) //after 1.75

Geo Indexes

- Geospatial index supported (good news for LBS)
- db.c.find({a:[50,50]}) using index {a:'2d'}
- db.c.find({a:{$near:[50,50]}}) using index {a:'2d'}
- Results are sorted closest - farthest
- db.c.find({a:{$within:{$box:[[40,40], [60,60]]}}}) using index {a:'2d'}

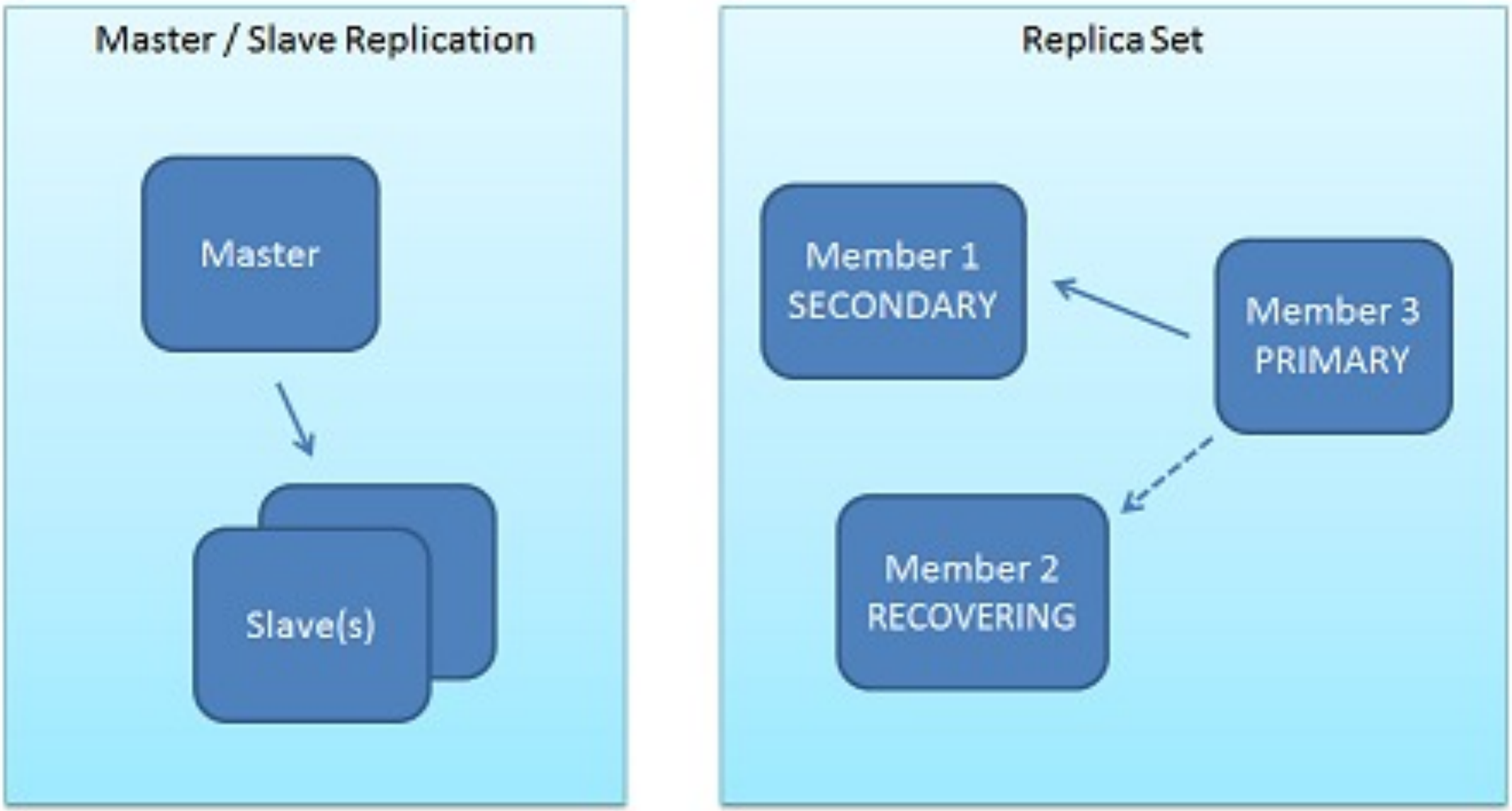
Atomic Operations

- $set $unset $inc
- $push - append a value to an array
- $pushAll - append several values to an array
- $pull - remove a value(s) from an existing array
- $pullAll - remove several value(s) from an existing array
- $bit - bitwise operations

Tips

- Update requires a write lock.
- Write lock is “greedy” right now
- Update in big collection is slow and block other writes during the update
- Query it first then update it will reduce the lock time

Replication



Replica set

- Replica sets are basically master/slave replication, adding automatic failover and automatic recovery of member nodes.
- A Replica Set consists of two or more nodes that are copies of each other. (i.e.: replicas)
- The Replica Set automatically elects a Primary (master) if there is no primary currently available.

Replica set

- Automated fail-over
- Distribute read load
- Simplified maintenance
- A Replica Set can have passive members that cannot become primary (for reading and backup)
- A Replica Set can have delayed nodes (in case of user error)

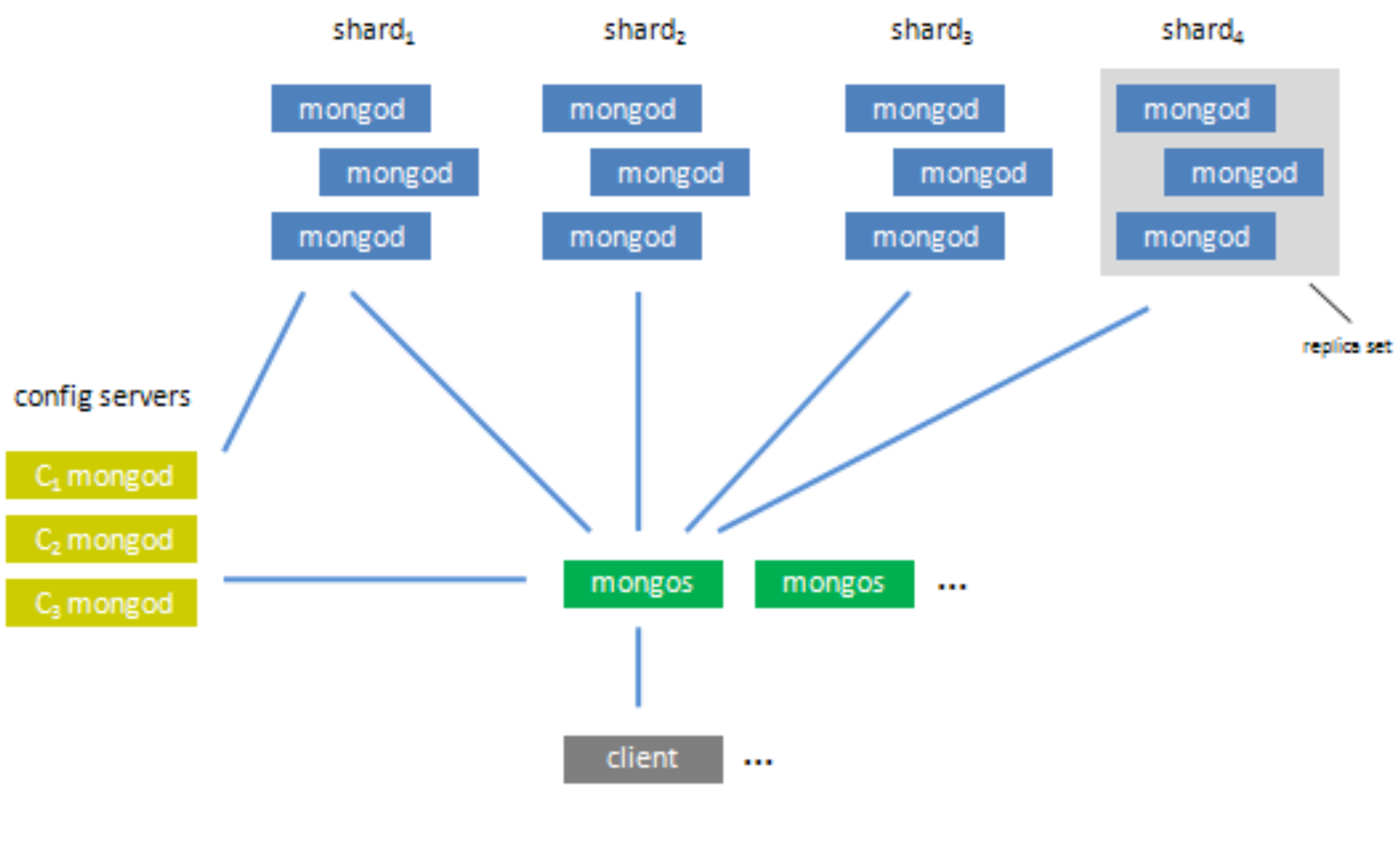
step down primary

- If you step down primary manually when primary is heavy, the replica set may can not elect new primary. (it is fixed in 1.8.0)
- May lose data when step down primary manually. (sometimes, not test in 1.8.0 yet)
- Clients may not know what happened.

What's happened when primary is down?

- Replica set will elect new primary, but how?
- what's arbiter?
- operator time
- votes
- priority
- may lose data when primary is down

Auto-Sharding



Monitoring and Diagnostics

- Query Profiler
- Http Console
- mongostat
- db.stat()
- db.serverStatus()

Database Profiler

- >db.setProfilingLevel(1,20)
  - { "was" : 0, "slowms" : 100, "ok" : 1 }
  - > db.getProfilingStatus() //after1.7x
  - { "was" : 1, "slowms" : 20 }

Database Profiler

- db.system.profile.find()
- {"ts" : "Thu Jan 29 2009 15:19:32 GMT-0500 (EST)" , "info" : "query test.$cmd ntoreturn:1 reslen:66 nscanned:0
query: { profile: 2 } nreturned:1 bytes:50" , "millis" : 0}
- db.system.profile.find().sort({$natural:-1})//To see newest information first

al Shell Edit View Window Help (1:40) 周二 下午

guolijing@guolijingdeMacBook-Pro: ~/work/git/wordnik-oss/src/main/java/com/wordnik/system/mongodb — mongostat — 139×38

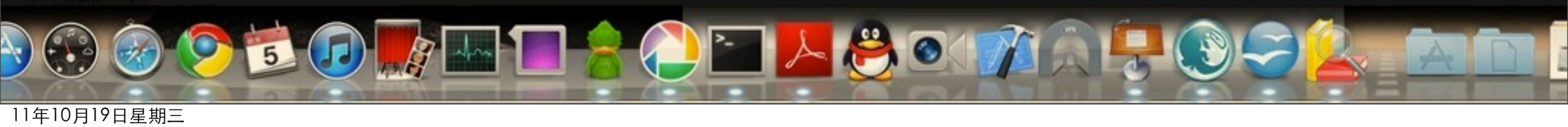
.. mongostat guolijing@g...o: ~ — bash

XXXXXXX

```
ery/s update/s delete/s getmore/s command/s flushes/s mapped  vsize   res faults/s locked % idx miss %     q t|r|w  con
  266       41        2        58        52         0  24214  29447  1545        3      4.3           0     0|0|0   55
  204       16       14        30        21         0  24214  29447  1545        1     27.9           0     0|0|0   55
  123       13        1        20        17         1  24214  29447  1545        4      3.6           0     0|0|0   55
  180        9        2        12        14         0  24214  29447  1545        3      4.4           0     0|0|0   55
  215       14        2        20        18         0  24214  29447  1545        0      4.3           0     0|0|0   55
  303       26        8        41        27         0  24214  29447  1545        0     17.1           0     1|0|1   55
  281       31        4        45        47         0  24214  29447  1545        0      8.4           0     0|0|0   55
  240       30        6        40        36         0  24214  29447  1545        3     12.5           0     0|0|0   55
  217       21        0        30        25         0  24214  29447  1545        1      0.8           0     0|0|0   55
  193        8        0        10        14         0  24214  29447  1545        0        0           0     0|0|0   55
  303       40       20        68        45         0  24214  29447  1545        1     47.1           0     0|0|0   55
  197       14        0        21        23         0  24214  29447  1545        0      0.1           0     0|0|0   55
  236       25       11        35        28         0  24214  29447  1545        1     23.4           0     1|1|0   55
  236       28        6        40        31         0  24214  29447  1545        0     10.3           0     0|0|0   55
  258       29        9        43        28         0  24214  29447  1545       19     22.5           0     0|0|0   55
  195        9        5        17        21         0  24214  29447  1545       16     10.5           0     5|4|1   55
  263       27        9        51        31         0  24214  29447  1546       50     21.1           0     0|0|0   55
```

mongostat

Double-click to edit



11年10月19日星期三

mongostat

- inserts/s - # of inserts per second
- query/s - # of queries per second
- update/s - # of updates per second
- delete/s - # of deletes per second
- getmore/s - # of get mores (cursor batch) per second

mongostat

- command/s - # of commands per second
- flushes/s - # of fsync flushes per second
- mapped - amount of data mmaped (total data size) megabytes
- visze - virtual size of process in megabytes
- res - resident size of process in megabytes

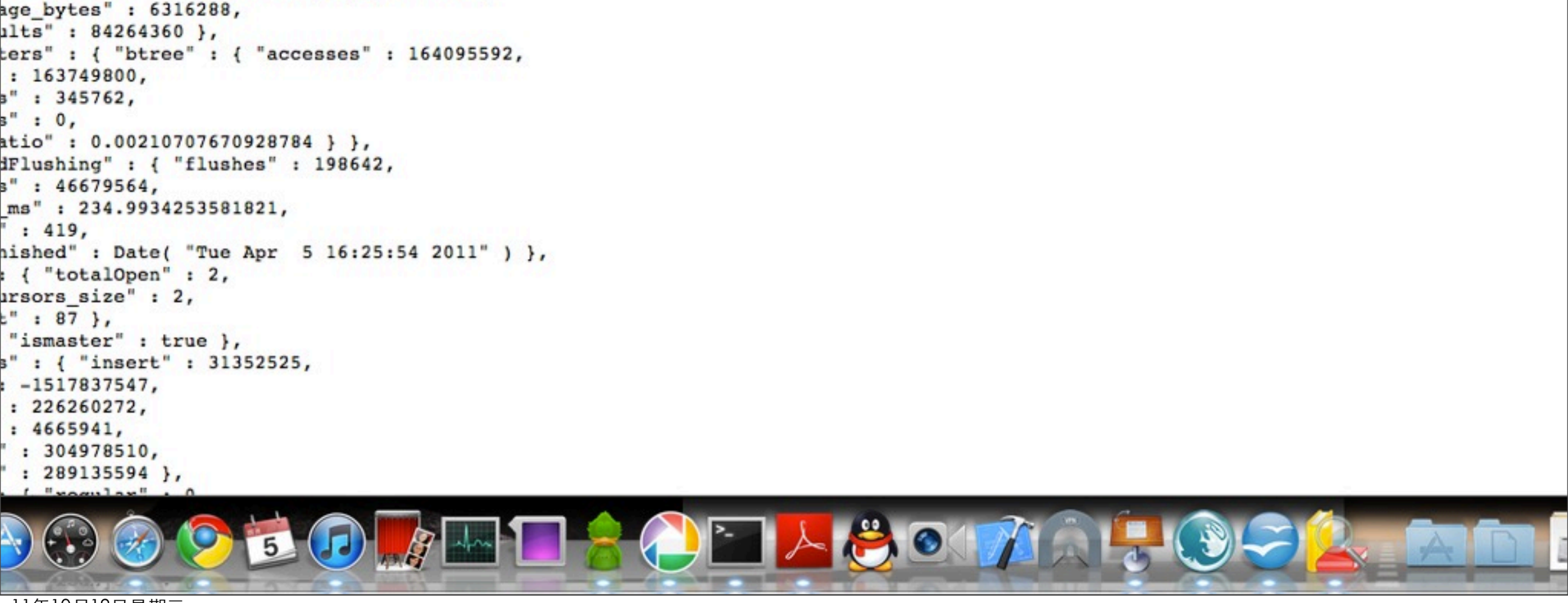
mongostat

- faults/s - # of pages faults/sec (linux only)
- locked - percent of time in global write lock
- idx miss (sampled) - percent of btree page misses
- q t|r|w write) - lock queue lengths (total|read|
- conn - number of open connections

File Edit View History Bookmarks Window Help

10.150.161.16:28017/serverStatus?text


```
: "1.6.4",
 11918600,
imate" : 11750316,
" : Date( "Tue Apr  5 16:26:43 2011" ),
x" : { "totalTime" : 11918600157058,
e" : 142437261871,
: 0.01195083818519166,
Queue" : { "total" : 0,
rs" : 0,
rs" : 0 } },
"bits" : 64,
t" : 1551,
" : 29343,
ed" : true,
 : 24214 },
ns" : { "current" : 540,
le" : 279 },
o" : { "note" : "fields vary by platform",
age_bytes" : 6316288,
ults" : 84264360 },
ters" : { "btree" : { "accesses" : 164095592,
 : 163749800,
s" : 345762,
s" : 0,
atio" : 0.00210707670928784 } },
dFlushing" : { "flushes" : 198642,
s" : 46679564,
_ms" : 234.9934253581821,
" : 419,
nished" : Date( "Tue Apr  5 16:25:54 2011" ) },
: { "totalOpen" : 2,
ursors_size" : 2,
t" : 87 },
 "ismaster" : true },
s" : { "insert" : 31352525,
: -1517837547,
 : 226260272,
 : 4665941,
" : 304978510,
" : 289135594 },
```

MongoDB with Numa

- WARNING: You are running on a NUMA machine.

  We suggest launching mongod like this to avoid performance problems:
  numactl --interleave=all mongod [other options]

Security

- Auth is supported in Replica set/Master-Slaves, Auto-Sharding
- bind_ip
- start auth in RS may have bug.(use http to check the status and try restart mongods)

Memory

- Memory-mapping
- Keep indexes in memory
- db.test.stats()
- db.test.totalIndexSize()
- RAM > indexes + hot data = better performance

some tips

- No transactions
- Fast-N-Loose by default (no safe/w/GLE)
- Indexing order matters; query optimizer helps
- One write thread,many query threads
- Memory Mapped Data Files
- BSON everywhere

Oplog of MongoDB

- Oplog in Replica set:
  - local.system.replset: store config of replica set, use rs.conf() to detail
  - local.oplog.rs: capped collection, use --oplogSize to set size
  - local.replset.minvalid: for sync status
  - NO INDEX

Oplog of MongoDB

- > db.test.insert({'name': 'guolijing', ‘fat’:‘false’})
- > db.oplog.rs.find().sort({$natural:-1})
- { "ts" : { "t" : 1318772440000, "i" : 1 }, "h" : NumberLong("1503388658822904667"), "op" : "i", "ns" : "test.test", "o" : { "_id" : ObjectId("4e9aded8bbf25c4665f212fc"), "name" : "guolijing" , ‘fat’:‘false’} }

Oplog of MongoDB

- {ts:{},h{}, op:{},ns:{},o:{},o2:{} }
  - Ts: 8 bytes of time stamp by 4 bytes Unix timestamp + 4 bytes since the count said.

This value is very important, in the elections (such as master is down unit), new primary would choose the biggest change as the ts new primary.
Op: 1 bytes of operation types, such as I said insert, d said the delete.
In the namespace ns: the operation.
O: the operation is corresponding to the document, that is, the content of the current operation (such as the update operation to update when the fields and value)
O2: perform update operation in the condition is limited to the update, where to have this property
Among them, can be the op several situations:

h:// hash mongodb use to make sure we are reading the right flow of ops and aren't on an out-of-date "fork"

Oplog of MongoDB

- > db.test.update({'name':'guolijing'},{$set:{'fat':'ture'}})
- > db.oplog.rs.find().sort({$natural:-1})
- { "ts" : { "t" : 1318775290000, "i" : 1 }, "h" : NumberLong("-5204953982587889486"), "op" : "u", "ns" : "test.test", "o2" : { "_id" : ObjectId("4e9aded8bbf25c4665f212fc") }, "o" : { "$set" : { "fat" : "ture" } } }

Feature of Oplog

- Replay same oplog is harmless
- Replay old oplogs is harmless if the ts in last one is newer than current database

Back up

- mongoexport mongoimport
- mongodump mongorestore
- data consistency? use --oplog
- use incremental back up
- read oplog and replay (wordnik tools)

Rollback MongoDB

- Use snapshot + oplog
- Use delayed secondary + oplog

Vertical Scala

- Old server 8G RAM +200G dis
- New server 64G RAM + 2T disk
- What can I do if my service should always keep online ?

Vertical Scala

- You're luck if using Replica set
- move **everything(include ip config)** from secondary to new service
- step down primary, do the same thing above
- well, problem solved

Build New Index

- Build a index is quite easy.....
- Build a index in a huge collections, such as 500GB ,and database is very buy, is it easy?

Build New Index

- Take a snapshot of current Mongodb service
- Build new index in snapshot
- replay oplog of working Mongod
- Replace the working Mongod

Best practices

- Replica set: one for primary, one for secondary, one for incremental back up with a arbiter.

Q & A

- www.mongoic.com is online !
- We're hiring...


QCon
北京站·2012年4月18~20日
www.qconbeijing.com（11月启动）
QCon杭州站官网和资料
www.qconhangzhou.com
全球企业开发大会
INTERNATIONAL
SOFTWARE DEVELOPMENT
CONFERENCE