MongoDB Cool Features

Brendan W. McAdams

10gen, Inc.

April 2011 @ MongoPhilly

mongoDB

Cool Features?

- There are lots of cool features in MongoDB. I'm going to discuss just a few.
  - MapReduce
  - GeoSpatial Indexes
  - GridFS
- Richard will discuss some others.
- Sharding, Replica Sets and many other things *are* cool. . . but not part of this talk.

mongoDB

MongoDB MapReduce

- MongoDB's Aggregation Functionality
- Write functions in JavaScript
- Reads from *one* collection, writes to *one* collection.
- Single Threaded per mongod...
  - In a single mongod / replica set environment: No parallelization
  - In sharded environments, one map/reduce is run per shard and re-reduced to combine all results (idempotence)

mongoDB

MongoDB MapReduce

Output Behavior

- Before 1.7.3: MapReduce creates a temporary collection. Can specify permanent collection via ‘out’. Contents of ‘out’ are overwritten after job is finished. Temp collections cleaned up when connection closes.
- Since 1.7.3: Specify ‘outType‘ parameter.
  - ‘normal’ is current behavior.
  - ‘merge’ merges old collection and new results, clobbering any existing keys.
  - ‘reduce’ runs a reduce operation if both new and old contain the same key.

- Sample Data: US Treasury Bond historical Bid Curves since January 1990, to calculate an annual average for the 10 year Treasury.
  - A sample of our dataset:

code/sample_treasury.js

```
{ "_id" : { "$date" : 631238400000 }, "dayOfWeek" :
    "TUESDAY", "bc3Year" : 7.9, "bc5Year" : 7.87,
    "bc10Year" : 7.94, "bc20Year" : null, "bc1Month" :
    null, "bc2Year" : 7.87, "bc3Month" : 7.83, "bc30Year"
    : 8, "bc1Year" : 7.81, "bc7Year" : 7.98, "bc6Month" :
    7.89 }
```

MongoDB MapReduce II

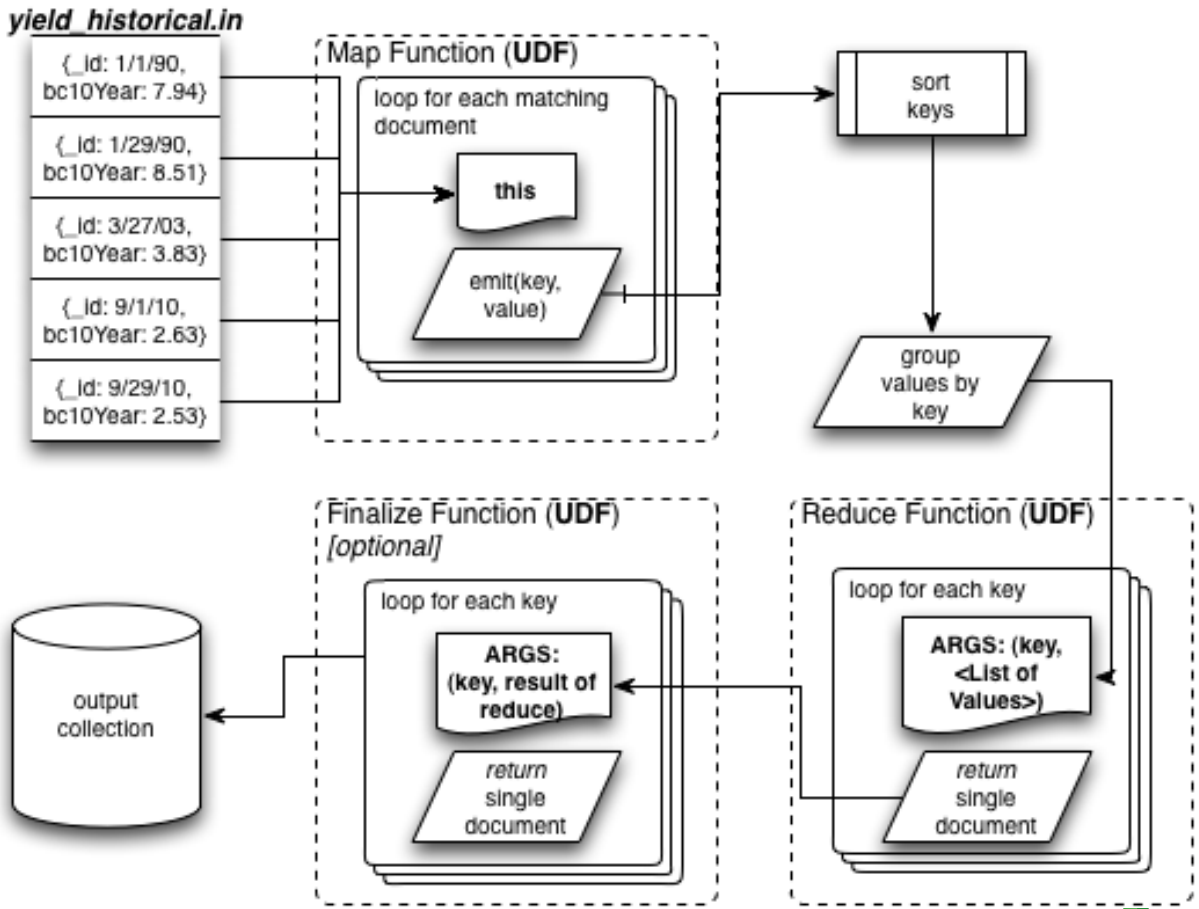
Running a MapReduce


```
{ "_id" : { "$date" : 631324800000 }, "dayOfWeek" :
    "WEDNESDAY", "bc3Year" : 7.96, "bc5Year" : 7.92,
    "bc10Year" : 7.99, "bc20Year" : null, "bc1Month" :
    null, "bc2Year" : 7.94, "bc3Month" : 7.89, "bc30Year"
    : 8.039999999999999, "bc1Year" : 7.85, "bc7Year" :
    8.039999999999999, "bc6Month" : 7.94 }
{ "_id" : { "$date" : 631411200000 }, "dayOfWeek" :
    "THURSDAY", "bc3Year" : 7.93, "bc5Year" : 7.91,
    "bc10Year" : 7.98, "bc20Year" : null, "bc1Month" :
    null, "bc2Year" : 7.92, "bc3Month" : 7.84, "bc30Year"
    : 8.039999999999999, "bc1Year" : 7.82, "bc7Year" :
    8.02, "bc6Month" : 7.9 }
{ "_id" : { "$date" : 631497600000 }, "dayOfWeek" :
    "FRIDAY", "bc3Year" : 7.94, "bc5Year" : 7.92,
    "bc10Year" : 7.99, "bc20Year" : null, "bc1Month" :
    null, "bc2Year" : 7.9, "bc3Month" : 7.79, "bc30Year" :
    8.06, "bc1Year" : 7.79, "bc7Year" : 8.029999999999999,
    "bc6Month" : 7.85 }
```


MongoDB MapReduce III

Running a MapReduce

- Job Anatomy (Single Server):



MongoDB MapReduce IV

Running a MapReduce

- The MongoDB JavaScript mapReduce:

code/mongo_treasury_mr.js

```
function m() {
    emit( this._id.getYear(), { count: 1, sum:
    this.bc10Year })
}

function r( year, values ) {
    var n = { count: 0,  sum: 0 }
    for ( var i = 0; i < values.length; i++ ){
        n.sum += values[i].sum;
        n.count += values[i].count;
    }

    return n;
}

function f( year, value ){
```

MongoDB MapReduce V

Running a MapReduce

```
    value.avg = value.sum / value.count;
    return value.avg;
}

result = db.yield_historical.in.mapReduce( m , r, {
    finalize: f });
```

- Job Output:

code/mongo_mr_out.js

```
{
    "result" : "tmp.mr.mapreduce_1291414680_16",
    "timeMillis" : 524,
    "counts" : {
        "input" : 5193,
        "emit" : 5193,
        "output" : 21
    },
    "ok" : 1,
}
```

MongoDB MapReduce VII

Running a MapReduce

- A bit more verbosely:

code/mongo_mr_out_verbose.js

```
> db.yield_historical.in.mapReduce( m , r, { finalize: f });
{
    "result" : "tmp.mr.mapreduce_1291414803_17",
    "timeMillis" : 389,
    "timing" : {
        "mapTime" : NumberLong(275),
        "emitLoop" : 345,
        "total" : 389
    },
    "counts" : {
        "input" : 5193,
        "emit" : 5193,
        "output" : 21
    },
    "ok" : 1,
}
```

MongoDB MapReduce VIII

Running a MapReduce

- Read the collection for your results:

code/mongo_mr_out_read.js

```
> db.tmp.mr.mapreduce_1291414803_17.find()
{ "_id" : 90, "value" : 8.552400000000002 }
{ "_id" : 91, "value" : 7.862360000000025 }
{ "_id" : 92, "value" : 7.008844621513946 }
{ "_id" : 93, "value" : 5.866279999999999 }
{ "_id" : 94, "value" : 7.085180722891565 }
{ "_id" : 95, "value" : 6.573920000000002 }
{ "_id" : 96, "value" : 6.443531746031743 }
{ "_id" : 97, "value" : 6.353959999999992 }
{ "_id" : 98, "value" : 5.262879999999994 }
{ "_id" : 99, "value" : 5.646135458167332 }
{ "_id" : 100, "value" : 6.030278884462145 }
{ "_id" : 101, "value" : 5.020685483870969 }
{ "_id" : 102, "value" : 4.61308 }
{ "_id" : 103, "value" : 4.013879999999999 }
{ "_id" : 104, "value" : 4.271320000000004 }
```

MongoDB MapReduce IX

Running a MapReduce

```
{ "_id" : 105, "value" : 4.28888000000001 }
{ "_id" : 106, "value" : 4.7949999999999955 }
{ "_id" : 107, "value" : 4.634661354581674 }
{ "_id" : 108, "value" : 3.6642629482071714 }
{ "_id" : 109, "value" : 3.264120000000037 }
has more
```

- It's possible to specify a query, sort and limit as well, to limit your input.

GeoSpatial Indexing I

- Search by Geospatial proximity with MongoDB...
- One Geoindex allowed per collection
- Index can be created on an array or a subdocument
- You must be consistent across all documents (e.g. same key names or order in array)
- I loaded the publicly available GTFS data for EPTA's Rail and Bus services.
- Quick & Dirty Python script to create the index:

GeoSpatial Indexing II

code/geospatial_idx.py

```
import pymongo
from pymongo import Connection

if float(pymongo.version) < 1.6:
    raise Exception("ERROR: This script requires PyMongo
    Version 1.6 or greater.")

connection = Connection()
db = connection['septa']
print "Indexing the Stops Data."
for row in db.stops.find():
    row['stop_geo'] = {'lat': float(row['stop_lat']), 'lon':
    float(row['stop_lon'])}
    db.stops.save(row)

db.stops.ensure_index([('stop_geo', pymongo.GEO2D)])
print "Reindexed stops with Geospatial data."

print "Indexing the Shapes data"
```

GeoSpatial Indexing III

```
for row in db.shapes.find():
    row['shape_pt_geo'] = {'lat': float(row['shape_pt_lat']),
                           'lon': float(row['shape_pt_lon'])}
    db.shapes.save(row)

db.shapes.ensure_index([('shape_pt_geo', pymongo.GEO2D)])
print "Reindexed shapes with Geospatial data."

print "Done."
```

mongoDB

GeoSpatial Indexing IV

- What are the 5 nearest SEPTA stops to our current location ('39.946332, -75.144009')?

code/geospatial.js

```
> db.stops.find({"stop_geo": {$near: [39.946332, -75.144009]}},
...             {"stop_name": 1, "stop_desc": 1}).limit(10)

{ "_id" : ObjectId("4db712934892e28e4f72cb78"), "stop_name" :
    "2nd St & Walnut St" }
{ "_id" : ObjectId("4db712934892e28e4f72cf36"), "stop_name" :
    "Dock St & Dock St" }
{ "_id" : ObjectId("4db712934892e28e4f72cb75"), "stop_name" :
    "Chestnut St & 2nd St" }
{ "_id" : ObjectId("4db712924892e28e4f72b43c"), "stop_name" :
    "Spruce St & 2nd St " }
{ "_id" : ObjectId("4db712934892e28e4f72cb79"), "stop_name" :
    "2nd St & Chestnut St" }
{ "_id" : ObjectId("4db712924892e28e4f72ba14"), "stop_name" :
    "Chestnut St & Front St " }
```

GeoSpatial Indexing V

```
{ "_id" : ObjectId("4db712934892e28e4f72c17c"), "stop_name" :
    "3rd St & Walnut St" }
{ "_id" : ObjectId("4db712934892e28e4f72d5ea"), "stop_name" :
    "Columbus Blvd & Walnut St  " }
{ "_id" : ObjectId("4db712934892e28e4f72d620"), "stop_name" :
    "Columbus Blvd & Spruce St " }
{ "_id" : ObjectId("4db712934892e28e4f72cb76"), "stop_name" :
    "Walnut St & 3rd St - FS" }
```

mongoDB

GeoSpatial Indexing VI

- It is possible to use a secondary key as well to further filter. The GTFS data is designed more relationally, but if it had a "train" or "bus" field...

code/geospatial_sec.js

```
> db.stops.ensureIndex({"location": "2d", "type": 1});

> db.stops.find({"stop_geo": {$near: [39.946332, -75.144009],
                              "type": "bus"}});
/**
 *  Hypothetical results would include only bus stops
 */
```

GeoSpatial Indexing VII

- '$near' queries can also take '$maxDistance' parameter which limits the search area.

code/geospatial_max.js

```
> db.stops.find({"stop_geo": {$near: [39.946332, -75.144009],
                              $maxDistance: 5})
```

- '$near' queries just get you the 'closest'. You can use '$within' to specify a shape such as '$box' (rectangle), '$center' (circle), '$polygon' (concave and convex polygons).

GeoSpatial Indexing VIII

- By default, treats things like a Map (flat)... but 1.8 has support for treating things like a sphere (e.g. the earth)
- In production use at Foursquare & Wordsquared (Formerly Scrabb.ly)
- It doesn't HAVE To be map coordinates. Scrabb.ly tracks the coordinates of infinite scrabble boards and the tiles' relation to 0, 0

GridFS: Scalable MongoDB File Storage I

- Specification for storing large files in MongoDB, supported in all official drivers as reference implementation.
- Works around 4MB BSON limit by breaking files into chunks.
- Two collections: ‘fs.files’ for metadata, ‘fs.chunks’ stores the individual file chunks.
- Sharding: Individual file chunks don’t shard but the files themselves will (e.g. File A goes on Server 1, File B goes on Server 2 but no chunks of A will be on 2)
- Experimental modules for Lighttpd and Nginx to serve static files directly from GridFS
- A Unit Test from Casbah (Scala Driver):

GridFS: Scalable MongoDB File Storage II

code/gridfs_spec.scala

```
package com.mongodb.casbah
package test

import com.mongodb.casbah.gridfs.Imports._

import java.security.MessageDigest
import java.io._

import org.specs._
import org.specs.specification.PendingUntilFixed

class GridFSSpec extends Specification with PendingUntilFixed {
  val logo_md5 = "479977b85391a88bbc1da1e9f5175239"
  val digest = MessageDigest.getInstance("MD5")


  "Casbah's GridFS Implementations" should {
    shareVariables()
    implicit val mongo = MongoConnection()("casbah_test")
```

GridFS: Scalable MongoDB File Storage III

```
mongo.dropDatabase()
val logo = new
FileInputStream("casbah-gridfs/src/test/resources/powered_by_mon
val gridfs = GridFS(mongo)

"Correctly save a file to GridFS" in {
  gridfs must notBeNull
  logo must notBeNull

  gridfs(logo) { fh =>
    fh.filename = "powered_by_mongo.png"
    fh.contentType = "image/png"
  }

}

"Find the file in GridFS later" in {
  val file = gridfs.findOne("powered_by_mongo.png")
  file must notBeNull
  file must haveSuperClass[GridFSDBFile]
  file.md5 must beEqualTo(logo_md5)
  println(file.md5)
```

GridFS: Scalable MongoDB File Storage IV

```
    }

  }

}


// vim: set ts=2 sw=2 sts=2 et:
```

- See the GridFS Spec...`http://www.mongodb.org/display/DOCS/GridFS+Specification`

Questions?

- Twitter: **@rit** | mongodb: **@mongodb** | 10gen: **@10gen**
- email: **brendan@10gen.com**
- Pressing Questions?
  - IRC - freenode.net **#mongodb**
  - MongoDB Users List - `http://groups.google.com/group/mongodb-user`
- 10gen is *hiring!* We need smart engineers in both NY and Bay Area: `http://10gen.com/jobs`

mongoDB