#codemash

MongoDB – Zero to Sharding

Sridhar Nanjundeswaran

Engineer, MongoDB Inc.

@snanjund

mongoDB

Workshop Agenda

- Introduction and building your first app
- Schema Design and Indexing
- Replication and Sharding
- Deployment ?

Part 1 – Introduction and building your first app

This Talk

- Quick introduction to mongoDB
- Data modeling in mongoDB, queries, geospatial, updates and map reduce.
- Using a location-based app as an example
- https://github.com/sridharn/codemash_2014/tree/master/firstapp

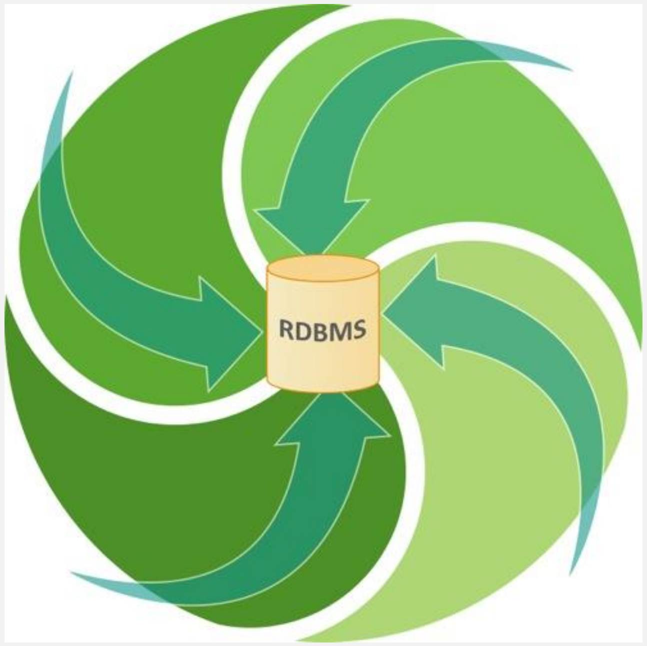
Relational Database Challenges

Data Types

- Unstructured data
- Semi-structured data
- Polymorphic data

Agile Development

- Iterative
- Short development cycles
- New workloads



Volume of Data

- Petabytes of data
- Trillions of records
- Tens of millions of queries per second

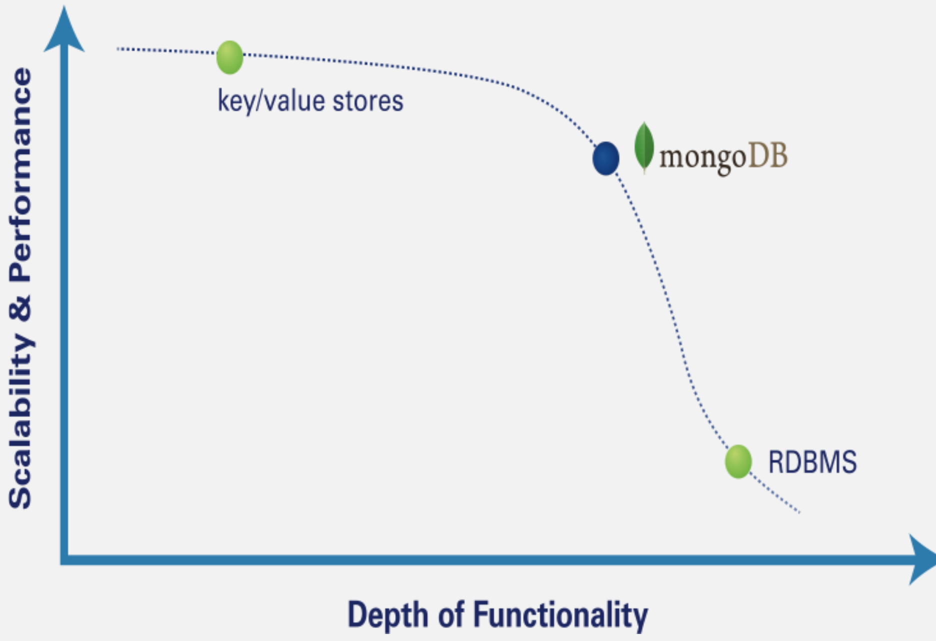
New Architectures

- Horizontal scaling
- Commodity servers
- Cloud computing

What is mongoDB?

MongoDB is a scalable, high-performance, open source, document database.

- Fast Querying
- In-place updates
- Full Index Support
- Replication /High Availability
- Auto-Sharding
- Aggregation; Map/Reduce
- DBMS of the year 2013 per DB-Engines.com ☺



Document database?

- JSON - json.org
  - { “name” : “MongoDB” }
- BSON - bsonspec.org
  - “\x16\x00\x00\x00\x02_id\x00\x08\x00\x00\x00MongoDB\x00\x00”
- BSON
  - Storage format
  - In wire protocol

Why BSON?

- JSON has powerful, limited set of datatypes
  - Mongo extends datatypes with Date, Int types, ObjectId,
- BSON is a binary representation of JSON
  - Optimized for performance and navigational abilities
  - Also compression

Where can you use it?

- MongoDB is Implemented in C++
- Windows, Linux, Mac OS-X, Solaris
- Packages available
  - OS X – Macports, Homebrew
  - Linux – Debian, Ubuntu, Fedora, CentOS…
  - Windows – MSIs (coming soon)

How can I connect to it?

Official MongoDB drivers

Microsoft® .NET™

C++

php

C

Scala

nodeJS

MongoDB Drivers

- Official Support for 10 languages
- Community drivers for tons more
  - R, lua etc.
- Drivers connect to mongo servers
- Drivers translate BSON into native types
- mongo shell is not a driver, but works like one in some ways
- Installed using typical means (npm, pecl, gem, pip)

Terminology

RDBMS	MongoDB
Table/View	Collection
Row(s)	Document
Index	Index
Partition	Shard
Join	Embedding/Linking
Fixed Schema	Flexible/Implied Schema

Example Document

```
{
      _id : ObjectId("4c4ba5c0672c685e5e8aabf3"),
      author : “Sridhar",
      date : ISODate("2014-01-02T11:52:27.442Z"),
      text : "About MongoDB...",
      tags : [ "tech", "databases", "nosql" ],
      comments : [{
          author : "Doug",
       date : ISODate("2014-01-03T17:22:21.124Z"), text
: "Best Post Ever!"
      }],
     comment_count : 1
 }
```

Why use mongoDB?

- Intrinsic support for fast, iterative development
- Super low latency access to your data
- Very little CPU overhead
- No additional caching layer required
- Built in replication and horizontal scaling support

Building your first app

Install MongoDB

1. Download
2. Unzip
3. Create data directory
4. Run mongod

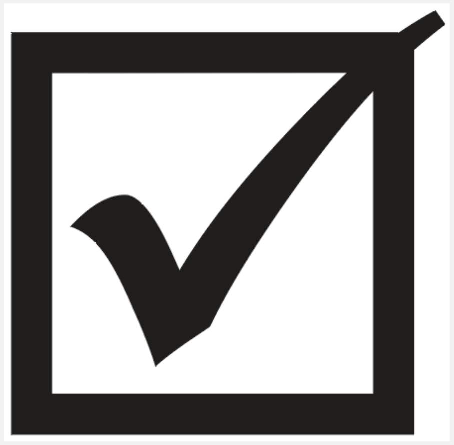
Sample scripts at

https://github.com/sridharn/codemash_2014/tree/master/firstapp

Yes it is as simple as that ☺

Building Your First MongoDB App

- Want to build an app where users can check in to a location



- Leave notes or comments about that location

Requirements

- "As a user I want to be able to find other locations nearby"

- Need to store locations (Offices, Restaurants, etc)
  - name, address, tags
  - coordinates
  - User generated content e.g. tips / notes

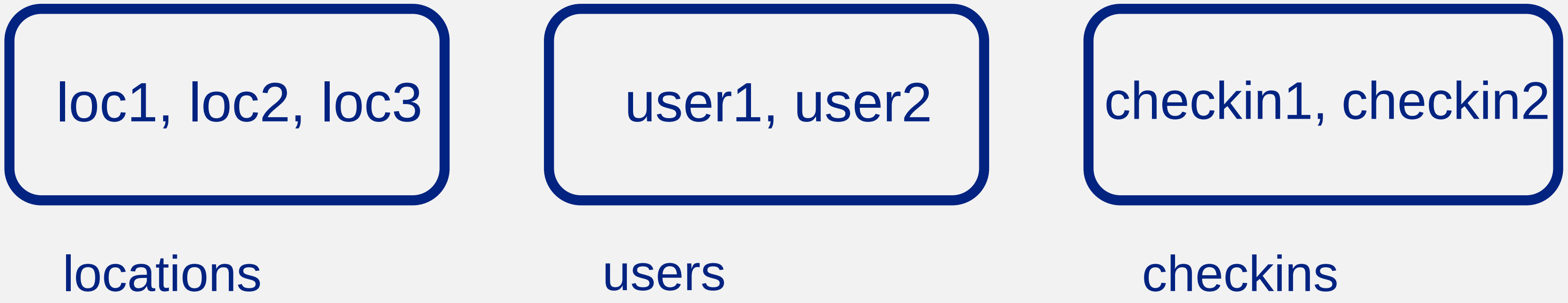
Requirements

"As a user I want to be able to 'checkin' to a location"

Checkins

- User should be able to 'check in' to a location
- Want to be able to generate statistics:
  - Recent checkins
  - Popular locations

Collections



Locations v1

```
> location_1 = {
      name: "Taj Mahal",
       address: "123 University Ave",
       city: "Palo Alto",
       zipcode: 94301
  }
```

Locations v1

```
> location_1 = {
      name: "Taj Mahal",
       address: "123 University Ave",
       city: "Palo Alto",
       zipcode: 94301
  }

> db.locations.insert(location_1)
```

Locations v1

```
> db.locations.findOne({name: "Taj Mahal"})

{
    "_id" : ObjectId("50e67a4f4b23019a4ab9b58c"),
    "name" : "Taj Mahal",
    "address" : "123 University Ave",
    "city" : "Palo Alto",
    "zipcode" : 94301
}
```

What is _id?

- _id is the primary key in MongoDB
- Automatically indexed
- Automatically created as an ObjectId if not provided
- Any unique immutable value could be used

What is ObjectId?

- ObjectId is a special 12 byte value
- Guaranteed to be unique across your cluster
- ObjectId("`50e67a4f4b23019a4ab9b58c`")

```
|-------------||---------||-----||----------|
       ts          mac      pid      inc
```

Locations v1 – Indexed find

```
> db.locations.ensureIndex({name: 1})


> db.locations.find({name: "Taj
Mahal"}).explain()

{

   "cursor" : "BtreeCursor name_1",

   "isMultiKey" : false,

   …
```

Locations v2

```
> location_2 = {
      name: "Lotus Flower",
       address: "234 University Ave",
       city: "Palo Alto",
       zipcode: 94301,
       tags: ["restaurant", "dumplings"]
  }


> db.locations.insert(location_2)
```

Locations v2

```
> db.locations.findOne({tags: "dumplings"})
{
    "_id" : ObjectId("50e67f334b23019a4ab9b59a"),
    "name" : "Lotus Flower",
    ….
> db.locations.ensureIndex({tags: 1})
> db.locations.find({tags: "dumplings"}).explain()
{
    "cursor" : "BtreeCursor tags_1",
    "isMultiKey" : true,
    …
```

Locations v3

```
> location_3 = {
      name: "El Capitan",
       address: "345 University Ave",
       city: "Palo Alto",
       zipcode: 94301,
       tags: ["restaurant", "tacos"],
       lat_long: [52.5184, 13.387]
  }

> db.locations.insert(location_3)
```

Locations v3

```
> db.locations.find({lat_long: {$near:[52.53, 13.4]}})
error: {
    "$err" : "can't find special index: 2d for: {
lat_long: { $near: [ 52.53, 13.4 ] } }",
    "code" : 13038
}
> db.locations.ensureIndex({lat_long: "2d"})
> db.locations.findOne({lat_long: {$near:[52.53,
13.4]}})
{
    "_id" : ObjectId("50e686ab4b23019a4ab9b59d"),
    "name" : "El Capitan",
    ...
```

Finding locations

```
// creating your indexes:
> db.locations.ensureIndex({tags: 1})
> db.locations.ensureIndex({name: 1})
> db.locations.ensureIndex({lat_long: "2d"})
// finding places:
> db.locations.find({lat_long: {$near:[52.53, 13.4]}})

// with regular expressions:
> db.locations.find({name: /^Taj/})
//  by tag:
> db.locations.find({tag: "dumplings"})
```

Updating Documents

Atomic operators:

$set, $unset, $inc, $push, $pushAll, $pull, $pullAll

Locations - adding tips

// adding a tip with update:

```
> db.locations.update(
    {name: "Lotus Flower"},
    {$push: {
      tips: {
        user: “Sridhar",
        date: ISODate("2012-09-21T11:52:27.442Z"),
»         tip: "The sesame dumplings are 
awesome!"}
»   }})
```

task - done

```
> db.locations.findOne({name:/^Lot/})
{
    "_id" : ObjectId("50e67f334b23019a4ab9b59a"),
    "address" : "234 University Ave",
    "city" : "Palo Alto",
    "name" : "Lotus Flower",
    "tags" : [
        "restaurant",
        "dumplings"
    ],
    "tips" : [
        {
            "user" : "Sridhar",
            "date" : ISODate("2012-09-21T11:52:27.442Z"),
            "tip" : "The sesame dumplings are awesome!"
        }
    ],
    "zipcode" : 94301
}
```

Requirements

"As a user I want to be able to 'checkin' to a location"

Checkins

- User should be able to 'check in' to a location
- Want to be able to generate statistics:
  - Recent checkins
  - Popular locations

Users and Checkins

```
> user_1 = {
      _id: "sridhar@10gen.com",
      name: "Sridhar",
      twitter: "snanjund",
      checkins: [
       {location: "Lotus Flower", ts: ISODate("2012-09-
21T11:52:27.442Z")},
       {location: "Taj Mahal", ts: ISODate("2012-09-
22T07:15:00.442Z")}
      ]
  }

> db.users.save(user_1)

> db.users.ensureIndex({"checkins.location": 1})
```

Simple Stats

```
//  find all users who've checked in here:

> db.users.find({"checkins.location":"Lotus Flower"},
{name:1, checkins:1})


//  find the last 10 checkins here:

> db.users.find({"checkins.location":"Lotus Flower"},
{name:1, checkins:1}).sort({"checkins.ts": -
1}).limit(10)
```

Hard to query for last 10

User and Checkins v2

```
> user_1 = {
      _id: "sridhar@10gen.com",
     name: "Sridhar",
      twitter: "snanjund",
  }
> location_id = db.locations.findOne({name:"Taj
Mahal"}, {_id:1})["_id"]
> checkin_1 = {
      location: location_id,
      user: "sridhar@10gen.com",
      ts: ISODate("2012-09-21T11:52:27.442Z")
  }
```

Simple Stats

```
// find all users who've checked in here:
> location_id = db.locations.find({"name":"Lotus Flower"})
> u_ids = db.checkins.find({location: location_id},
                           {_id: -1, user: 1})

> users = db.users.find({_id: {$in: u_ids}})
// find the last 10 checkins here:

> db.checkins.find({location: location_id})

                  .sort({ts: -1}).limit(10)
// count how many checked in today:

> db.checkins.find({location: location_id,

                    ts: {$gt: midnight}}

               ).count()
```

Aggregation- in Mongo 2.2+

```
// Find most popular locations
> agg = db.checkins.aggregate(
    {$match: {ts: {$gt: now_minus_3_hrs}}},
    {$group: {_id: "$location", numEntries:
{$sum: 1}}}
  )

> agg.result
  [{"_id": "Lotus Flower", "numEntries" : 17}]
```

Map Reduce

```
// Find most popular locations
> map_func = function() {
    emit(this.location, 1);
  }

> reduce_func = function(key, values) {
    return Array.sum(values);
  }

> db.checkins.mapReduce(map_func, reduce_func,
    {query: {ts: {$gt: now_minus_3_hrs}},
     out: "result"})

> db.result.findOne()
  {"_id": "Lotus Flower", "value" : 17}
```

#codemash

Coming Next:
Part 2 : Schema Design and Indexing

Sridhar Nanjundeswaran

Engineer, MongoDB Inc.

@snanjund

mongoDB