# MongoDB – A hands on Intro

Sridhar Nanjundeswaran
Technical Lead, 10Gen
@snanjund
March 12, 2013

mongoDB

10gen | the MongoDB company

# This Talk

- Quick introduction to mongoDB
- Hands on mongoDB
  - Using a location-based app as an example
  - https://github.com/sridharn/sfbayazure_2013_03_12
- Deployment to Windows Azure
- 2.4 new features

# Why MongoDB? What is it?

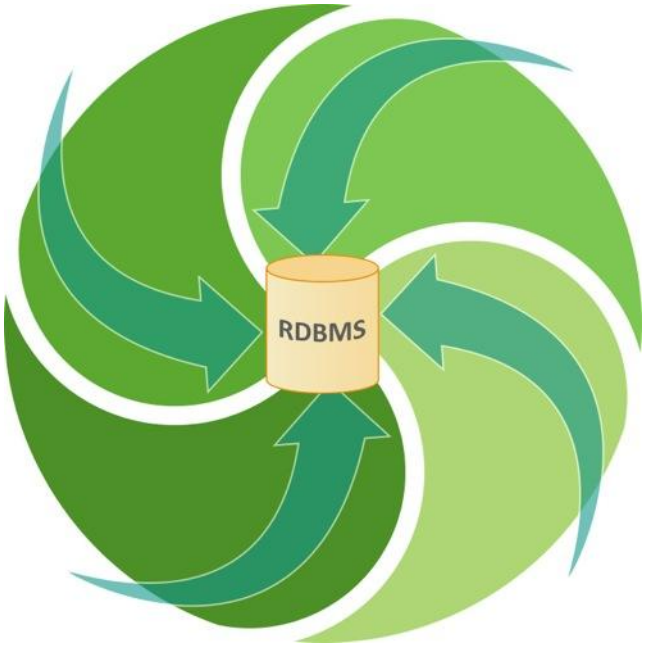
# Relational Database Challenges

**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

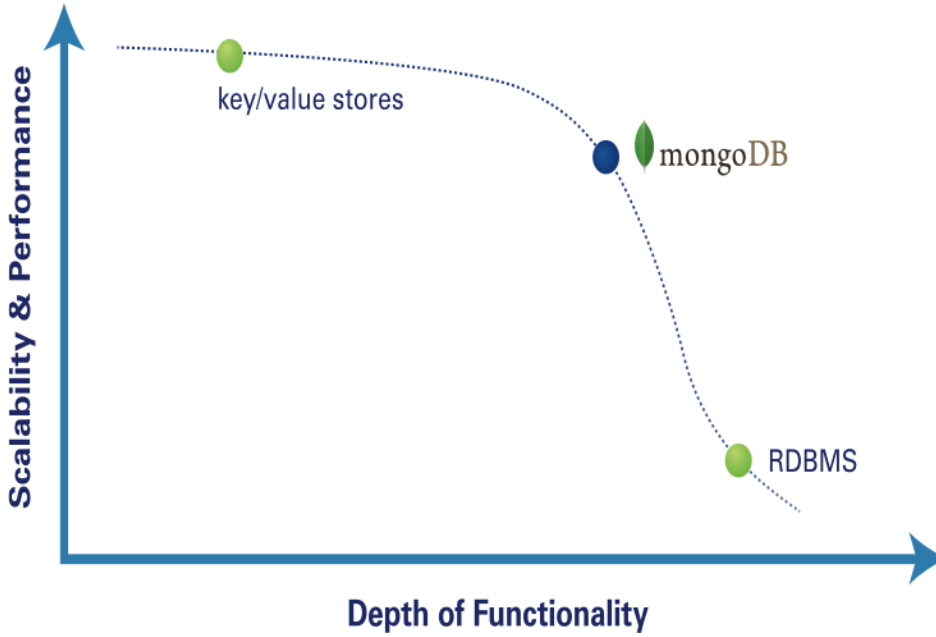
**New Architectures**

- 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
- GridFS



# BSON

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

See: bsonspec.org

# 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…

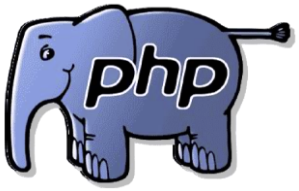
# How can I connect to it?

## Official 10Gen drivers









# MongoDB Drivers

- Official Support for 13 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 (nuget, npm, pecl, gem, pip)

# Terminology

| RDBMS | MongoDB |
|---|---|
| Table | Collection |
| Row(s) | Document |
| Index | Index |
| Partition | Shard |
| Join | Embedding/Linking |
| Fixed Schema | Dynamic Schema |

# 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

| loc1, loc2, loc3 | user1, user2 | checkin1, checkin2 |
| --- | --- | --- |
| locations | users | checkins |

# Install

- Install DB
- Startup
- Shell
- Code at https://github.com/sridharn/sfbayazure_2013_03_12

# 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)

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

# 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, $bit

# Inserting 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"})
```

# 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)
```

# 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: - Warning!

```
> 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

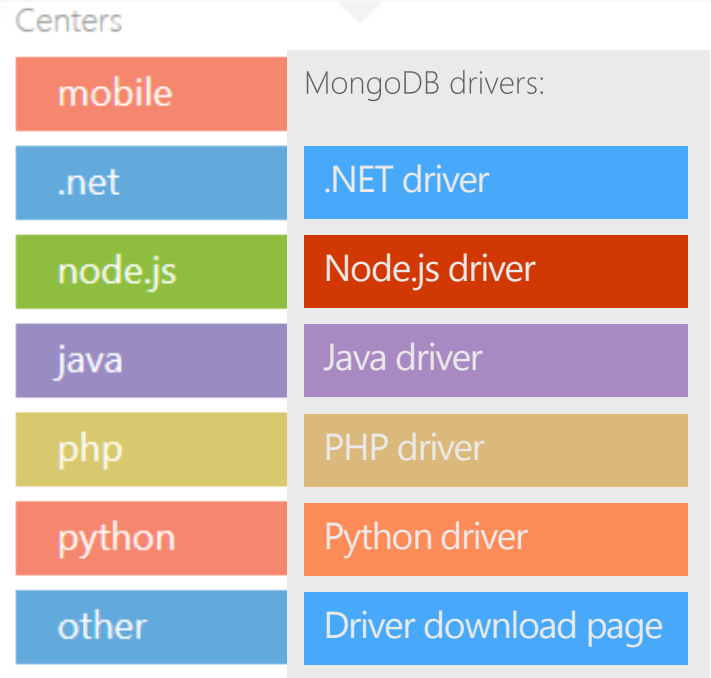
```
// Find most popular locations
> map_func = function() {
    emit(this.location, 1);
  }

> 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}
```

# Deployment

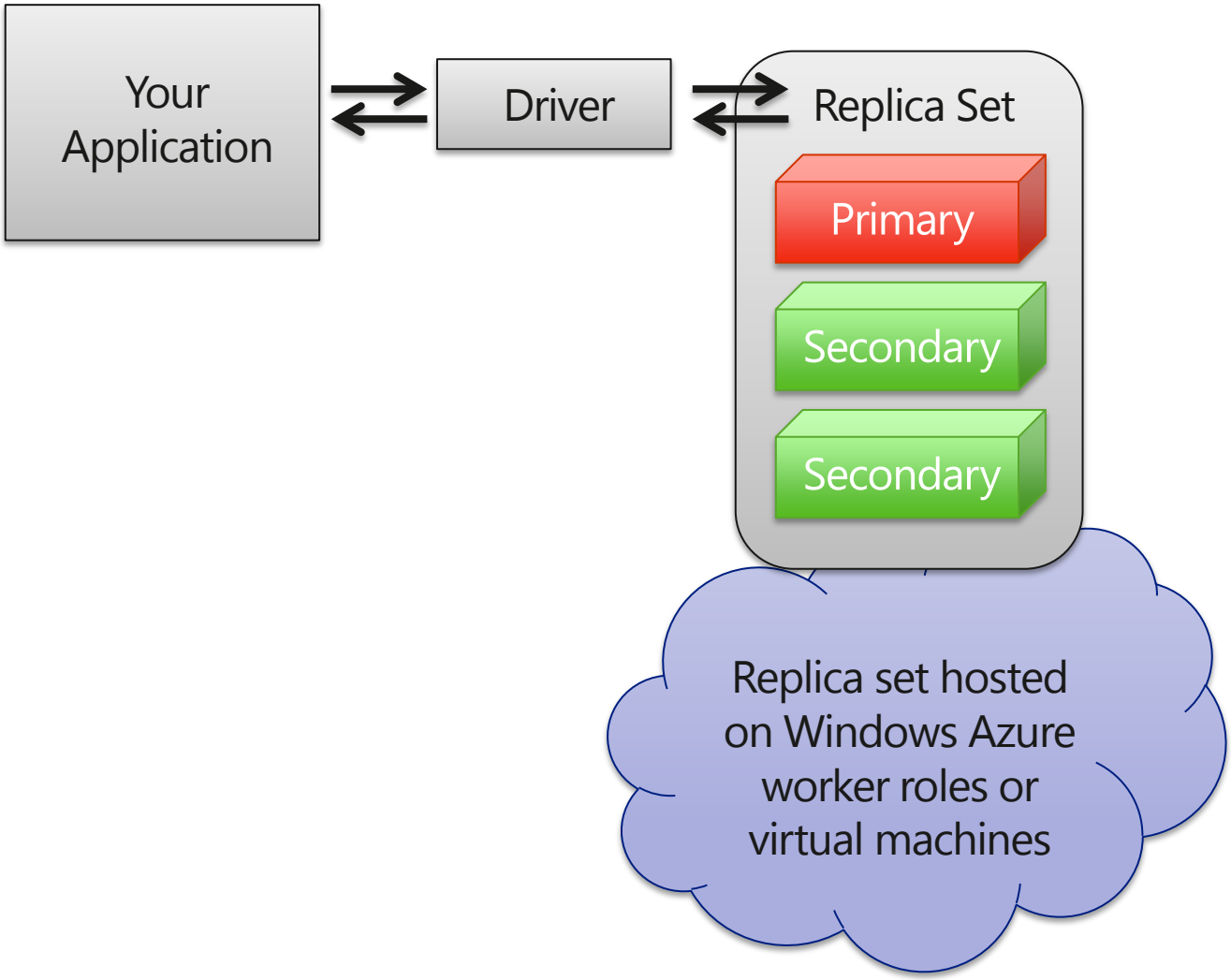
# MongoDB and Windows Azure

- Choice of popular programming languages

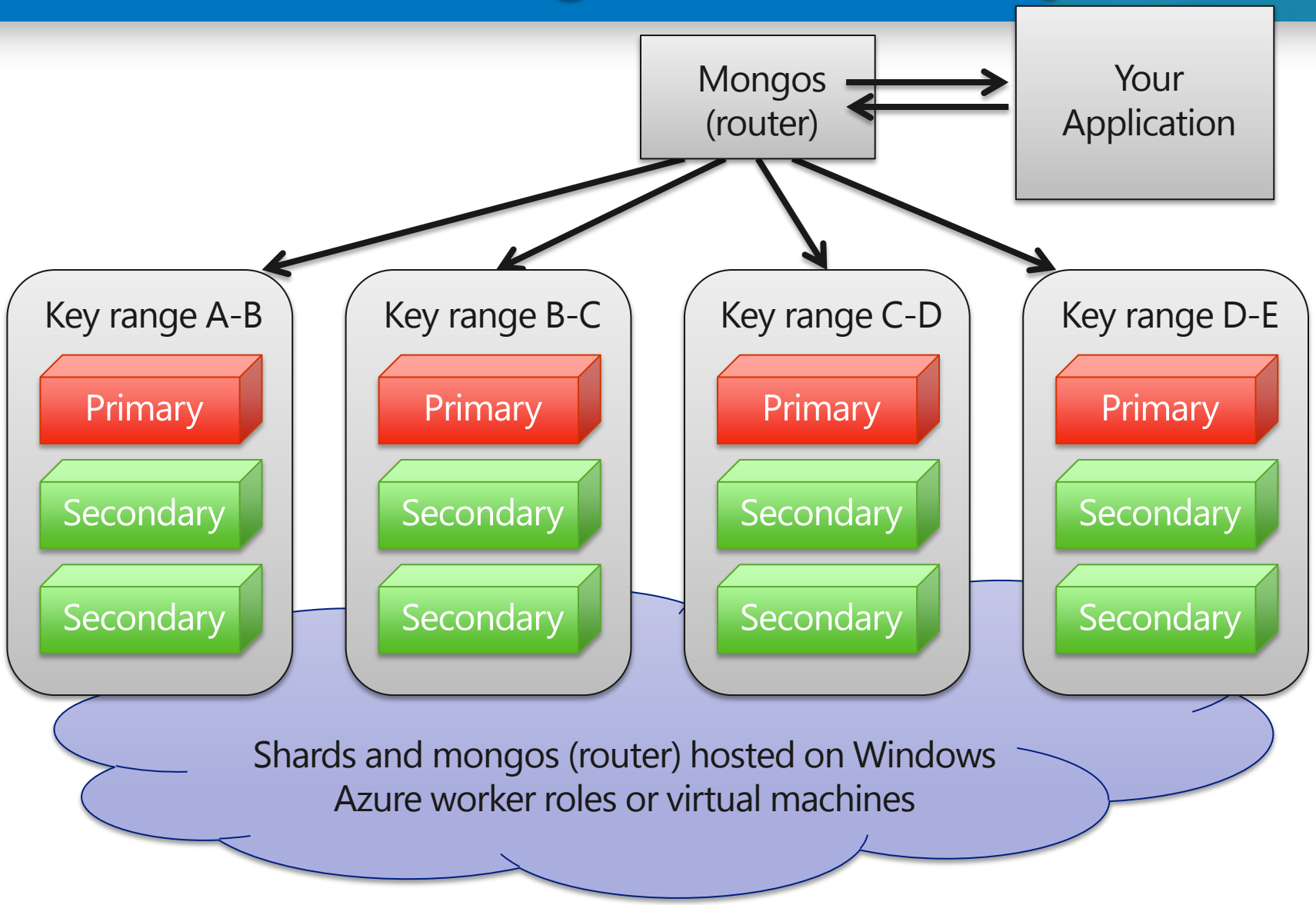
- Deploy on Cloud Services or Virtual Machines



# Replica Sets – High availability



# Sharding - Scalability



# Virtual Machine Sizes

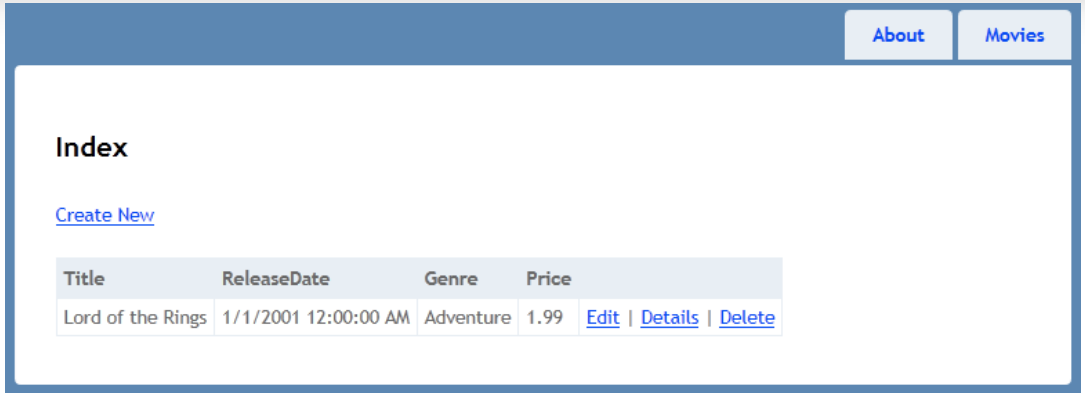
| VM Size | CPU Cores | Memory | Bandwidth (Mbps) | # Data Disks |
|---|---|---|---|---|
| Extra Small | Shared | 768 MB | 5 | 1 |
| Small | 1 | 1.75 GB | 100 | 2 |
| Medium | 2 | 3.5 GB | 200 | 4 |
| Large | 4 | 7 GB | 400 | 8 |
| Extra Large | 8 | 14 GB | 800 | 16 |

Each Persistent Data Disk Can be up to 1 TB

# Windows Azure Cloud Services (PaaS)

- Considerations
  - Instances can reboot
  - IPs can change
  - Input endpoints are load balanced
  - Azure load balancer timeout
- Official Solution
  - Replica set support
  - Should work as is on emulator
  - Only need to configure storage settings
  - Source – https://www.github.com/mongodb/mongo-azure
  - Issues – http://jira.mongodb.org/browse/AZURE

# Demo – Cloud Services (PaaS)




App deployed with the MongoDB Replica Set Roles for Windows Azure

## Replica Set

**Status:** OK with **replica set name:** rs

| Id | Name | Health | State | Last Heart Beat | Last Operation | Ping (to Primary) |
|---|---|---|---|---|---|---|
| 0 | rs_0:27017 | Up | Secondary | 9/14/2012 3:49:59 PM | 9/14/2012 2:57:20 PM | 1 |
| 1 | rs_1:27017 | Up | Primary | *N/A* | 9/14/2012 2:57:20 PM | *N/A* |
| 2 | rs_2:27017 | Up | Secondary | *N/A* | 9/14/2012 2:57:20 PM | 0 |

# Windows Azure Virtual Machines (IaaS)

- Store data in Data Disk
- Multiple instances in a service vs instances across services
- Linux possible
- You need to secure the endpoints
  - Firewall on windows vs linux

# Demo – Virtual Machines (IaaS)

```
azure vm create sn0312-c"OpenLogic__OpenLogic-CentOS-62-20120531-
en-us-30GB.vhd" username password -l "West US" -e
azure vm create sn0312-c "OpenLogic__OpenLogic-CentOS-62-
20120531-en-us-30GB.vhd" username password -l "West US" -e 23 -c
azure vm create sn0312-c "OpenLogic__OpenLogic-CentOS-62-
20120531-en-us-30GB.vhd" username password -l "West US" -e 24 -c

azure vm endpoint create sn0312-c 27017 27017
azure vm endpoint create sn0312-c-2 27018 27018
azure vm endpoint create sn0312-c-3 27019 27019
```

To set up MongoDB, SSH into each instance and:

```
get mongodb binaries and install
create db dir
start mongod, initialize replica set
```

# Resources

- **MongoDB Installer for Windows Azure:** http://datamarket.azure.com/application/6ee0c678-21eb-4aff-b345-b371d0c92ecc
- **MongoDB on Azure:** http://docs.mongodb.org/ecosystem/platforms/windows-azure/
- **MongoDB Experts video series:** http://blogs.msdn.com/b/interoperability/archive/2012/06/01/mongodb-experts-video-series.aspx
- **Install MongoDB on a Centos virtual machine in Windows Azure:** http://www.windowsazure.com/en-us/manage/linux/common-tasks/mongodb-on-a-linux-vm/
- **Node.js Web Application with Storage on MongoDB (Virtual Machine):** http://waweb.windowsazure.com/en-us/develop/nodejs/tutorials/website-with-mongodb-(mac)/

# Next Steps

# Recent Release History

**1.8 March '11**

- Journaling
- Sharding and Replica set enhancements
- Spherical geo search

**2.0 Sept '11**

- Index enhancements to improve size and performance
- Authentication with sharded clusters
- Replica Set Enhancements
- Concurrency improvements

**2.2 Aug '12**

- Aggregation Framework
- Multi-Data Center Deployments
- Improved Performance and Concurrency

**2.4 Winter '13**

10gen | the MongoDB company

mongoDB

# 2.2 Release August 2012

- Concurrency: yielding + db level locking
- New aggregation framework
- TTL Collections
- Improved free list implementation
- Tag aware sharding
- Read Preferences
- http://docs.mongodb.org/manual/release-notes/2.2/

# 2.4 Highlights

- Security - SASL, Kerberos,  additional privileges
- Hash-based Sharding
- Geospatial Indexing: query intersecting polygons
- Aggregation framework: faster and more features
- V8 JavaScript engine replaces SpiderMonkey
- Replica set flapping reduced, initial sync for replication much faster
- MMS running in your own data center
- Text search (experimental)

# Ongoing Work

- Security: finer-level authorization and auditing
- Improved management of sharded clusters
  - Better sharing of network resources in sharded environments (sockets, connections, etc)
- Additional features for Text Search
- Collection-level locking

# Where do I go next?

- Questions
  - mongodb-user google group - https://groups.google.com/forum/?fromgroups#!forum/mongodb-user
  - Stackoverflow - http://stackoverflow.com/questions/tagged/mongodb
  - IRC –freenode.net#mongodb
- Meetups
  - Office hours
    - SF -Epicenter Café alternate Mondays 4 – 6pm
    - Palo Alto – 10Gen office alternate Wednesday 5 – 7pm
  - Meetups
    - SF - http://www.meetup.com/San-Francisco-MongoDB-User-Group/
    - Silicon Valley - http://www.meetup.com/MongoDB-SV-User-Group/
- Existing bugs and issues
  - http://jira.mongodb.org

# MongoDB San Francisco

- May 10, 2013 at the Palace Hotel
- Has sold out 3 years in a row
- Still accepting proposals: Submit at 10gen.com/talk-proposal
- Register at mongoSF.com
  - Use discount code azure10

# Questions?

Sridhar Nanjundeswaran, 10gen
@snanjund

mongoDB

10gen | the MongoDB company