INTRODUCTION TO MONGODB AND RUBY

Gary J. Murakami, Ph.D.

Lead Engineer & Ruby Evangelist

10gen (the MongoDB company)

gary.murakami@10gen.com

Tuesday, September 25, 2012

https://github.com/gjmurakami-10gen/talk-mother-lode

THE CALIFORNIA GOLD RUSH OF 1849

MONGODB: RUBY'S MOTHER LODE

Ruby is a gem of a programming language
dynamic, multi-paradigm
expressive, powerful, enjoyable

MongoDB is the mother lode of DBs
BSON documents, high-performance
indexing, replication, sharding
flexible, powerful, enjoyable

Ruby and **MongoDB** are well matched

OUTFITTING BEYOND YOUR PICKAXE BOOK

MAC OS X- MONGODB QUICKSTART

```
$ brew update
$ brew install mongodb
$ mongod
```

RUBY DRIVER SETUP - RUBY LANGUAGE CENTER

```
$ gem install mongo
$ gem install bson_ext
```

Gemfile

```
gem 'mongo'
gem 'bson_ext'
```

RUBYMINE

IDE includes support for Mongoid and MongoDB

LET'S DIG IN

TWITTER FRIENDS - LOAD USERS

```
require 'mongo'
require 'httpclient'
require 'json'

screen_name = ARGV[0] || 'garymurakami'
friends_ids_uri = "https://api.twitter.com/1/friends/ids.json?cursor=-1&screen_name=#{screen_name}"
friend_ids = JSON.parse(HTTPClient.get(friends_ids_uri).body)['ids']
```

a few lines of Ruby gets and parses **JSON** data from the web - twitter **API**

- libraries/gems
  - mongo, bson, bson_ext
  - httpclient, json
- language capability
  - method call chaining

TWITTER FRIENDS - LOAD USERS (CONTINUED)

```
connection = Mongo::Connection.new
db = connection['twitter']
collection = db['users']

friend_ids.each_slice(100) do |ids| # best practices
    users_lookup_uri = "https://api.twitter.com/1/users/lookup.json?user_id=#{ids.join(',')}"
    response = HTTPClient.get(users_lookup_uri)

    docs = JSON.parse(response.body) # high-level objects - Array of Hashes
    docs.each{|doc| doc['_id'] = doc['id']} # user supplied _id
    collection.insert(docs, :safe => true) # no schema! - bulk insert - best practices
end
puts "users:#{collection.count}"
```

a small number of lines load **JSON** data from the web into a collection

- DB connection, database, collection
- docs - Ruby Array of Hashes from JSON parsing
- doc primary _id set by user or driver
- collection bulk insert and safe mode
- collection count

HITTING PAYDIRT

TWITTER TOP 10 USERS BY FOLLOWERS COUNT

```
users = Mongo::Connection.new['twitter']['users']

users.find({}, :sort => { followers_count: -1 }, :limit => 10).each do |doc|
    puts doc.values_at('followers_count', 'screen_name').join("\t")
end
```

a simple query with sort and limit

- find - cursor enumerable
- expressed in Ruby code
- driver serializes into **BSON** and **Mongo Wire Protocol**
- no text language generation or parsing

```
1409213     Dropbox
25092       MongoDB
9074        oscon
8148        railsconf
7302        spf13
4374        10gen
2259        kchodorow
1925        rit
1295        dmerr
1148        aaronheckmann
```

DIGGING DEEPER

TWITTER - LOAD TWEETS (SIMPLIFIED)

```
    db = Mongo::Connection.new['twitter']
    users = db['users']
    tweets = db['tweets']
    tweets.ensure_index('user.id') # dot notation to specify subfield

    users.find({}, {fields: {id: true, screen_name: true, since_id: true}}).each do |user|
        twitter_user_timeline_uri =  "https://api.twitter.com/1/statuses/user_timeline.json?user_id=#{us
er['id']}&count=200&include_rts=true&contributor_details=true"
        twitter_user_timeline_uri += "since_id=#{user['since_id']}" if user['since_id']
        response = HTTPClient.get(twitter_user_timeline_uri)

        docs = JSON.parse(response.body) # high-level objects
        docs.each{|doc| doc['_id'] = doc['id']} # user supplied _id
        tweets.insert(docs, :continue_on_error => true) # bulk insert
        users.update({_id: user['_id']}, {'$set' => {'since_id' => docs.last['id']}})
        puts tweets.count(:query => {'user.id' => user['id']})
    end
    puts "tweets:#{tweets.count}"
```

a small number of lines loads and records tweets

- indexing - secondary indexes
- retrieving a subset of fields
- update - record since_id
- count with query selector

TWITTER TOP 10 TWEETS FOR MONGODB BY RETWEET_COUNT

```
    tweets = Mongo::Connection.new['twitter']['tweets']

    screen_name = 'MongoDB'

    tweets.find({'user.screen_name' => screen_name}, :sort => { retweet_count: -1 }, :limit => 10).each
 do |doc|
        puts doc.values_at('retweet_count', 'text').join("\t")
    end
```

a simple query with a query selector, sort and limit

- **MongoDB query language**
- dot notation for nested sub-document fields

```
   172 #MongoDB v2.2 released http://t.co/sN6Rzc7D
   77  #mongoDB2dot2 officially released, w/ Advanced Aggregation Framework, Multi-Data Center Deployment
 + 600 new features http://t.co/DMSdWGwN
   59  Announcing free  online MongoDB classes. http://t.co/RIoAb7l7
   30  RT @jmikola: Introducing mongoqp: a frontend for #MongoDB's query profiler https://t.co/ROCSzs6W
   29  Sign up for free online MongoDB courses starting in October http://t.co/Im68Q4NM
   24  1,000+ have signed up for free MongoDB training since the courses were announced today http://t.co
/oINxfH1t
   22  How Disney built a big data platform on a startup budget using MongoDB. http://t.co/8qW5yc7T
   21  #mongoDB2dot2, available for download. http://t.co/kcGhUDhI
   19  MongoDB Sharding Visualizer http://t.co/onIG08jv another product of #10genLabs
   16  MongoDB Java Driver 2.9.0 released http://t.co/UDvZoAYV
```

STRIKING GOLD

BSON - BINARY JSON

http://bsonspec.org/

```
{"BSON": ["awesome", 5.05, 1986]}       →
"\x31\x00\x00\x00
\x04BSON\x00
&\x00\x00\x00
\x020\x00\x08\x00\x00\x00awesome\x00
\x011\x00333333\x14@
\x102\x00\xc2\x07\x00\x00
\x00
\x00"
```

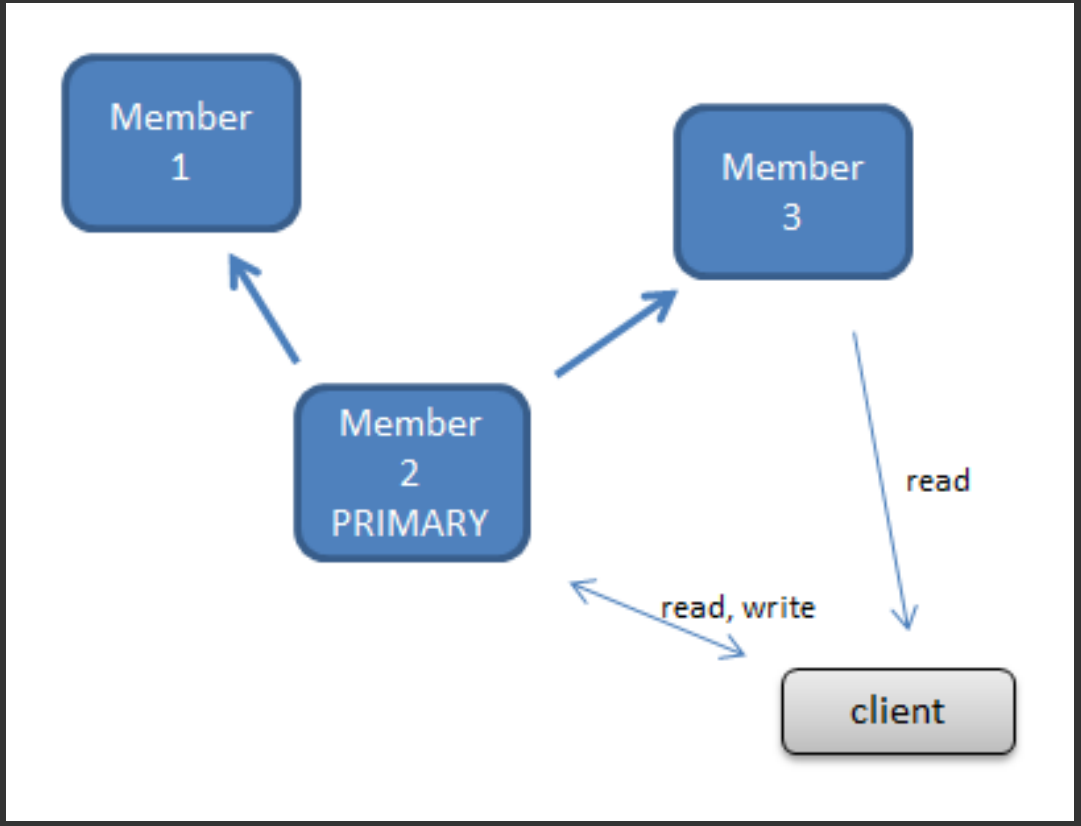
1. base types extended to match modern programming languages
2. transfer over wire - **Mongo Wire Protocol**
   - request payload includes query and documents to server
   - response and documents to client
3. store on server disk
4. index on server

   fast - no parsing, no server-side data repackaging to disk

REPLICA SETS

multiple database server nodes

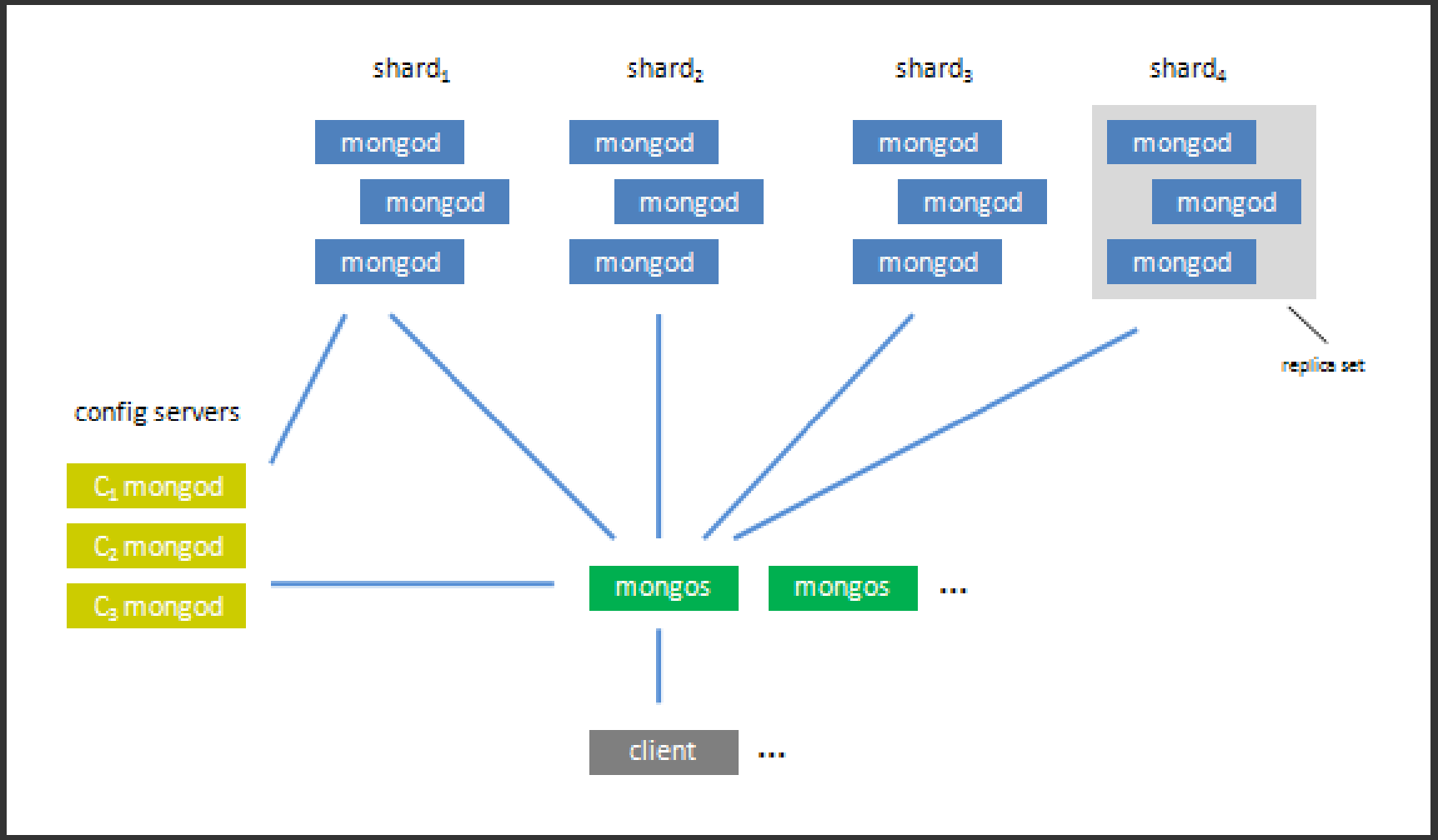
one primary member, many secondary members



- high availability
  - auto-failover via election by majority
  - write concern
- read scaling - read preferences

- backups
- upgrades
- use (hidden) replica for secondary workload
  - analytics
  - data-processing
  - integration with external systems

SHARDING


shard1
shard2
shard3
shard4
mongod
mongod
mongod
mongod
mongod
mongod
mongod
mongod
mongod
mongod
mongod
mongod
replica set
config servers
C1 mongod
C2 mongod
C3 mongod
mongos
mongos
...
client
...

- horizontal scaling across multiple nodes
  - automatic balancing
  - failover - each shard can be a replica set
- shard key - partition by range
- chunks up to 64MB
- routing processes - mongos

NUGGETS - MORE MONGODB FEATURES

- **mongo** shell
  - JavaScript, native for server-side processing
  - highly recommended for development and administration
- **map/reduce**
  - batch processing of data and aggregation operations
- **aggregation framework**
  - calculate aggregated values without having to use map-reduce
  - pipeline, operators, expressions
- **GridFS**
  - specification for storing large files in MongoDB chunks usually 256KB
- **Capped collections**
  - fixed sized collections that have a very high performance auto-FIFO age-out feature
- **Hadoop connector**
  - reading MongoDB data into Hadoop as well as writing map/reduce results back out

FULL-BORE MINING WITH WEB FRAMEWORKS

Ruby on Rails, Sinatra, etc.

MONGOMAPPER ODM - JOHN NUNEMAKER


```
class User
    include MongoMapper::Document
    key :name, String
    key :age,  Integer
    many :hobbies
end

class Hobby
    include MongoMapper::EmbeddedDocument
    key :name,    String
    key :started, Time
end

user = User.new(:name => 'Brandon')
user.hobbies.build(:name => 'Programming', :started => 10.years.ago)
user.save!

User.where(:name => 'Brandon').first
```

- simplified model different from ActiveRecord
- good example for Ruby metaprogramming
- small is beautiful

MONGOID ODM - DURRAN JORDAN

```
class Artist
    include Mongoid::Document
    field :name, type: String
    embeds_many :instruments
end

class Instrument
    include Mongoid::Document
    field :name, type: String
    embedded_in :artist
end

syd = Artist.where(name: "Syd Vicious").between(age: 18..25).first
syd.instruments.create(name: "Bass")
syd.with(database: "bands", session: "backup").save!
```

- documents
- persistence
- querying
- ActiveModel relations (belongs_to, has_one, has_many, habtm)
- nested attributes
- validation
- indexing
- extras

MOPED DRIVER - BERNERD SCHAEFER

```
session = Moped::Session.new([ "127.0.0.1:27017" ])
session.use "echo_test"

session.with(safe: true) do |safe|
safe[:artists].insert(name: "Syd Vicious")
end

session[:artists].find(name: "Syd Vicious").update(:$push => { instruments: { name: "Bass" }})
```

a MongoDB driver for Ruby which exposes a simple, elegant, and fast API

- fast, thread-safe, uses **sidekiq**
- pure Ruby - no C extensions
- metaprogramming - extends standard classes
- Ruby > 1.9
- used by Mongoid 3

JRUBY

- The Best of the JVM
  - High performance
  - Real threading - multi-core parallelism
  - Vast array of libraries
- It's Just Ruby
  - Ruby 1.8.7 & 1.9.2 compatible
  - Both versions in single install
- bson_java
- Mongo Hadoop connector

FOOL'S GOLD - RDBMS CASE STUDY

ADOBE PHOTOSHOP LIGHTROOM

"Lightroom 3 Catalog.lrcat"

- 0.5 GB
- 66 Sqlite3 tables
- 20 tables with column image INTEGER
- can be loaded with 40 lines of Ruby
- can be reduced to 1 image collection

THE COST OF USING RDBMS FOR PERSISTENCE

- RDBMS flattened and normalized our data into rigid tables
- RDBMS costs O(10**6) - O(1000000) and more with many joins
  - Memory O(10**9/sec) - O(1000000000)
  - RDBMS O(10**3/sec) - O(1000)

THE MOTHER LODE

MONGODB SUMMARY

- replica sets for high availability
- sharded clusters for scaling

- very high performance
  - secondary indices
  - BSON is fast for transfer, storage, queries
  - server-side processing
    - map-reduce, aggregation
    - Hadoop connector
- very low cost - open source - community

MONGODB CONCLUSION

- enjoyable software development
  - 10gen and Moped Ruby drivers
  - MongoMapper and Mongoid ODMs
  - no schema migrations forced

- **well-matched to programming languages**
  - bigger data
  - Ruby and objects

ANNOUNCEMENT AND QUESTIONS

MONGODB AND RUBY BLOGGING CONTEST

Objective

Write about best practices for using MongoDB with Ruby.

Post your blog posts in the comments section of
this blog post by October 10 to be entered to win.

http://blog.10gen.com/post/32158163299/mongodb-and-ruby-blogging-contest

Google: mongodb and ruby blogging contest

THANK YOU

AND MAY YOU STRIKE GOLD

```
{
    name: "Gary J. Murakami, Ph.D.",
    title: "Lead Engineer and Ruby Evangelist",
    company: "10gen (the MongoDB company)",
    phone: "1-866-237-8815 x8015",
    mobile: "1-908-787-6621",
    email: "gary.murakami@10gen.com",
    im: "gjmurakami (AIM)",
    twitter: "@GaryMurakami",
    blog: "grayghostwriter.blogspot.com",
    website: "www.nobell.org",
    linkedin: "www.linkedin.com/pub/gary-murakami/1/36a/327",
    facebook: "facebook.com/gary.j.murakami"
}
```

- **https://github.com/gjmurakami-10gen/talk-mother-lode**
- git://github.com/gjmurakami-10gen/talk-mother-lode.git

- **MongoDB**
- **Ruby Language Center**
- **Github mongodb/mongo-ruby-driver**