MongoDB & Mongoid

By Reid Morrison, Software Architect
@reidmorrison

MongoDB

- Document Oriented Data Store
  - Schemaless
    - No more database migrations
    - Stores JSON like documents

MongoDB is Web Scale

- Very High performance reads
  - Memory Caching (removes the need for memcache)
- Sharding
  - Horizontal Scaling
  - Automatic Distribution & Balancing
- Atomic Updates
  - Reduced Locking
  - No More DeadLocks
- Replication
- High Availability

MongoDB Terms

- Database
- Collection (Table)
- Document (Row)
- Field (Column)
- Referenced Documents
- Embedded Documents

Document

```
{
  "_id" : ObjectId("4d3ed089fb60ab534684b7e9"),
  "first_name" : "Joe",
  "last_name": "Bloggs",
  "addresses" : [
    {
      "_id" : ObjectId("4d3ed089fb60ab534684b7e0"),
      "city" : "Tampa",
      "state" : "FL"
    }
  ]
}
```

Mongoid

- ActiveRecord like API into MongoDB

```
person = Person.new(:name => “Joe Bloggs”)
person.save!
Person.create(:name => “Joe Bloggs”)
Person.where(:name => “Joe Bloggs”).first
```

- Specifies & Enforces Schema
  - Automatic data type conversion
- Rails Validations, Generators, etc.

Mongoid Document

```
class Person
  include Mongoid::Document

  embeds_many :addresses
  field :name,      :type => String
  field :birthday,  :type => Date
end
```

```
class Address
  include Mongoid::Document

  embedded_in :person
  field :city,    :type => String
  field :state, :type => String
end
```

Installation

- For Bundler, add to Gemfile:

```
gem 'mongoid', '~> 2.4'
gem 'bson_ext', '~>1.5'    # Ruby MRI only
```

  - bundle install
- Create config file

```
rails generate mongoid:config
```

Without Mongoid

- Spelling errors
  - "first_name" : "Joe" != "firstname" : "Joe"
- Incorrect types
  - "age" : "10" != "age" : 10
- Invalid _id type
  - ObjectId("4d3ed089fb60ab534684b7e9") != "4d3ed089fb60ab534684b7e9"
- Have to manually convert all dates and times to/from UTC Time

Schema Design

- Embedded Documents
  - Single read
  - Must be related
  - Needed to do "joins"
- Referenced Documents
  - Too big to embed everything
  - Consider the network overhead of reading large objects
  - "Joins" done in client code or via map-reduce
- One size does not fit all
  - Consider separate databases
  - Frequent access smaller data that can fit in memory
  - Large data sets across a shared cluster
  - Separate database just for Logging, temporary collections
  - We run 4 databases in prod

Sharding

- Scales horizontally
  - More disk and CPU
- Sharding Key choice is critical
  - Majority of reads must use sharding key
  - Without key, reads each shard sequentially
- Add new shards early
- Only shard when you absolutely have to
  - Steep learning curve
  - Pilot system / collection

MongoDB Benefits

- Replaces memcache, redis and possibly RDBMS
- Co-exists with RDBMS
- Capped collections
  - Highly performant
  - Excellent for logs etc
- Schema changes at any time
  - New documents have new fields
- Web Scale

Mongoid Benefits

- Schema is defined and enforced in the model
  - not in the database
- Active community
- Supports Rails 2 and 3
- Other features
  - Versioning
  - Caching
  - Composite Keys, Sharding keys
  - Timestamps: created_at, updated_at
- Coexists with MySQL
  - Can disable Mongoid generating models

MongoDB Issues

- Recoverability
  - What if the connection fails during an insert?
    - Use a unique key to check for duplicate key exception on retry
    - Otherwise, check if it is already saved, otherwise retry
    - Always use safe operations since a network failure will result in lost data
  - Add your own retry code
- Durability
  - Use Replica Sets
- Consistency
  - Use Mongoid to enforce Schema
- Sharding
  - Choosing the right sharding key is hard
    - Make the sharding key unique if possible
- Stay away from the bleeding-edge

MongoDB Ruby driver

- Reliability and HA issues
  - Designed for use against a local server
  - Using a client across a WAN or Internet is troublesome
    - No automatic retry on failure
      - Multithreaded Apps - JRuby
    - https://gist.github.com/1498297
- Returns Mongo Connection failures as Operation Failures when going through a Mongo Router
- Developers are working on these issues

Diagnostics

- High Lock Percentages
  - Page Faults
  - Missing indexes
  - Connection Saturation
  - Shard re-balancing in progress
    - Change shard balancing window to off-peak periods
- Read or write operation queues building up
  - Operations have required indexes?
  - Move reads to slaves
    - Useful for apps that are read intensive
- CPU or Disk Utilization high?
  - Newer, faster servers
  - Consider Sharding
- Current Operation
  - db.currentOp()
- Disk Utilization
  - iostat -mxt 5
- Journaling or Flushing
  - RAID10
  - SSD
  - SAN / DAS
- Active Connections
- Network I/O
  - Network bandwidth sufficient
  - Can Queries limit the fields returned returned
  - Is schema too large, consider breaking into multiple collections
- Fragmentation (use: db.collection.stats())
  - storageSize / size < 1.5 ?
  - Not able to reclaim free space as quickly as needed
- Padding level > 1.5 is too high
  - Invalid data model
  - Embedded documents growing
  - Review schema design
- Compact Slaves
  - Compact Replica-set slaves to reduce fragmentation
  - Only available in Mongo 2.0 or above

Mongoid Issues

- Some Limited built-in retry capabilities
  - Configure retries through config file
- Mongoid Documentation is light
  - Assumes knowledge of MongoDB

Mongoid Recommendations

- Namespace Mongoid Models in large applications

```
embeds_many :addresses, :class_name => 'Clarity::Address'
```

- Enable Retries in config file
- Don't update MongoDB directly
- Turn on Safe writes by default

```
persist_in_safe_mode: true
```

MongoDB Recommendations

- Read from local secondaries, write to primary
  - No ACID, so use unique primary key on all important collections
  - Insert retry must fail if first insert succeeded
- Enable Slow Query log

MongoDB Recommendations

- RAID10 Disk, with SSD even better
- Enough physical memory to hold working set
- ext4 or XFS for data storage
- Turn off atime on data disk
- Monitoring tools
  - Mongo Monitoring https://mms.10gen.com/user/login
  - Hyperic HQ
  - mongostat
- Chef Deployment

Getting Started

- Pilot MongoDB
  - Start Small
  - Replace a MemCache or Redis collection
- Experiment
  - Sharded Cluster
  - Sharding Key
    - Cannot be changed

Questions

MongoDB: http://www.mongodb.org/

Mongoid: http://mongoid.org/

Reid Morrison

@reidmorrison

reidmo@gmail.com

www.linkedin.com/in/reidmorrison

Other Gems

- active_record_slave
  - Replacement for read from slave
  - Supports dynamic SQL calls, AREL, etc
  - Highly performant with no overhead for calls to master/primary
- sync_attr
  - Thread-safe Synchronized attributes and class variables for lazy loading and/or default values
  - Don't have to stick everything into a Rails initializer
- Jms4jruby
  - JMS API for JRuby to talk to ActiveMQ, HornetQ, WebSphere MQ, Oracle AQ, any JMS provider.
- hyperic-mongodb
  - Monitoring a MongoDB sharded cluster using Hyperic HQ
- RubyWMQ
  - Ruby MRI gem for communicating with IBM WebSphere MQ