MongoDB’s Java Driver

Bryan Reinero
bryan.reinero@10gen.com
Topics we’ll cover

• Main responsibilities of the driver
• New features for version 2.9.0
Driver’s Main Responsibilities

- Manage connections
- BSON serialization / deserialization
- Manage cursors
- Maintain replica set awareness
/** database connection with internal pooling */
public class Mongo

/** The mechanism managing the pool */
public class DBPortPool extends SimplePool<DBPort>

/** Various settings for the driver */
public class MongoOptions
Wire Protocol

- TCP/IP socket based request/response
- Exchanges BSON objects between client & server
- BSON – a traversable, binary encoded serialization of JSON documents
- BSON is the primary data representation in MongoDB

http://bsonspec.org/
Encoding / Decoding BSON

interface DBDecoder

DBObject decode(InputStream in, DBCollection collection);

interface DBEncoder

int writeObject(OutputBuffer buf, BSONObject o);
On to the good stuff…

Reading, Writing & Arithmetic
A simple example...
Basics of Replication

- **Node 1**: Secondary
- **Node 2**: Secondary
- **Node 3**: Primary

Arrow connections indicate:
- Replication
- Heartbeat
Strong Consistency
Eventual Consistency

- **Primary**
- **Secondary**

Client Application → Driver → Write → Primary → Secondary → Read → Secondary → Read
Read Preferences

- PRIMARY
- PRIMARY PREFERRED
- SECONDARY
- SECONDARY PREFERRED
- NEAREST
Using Read Preferences...
Writes & Durability

- Fire and forget
- Wait for error
- Wait for journal commit
- Wait for fsync
- Wait for replication to secondary
Using write concerns...
Tagging

- Each node within a replica set can be marked with descriptors called tags
- Tags can indicate a node’s location, membership in a set, or other characteristics
- Use tags to target specific nodes for read or write operations
Using Tags

tags" : {
  "datacenter" : "Los Angeles",
  "region" : "US_West"
}

Node 1

tags" : {
  "datacenter" : ”San Jose”,
  "region" : ”US_West”
}

Node 2

tags" : {
  "datacenter" : “Richmond”,
  "region" : ”US_East”
}

Node 3
Using tags...
Aggregation Framework

- Calculate aggregate values without having to use map-reduce
- Written in C++, high performance
- Declarative, no javascript
- Extensible, new operations easily added
Aggregation Operators

- $match
- $project
- $unwind
- $group
- $sort
- $skip
- $limit
Aggregation Example

sample document:

```json
{
  "_id": ObjectId("5029e745a0988a275aefd0c0"),
  "name": "quiz",
  "score": 99,
  "student": 7,
  "year": "junior"
}
```
Pipelining Operations

$match

$match : { year : 'junior' }

$project

$project : { name : 1, score : 1 }

$group

$group: {
  _id: '$name',
  average: { $avg: '$score' }
}

10gen

mongoDB
What the command looks like…

```
{
    "aggregate" : "scores",
    "pipeline" : [
        {
            "$match" : {
                "year" : "junior"
            }
        },
        {
            "$project" : {
                "name" : 1,
                "score" : 1,
                "_id" : 0
            }
        },
        {
            "$group" : {
                "_id" : "$name",
                "average" : {
                    "$avg" : "$score"
                }
            }
        }
    ]
}
```
The aggregation helper...
There’s plenty more!...

- Tutorials http://www.mongodb.org/display/DOCS/Java+Tutorial
- Javadocs http://api.mongodb.org/java/current/
- Jira https://jira.mongodb.org/secure/Dashboard.jspa
Join us!

We’re Hiring!

Bryan.reinero@10gen.com