

# The Elephant In the Room: MongoDB + Hadoop

Brendan W. McAdams

10gen, Inc.

Mongo Chicago Conference - October 2010

# Outline

- 1 Introduction
  - Why Integrate?
  - MongoDB + MapReduce in Java
  - Using Pig for High Level Data Analysis

# Hadoop Explained...

- Started in February 2006 as part of the Apache Lucene project
- Based upon Google's MapReduce and GFS Papers
- Allows distributed, scalable data processing of huge datasets
- Java based, but with support for other JVM and Non-JVM Languages
- Lots of ecosystem tools to simplify interaction such as Pig and Hive
- In use at New York Times, Last.fm, Yahoo!, Amazon, Facebook and *many more companies...*
- Great tools for temporary Hadoop clusters such as the Cloudera Cluster Tools, Apache Whirr and Amazon's Elastic MapReduce.

# Why Integrate MongoDB?

- Easiest Answer: People keep asking for it . . .
- Limitations in MongoDB's built in MapReduce
  - ▶ Language: JavaScript only; not everyone wants to write JavaScript for data processing.
  - ▶ Concurrency: Current JS Implementation is limited to one JS execution per server at a time.
  - ▶ Scalability: Not a lot of ability to scale MongoDB's MapReduce except in cases of sharding.

# Why Integrate MongoDB?

- Easiest Answer: People keep asking for it ...
- Limitations in MongoDB's built in MapReduce
  - ▶ Language: JavaScript only; not everyone wants to write JavaScript for data processing.
  - ▶ Concurrency: Current JS Implementation is limited to one JS execution per server at a time.
  - ▶ Scalability: Not a lot of ability to scale MongoDB's MapReduce except in cases of sharding.

# Why Integrate MongoDB?

- Easiest Answer: People keep asking for it . . .
- Limitations in MongoDB's built in MapReduce
  - ▶ Language: JavaScript only; not everyone wants to write JavaScript for data processing.
  - ▶ Concurrency: Current JS Implementation is limited to one JS execution per server at a time.
  - ▶ Scalability: Not a lot of ability to scale MongoDB's MapReduce except in cases of sharding.

# Why Integrate MongoDB?

- Easiest Answer: People keep asking for it . . .
- Limitations in MongoDB's built in MapReduce
  - ▶ Language: JavaScript only; not everyone wants to write JavaScript for data processing.
  - ▶ Concurrency: Current JS Implementation is limited to one JS execution per server at a time.
  - ▶ Scalability: Not a lot of ability to scale MongoDB's MapReduce except in cases of sharding.

# Mongo + Hadoop ...

- Integrating MongoDB and Hadoop to read & write data from/to MongoDB via Hadoop
- 10gen has been working on a plugin to integrate the two systems, written in Pure Java
- About 6 months ago I explored the idea in Scala with a project called 'Luau'
- Support for pure MapReduce as well as Pig (Currently output only - input coming soon)
- With Hadoop Streaming (soon), write your MapReduce in Python or Ruby



# MongoDB + MapReduce in Java I

```
// WordCount.java

import java.io.*;
import java.util.*;

import org.bson.*;
import com.mongodb.*;
import com.mongodb.hadoop.*;

import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.*;

/**
 * test.in
 * db.in.insert( { x : "eliot was here" } )
 * db.in.insert( { x : "eliot is here" } )
 * db.in.insert( { x : "who is here" } )
 * =
 */
public class WordCount {

    public static class TokenizerMapper extends Mapper<Object, BSONObject, Text,
        IntWritable>{

        private final static IntWritable one = new IntWritable(1);
        private Text word = new Text();

        public void map(Object key, BSONObject value, Context context )
            throws IOException, InterruptedException {
```

# MongoDB + MapReduce in Java II

```
        System.out.println( "key: " + key );
        System.out.println( "value: " + value );

        StringTokenizer itr = new StringTokenizer(value.get( "x" ).toString());
        while (itr.hasMoreTokens()) {
            word.set(itr.nextToken());
            context.write(word, one);
        }
    }
}

public static class IntSumReducer extends Reducer<Text,IntWritable,Text,IntWritable> {

    private IntWritable result = new IntWritable();

    public void reduce(Text key, Iterable<IntWritable> values, Context context )
        throws IOException, InterruptedException {

        int sum = 0;
        for (IntWritable val : values) {
            sum += val.get();
        }
        result.set(sum);
        context.write(key, result);
    }
}

public static void main(String[] args)
    throws Exception {
```

# MongoDB + MapReduce in Java III

```
Configuration conf = new Configuration();
conf.set( "MONGO_INPUT" , "mongodb://localhost/test.in" );
conf.set( "MONGO_OUTPUT" , "mongodb://localhost/test.out" );

Job job = new Job(conf, "word count");

job.setJarByClass(WordCount.class);

job.setMapperClass(TokenizerMapper.class);

job.setCombinerClass(IntSumReducer.class);
job.setReducerClass(IntSumReducer.class);

job.setOutputKeyClass(Text.class);
job.setOutputValueClass(IntWritable.class);

job.setInputFormatClass( MongoInputFormat.class );
job.setOutputFormatClass( MongoOutputFormat.class );

System.exit(job.waitForCompletion(true) ? 0 : 1);
}
```

# The Results I

```
> db.out.find()
{ "_id" : "talking", "value" : 1 }
{ "_id" : "tender,", "value" : 1 }
{ "_id" : "tender.", "value" : 1 }
{ "_id" : "them--\\", "value" : 1 }
{ "_id" : "there's", "value" : 3 }
{ "_id" : "there,\\", "value" : 1 }
{ "_id" : "there.\\", "value" : 1 }
{ "_id" : "thing,\\", "value" : 1 }
{ "_id" : "thing?\\", "value" : 1 }
{ "_id" : "things", "value" : 3 }
{ "_id" : "things.\\", "value" : 3 }
{ "_id" : "think", "value" : 5 }
{ "_id" : "thirsty,\\", "value" : 1 }
{ "_id" : "this", "value" : 7 }
{ "_id" : "this.\\", "value" : 1 }
{ "_id" : "those", "value" : 1 }
{ "_id" : "thought", "value" : 1 }
{ "_id" : "threaded", "value" : 1 }
{ "_id" : "throw", "value" : 1 }
{ "_id" : "tidy.", "value" : 2 }
has more
```

# Sample Input Data I

2A9EABFB35F5B954	970916105432	+md foods +proteins
BED75271605EBD0C	970916001949	yahoo chat
BED75271605EBD0C	970916001954	yahoo chat
BED75271605EBD0C	970916003523	yahoo chat
BED75271605EBD0C	970916011322	yahoo search
BED75271605EBD0C	970916011404	yahoo chat
BED75271605EBD0C	970916011422	yahoo chat
BED75271605EBD0C	970916012756	yahoo caht
BED75271605EBD0C	970916012816	yahoo chat
BED75271605EBD0C	970916023603	yahoo chat
BED75271605EBD0C	970916025458	yahoo caht
BED75271605EBD0C	970916025516	yahoo chat
BED75271605EBD0C	970916030348	yahoo chat
BED75271605EBD0C	970916034807	yahoo chat
BED75271605EBD0C	970916040755	yahoo chat
BED75271605EBD0C	970916090700	hawaii chat universe
BED75271605EBD0C	970916094445	yahoo chat
BED75271605EBD0C	970916191427	yahoo chat
BED75271605EBD0C	970916201045	yahoo chat
BED75271605EBD0C	970916201050	yahoo chat
BED75271605EBD0C	970916201927	yahoo chat
824F413FA37520BF	970916184809	garter belts
824F413FA37520BF	970916184818	garter belts
824F413FA37520BF	970916184939	lingerie
824F413FA37520BF	970916185051	spiderman
824F413FA37520BF	970916185155	tommy hilfiger
824F413FA37520BF	970916185257	calgary
824F413FA37520BF	970916185513	calgary
824F413FA37520BF	970916185605	exhibitionists
824F413FA37520BF	970916190220	exhibitionists
824F413FA37520BF	970916191233	exhibitionists

# Sample Input Data II

7A8D9CFC957C7FCA	970916064707	duron paint
7A8D9CFC957C7FCA	970916064731	duron paint
A25C8C765238184A	970916103534	brookings
A25C8C765238184A	970916104751	breton liberation front
A25C8C765238184A	970916105238	breton
A25C8C765238184A	970916105322	breton liberation front
A25C8C765238184A	970916105539	breton
A25C8C765238184A	970916105628	breton
A25C8C765238184A	970916105723	front de liberation de la bretagne
A25C8C765238184A	970916105857	front de liberation de la bretagne
6589F4342B215FD4	970916125147	afghanistan
6589F4342B215FD4	970916125158	afghanistan
6589F4342B215FD4	970916125407	afghanistan
16DE160B4FFE3B85	970916050356	eia rs232-c
16DE160B4FFE3B85	970916050645	nullmodem
16DE160B4FFE3B85	970916050807	nullmodem
563FC9A7E8A9022A	970916042826	organizational chart
563FC9A7E8A9022A	970916221456	organizational chart of uae's companies
563FC9A7E8A9022A	970916221611	organizational chart of dubai duty free

# The Pig Script I

```
-- Query Phrase Popularity (local mode)

-- This script processes a search query log file from the Excite search engine and finds
  search phrases that occur with particular high frequency during certain times of the
  day.

-- Register the tutorial JAR file so that the included UDFs can be called in the script.

-- Based on the Pig tutorial ,modified for Mongo support tests
REGISTER src/examples/pigtutorial.jar;
REGISTER mongo-hadoop.jar;
REGISTER lib/mongo-java-driver.jar;

-- Use the PigStorage function to load the excite log file into the raw bag as an array of
  records.
-- Input: (user,time,query)
raw = LOAD 'excite-small.log' USING PigStorage('\t') AS (user, time, query);

-- Call the NonURLDetector UDF to remove records if the query field is empty or a URL.
clean1 = FILTER raw BY org.apache.pig.tutorial.NonURLDetector(query);

-- Call the ToLower UDF to change the query field to lowercase.
clean2 = FOREACH clean1 GENERATE user, time, org.apache.pig.tutorial.ToLower(query) as query;

-- Because the log file only contains queries for a single day, we are only interested in
  the hour.
-- The excite query log timestamp format is YYMMDDHHMMSS.
-- Call the ExtractHour UDF to extract the hour (HH) from the time field.
houred = FOREACH clean2 GENERATE user, org.apache.pig.tutorial.ExtractHour(time) as hour,
  query;
```

# The Pig Script II

```
-- Call the NGramGenerator UDF to compose the n-grams of the query.
ngramed1 = FOREACH houred GENERATE user, hour,
    flatten(org.apache.pig.tutorial.NGramGenerator(query)) as ngram;

-- Use the DISTINCT command to get the unique n-grams for all records.
ngramed2 = DISTINCT ngramed1;

-- Use the GROUP command to group records by n-gram and hour.
hour_frequency1 = GROUP ngramed2 BY (ngram, hour);

-- Use the COUNT function to get the count (occurrences) of each n-gram.
hour_frequency2 = FOREACH hour_frequency1 GENERATE flatten($0), COUNT($1) as count;

-- Use the GROUP command to group records by n-gram only.
-- Each group now corresponds to a distinct n-gram and has the count for each hour.
uniq_frequency1 = GROUP hour_frequency2 BY group::ngram;

-- For each group, identify the hour in which this n-gram is used with a particularly high
    frequency.
-- Call the ScoreGenerator UDF to calculate a "popularity" score for the n-gram.
uniq_frequency2 = FOREACH uniq_frequency1 GENERATE flatten($0),
    flatten(org.apache.pig.tutorial.ScoreGenerator($1));

-- Use the FOREACH-GENERATE command to assign names to the fields.
uniq_frequency3 = FOREACH uniq_frequency2 GENERATE $1 as hour, $0 as ngram, $2 as score, $3
    as count, $4 as mean;

-- Use the FILTER command to move all records with a score less than or equal to 2.0.
filtered_uniq_frequency = FILTER uniq_frequency3 BY score > 2.0;

-- Use the ORDER command to sort the remaining records by hour and score.
```



# The Pig Script III

```
ordered_uniq_frequency = ORDER filtered_uniq_frequency BY hour, score;

-- Use the PigStorage function to store the results.
-- Output: (hour, n-gram, score, count, average_counts_among_all_hours)
STORE ordered_uniq_frequency INTO 'mongodb://localhost/test.pig.output' USING
    com.mongodb.hadoop.pig.MongoStorage;
```

# The Output I

```
> db.pig.output.find()
{ "_id" : ObjectId("4cbbbd9487e836b7e0dc1052"), "hour" : "07", "ngram" : "new", "score" :
  2.4494897427831788, "count" : NumberLong(2), "mean" : 1.1428571428571426 }
{ "_id" : ObjectId("4cbbbd9487e836b7e1dc1052"), "hour" : "08", "ngram" : "pictures", "score"
  : 2.04939015319192, "count" : NumberLong(3), "mean" : 1.4999999999999998 }
{ "_id" : ObjectId("4cbbbd9487e836b7e2dc1052"), "hour" : "08", "ngram" : "computer", "score"
  : 2.4494897427831788, "count" : NumberLong(2), "mean" : 1.1428571428571426 }
{ "_id" : ObjectId("4cbbbd9487e836b7e3dc1052"), "hour" : "08", "ngram" : "s", "score" :
  2.545584412271571, "count" : NumberLong(3), "mean" : 1.3636363636363635 }
{ "_id" : ObjectId("4cbbbd9487e836b7e4dc1052"), "hour" : "10", "ngram" : "free", "score" :
  2.2657896674010605, "count" : NumberLong(4), "mean" : 1.923076923076923 }
{ "_id" : ObjectId("4cbbbd9487e836b7e5dc1052"), "hour" : "10", "ngram" : "to", "score" :
  2.6457513110645903, "count" : NumberLong(2), "mean" : 1.125 }
{ "_id" : ObjectId("4cbbbd9487e836b7e6dc1052"), "hour" : "10", "ngram" : "pics", "score" :
  2.794002794004192, "count" : NumberLong(3), "mean" : 1.3076923076923075 }
{ "_id" : ObjectId("4cbbbd9487e836b7e7dc1052"), "hour" : "10", "ngram" : "school", "score" :
  2.828427124746189, "count" : NumberLong(2), "mean" : 1.1111111111111114 }
{ "_id" : ObjectId("4cbbbd9487e836b7e8dc1052"), "hour" : "11", "ngram" : "pictures", "score"
  : 2.04939015319192, "count" : NumberLong(3), "mean" : 1.4999999999999998 }
{ "_id" : ObjectId("4cbbbd9487e836b7e9dc1052"), "hour" : "11", "ngram" : "in", "score" :
  2.1572774865200244, "count" : NumberLong(3), "mean" : 1.4285714285714284 }
{ "_id" : ObjectId("4cbbbd9487e836b7eadc1052"), "hour" : "13", "ngram" : "the", "score" :
  3.1309398305840723, "count" : NumberLong(6), "mean" : 1.9375 }
{ "_id" : ObjectId("4cbbbd9487e836b7ebdc1052"), "hour" : "14", "ngram" : "music", "score" :
  2.1105794120443453, "count" : NumberLong(4), "mean" : 1.6666666666666667 }
{ "_id" : ObjectId("4cbbbd9487e836b7ecdc1052"), "hour" : "14", "ngram" : "city", "score" :
  2.2360679774997902, "count" : NumberLong(2), "mean" : 1.1666666666666665 }
{ "_id" : ObjectId("4cbbbd9487e836b7eddc1052"), "hour" : "14", "ngram" : "university",
  "score" : 2.412090756622109, "count" : NumberLong(3), "mean" : 1.4000000000000001 }
{ "_id" : ObjectId("4cbbbd9487e836b7eedc1052"), "hour" : "15", "ngram" : "adult", "score" :
  2.8284271247461903, "count" : NumberLong(2), "mean" : 1.1111111111111112 }
```

# The Output II

```
{ "_id" : ObjectId("4cbbbd9487e836b7efdc1052"), "hour" : "17", "ngram" : "chat", "score" :  
  2.9104275004359965, "count" : NumberLong(3), "mean" : 1.2857142857142854 }  
{ "_id" : ObjectId("4cbbbd9487e836b7f0dc1052"), "hour" : "19", "ngram" : "in", "score" :  
  2.1572774865200244, "count" : NumberLong(3), "mean" : 1.4285714285714284 }  
{ "_id" : ObjectId("4cbbbd9487e836b7f1dc1052"), "hour" : "19", "ngram" : "car", "score" :  
  2.23606797749979, "count" : NumberLong(3), "mean" : 1.3333333333333333 }  
{ "_id" : ObjectId("4cbbcf8194b936b7073ae8cb"), "hour" : "07", "ngram" : "new", "score" :  
  2.4494897427831788, "count" : NumberLong(2), "mean" : 1.1428571428571426 }  
{ "_id" : ObjectId("4cbbcf8194b936b7083ae8cb"), "hour" : "08", "ngram" : "pictures", "score"  
  : 2.04939015319192, "count" : NumberLong(3), "mean" : 1.4999999999999998 }  
has more
```

# Questions?

## Contact Info

- Contact Me

- ▶ twitter: @rit
- ▶ email: **brendan@10gen.com**
- ▶ mongo-hadoop ... Available Soon in Alpha Form:  
`http://github.com/mongodb/mongo-hadoop`

- Pressing Questions?

- ▶ IRC - freenode.net **#mongodb**
- ▶ MongoDB Users List -  
`http://groups.google.com/group/mongodb-user`