MongoDB Plugin & Toolchain for Hadoop

Brendan W. McAdams

10gen, Inc.

Dec. 3, 2010 @ MongoSV

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?

- Language: JavaScript only; not everyone wants to write JavaScript for data processing.
  - Static Typing: JavaScript is dynamically typed which has limited value for jobs where a stronger type system is desired.
  - JVM Ecosystem: The JVM has a large number of libraries available for assisting in calculations and analysis which are available from Hadoop.
- 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.
- Integration: More development shops are integrating their data analysis into Hadoop jobs—opportunity to merge data from multiple systems including MongoDB for analysis.
- Hadoop is not a panacea—queries won't necessarily be fast(er) but it can handle larger scale and free up your MongoDB servers for long jobs.

Why Integrate MongoDB?

- Language: JavaScript only; not everyone wants to write JavaScript for data processing.
  - Static Typing: JavaScript is dynamically typed which has limited value for jobs where a stronger type system is desired.
  - JVM Ecosystem: The JVM has a large number of libraries available for assisting in calculations and analysis which are available from Hadoop.
- 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.
- Integration: More development shops are integrating their data analysis into Hadoop jobs—opportunity to merge data from multiple systems including MongoDB for analysis.
- Hadoop is not a panacea—queries won't necessarily be fast(er) but it can handle larger scale and free up your MongoDB servers for long jobs.

Why Integrate MongoDB?

- Language: JavaScript only; not everyone wants to write JavaScript for data processing.
  - Static Typing: JavaScript is dynamically typed which has limited value for jobs where a stronger type system is desired.
  - JVM Ecosystem: The JVM has a large number of libraries available for assisting in calculations and analysis which are available from Hadoop.
- 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.
- Integration: More development shops are integrating their data analysis into Hadoop jobs—opportunity to merge data from multiple systems including MongoDB for analysis.
- Hadoop is not a panacea—queries won't necessarily be fast(er) but it can handle larger scale and free up your MongoDB servers for long jobs.

Why Integrate MongoDB?

- Language: JavaScript only; not everyone wants to write JavaScript for data processing.
    - Static Typing: JavaScript is dynamically typed which has limited value for jobs where a stronger type system is desired.
    - JVM Ecosystem: The JVM has a large number of libraries available for assisting in calculations and analysis which are available from Hadoop.
- 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.
- Integration: More development shops are integrating their data analysis into Hadoop jobs—opportunity to merge data from multiple systems including MongoDB for analysis.
- Hadoop is not a panacea—queries won't necessarily be fast(er) but it can handle larger scale and free up your MongoDB servers for long jobs.

Why Integrate MongoDB?

- Language: JavaScript only; not everyone wants to write JavaScript for data processing.
  - Static Typing: JavaScript is dynamically typed which has limited value for jobs where a stronger type system is desired.
  - JVM Ecosystem: The JVM has a large number of libraries available for assisting in calculations and analysis which are available from Hadoop.
- 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.
- Integration: More development shops are integrating their data analysis into Hadoop jobs—opportunity to merge data from multiple systems including MongoDB for analysis.
- Hadoop is not a panacea—queries won't necessarily be fast(er) but it can handle larger scale and free up your MongoDB servers for long jobs.

Why Integrate MongoDB?

- Language: JavaScript only; not everyone wants to write JavaScript for data processing.
  - Static Typing: JavaScript is dynamically typed which has limited value for jobs where a stronger type system is desired.
  - JVM Ecosystem: The JVM has a large number of libraries available for assisting in calculations and analysis which are available from Hadoop.
- 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.
- Integration: More development shops are integrating their data analysis into Hadoop jobs—opportunity to merge data from multiple systems including MongoDB for analysis.
- Hadoop is not a panacea—queries won't necessarily be fast(er) but it can handle larger scale and free up your MongoDB servers for long jobs.

Why Integrate MongoDB?

- Language: JavaScript only; not everyone wants to write JavaScript for data processing.
  - Static Typing: JavaScript is dynamically typed which has limited value for jobs where a stronger type system is desired.
  - JVM Ecosystem: The JVM has a large number of libraries available for assisting in calculations and analysis which are available from Hadoop.
- 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.
- Integration: More development shops are integrating their data analysis into Hadoop jobs—opportunity to merge data from multiple systems including MongoDB for analysis.
- Hadoop is not a panacea—queries won't necessarily be fast(er) but it can handle larger scale and free up your MongoDB servers for long jobs.

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, Ruby or Perl

MongoDB + MapReduce in Java I

We’re going to walk through a sample application which parses US Treasury Bond historical Bid Curves since January 1990, and calculates an annual average for the 10 year Treasury.

- A sample of our dataset:

code/sample_treasury.js

```
{ "_id" : { "$date" : 631238400000 }, "dayOfWeek" : "TUESDAY",
    "bc3Year" : 7.9, "bc5Year" : 7.87, "bc10Year" : 7.94,
    "bc20Year" : null, "bc1Month" : null, "bc2Year" : 7.87,
    "bc3Month" : 7.83, "bc30Year" : 8, "bc1Year" : 7.81,
    "bc7Year" : 7.98, "bc6Month" : 7.89 }
{ "_id" : { "$date" : 631324800000 }, "dayOfWeek" : "WEDNESDAY",
     "bc3Year" : 7.96, "bc5Year" : 7.92, "bc10Year" : 7.99,
    "bc20Year" : null, "bc1Month" : null, "bc2Year" : 7.94,
    "bc3Month" : 7.89, "bc30Year" : 8.039999999999999,
    "bc1Year" : 7.85, "bc7Year" : 8.039999999999999, "bc6Month"
    : 7.94 }
```

MongoDB + MapReduce in Java II

```
{ "_id" : { "$date" : 631411200000 }, "dayOfWeek" : "THURSDAY",
    "bc3Year" : 7.93, "bc5Year" : 7.91, "bc10Year" : 7.98,
    "bc20Year" : null, "bc1Month" : null, "bc2Year" : 7.92,
    "bc3Month" : 7.84, "bc30Year" : 8.039999999999999,
    "bc1Year" : 7.82, "bc7Year" : 8.02, "bc6Month" : 7.9 }
{ "_id" : { "$date" : 631497600000 }, "dayOfWeek" : "FRIDAY",
    "bc3Year" : 7.94, "bc5Year" : 7.92, "bc10Year" : 7.99,
    "bc20Year" : null, "bc1Month" : null, "bc2Year" : 7.9,
    "bc3Month" : 7.79, "bc30Year" : 8.06, "bc1Year" : 7.79,
    "bc7Year" : 8.029999999999999, "bc6Month" : 7.85 }
```

MongoDB + MapReduce in Java III

- The MongoDB JavaScript version of our demo:

code/mongo_treasury_mr.js

```
function m() {
    emit( this._id.getYear(), { count: 1, sum: this.bc10Year })
}

function r( year, values ) {
    var n = { count: 0,  sum: 0 }
    for ( var i = 0; i < values.length; i++ ){
        n.sum += values[i].sum;
        n.count += values[i].count;
    }

    return n;
}

function f( year, value ){
    value.avg = value.sum / value.count;
    return value.avg;
}
```

MongoDB + MapReduce in Java IV

```

result = db.yield_historical.in.mapReduce( m , r, { finalize: f
    });
```

MongoDB + MapReduce in Java V

- The same work can be done in Hadoop. . .

MongoDB Hadoop Job for Treasury Data I
Configuration

The Configuration is specified in XML

code/mongo–treasury_yield.xml

```
<?xml version="1.0"?>
<configuration>
  <property>
    <!-- run the job verbosely ? -->
    <name>mongo.job.verbose</name>
    <value>true</value>
  </property>
  <property>
    <!-- Run the job in the foreground and wait for response, or
    background it? -->
    <name>mongo.job.background</name>
    <value>false</value>
  </property>
  <property>
    <!-- If you are reading from mongo, the URI -->
    <name>mongo.input.uri</name>
    <value>mongodb://localhost/demo.yield_historical.in</value>
```

MongoDB Hadoop Job for Treasury Data II

Configuration

```
  </property>
  <property>
    <!-- If you are writing to mongo, the URI -->
    <name>mongo.output.uri</name>
    <value>mongodb://localhost/demo.yield_historical.out</value>
  </property>
  <property>
    <!-- The query, in JSON, to execute [OPTIONAL] -->
    <name>mongo.input.query</name>
    <!--<value>{"x": {"$regex": "^eliot", "$options": ""}}</value>-->
    <value></value>
  </property>
  <property>
    <!-- The fields, in JSON, to read [OPTIONAL] -->
    <name>mongo.input.fields</name>
    <value></value>
  </property>
  <property>
    <!-- A JSON sort specification for read [OPTIONAL] -->
    <name>mongo.input.sort</name>
    <value></value>
```

MongoDB Hadoop Job for Treasury Data III

Configuration

```
  </property>
  <property>
    <!-- The number of documents to limit to for read [OPTIONAL] -->
    <name>mongo.input.limit</name>
    <value>0</value> <!-- 0 == no limit -->
  </property>
  <property>
    <!-- The number of documents to skip in read [OPTIONAL] -->
    <!-- TODO - Are we running limit() or skip() first? -->
    <name>mongo.input.skip</name>
    <value>0</value> <!-- 0 == no skip -->
  </property>
  <property>
    <!-- Class for the mapper -->
    <name>mongo.job.mapper</name>
    <value>com.mongodb.hadoop.examples.TreasuryYieldMapper</value>
  </property>
  <property>
    <!-- Reducer class -->
    <name>mongo.job.reducer</name>
    <value>com.mongodb.hadoop.examples.TreasuryYieldReducer</value>
```

MongoDB Hadoop Job for Treasury Data IV
Configuration

```
  </property>
  <property>
    <!-- InputFormat Class -->
    <name>mongo.job.input.format</name>
    <value>com.mongodb.hadoop.MongoInputFormat</value>
  </property>
  <property>
    <!-- OutputFormat Class -->
    <name>mongo.job.output.format</name>
    <value>com.mongodb.hadoop.MongoOutputFormat</value>
  </property>
  <property>
    <!-- Output key class for the output format -->
    <name>mongo.job.output.key</name>
    <value>org.apache.hadoop.io.IntWritable</value>
  </property>
  <property>
    <!-- Output value class for the output format -->
    <name>mongo.job.output.value</name>
    <value>org.apache.hadoop.io.DoubleWritable</value>
  </property>
```

MongoDB Hadoop Job for Treasury Data V
Configuration

```
  <property>
    <!-- Output key class for the mapper [optional] -->
    <name>mongo.job.mapper.output.key</name>
    <value></value>
    <value>org.apache.hadoop.io.IntWritable</value>
  </property>
  <property>
    <!-- Output value class for the mapper [optional] -->
    <name>mongo.job.mapper.output.value</name>
    <value>org.apache.hadoop.io.DoubleWritable</value>
  </property>
  <property>
    <!-- Class for the combiner [optional] -->
    <name>mongo.job.combiner</name>
    <value>com.mongodb.hadoop.examples.TreasuryYieldReducer</value>
  </property>
  <property>
    <!-- Partitioner class [optional] -->
    <name>mongo.job.partitioner</name>
    <value></value>
  </property>
```

MongoDB Hadoop Job for Treasury Data VI

Configuration

```
  <property>
    <!-- Sort Comparator class [optional] -->
    <name>mongo.job.sort_comparator</name>
    <value></value>
  </property>

</configuration>
```

MongoDB Hadoop Job for Treasury Data I
Configuration

The Job is loaded & configured via a Java Tool:

code/TreasuryYieldXMLConfig.java

```java
// TreasuryYieldXMLConfig.java
/*
 * Copyright 2010 10gen Inc.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * ...
 */
package com.mongodb.hadoop.examples;

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

import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.*;
import org.apache.hadoop.util.*;
import org.bson.*;
```

MongoDB Hadoop Job for Treasury Data II

Configuration

```java

import com.mongodb.hadoop.util.*;

public class TreasuryYieldXMLConfig extends MongoTool {

    static {
        // Load the XML config defined in hadoop-local.xml
        Configuration.addDefaultResource(
    "src/examples/hadoop-local.xml" );
        Configuration.addDefaultResource(
    "src/examples/mongo-defaults.xml" );
    }

    public static void main( String[] args ) throws Exception{
        final int exitCode = ToolRunner.run( new
    TreasuryYieldXMLConfig(), args );
        System.exit( exitCode );
    }
}
```

MongoDB Hadoop Job for Treasury Data I
Configuration

MongoTool is a helper utility for doing XML configs with queries, etc:

code/MongoTool.java

```java
// MongoTool.java
/*
 * Copyright 2010 10gen Inc.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * ...
 */

package com.mongodb.hadoop.util;

import java.util.Map.Entry;

import org.apache.commons.logging.*;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.mapreduce.*;
import org.apache.hadoop.util.*;

```

MongoDB Hadoop Job for Treasury Data II
Configuration

```
/**
 * Tool for simplifying the setup and usage of Mongo Hadoop jobs
 * using the Tool / Configured interfaces for use w/ a ToolRunner
 * Primarily useful in cases of XML Config files.
 *
 * @author Brendan W. McAdams <brendan@10gen.com>
 */
public class MongoTool extends Configured implements Tool {
    private static final Log log = LogFactory.getLog( MongoTool.class
     );

    public int run( String[] args ) throws Exception{
        /**
         * ToolRunner will configure/process/setup the config
         * so we need to grab the classlevel one
         * This will be inited with any loaded xml files or -D
    prop=value params
         */
        final Configuration conf = getConf();

```

MongoDB Hadoop Job for Treasury Data III
Configuration

```
        log.info( "Created a conf: '" + conf + "' on {" +
    this.getClass() + "} as job named '" + _jobName + "'" );

        for ( final Entry<String, String> entry : conf ) {
            log.trace( String.format( "%s=%s\n", entry.getKey(),
    entry.getValue() ) );
        }

        final Job job = new Job( conf , _jobName );
        /**
         * Any arguments specified with -D <property>=<value>
         * on the CLI will be picked up and set here
         * They override any XML level values
         * Note that -D<space> is important - no space will
         * not work as it get spicked up by Java itself
         */
        // TODO - Do we need to set job name somehow more
    specifically?
        // This may or may not be correct/sane
        job.setJarByClass( this.getClass() );
        final Class mapper = MongoConfigUtil.getMapper( conf );
```

MongoDB Hadoop Job for Treasury Data IV
Configuration

```java

        log.info( "Mapper Class: " + mapper );
        job.setMapperClass( mapper );
        job.setCombinerClass( MongoConfigUtil.getCombiner( conf ) );
        job.setReducerClass( MongoConfigUtil.getReducer( conf ) );

        job.setOutputFormatClass( MongoConfigUtil.getOutputFormat( conf ) );
        job.setOutputKeyClass( MongoConfigUtil.getOutputKey( conf ) );
        job.setOutputValueClass( MongoConfigUtil.getOutputValue( conf ) );

        job.setInputFormatClass( MongoConfigUtil.getInputFormat( conf ) );

        /**
         * Determines if the job will run verbosely e.g. print debug output
         * Only works with foreground jobs
         */
```

MongoDB Hadoop Job for Treasury Data V

Configuration

```java
    final boolean verbose = MongoConfigUtil.isJobVerbose( conf );
    /**
     * Run job in foreground aka wait for completion or background?
     */
    final boolean background = MongoConfigUtil.isJobBackground( conf );
    try {
        if ( background ) {
            log.info( "Setting up and running MapReduce job in background." );
            job.submit();
            return 0;
        }
        else {
            log.info( "Setting up and running MapReduce job in foreground, will wait for results.  {Verbose? " + verbose + "}" );
            return job.waitForCompletion( true ) ? 0 : 1;
        }
    }
```

MongoDB Hadoop Job for Treasury Data VI
Configuration

```
        catch ( final Exception e ) {
            log.error( "Exception while executing job... ", e );
            return 1;
        }
    }

    /**
     * Main will be a necessary method to run the job - suggested
    implementation
     * template:
     * public static void main(String[] args) throws Exception {
     * int exitCode = ToolRunner.run(new <YourClass>(), args);
     * System.exit(exitCode);
     * }
     *
     */

    /**
     * SET ME
     * Defines the name of the job on the cluster.
     * Left non-final to allow tweaking with serial #s, etc
```

MongoDB Hadoop Job for Treasury Data VII
Configuration

```
         **/
        String _jobName = "<unnamed MongoTool job>";
}
```

MongoDB Hadoop Job for Treasury Data I

Mapper

The Map function:

code/TreasuryYieldMapper.java

```java
// TreasuryYieldMapper.java
/*
 * Copyright 2010 10gen Inc.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * ...
 */
package com.mongodb.hadoop.examples;

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

import org.apache.commons.logging.*;

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

```

MongoDB Hadoop Job for Treasury Data II
Mapper

```java
import org.apache.hadoop.mapreduce.*;
import org.apache.hadoop.util.*;
import org.bson.*;

import com.mongodb.hadoop.util.*;

public class TreasuryYieldMapper extends Mapper<java.util.Date,
    BSONObject, IntWritable, DoubleWritable> {

    private static final Log log = LogFactory.getLog(
    TreasuryYieldMapper.class );

    public void map( java.util.Date key , BSONObject value , Context
    context ) throws IOException, InterruptedException{

        int year = key.getYear() + 1900;
        double bid10Year = ((Number) value.get( "bc10Year"
    )).doubleValue();

        context.write( new IntWritable( year ), new DoubleWritable(
    bid10Year ) );
```

MongoDB Hadoop Job for Treasury Data III
Mapper

```

        }

}
```

MongoDB Hadoop Job for Treasury Data I

Reducer

The Reduce function:

code/TreasuryYieldReducer.java

```
// TreasuryYieldMapper.java
/*
 * Copyright 2010 10gen Inc.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * ...
 */
package com.mongodb.hadoop.examples;

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

import org.apache.commons.logging.*;

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

```

MongoDB Hadoop Job for Treasury Data II

Reducer

```java
import org.apache.hadoop.mapreduce.*;
import org.apache.hadoop.util.*;
import org.bson.*;

import com.mongodb.hadoop.util.*;

public class TreasuryYieldReducer extends Reducer<IntWritable,
    DoubleWritable, IntWritable, DoubleWritable> {

    private static final Log log = LogFactory.getLog(
    TreasuryYieldReducer.class );

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

        int count = 0;
        double sum = 0;
        for ( final DoubleWritable value : values ) {
            log.debug( "Key: " + key + " Value: " + value );
            sum += value.get();
```

MongoDB Hadoop Job for Treasury Data III

Reducer

```java
                count++;
            }

            double avg = sum / count;

            log.info( "Average 10 Year Treasury for " + key.get() + " was
        " + avg );

            context.write( key, new DoubleWritable( avg ) );
        }

}
```

mongoDB

MongoDB Hadoop Job For Treasury Data
Running

- Make sure the Java Driver is in the 'lib' directory on every cluster member, e.g. `sudo cp lib/mongo-java-driver-2.3.jar /usr/lib/hadoop-0.20/lib/` (Bounce hadoop after adding libraries)
- Build the Hadoop Jar (`ant resolve; ant jar`)
- Run the Job!

A Modified Pig Tutorial with MongoDB Output I

- We took the stock Pig Tutorial and modified it to save to MongoDB
- Input file is an anonymized Search Engine Log:

code/excite_log_snippet.txt

```
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
```

A Modified Pig Tutorial with MongoDB Output II

```
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
```

- The Script is only lightly modified from stock pig:

code/pigtutorial.pig

```
/*
 * Licensed to the Apache Software Foundation (ASF) under one
 * or more contributor license agreements.  See the NOTICE file
 * distributed with this work for additional information
 * regarding copyright ownership.  The ASF licenses this file
 * to you under the Apache License, Version 2.0 (the
 * "License"); you may not use this file except in compliance
```

A Modified Pig Tutorial with MongoDB Output III

```
 * with the License.  You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing,
    software
 * distributed under the License is distributed on an "AS IS"
    BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express
    or implied.
 * See the License for the specific language governing
    permissions and
 * limitations under the License.
 */

-- 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.

```

A Modified Pig Tutorial with MongoDB Output IV

```
-- 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 examples/pigtutorial/lib/pigtutorial.jar;
REGISTER mongo-hadoop.jar;
REGISTER lib/mongo-java-driver-2.3.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;
```

A Modified Pig Tutorial with MongoDB Output V

```

-- 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;

-- 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);

```

A Modified Pig Tutorial with MongoDB Output VI

```
-- 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;

```

A Modified Pig Tutorial with MongoDB Output VII

```
-- 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.
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/demo.pig.output' USING
     com.mongodb.hadoop.pig.MongoStorage;
```

Mongo Hadoop Plugin

- Starting today, you can download and use the Hadoop Plugin: `https://github.com/mongodb/mongo-hadoop`
- Still an Alpha—*not* production ready.
- Continuing Development...
  - Working on support for Hadoop Streaming, which allows Python, Ruby, and Perl to be used instead of Java (Waiting to migrate to .21 for Binary support)
  - Pig Input support.
  - Support for Sharding concurrency (Once one of you asks for it <hint-hint>)
  - Exploring other complimentary toolchains like Cascading.
  - Tell us how you are using Hadoop or how you want to integration MongoDB + Hadoop... So we can focus on the areas you, the users, want and need.

Questions?

- Twitter: **@rit** | mongodb: **@mongodb** | 10gen: **@10gen**
- email: **brendan@10gen.com**
- mongo-hadoop . . . Available in a Pre-Production Alpha: `https://github.com/mongodb/mongo-hadoop`
- Report Bugs, request features: `https://github.com/mongodb/mongo-hadoop/issues`
- Pressing Questions?
  - IRC - freenode.net **#mongodb**
  - MongoDB Users List - `http://groups.google.com/group/mongodb-user`
- 10gen is *hiring!* We need smart engineers in both NY and Bay Area: `http://10gen.com/jobs`
- Up Next: 15 Minute Break followed by Mathias Stearn on "MongoDB Internals: The Storage Engine"