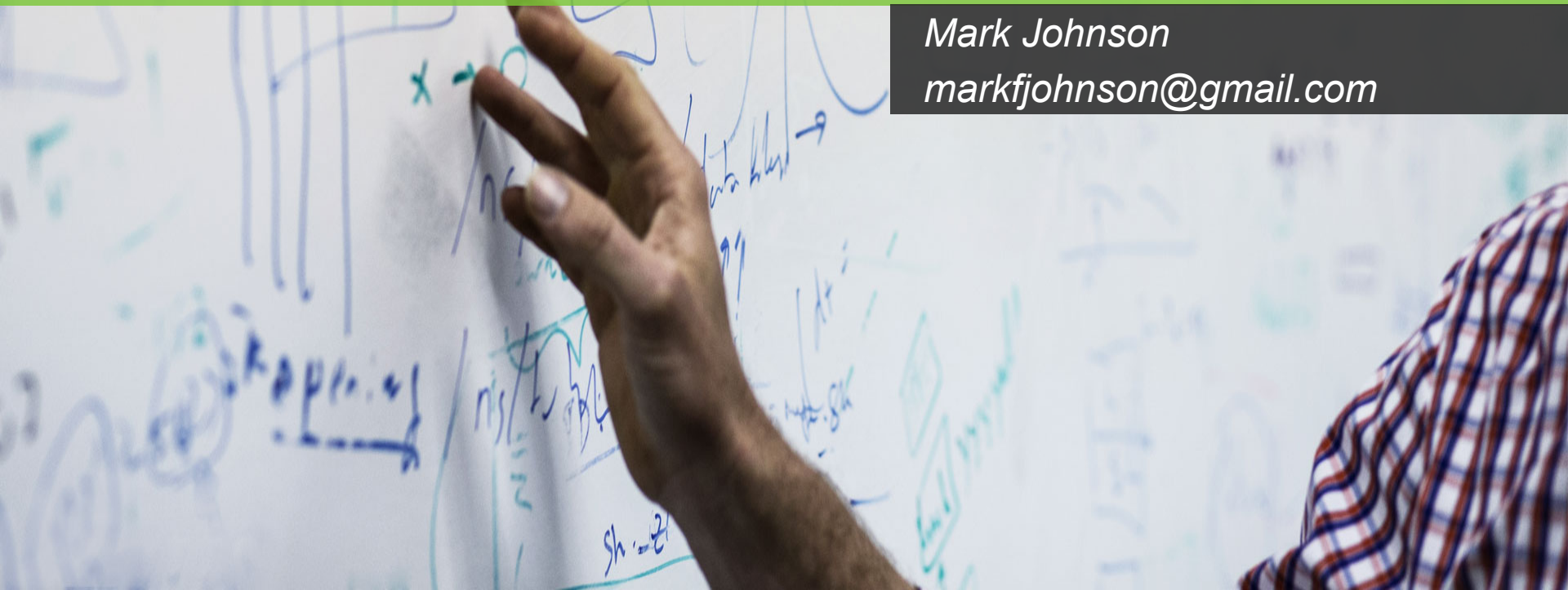
Applying Testing Techniques to Hadoop Development



Mark Johnson
markfjohnson@gmail.com

Who Am I?

Mark Johnson
Regional Director Services -
Hortonworks

- **Too many years of programming experience**
- **Created lots of bugs in my career…but hate for them to be found by others**

Hortonworks

Who are You?

- Technologist
- Programmer

- Want to produce low defect or defect free code in Hadoop

Good Hadoop Development is Hard!


Lots of Data !

- **Volume**
- **Velocity**
- **Variety**

Fast Release Pressure!

BUGS!

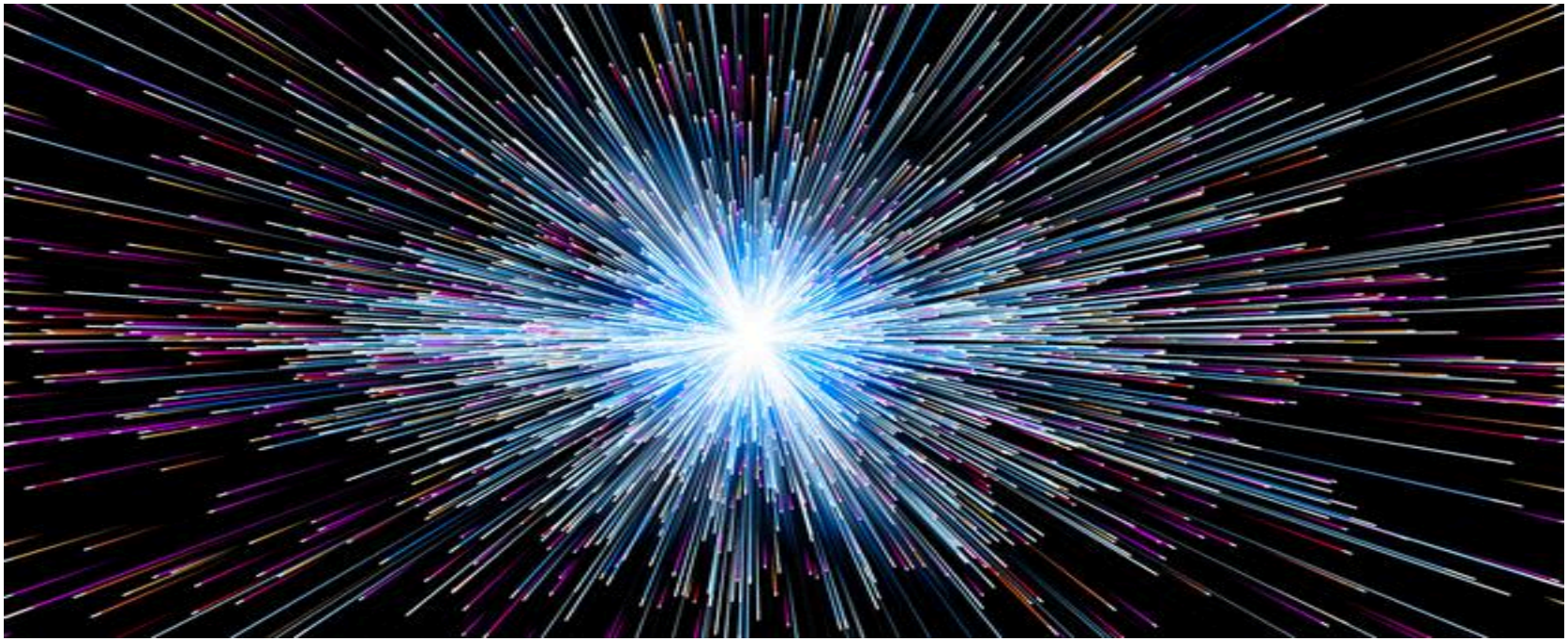
Will anyone really notice that little problem which affects only 1% of the rows?

YES!

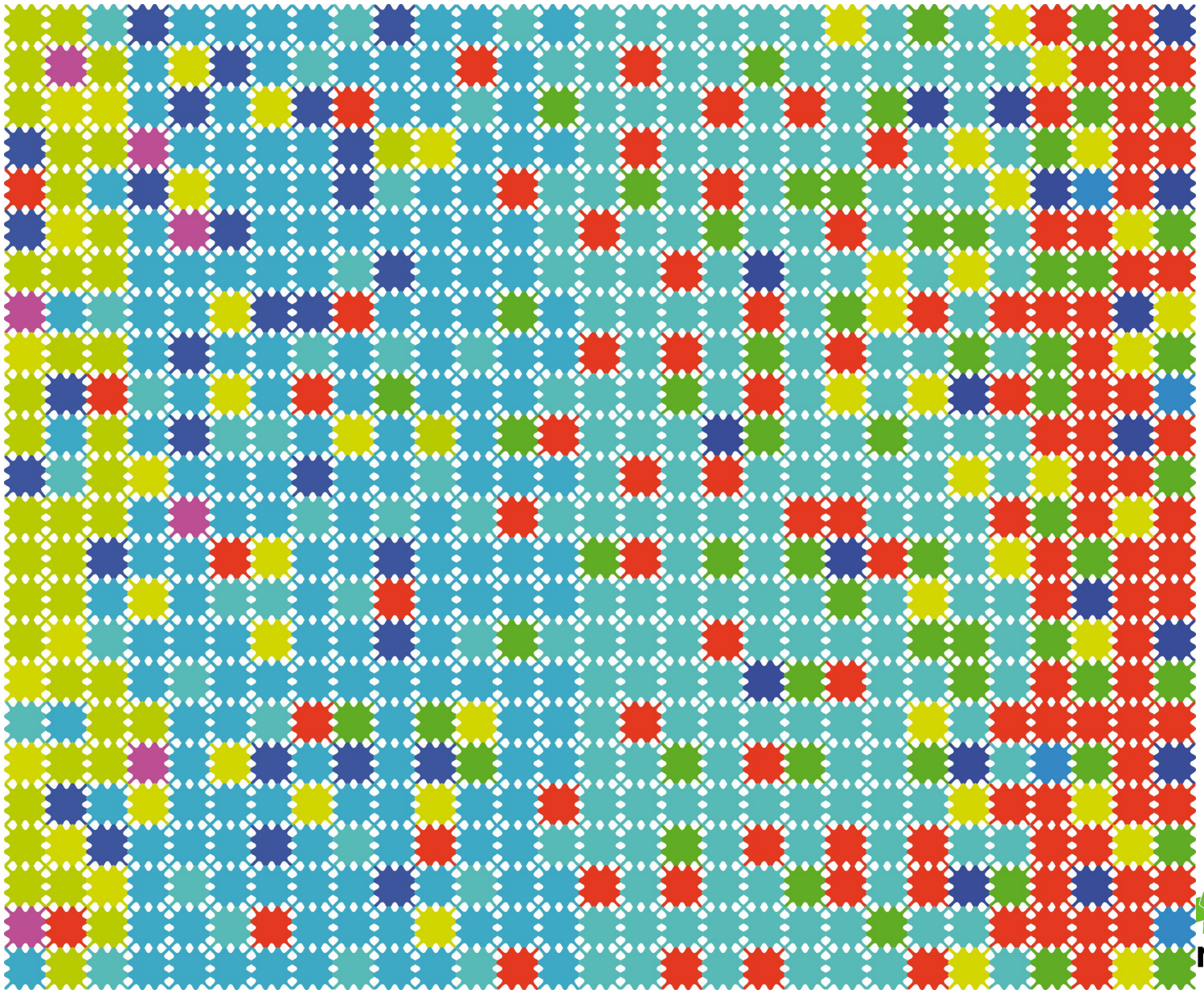
1% of the financial transactions on Wall Street is a lot of money!

Big Data can amplify small problems!

Problem #1: Speed of Light

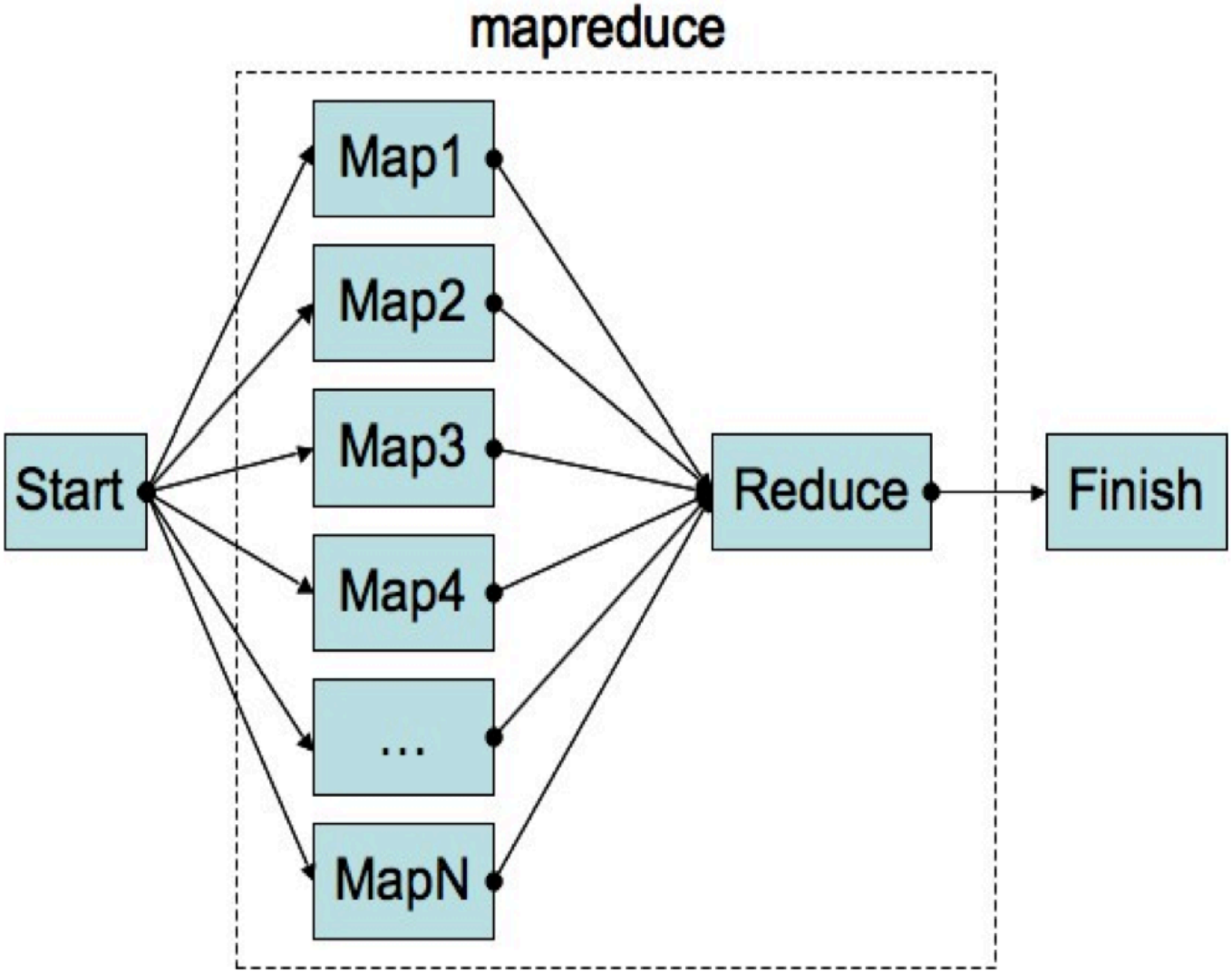


Problem #2: Data Permutations



nworks

Problem #3: Processing distributed across multiple nodes



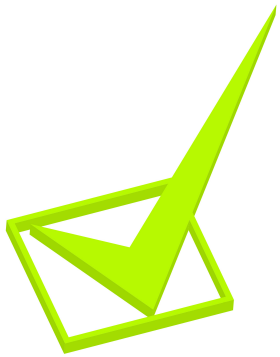
onworks

How are you currently preventing BUGS?

Consequences

The Longer you wait to find and fix a problem the more time it will take to fix it!

Verify “DONE”



What 4 things do we need to Verify?

1. Does it run?

2.) Functional Correctness to Requirements

- Data Filters
- Loops and branches
- Calculations
- Output

3. Resources Correctly Referenced

- Missing files
- Bad file names
- Etc.

4. Exceptions and errors properly handled

- System left in a safe state
- User / SysAdmin informed of the failure
- Idempotency

Materiality Rule

Materiality Rule: Start with High value tests (easy to test AND valued by business)

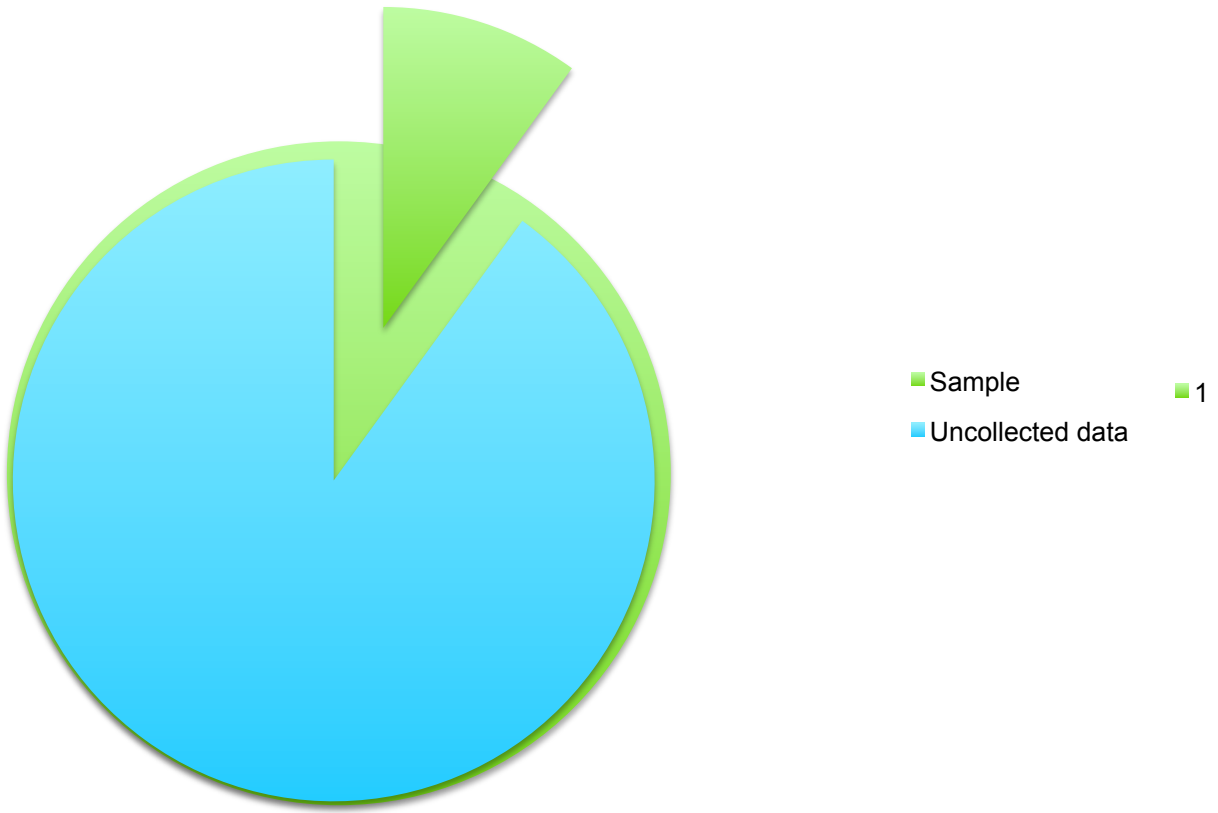
Light & Fast : Don’t create an overly heavy test process

Positive Tests:

- Tests should contain one general data condition
- Individual tests only for those determined by specific data conditions

Negative Tests:

- Data not present



Hortonworks

Traditional Sampling Problems

- **Sample error** reflects the risk that, purely by chance, a randomly chosen sample of opinions does not reflect the true views of the population. The “margin of error” reported in opinion polls reflects this risk and the larger the sample, the smaller the margin of error
- **sampling bias** is a bias in which a sample is collected in such a way that some members of the intended population are less likely to be included than others. (wikipedia)

PIG SAMPLE

```
A = LOAD '2014QTR1_num.dat' USING PigStorage('\t');
X = SAMPLE A 0.01;
B = group X all;
C = foreach B generate COUNT(X);
STORE C INTO 'Results.txt';
```

Manual Sampling

- **Build the sample dataset based on your program's logic.**
- **Does not need to be more than a few rows for each test.**
- **Can get embedded within your test program to facilitate test management.**

Apache MRUnit™

- MapReduce testing framework
- Accepts a small record sample for each test
- Test one thing per test
- Does not reference HDFS directly
- Tests Map and Reduce methods separetly

Example: Word Count Program - Mapper

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

    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();
    List stopList = new ArrayList<String>(Arrays.asList("a","the","this","it","there","can","be"));


    public void map(Object key, Text value, Context context
    ) throws IOException, InterruptedException {
        StringTokenizer itr = new StringTokenizer(value.toString());
        while (itr.hasMoreTokens()) {
            String tok = itr.nextToken();
            if (stopList.contains(tok.toLowerCase())) {
                System.out.println(tok);
                word.set(tok);
                context.write(word, one);
            }
        }
    }
}
```

- **Does the tokenize work properly**
- **Does the Mapper properly filter ‘stop’ words**
- **Is the counter properly initialized**

Example: Word Count - Reducer

```
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);
    }
}
```

- **Keys counter values grouped properly**
- **Counter values are properly aggregated**

Setup MRUnit test suite

```
@Before
public void setUp() {

    WordCount.TokenizerMapper mapper = new WordCount.TokenizerMapper();
    WordCount.IntSumReducer reducer = new WordCount.IntSumReducer();


    mapDriver = MapDriver.newMapDriver(mapper);
    reduceDriver = ReduceDriver.newReduceDriver(reducer);
    mapReduceDriver = MapReduceDriver.newMapReduceDriver(mapper, reducer);
}
```

- **MapDriver:**
- **ReduceDriver:**
- **MapReduceDriver:**

MRUnit Mapper test

```
@Test
public void testMapper() throws Exception {
    String inputString = "This map feature can be used when This map tasks";
    mapDriver.withInput(new LongWritable(), new Text(inputString));
    mapDriver.withOutput(new Text("map"), new IntWritable(1));
    mapDriver.withOutput(new Text("feature"), new IntWritable(1));
    mapDriver.withOutput(new Text("used"), new IntWritable(1));
    mapDriver.withOutput(new Text("when"), new IntWritable(1));
    mapDriver.withOutput(new Text("map"), new IntWritable(1));
    mapDriver.withOutput(new Text("tasks"), new IntWritable(1));

    mapDriver.runTest();
}
```

Test Results

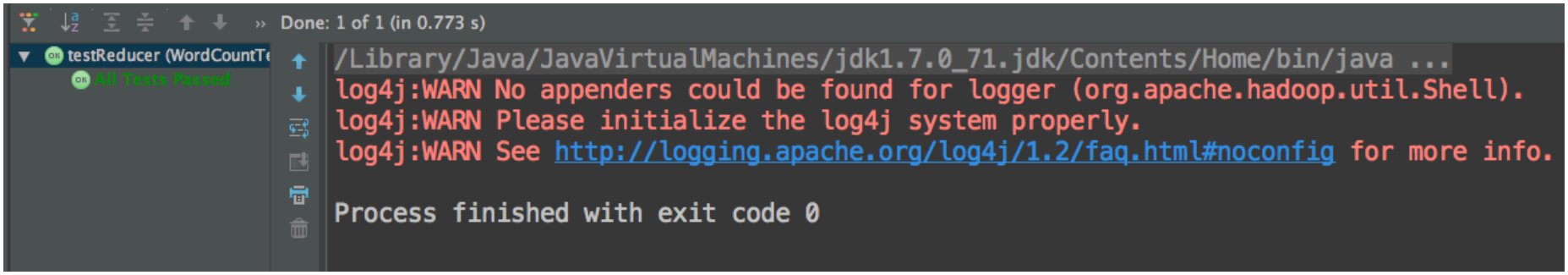
```
java.lang.AssertionError: 6 Error(s): (Missing expected output (map, 1) at position 0, got (This, 1).
    at org.apache.hadoop.mrunit.internal.util.Errors.assertNone(Errors.java:73)
    at org.apache.hadoop.mrunit.TestDriver.validate(TestDriver.java:768)
    at org.apache.hadoop.mrunit.TestDriver.runTest(TestDriver.java:641)
    at org.apache.hadoop.mrunit.TestDriver.runTest(TestDriver.java:627)
    at WordCountTest.testMapper(WordCountTest.java:42) <24 internal calls>
```

MRUnit Reducer Test

```
@Test
public void testReducer() throws Exception {
    List<IntWritable> values = new ArrayList<~>();
    values.add(new IntWritable(1));
    values.add(new IntWritable(1));
    reduceDriver.withInput(new Text("map"), values);
    reduceDriver.withOutput(new Text("map"), new IntWritable(2));

    reduceDriver.runTest();
}
```

Test Results



Hortonworks

Traditional Pig Functional Testing tools

DUMP

ILLUSTRATE

DESCRIBE

EXPLAIN

PigUnit to test Pig scripts

```
data =
    LOAD 'input'
    AS (query:CHARARRAY);

queries_group =
    GROUP data
    BY query;

queries_count =
    FOREACH queries_group
    GENERATE
        group AS query,
        COUNT(data) AS total;

queries_ordered =
    ORDER queries_count
    BY total DESC, query;

queries_limit =
    LIMIT queries_ordered $n;

STORE queries_limit INTO 'output';
```

- **Only one LOAD operation per test script.**
- **PigUnit overrides STORE, LOAD and DUMP**

Hortonworks

PigUnit – Setup

```
String[] args = {
        "n=2",
};

PigTest test = null;

test = new PigTest("top_queries.pig", args);


String[] input = {
        "yahoo",
        "yahoo",
        "yahoo",
        "twitter",
        "facebook",
        "facebook",
        "linkedin",
};
```

- **PigTest references your unmodified pig script for testing.**
- **Input:**
  - Include just the rows required to test logic

PigUnit: Assert

```
String[] output = {
        "(yahoo,3)",
        "(facebook,2)",
};

test.assertOutput("data", input2, "queries_limit", output);
```

```
HadoopVersion   PigVersion  UserId  StartedAt   FinishedAt  Features
0.20.2  0.13.0  mjohnson    2015-03-11 09:44:16 2015-03-11 09:44:19 GROUP_BY,ORDER_BY,LIMIT

Success!

Job Stats (time in seconds):
JobId   Maps    Reduces MaxMapTime  MinMapTIme  AvgMapTime  MedianMapTime   MaxReduceTime   MinReduceTime   AvgReduceTime   MedianReducetime    Alias   Feature Outputs
job_local_0001  1   1   n/a n/a n/a n/a n/a n/a n/a n/a data,queries_count,queries_group    GROUP_BY,COMBINER
job_local_0002  1   1   n/a n/a n/a n/a n/a n/a n/a n/a queries_ordered SAMPLER
job_local_0003  1   1   n/a n/a n/a n/a n/a n/a n/a n/a queries_ordered ORDER_BY,COMBINER
job_local_0004  1   1   n/a n/a n/a n/a n/a n/a n/a n/a queries_ordered     file:/tmp/temp-1160886547/tmp409330481,

Input(s):

Output(s):
Successfully stored 2 records in: "file:/tmp/temp-1160886547/tmp409330481"

Counters:
Total records written : 2
Total bytes written : 0
Spillable Memory Manager spill count : 0
Total bags proactively spilled: 0
Total records proactively spilled: 0
```

PigUnit: Output Job Order

```
Job DAG:
job_local_0001 -> job_local_0002,
job_local_0002 -> job_local_0003,
job_local_0003 -> job_local_0004,
job_local_0004
```

BeeTest

Developed by: Adam Kawa

GitHub Project: https://github.com/kawaa/Beetest.git

Beetest provides a simple ‘unit’ test capability on a given Hadoop Hive script which runs on a small cluster to validate a Hive script

Beetest: Inputs and Outputs

Directory containing the following files defining the test process:

setup.hql – The HQL script to setup the environment

select.hql – The HQL script to test

Input.tsv – input data

Expected.txt – the script's expected output

variables.properties – the Beetest properties

Setting up BeeTest

```
[root@sandbox artist-count]# ls -lt
total 20
-rw-r--r-- 1 root root  19 Mar  1 19:02 expected.txt
-rw-r--r-- 1 root root 296 Mar  1 18:49 input.tsv
-rw-r--r-- 1 root root  97 Mar  1 18:49 select.hql
-rw-r--r-- 1 root root 246 Mar  1 18:49 setup.hql
-rw-r--r-- 1 root root  13 Mar  1 18:49 variables.properties
```

```
./run-test.sh artist-count local-config/
```

BeeTest: Executing the test

select.hql

```
SELECT artist, COUNT(*) AS cnt
  FROM ${table}
GROUP BY artist
ORDER BY cnt DESC
   LIMIT 2;
```

```
./run-test.sh artist-count local-config/
```

- **${table} – value defined in variables.properties**

BeeTest – Test Results

```
Mar 01, 2015 7:27:10 PM com.spotify.beetest.Utils runCommand
INFO: Time taken: 14.798 seconds
Mar 01, 2015 7:27:10 PM com.spotify.beetest.TestQueryExecutor run
INFO: Asserting: artist-count/expected.txt and /tmp/beetest-test-595069386-output_595069386/000000_0
[root@sandbox examples]#
```

BeeTest: Test failures

```
Mar 01, 2015 7:03:17 PM com.spotify.beetest.Utils runCommand
INFO: Time taken: 14.75 seconds
Mar 01, 2015 7:03:17 PM com.spotify.beetest.TestQueryExecutor run
INFO: Asserting: artist-count/expected.txt and /tmp/beetest-test-802489226-output_802489226/000000_0
Exception in thread "main" junitx.framework.ComparisonFailure: Output does not match Line [2] expected:<Coldplay       5> but was:<Coldplay    3> ['3']
        at junitx.framework.Assert.assertEquals(Assert.java:120)
        at junitx.framework.FileAssert.assertEquals(FileAssert.java:175)
        at junitx.framework.FileAssert.assertEquals(FileAssert.java:116)
        at com.spotify.beetest.TestQueryExecutor.run(TestQueryExecutor.java:62)
        at com.spotify.beetest.TestQueryExecutor.main(TestQueryExecutor.java:77)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:57)
        at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
        at java.lang.reflect.Method.invoke(Method.java:606)
        at org.apache.hadoop.util.RunJar.main(RunJar.java:212)
[root@sandbox examples]#
```

Getting started with your test initiative

1. **Start Simple**
2. **Materiality rule: Focus on Highest value and easiest tests first**
3. **Data Sampling: Use the smallest datasets possible**
4. **Keep tests “light and fast”**
5. **Keep Hadoop code and tests in a Source Code Management system**
6. **Use Automated test environment (Jenkins, AntHill Pro, etc.)**
7. **Maintain and publicly publish historical test results**

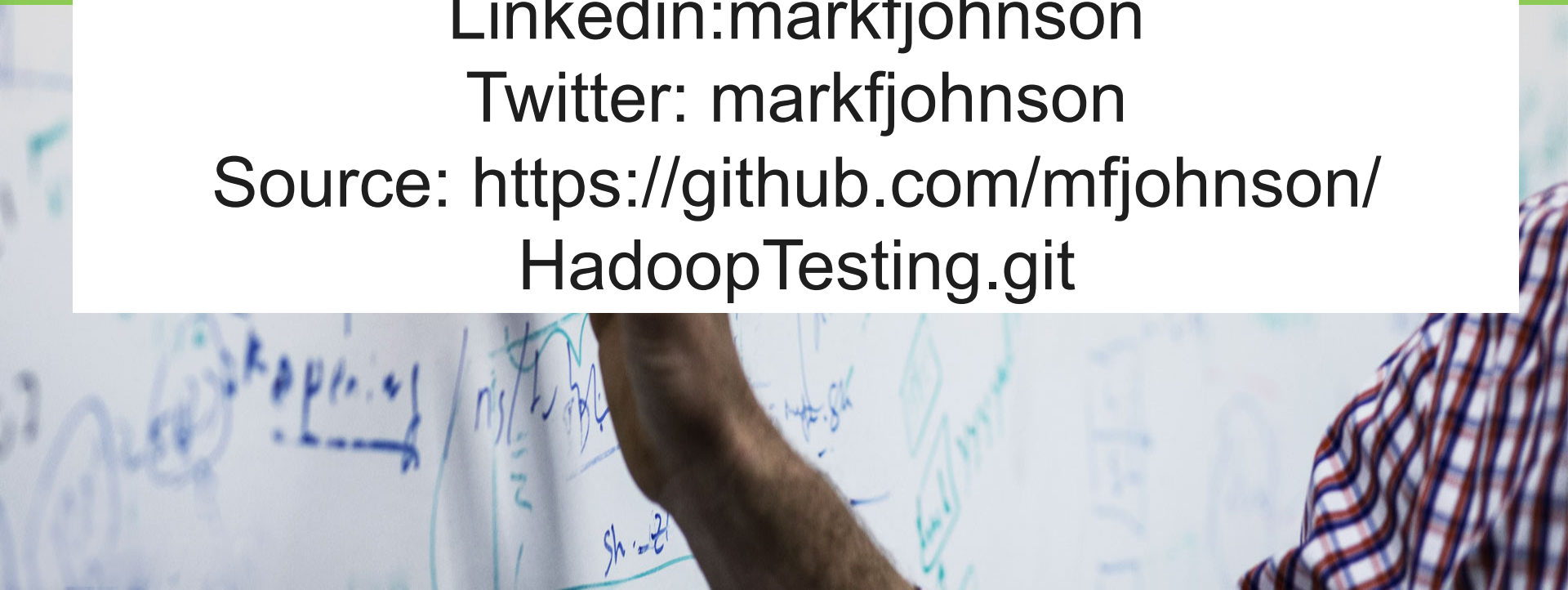
Hadoop Test Tool Wrap Up

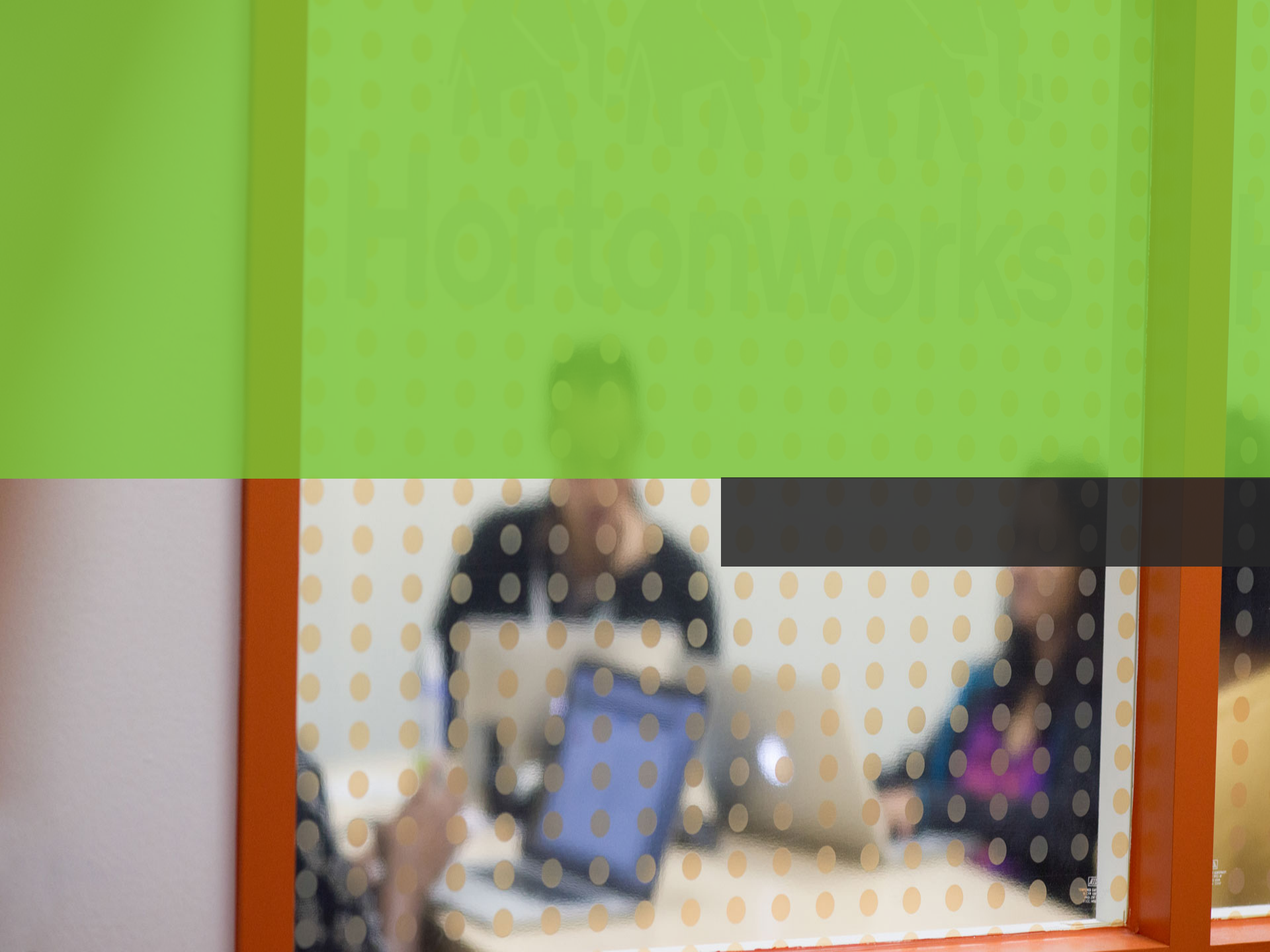
- **Tools and Techniques**
  - Data Sampling
  - MRUnit
  - PigUnit
  - Beetest

Testing environment still imature but good enough to start using now

Mark Johnson
Regional Director Services
Hortonworks

markfjohnson@gmail.com
mjohnson@hortonworks.com
Linkedin:markfjohnson
Twitter: markfjohnson
Source: https://github.com/mfjohnson/HadoopTesting.git




FortonMorks