

# Testing Hadoop Applications

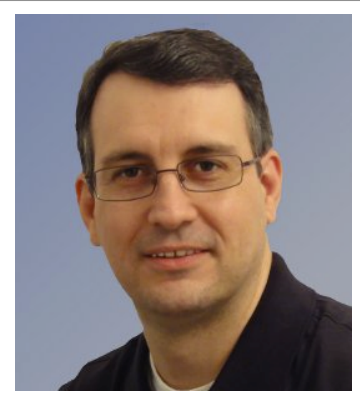
---

Tom Wheeler



# About The Instructor...

---



## Tom Wheeler

Software Engineer, etc.

Greater St. Louis Area | Information Technology and Services

---

**Current:**    **Curriculum Developer** at **Cloudera**

Past:    Principal Software Engineer at Object Computing, Inc. (Boeing)

Software Engineer, Level IV at WebMD

Senior Software Engineer at Teralogix

Senior Programmer/Analyst at A.G. Edwards and Sons, Inc.

# About The Presentation...

---

- What's ahead
  - Unit Testing
  - Integration Testing
  - Performance Testing
  - Diagnostics



# About The Audience...

---

- What I expect from you
  - Basic knowledge of Apache Hadoop
  - Basic understanding of MapReduce (Java)
  - Basic experience with UNIX / Linux

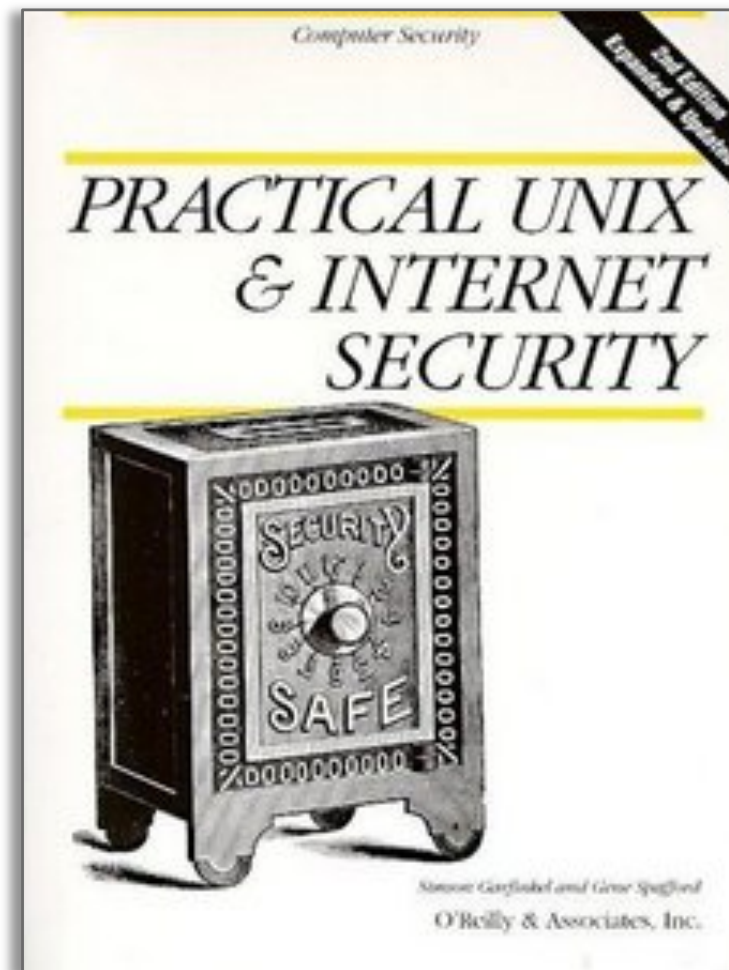




# Fundamental Concepts

---

- What is testing?
  - In my view, it's an application of computer security



“ A computer is secure if you can  
depend on it and its software  
to behave as you expect. ”

# Pre-Production Testing

---

- Unit testing
  - Verifies function of each “unit” in isolation
- Integration testing
  - Verifies that the system works as a whole
- The above tests should be done **before production**



# Performance Testing

---

- We also need to ensure efficient use of resources
  - This is done with performance testing
- Performance testing can be divided into two areas
  - Optimizing the code you've written
  - Optimizing cluster performance





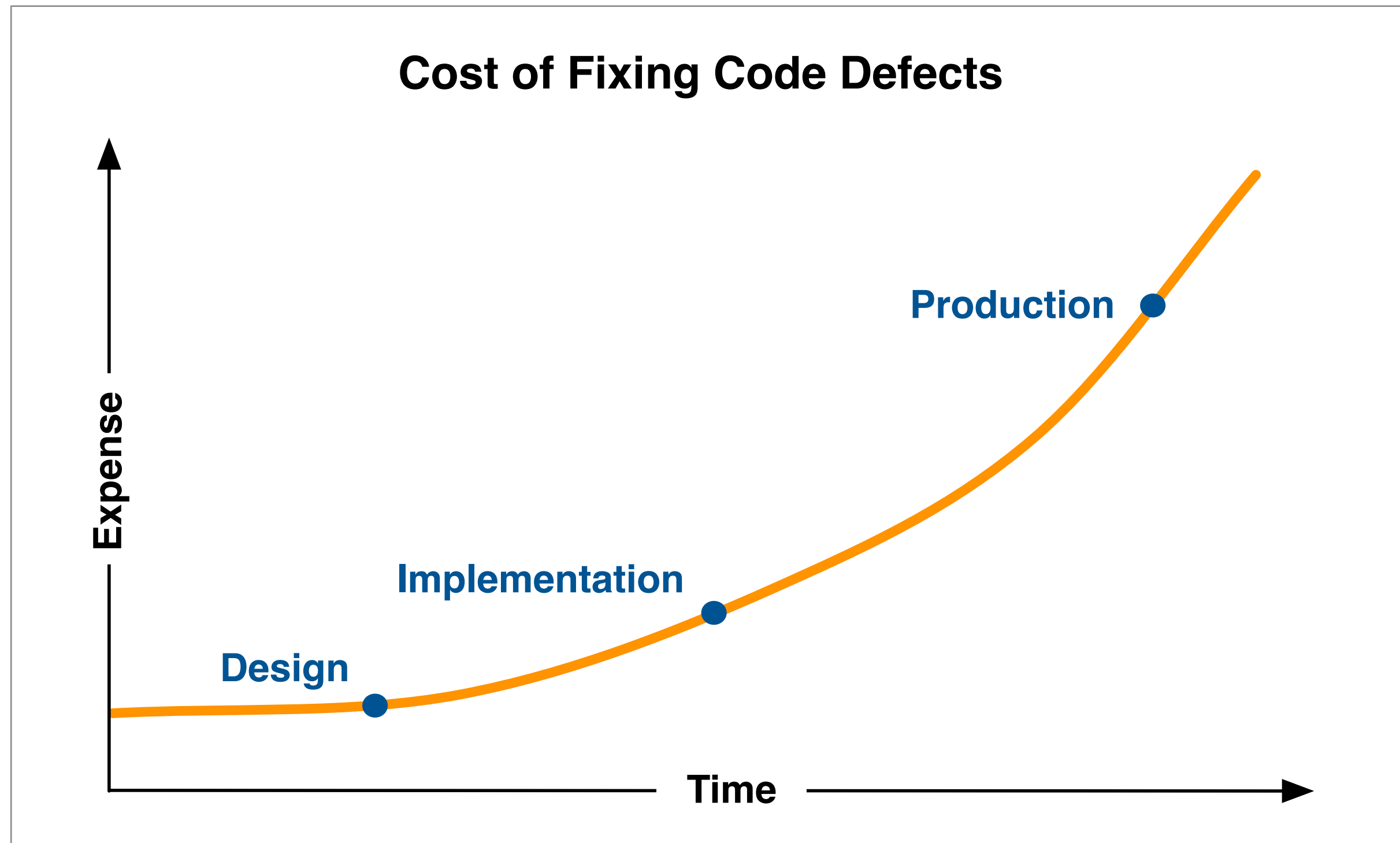
# Diagnostics

---

- We're not perfect...
  - Sometimes problems make it to production
- Diagnostics help us find and fix problems



# Why is Testing Important?



# Unit Testing



# Overview of Unit Testing

---

- Verifies correctness for a unit of code
- Key features of unit tests
  - Simple
  - Isolated
  - Deterministic
  - Automated





# Benefits of Unit Tests

---

- An investment in code quality
  - Can prevent regressions
- Actually saves development time
  - Tests run much faster than Hadoop jobs
  - Helps with refactoring



# Introducing JUnit

---

- JUnit is a popular library for Java unit testing
  - Open source (CPL)
  - Add JAR file to your project
  - Inspired many “xUnit” libraries for other languages



# JUnit Basics

Import statements removed for brevity

```
1 public class Calculator {  
2     public int add(int first, int second) {  
3         return first + second;  
4     }  
5 }
```

Our class

```
1 public class CalculatorTest {  
2  
3     private Calculator calc;  
4  
5     @Before  
6     public void setUp() {  
7         calc = new Calculator();  
8     }  
9  
10    @Test  
11    public void testAdd() {  
12        assertEquals(8, calc.add(5, 3));  
13    }  
14 }
```

Our test

# Log Event Counting Example

---

- Let's look at a simple MapReduce example
- Our goal is to summarize log events by level
  - Mapper: Parses log to extract level
  - Reducer: Sums up occurrences by level
- Seeing the data flow first will help illustrate the job





# Mapper Input

---

- Log data produced by an application using Log4J

```
2012-09-06 22:16:49.391 CDT INFO "This can wait"  
2012-09-06 22:16:49.392 CDT INFO "Blah blah"  
2012-09-06 22:16:49.394 CDT WARN "Hmmm..."  
2012-09-06 22:16:49.395 CDT INFO "More blather"  
2012-09-06 22:16:49.397 CDT WARN "Hey there"  
2012-09-06 22:16:49.398 CDT INFO "Spewing data"  
2012-09-06 22:16:49.399 CDT ERROR "Oh boy!"
```

# Mapper Output

---

- Key is log level parsed from a single line in the file
- Value is a literal 1 (since there's one level per line)

INFO	1
INFO	1
WARN	1
INFO	1
WARN	1
INFO	1
ERROR	1

# Reducer Input

---

- Hadoop sorts and groups the keys

<b>ERROR</b>	[1]
<b>INFO</b>	[1, 1, 1, 1]
<b>WARN</b>	[1, 1]

# Reducer Output

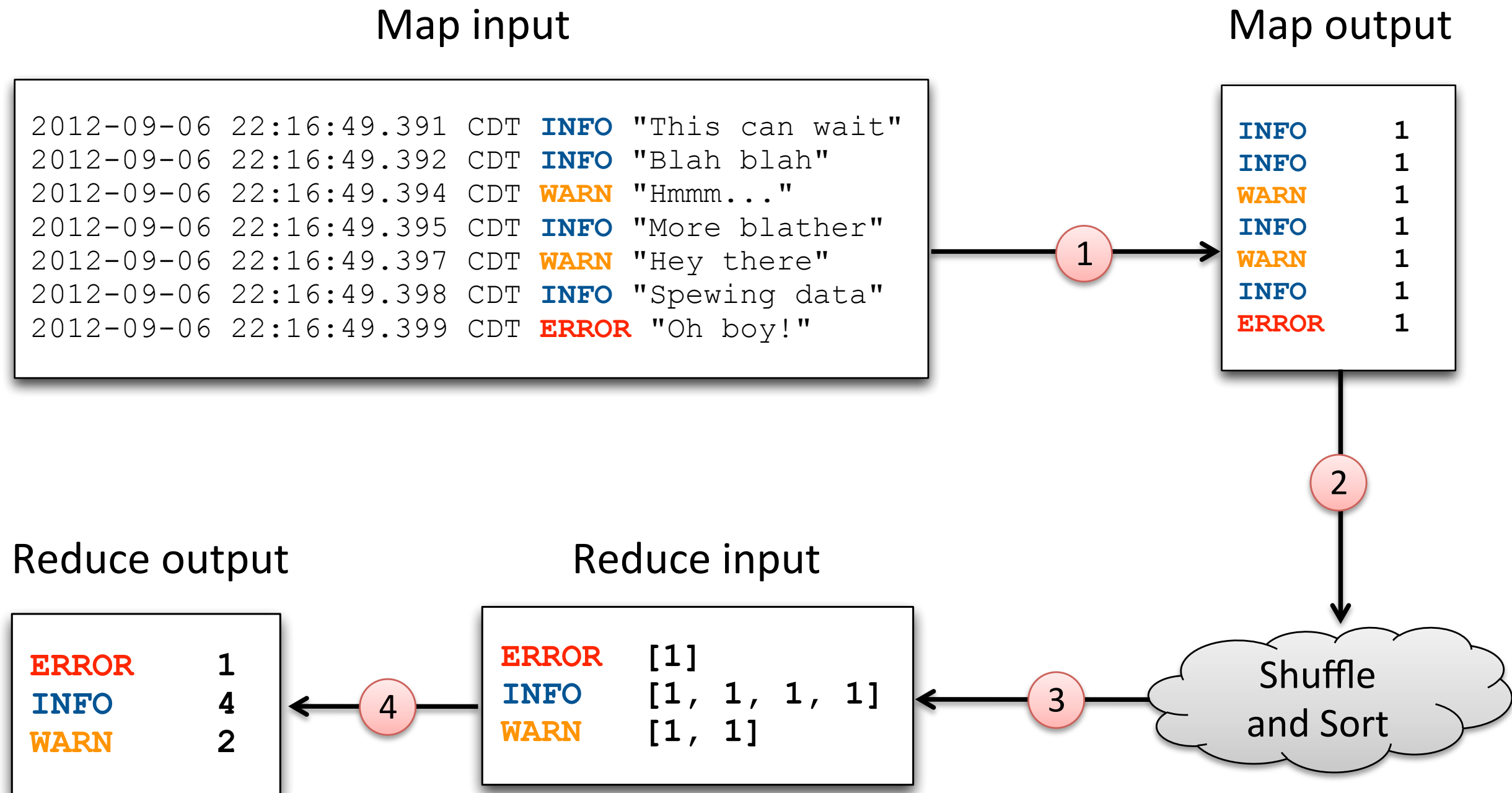
---

- The final output is a per-level summary

<b>ERROR</b>	<b>1</b>
<b>INFO</b>	<b>4</b>
<b>WARN</b>	<b>2</b>



# Data Flow for Entire MapReduce Job



# Show Me the Code...

---

- Let's examine the code for this job
  - Then I'll run it
  - And soon we'll see how to test and improve it



# What's Wrong With This?

---

- Mapper is complex and hard to test
  - How could we improve it?



# Oh No... We Found a Bug!

---

- We're missing some lines
  - Newer Log4J versions have an additional TRACE level



# Unit Testing and External Dependencies

---

- Test core business logic with JUnit
  - Mapper and Reducer still aren't fully tested
  - How can we verify *them*?



# Introducing MRUnit

---

- It's a JUnit extension for testing MapReduce code
  - Open source (Apache licensed)
  - Active development team
- Simulates much of Hadoop's core API, including
  - `InputSplit`
  - `OutputCollector`
  - `Reporter`
  - `Counters`





# MRUnit Demo

---

- I'll now demonstrate how to use MRUnit
  - Testing the Mapper
  - Testing the Reducer



# Testing Counters with MRUnit

---

- It's possible to further simplify this job with Counters
  - We don't even need a reducer at all
- Let's see how to rewrite the code
  - Then let's see how to test it with MRUnit



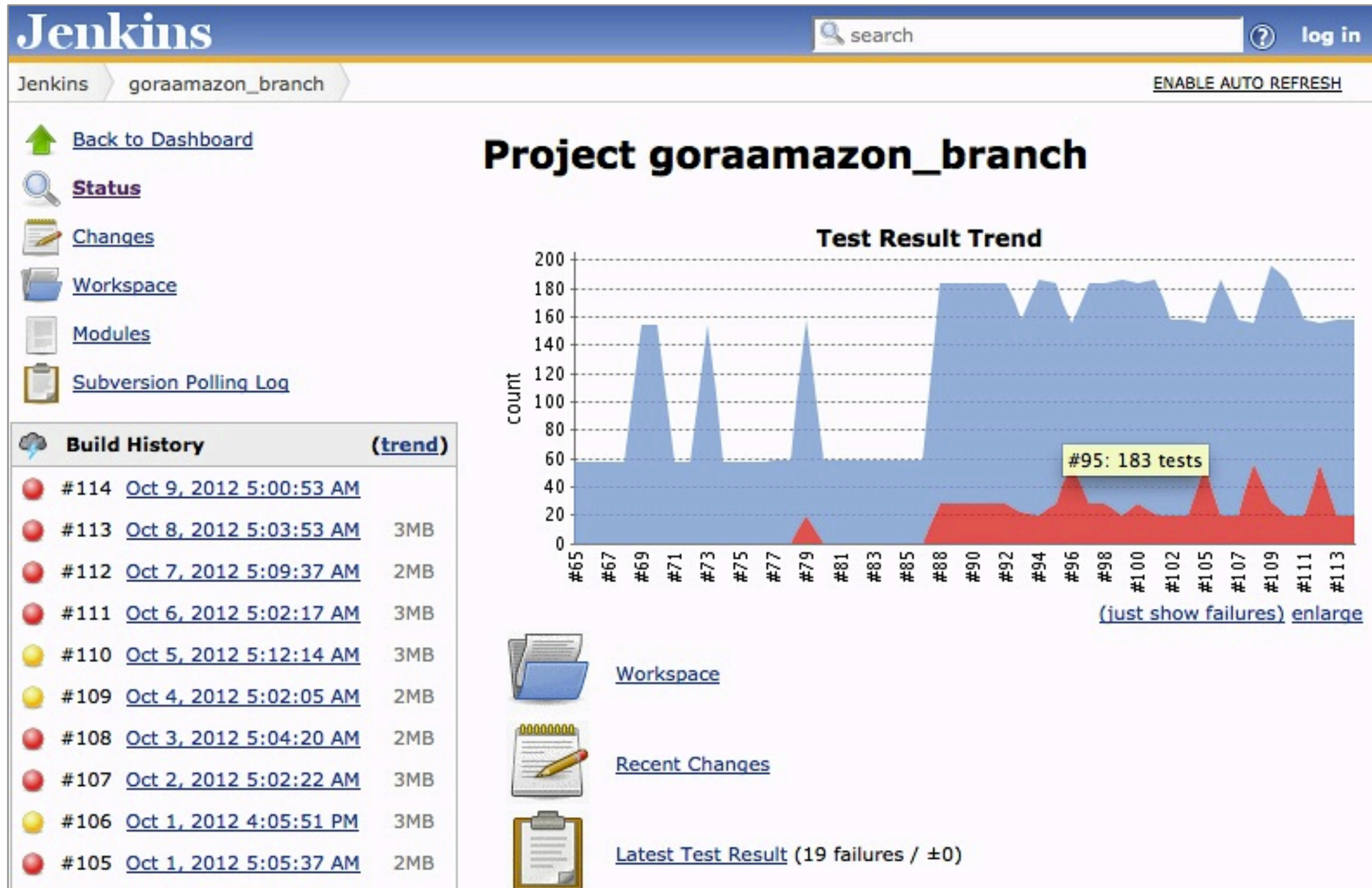
# Limitations of MRUnit

---

- There are a few things MRUnit can't test (yet)
  - Multiple lines of input
  - Jobs that use DistributedCache
  - Partitioners
  - Hadoop Streaming



# Automatically Running Tests with Jenkins



# Unit Testing Recap

---

- Your unit tests should have these properties
  - Simple
  - Isolated
  - Deterministic
  - Automated
- Decoupling your code aids in testing
  - Most business logic can be tested with JUnit
  - MRUnit is helpful for testing what remains





# Integration Testing



# Overview of Integration Testing

---

- Verifies that the system works as a whole
- This can mean two things
  - Your units of code work with one another
  - Your code works with the underlying system





# Testing Mappers and Reducers Together

---

- Unit tests verify Mappers and Reducers separately
  - Also need to ensure they work *together*
- MRUnit can help here too
  - Even if your job has a `Combiner`



# MiniMRCluster and MiniDFSCluster

---

- MRUnit can test Mapper and Reducer integration
  - But not the *entire job*
- Two “mini-cluster” classes
  - MiniMRCluster simulates MapReduce
  - MiniDFSCluster simulates HDFS
- Let’s look at an example...



# Functional Testing

---

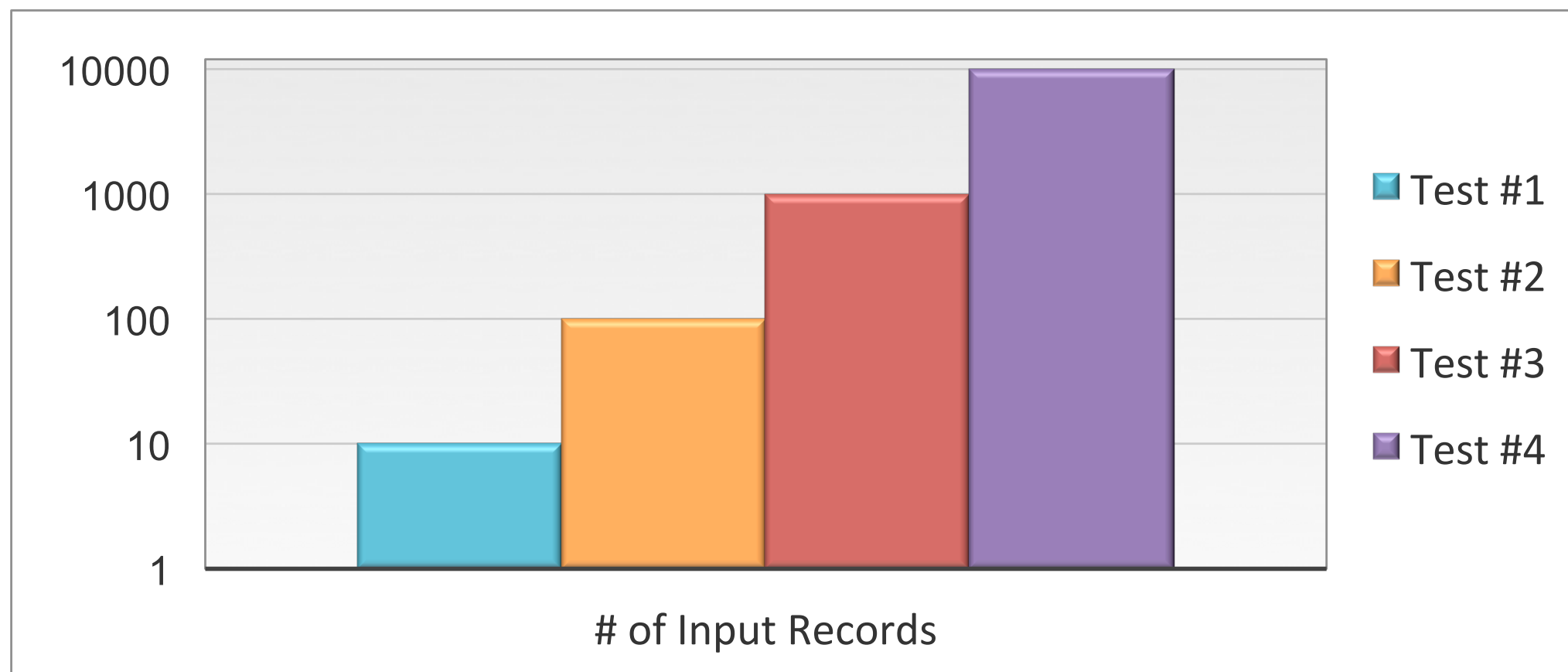
- Unit tests are a great way to verify your code
  - But amount of input data is limited
- Functional tests can help with this
  - But they're ***not*** a substitute for unit testing



# Scaling Up

---

- Start with a small amount of input data
  - Gradually increase it as you gain confidence in your code



# Data Sampling

---

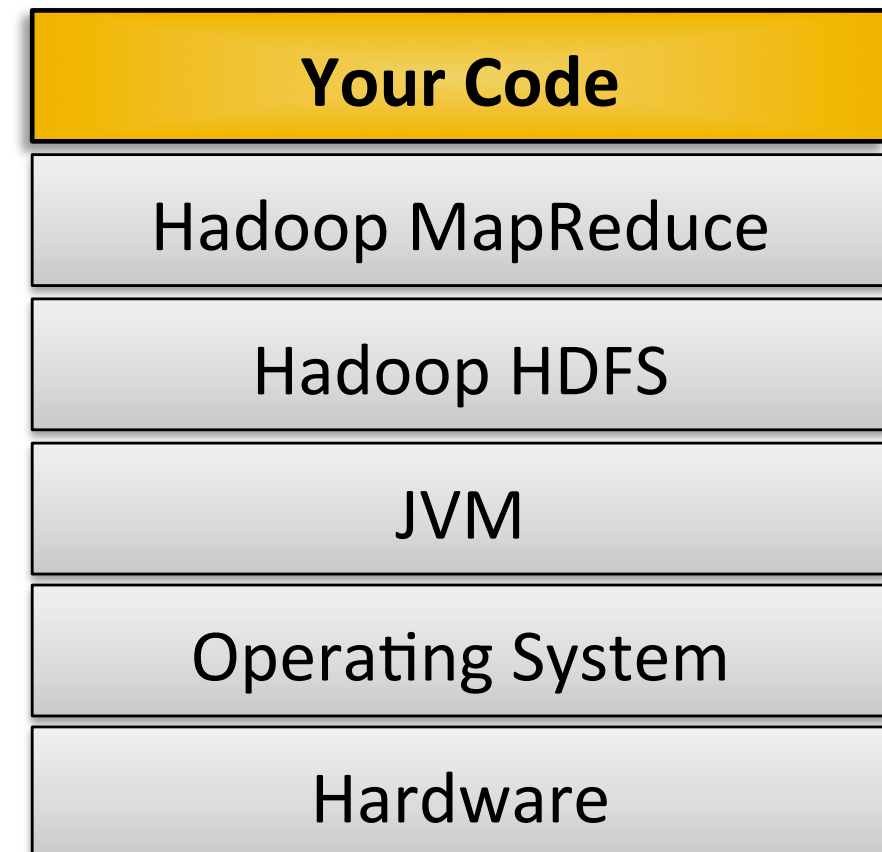
- Two common techniques
  - Random sampling
  - Biased sampling



# Integration Testing the Hadoop Stack

---

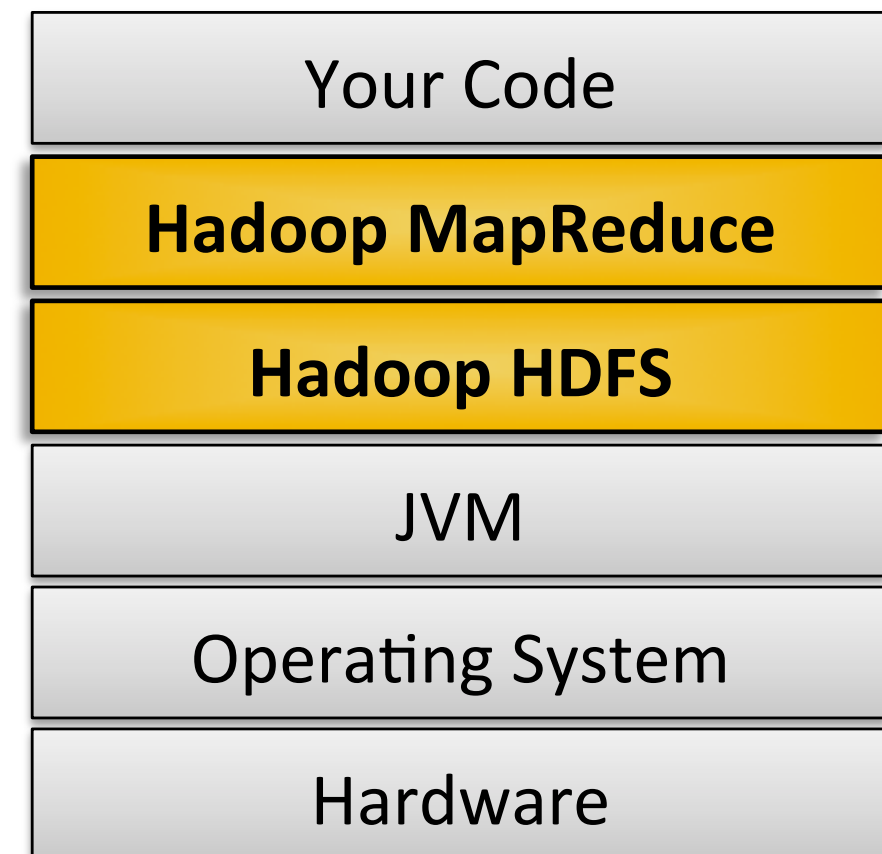
- Your code depends on many other components
  - Your code doesn't function properly unless they do



# Hadoop Integration Testing

---

- Let somebody else do this for you
  - Apache Bigtop
  - Cloudera's Distribution including Apache Hadoop (CDH)

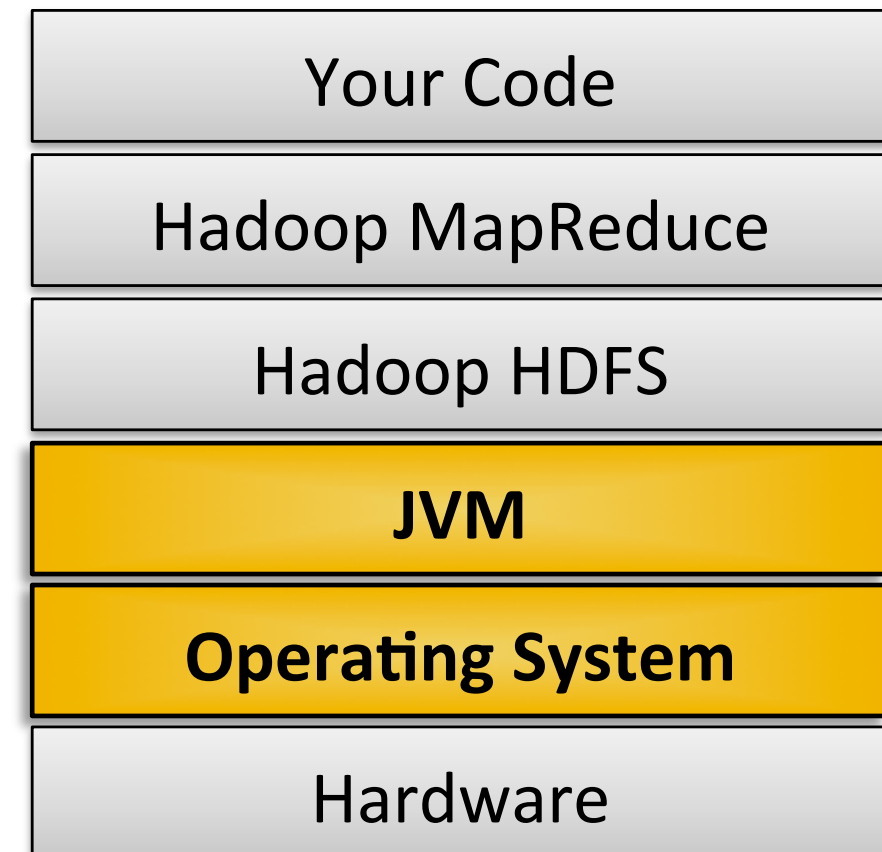




# Follow the Crowd

---

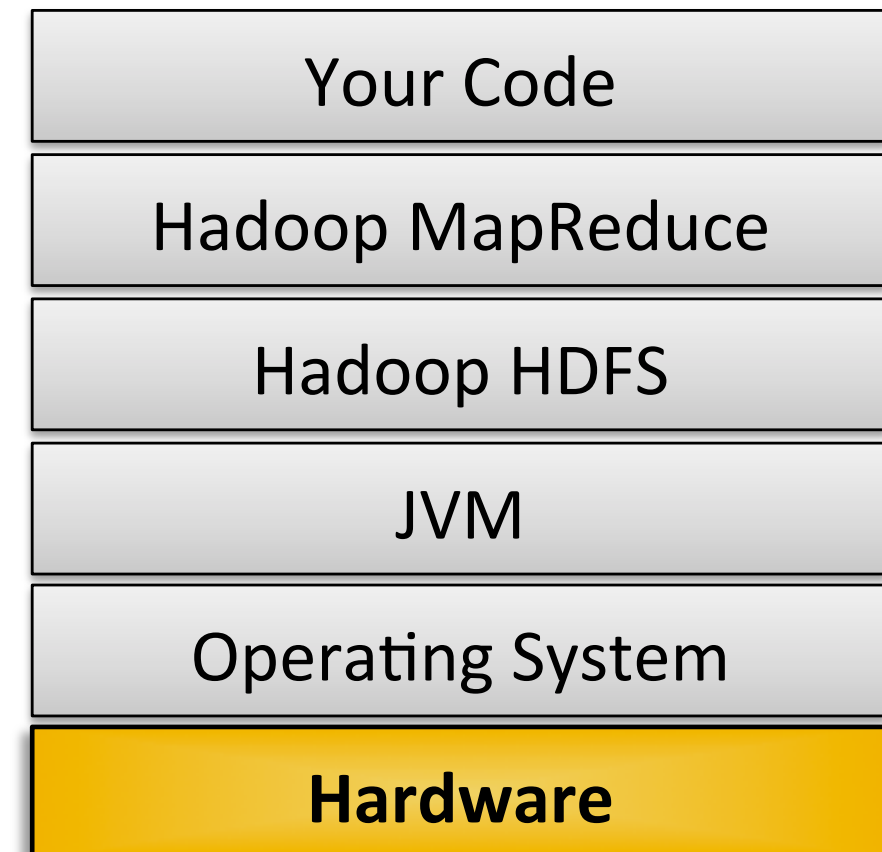
- JVM
  - See “Hadoop Java Versions” on Wiki
  - Current recommendation is 1.6.0\_31
- Operating system
  - Use Linux
- Make dev == production



# Cluster Hardware

---

- Hadoop uses industry standard hardware
  - But quality can make a difference



# Integration Testing with iTest

---

- iTest is a product of Apache Bigtop
  - Can be used for custom integration testing
- iTest is written in Groovy
  - The tests themselves can be in Java or Groovy



# iTest Example

---

```
1  public class RunHadoopJobsTest {
2
3      @Test
4      public void testJobSuccess() {
5          Shell sh = new Shell("/bin/bash -s");
6          sh.exec("hadoop jar /tmp/myjob.jar MyDriver input output");
7          assertEquals(0, sh.getRet());
8      }
9
10     @Test
11     public void testJobFail() {
12         Shell sh = new Shell("/bin/bash -s");
13         sh.exec("hadoop jar /tmp/myjob.jar MyDriver bogus_args");
14         assertEquals(1, sh.getRet());
15     }
16 }
```

Import statements removed for brevity

# Integration Testing Recap

---

- You can do integration testing in several ways
  - MRUnit can test the Mapper/Reducer combination
  - You can use MiniMRCluster and MiniDFSCluster
  - You can use iTest
- Focus on your own code
  - Follow the crowd with the JVM and OS
  - Your development environment should mirror production



# Performance Testing



# Overview of Performance Testing

---

- Verify efficient use of resources
  - Code
  - Cluster





# Resource Utilization

---

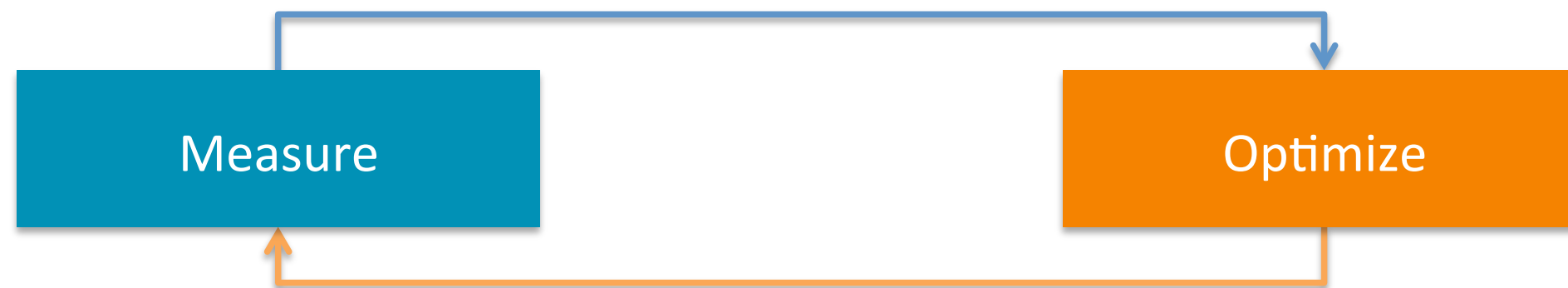
- Optimization is a tradeoff between resources
  - CPU
  - Memory
  - Disk I/O
  - Network I/O
  - Developer time
  - Hardware cost



# Metrics

---

- Two ways to collect performance metrics
  - Code profiling
  - Benchmarks
- Benchmarks can be further subdivided
  - **Micro**
  - **Macro**



# Profiling

```
1  public class ExampleDriver {
2      public static void main(String[] args) throws Exception {
3
4          JobConf conf = new JobConf(ExampleDriver.class);
5          conf.setJobName("My Slow Job");
6
7          FileInputFormat.addInputPath(conf, new Path(args[0]));
8          FileOutputFormat.setOutputPath(conf, new Path(args[1]));
9
10         conf.setProfileEnabled(true);
11         conf.setProfileParams("-agentlib:hprof=cpu=samples,heap=sites," +
12                               "depth=6,force=n,thread=y,verbose=n,file=%s");
13
14         conf.setMapperClass(MySlowMapper.class);
15         conf.setReducerClass(MySlowReducer.class);
16
17         // other driver code follows...
```

# Profiling

---

- One `.profile` file per Map / Reduce task attempt

```
$ ls *.profile
attempt_201206281229_0070_m_000000_0.profile
attempt_201206281229_0070_r_000000_0.profile
attempt_201206281229_0070_m_000001_0.profile
attempt_201206281229_0070_r_000001_0.profile
attempt_201206281229_0070_m_000002_0.profile
attempt_201206281229_0070_r_000002_0.profile
```

# Microbenchmarks

---

- Focus on one aspect of performance
  - Be sure to choose the one that's *really* important for you
- WordCount
  - The simplest benchmark of all



# TeraSort

---

- TeraSort measures time to sort a large set of data

```
$ hadoop jar hadoop-examples.jar \  
    teragen 10000000000 generated
```

```
$ hadoop jar hadoop-examples.jar \  
    terasort generated sorted
```

```
$ hadoop jar hadoop-examples.jar \  
    teravalidate sorted validated
```

Backslashes denote line continuation and are not part of the command



# MRBench

---

- TeraSort tests a *single large job*
  - MRBench can test *many small jobs*

```
$ hadoop jar hadoop-mapreduce-*tests.jar \  
  mrbench \  
  -numRuns 50 \  
  -maps 5 \  
  -reduces 1 \  
  -inputLines 10  
  
# ... Lots of console output ...  
DataLines Maps    Reduces    AvgTime (milliseconds)  
10         5        1         22253
```



# NNBench

---

- Load test for the HDFS NameNode

```
$ hadoop jar hadoop-mapreduce-*tests.jar \
  nnbench \
  -operation create_write \
  -maps 5 \
  -numberOfFiles 5000 \
  -baseDir myoutputdir

# ... Lots of console output ...
TPS: Create/Write/Close: 1830
Avg exec time (ms): Create/Write/Close: 530.833
                  Avg Lat (ms): Create/Write: 270.636
                        Avg Lat (ms): Close: 250.706
```

# TestDFSIO: Write Performance

---

- First, write the data...

```
$ hadoop jar hadoop-mapreduce-*tests.jar \
    TestDFSIO \
    -write \
    -nrFiles 1000 \
    -fileSize 500

# ... Lots of console output ...
Total MBytes processed: 50000.0
    Throughput mb/sec: 4.280953865016388
Average IO rate mb/sec: 6.916503429412842
    IO rate std deviation: 6.55725357815042
    Test exec time sec: 1050.286
```

# TestDFSIO: Read Performance

---

- Next, read the data...

```
$ hadoop jar hadoop-mapreduce-*tests.jar \
    TestDFSIO \
    -read \
    -nrFiles 1000 \
    -fileSize 500

# ... Lots of console output ...
Total MBytes processed: 50000.0
    Throughput mb/sec: 64.89292667099286
Average IO rate mb/sec: 89.74160766601562
    IO rate std deviation: 48.01357337794101
    Test exec time sec: 530.817
```

# TestDFSIO: Deleting Data

---

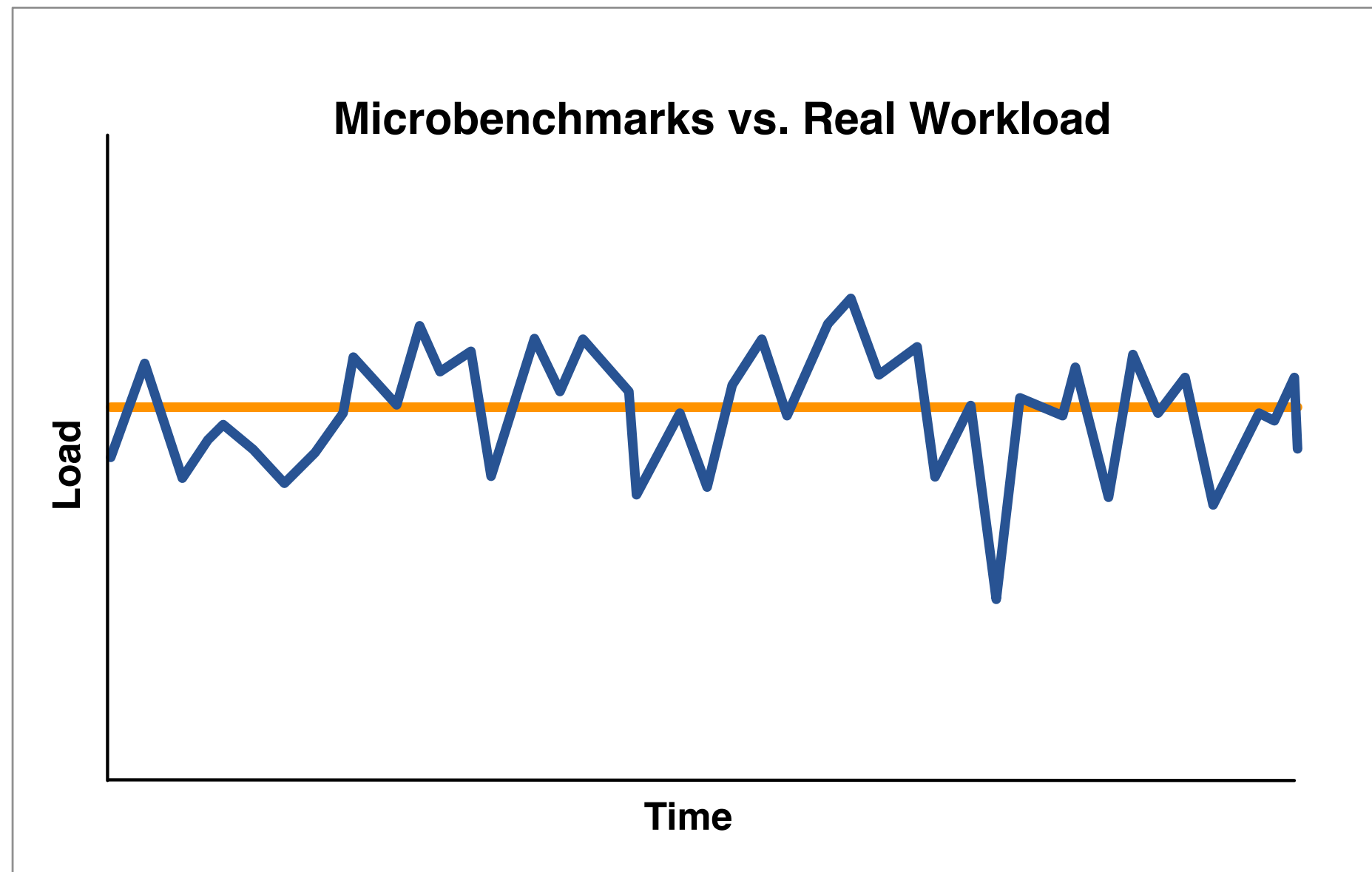
- Finally, remove generated data

```
$ hadoop jar hadoop-mapreduce-*tests.jar \  
    TestDFSIO \  
    -clean
```

# Comprehensive Performance Suites

---

- Simulating actual workload is important



# GridMix3

---

- Generate trace data using Rumen
  - Can also (optionally) scale this data
- Feed trace data to GridMix3



# SWIM

---

- SWIM: **S**tatistical **W**orkload **I**njector for **M**apReduce
  - Scales better than GridMix3
  - Developed in collaboration with UC Berkeley
  - Based on real-life workloads spanning several industries
    - Web
    - Telecommunications
    - Media
    - Retail
  - More info: <https://github.com/SWIMProjectUCB/>





# Performance Testing Recap

---

- Verifies efficient resource usage
  - Code
  - Cluster
- There's no general solution
  - Try to measure and tune what matters *for you*



# Diagnostics



# Overview of Diagnostics

---

- Testing is proactive
  - Diagnostics are reactive
- Many diagnostic tools available, including
  - Web UI
  - Logs
  - Counters
  - Job history



# Web UI

---

- Each Hadoop daemon has its own Web application

Daemon	URL
JobTracker	<code>http://hostname:50030/</code>
TaskTracker	<code>http://hostname:50060/</code>
NameNode	<code>http://hostname:50070/</code>
Secondary NameNode	<code>http://hostname:50090/</code>
DataNode	<code>http://hostname:50075/</code>

# JobTracker Web UI Example

training01 Hadoop Map/Reduce Administration [SP]

http://myjobtracker.example.com:50030/jobtracker.jsp

Google

## training01 Hadoop Map/Reduce Administration

Quick Links

State: RUNNING  
Started: Thu Jun 28 12:29:55 PDT 2012  
Version: 2.0.0-mr1-cdh4.0.0, Unknown  
Compiled: Mon Jun 4 17:31:19 PDT 2012 by jenkins from Unknown  
Identifier: 201206281229

### Cluster Summary (Heap Size is 28.94 MB/560 MB)

Running Map Tasks	Running Reduce Tasks	Total Submissions	Nodes	Occupied Map Slots	Occupied Reduce Slots	Reserved Map Slots	Reserved Reduce Slots	Map Task Capacity	Reduce Task Capacity	Avg. Tasks/Node	Blacklisted Nodes	Excluded Nodes
0	0	171	3	0	0	0	0	48	24	24.00	0	0

### Scheduling Information

Queue Name	State	Scheduling Information
default	running	N/A

Filter (Jobid, Priority, User, Name)   
Example: 'user:smith 3200' will filter by 'smith' only in the user field and '3200' in all fields

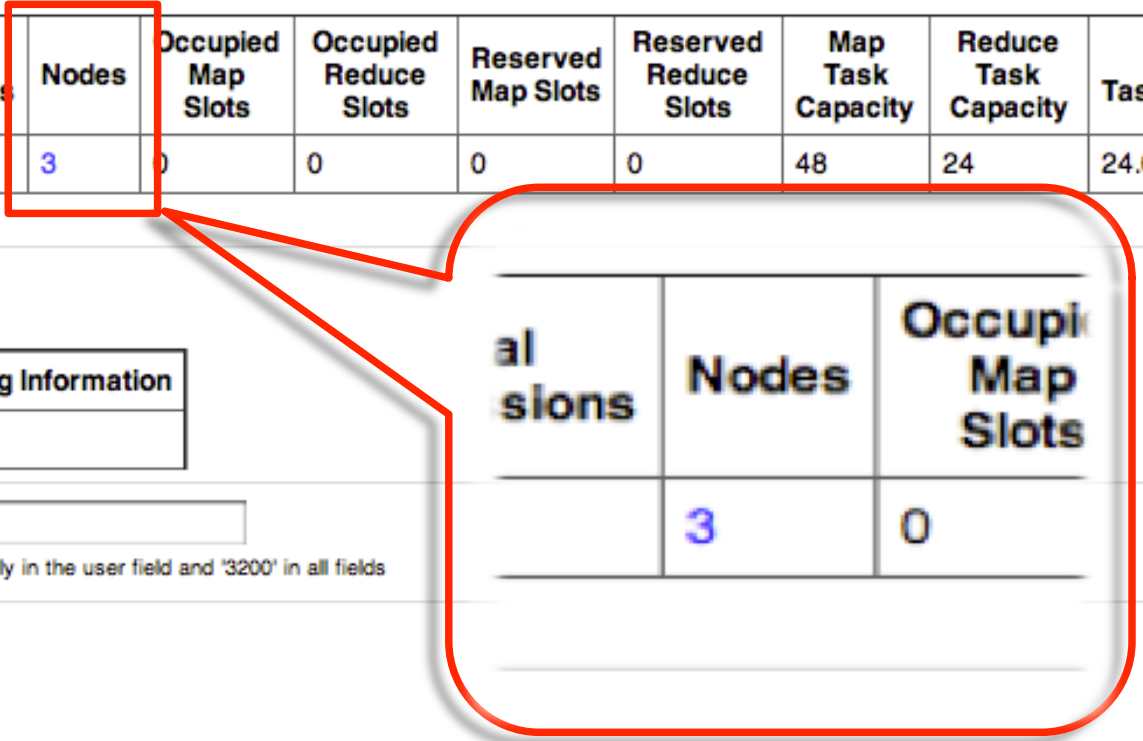
### Running Jobs

none

### Completed Jobs

Jobid	Priority	User	Name	Map %	Map	Maps	Reduce %	Reduce	Reduces	Job Scheduling	Diagn
-------	----------	------	------	-------	-----	------	----------	--------	---------	----------------	-------

Display a menu



# TaskTracker Web UI Example



Version: 2.0.0-mr1-cdh4.0.0, Unknown

Compiled: Mon Jun 4 17:31:19 PDT 2012 by jenkins from Unknown

## Running tasks

Task Attempts	Status	Progress	Errors
attempt_201206281229_0186_r_000008_0	RUNNING	0.00%	
attempt_201206281229_0186_r_000009_0	RUNNING	0.00%	
attempt_201206281229_0186_r_000010_0	RUNNING	0.00%	
attempt_201206281229_0186_r_000011_0	RUNNING	0.00%	

## Non-Running Tasks

Task Attempts	Status
attempt_201206281229_0186_m_000001_0	SUCCEEDED

## Tasks from Running Jobs

Task Attempts	Status	Progress	Errors
attempt_201206281229_0186_m_000001_0	SUCCEEDED	100.00%	
attempt_201206281229_0186_r_000010_0	RUNNING	0.00%	

This is just an excerpt of a larger page



# JobTracker Job History Page

---

User: twheeler

Job Name: PiEstimator

Job File: [hdfs://training01.mtv.cloudera.com:8020/user/twheeler/.staging/job\\_201206281229\\_0182/job.xml](hdfs://training01.mtv.cloudera.com:8020/user/twheeler/.staging/job_201206281229_0182/job.xml)

Submit Host: training01.mtv.cloudera.com

Submit Host Address: 172.20.63.11

Job-ACLs: All users are allowed

Job Setup: [Successful](#)

Status: Succeeded

Started at: Wed Oct 10 20:28:57 PDT 2012

Finished at: Wed Oct 10 20:29:22 PDT 2012

Finished in: 25sec

Job Cleanup: [Successful](#)

Kind	% Complete	Num Tasks	Pending	Running	Complete	Killed	Failed/Killed Task Attempts
map	<div><div></div></div> 100.00%	8	0	0	8	0	0 / 0
reduce	<div><div></div></div> 100.00%	1	0	0	1	0	0 / 0



# Task Attempt List

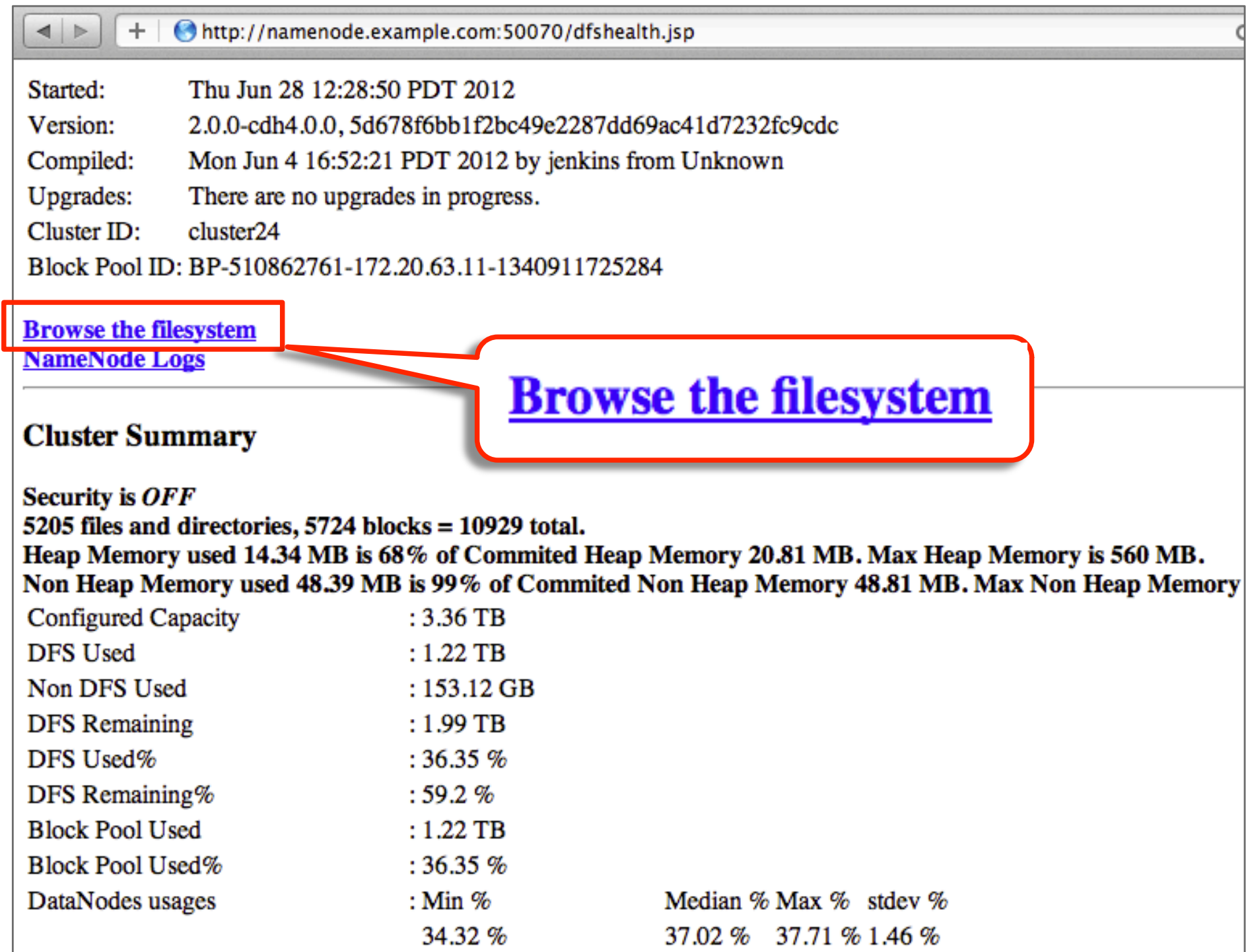
## All Tasks

Task	Complete	Start Time	Finish Time
<a href="#">task_201206281229_0182_m_000000</a>	100.00% <div></div>	10-Oct-2012 20:29:03	10-Oct-2012 20:29:09 (6sec)
<a href="#">task_201206281229_0182_m_000001</a>	100.00% <div></div>	10-Oct-2012 20:29:03	10-Oct-2012 20:29:09 (5sec)
<a href="#">task_201206281229_0182_m_000002</a>	100.00% <div></div>	10-Oct-2012 20:29:03	10-Oct-2012 20:29:09 (5sec)
<a href="#">task_201206281229_0182_m_000003</a>	100.00% <div></div>	10-Oct-2012 20:29:03	10-Oct-2012 20:29:08 (4sec)
<a href="#">task_201206281229_0182_m_000004</a>	100.00% <div></div>	10-Oct-2012 20:29:03	10-Oct-2012 20:29:09 (6sec)
<a href="#">task_201206281229_0182_m_000005</a>	100.00% <div></div>	10-Oct-2012 20:29:03	10-Oct-2012 20:29:09 (6sec)
<a href="#">task_201206281229_0182_m_000006</a>	100.00% <div></div>	10-Oct-2012 20:29:03	10-Oct-2012 20:29:10 (7sec)
<a href="#">task_201206281229_0182_m_000007</a>	100.00% <div></div>	10-Oct-2012 20:29:03	10-Oct-2012 20:29:10 (7sec)

Some columns removed to better fit the screen



# NameNode Web UI



Started: Thu Jun 28 12:28:50 PDT 2012  
Version: 2.0.0-cdh4.0.0, 5d678f6bb1f2bc49e2287dd69ac41d7232fc9cdc  
Compiled: Mon Jun 4 16:52:21 PDT 2012 by jenkins from Unknown  
Upgrades: There are no upgrades in progress.  
Cluster ID: cluster24  
Block Pool ID: BP-510862761-172.20.63.11-1340911725284

[Browse the filesystem](#)  
[NameNode Logs](#)

**Browse the filesystem**

### Cluster Summary

Security is *OFF*  
5205 files and directories, 5724 blocks = 10929 total.  
Heap Memory used 14.34 MB is 68% of Committed Heap Memory 20.81 MB. Max Heap Memory is 560 MB.  
Non Heap Memory used 48.39 MB is 99% of Committed Non Heap Memory 48.81 MB. Max Non Heap Memory

Configured Capacity	: 3.36 TB		
DFS Used	: 1.22 TB		
Non DFS Used	: 153.12 GB		
DFS Remaining	: 1.99 TB		
DFS Used%	: 36.35 %		
DFS Remaining%	: 59.2 %		
Block Pool Used	: 1.22 TB		
Block Pool Used%	: 36.35 %		
DataNodes usages	: Min %	Median %	Max % stdev %
	34.32 %	37.02 %	37.71 % 1.46 %

# Browsing HDFS

## Contents of directory [/](#)

Goto :

Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
<a href="#">hbase</a>	dir				2012-06-28 12:29	rwxr-xr-x	hbase	hbase
<a href="#">tmp</a>	dir				2012-07-19 07:40	rwxrwxrwt	hdfs	hdfs
<a href="#">user</a>	dir				2012-09-14 04:18	rwxr-xr-x	hdfs	supergroup

[Go back to DFS home](#)



# Logs

---

- Most important logs for MapReduce developers
  - stdout
  - stderr
  - syslog
- Demo: How to debug error using Web UI and logs





# Custom Logging

- Use Log4J for logging in your own code

```
1  import org.apache.log4j.Logger;
2  // other imports omitted for brevity...
3
4  public class MyMap extends MapReduceBase implements
5      Mapper<LongWritable, Text, Text, IntWritable> {
6
7      private static final Logger logger = Logger.getLogger(MyMap.class);
8
9      public void map(LongWritable key, Text value,
10         OutputCollector<Text, IntWritable> output, Reporter reporter)
11         throws IOException {
12
13         logger.info("Some important message goes here");
```

# Log Configuration and Control

- Excerpt of a simple `log4j.properties` file
  - Line 8 sets level for logs generated by our mapper
  - Line 9 sets level for an entire Hadoop package

```
1 log4j.rootLogger=INFO, stdout
2 log4j.appender.stdout=org.apache.log4j.ConsoleAppender
3 log4j.appender.stdout.Target=System.out
4 log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
5 log4j.appender.stdout.layout.ConversionPattern=\
6     %d{ABSOLUTE} %5p %c{1}:%L - %m%n
7
8 log4j.logger.com.example.foo.MyMapper=DEBUG
9 log4j.logger.org.apache.hadoop.mapred=ERROR
10
```

Date priority (level), category (class):line – message + newline

# Beware Logging Performance

---

- Logging *can* cause performance problems
  - How many problems do you spot here?

```
1 // deep inside our mapper...
2 String[] fields = line.split(" ");
3
4 logger.info("Read line: " + line);
5 logger.info("Line has " + fields.length + " fields");
6
```

- Let's see how to improve this...





# Logging Wisely

---

- This revision is much improved

```
1 // Only print bad records
2 if (fields.length != 4) {
3     // add helpful info about where the bad records reside
4     FileSplit fs = (FileSplit)reporter.getInputSplit();
5     String path = fs.getPath().toString();
6
7     // log at a more appropriate level
8     logger.warn("Read bad line: " + line + " from: " + path);
9 }
10
11 // avoid unnecessary string concatenation
12 if (logger.isDebugEnabled()) {
13     // log at a more appropriate level
14     logger.debug("Line has " + fields.length + " fields");
15 }
16
```

# Counters

## Counters for attempt\_201206281229\_0067\_m\_000001\_0

File System Counters		
	FILE: Number of bytes read	0
	FILE: Number of bytes written	68,202
	FILE: Number of read operations	0
	FILE: Number of large read operations	0
	FILE: Number of write operations	0
	HDFS: Number of bytes read	63,501
	HDFS: Number of bytes written	0
	HDFS: Number of read operations	2
	HDFS: Number of large read operations	0
	HDFS: Number of write operations	0
Map-Reduce Framework		
	Map input records	1,797
	Map output records	1,797
	Map output bytes	14,376
	Input split bytes	139
	Combine input records	0
	Combine output records	0
	Spilled Records	1,797
	CPU time spent (ms)	3,130
	Physical memory (bytes) snapshot	336,834,560
	Virtual memory (bytes) snapshot	1,543,569,408
	Total committed heap usage (bytes)	503,971,840
org.apache.hadoop.mapreduce.lib.input.FileInputFormatCounter		
	BYTES_READ	63,352

Map input records 1,797

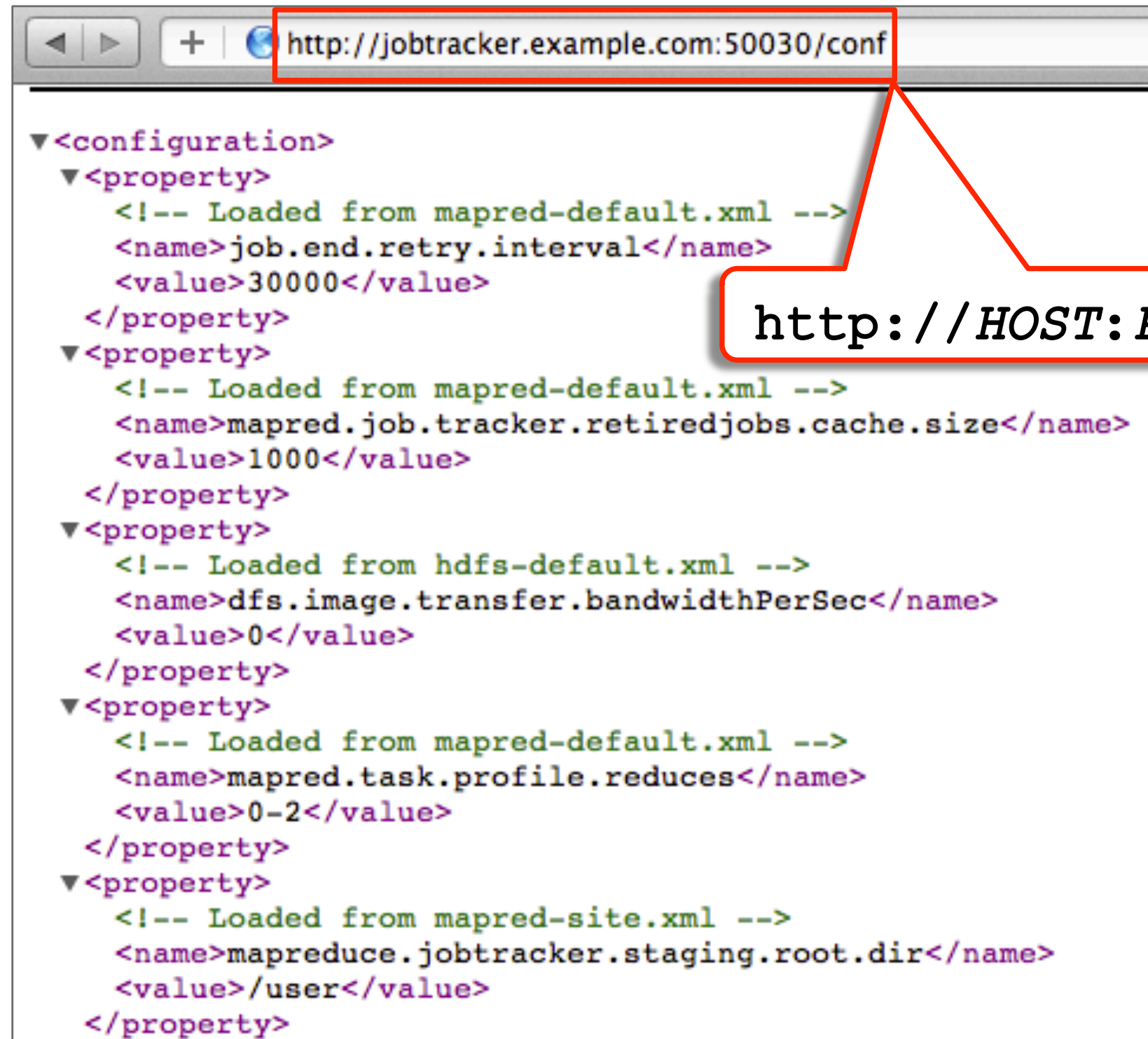
# Custom Record Counters

---

- Custom counters are also useful for diagnostics
  - Let me demonstrate...



# Viewing Current Configuration



```
▼<configuration>
  ▼<property>
    <!-- Loaded from mapred-default.xml -->
    <name>job.end.retry.interval</name>
    <value>30000</value>
  </property>
  ▼<property>
    <!-- Loaded from mapred-default.xml -->
    <name>mapred.job.tracker.retiredjobs.cache.size</name>
    <value>1000</value>
  </property>
  ▼<property>
    <!-- Loaded from hdfs-default.xml -->
    <name>dfs.image.transfer.bandwidthPerSec</name>
    <value>0</value>
  </property>
  ▼<property>
    <!-- Loaded from mapred-default.xml -->
    <name>mapred.task.profile.reduces</name>
    <value>0-2</value>
  </property>
  ▼<property>
    <!-- Loaded from mapred-site.xml -->
    <name>mapreduce.jobtracker.staging.root.dir</name>
    <value>/user</value>
  </property>
```

`http://HOST:PORT/conf`



# Job Properties

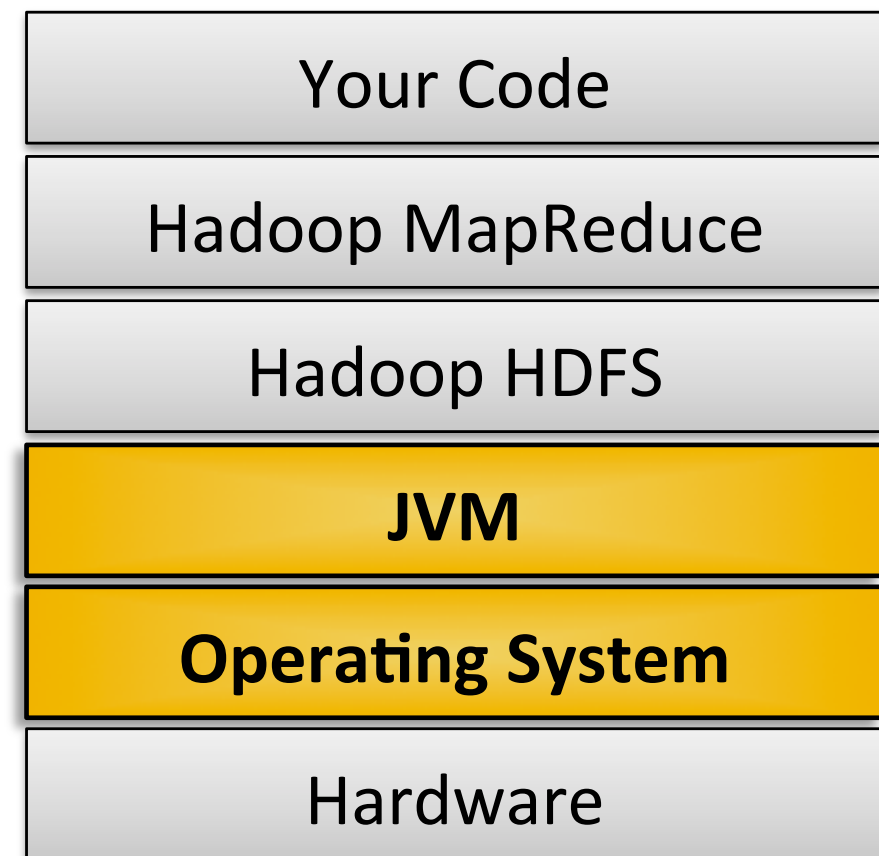
## Job Configuration: JobId - job\_201206281229\_0067

name	
job.end.retry.interval	30000
mapred.job.tracker.retiredjobs.cache.size	1000
mapred.queue.default.acl-administer-jobs	*
dfs.image.transfer.bandwidthPerSec	0
mapred.task.profile.reduces	0-2
mapreduce.jobtracker.staging.root.dir	\${hadoop.tmp.dir}/mapred/staging
mapred.job.reuse.jvm.num.tasks	1
dfs.block.access.token.lifetime	600
fs.AbstractFileSystem.file.impl	org.apache.hadoop.fs.local.LocalFs
mapred.reduce.tasks.speculative.execution	false
mapred.job.name	Stock analyzer
hadoop.http.authentication.kerberos.keytab	\${user.home}/hadoop.keytab
io.seqfile.sorter.recordlimit	1000000

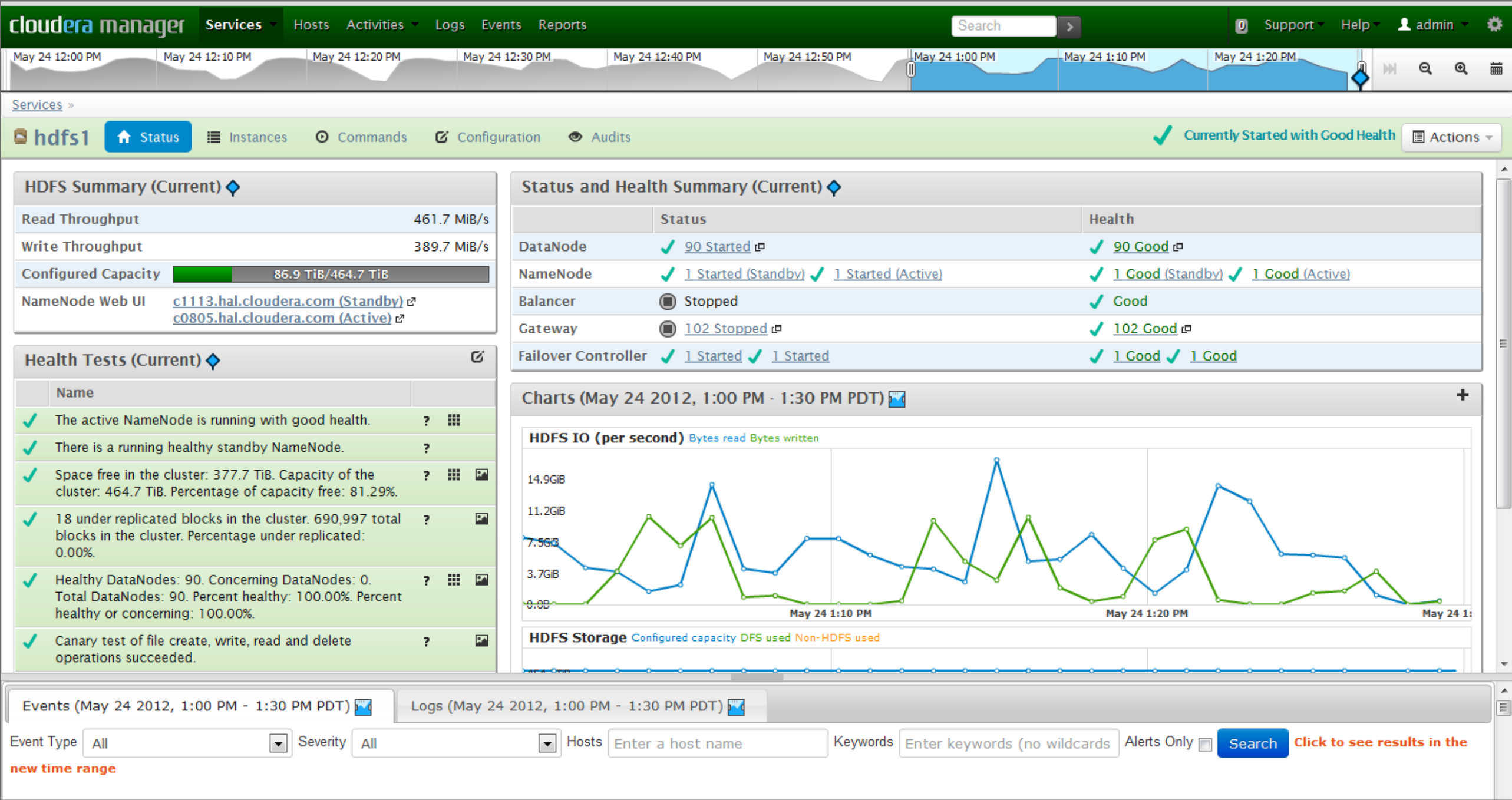
# Other Tools

---

- Generic diagnostic tools
  - `jps`
  - `top`
  - `iostat`
  - `vmstat`
  - `netstat`
  - `ulimit`
  - `sysctl`



# Cloudera Manager





# Diagnostics Recap

---

- Hadoop provides a lot of diagnostic information
  - Most of this is available via the Web UI



# Recap

---

- There are many types of testing
  - Unit
  - Integration
  - Performance
- Diagnostics can help catch problems that slip by



# Tool Recap

---

- Unit Testing
  - JUnit
  - MRUnit
- Integration Testing
  - MRUnit
  - MiniDFSCluster / MiniMRCluster
  - iTest



# Tool Recap (cont'd)

---

- Performance Testing
  - Micro:
    - TeraSort
    - MRBench
    - NNBench
    - TestDFSIO
  - Macro:
    - GridMix3
    - SWIM



# Tool Recap (cont'd)

---

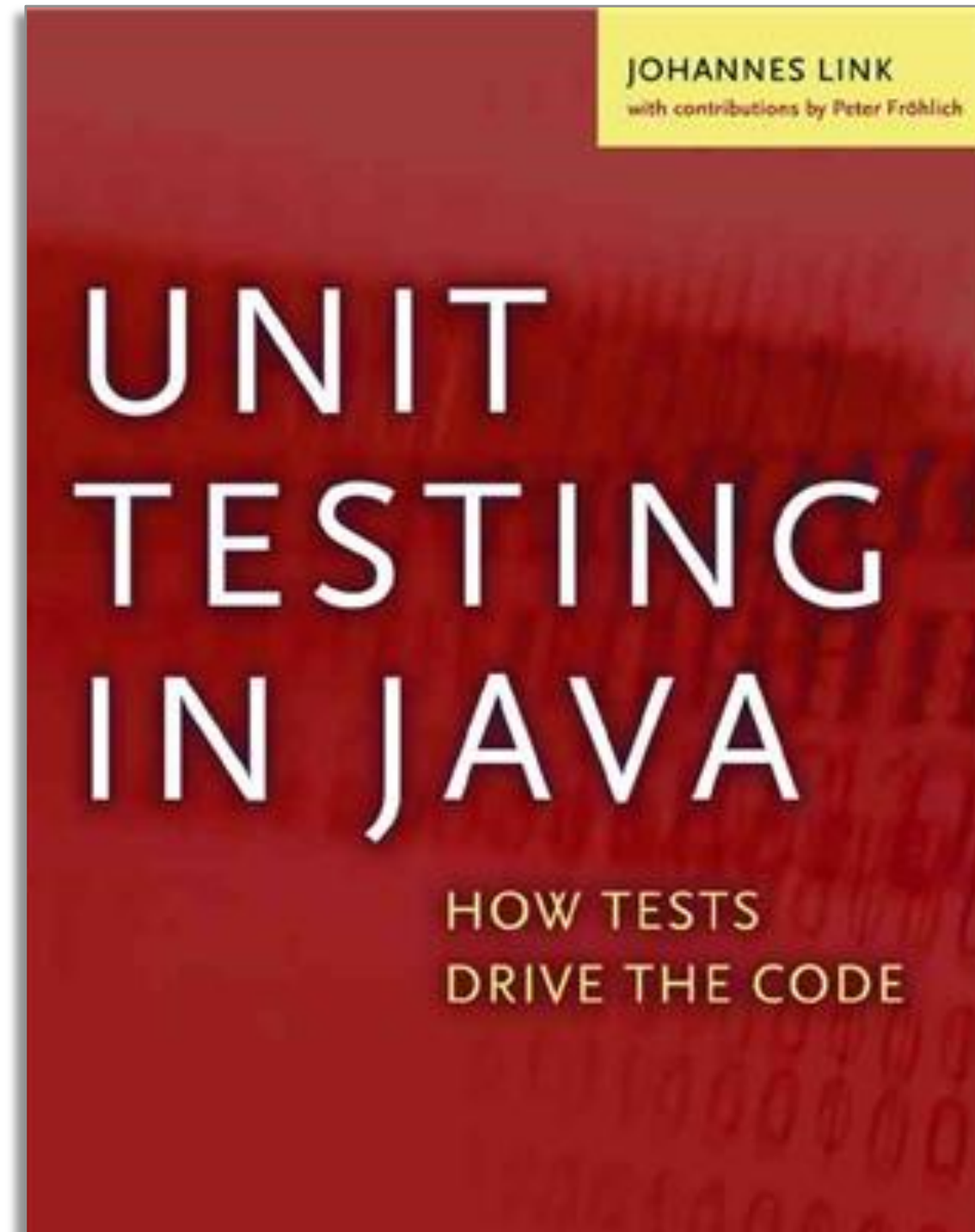
- Diagnostics
  - Web UI
  - Logs
  - Counters
  - Job History



# Resources for More Info

---

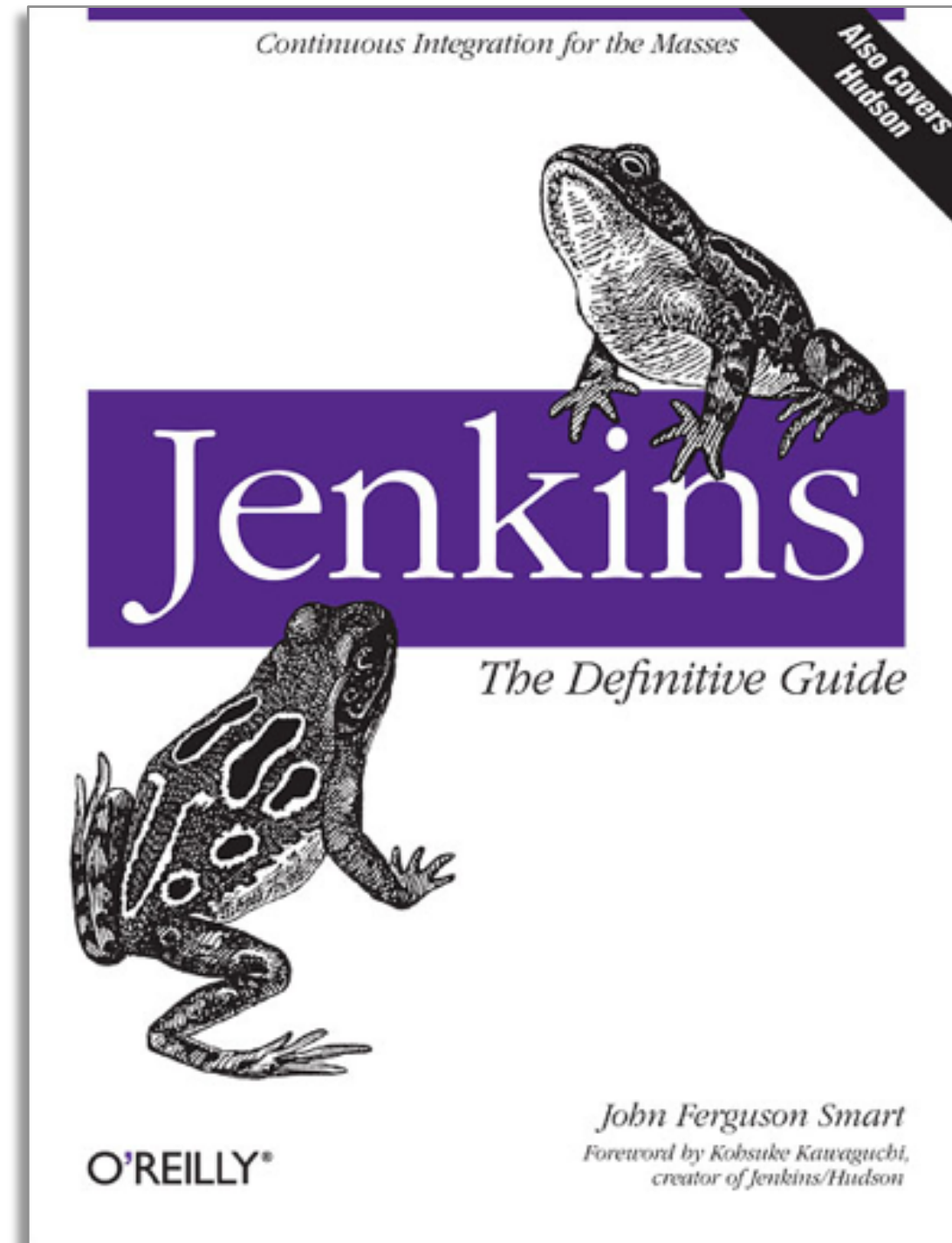
- Johannes Link
- ISBN: 1-55860-868-0





# Resources for More Info

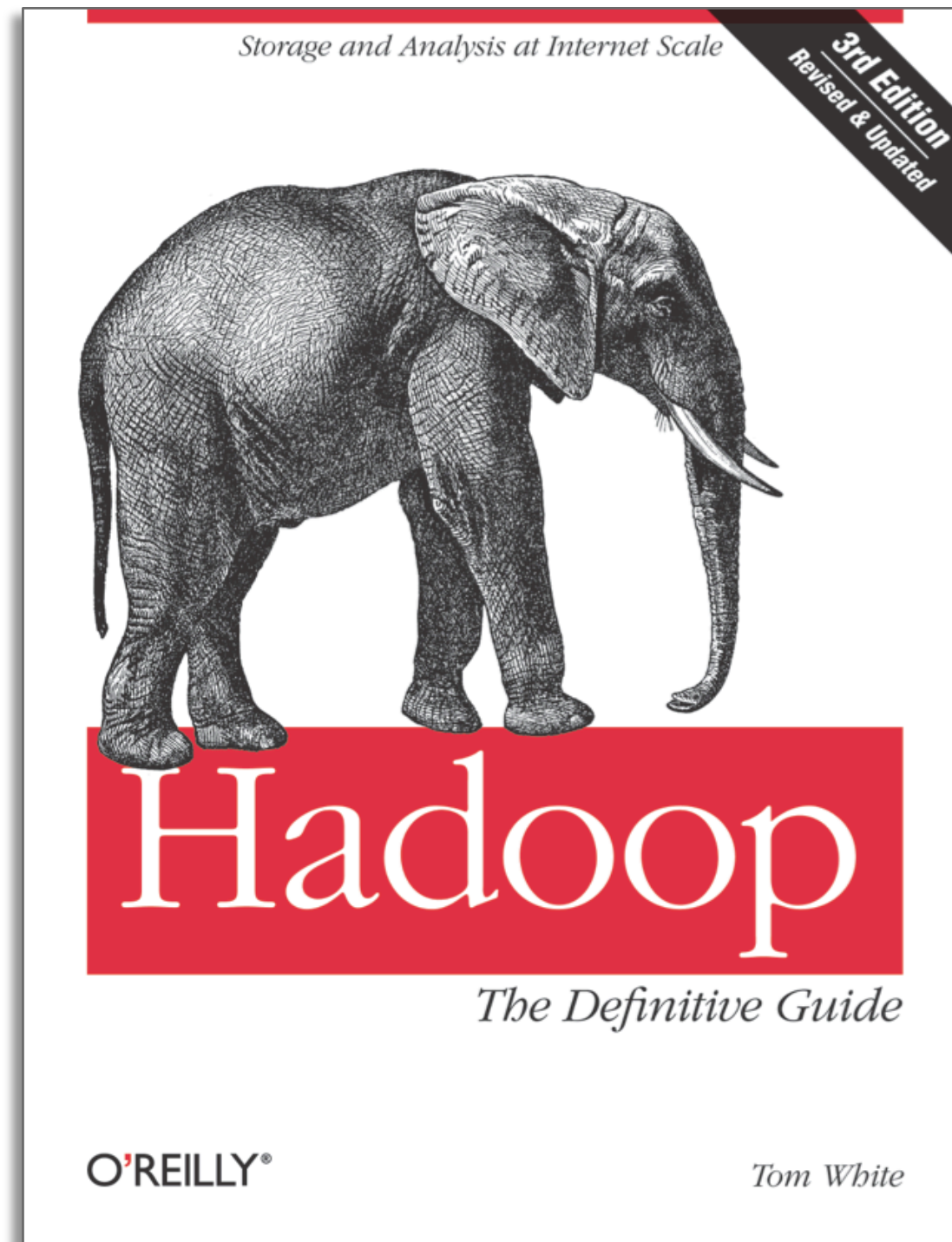
- John Ferguson Smart
- ISBN: 1-449-30535-0





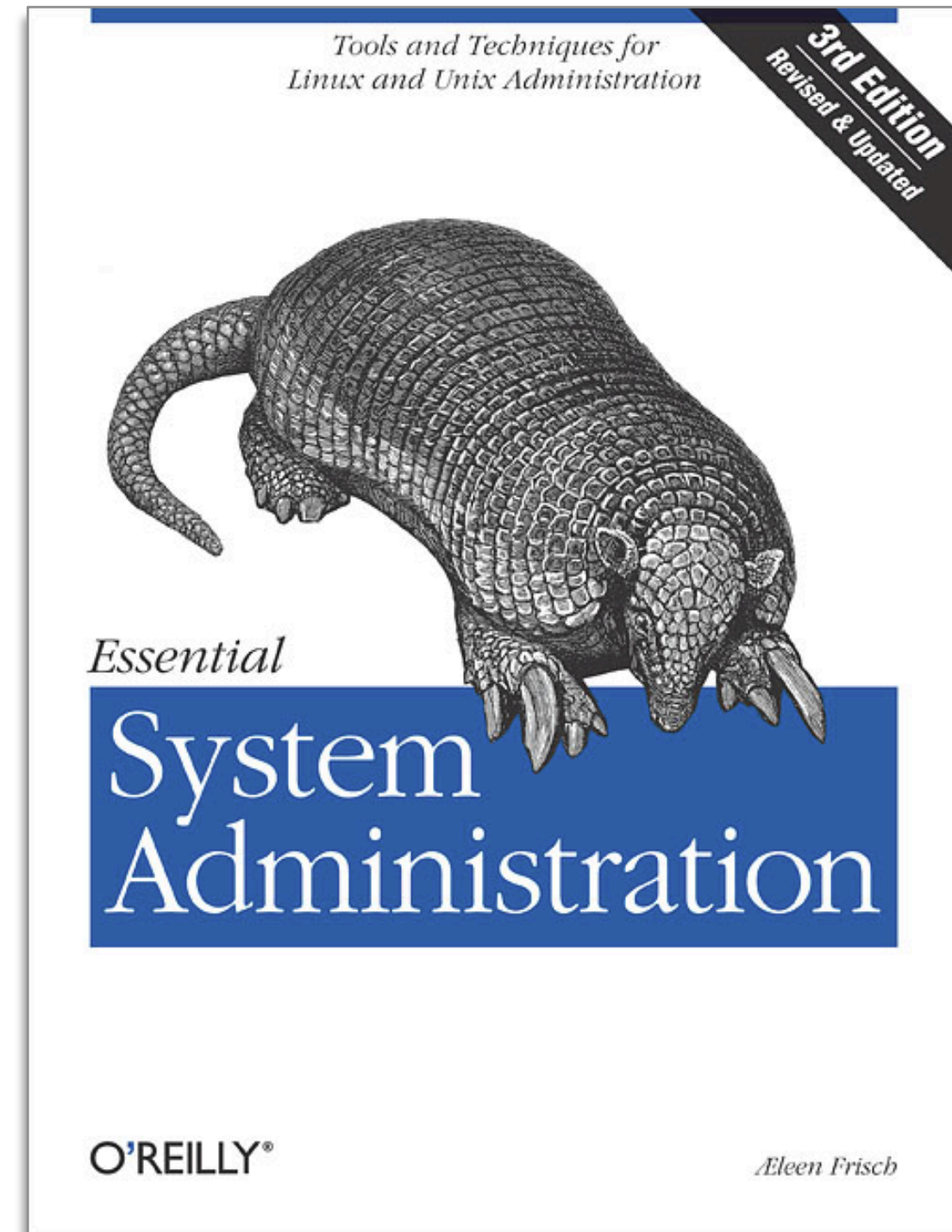
# Resources for More Info (cont'd)

- Tom White
- ISBN: 1-449-31152-0



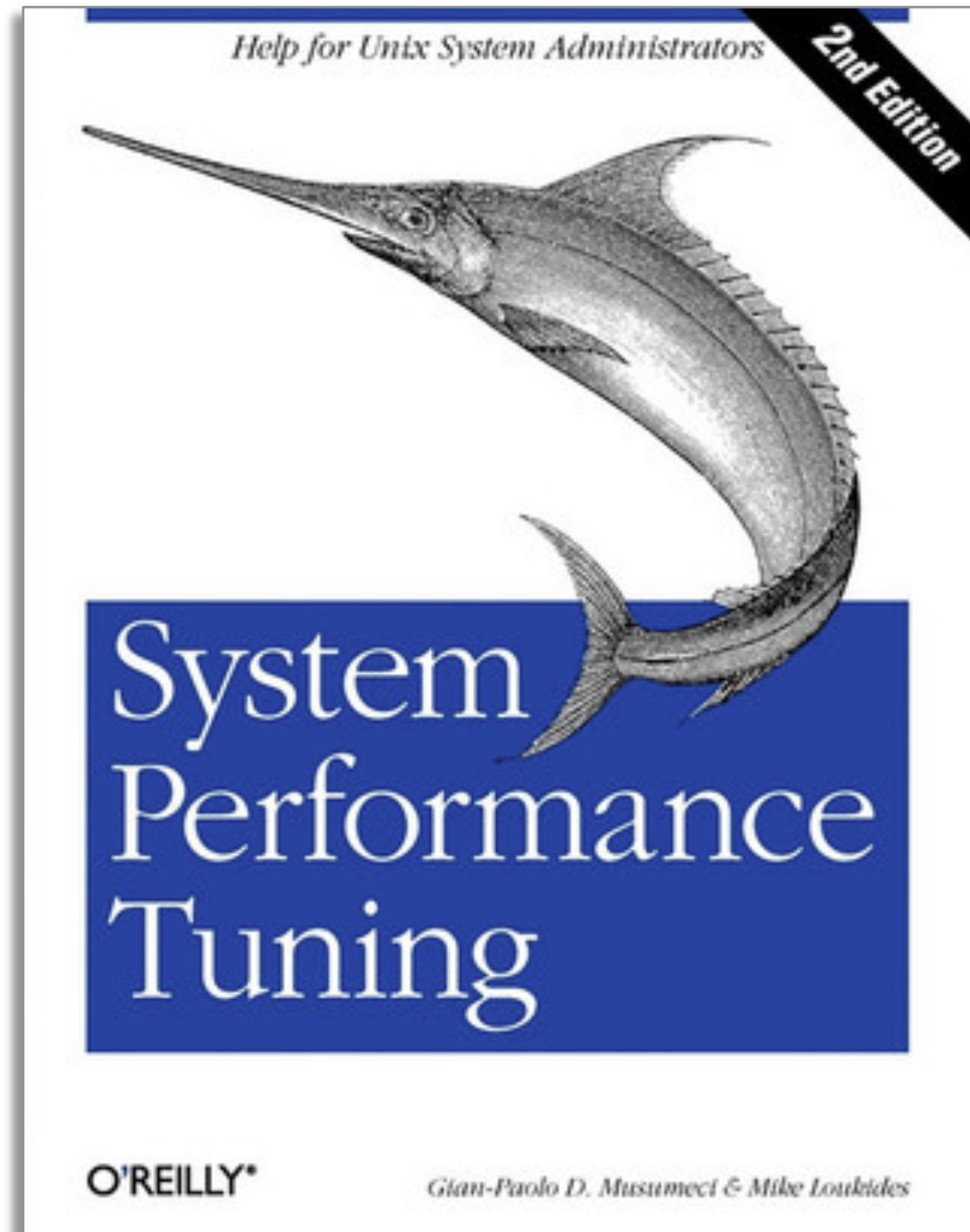
# Resources for More Info (cont'd)

- Æleen Frisch
- ISBN: 0-596-00343-9



# Resources for More Info (cont'd)

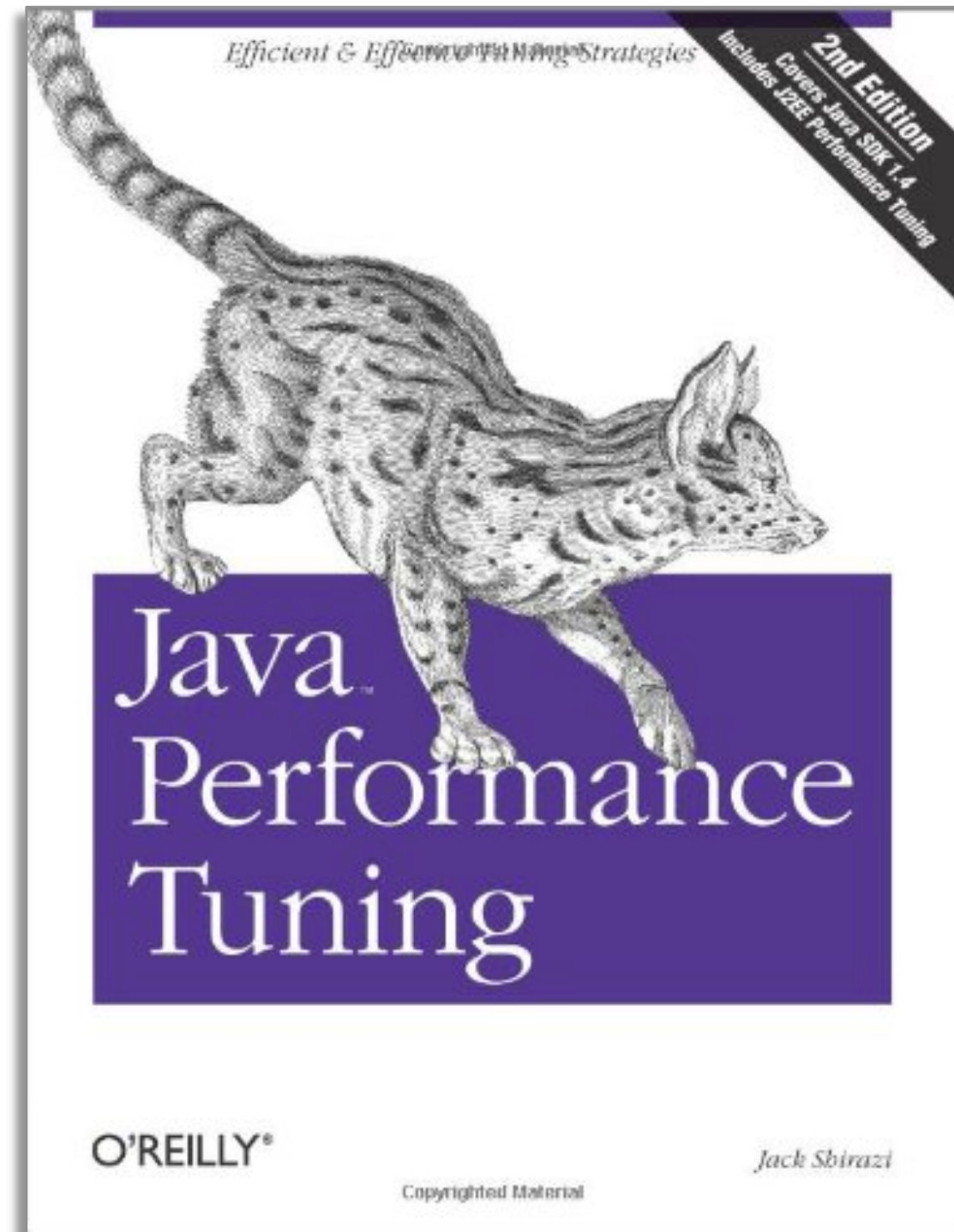
- Gian-Paolo Musumeci and Mike Loukides
- ISBN: 0-596-00284-X





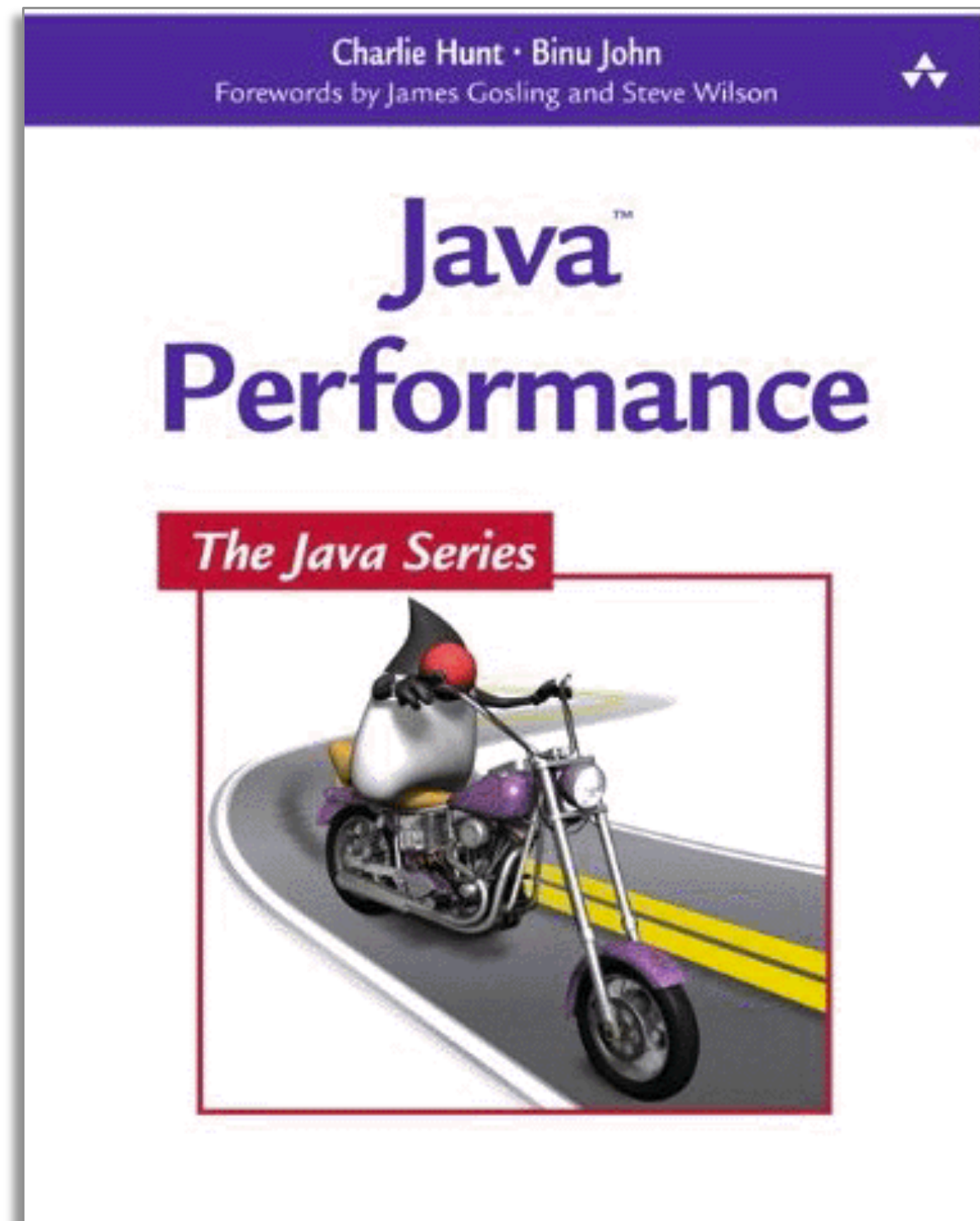
# Resources for More Info (cont'd)

- Jack Shirazi
- ISBN: 0-596-00377-3



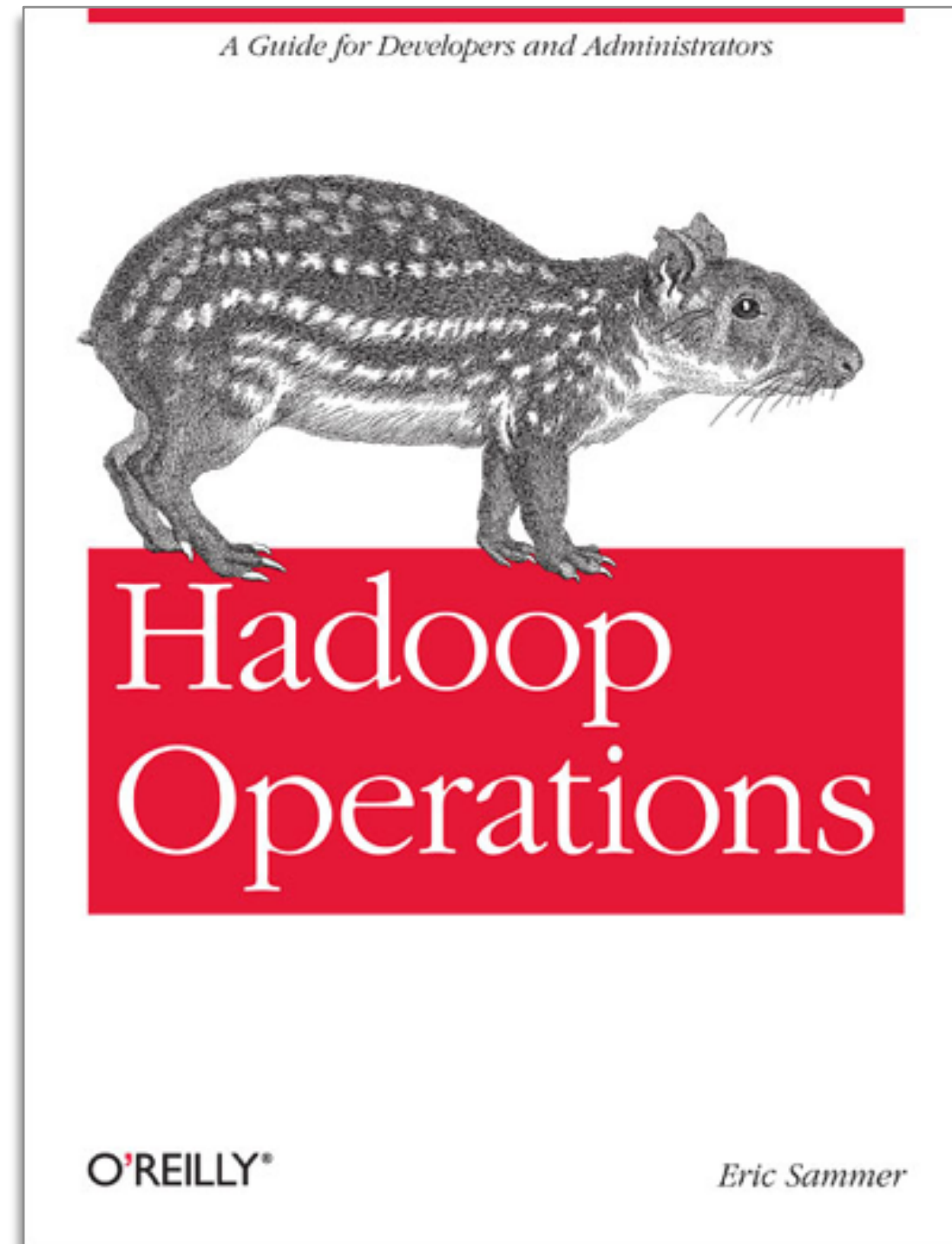
# Resources for More Info (cont'd)

- Charlie Hunt and Binu John
- ISBN: 0-137-14252-8



# Resources for More Info (cont'd)

- Eric Sammer
- ISBN: 1-449-32705-2



# Conclusion

---

- Testing helps develop trust in a system
  - To behave as you expect it to





# Thank You!

---

Tom Wheeler, Curriculum Developer, Cloudera