Testing Hadoop Applications

Tom Wheeler
About The Presenter...

Tom Wheeler
Software Engineer, etc.
Greater St. Louis Area | Information Technology and Services

Current: **Senior 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)
- But I’ll assume no particular knowledge of testing
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.”
Definitions

- **Unit testing**
  - Verifies function of each “unit” in isolation
- **Integration testing**
  - Verifies that the system works as a whole
- **Performance testing**
  - Verifies that code makes efficient use of resources
- **Diagnostics**
  - More about forensics than testing
Why is Testing Important?

Cost of Fixing Code Defects

- Design
- Implementation
- Production

Expense vs. Time
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

public class CalculatorTest {

    private Calculator calc;

    @Before
    public void setUp() {
        calc = new Calculator();
    }

    @Test
    public void testAdd() {
        assertEquals(8, calc.add(5, 3));
    }
}

public class Calculator {

    public int add(int first, int second) {
        return first + second;
    }
}

Our class

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

<table>
<thead>
<tr>
<th>Date/Time</th>
<th>Level</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>2012-09-06 22:16:49.391 CDT</td>
<td>INFO</td>
<td>&quot;This can wait&quot;</td>
</tr>
<tr>
<td>2012-09-06 22:16:49.392 CDT</td>
<td>INFO</td>
<td>&quot;Blah blah&quot;</td>
</tr>
<tr>
<td>2012-09-06 22:16:49.394 CDT</td>
<td>WARN</td>
<td>&quot;Hmmm...&quot;</td>
</tr>
<tr>
<td>2012-09-06 22:16:49.395 CDT</td>
<td>INFO</td>
<td>&quot;More blather&quot;</td>
</tr>
<tr>
<td>2012-09-06 22:16:49.397 CDT</td>
<td>WARN</td>
<td>&quot;Hey there&quot;</td>
</tr>
<tr>
<td>2012-09-06 22:16:49.398 CDT</td>
<td>INFO</td>
<td>&quot;Spewing data&quot;</td>
</tr>
<tr>
<td>2012-09-06 22:16:49.399 CDT</td>
<td>ERROR</td>
<td>&quot;Oh boy!&quot;</td>
</tr>
</tbody>
</table>
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)
Reducer Input

- Hadoop sorts and groups the keys

<table>
<thead>
<tr>
<th>Level</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERROR</td>
<td>[1]</td>
</tr>
<tr>
<td>INFO</td>
<td>[1, 1, 1, 1]</td>
</tr>
<tr>
<td>WARN</td>
<td>[1, 1]</td>
</tr>
</tbody>
</table>
Reducer Output

- The final output is a per-level summary

<table>
<thead>
<tr>
<th>Level</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERROR</td>
<td>1</td>
</tr>
<tr>
<td>INFO</td>
<td>4</td>
</tr>
<tr>
<td>WARN</td>
<td>2</td>
</tr>
</tbody>
</table>
Data Flow for Entire MapReduce Job

Map input:
- 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!"

Reduce input:
- ERROR [1]
- INFO [1, 1, 1, 1]
- WARN [1, 1]

Reduce output:
- ERROR 1
- INFO 4
- WARN 2

Map output:
- INFO 1
- INFO 1
- WARN 1
- INFO 1
- WARN 1
- ERROR 1

Shuffle and Sort:

1. Map input
2. Shuffle and Sort
3. Reduce input
4. Reduce output
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?
• Refactor to decouple business logic from Hadoop API
Unit Testing and External Dependencies

- We can validate core business logic with JUnit
  - But our 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
Limitations of MRUnit

- There are a few things MRUnit can’t test (yet)
  - Multiple lines of input for a Mapper
  - Jobs that use DistributedCache
  - Partitioners
  - Hadoop Streaming
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
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 see an example...
Integration Testing the Hadoop Stack

• Your code depends on many other components
  • Your code doesn’t function properly unless they do

[Diagram showing dependencies of your code on Hadoop MapReduce, Hadoop HDFS, JVM, Operating System, and Hardware]
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
public class RunHadoopJobsTest {

   @Test
   public void testJobSuccess() {
      Shell sh = new Shell("/bin/bash -s");
      sh.exec("hadoop jar /tmp/myjob.jar MyDriver input output");
      assertEquals(0, sh.getRet());
   }

   @Test
   public void testJobFail() {
      Shell sh = new Shell("/bin/bash -s");
      sh.exec("hadoop jar /tmp/myjob.jar MyDriver bogus_args");
      assertEquals(1, sh.getRet());
   }
}
Performance Testing
Resource Utilization

- Optimization is a tradeoff between resources
  - CPU
  - Memory
  - Disk I/O
  - Network I/O
  - Developer time
  - Hardware cost
public class ExampleDriver {
    public static void main(String[] args) throws Exception {

        JobConf conf = new JobConf(ExampleDriver.class);
        conf.setJobName("My Slow Job");

        FileInputFormat.addInputPath(conf, new Path(args[0]));
        FileOutputFormat.setOutputPath(conf, new Path(args[1]));

        conf.setProfileEnabled(true);
        conf.setProfileParams("-agentlib:hprof=cpu=samples,heap=sites," +
                            "depth=6,force=n,thread=y,verbose=n,file=%s");

        conf.setMapperClass(MySlowMapper.class);
        conf.setReducerClass(MySlowReducer.class);

        // 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
```
Diagnostics
Overview of Diagnostics

• Testing is proactive
  • Diagnostics are reactive

• Many diagnostic tools available, including
  • Web UI
  • Logs
  • Counters
  • Job history
Each Hadoop daemon has its own Web application

<table>
<thead>
<tr>
<th>Daemon</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>JobTracker</td>
<td><a href="http://hostname:50030/">http://hostname:50030/</a></td>
</tr>
<tr>
<td>TaskTracker</td>
<td><a href="http://hostname:50060/">http://hostname:50060/</a></td>
</tr>
<tr>
<td>NameNode</td>
<td><a href="http://hostname:50070/">http://hostname:50070/</a></td>
</tr>
<tr>
<td>Secondary NameNode</td>
<td><a href="http://hostname:50090/">http://hostname:50090/</a></td>
</tr>
<tr>
<td>DataNode</td>
<td><a href="http://hostname:50075/">http://hostname:50075/</a></td>
</tr>
</tbody>
</table>
JobTracker Web UI Example
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
Submit Host: training01.mtv.cloudera.com
Submit Host Address: 172.20.63.11
Job-ACLs: All users are allowed
Job Setup: Successful
Status: Succeeded
Finished in: 25sec
Job Cleanup: Successful

<table>
<thead>
<tr>
<th>Kind</th>
<th>% Complete</th>
<th>Num Tasks</th>
<th>Pending</th>
<th>Running</th>
<th>Complete</th>
<th>Killed</th>
<th>Failed/Killed Task Attempts</th>
</tr>
</thead>
<tbody>
<tr>
<td>map</td>
<td>100.00%</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>0</td>
<td>0/0</td>
</tr>
<tr>
<td>reduce</td>
<td>100.00%</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0/0</td>
</tr>
</tbody>
</table>
### Task Attempt List

#### All Tasks

<table>
<thead>
<tr>
<th>Task</th>
<th>Complete</th>
<th>Start Time</th>
<th>Finish Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>task_201206281229_0182_m_000000</td>
<td>100.00%</td>
<td>10-Oct-2012 20:29:03</td>
<td>10-Oct-2012 20:29:09 (6sec)</td>
</tr>
<tr>
<td>task_201206281229_0182_m_000001</td>
<td>100.00%</td>
<td>10-Oct-2012 20:29:03</td>
<td>10-Oct-2012 20:29:09 (5sec)</td>
</tr>
<tr>
<td>task_201206281229_0182_m_000002</td>
<td>100.00%</td>
<td>10-Oct-2012 20:29:03</td>
<td>10-Oct-2012 20:29:09 (5sec)</td>
</tr>
<tr>
<td>task_201206281229_0182_m_000003</td>
<td>100.00%</td>
<td>10-Oct-2012 20:29:03</td>
<td>10-Oct-2012 20:29:08 (4sec)</td>
</tr>
<tr>
<td>task_201206281229_0182_m_000004</td>
<td>100.00%</td>
<td>10-Oct-2012 20:29:03</td>
<td>10-Oct-2012 20:29:09 (6sec)</td>
</tr>
<tr>
<td>task_201206281229_0182_m_000005</td>
<td>100.00%</td>
<td>10-Oct-2012 20:29:03</td>
<td>10-Oct-2012 20:29:09 (6sec)</td>
</tr>
<tr>
<td>task_201206281229_0182_m_000006</td>
<td>100.00%</td>
<td>10-Oct-2012 20:29:03</td>
<td>10-Oct-2012 20:29:10 (7sec)</td>
</tr>
<tr>
<td>task_201206281229_0182_m_000007</td>
<td>100.00%</td>
<td>10-Oct-2012 20:29:03</td>
<td>10-Oct-2012 20:29:10 (7sec)</td>
</tr>
</tbody>
</table>

Some columns removed to better fit the screen.
Logs

• Most important logs for MapReduce developers
  • stdout
  • stderr
  • syslog

• Demo: How to debug error using Web UI and logs
## Counters

### Counters for attempt_201206281229_0067_m_000001_0

<table>
<thead>
<tr>
<th>File System Counters</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>FILE: Number of bytes read</td>
<td>0</td>
</tr>
<tr>
<td>FILE: Number of bytes written</td>
<td>66,292</td>
</tr>
<tr>
<td>FILE: Number of read operations</td>
<td>0</td>
</tr>
<tr>
<td>FILE: Number of large read operations</td>
<td>0</td>
</tr>
<tr>
<td>FILE: Number of write operations</td>
<td>0</td>
</tr>
<tr>
<td>HDFS: Number of bytes read</td>
<td>63,501</td>
</tr>
<tr>
<td>HDFS: Number of bytes written</td>
<td>0</td>
</tr>
<tr>
<td>HDFS: Number of read operations</td>
<td>2</td>
</tr>
<tr>
<td>HDFS: Number of large read operations</td>
<td>0</td>
</tr>
<tr>
<td>HDFS: Number of write operations</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Map-Reduce Framework</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Map input records</td>
<td>1,797</td>
</tr>
<tr>
<td>Map output records</td>
<td>1,797</td>
</tr>
<tr>
<td>Map output bytes</td>
<td>14,376</td>
</tr>
<tr>
<td>Input split bytes</td>
<td>139</td>
</tr>
<tr>
<td>Combine input records</td>
<td>0</td>
</tr>
<tr>
<td>Combine output records</td>
<td>0</td>
</tr>
<tr>
<td>Spilled Records</td>
<td>1,797</td>
</tr>
<tr>
<td>CPU time spent (ms)</td>
<td>3,130</td>
</tr>
<tr>
<td>Physical memory (bytes)</td>
<td>336,834,560</td>
</tr>
<tr>
<td>Virtual memory (bytes)</td>
<td>1,543,569,408</td>
</tr>
<tr>
<td>Total committed heap usage (bytes)</td>
<td>503,971,840</td>
</tr>
</tbody>
</table>

org.apache.hadoop.mapreduce.lib.input.FileInputFormatCounter

BYTES_READ | 63,352
Viewing Current Configuration

http://HOST:PORT/conf
# Job Properties

## Job Configuration: Jobld - job_201206281229_0067

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

- There are many types of testing
  - Unit
  - Integration
  - Performance
- Diagnostics help us find the problems we missed
Resources for More Info

- Johannes Link
Resources for More Info (cont’d)

- Tom White
- ISBN: 1-449-31152-0
Resources for More Info (cont’d)

• Charlie Hunt and Binu John
• ISBN: 0-137-14252-8
Resources for More Info (cont’d)

- Eric Sammer
Conclusion

- Testing helps develop trust in a system
  - It should behave as you expect it to
Thank You!

Tom Wheeler, Senior Curriculum Developer, Cloudera