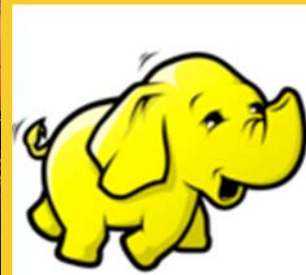




MapReduce Running Jobs

Originals of slides and source code for examples: <http://www.coreservlets.com/hadoop-tutorial/>
Also see the customized Hadoop training courses (onsite or at public venues) – <http://courses.coreservlets.com/hadoop-training.html>

Customized Java EE Training: <http://courses.coreservlets.com/>
Hadoop, Java, JSF 2, PrimeFaces, Servlets, JSP, Ajax, jQuery, Spring, Hibernate, RESTful Web Services, Android.
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.



For live customized Hadoop training (including prep for the Cloudera certification exam), please email info@coreservlets.com

Taught by recognized Hadoop expert who spoke on Hadoop several times at JavaOne, and who uses Hadoop daily in real-world apps. Available at public venues, or customized versions can be held on-site at your organization.

- Courses developed and taught by Marty Hall
 - JSF 2.2, PrimeFaces, servlets/JSP, Ajax, jQuery, Android development, Java 7 or 8 programming, custom mix of topics
 - Courses available in any state or country. Maryland/DC area companies can also choose afternoon/evening courses.
- Courses developed and taught by coreservlets.com experts (edited by Marty)
 - Spring, Hibernate/JPA, GWT, Hadoop, HTML5, RESTful Web Services

Contact info@coreservlets.com for details



Agenda

- **Tool, ToolRunner and GenericOptionsParser**
- **Running MapReduce Locally**
- **Running MapReduce on Cluster**
- **Packaging MapReduce Jobs**
- **MapReduce CLASSPATH**
- **Submitting Jobs**
- **Logs and Web UI**

4

Tool and ToolRunner

- **Utilize Tool and ToolRunner to stage and configure MapReduce jobs**
- **Tool – an interface designed to deal with command line arguments**

- Standard for any MapReduce application

```
public interface Tool extends Configurable {  
    int run(String [] args) throws Exception;  
}
```

- Configurable interface defines a getter and setter for Configuration object

```
public interface Configurable {  
    void setConf(Configuration conf);  
    Configuration getConf();  
}
```

5

ToolRunner

- **Utility to run classes that implement Tool**
- **Delegates to GenericOptionsParser**
 - Utility that parses command line arguments
 - Sets the arguments on Configuration object
 - Enables the command line usage we've already seen:
 - \$yarn command [genericOptions]
 - Usually NOT used directly

6

Tool and ToolRunner Usage

```
public class MyTool extends Configured implements Tool{  
  
    @Override  
    public int run(String[] args) throws Exception {  
        Job job =  
            Job.getInstance(getConf(), "MyToolJob");  
        ...  
        ...  
        return job.waitForCompletion(true) ? 0 : 1;  
    }  
  
    public static void main(String[] args) throws Exception {  
        int exitCode = ToolRunner.run(new MyTool(), args);  
        System.exit(exitCode);  
    }  
}
```

Configured implements
getters and setters for
Configuration object

Typical
creation of
Job object

Execute the job

7

GenericOptionsParser

```
$ yarn command [genericOptions] [commandOptions]
```

Generic Option	Description
-conf <conf_file.xml>	Adds the properties inside the provided file to the Configuration object
-Dproperty=value	Set's the provided property to the provided value in the Configuration object
-fs URI	-fs URI Overrides the default filesystem with the provided URI; similarly you can accomplish the same via -Dfs.default.name=URI
-files <file,file,file>	Makes the provided files readily available to MapReduce jobs by copying files to DistributedCache
-libjars <f.jar, f2.jar>	Adds the provided jars to the tasks' CLASSPATH for the MapReduce job

```
$ yarn jar $PLAY_AREA/HadoopSamples.jar \  
  mr.wordcount.StartWithCountJob_HBase \  
  -libjars $HBASE_HOME/hbase-0.92.1-cdh4.0.0-security.jar
```

8

GenericOptionsParser Format and Example

```
$ yarn command [genericOptions] [commandOptions]
```



```
$yarn \  
command { jar $PLAY_AREA/HadoopSamples.jar \  
          mr.wordcount.StartsWithCountJob \  
generic  { -libjars hbase-0.92.1-cdh4.0.0b2-security.jar \  
Options  { -D mapreduce.job.reduces=10 \  
command  { /training/playArea/hamlet.txt \  
Options  { /training/playArea/wordCount/
```

**Available MapReduce Options:

<http://hadoop.apache.org/docs/r2.0.0-alpha/hadoop-mapreduce-client/hadoop-mapreduce-client-core/mapred-default.xml>

9

Running MapReduce Locally

- **Hadoop is packaged with a local job runner**
 - Run MapReduce code in a single JVM
 - Great for IDE usage, can even use a debugger
 - Handy for testing
 - Note: Can only support single Reducer and silently ignores when more than 1 reduce is configured
- **Enable local mode by setting `mapreduce.framework.name` property to local**

```
$ yarn jar $PLAY_AREA/HadoopSamples.jar \
    mr.wordcount.StartsWithCountJob \
    -D mapreduce.framework.name=local \
    /training/playArea/hamlet.txt \
    /training/playArea/wordCount/
```

10

Running MapReduce Locally

- **Previous call still utilized HDFS**
 - HDFS is set as the default FileSystem in `core-site.xml`
- **You can override default file system so full MapReduce life-cycle is executed locally**
 - `fs` generic option

```
$ yarn jar $PLAY_AREA/HadoopSamples.jar \
    mr.wordcount.StartsWithCountJob \
    -D mapreduce.framework.name=local \
    -fs file:/// \
    /home/hadoop/Training/exercises/sample_data \
    $PLAY_AREA/wordCountOutput/
```

- Specify filesystem in each path

```
$ yarn jar $PLAY_AREA/HadoopSamples.jar \
    mr.wordcount.StartsWithCountJob \
    -D mapreduce.framework.name=local \
    file:/home/hadoop/Training/exercises/sample_data \
    file:$PLAY_AREA/wordCountOutput/
```

11

LocalJobRunner in Unit Test - StartsWithCountJobTests.java

```
public class StartsWithCountJobTests {  
  
    private File inputFile =  
        new File("../target/test/input.txt");  
    private File output = new File("../target/test-result/");  
  
    @Before  
    public void setUpTest() throws IOException{  
        FileUtils.deleteQuietly(inputFile);  
        FileUtils.deleteQuietly(output);  
        FileUtils.writeStringToFile(inputFile,  
            "this is just a test, yes it is");  
    }  
    ...  
    ...  
    ...  
}
```

This will serve as an
input text to the job
under test

12

StartsWithCountJobTests.java

```
@Test  
public void testRun() throws Exception {  
    Configuration conf = new Configuration();  
    conf.set("mapreduce.framework.name", "local");  
    conf.set("fs.default.name", "file:///");  
  
    StartsWithCountJob underTest = new StartsWithCountJob();  
    underTest.setConf(conf);  
  
    int exitCode = underTest.run(  
        new String[]{inputFile.getAbsolutePath(),  
            output.getAbsolutePath()});  
    assertEquals("Returned error code.", 0, exitCode);  
    assertTrue(new File(output, "_SUCCESS").exists());  
    Map<String,Integer> resAsMap =  
        getResultAsMap(new File(output, "part-r-00000"));  
  
    assertEquals(5, resAsMap.size());  
    assertEquals(2, resAsMap.get("t").intValue());  
    assertEquals(3, resAsMap.get("i").intValue());  
    assertEquals(1, resAsMap.get("j").intValue());  
    assertEquals(1, resAsMap.get("a").intValue());  
    assertEquals(1, resAsMap.get("y").intValue());  
}
```

Configure the job to
run locally

Use local filesystem

Execute the job

For verification,
convert contents of
reduce output to a
map

13

StartsWithCountJobTests.java

```
private Map<String, Integer> getResultAsMap(File file)
    throws IOException {

    Map<String, Integer> result = new HashMap<String, Integer>();
    String contentOfFile = FileUtils.readFileToString(file);
    for (String line : contentOfFile.split("\n")){
        String [] tokens = line.split("\t");
        result.put(tokens[0], Integer.parseInt(tokens[1]));
    }
    return result;
}
```

Parse expected file

(1) there is a key-value pair per line

(2) key is tab separated from value

Here is sample format

a	1
i	3
j	1
t	2
y	1

14

Provide Local Configuration File

- **Another way to configure your job is to create a configuration file and provide it via `-conf` options**

```
$ yarn jar $PLAY_AREA/HadoopSamples.jar \
    mr.wordcount.StartsWithCountJob \
    -conf $PLAY_AREA/local/run-local-config.xml \
    /home/hadoop/Training/exercises/sample_data \
    $PLAY_AREA/wordCountOutput/
```

- **Each property specified in `run-local-config.xml` will be set on `StartsWithCountJob`'s Configuration object**

15

run-local-config.xml

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
  <property>
    <name>fs.default.name</name>
    <value>file:///</value>
  </property>
  <property>
    <name>mapreduce.framework.name</name>
    <value>local</value>
  </property>
</configuration>
```

16

Running Your Job on a Cluster

- 1. Package a Job**
 - Set up Job's and Task's CLASSPATH
- 2. Submit a Job**
- 3. Monitor a Job**

17

Package a Job

- **Package classes into a JAR file**
 - Already been doing that
 - Will submit the job to the Hadoop/YARN cluster
 - `$yarn jar MyJar.jar com.jobs.JobTool`
 - May want to utilize a tool to create a jar file such as Maven or Ivy
- **In the implementation of Tool call `Job.setJarByClass()`**
 - Provide a class that exists in your jar
 - Framework will locate the jar by scanning classpath for the provided class
- **Optionally package dependencies**
 - Package jar files in a lib sub-directory inside your jar
 - Package resource files in the classes sub-directory inside your jar

18

MapReduce CLASSPATH

- **Client's CLASSPATH**
 - “`$yarn jar blah.jar com.JobClass`” command executes within “client” JVM
 - Tool's implementation CLASSPATH
- **Task's CLASSPATH**
 - Map and Reduce tasks
 - Executes on the cluster => remote machine(s)

19

Client's CLASSPATH

- **CLASSPATH is made of**
 - The classes in the provided JAR which contains the job
 - Jar files in the lib sub-directory of the job's Jar
 - Resource files in the classes sub-directory of the job's Jar
 - JARs specified on the HADOOP_CLASSPATH environment variable
 - IMPORTANT: set in the \$HADOOP_CONF_DIR/hadoop-env.sh
- **To see what is on CLASSPATH**

```
$ yarn classpath
```

```
/home/hadoop/Training/CDH4/hadoop-2.0.0-cdh4.0.0/conf:/home/hadoop/Training/CDH4/hadoop-2.0.0-cdh4.0.0/conf:/home/hadoop/Training/CDH4/hadoop-2.0.0-cdh4.0.0/conf:/home/hadoop/Training/CDH4/hadoop-2.0.0-cdh4.0.0/share/hadoop/common/lib/*....
```

20

Task's CLASSPATH

- **Task's CLASSPATH is made of**
 - The classes in the actual JAR that contains the job
 - Jar files in the lib sub-directory of the job's Jar
 - Resource files in the classes sub-directory of the job's Jar
 - Jars added to classpath via DistributedCache
 - \$yarn jar job.jar com.Job -libjars jar1.jar,jar2.jar
 - job.addFileToClassPath(path)
- **Does NOT use \$HADOOP_CLASSPATH environment variable**
- **Benefit vs. Fallback for DistributedCache usage**
 - Add them once, no need to build them into the JAR
 - Reduces bandwidth usage
 - NOT as flexible as other approach - if you package classes into a JAR then each job can utilize their own unique dependencies

21

Dependency Conflicts

- **Hadoop's Jars are added to the CLASSPATH as well**
 - User's and Framework's code runs off the same CLASSPATH
- **Hadoop's internal dependencies by default take priority over the provided Jars**
- **User can override jar loading precedence**
 - Client's CLASSPATH by setting HADOOP_USER_CLASSPATH_FIRST environment variable to true
 - Task's CLASSPATH by setting mapreduce.task.classpath.first property to true.
 - **WARNING**: Changing will alter class loading for the Hadoop itself which may cause unexpected results. Use with caution.

22

Submit a Job

```
yarn jar ~/Training/play_area/HadoopSamples.jar \  
  mr.wordcount.StartsWithCountJob \  
  /training/playArea/hamlet.txt \  
  /training/playArea/wordCount/  
  
...  
...  
(Job.java:monitorAndPrintJob(1270)) - Running job:  
job_1339291219653_0026  
(Job.java:monitorAndPrintJob(1291)) - Job job_1339291219653_0026  
running in uber mode : false  
(Job.java:monitorAndPrintJob(1298)) - map 0% reduce 0%  
(Job.java:monitorAndPrintJob(1298)) - map 100% reduce 0%  
(Job.java:monitorAndPrintJob(1298)) - map 100% reduce 100%  
(Job.java:monitorAndPrintJob(1309)) - Job job_1339291219653_0026  
completed successfully  
(Job.java:monitorAndPrintJob(1316)) - Counters: 43  
File System Counters  
  FILE: Number of bytes read=798  
  FILE: Number of bytes written=99384  
  FILE: Number of read operations=0  
  FILE: Number of large read operations=0  
...
```

Refer to this job by this id

job.waitForCompletion(true) executes the job and prints progress to the screen

23

Job ID

Resource Manager
Start Timestamp

Counter

job_1339803233775_0002

- **Use Job ID as a reference**

- In the logs
- In the Web UI
- Via \$mapred job command

24

\$mapred job Commandx

- **Get the status of a job**

- \$ mapred job -status <job_id>

- **Kill a Job**

- \$ mapred job -kill <job_id>

- **View logs of a task attempt**

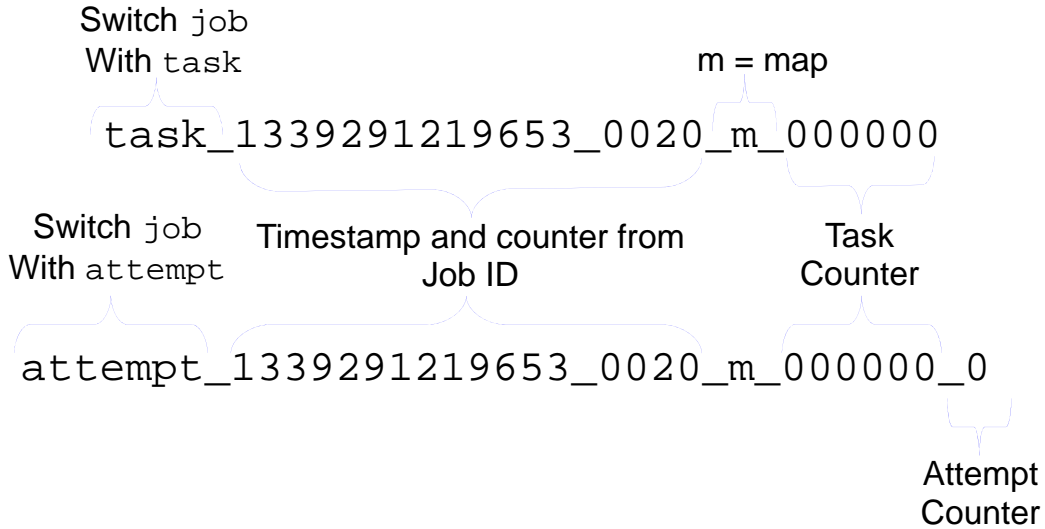
- \$ mapred job -logs <job_id> <attempt_id>
 - Can not view logs of a running job – use management Web UI

- **Learn about other options**

- \$ mapred job

25

Task ID and Attempt ID



- **Job is made of tasks**
- **Tasks are made of attempts**

26

Logs

- **Hadoop uses Log4j for logging, to learn more about log4j please visit <http://logging.apache.org/log4j/1.2/>**
- **Hadoop maintains several log types**
 - System Daemon Logs
 - Audit Logs
 - MapReduce Job and Task History Logs

27

System Daemon Logs

- **Logs' format**

product + username + daemon + machine hostname . out

- **HDFS Logs located under \$HADOOP_LOG_DIR**

- set hadoop-env.sh.

- hadoop-dima-namenode-host.out
- hadoop-dima-datanode-host.out
- hadoop-dima-secondarynamenode-host.out

- **MapReduce and Yarn Logs located under \$YARN_LOG_DIR**

- Set in yarn-env.sh

- yarn-dima-resourcemanager-host.out
- yarn-dima-nodemanager-host.out
- yarn-dima-historyserver-host.out

28

System Daemon Logs

- **Daemon Logs are configured by editing log4j configuration**

- \$HADOOP_CONF_DIR/log4j.properties

29

Audit Log(s)

- **Hadoop is capable of logging audit events**
 - HDFS Audit
 - MapReduce Audit
- **Implemented via log4j**
- **Modify**
\$HADOOP_CONF_DIR/log4j.properties
 - For example:
log4j.logger.org.apache.hadoop.hdfs.server.namenode.FS
Namesystem.audit = INFO
- **Full instructions can be found at**
 - <http://wiki.apache.org/hadoop/HowToConfigure>

30

MapReduce Job's Logs

- **Job and task information and their logs are archived to the history server**
 - View Job and Tasks Logs
 - History Server Hosts Web UI
 - <http://localhost:19888/jobhistory>
- **Can also use \$mapred job command**
 - \$ mapred job -logs job_1339817050993_0001 | more
 - Get logs for the job
 - \$ mapred job -logs job_1339817050993_0001
attempt_1339817050993_0001_m_000000_0 | more
 - Get logs for the task's attempt

31

MapReduce Job's Logs

- **By default history logs are archived to HDFS**
 - /tmp/logs
 - /tmp/hadoop-yarn
- **The location of the log can be affected by several properties**
 - mapred-site.xml
 - mapreduce.jobhistory.intermediate-done-dir: MapReduce jobs write their history files here
 - mapreduce.jobhistory.done-dir: History Server archives job files here
 - yarn-site.xml
 - yarn.nodemanager.remote-app-log-dir: Application logs are moved here after completion, yarn.log-aggregation-enable property needs to be set to true

32

MapReduce Job's Logs

- **By default Map and Reduce tasks will log at INFO level**
- **Modify logging level by setting Hadoop Job properties**
 - Map Tasks: mapred.map.child.log.level
 - Reduce Tasks: mapred.reduce.child.log.level
- **You can also modify logging level at the command line**

```
$ yarn jar $PLAY_AREA/Solutions.jar \  
    mapRed.inputAndOutput.UniqueCounterTool \  
    -Dmapred.map.child.log.level=DEBUG
```

33

History Server - Web UI



MapReduce Job job_1339817050993_0001

Logged in as: weou...

Job ID

Job Name: StartsWithCount:HBase
User Name: hadoop
Queue: default
State: SUCCEEDED
Uberized: false
Started: Fri Jun 15 23:27:54 EDT 2012
Finished: Fri Jun 15 23:28:06 EDT 2012
Elapsed: 11sec
Diagnostics:
Average Map Time: 7sec
Average Reduce Time: 0sec
Average Shuffle Time: 7sec
Average Merge Time: 7sec

Jobs Logs

ApplicationMaster	Attempt Number	Start Time	Node	Logs	
	1	Fri Jun 15 23:27:51 EDT 2012	localhost:8042	logs	
Task Type	Total	Complete			
Map	2	2			
Reduce	1	1			
Attempt Type	Failed	Killed	Successful		
Maps	0	0	2		
Reduces	0	0	1		

Task Details & Logs

34

© 2012 coreservlets.com and [Dima May](#)



Wrap-Up

Customized Java EE Training: <http://courses.coreservlets.com/>

Hadoop, Java, JSF 2, PrimeFaces, Servlets, JSP, Ajax, jQuery, Spring, Hibernate, RESTful Web Services, Android.

Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

Summary

- **We learned how to**
 - Utilize Tool, ToolRunner and GenericOptionsParser
 - Run MapReduce Locally
 - Run MapReduce on Cluster
 - Package MapReduce Jobs
 - Control MapReduce CLASSPATH
 - Submit Jobs
 - View logs via script and Web UI

36

© 2012 coreservlets.com and [Dima May](#)



Questions?

More info:

<http://www.coreservlets.com/hadoop-tutorial/> – Hadoop programming tutorial

<http://courses.coreservlets.com/hadoop-training.html> – Customized Hadoop training courses, at public venues or onsite at *your* organization

<http://courses.coreservlets.com/Course-Materials/java.html> – General Java programming tutorial

<http://www.coreservlets.com/java-8-tutorial/> – Java 8 tutorial

<http://www.coreservlets.com/JSF-Tutorial/jsf2/> – JSF 2.2 tutorial

<http://www.coreservlets.com/JSF-Tutorial/primefaces/> – PrimeFaces tutorial

<http://coreservlets.com/> – JSF 2, PrimeFaces, Java 7 or 8, Ajax, jQuery, Hadoop, RESTful Web Services, Android, HTML5, Spring, Hibernate, Servlets, JSP, GWT, and other Java EE training

Customized Java EE Training: <http://courses.coreservlets.com/>

Hadoop, Java, JSF 2, PrimeFaces, Servlets, JSP, Ajax, jQuery, Spring, Hibernate, RESTful Web Services, Android.

Developed and taught by well-known author and developer. At public venues or onsite at *your* location.