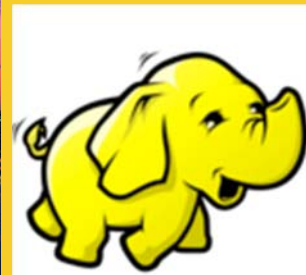




Map Reduce Features

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

- **Counters**
- **Speculative Execution**
- **Distributed Cache**

4

Counters

- **Instrument Job's metrics**
 - Gather statistics
 - Quality control – confirm what was expected
 - Diagnostics
- **Framework provides a set of built-in metrics**
 - For example bytes processed for input and output
- **User can create new counters**
 - Number of records consumed
 - Number of errors or warnings
- **Counters are divided into groups**
- **Tracks total, mapper and reducer counts**

5

Built-in Counters

- Maintains and sums up counts
- Several groups for built-in counters
 - Job Counters – documents number of map and reduce tasks launched, number of failed tasks
 - File System Counters – number of bytes read and written
 - Map-Reduce Framework – mapper, reducer, combiner input and output records counts, time and memory statistics

6

Job Counters

- Web UI exposes counters for each Job



Counters for job_1339260384899_0001

Counter Group	Name	Counters			
		Map	Reduce	Total	
File System Counters	FILE: Number of bytes read	298	0	298	
	FILE: Number of bytes written	104542	0	104542	
	FILE: Number of large read operations	0	0	0	
	FILE: Number of read operations	0	0	0	
	FILE: Number of write operations	0	0	0	
	HDFS: Number of bytes read	51853	0	51853	
	HDFS: Number of bytes written	62586	0	62586	
	HDFS: Number of large read operations	0	0	0	
	HDFS: Number of read operations	37	0	37	
	HDFS: Number of write operations	14	0	14	
Job Counters	Data-local map tasks	0	0	1	
	Launched map tasks	0	0	1	
	Total time spent by all maps in occupied slots (ms)	0	0	4408	
Map-Reduce Framework	CPU time spent (ms)	1920	0	1920	
	Failed Shuffles	0	0	0	
	GC time elapsed (ms)	21	0	21	
	Input split bytes	169	0	169	
	Map input records	1	0	1	
	Map output records	0	0	0	
	Merged Map outputs	0	0	0	
	Physical memory (bytes) snapshot	85540864	0	85540864	
	Spilled Records	0	0	0	
	Total committed heap usage (bytes)	37355520	0	37355520	

7

User Defined Counters

- **You can create new counters**
- **Increment counters in Reducer and/or Mapper classes**
 - Framework accurately sums up counts between various stages and produces totals
 - Accounts for failed tasks

8

Implement User-Defined Counters

- 1. Retrieve Counter from Context object**
 - Framework injects Context object into map and reduce methods
- 2. Increment Counter's value**
 - Can increment by 1 or more

9

1: Retrieve Counter from Context

- **Utilize Context object**
 - void map(Key key, Value value, Context context)
 - void reduce(Key key, Iterable<Value> values, Context context)
- **Although map's and reduce's Context type is not the same they both extend from**

```
org.apache.hadoop.mapreduce.TaskAttemptContext
```

- **TaskAttemptContext provides two ways to retrieve counters**
 - public Counter getCounter(String groupName, String counterName);
 - public Counter getCounter(Enum<?> counterName);
 - Figures out group name from fully qualified classname of enum - enum.getDeclaringClass().getName()

10

2: Increment Counter's Value

- **Increment or even set the value**
 - void setValue(long value);
 - void increment(long incr);

11

StartsWithCountMapper with Counters

- Recall the StartsWithCountJob
- Update Mapper to document counts for
 - Total tokens processed
 - Number of tokens that start with uppercase
 - Number of tokens that start with lowercase
- First create an enum to reference these counters:

```
public enum Tokens {  
    Total, FirstCharUpper, FirstCharLower  
}
```

12

StartsWithCountMapper_UserCounters.java

```
@Override  
protected void map(LongWritable key, Text value, Context context)  
    throws IOException, InterruptedException {  
  
    StringTokenizer tokenizer = new StringTokenizer(value.toString());  
    while (tokenizer.hasMoreTokens()) {  
        String token = tokenizer.nextToken();  
        reusableText.set(token.substring(0, 1));  
        context.write(reusableText, countOne);  
  
        context.getCounter(Tokens.Total).increment(1);  
        char firstChar = token.charAt(0);  
  
        if (Character.isUpperCase(firstChar)){  
            context.getCounter(Tokens.FirstCharUpper).increment(1);  
        } else {  
            context.getCounter(Tokens.FirstCharLower).increment(1);  
        }  
    }  
}
```

Keep count of total tokens processed

Stats on tokens that start with upper case vs. lowercase

13

Run StartsWithCountMapper_UserCounters

```
$ yarn jar $PLAY_AREA/HadoopSamples.jar  
mr.wordcount.StartsWithCountJob_UserCounters  
/training/data/hamlet.txt /training/playArea/wordCount/  
...  
...  
...  
Map output records=34189  
Map output bytes=205134  
Combine input records=34189  
Combine output records=69  
Reduce input records=69  
Reduce output records=69  
mr.wordcount.StartsWithCountMapper_UserCounters$Tokens  
FirstCharLower=26080  
FirstCharUpper=8109  
Total=34189  
File Input Format Counters  
Bytes Read=211294  
File Output Format Counters  
Bytes Written=385
```

Job configures new mapper with counts

Group was derived from the class name

Total # of tokens should match Map output records

14

Customize Counter's Names

- Can customize counter and group names when using enums
 1. Create a properties file <classname>.properties defining counter name properties
 - Inner classes are substituted by underscore
 - For example: org.com.MyMapper\$CustomEnum would be MyMapper_CustomEnum.properties
 2. Place properties file in the same package as the class that defines Enum

15

1: Create Properties File

- **In our case the enum was defined in**
 - mr.wordcount.StartsWithCountMapper_UserCounters\$Tokens
- **Therefore the file is to be named**
 - StartsWithCountMapper_UserCounters_Tokens.properties
- **Define Group and Counter names:**

```
CounterGroupName = Token Processed
Total.name=Total Tokens Processed
FirstCharUpper.name=Tokens start with Uppercase
FirstCharLower.name=Tokens start with Lowercase
```

16

Test Counter Re-Naming

```
$ yarn jar $PLAY_AREA/HadoopSamples.jar
mr.wordcount.StartsWithCountJob_UserCounters
/training/data/hamlet.txt /training/playArea/wordCount/
...
...
...
    Map output records=34189
    Map output bytes=205134
    Combine input records=34189
    Combine output records=69
    Reduce input records=69
    Reduce output records=69
Token Processed
    Tokens start with Lowercase=26080
    Tokens start with Uppercase=8109
    Total Tokens Processed =34189
File Input Format Counters
  Bytes Read=211294
File Output Format Counters
  Bytes Written=385
```

New Names are mapped by the framework

17

Retrieving Counters

1. Web-UI



Counter Group	Name	Value	Min	Max	Total
Job	F&B: Number of bytes read	298	0	298	298
	F&B: Number of bytes written	204542	0	204542	204542
	F&B: Number of single record operations	0	0	0	0
	F&B: Number of read operations	0	0	0	0
	F&B: Number of write operations	0	0	0	0
File System Counters	HDFS: Number of bytes read	10883	0	10883	10883
	HDFS: Number of bytes written	42586	0	42586	42586
	HDFS: Number of single record operations	0	0	0	0
	HDFS: Number of read operations	37	0	37	37
	HDFS: Number of write operations	14	0	14	14
Job Counters	Time spent in map tasks	0	0	1	1
	Time spent in reduce tasks	0	0	1	1
	Total time spent by all runners in user-defined map tasks	0	0	4838	4838
Map Reduce Framework	ChY: Long count errors	1530	0	1530	1530
	Failed Shuffle	0	0	0	0
	GC time spent in map	25	0	25	25
	Map output bytes	189	0	189	189
	Map output records	1	0	1	1
Map Reduce Framework	Map output records	0	0	0	0
	Physical memory (in MB) requested	8540864	0	8540864	8540864
	Virtual memory	0	0	0	0
	Total committed heap memory (in MB)	3739328	0	3739328	3739328

2. Command line

- \$ mapred job -status <job_id>

3. Java API

- Further analyze counts
- Store in a database

18

Java API to Retrieve Counters

- **Print all the counters after the job is done**
 - Snippet from StartsWithCountJob_PrintCounters.java

```
int resultCode = job.waitForCompletion(true) ? 0 : 1;

System.out.println("Job is complete! Printing Counters:");
Counters counters = job.getCounters();

for (String groupName : counters.getGroupNames()){
    CounterGroup group = counters.getGroup(groupName);
    System.out.println(group.getDisplayName());

    for (Counter counter : group.getUnderlyingGroup()){
        System.out.println("  " + counter.getDisplayName() +
            " = " + counter.getValue());
    }
}
```

19

Speculative Execution

- **Job is decomposed into small tasks**
- **Job is as fast as the slowest task**
- **Given 100s or even 1000s of tasks**
 - Few tasks may run very slowly (hardware issues, other activity on that machine, configuration, etc...)
- **MapReduce framework strives to resolve slow running tasks by spawning the same task on a different machine**
 - Doesn't start speculative tasks immediately

20

Speculative Execution

- **Will spawn a speculative task when**
 - All the tasks have been started
 - Task has been running for an extended period of time
 - over a minute
 - Did not make significant progress as compared to the rest of the running tasks
- **After task's completion duplicates are killed**
- **Just an optimization**

21

Speculative Execution

- **Can be turned off by setting these properties to false**
 - `mapred.map.tasks.speculative.execution`
 - Turn on/off speculative execution for map phase
 - `mapred.reduce.tasks.speculative.execution`
 - Turn on/off speculative execution for reduce phase
- **When should I disable Speculative Execution?**
 - Task is outputting directly to a shared resource; then starting a duplicate task may cause unexpected results
 - Minimize cluster and bandwidth usage; duplicate tasks use up resources

22

Distributed Cache

- **A mechanism to distribute files**
- **Make them available to MapReduce task code**
- **yarn command provides several options to add distributed files**
- **Can also use Java API directly**
- **Supports**
 - Simple text files
 - Jars
 - Archives: zip, tar, tgz/tar.gz

23

Distributed Cache via \$ yarn Command

- **Update StartWithCount job to exclude specified start letters**
 1. Load a file which contains start letters to exclude onto distributed cache
 - utilize -files parameter with the yarn command
 2. Update Map code to utilize the exclude file

24

1: Load Exclude File Onto DistributedCache

```
$ cat $PLAY_AREA/data/startWithExcludeFile.txt  
b  
c  
d  
e  
f  
G
```

Exclude tokens that start with these letters

```
$ yarn jar $PLAY_AREA/HadoopSamples.jar \  
mr.wordcount.StartsWithCountJob_DistCache \  
-files $PLAY_AREA/data/startWithExcludeFile.txt \  
/training/data/hamlet.txt \  
/training/playArea/wordCount/
```

Using -files option yarn command will place the file onto DistributedCache.

25

2: Utilize Exclude File in the Map Code

```
public class StartsWithCountMapper_DistCache extends
    Mapper<LongWritable, Text, Text, IntWritable> {
    private Logger log =
        Logger.getLogger(StartsWithCountMapper_DistCache.class);
    private final static IntWritable countOne =
        new IntWritable(1);
    private final Text reusableText = new Text();

    public final static String EXCLUDE_FILE =
        "startWithExcludeFile.txt";
    private final Set<String> excludeSet = new HashSet<String>();
    ...
    ...
    ...
```

Will be able to directly
reference the file without
absolute path;
Constructs an exclude set

26

2: Utilize Exclude File in the Map Code

```
...
...
@Override
protected void setup(Context context) throws IOException,
    InterruptedException {
    FileReader reader = new FileReader(new File(EXCLUDE_FILE));
    try {
        BufferedReader bufferedReader = new BufferedReader(reader);
        String line;
        while ((line = bufferedReader.readLine()) != null) {
            excludeSet.add(line);
            log.info("Ignoring words that start with [" + line + "]);
        }
    } finally {
        reader.close();
    }
}
...
...
```

Framework takes care of all the magic;
the file is now stored locally and can
be referenced with just a name

27

2: Utilize Exclude File in the Map Code

```
...
...
@Override
protected void map(LongWritable key, Text value,
    Context context) throws IOException, InterruptedException {

    StringTokenizer tokenizer =
        new StringTokenizer(value.toString());

    while (tokenizer.hasMoreTokens()) {
        String firstLetter =
            tokenizer.nextToken().substring(0, 1);

        if (!excludeSet.contains(firstLetter)) {
            reusableText.set(firstLetter);
            context.write(reusableText, countOne);
        }
    }
}
}
```

28

Output of Distributed Cache Job

```
$ hdfs dfs -cat /training/playArea/wordCount/part-r-00000
...
...
[      122
-       1
a     2370
h     1724
i     1468
j       65
k     222
...
...
```

← Exclude letters are missing

29

Distributed Cache Inner-Workings

- **Accepts two types: files and archives**
 - Archives are unarchived on the local node
- **Items specified to the \$yarn command via -files, -libjars and -archives are copied to HDFS**
- **Prior to task execution these files are copied locally from HDFS**
 - Files now reside on a local disk – local cache
- **Files provided to the -libjars are appended to task's CLASSPATH**
- **Locally cached files become qualified to be deleted after all tasks utilizing cache complete**

30

Distributed Cache Inner-Workings

- **Files in the local cache are deleted after a 10GB threshold is reached**
 - Allow space for new files
 - Configured via `yarn.nodemanager.localizer.cache.target-size-mb` property
- **Local cache is stored under**
`${yarn.nodemanager.local-dirs}/usercache/$user/filecache`
 - Task code is not aware of the location
 - Symbolic link is created for each file, that's why we were able to reference a file without the path
 - `FileReader reader = new FileReader(new File(EXCLUDE_FILE))`

31

Java API - DistributedCache

- **Typically don't need to use directly**
 - Delegate to \$yarn command (with -files, -libjars or -archive options)
- **However when programmatic involvement is necessary use DistributedCache class**
 - File(s) to be cached must exist on HDFS
 - With \$yarn command framework takes care of this step for you
 - In the Tool – Place data into cache via methods on Job
 - In the Task – Retrieve the file from the cache

32

1: File(s) to be Cached Must Exist on HDFS

- **Add file to HDFS, either via command line or via FileSystem Java API**
- **Major difference from using \$yarn command where framework adds it to HDFS on your behalf**

```
$ hdfs dfs -put startWithExcludeFile.txt /training/data/  
$ hdfs dfs -ls /training/data/  
Found 3 items  
 0 2011-12-24 11:21 /training/data/glob  
22 2011-12-20 22:12 /training/data/readMe.txt  
12 2012-06-17 16:08 /training/data/startWithExcludeFile.txt
```

33

2: In the Tool – Place Data Onto Cache via Methods on Job

StartsWithCountJob DistCacheAPI.java

```
....
public int run(String[] args) throws Exception {
    Job job = Job.getInstance(getConf(), getClass().getSimpleName());
    job.setJarByClass(getClass());
    ...
    ...
    ...
    Path toCache = new Path("/training/data/startWithExcludeFile.txt");
    job.addCacheFile(toCache.toUri());
    job.createSymlink();

    return job.waitForCompletion(true) ? 0 : 1;
}
....
```

← Add file to DistributedCache

Create symbolic links for all files in DistributedCache;
without the links you would have to use fully qualified
path, in this case "/training/data/startWithExcludeFile.txt"

34

3: In the Task – Retrieve the File From the Cache

- Same as before:

StartsWithCountMapper DistCache.java

```
...
...
@Override
protected void setup(Context context) throws IOException,
    InterruptedException {
    FileReader reader = new FileReader(new File(EXCLUDE_FILE));
    try {
        BufferedReader bufferedReader = new BufferedReader(reader);
        String line;
        while ((line = bufferedReader.readLine()) != null) {
            excludeSet.add(line);
            log.info("Ignoring words that start with ["+line+"]");
        }
    } finally {
        reader.close();
    }
}
...
...

```

35



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 about**
 - Counters
 - Speculative Execution
 - Distributed Cache



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.