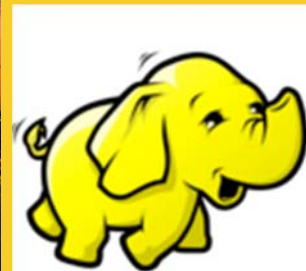




Map Reduce 2.0 Developing First MapReduce Job

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

- **Introduce MapReduce framework**
- **Implement first MapReduce Job**

4

MapReduce

- **Divided in two phases**
 - Map phase
 - Reduce phase
- **Both phases use key-value pairs as input and output**
- **The implementer provides map and reduce functions**
- **MapReduce framework orchestrates splitting, and distributing of Map and Reduce phases**
 - Most of the pieces can be easily overridden

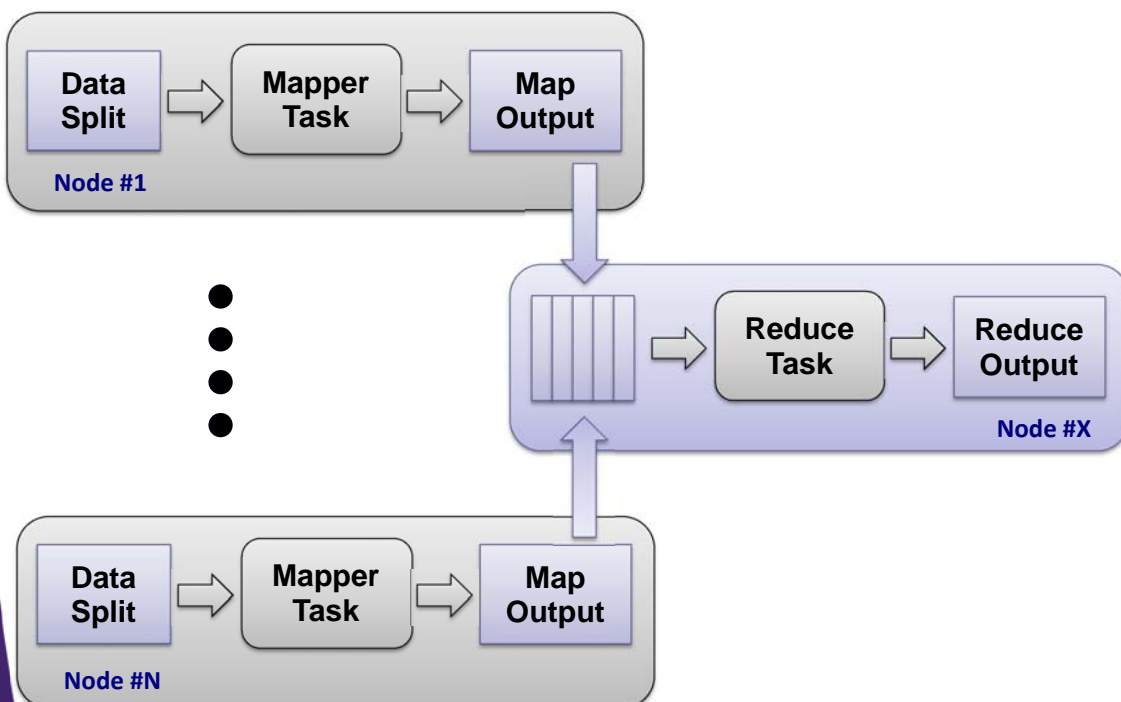
5

MapReduce

- **Job** – execution of map and reduce functions to accomplish a task
 - Equal to Java's main
- **Task** – single Mapper or Reducer
 - Performs work on a fragment of data

6

Map Reduce Flow of Data



7

First Map Reduce Job

• StartsWithCount Job

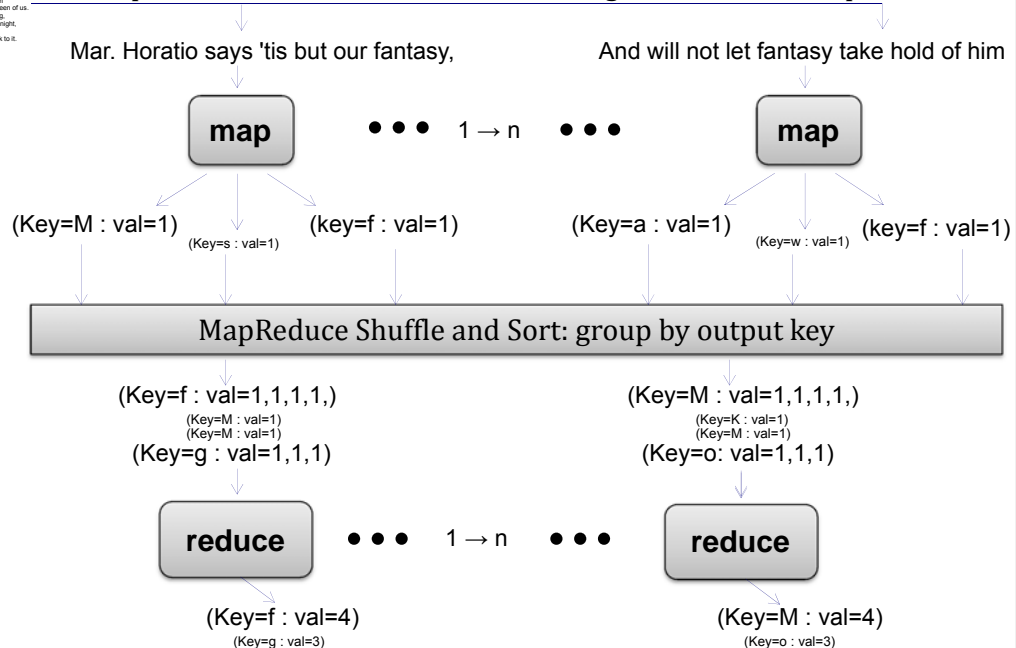
- Input is a body of text from HDFS
 - In this case hamlet.txt
- Split text into tokens
- For each first letter sum up all occurrences
- Output to HDFS

8

Word Count Job

Mar. What has this thing appear'd again to-night?
Hor. I have seen nothing.
Mar. Horatio says 'tis but our fantasy,
And will not let our better parts of him
Touching this deaded night, twice seen of us.
Thou'rt mad. I have entreated him abroad;
With us to watch the minutes of this night,
That if again this apparition come,
He may approve our eyes and speak to it.
Hor. Tush, tush, 'twere not appear.

MapReduce breaks text into lines feeding each line into map functions



9

StartsWithCount Job

1. Configure the Job

- Specify Input, Output, Mapper, Reducer and Combiner

2. Implement Mapper

- Input is text – a line from hamlet.txt
- Tokenize the text and emit first character with a count of 1 - <token, 1>

3. Implement Reducer

- Sum up counts for each letter
- Write out the result to HDFS

4. Run the job

10

1: Configure Job

• Job class

- Encapsulates information about a job
- Controls execution of the job

```
Job job = Job.getInstance(getConf(), "StartsWithCount");
```

• A job is packaged within a jar file

- Hadoop Framework distributes the jar on your behalf
- Needs to know which jar file to distribute
- The easiest way to specify the jar that your job resides in is by calling `job.setJarByClass`

```
job.setJarByClass(getClass());
```

- Hadoop will locate the jar file that contains the provided class

11

1: Configure Job - Specify Input

```
TextInputFormat.addInputPath(job, new Path(args[0]));  
job.setInputFormatClass(TextInputFormat.class);
```

- **Can be a file, directory or a file pattern**
 - Directory is converted to a list of files as an input
- **Input is specified by implementation of InputFormat - in this case TextInputFormat**
 - Responsible for creating splits and a record reader
 - Controls input types of key-value pairs, in this case LongWritable and Text
 - File is broken into lines, mapper will receive 1 line at a time

12

Side Node – Hadoop IO Classes

- **Hadoop uses it's own serialization mechanism for writing data in and out of network, database or files**
 - Optimized for network serialization
 - A set of basic types is provided
 - Easy to implement your own
- **org.apache.hadoop.io package**
 - LongWritable for Long
 - IntWritable for Integer
 - Text for String
 - Etc...

13

1: Configure Job - Specify Output

```
TextOutputFormat.setOutputPath(job, new Path(args[1]));  
job.setOutputFormatClass(TextOutputFormat.class);
```

- **OutputFormat defines specification for outputting data from Map/Reduce job**
- **Count job utilizes an implementation of OutputFormat - TextOutputFormat**
 - Define output path where reducer should place its output
 - If path already exists then the job will fail
 - Each reducer task writes to its own file
 - By default a job is configured to run with a single reducer
 - Writes key-value pair as plain text

14

1: Configure Job - Specify Output

```
job.setOutputKeyClass(Text.class);  
job.setOutputValueClass(IntWritable.class);
```

- **Specify the output key and value types for both mapper and reducer functions**
 - Many times the same type
 - If types differ then use
 - setMapOutputKeyClass()
 - setMapOutputValueClass()

15

1: Configure Job

- **Specify Mapper, Reducer and Combiner**
 - At a minimum will need to implement these classes
 - Mappers and Reducer usually have same output key

```
job.setMapperClass(StartsWithCountMapper.class);  
job.setReducerClass(StartsWithCountReducer.class);  
job.setCombinerClass(StartsWithCountReducer.class);
```

16

1: Configure Job

- **job.waitForCompletion(true)**
 - Submits and waits for completion
 - The boolean parameter flag specifies whether output should be written to console
 - If the job completes successfully 'true' is returned, otherwise 'false' is returned

17

Our Count Job is configured to

- **Chop up text files into lines**
- **Send records to mappers as key-value pairs**
 - Line number and the actual value
- **Mapper class is `StartsWithCountMapper`**
 - Receives key-value of `<IntWritable,Text>`
 - Outputs key-value of `<Text, IntWritable>`
- **Reducer class is `StartsWithCountReducer`**
 - Receives key-value of `<Text, IntWritable>`
 - Outputs key-values of `<Text, IntWritable>` as text
- **Combiner class is `StartsWithCountReducer`**

18

1: Configure Count Job

Provides Configuration support.
More on this later...

```
public class StartsWithCountJob extends Configured implements Tool{
    @Override
    public int run(String[] args) throws Exception {
        Job job = Job.getInstance(getConf(), "StartsWithCount");
        job.setJarByClass(getClass());

        // configure output and input source
        TextInputFormat.addInputPath(job, new Path(args[0]));
        job.setInputFormatClass(TextInputFormat.class);

        // configure mapper and reducer
        job.setMapperClass(StartsWithCountMapper.class);
        job.setCombinerClass(StartsWithCountReducer.class);
        job.setReducerClass(StartsWithCountReducer.class);
    }
}
```

...

...

19

StartsWithCountJob.java Continued...

```
...  
        // configure output  
        TextOutputFormat.setOutputPath(job, new Path(args[1]));  
        job.setOutputFormatClass(TextOutputFormat.class);  
        job.setOutputKeyClass(Text.class);  
        job.setOutputValueClass(IntWritable.class);  
  
        return job.waitForCompletion(true) ? 0 : 1;  
    }  
  
    public static void main(String[] args) throws Exception {  
        int exitCode = ToolRunner.run(  
            new StartsWithCountJob(), args);  
        System.exit(exitCode);  
    }  
}
```

Will need an actual java main that will execute the job. More on this later...

20

2: Implement Mapper class

- **Class has 4 Java Generics** parameters**
 - (1) input key (2) input value (3) output key (4) output value
 - Input and output utilizes hadoop's IO framework
 - org.apache.hadoop.io
- **Your job is to implement map() method**
 - Input key and value
 - Output key and value
 - Logic is up to you
- **map() method injects Context object, use to:**
 - Write output
 - Create your own counters

**Java Generics provide a mechanism to abstract Java types. To learn more visit <http://docs.oracle.com/javase/tutorial/extra/generics/index.html>

21

2: Implement Mapper

```
public class StartsWithCountMapper extends Mapper<LongWritable, Text, Text, IntWritable> {  
    private final static IntWritable countOne = new IntWritable(1);  
    private final Text reusableText = new Text();
```

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

```
        StringTokenizer tokenizer = new StringTokenizer(value.toString());  
        while (tokenizer.hasMoreTokens()) {  
            reusableText.set(tokenizer.nextToken().substring(0, 1));  
            context.write(reusableText, countOne);  
        }
```

```
    }  
}
```

Input key and value



Output key and value



22

3: Implement Reducer

- **Analogous to Mapper – generic class with four types**
 - (1) input key (2) input value (3) output key (4) output value
 - The output types of map functions must match the input types of reduce function
 - In this case Text and IntWritable
 - Map/Reduce framework groups key-value pairs produced by mapper by key
 - For each key there is a set of one or more values
 - Input into a reducer is sorted by key
 - Known as Shuffle and Sort
 - Reduce function accepts key->setOfValues and outputs key-value pairs
 - Also utilizes Context object (similar to Mapper)

23

3: Implement Reducer

```
public class StartsWithCountReducer extends
    Reducer<Text, IntWritable, Text, IntWritable> {

    @Override
    protected void reduce(Text token,
        Iterable<IntWritable> counts,
        Context context) throws IOException, InterruptedException {
        int sum = 0;

        for (IntWritable count : counts) {
            sum += count.get();
        }
        context.write(token, new IntWritable(sum));
    }
}
```

Input key and a set of corresponding values

Produce key-value pairs

24

3: Reducer as a Combiner

- **Combine data per Mapper task to reduce amount of data transferred to reduce phase**
- **Reducer can very often serve as a combiner**
 - Only works if reducer's output key-value pair types are the same as mapper's output types
- **Combiners are not guaranteed to run**
 - Optimization only
 - Not for critical logic
- **More about combiners later**

25

4: Run Count Job

```
$ yarn jar $PLAY_AREA/HadoopSamples.jar \
  mr.wordcount.StartsWithCountJob \
  /training/data/hamlet.txt \
  /training/playArea/wordCount/
. . . .
. . . .
2012-05-15 18:03:25,372 INFO mapreduce.Job (Job.java:submit(1225)) - The url to track the job: http://hadoop-
  laptop:8088/proxy/application_1336894075975_0011/
2012-05-15 18:03:25,373 INFO mapreduce.Job (Job.java:monitorAndPrintJob(1270)) - Running job:
  job_1336894075975_0011
2012-05-15 18:03:31,939 INFO mapreduce.Job (Job.java:monitorAndPrintJob(1291)) - Job job_1336894075975_0011
  running in uber mode : false
2012-05-15 18:03:31,941 INFO mapreduce.Job (Job.java:monitorAndPrintJob(1298)) - map 0% reduce 0%
2012-05-15 18:03:45,056 INFO mapreduce.Job (Job.java:monitorAndPrintJob(1298)) - map 33% reduce 0%
2012-05-15 18:03:48,082 INFO mapreduce.Job (Job.java:monitorAndPrintJob(1298)) - map 38% reduce 0%
2012-05-15 18:03:57,131 INFO mapreduce.Job (Job.java:monitorAndPrintJob(1298)) - map 52% reduce 0%
2012-05-15 18:04:00,177 INFO mapreduce.Job (Job.java:monitorAndPrintJob(1298)) - map 72% reduce 0%
2012-05-15 18:04:03,194 INFO mapreduce.Job (Job.java:monitorAndPrintJob(1298)) - map 72% reduce 16%
2012-05-15 18:04:09,230 INFO mapreduce.Job (Job.java:monitorAndPrintJob(1298)) - map 78% reduce 16%
2012-05-15 18:04:12,244 INFO mapreduce.Job (Job.java:monitorAndPrintJob(1298)) - map 82% reduce 16%
2012-05-15 18:04:21,292 INFO mapreduce.Job (Job.java:monitorAndPrintJob(1298)) - map 83% reduce 16%
2012-05-15 18:04:22,312 INFO mapreduce.Job (Job.java:monitorAndPrintJob(1298)) - map 100% reduce 16%
2012-05-15 18:04:23,324 INFO mapreduce.Job (Job.java:monitorAndPrintJob(1298)) - map 100% reduce 100%
2012-05-15 18:04:23,329 INFO mapreduce.Job (Job.java:monitorAndPrintJob(1309)) - Job job_1336894075975_0011
  completed successfully
2012-05-15 18:04:23,464 INFO mapreduce.Job (Job.java:monitorAndPrintJob(1316)) - Counters: 44
File System Counters
FILE: Number of bytes read=1010922
FILE: Number of bytes written=1494114
FILE: Number of read operations=0
FILE: Number of large read operations=0
FILE: Number of write operations=0
HDFS: Number of bytes read=97833472
. . . .
```

26

Output From Your Job

- **Provides job id**

```
INFO mapreduce.Job (Job.java:monitorAndPrintJob(1270)) -
Running job: job_1337387252832_0002
```

- Used to identify, monitor and manage the job

- **Shows number of generated splits**

```
mapreduce.JobSubmitter (JobSubmitter.java:submitJobInternal(362))
- number of splits:1
```

- **Reports the Progress**

```
INFO mapreduce.Job (Job.java:monitorAndPrintJob(1298)) - map 0% reduce 0%
INFO mapreduce.Job (Job.java:monitorAndPrintJob(1298)) - map 100% reduce 0%
```

- **Displays Counters – statistics for the job**

- Sanity check that the numbers match what you expected

27

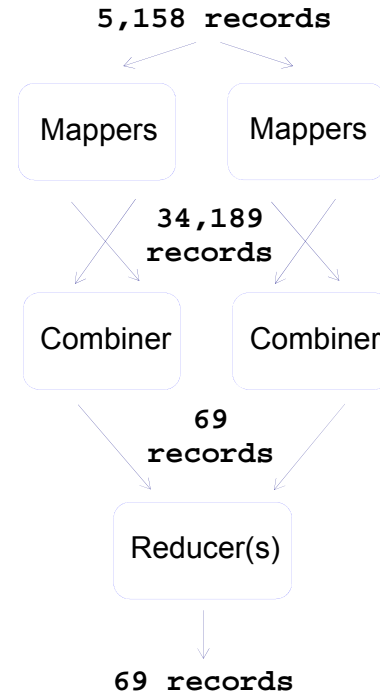
4: Run Count Job - Continued

...
...

Map-Reduce Framework

```
Map input records=5158
Map output records=34189
Map output bytes=205134
Map output materialized bytes=558
Input split bytes=115
Combine input records=34189
Combine output records=69
Reduce input groups=69
Reduce shuffle bytes=558
Reduce input records=69
Reduce output records=69
Spilled Records=138
Shuffled Maps =1
Failed Shuffles=0
Merged Map outputs=1
GC time elapsed (ms)=62
```

...
...



28

Output of Count Job

```
$ hdfs dfs -cat /training/playArea/wordCount/part-r-00000 | more
"      8
#      1
&      1
'     185
(     86
*     16
-      1
.      4
/      1
1     29
2      7
3      2
4      1
6      2
9      1
<     12
?      2
A     722
B     325
...
```

- **Output is written to the configured output directory**
 - /training/playArea/wordCount/
- **One output file per Reducer**
 - part-r-xxxxx format
- **Output is driven by TextOutputFormat class**

29

\$yarn command

- **yarn script with a class argument command launches a JVM and executes the provided Job**

```
$ yarn jar $PLAY_AREA/HadoopSamples.jar \  
    mr.wordcount.StartsWithCountJob    \  
    /training/playArea/hamlet.txt      \  
    /training/playArea/wordCount/
```

- **You could use straight java but yarn script is more convenient**
 - Adds hadoop's libraries to CLASSPATH
 - Adds hadoop's configurations to Configuration object
 - Ex: core-site.xml, mapred-site.xml, *.xml
 - You can also utilize \$HADOOP_CLASSPATH environment variable

30

© 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

- **In this lecture we**
 - Wrote a MapReduce Job
 - Implemented Map and Reduce Functions
 - Executed the job and analyzed the output

32

© 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.