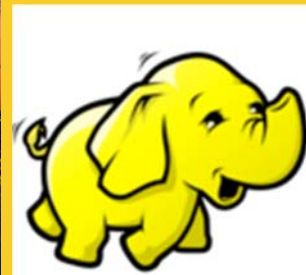




# Map Reduce Workflows

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](mailto: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](http://coreservlets.com) experts (edited by Marty)
  - Spring, Hibernate/JPA, GWT, Hadoop, HTML5, RESTful Web Services

Contact [info@coreservlets.com](mailto:info@coreservlets.com) for details



# Agenda

- **Workflows Introduction**
- **Decomposing Problems into MapReduce Workflow**
- **Using JobControl class**

4

# MapReduce Workflows

- **We've looked at single MapReduce job**
- **Complex processing requires multiple steps**
  - Usually manifest in multiple MapReduce jobs rather than complex map and reduce functions
- **May also want to consider higher-level MapReduce abstractions**
  - Pig, Hive, Cascading, Cascalog, Crunch
  - Focus on business logic rather than MapReduce translation
  - On the other hand you'll need to learn another syntax and methodology
- **This lecture will focus on building MapReduce Workflows**

5

# Decomposing Problems into MapReduce Jobs

- **Small map-reduce jobs are usually better**
  - Easier to implement, test and maintain
  - Easier to scale and re-use
- **Problem:**
- **Find a letter that occurs the most in the provided body of text**

6

# Decomposing the Problem

- **Calculate number of occurrences of each letter in the provided body of text**
- **Traverse each letter comparing occurrence count**
- **Produce start letter that has the most occurrences**

(For so this side of our known world esteem'd him)  
Did slay this Fortinbras; who, by a seal'd compact,  
Well ratified by law and heraldry,  
Did forfeit, with his life, all those his lands  
Which he stood seiz'd of, to the conqueror;  
Against the which a moiety competent  
Was gaged by our king; which had return'd  
To the inheritance of Fortinbras,

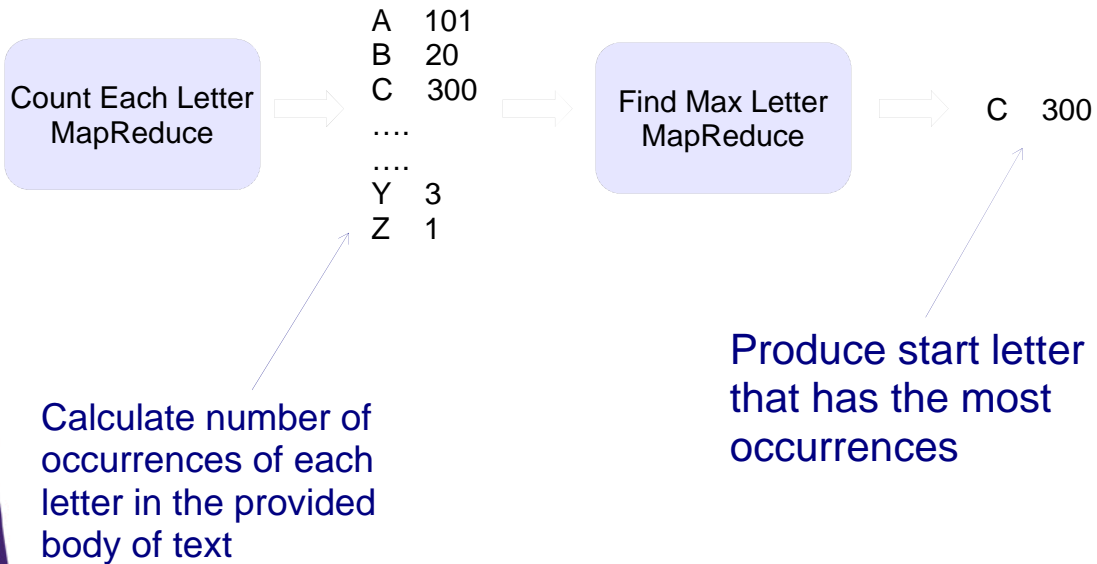
A vertical line separates the text from the table. A blue arrow points from the text to the table. Another blue arrow points from the table to the final result 'T'.

A	89530
B	3920
..	
..	
Z	876

T 495959

7

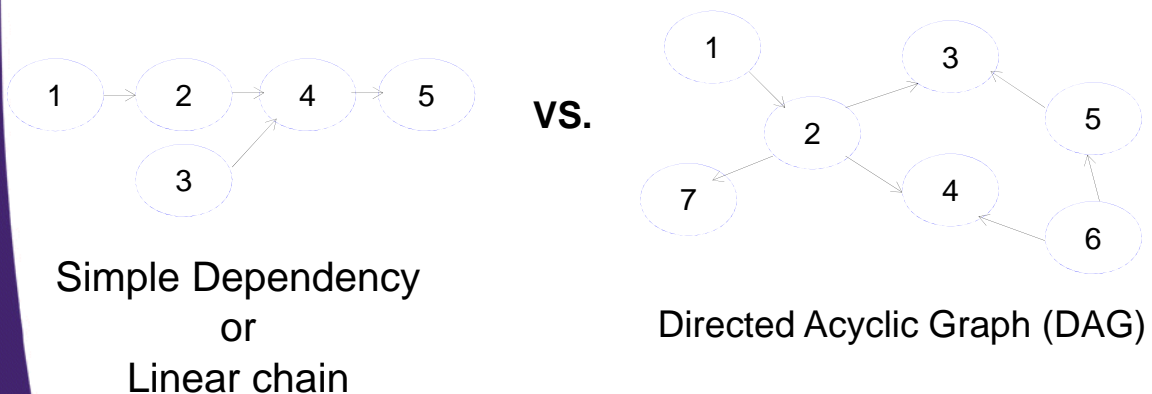
# Decomposing the Problem



8

# MapReduce Workflows

- **Your choices can depend on complexity of workflows**
  - Linear chain or simple set of dependent Jobs vs. Directed Acyclic Graph (DAG)
  - [http://en.wikipedia.org/wiki/Directed\\_acyclic\\_graph](http://en.wikipedia.org/wiki/Directed_acyclic_graph)



9

# MapReduce Workflows

- **JobControl class**
  - Create simple workflows
  - Represents a graph of Jobs to run
  - Specify dependencies in code
- **Oozie**
  - An engine to build complex DAG workflows
  - Runs in its own daemon
  - Describe workflows in set of XML and configuration files
  - Has coordinator engine that schedules workflows based on time and incoming data
  - Provides ability to re-run failed portions of the workflow

10

# Workflow with JobControl

- 1. Create JobControl**
  - Implements `java.lang.Runnable`, will need to execute within a Thread
- 2. For each Job in the workflow Construct ControlledJob**
  - Wrapper for Job instance
  - Constructor takes in dependent jobs
- 3. Add each ControlledJob to JobControl**
- 4. Execute JobControl in a Thread**
  - Recall JobControl implements Runnable
- 5. Wait for JobControl to complete and report results**
  - Clean-up in case of a failure

11

# 1: Create JobControl

```
public class MostSeenStartLetterJobControl
    extends Configured implements Tool{

    private final Logger log =
        Logger.getLogger(MostSeenStartLetterJobControl.class);

    @Override
    public int run(String[] args) throws Exception {
        String inputText = args[0];
        String finalOutput = args[1];

        String intermediateTempDir = "/" +
            getClass().getSimpleName() + "-tmp";
        Path intermediatePath = new Path(intermediateTempDir);
        deleteIntermediateDir(intermediatePath);

        try {
            JobControl control =
                new JobControl("Worklfow-Example");
            ...
            ...
        }
    }
}
```

Manage intermediate output directory

JobControl manages workflow

12

# 2: For Each Job in the Workflow Construct ControlledJob

```
...
...
ControlledJob step1 = new ControlledJob(
    getCountJob(inputText, intermediateTempDir), null);

ControlledJob step2 = new ControlledJob(
    getMostSeenJob(intermediateTempDir, finalOutput),
    Arrays.asList(step1));
...
...
```

Returns Job object with properly configured job

Specify dependencies, In this case step2 depends on step1

13

## 2: Build First Job

```
private Job getCountJob(String inputText, String tempOutputPath)
throws IOException {
    Job job = Job.getInstance(getConf(), "StartsWithCount");
    job.setJarByClass(getClass());

    // configure output and input source
    TextInputFormat.addInputPath(job, new Path(inputText));
    job.setInputFormatClass(TextInputFormat.class);

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

    // configure output
    TextOutputFormat.setOutputPath(job, new Path(tempOutputPath));
    job.setOutputFormatClass(TextOutputFormat.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);

    return job;
}
```

Build each job the same way we've done before

14

## 2: Build Second Job

```
private Job getMostSeenJob(String intermediateTempDir,
String finalOutput) throws IOException {
    Job job = Job.getInstance(getConf(), "MostSeenStartLetter");
    job.setJarByClass(getClass());

    // configure output and input source
    KeyValueTextInputFormat.addInputPath(job,
new Path(intermediateTempDir));
    job.setInputFormatClass(KeyValueTextInputFormat.class);

    // configure mapper and reducer
    job.setMapperClass(MostSeenStartLetterMapper.class);
    job.setCombinerClass(MostSeendStartLetterReducer.class);
    job.setReducerClass(MostSeendStartLetterReducer.class);

    // configure output
    TextOutputFormat.setOutputPath(job, new Path(finalOutput));
    job.setOutputFormatClass(TextOutputFormat.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);

    return job;
}
```

Build each job the same way we've done before

15

## 3: Add Each ControlledJob to JobControl

```
...  
...  
control.addJob(step1);  
control.addJob(step2);  
...  
...
```

16

## 4: Execute JobControl in a Thread

```
...  
...  
Thread workflowThread = new Thread(control,  
    "Workflow-Thread");  
  
workflowThread.setDaemon(true);  
workflowThread.start();  
  
...  
...
```

17



## 5: Wait for JobControl to Complete

```
...
...
while (!control.allFinished()){
    Thread.sleep(500);
}

if ( control.getFailedJobList().size() > 0 ){
    log.error(control.getFailedJobList().size() +
        " jobs failed!");

    for ( ControlledJob job : control.getFailedJobList()){
        log.error(job.getJobName() + " failed");
    }
} else {
    log.info("Success!! Workflow completed [" +
        control.getSuccessfulJobList().size() + "] jobs");
}
...
...
```

Simply wait for all jobs to complete

Report results or Display errors

18

## Run JobControl Example

```
yarn jar $PLAY_AREA/HadoopSamples.jar \
mr.workflows.MostSeenStartLetterJobControl \
/training/data/hamlet.txt \
/training/playArea/wordCount
```



### All Applications

Logged in as: dr.who

#### Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes
2	0	0	2	0	0 KB	8 GB	0 KB	1	0	0	0	0

Show 20 entries

ID	User	Name	Queue	StartTime	FinishTime	State	FinalStatus	Progress	Tracking UI
<a href="#">application_1342060355099_0001</a>	hadoop	StartsWithCount	default	11-Jul-2012 22:32:53	11-Jul-2012 22:33:16	FINISHED	SUCCEEDED	<div style="width: 100%;"></div>	<a href="#">History</a>
<a href="#">application_1342060355099_0002</a>	hadoop	MostSeenStartLetter	default	11-Jul-2012 22:33:14	11-Jul-2012 22:33:33	FINISHED	SUCCEEDED	<div style="width: 100%;"></div>	<a href="#">History</a>

Showing 1 to 2 of 2 entries

First Previous 1 Next Last

Two Jobs got executed

19

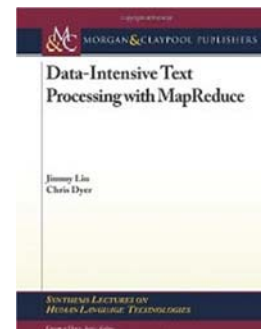
# JobControl Workflow Result

```
$ hdfs dfs -cat /training/playArea/wordCount/part-r-00000  
t      3711
```

20

# Study MapReduce Algorithms

- “Data-Intensive Text Processing with MapReduce” by Jimmy Lin and Chris Dyer
- Download the book for free
  - <http://lintool.github.com/MapReduceAlgorithms/index.html>
- Buy a copy
  - [http://www.amazon.com/Data-Intensive-Processing-MapReduce-Synthesis-Technologies/dp/1608453421/ref=sr\\_1\\_1?ie=UTF8&qid=1340464379&sr=8-1&keywords=Data-Intensive+Text+Processing+with+MapReduce](http://www.amazon.com/Data-Intensive-Processing-MapReduce-Synthesis-Technologies/dp/1608453421/ref=sr_1_1?ie=UTF8&qid=1340464379&sr=8-1&keywords=Data-Intensive+Text+Processing+with+MapReduce)



21

# Study MapReduce Patterns



## MapReduce Design Patterns

Donald Miner (Author), Adam Shook (Author)  
O'Reilly Media (November 22, 2012)

22

© 2012 [coreservlets.com](http://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**
  - Decompose Problems into MapReduce Workflow
  - Utilize JobControl to implement MapReduce workflow

24

© 2012 [coreservlets.com](http://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.