

## **Module 4**

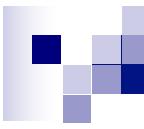
# **HBase, MapReduce and Oozie**

Thanachart Numnonda, Executive Director, IMC Institute

Mr. Aekanun Thongtae, Big Data Consultant, IMC Institute

Thanisa Numnonda, Faculty of Information Technology,

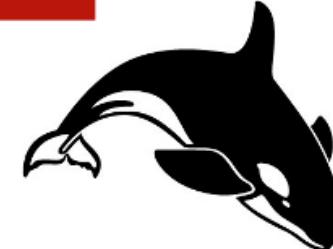
King Mongkut's Institute of Technology Ladkrabang



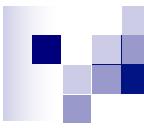
# Introduction

An open source, non-relational, distributed database

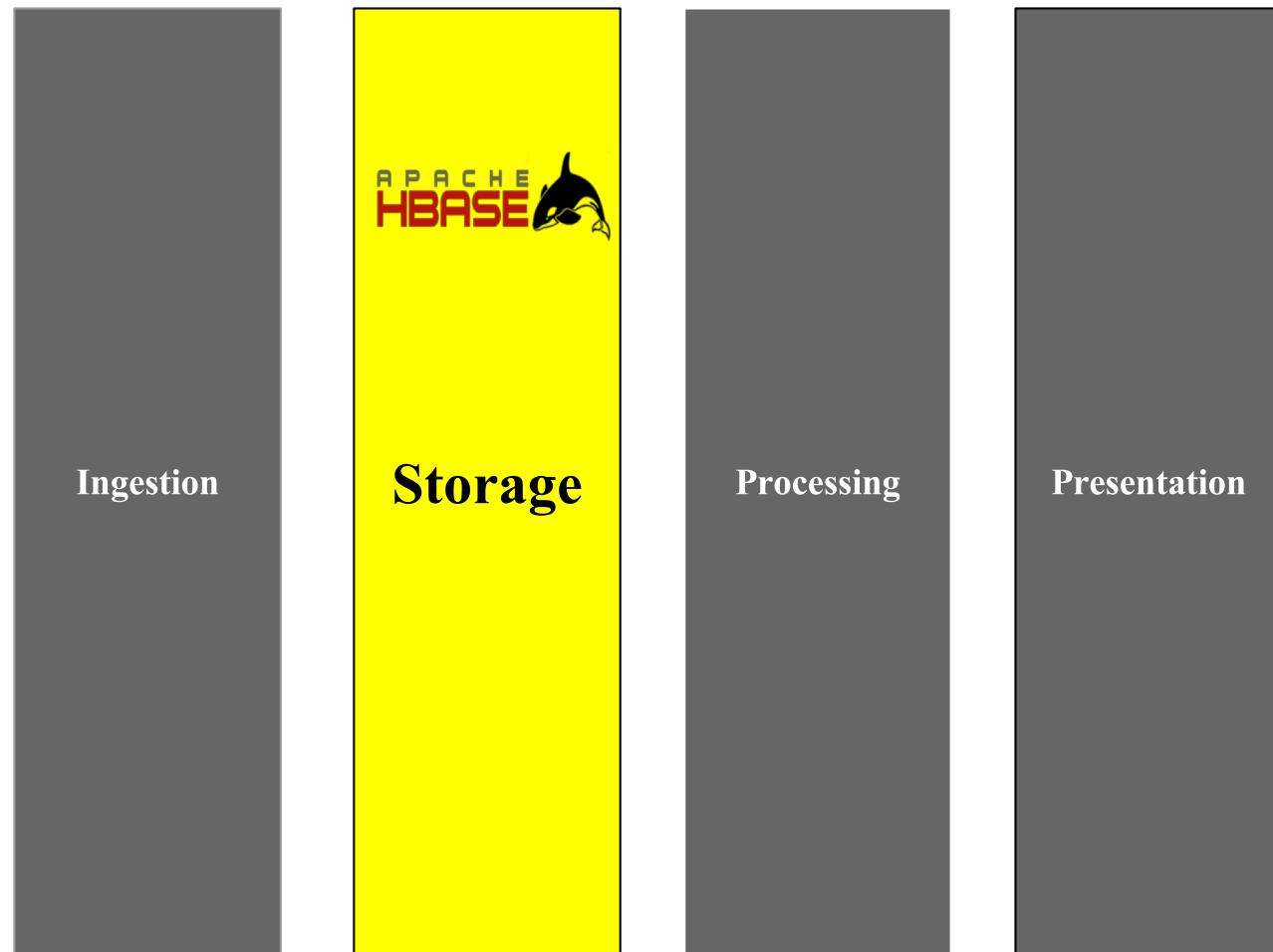
A P A C H E  
**H<sup>BASE</sup>**

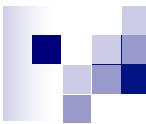


- Apache HBase™ is the Hadoop database, a distributed, scalable, big data store.
- Use Apache HBase™ when you need random, realtime read/write access to your Big Data.
- Goal is the hosting of very large tables -- billions of rows X millions of columns



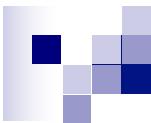
# Key Infrastructure Elements





# HBase Features

- Hadoop database modelled after Google's Bigtable
- Column oriented data store, known as Hadoop Database
- Support random realtime CRUD operations (unlike HDFS)
- NoSQL Database
- Open source, written in Java
- Run on a cluster of commodity hardware



## Data Model in HBase

- Column-oriented database: the tables are sorted by row
  - Tables
    - Table is a collection of rows.
    - Row is a collection of column families.
    - Column family is a collection of columns.
    - Column is a collection of key value pairs.

- Given this RDBMS:

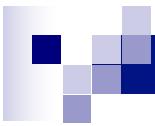
ID (Primary key)	Last name	First name	Password	Timestamp
1234	Smith	John	Hello, world!	20130710
5678	Cooper	Joyce	wysiwyg	20120825
5678	Cooper	Joyce	wisigig	20130916

- Logical view in HBase:

Row-Key	Value (CF, Qualifier, Version)
1234	info {'lastName': 'Smith', 'firstName': 'John'} pwd {'password': 'Hello, world!'} 
5678	info {'lastName': 'Cooper', 'firstName': 'Joyce'} pwd {'password': 'wysiwyg'@ts 20130916, 'password': 'wisigig'@ts 20120825}

## HBase vs. RDBMS

	HBase	RDBMS
Hardware architecture	Similar to Hadoop. Clustered commodity hardware. Very affordable.	Typically large scalable multiprocessor systems. Very expensive.
Fault Tolerance	Built into the architecture. Lots of nodes means each is relatively insignificant. No need to worry about individual node downtime.	Requires configuration of the HW and the RDBMS with the appropriate high availability options.
Typical Database Size	Terabytes to Petabytes - hundred of millions to billions of rows.	Gigabytes to Terabytes – hundred of thousands to millions of rows.
Data Layout	A sparse, distributed, persistent, multidimensional sorted map.	Rows or column oriented.
Data Types	Bytes only.	Rich data type support.
Transactions	ACID support on a single row only	Full ACID compliance across rows and tables
Query Language	API primitive commands only, unless combined with Hive or other technology	SQL
Indexes	Row Key only unless combined with other technologies such as Hive or IBM's BigSQL	Yes
Throughput	Millions of queries per second	Thousands of queries per second



# HBase Shell Commands

- List all tables:

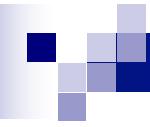
```
hbase (main) :001:0> list
```

- Create tables:

```
hbase (main) :002:0> create 'testTable', 'cf'
```

- Insert data into a table:

```
hbase (main) :003:0> put 'testTable', 'rowA', 'cf:columnName', 'val1'
```



# HBase Shell Commands (Cont.)

- Retrieve data from a table:

```
hbase(main):009:0> get 'testTable', 'rowA'
```

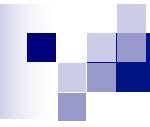
- Iterate through a table:

```
hbase(main):011:0> scan 'testTable'
```

- Delete a table:

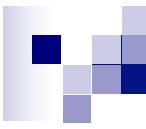
```
hbase(main):012:0> disable 'testTable'
```

```
hbase(main):013:0> drop 'testTable'
```



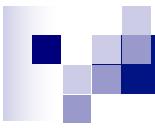
# Which one to use?

- **HDFS**
  - Only append dataset (no random write)
  - Read the whole dataset (no random read)
- **HBase**
  - Need random write and/or read
  - Has thousands of operation per second on TB+ of data
- **RDBMS**
  - Data fits on one big node
  - Need full transaction support
  - Need real-time query capabilities



# **Hands-On: Running HBase**

---



# Hbase shell

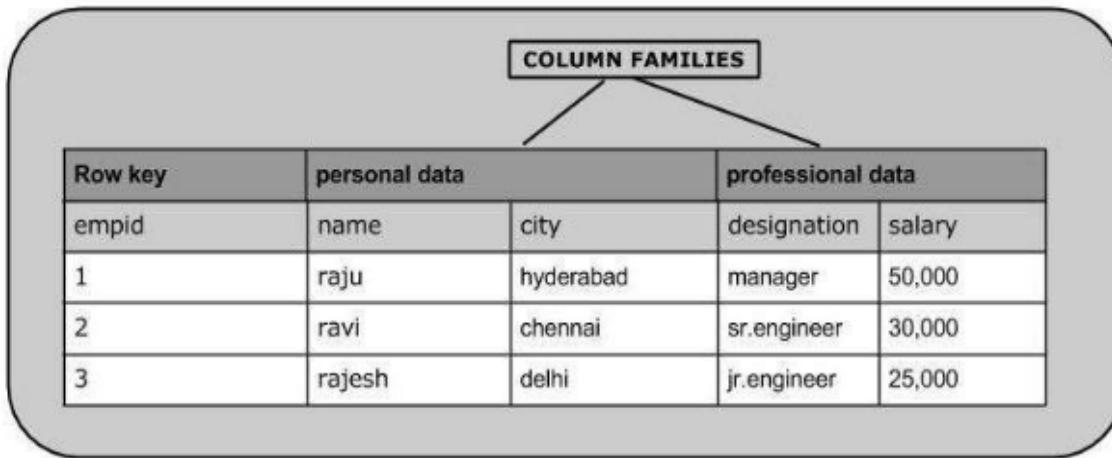
Row key	personal data	professional data

```
[root@quickstart /]# hbase shell  
hbase(main):001:0> create 'employee', 'personal  
data','professional data'
```

```
hbase(main):002:0> list
```

```
TABLE  
employee  
1 row(s) in 0.0310 seconds
```

# Create Data



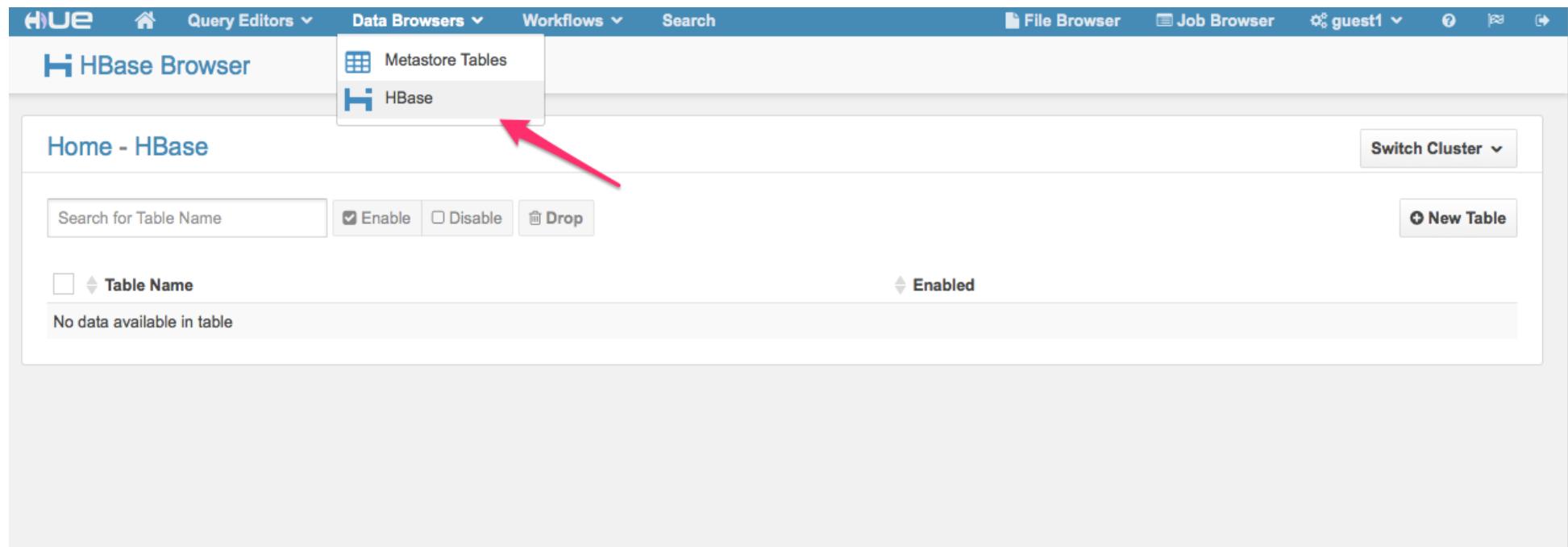
```
hbase(main):023:0> put 'employee','1','personal data:name',  
'raju'
```

```
hbase(main):024:0> put 'employee','1','personal data:city',  
'hyderabad'
```

```
hbase(main):025:0> put 'employee','1','professional  
data:designation','manager'
```

```
hbase(main):026:0> put 'employee','1','professional  
data:salary','50000'
```

# Running HBase Browser



# Viewing Employee Table

The screenshot shows the Hue HBase Browser interface. At the top, there's a navigation bar with links for Home, Query Editors, Data Browsers, Workflows, Search, and Security. Below the navigation bar, the title "HBase Browser" is displayed. The main area is titled "Home - Cluster". On the left, there's a search bar labeled "Search for Table Name" with dropdown options for "Enable", "Disable", and "Drop". To the right of the search bar are buttons for "New Table" and "Switch Cluster". Below the search bar, there's a table listing tables with columns for "Table Name" and "Enabled". The "employee" table is listed with its "Enabled" status checked. On the far right, there are filter and sort controls.

This screenshot shows a detailed view of the "employee" table row from the previous interface. The top navigation bar remains the same. The main title is now "Home - Cluster / employee". The search bar at the top includes filters for "personal data:" and "professional data:". Below the search bar, there are buttons for "Filter Columns/Families", "All", and "Sort By ASC". The data is presented in a grid format. The first row has a red border and contains the number "1". The columns are labeled "personal data: city", "personal data: name", "professional data: designation", and "professional data: salary". The values for the "city" column are "hyderabad", for "name" are "raju", for "designation" are "manager", and for "salary" are "50000".

# Create a table in HBase

HBase Browser

Home - HBase

Switch Cluster ▾

Search for Table Name  Enable  Disable  Drop

Table Name  Enabled

New Table



## Create New Table

Table Name:

Student

Column Families:

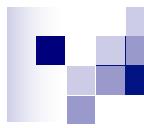
cf

Add a column property

Add an additional column family

Cancel

Submit



# Insert a new row in a table

HBase Browser

Home - HBase / Student

Switch Cluster ▾

row\_key, row\_prefix\* +scan\_len [col1, family:col2, fam3:, col\_prefix\* +3, fam:  cf:  Filter Columns/Families  All Sort By DESC ▾

No rows to display.

Fetched 123 in 0.374 seconds.



# Add field into a new row

## Insert New Row

Row Key

123

cf:firstname

Thanachart

cf:lastname

Numnonda

 Add Field

Cancel

Submit

## Insert New Row

Row Key

124

cf:firstname

Somchai

 Add Field

Cancel

Submit

Home - HBase / Student

Switch Cluster ▾

row\_key, row\_prefix\* +scan\_len [col1, family:col2, fam3:, col\_prefix\* +3, fam:  ct:



Filter Columns/Families

All

Sort By DESC ▾

124

cf: firstname

Somchai

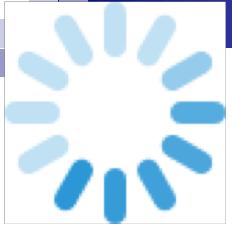
123

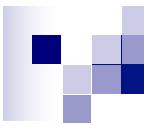
cf: firstname

cf: lastname

Thanachart

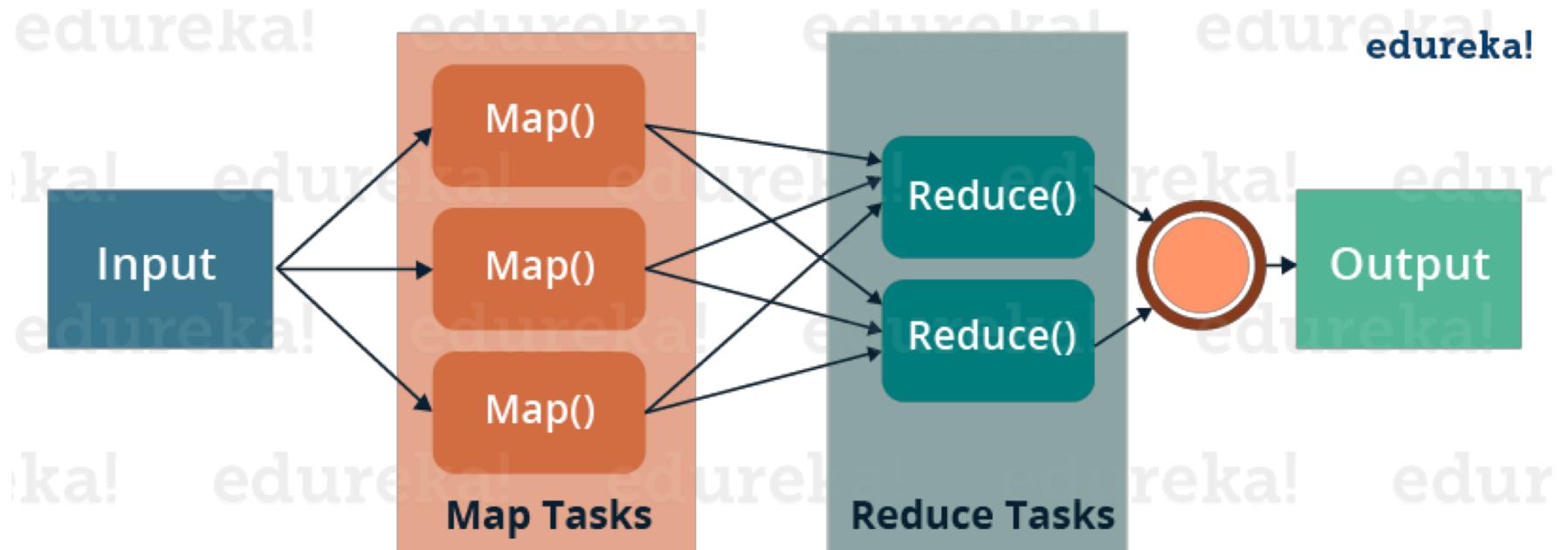
Numnonda



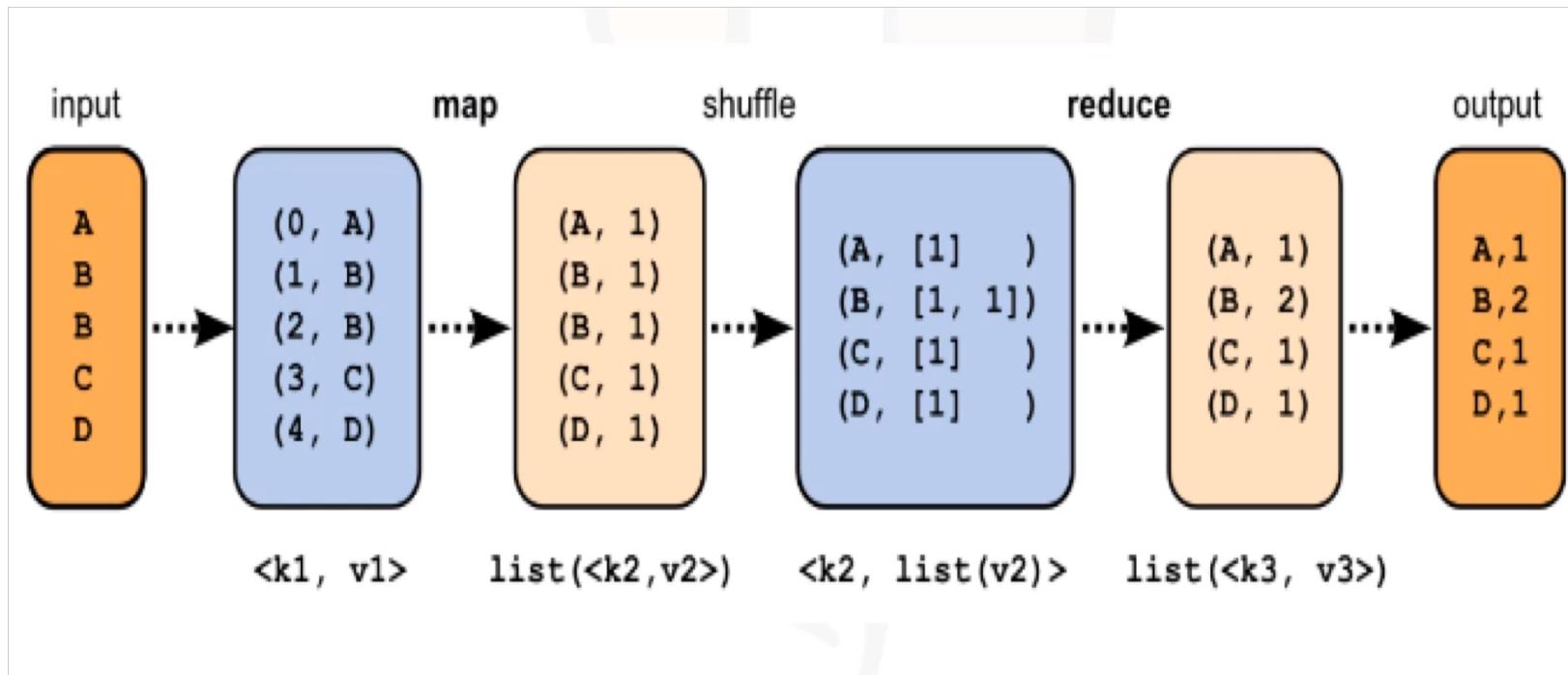


# MapReduce Framework

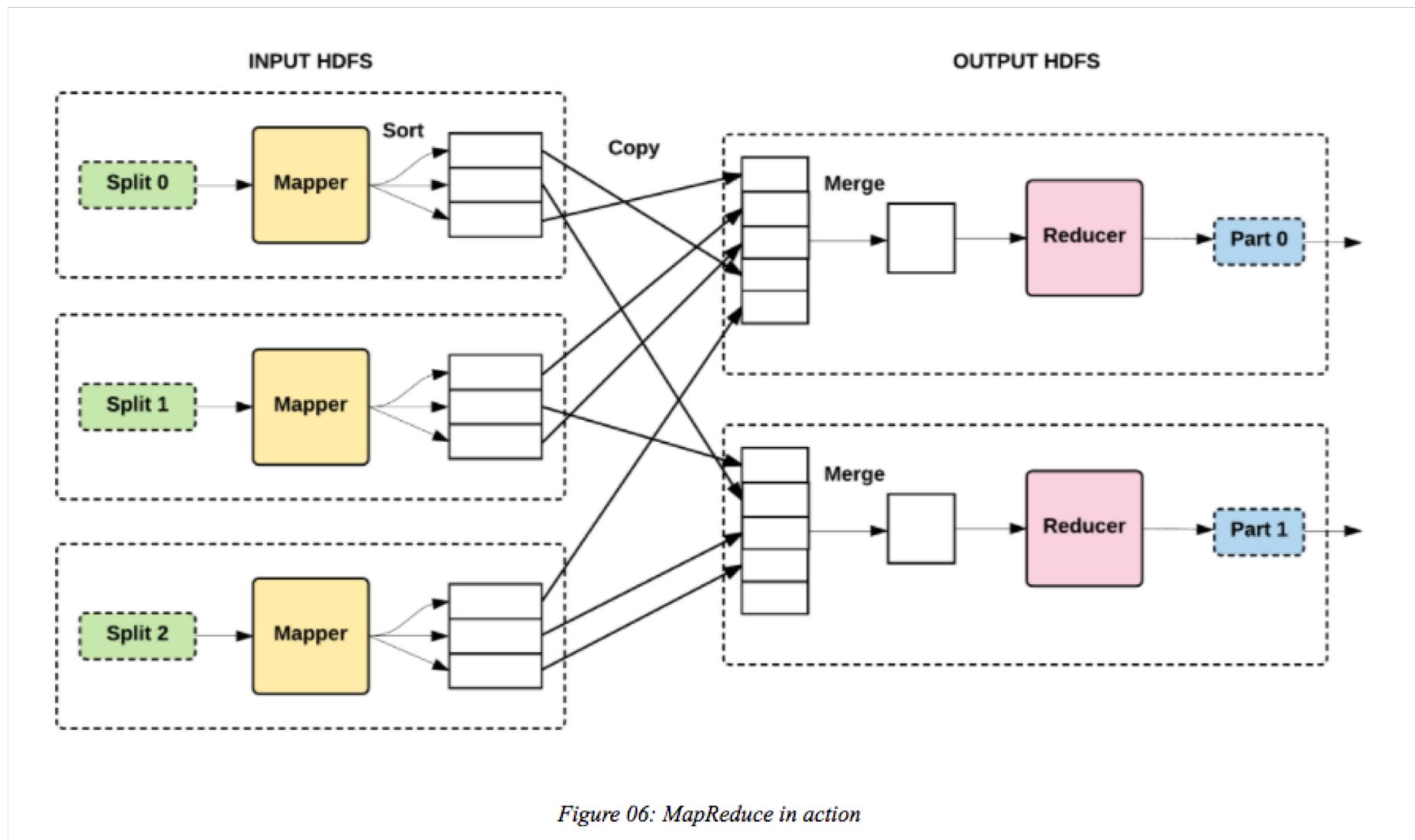
- MapReduce is a programming framework that allows us to perform distributed and parallel processing on large data sets in a distributed environment.



# MapReduce Framework



# MapReduce-Architecture View

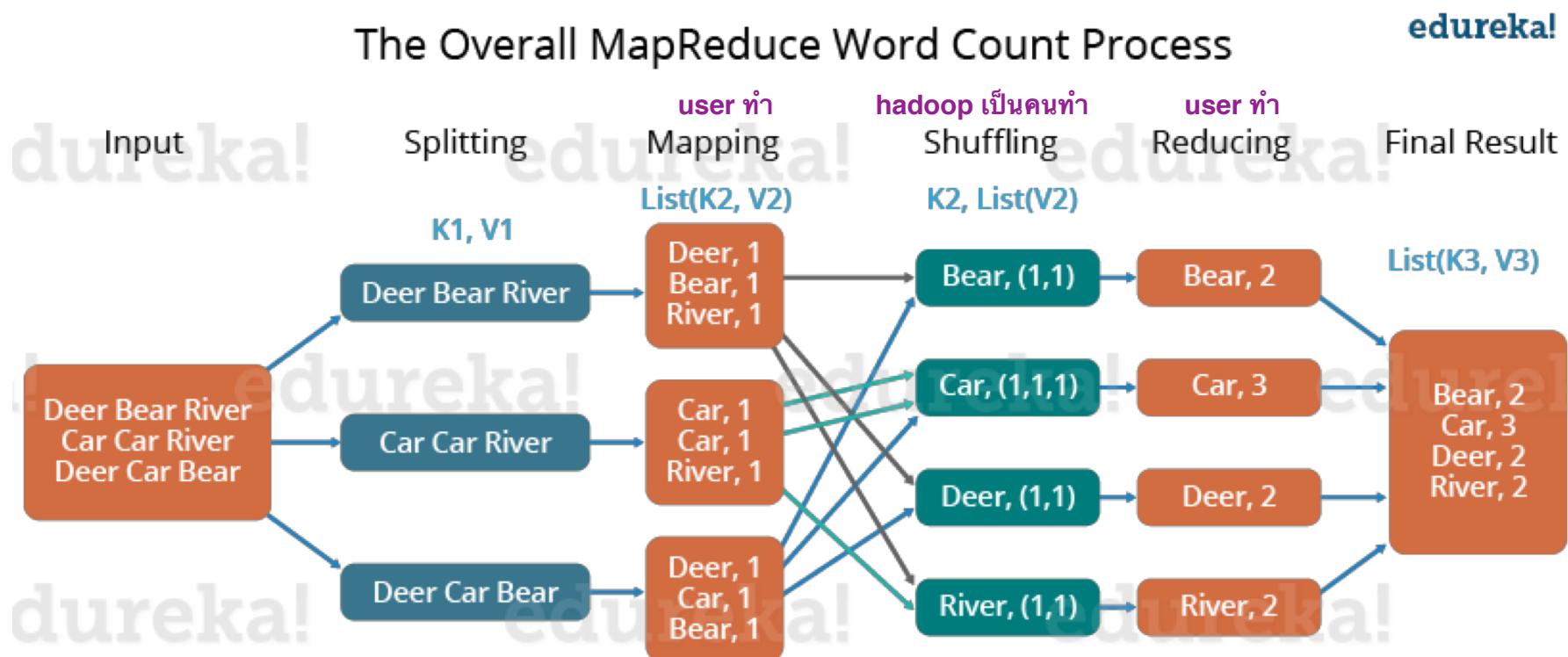


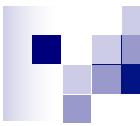
# MapReduce Tutorial: A Word Count Example of MapReduce

- Text file: example.txt

*Dear, Bear, River, Car, Car, River, Deer, Car and Bear*

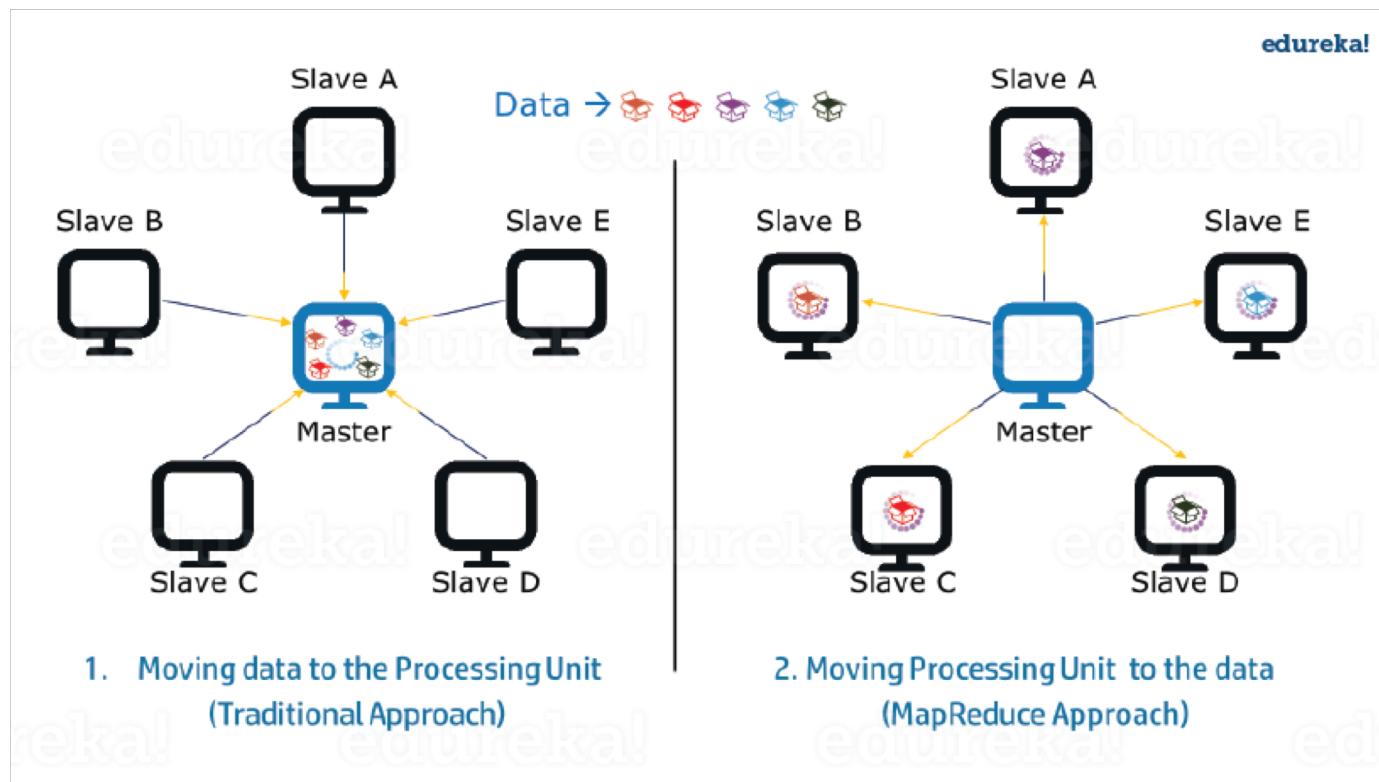
- Finding: number of occurrences of those unique words

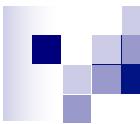




# MapReduce

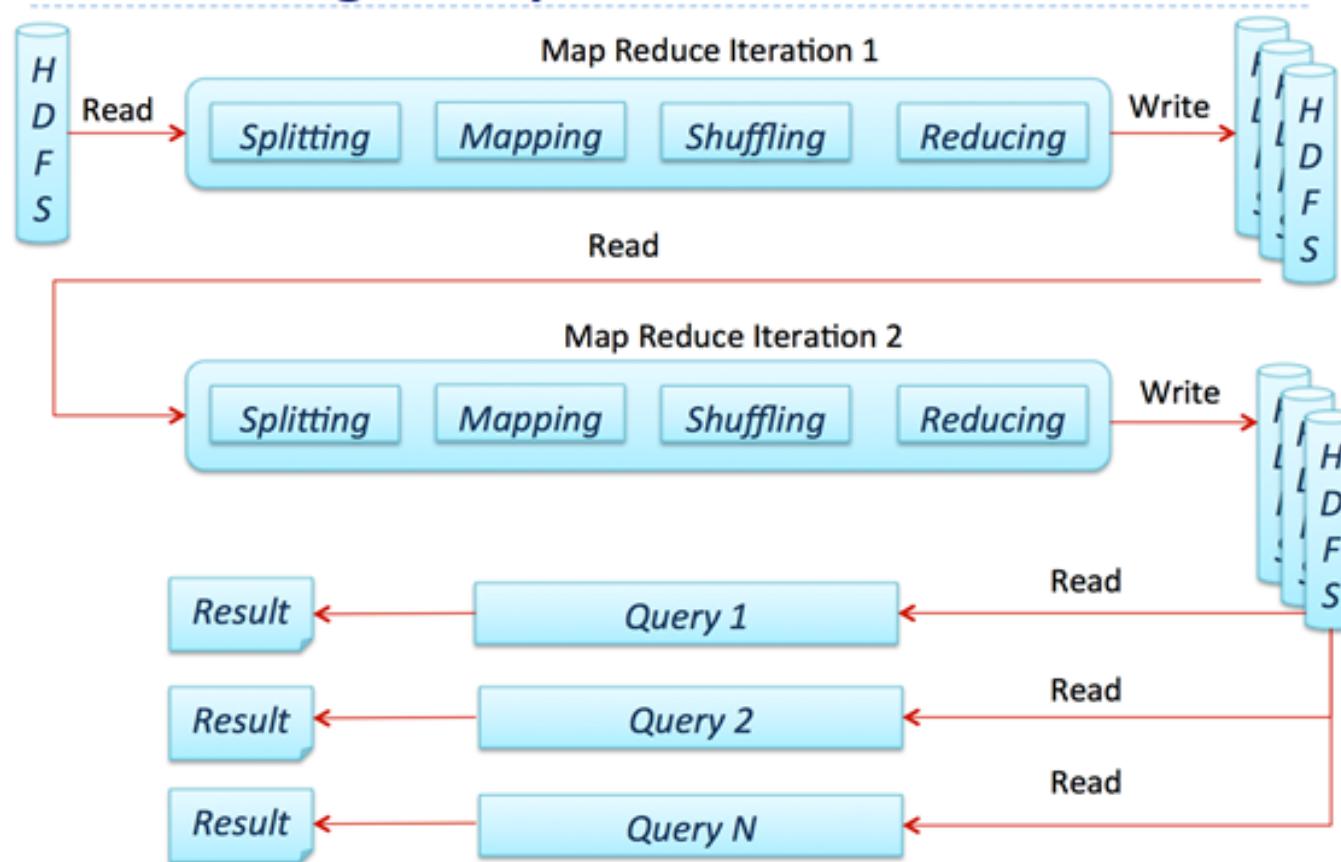
- Parallel Processing
  - The time taken to process the data gets reduced by a tremendous amount
- Data Locality
  - Instead of moving data to the processing unit, we are moving processing unit to the data



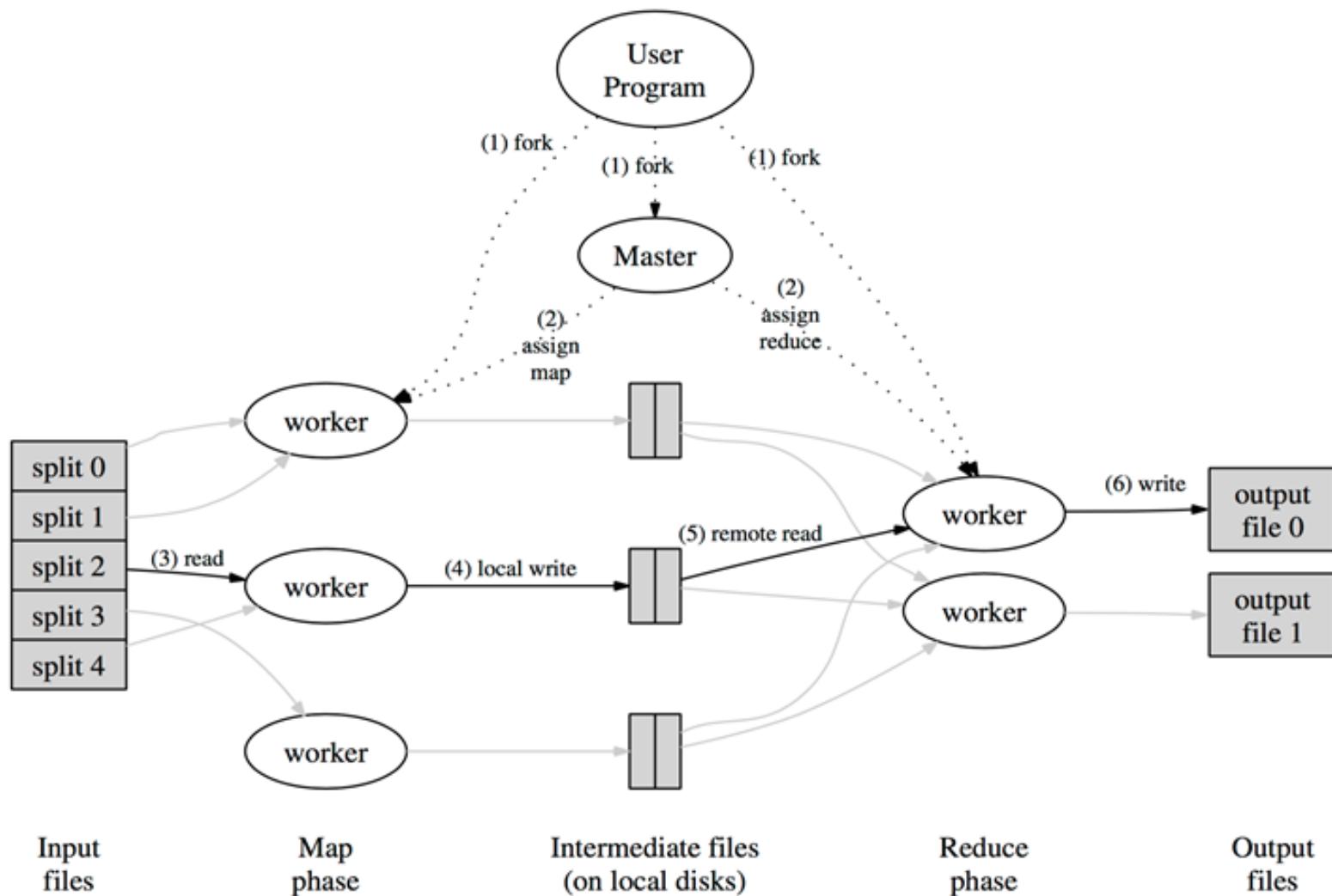


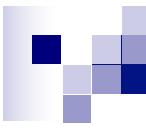
# Typical Hadoop Job: Map-Reduce

## Data Sharing in Map Reduce



# Typical Hadoop Job: Map-Reduce (Cont.)





# **Hands-On: Writing you MapReduce Program**

---

# Example MapReduce: WordCount

```
package org.apache.hadoop.examples;

import java.io.IOException;
import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;

public class WordCount {

    public static class TokenizerMapper
        extends Mapper<Object, Text, Text, IntWritable>{

        private final static IntWritable one = new IntWritable(1);
        private Text word = new Text();

        public void map(Object key, Text value, Context context
                        ) throws IOException, InterruptedException {
            StringTokenizer itr = new StringTokenizer(value.toString());
            while (itr.hasMoreTokens()) {
                word.set(itr.nextToken());
                context.write(word, one);
            }
        }
    }
}
```

```
public static class IntSumReducer
    extends Reducer<Text,IntWritable,Text,IntWritable> {
private IntWritable result = new IntWritable();

    public void reduce(Text key, Iterable<IntWritable> values,
                      Context context
                      ) throws IOException, InterruptedException {
        int sum = 0;
        for (IntWritable val : values) {
            sum += val.get();
        }
        result.set(sum);
        context.write(key, result);
    }
}

public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    String[] otherArgs = new GenericOptionsParser(conf, args).getRemainingArgs();
    if (otherArgs.length != 2) {
        System.err.println("Usage: wordcount <in> <out>");
        System.exit(2);
    }
    Job job = new Job(conf, "word count");
    job.setJarByClass(WordCount.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(otherArgs[0]));
    FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
}
```

# Running MapReduce Program

```
[cloudera@quickstart ~]$ cd workspace/  
[cloudera@quickstart workspace]$ hadoop jar wordcount.jar org.myorg.WordCount input/* output/wordcount_output  
15/02/08 10:30:31 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032  
15/02/08 10:30:32 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032  
15/02/08 10:30:33 WARN mapreduce.JobSubmitter: Hadoop command-line option parsing not performed. Implement the Tool interface with ToolRunner to remedy this.  
15/02/08 10:30:33 INFO mapred.FileInputFormat: Total input paths to process : 1  
15/02/08 10:30:34 INFO mapreduce.JobSubmitter: number of splits:2  
15/02/08 10:30:34 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1423408479621_0009  
15/02/08 10:30:35 INFO impl.YarnClientImpl: Submitted application application_1423408479621_0009  
15/02/08 10:30:35 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8088/proxy/application_142  
15/02/08 10:30:35 INFO mapreduce.Job: Running job: job_1423408479621_0009  
15/02/08 10:30:52 INFO mapreduce.Job: Job job_1423408479621_0009 running in uber mode : false  
15/02/08 10:30:52 INFO mapreduce.Job: map 0% reduce 0%  
15/02/08 10:31:22 INFO mapreduce.Job: map 58% reduce 0%  
15/02/08 10:31:25 INFO mapreduce.Job: map 100% reduce 0%  
15/02/08 10:31:52 INFO mapreduce.Job: map 100% reduce 100%  
15/02/08 10:31:53 INFO mapreduce.Job: Job job_1423408479621_0009 completed successfully  
15/02/08 10:31:53 INFO mapreduce.Job: Counters: 49
```

```
$ wget -O wordcount.jar  
https://drive.google.com/uc?export=download&id=1kn4GSjxe8dvaYOrKqUPqJwwY1HnoZ2BK  
$ hdfs dfs -put wordcount.jar /user/cloudera/input/  
$ hadoop jar wordcount.jar org.myorg.WordCount  
/user/cloudera/input/* /user/cloudera/output/wordcount
```

# Reviewing MapReduce Job in Hue

The screenshot shows the Hue Job Browser interface. At the top, there is a navigation bar with links for Home, Query Editors, Data Browsers, Workflows, Search, and Security. Below the navigation bar, the title "Job Browser" is displayed next to a blue water droplet icon. On the left, there are search fields for "Username" and "Text". To the right of these fields are four colored buttons: green for "Succeeded", orange for "Running", red for "Failed", and dark grey for "Killed". The main area displays a table of jobs. The columns are: Logs, ID, Name, Application Type, Status, User, Maps, Reduces, Queue, Priority, Duration, and Submitted. A single job entry is shown: "Logs" (represented by a small icon), "ID" (1465875170640\_0001), "Name" (wordcount), "Application Type" (MAPREDUCE), "Status" (SUCCEEDED), "User" (root), "Maps" (100%), "Reduces" (100%), "Queue" (root.root), "Priority" (N/A), "Duration" (21s), and "Submitted" (06/13/16 21:32:37). At the bottom left, it says "Showing 1 to 1 of 1 entries". At the bottom right, there are navigation buttons for "← Previous", "1", and "Next →".

Logs	ID	Name	Application Type	Status	User	Maps	Reduces	Queue	Priority	Duration	Submitted
	1465875170640_0001	wordcount	MAPREDUCE	SUCCEEDED	root	100%	100%	root.root	N/A	21s	06/13/16 21:32:37

# Reviewing MapReduce Job in Hue

The screenshot shows the Hue Job Browser interface for a completed MapReduce job. The job ID is 146587517064... and the type is MR2. The status is SUCCEEDED. The recent tasks table lists three tasks: two MAP tasks and one REDUCE task, all of which have completed successfully.

JOB ID	wordcount		
TYPE	MR2		
USER	root		
STATUS	SUCCEEDED		
LOGS	Logs		
Attempts	Tasks	Metadata	Counters
<b>Recent Tasks</b>			
Logs	Tasks	Type	
	task_1465875170640_0001_m_000000	MAP	
	task_1465875170640_0001_m_000001	MAP	
	task_1465875170640_0001_r_000000	REDUCE	

# Reviewing MapReduce Output Result

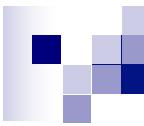
The screenshot shows the Hue File Browser interface. At the top, there is a navigation bar with links for Query Editors, Data Browsers, Workflows, Search, Security, and various system icons. Below the navigation bar, the title "File Browser" is displayed. In the center, there is a search bar labeled "Search for file name" and a "Actions" dropdown menu. To the right of the search bar is a "Move to trash" button. Below the search bar, the current path is shown as "/ Home / user / cloudera / output / wordcount". The path "/ user / cloudera / output" is highlighted with a red oval, and a red arrow points from the top right towards this oval. On the right side of the path, there is a pencil icon for editing. Further down, there is a "History" and a "Trash" link. The main area displays a table of files in the "wordcount" directory:

	Name	Size	User	Group	Permissions	Date
<input type="checkbox"/>	↑		cloudera	cloudera	drwxr-xr-x	June 13, 2016 09:32 PM
<input type="checkbox"/>	.		root	cloudera	drwxr-xr-x	June 13, 2016 09:32 PM
<input type="checkbox"/>	_SUCCESS	0 bytes	root	cloudera	-rw-r--r--	June 13, 2016 09:32 PM
<input type="checkbox"/>	part-00000	44 bytes	root	cloudera	-rw-r--r--	June 13, 2016 09:32 PM

# Reviewing MapReduce Output Result

The screenshot shows the Hue File Browser interface. The top navigation bar includes links for Home, Query Editors, Data Browsers, Workflows, Search, Security, and various system icons. The main title is "File Browser". On the left, there's a sidebar with "ACTIONS" and several options: "View as binary", "Edit file", "Download", "View file location", and "Refresh". Below this is an "INFO" section. The central content area displays the path "/ user / cloudera / output / wordcount / part-00000" and a list of words with their counts:

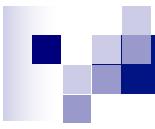
Word	Count
a	205807
e	315232
i	174282
o	192879
u	65433



# Lecture

---

# Understanding Oozie

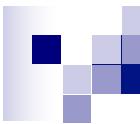


# Introduction

Workflow scheduler for Hadoop



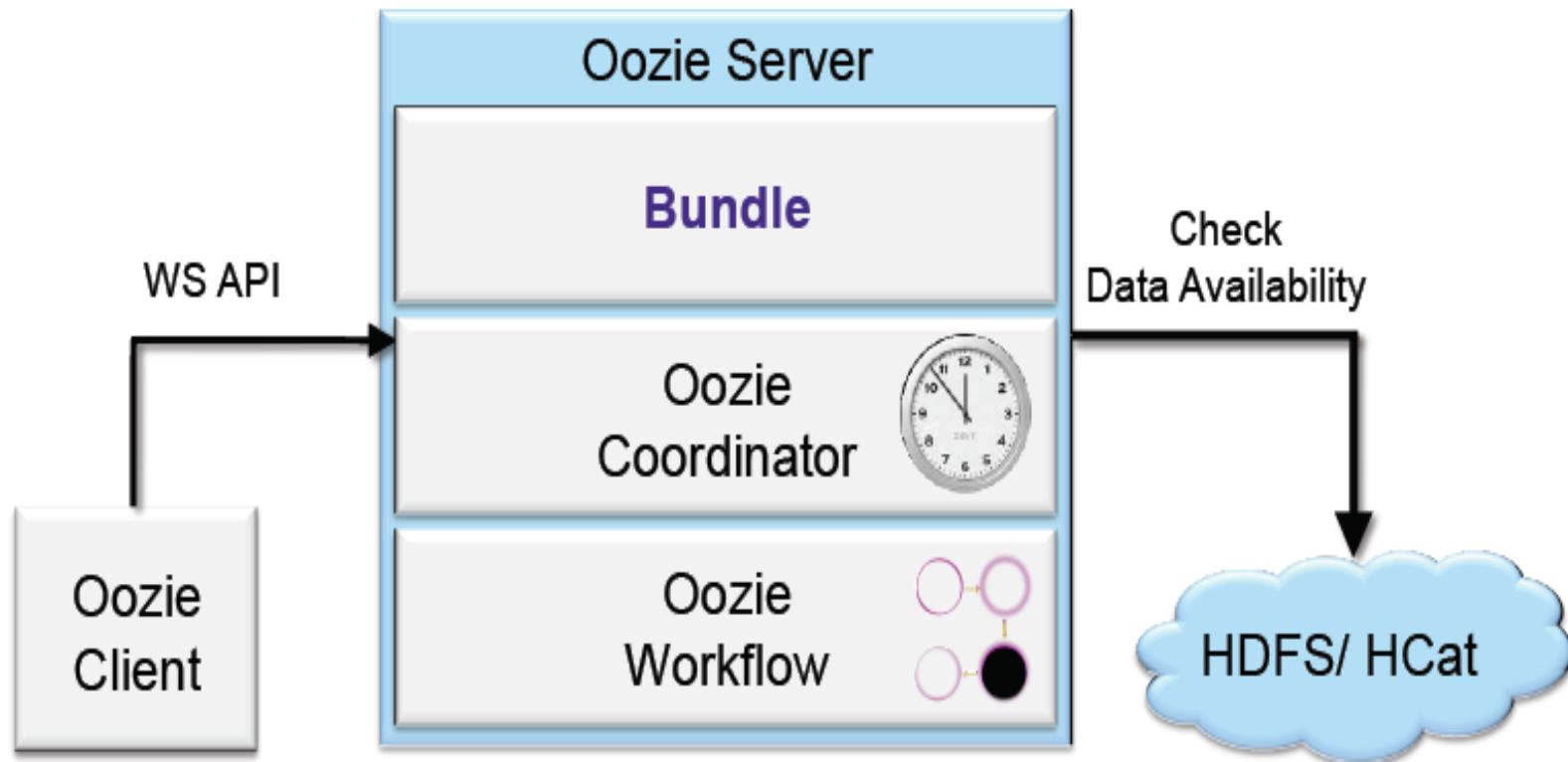
Oozie is a workflow scheduler system to manage Apache Hadoop jobs. Oozie is integrated with the rest of the Hadoop stack supporting several types of Hadoop jobs out of the box (such as Java map-reduce, Streaming map-reduce, Pig, Hive, Sqoop and Distcp) as well as system specific jobs (such as Java programs and shell scripts).



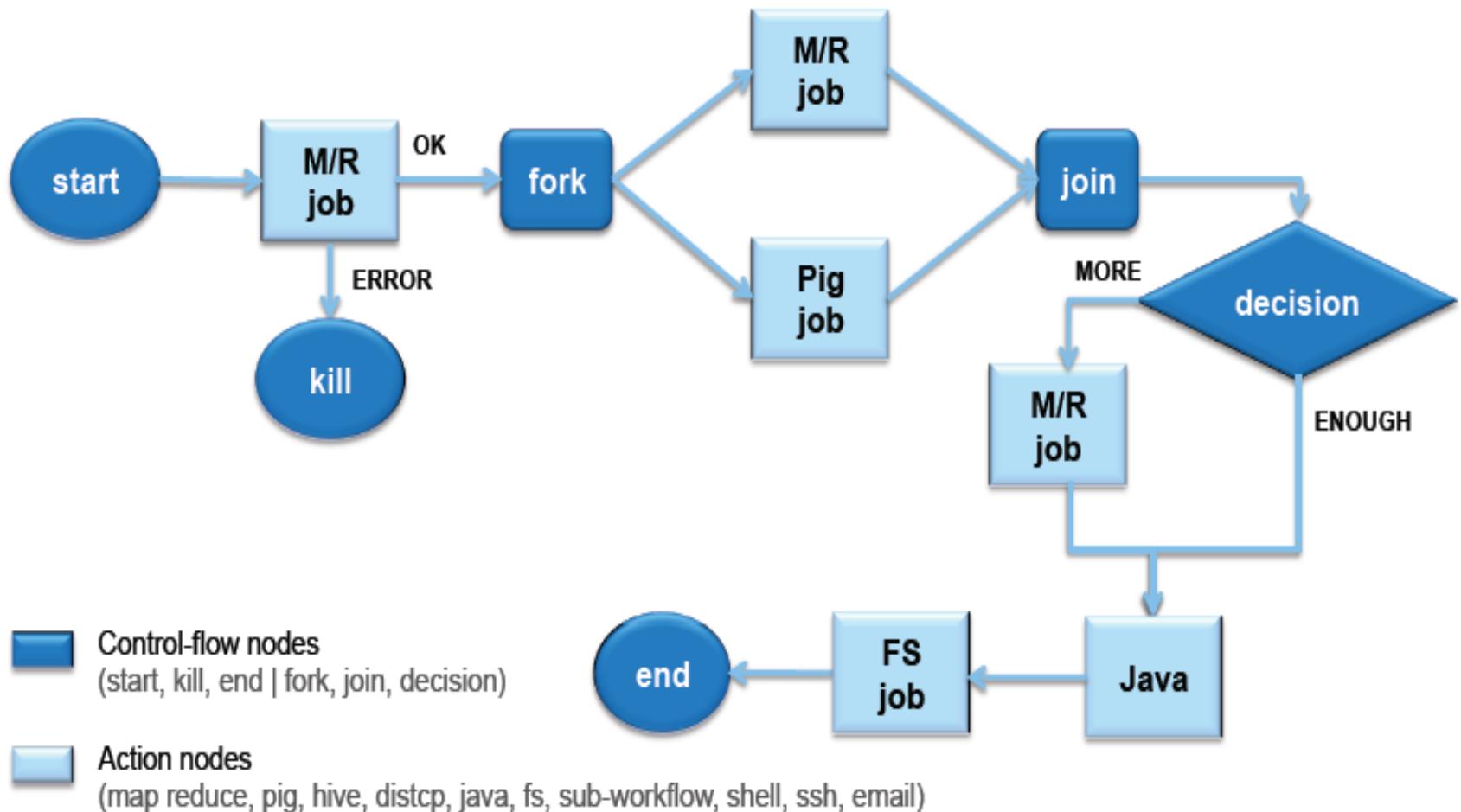
# What is Oozie?

- Work flow scheduler for Hadoop
- Manages Hadoop Jobs
- Integrated with many Hadoop apps i.e. Pig, Hive
- Scalable
- Schedule jobs
- A work flow is a collection of actions.
- A work flow is
  - Arranged as a DAG ( direct acyclic graph )
  - Graph stored as hPDL ( XML process definition )

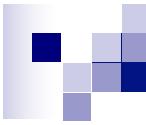
# Oozie Server



# Oozie Workflow



Source: Oozie – Now and Beyond, Yahoo, 2013



# **Hands-On: Running Map Reduce using Oozie workflow**

---

# Using Hue: select WorkFlows >> Editors >> Workflows

The screenshot shows the Hue interface with the 'Workflows' tab selected. The main area displays a table titled 'Workflow Editor' with one entry:

Name	Description	Owner	Last Modified
My Workflow		guest	2015-10-23T23:27Z

Below the table, a message indicates "Showing 1 to 1 of 1 entries".



# Create a new workflow

- Click Create button; the following screen will be displayed
- Name the workflow as WordCountWorkflow

The screenshot shows the Hue interface with the 'Workflows' tab selected. The 'Workflows' section is titled 'Workflow Editor'. It includes a search bar, 'Submit', 'Share', 'Copy', and 'Delete' buttons, and a 'Create' button which is highlighted with a red arrow. A table lists one workflow entry: 'My Workflow' by 'guest' last modified on '2015-10-23T23:27Z'. Navigation buttons for 'Previous', 'Next', and page number '1' are at the bottom.

Name	Description	Owner	Last Modified
My Workflow		guest	2015-10-23T23:27Z

The screenshot shows the Hue Oozie Editor interface. At the top, there is a navigation bar with links for Home, Query Editors, Data Browsers, Workflows, Search, File Browser, Job Browser, guest, and various system icons. Below the navigation bar, the main area is titled "Oozie Editor" and "Workflows". It displays a list of actions: "WordCountWorkflow" (selected), Add a description..., and several other actions represented by icons. A red arrow points to the title "WordCountWorkflow".

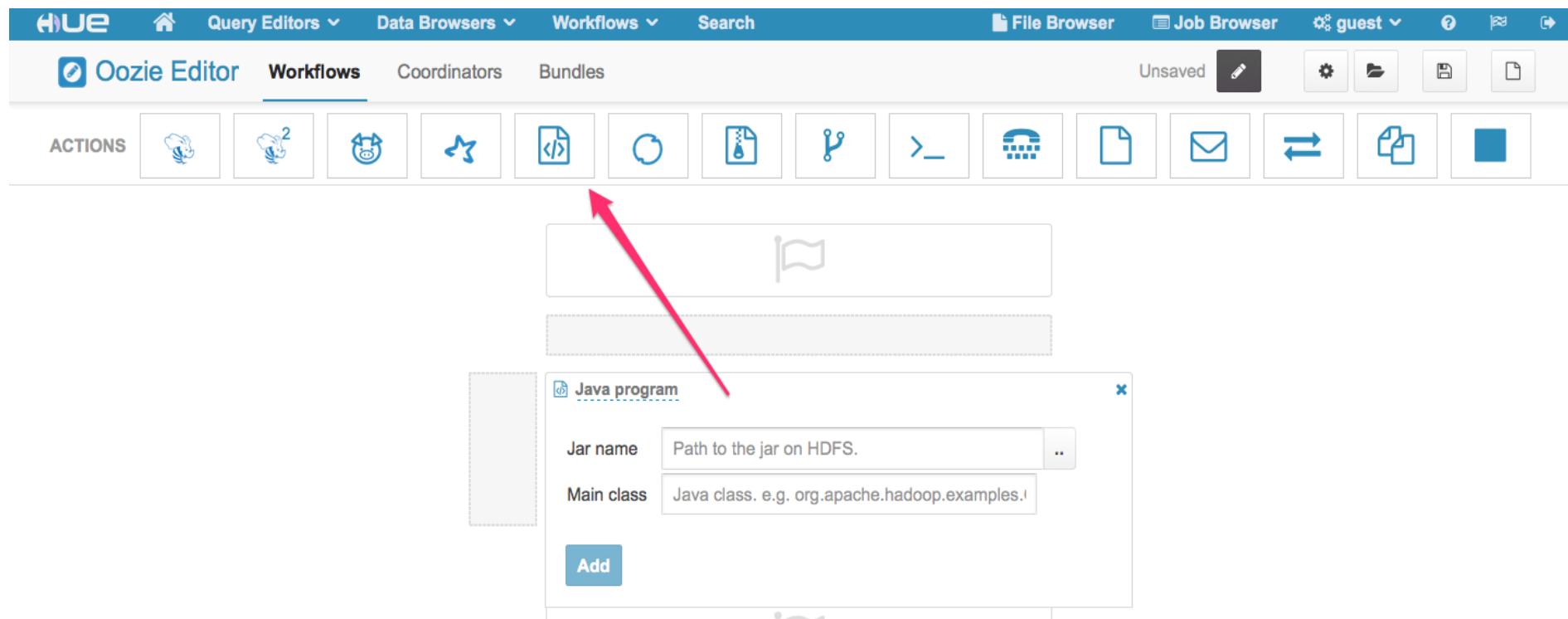
WordCountWorkflow

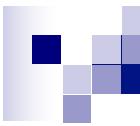
Add a description...

Drop your action here

# Select a Java job for the workflow

From the Oozie editor, drag **Java Program** and drop between start and end

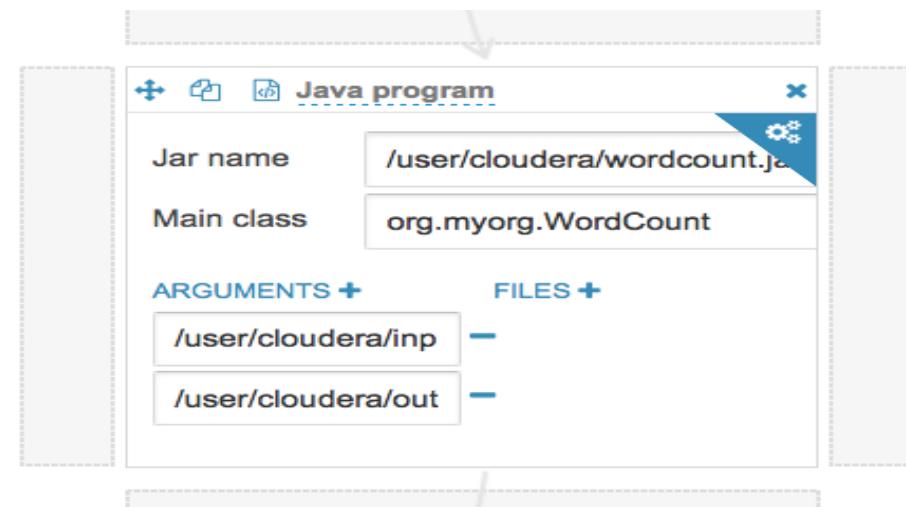




# Edit the Java Job

Assign the following value

- Jar name: wordcount.jar (select ... choose upload from local machine)
- Main Class: org.myorg.WordCount
- Arguments: /user/cloudera/input/\*  
/user/cloudera/output1/wordcount



# Submit the workflow

- Click Save
- Then click submit

