Big Data Hadoop and Spark Developer

Distributed Processing: MapReduce Framework

Introduction to MapReduce

- MapReduce is a programming model that simultaneously processes and analyzes huge data sets logically into separate clusters.
- **Map** sorts the data, whereas **Reduce** segregates it into logical clusters, thus removing the bad data and retaining the necessary information.

Learning Objectives

By the end of this lesson, you will be able to:

- Perform distributed processing in MapReduce
- Illustrate how MapReduce deals with distributed processing issues
- Control the flow of mappers and reducers
- Optimize MapReduce jobs

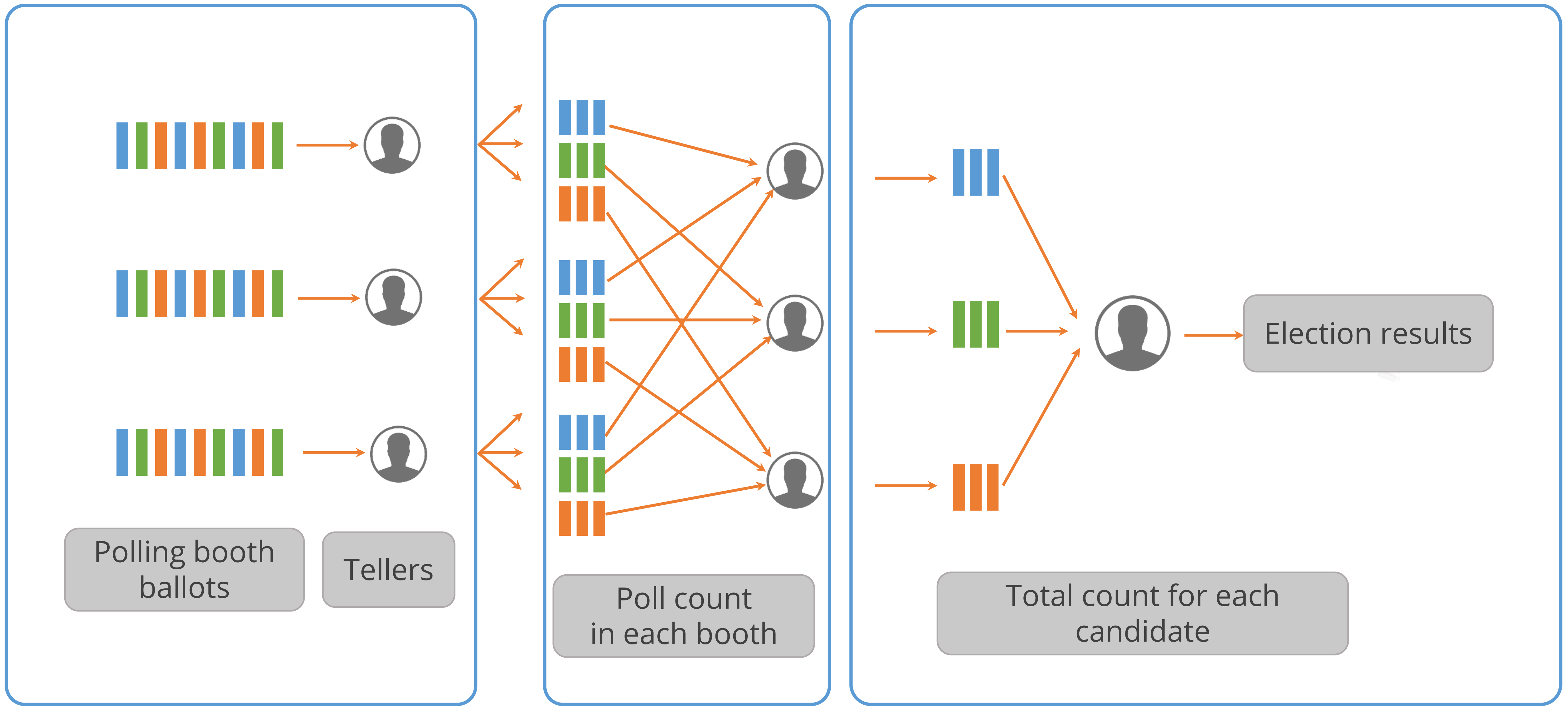
Distributed Processing in MapReduce

Why MapReduce?

Huge amounts of data were stored on single servers before 2004.

MapReduce: Analogy

Here is a diagram for an analogy of how MapReduce works.



MapReduce: Analogy

Here is an example of election results to understand how MapReduce works.

One month to receive the election results

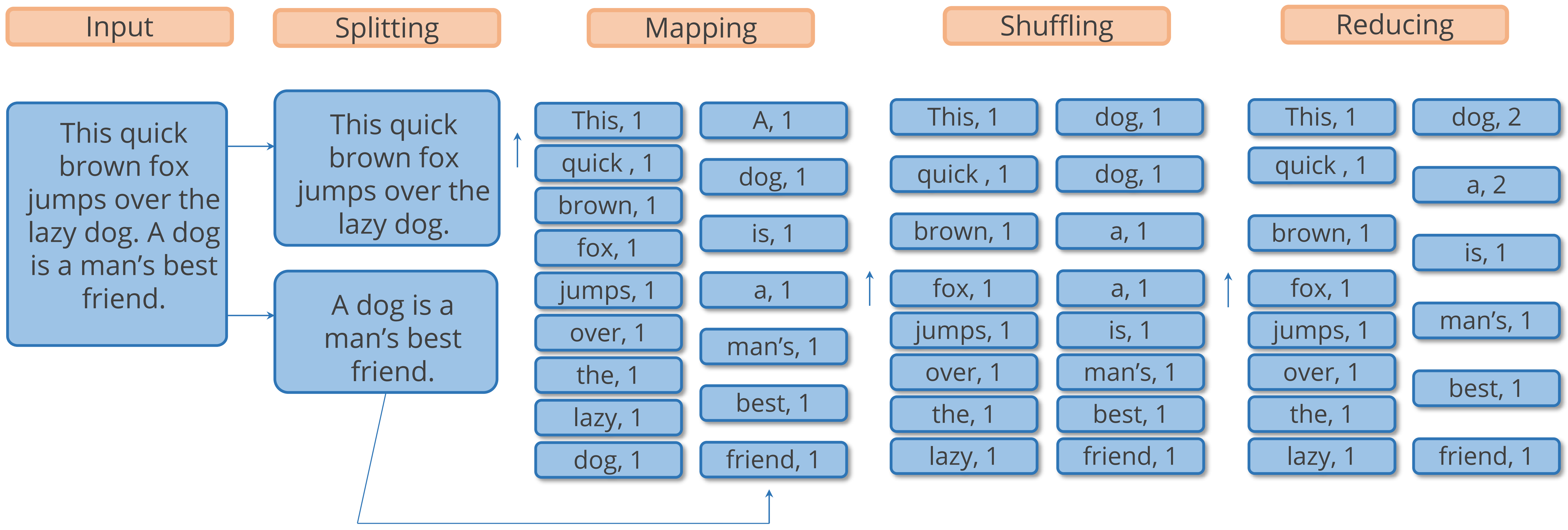
Individual work

Results obtained in one or two days

Parallel work

MapReduce: Word Count Example

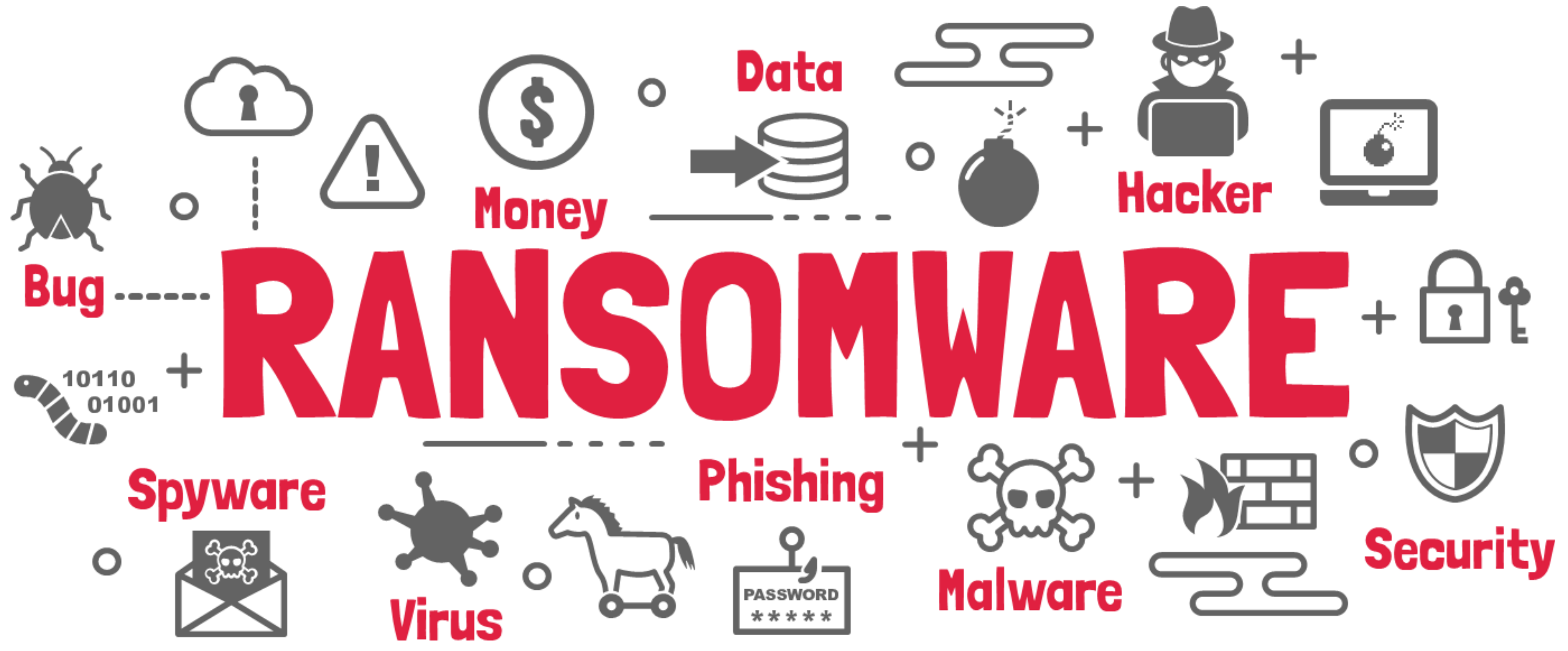
The overall MapReduce word count process is shown below:



Case Study: Flipkart Dodged WannaCry Ransomware

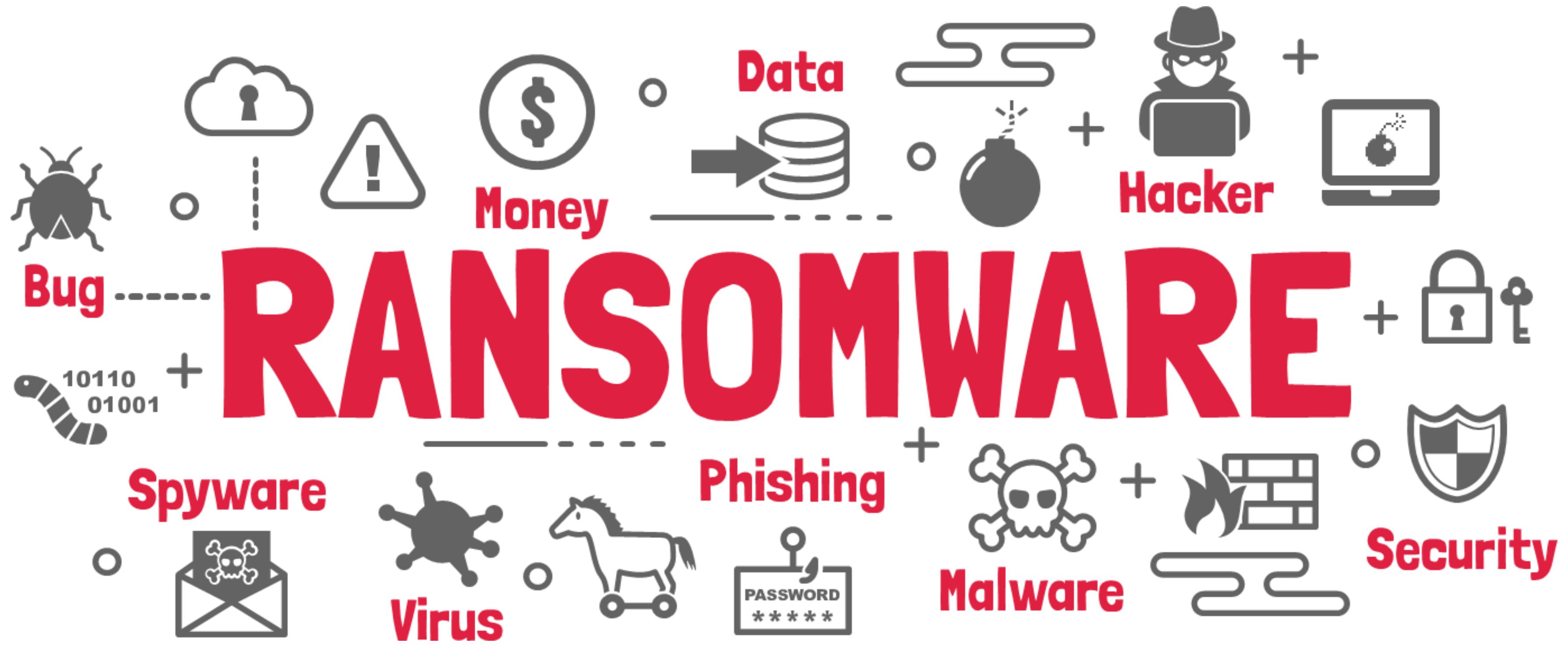
Case Study: Flipkart Dodged WannaCry Ransomware

Ransomware caused a stir by infiltrating Windows systems and encrypting data. To recover the data, the owner or organization was forced to pay a ransom in Bitcoin.



Case Study: Flipkart Dodged WannaCry Ransomware

The WannaCry ransomware was rapidly spreading across systems in 2018, particularly on Windows-based OS. Flipkart (backed by Walmart) ran 90 percent of its servers on Windows, increasing the probability of an attack.



Case Study: Flipkart Dodged WannaCry Ransomware

Challenges

- Flipkart managed to acquire a security patch from Microsoft to fix the vulnerabilities. However, the main issue was how and when to apply the patch to the systems.

- There were almost 10,000 servers that needed to be patched. The biggest question was at what time the patching should happen.

Case Study: Flipkart Dodged WannaCry Ransomware

Solution

- Server logs were analyzed to find the precise time when traffic on the servers was low and the patch could be applied.
- MapReduce, a Hadoop component, was selected to analyze and process server logs.

Case Study: Flipkart Dodged WannaCry Ransomware

Using MapReduce on 200 terabytes of data, Flipkart found that the request count was lowest at 4 a.m. every day and decided to shut down the servers to apply the Microsoft patch.

MapReduce Terminologies

MapReduce Terminologies

Keys

- It is a recording entity received at the mapper given by RecordReader.
- For example, in (country,1), country is a key. Keys are driven by use case and are selected by users.

Value

- It is a part of a recording entity and holds data for computing.
- For example, in (country,1), 1 is a value.

InputSplit

- It's a logical representation of data that InputFormat generates and maps to a single task that a mapper will execute.
- It describes a unit of work that contains a single map task in MapReduce.

MapReduce Terminologies

RecordReader	• It is an internal framework that converts the byte-oriented view of data from InputSplit to key-value pairs and passes them on to the Map.
Mapper	• It is a self-sustained function that takes input from RecordReader and processes it. • The input for Mapper is a Map (K, V), and the output is a Map (K1, V1). It is stored on HDFS in the temp directory.
Reducer	• It is a function that takes input from the Mapper output. • Then, it performs computations, such as sum, filter, and aggregation, on them and provides a Map (K2, V2) as an output.

MapReduce Terminologies

Partitioner

- It decides the partitioning of keys for the intermediate map outputs.
- It also controls the middle map output segment.

Counter

- It is a function to gather statistics about the MapReduce job.
- These are inbuilt but can be user-defined as well.

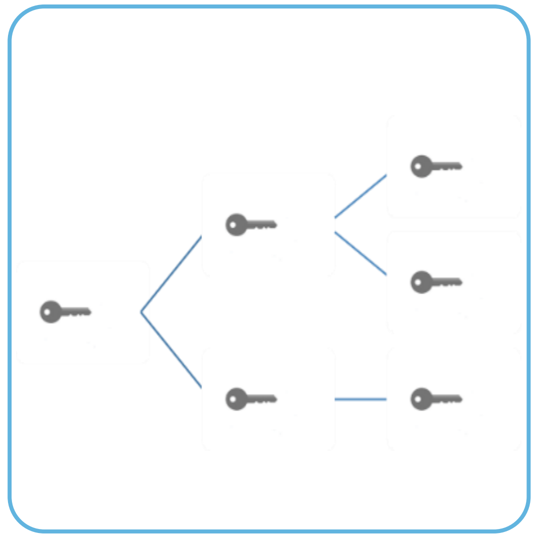
Combiner

- It is also known as a mini-reducer.
- It works at the mapper level and optimizes data transfer to reduce congestion before passing it to the reducer.

MapReduce Essentials

The job input is specified in key-value pairs:

A user-defined map function is applied to each input record to produce a list of intermediate key-value pairs.



A user-defined reduce function is called once for each distinct key in the map output.

MapReduce Essentials

The essential steps of each MapReduce phase include:

- The number of reduce tasks can be defined by the users.
- Each reduce task is assigned a set of record groups, that is, intermediate records corresponding to a group of keys.
- For each group, a user-defined reduce function is applied to the recorded values.
- The reduce tasks are read from every map task, and each read returns the record groups for that reduce task.

Reduce phase cannot start until all mappers have finished processing.

Map Execution Phases

Map Execution Phases

Map phase

- Reads assigned input split from HDFS
- Parses input into records as key-value pairs
- Applies map function to each record
- Informs master node of its completion

Partition phase

- Each mapper must determine which reducer will receive each of the outputs
- For any key, the destination partition is the same
- Number of partitions is equal to the number of reducers

Shuffle phase

- Fetches input data from all map tasks for the portion corresponding to the reduce tasks' bucket

Sort phase

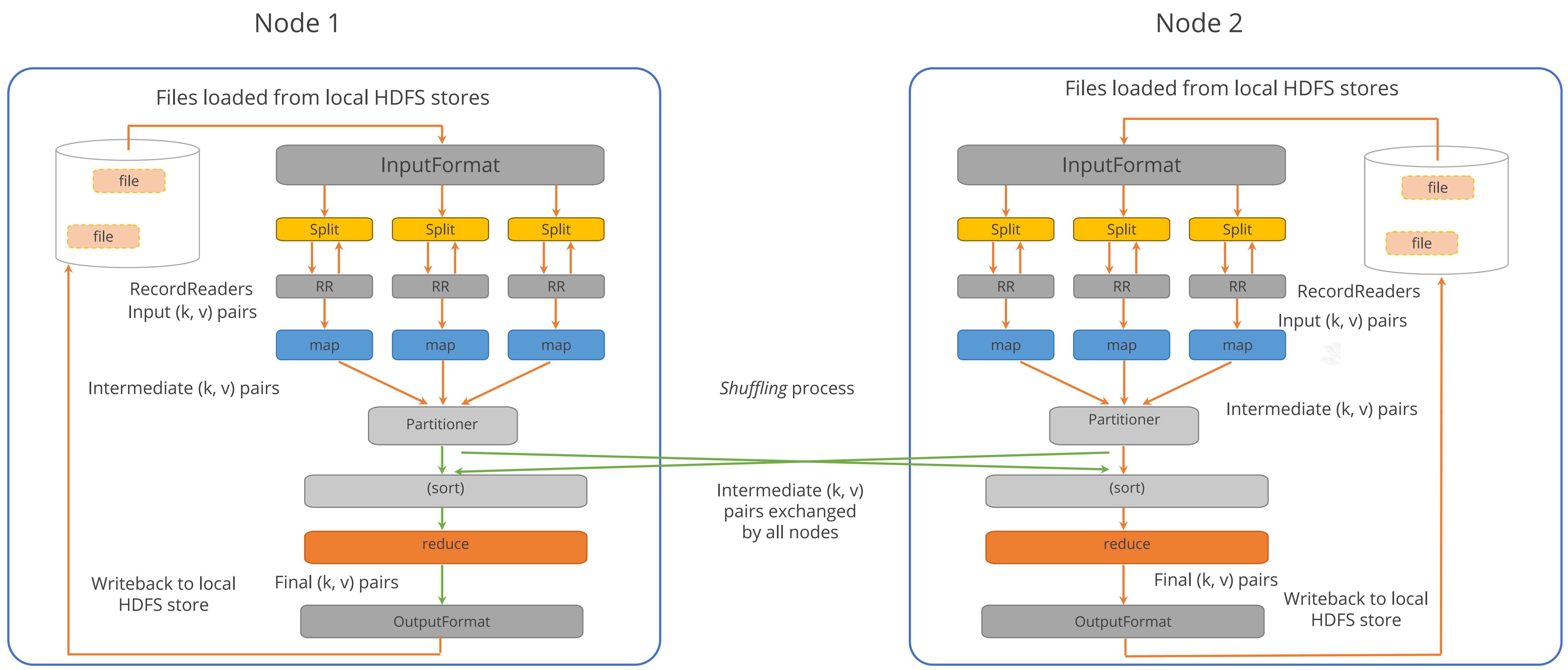
- Merge sorts all map outputs into a single run

Reduce phase

- Applies user-defined reduce function to the merged run

Map Execution: Distributed Two-Node Environment

Here is a diagram representing a two-node distributed environment.



MapReduce Jobs

MapReduce Jobs

A job is a MapReduce program that causes multiple map and reduce functions to run parallelly over the life of the program.

ApplicationMaster and NodeManager functions:

ApplicationMaster

- Is responsible for the execution of a single application or MapReduce job
- Divides job requests into tasks and assigns them to NodeManagers running on the slave node

NodeManager

- Has many dynamic resource containers
- Executes each active map or reduce task regularly with the ApplicationMaster

MapReduce and Associated Tasks

Map process
Is an initial ingestion and transformation step where initial input records are processed parallely

ApplicationMaster
Keeps track of a MapReduce job

Reduce process
Is an aggregation or summarization step in which all associated records are processed together

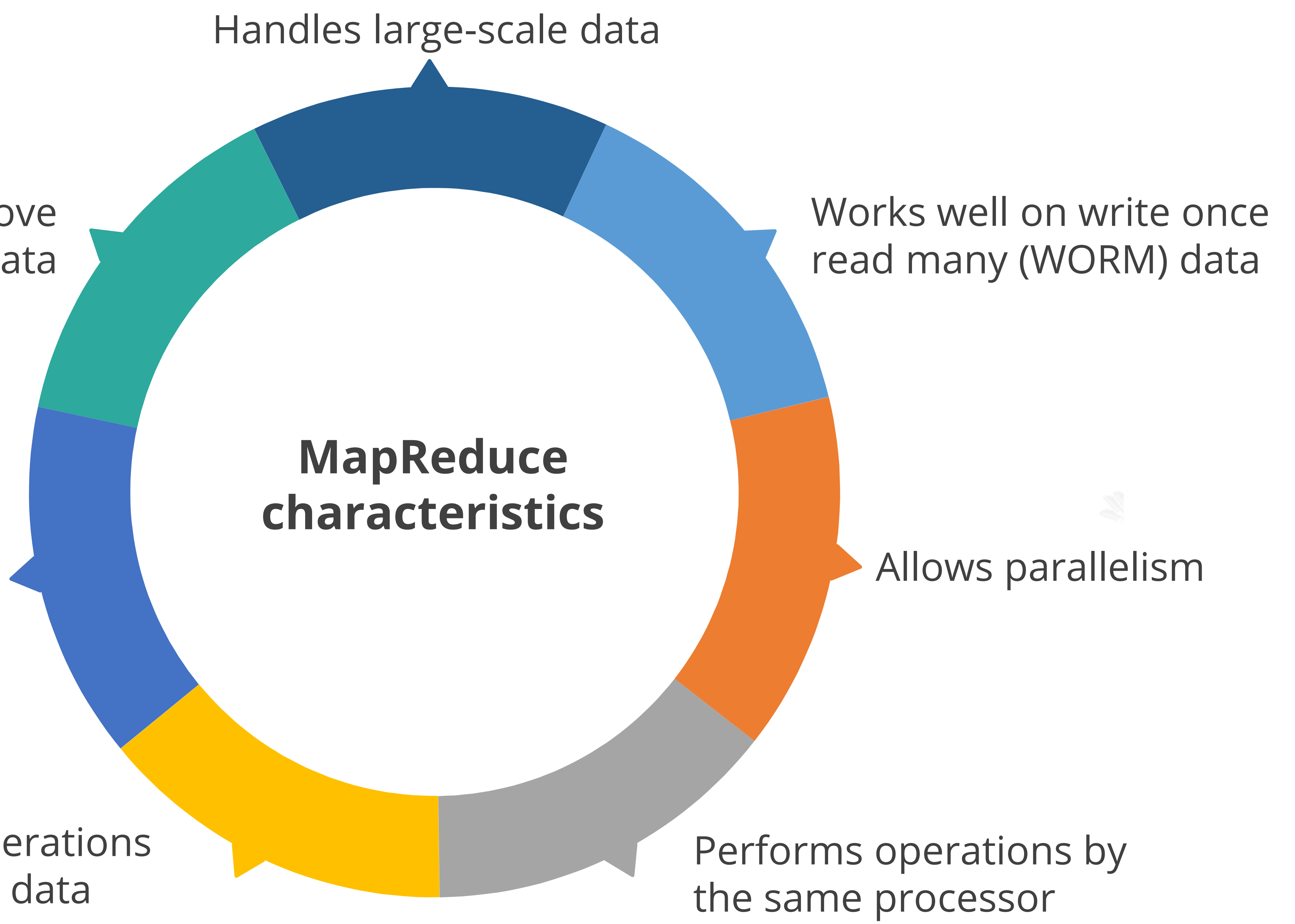
NodeManager
Keeps track of individual map tasks and can run in parallel to other NodeManager

Hadoop MapReduce Job Work Interaction

Hadoop MapReduce

Job submission	Hadoop MapReduce job is submitted by a client in the form of an input file or a number of input split of files containing data.
Input split distribution	MapReduce ApplicationMaster distributes the input split to separate NodeManagers.
Success or failure status	ResourceManager gathers the final output and informs the client of the success or failure status.
Resubmission of tasks	MapReduce ApplicationMaster resubmits the tasks to an alternate NodeManager if the DataNode fails.
Coordination with NodeManager	MapReduce ApplicationMaster coordinates with the NodeManagers.

Characteristics of MapReduce


Handles large-scale data
Works well on write once read many (WORM) data
Allows parallelism
Performs operations by the same processor
Provides operations near the data
Leverages commodity hardware and storage
Attends to split and move data
MapReduce characteristics

Real-Time Uses of MapReduce

- Algorithms
- Sorting
- Data mining
- Search engine operations
- Enterprise analytics
- Gaussian analysis
- Semantic web 3.0

Building a MapReduce Program

Building a MapReduce Program

The steps to build a MapReduce program are:

1 Determine the data

2 Design and implement a solution

3 Compile the source code

4 Package the code

5 Configure the application

6 Load the data

7 Launch the job and monitor

8 Study the result

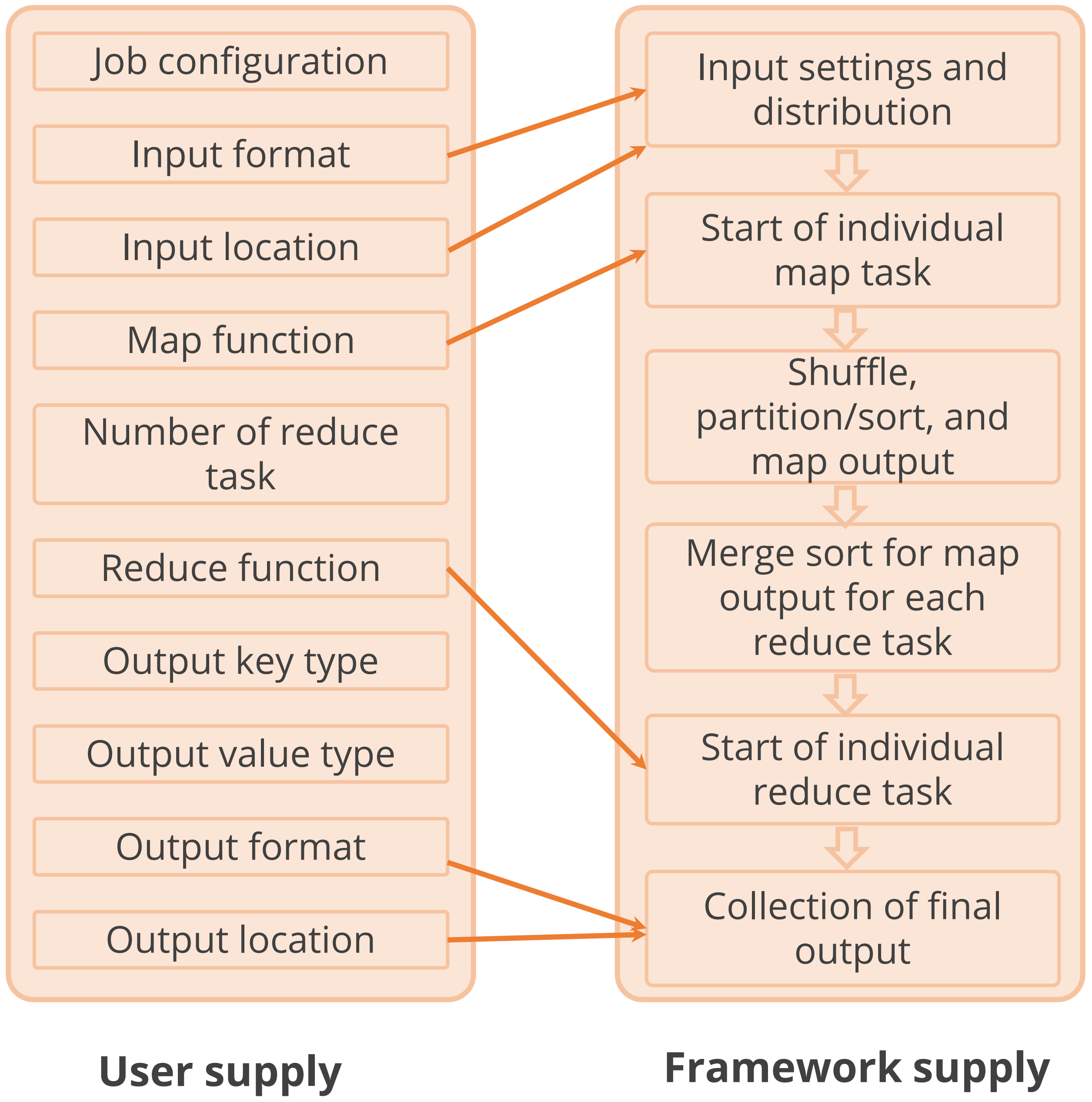
Hadoop MapReduce Requirements

The user or developer is required to set the framework with the following parameters:

- Locations of the job input
- Locations of the job output
- Input format
- Output format
- Class containing the map function
- Class containing the reduce function

Set of Classes

The image below shows the set of classes under user supply and the framework supply.


Job configuration
Input format
Input location
Map function
Number of reduce task
Reduce function
Output key type
Output value type
Output format
Output location
Input settings and distribution
Start of individual map task
Shuffle, partition/sort, and map output
Merge sort for map output for each reduce task
Start of individual reduce task
Collection of final output
User supply
Framework supply

- ResourceManager accepts the input and hands over the job to ApplicationMaster, which divides the job into tasks.
- NodeManager completes the assignment by executing the map task as part of container execution.
- The reducer starts the merging process.
- The output is collected.

MapReduce - Responsibilities

Responsibilities

Developer

- Setting up the job
- Specifying the input location
- Ensuring correct format and location of input

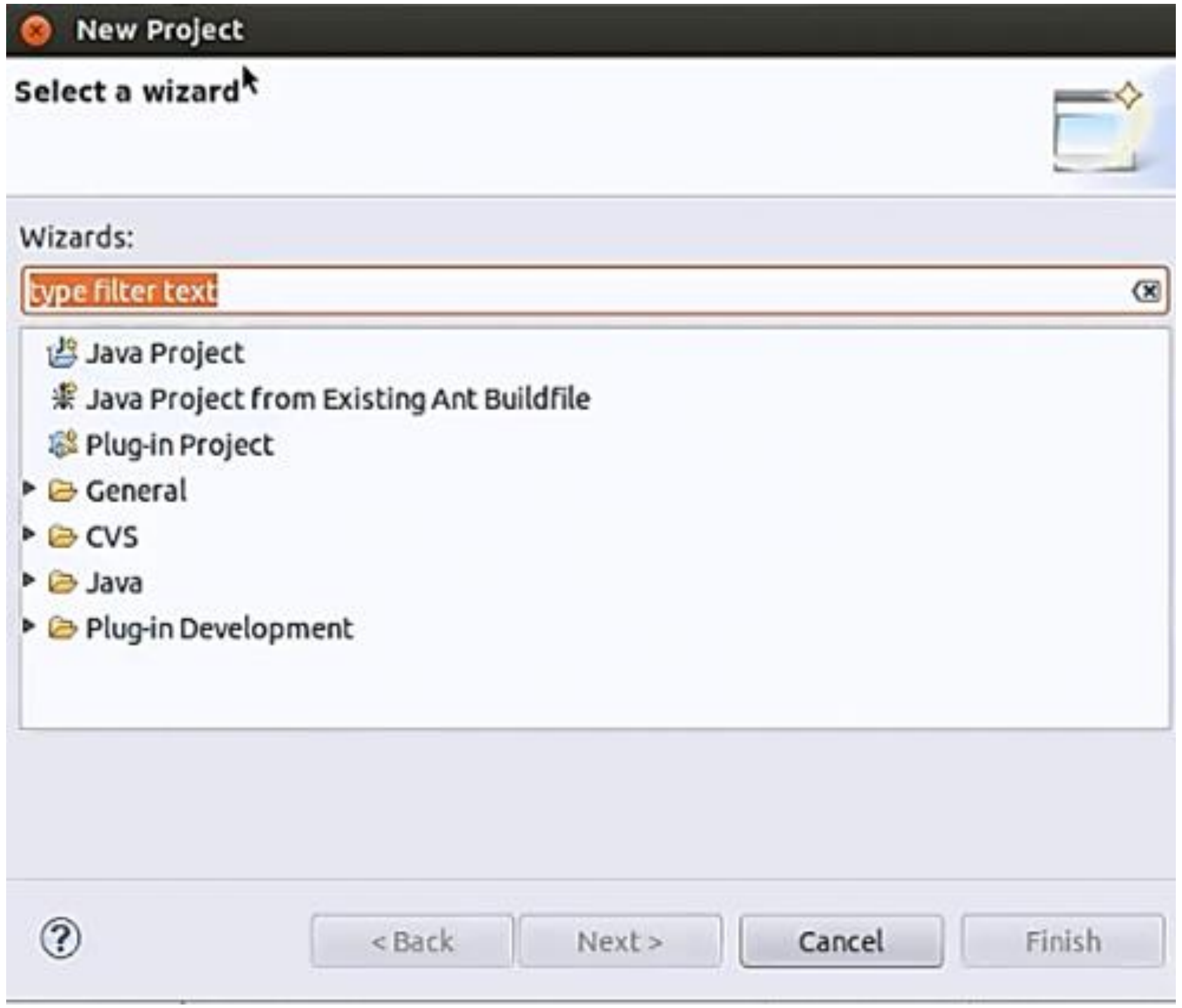
Framework

- Distributing the job
- Running the map operation
- Performing shuffling and sorting
- Reducing phases
- Placing the output
- Informing the user of *job completion* status

Creating a New Project on MapReduce

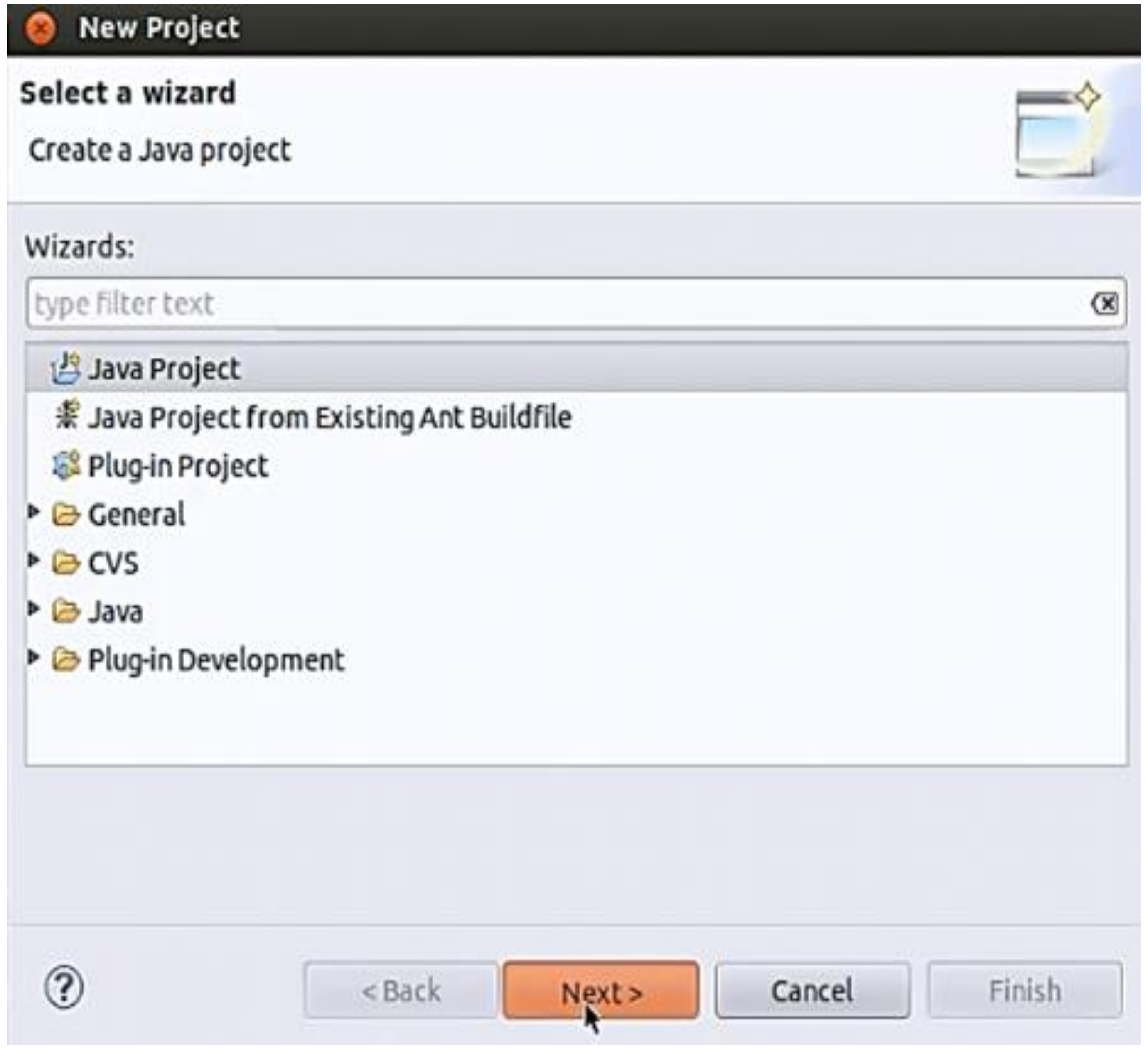
Create a New Project: Step 1

Create a new project and add essential JAR files to run MapReduce programs



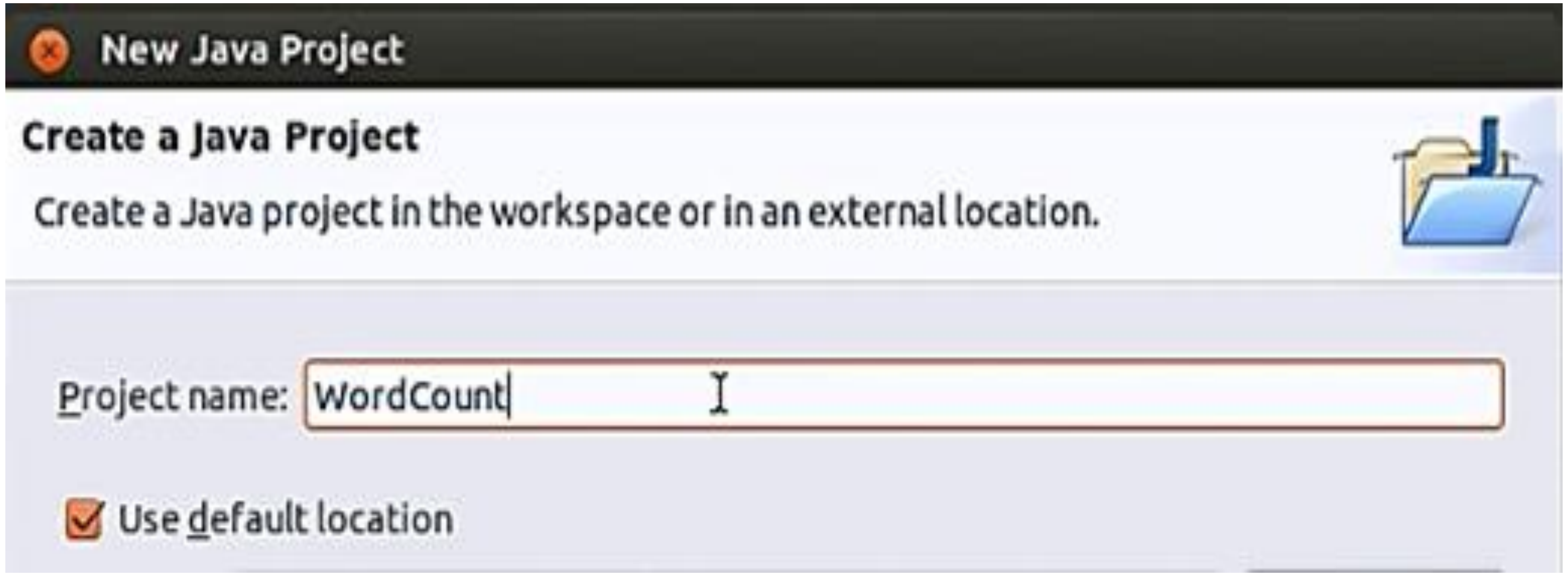
Create a New Project: Step 2

Select **Java Project**, and then click the **Next**



Create a New Project: Step 3

Enter the project name



New Java Project

Create a Java Project

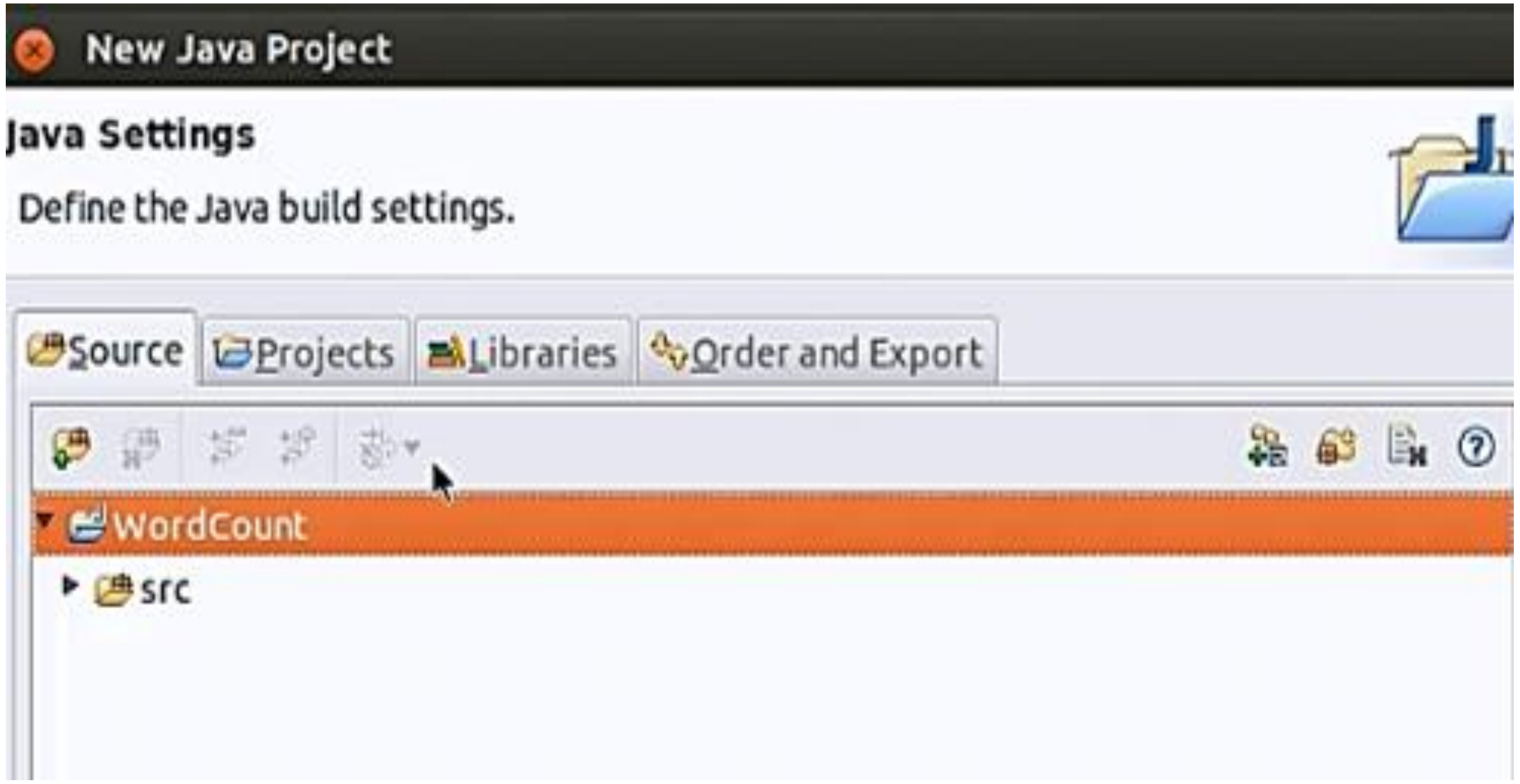
Create a Java project in the workspace or in an external location.

Project name: WordCount

☑ Use default location

Create a New Project: Step 4

Include JAR files from the Hadoop framework to ensure that the programs locate the dependencies in one location



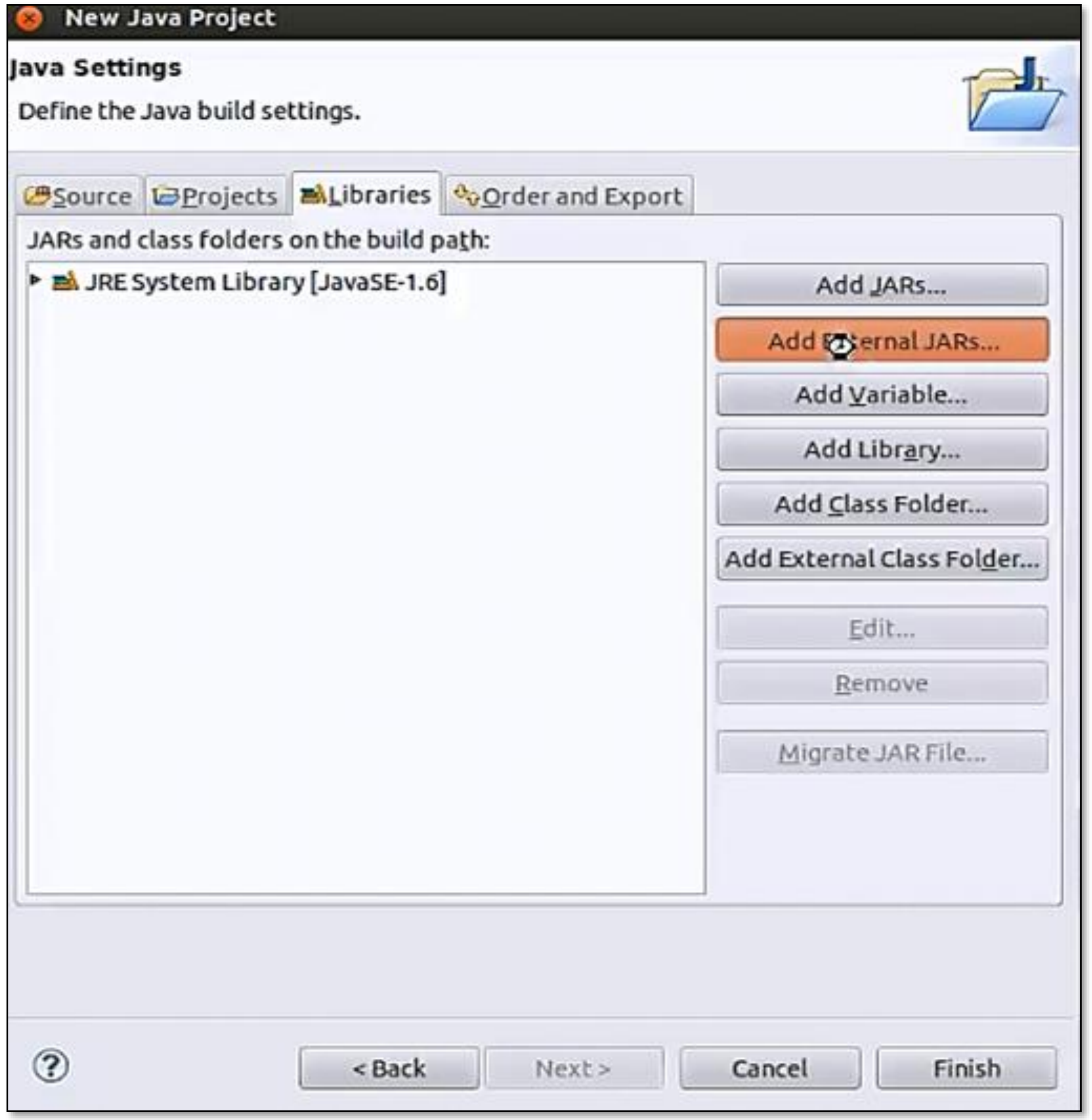
New Java Project

Java Settings

Define the Java build settings.

Create a New Project: Step 5

Add the essential JAR files


New Java Project

Java Settings

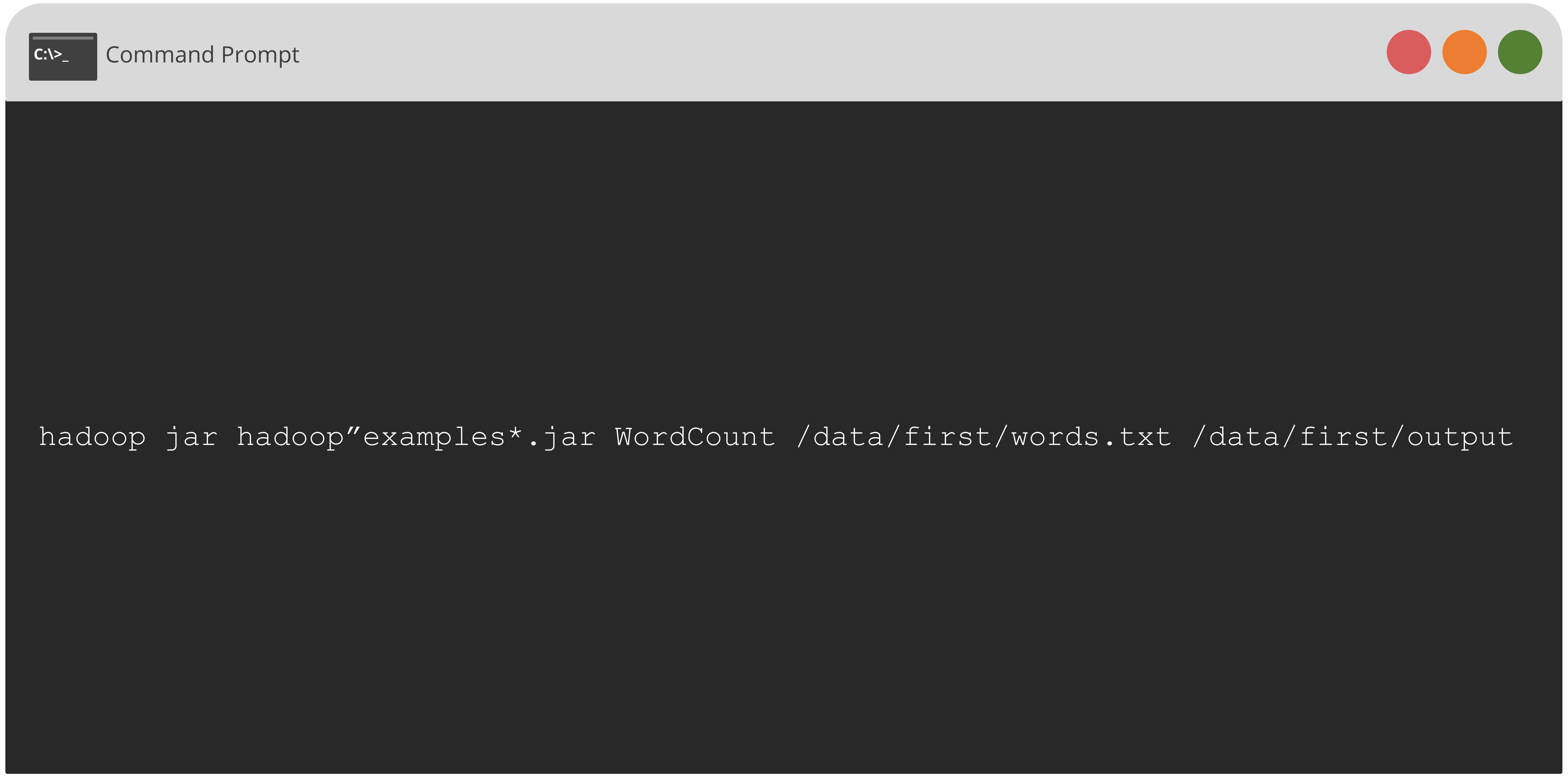
Define the Java build settings.

JARs and class folders on the build path:

JRE System Library [JavaSE-1.6]

Checking Hadoop Environment for MapReduce

Ensure that the machine setup can perform MapReduce operations



```
hadoop jar hadoop”examples*.jar WordCount /data/first/words.txt /data/first/output
```

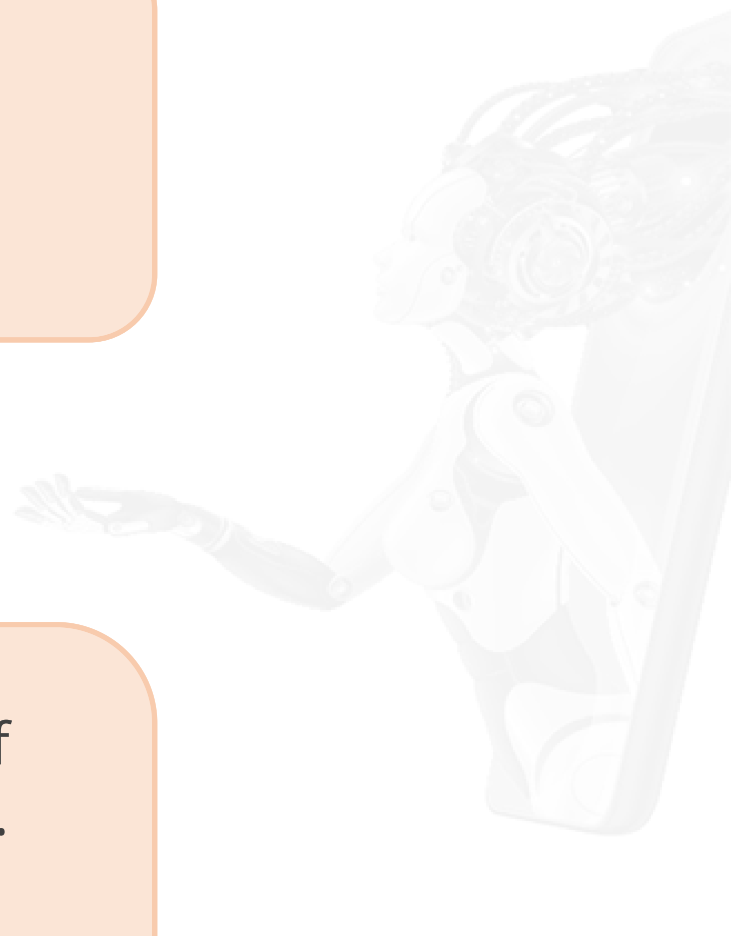
Controlling the Flow of Mapper and Reducer

Number of Maps

- The number of maps is dependent on the total number of blocks in the input dataset.
- Having 10 to 100 maps per node is a common rule.

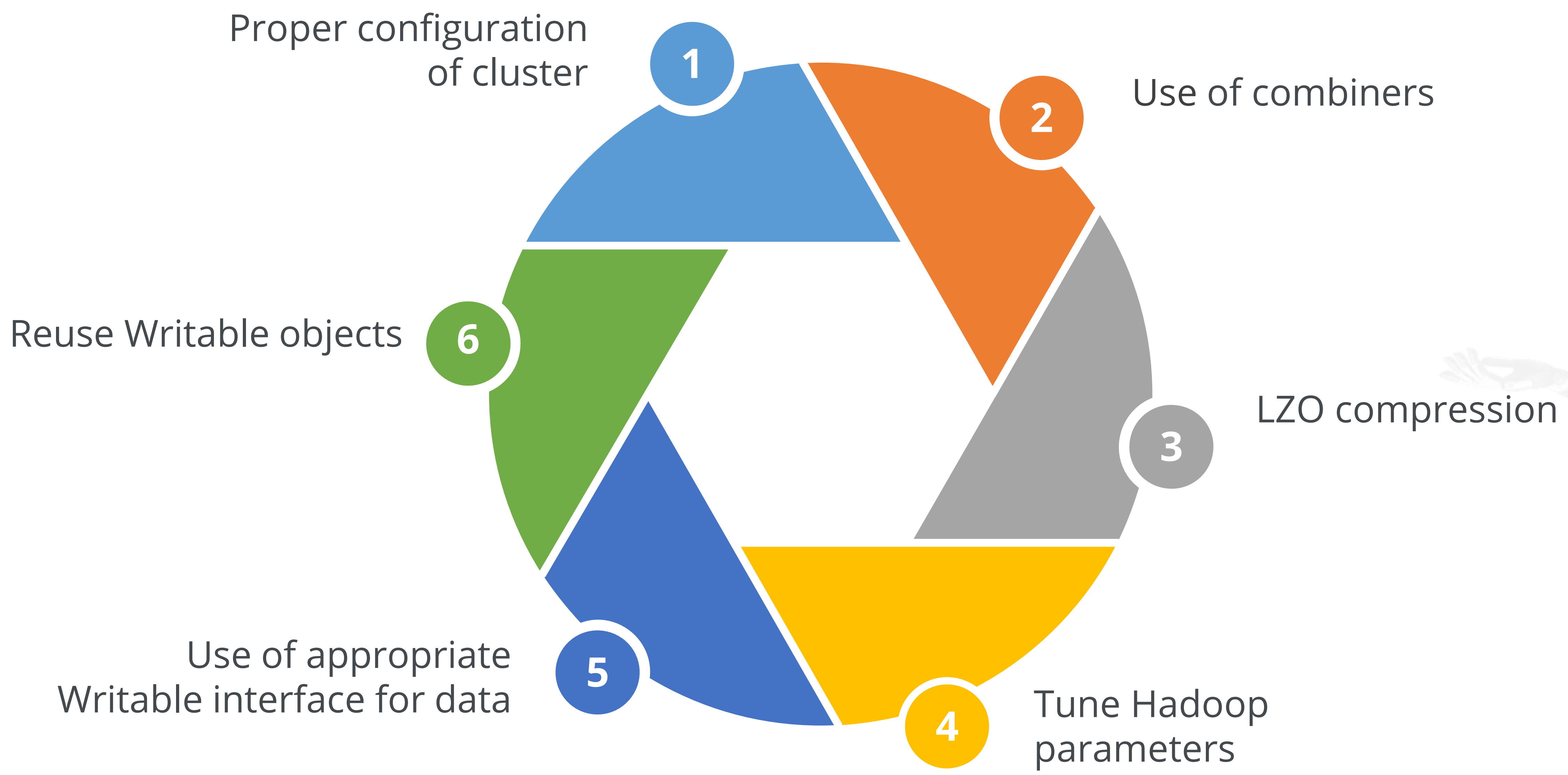
Number of Reducers

- The reducer uses a formula that multiplies 0.95 or 1.75 by the product of the number of nodes and the maximum number of containers per node.
- 0.95* (number of nodes multiplied by the maximum number of containers per node)
- The reducer can also be set to none.

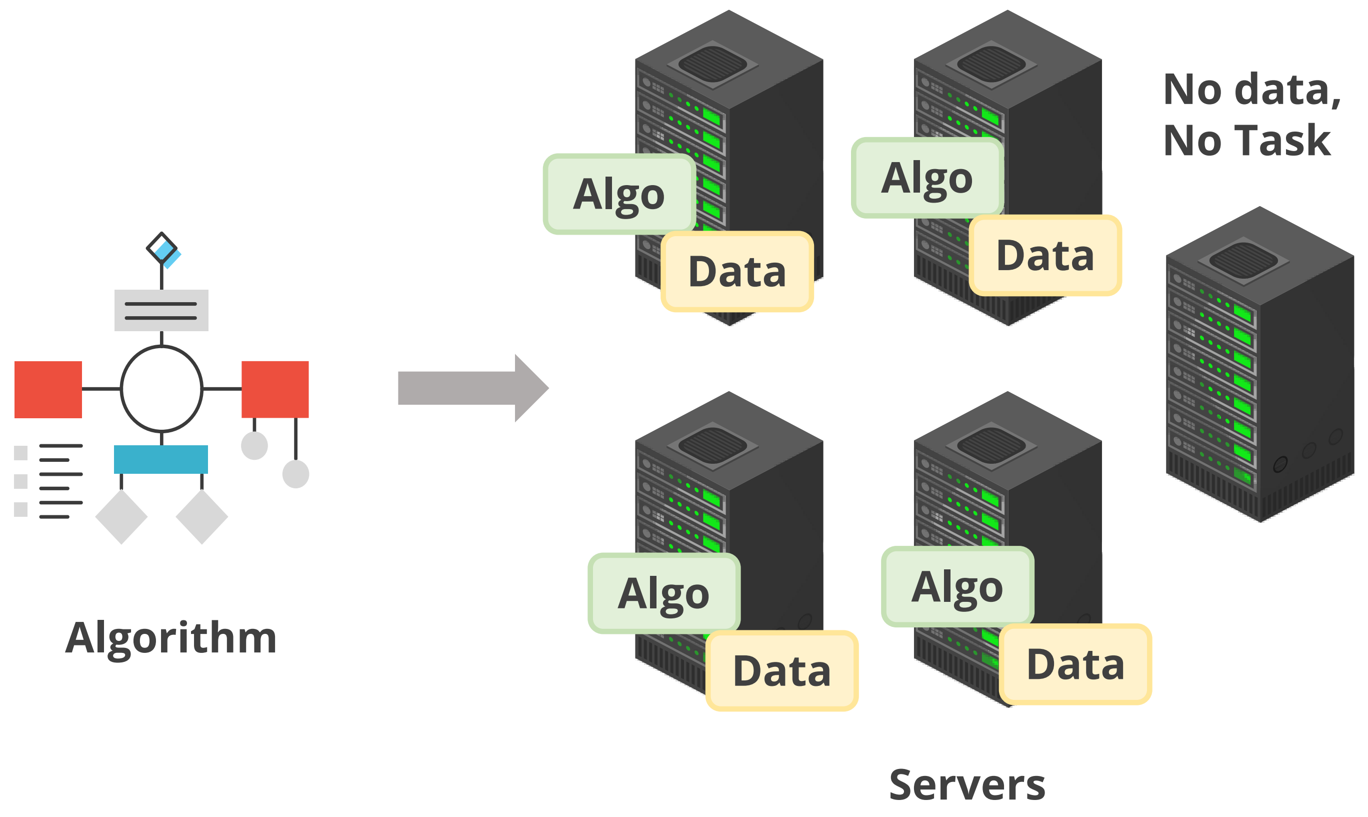


Optimization of MapReduce Jobs

Follow the techniques to achieve maximum performance from the MR cluster:



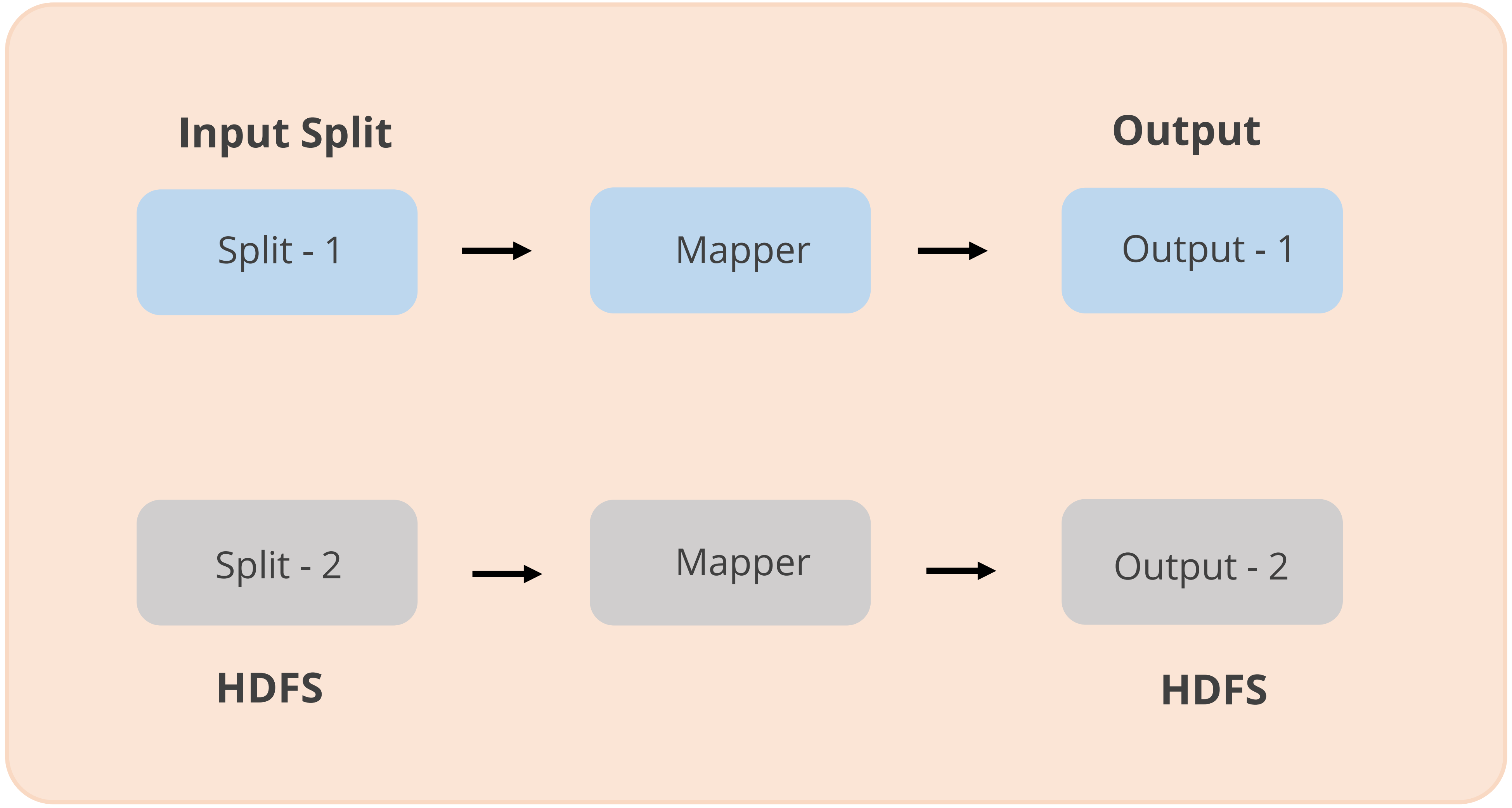
Fault-Tolerance and Data Locality



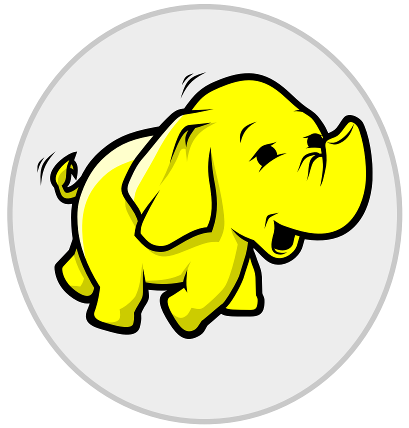
- Fault tolerance refers to the system's capacity to continue running without interruption when one or more of a system's components fail.
- Data locality is a way to move job computation close to where the actual data resides instead of moving a large amount data to computation.
- It helps to minimize the overall network congestion and increases the overall throughput of the system.
- As we can see in the graphic, the data has not been distributed in the last node. When MR runs, it ignores the last node and no task is sent. Hence, there is no output from that node.

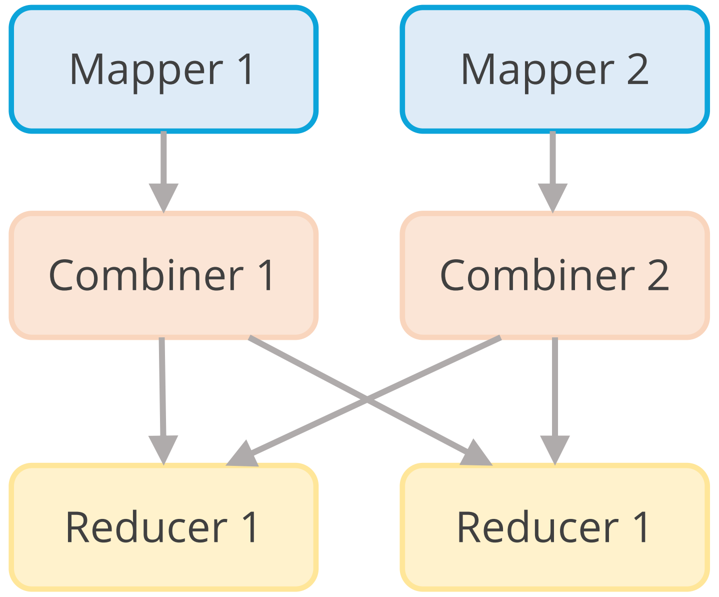
Working with Map-Only Jobs

It is a process in which the mapper executes the tasks without a reducer. This performs well when one transformation is required without shuffling. Hence, the performance improves.



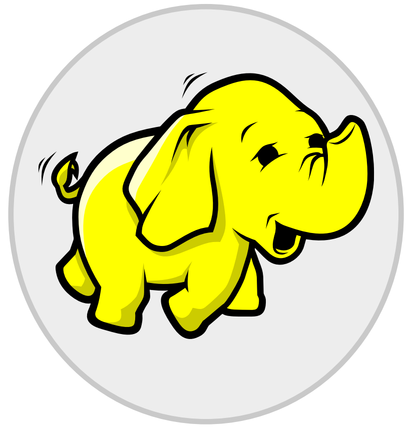
Combiners in MapReduce

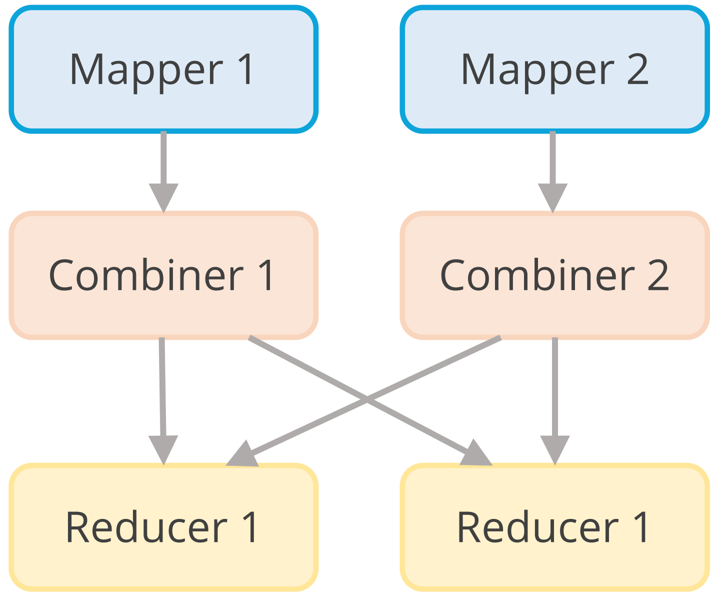

Combiner In
Hadoop



- Combiners are local Reducer.
- It reduces network congestion by acting as a bridge between the Mapper and Reducer.
- The output produced by the Mapper is the intermediate output in terms of key-value pairs.
- If the output is large, it cannot be directly fed to the Reducer as it would cause Network Congestion.

Combiners in MapReduce

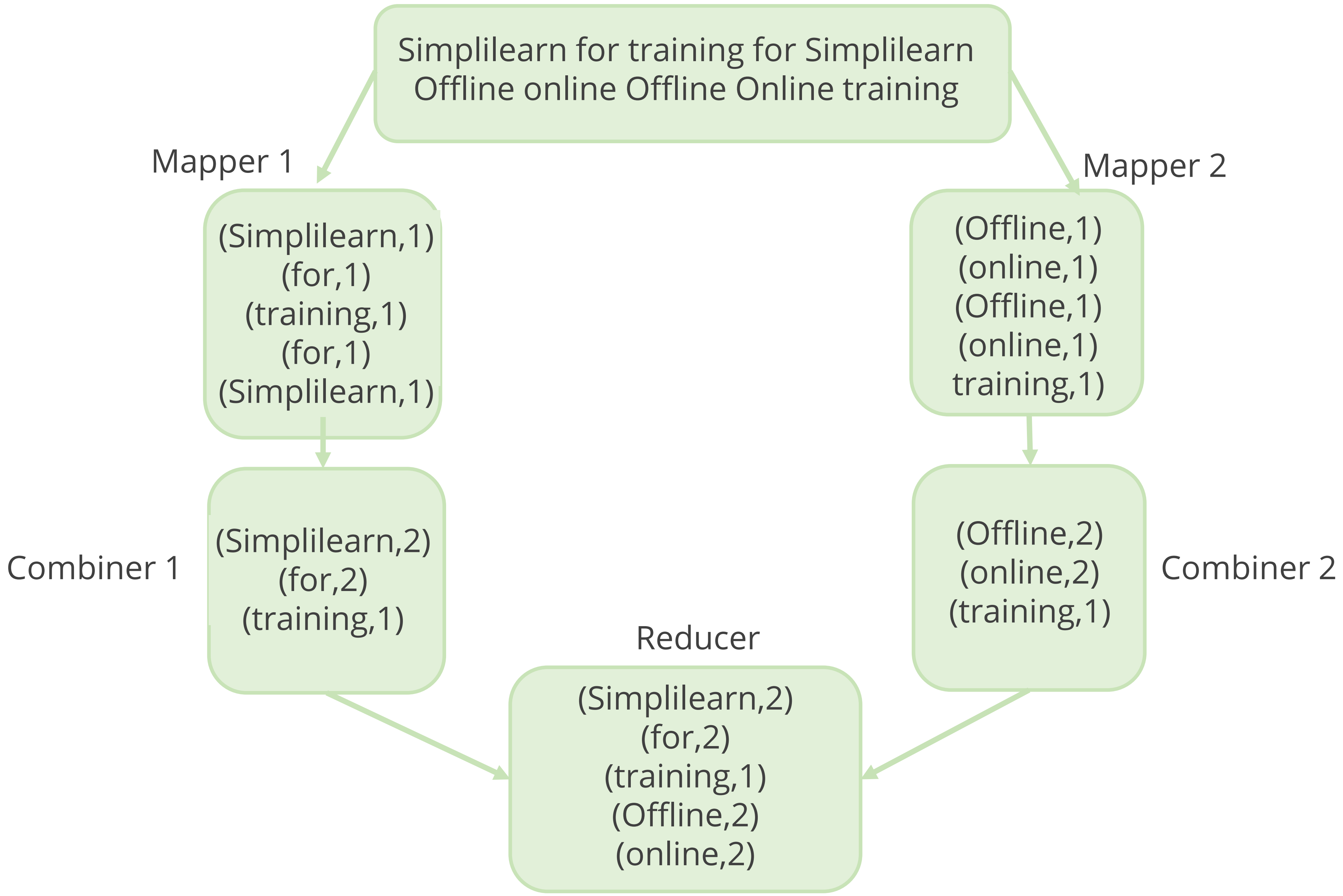

Combiner In Hadoop



- A Combiner is used to minimize congestion.
- It is also known as a semi-reducer.
- It is optional in the MapReduce program.
- It provides a summary of large datasets.
- It improves the overall performance while dealing with large datasets.

MapReduce Program with Combiner

In a program, the combiner accepts inputs from the Map class and then passes the output key-value pairs to the Reducer class.



Simplilearn for training for Simplilearn Offline online Offline Online training
Mapper 1
Mapper 2
(Simplilearn,1) (for,1) (training,1) (for,1) (Simplilearn,1)
(Offline,1) (online,1) (Offline,1) (online,1) training,1)
Combiner 1
(Simplilearn,2) (for,2) (training,1)
Combiner 2
(Offline,2) (online,2) (training,1)
Reducer
(Simplilearn,2) (for,2) (training,1) (Offline,2) (online,2)

Assisted Practice : Execution of MapReduce Job

Duration: 10 mins

Problem Scenario: Write the commands to perform partitions using the JAR files

Objective: In this demonstration, you will use JAR files and the wordcount.txt file to perform the partitions using the MapReduce job.

Dataset Name: "wordcount.txt"

Assisted Practice: Execution of MapReduce Job

Tasks to Perform:

Step 1: Download the **Hadoop-mapreduce-example.jar** file and **wordcount.txt** file

Step 2: Log in to the FTP using the username and password from the lab and upload the file

Step 3: Log in to the Web Console using the username and password from the lab and create a new directory **demo** in HDFS using the **mkdir** command

Step 4: Push the **wordcount.txt** file into the directory using the put command

Step 5: Execute the command to move the **Hadoop-mapreduce-example.jar** file to the HDFS directory

Step 6: View the files in the **Output** folder with the part files

Note: The solution to this assisted practice is provided under the course resources section.

Key Takeaways

- MapReduce divides and analyzes large data sets into logical clusters.
- A job is a MapReduce program that generates multiple maps and reduces functions throughout the program's lifetime.
- Tested techniques must be used to achieve maximum performance from the MR cluster.
- The MapReduce job divides the input dataset into separate chunks, which are then processed in parallel by the map jobs.

Knowledge Check

Knowledge Check

1

Which of the following defines the data locality feature of MapReduce?

A. Reduce task in MapReduce is performed using the map() function

B. Reduce task in MapReduce is performed using the mapper() function

C. Placing data close to NameNode and computing it over DataNodes

D. Compute data by placing it as close as possible

Knowledge Check

1

Which of the following defines the data locality feature of MapReduce?

A. Reduce task in MapReduce is performed using the map() function

B. Reduce task in MapReduce is performed using the mapper() function

C. Placing data close to NameNode and computing it over DataNodes

D. Compute data by placing as close as possible

The correct answer is **D**

Data locality is the process of moving the computation close to where the actual data resides on the node instead of moving large data sets to the computation.

Knowledge Check

2

What is the num parameter in the below line of code:

conf.setNumMapTasks(int num)

A. Number of map tasks for this job

B. Number of reducer tasks for this job

C. Number of splits for this job

D. None of the above

Knowledge Check
2

What is the num parameter in the below line of code:
conf.setNumMapTasks(int num)

A. Number of map tasks for this job

B. Number of reducer tasks for this job

C. Number of splits for this job

D. None of the above

The correct answer is **A**

The num parameter defines the number of map tasks for this job. This is only a hint at the framework. The actual number of spawned map tasks depends on the number of InputSplits generated by the job.

Knowledge Check

3

How many phases exist in MapReduce?

A.	4
B.	5
C.	6
D.	2

Knowledge Check

3

How many phases exist in MapReduce?

A. 4

B. 5

C. 6

D. 2

The correct answer is **B**

MapReduce consists of five phases: map phase, partition phase, shuffle phase, sort phase, and reduce phase.

Knowledge Check

4

Which of the following cannot be used as an optimization technique in MapReduce?

A. Use of combiners

B. LZO compression

C. Prefer not to use writable interface for data

D. Proper configuration of cluster

Knowledge Check

4

Which of the following cannot be used as an optimization technique in MapReduce?

A. Use of combiners

B. LZO compression

C. Prefer not to use writable interface for data

D. Proper configuration of cluster

The correct answer is **C**

Prefer not to use writable interface in MapReduce is the correct option as it is not used as an optimization technique in MapReduce.

Lesson-End Project: Count the Number of Words Using MapReduce

Problem Scenario:
Philip is working in a company as a Data Engineer whose focus is to count the number of times a word occurs in a corpus. He has a copy of sample data from the internet. He wants to perform a computation with a list of words and understand how many times it occurred. This can be achieved using MapReduce.

Objective:
The objective is to use MapReduce to count the number of words from a text file.

Dataset Name: wordcount.txt
hadoop-mapreduce-example.jar

Lesson-End Project: Count the Number of Words Using MapReduce

Tasks to Perform:

1. Create a sample file with the name mapreducedemo.java
2. Create two functions Mapper and Reduce with the main method
3. Now, convert JAVA files into JAR files and give it the name **“hadoop-mapreduce-example.jar”**
4. After this, upload the JAR file into FTP
5. Upload the **“wordcount.txt”** file into HDFS and ensure to copy the path where it has been uploaded
6. Run the MapReduce code in Webconsole

Thank You