SLIIT
Discover Your Future

Hadoop and MapReduce

Year 3 – Semester 2

B.Sc. (Hons) in Information Technology Specializing in Software Engineering

Contents

- What is MapReduce ?
- MapReduce Benefits
- MapReduce Phases
- MapReduce Input/Output Format
- How MapReduce Organizes work ?
- Wordcount Example
- MapReduce Flowchart

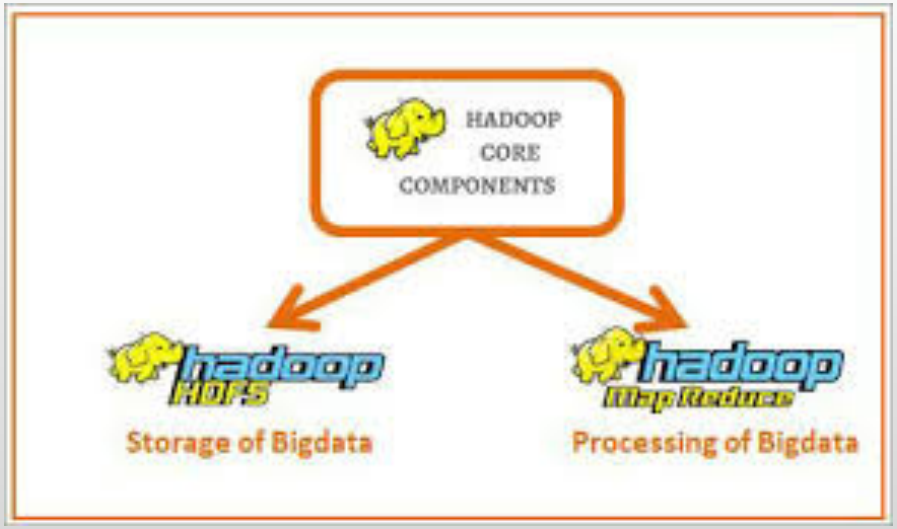
SLIIT
Discover Your Future

What is MapReduce ?

What is MapReduce?

MapReduce is a framework for writing applications that process large amounts of structured and unstructured data stored in the Hadoop Distributed File System (HDFS).

It is used to process vast amounts of data (multi-terabyte data-sets) in-parallel on large clusters (thousands of nodes) of commodity hardware in a reliable, fault- tolerant manner.



What is MapReduce?

- The MapReduce framework and the Hadoop Distributed File System run on the same set of nodes. This configuration allows the framework to effectively schedule tasks on the nodes where data is already present, resulting in very high aggregate bandwidth across the cluster.
- A MapReduce program has two components: one that implements the mapper, and another that implements the reducer.
- A MapReduce job usually splits the input data-set into independent chunks which are processed by the map tasks in a completely parallel manner.
- The framework shuffle and sorts the outputs of the maps, which are then input to the reduce tasks.

What is MapReduce?

Both the input and the output of the job are stored in a file-system. The framework takes care of scheduling tasks, monitoring them and re-executes the failed tasks.

In MapReduce, users write a client application that submits one or more jobs that contain user-supplied map and reduce code and a job configuration file to a cluster of machines.

The job contains a map function and a reduce function, along with job configuration information that controls various aspects of its execution.

The map and reduce functions in Hadoop MapReduce have the following general form:

Map: (K1, V1) → list (K2, V2)

Reduce: (K2, list (V2)) → list (K3, V3)

SLIIT
Discover Your Future

MapReduce Benefits

MapReduce Benefits

MapReduce is useful for batch processing on terabytes or petabytes of data stored in Apache Hadoop.
The following tables describes some of MapReduce's key benefits:

Benefit	Description
Simplicity	Developers can write applications in their language of choice, such as Java, C++ or Python, and MapReduce jobs are easy to run
Scalability	MapReduce can process petabytes of data, stored in HDFS on one cluster
Speed	Parallel processing means that MapReduce can take problems that used to take days to solve and solve them in hours or minutes
Recovery	MapReduce takes care of failures. If a machine with one copy of the data is unavailable, another machine has a copy of the same key/value pair, which can be used to solve the same sub-task. The JobTracker keeps track of it all.
Minimal data motion	MapReduce moves compute processes to the data on HDFS and not the other way around. Processing tasks can occur on the physical node where the data resides. This significantly reduces the network I/O patterns and contributes to Hadoop's processing speed.

SLIIT
Discover Your Future

MapReduce Input/Output Format

SLIIT
Discover Your Future

MapReduce Input/Output Format

InputFormat:

How these input files are split up and read is defined by the InputFormat. An InputFormat is a class that provides the following functionality:

- Selects the files or other objects that should be used for input.
- Defines the InputSplits that break a file into tasks.
- Provides a factory for RecordReader objects that read the file.

An InputFormat is responsible for creating the input splits and dividing them into records

We can choose InputFormat to apply to our input files for a job by calling the **setInputFormat()** method of the JobConf object that defines the job.

MapReduce Input/Output Format

A table of standard InputFormat is given below.

InputFormat	Description	Key	Value
TextInputFormat	Default format; reads lines of text files	The byte offset of the line	The line contents
KeyValueInputFormat	Parses lines into key	Everything up to the first tab character	The remainder of the line
SequenceFileInputFormat	A Hadoop-specific high-performance binary format	User-defined	User-defined

SLIIT
Discover Your Future

MapReduce Input/Output Format

A table of standard OutputFormat is given below.

InputFormat	Description
TextOutputFormat	Default; writes lines in ”key \t value” form
SequenceFileOutputFormat	Writes binary files suitable for reading into subsequent MapReduce jobs
NullOutputFormat	Disregards its inputs

SLIIT
Discover Your Future

MapReduce Phases

MapReduce Phases

- **Input Splits :**

An input to a MapReduce job is divided into fixed-size pieces called **input splits.** Input split is a chunk of the input that is consumed by a single map.

- **Mapping :**

This is the very first phase in the execution of map-reduce program. In this phase data in each split is passed to a mapping function to produce output values. In our example, a job of mapping phase is to count a number of occurrences of each word from input splits and prepare a list in the form of <word, frequency>

- **Shuffling :**

This phase consumes the output of Mapping phase. Its task is to consolidate the relevant records from Mapping phase output. In our example, the same words are clubbed together along with their respective frequency.

- **Reducing :**

In this phase, output values from the Shuffling phase are aggregated. This phase combines values from Shuffling phase and returns a single output value. In short, this phase summarizes the complete dataset.

SLIIT
Discover Your Future

How MapReduce Organizes work?

MapReduce Phases

Hadoop divides the job into tasks. There are two types of tasks:

- **Map tasks** (Splits & Mapping)
- **Reduce tasks** (Shuffling, Reducing)

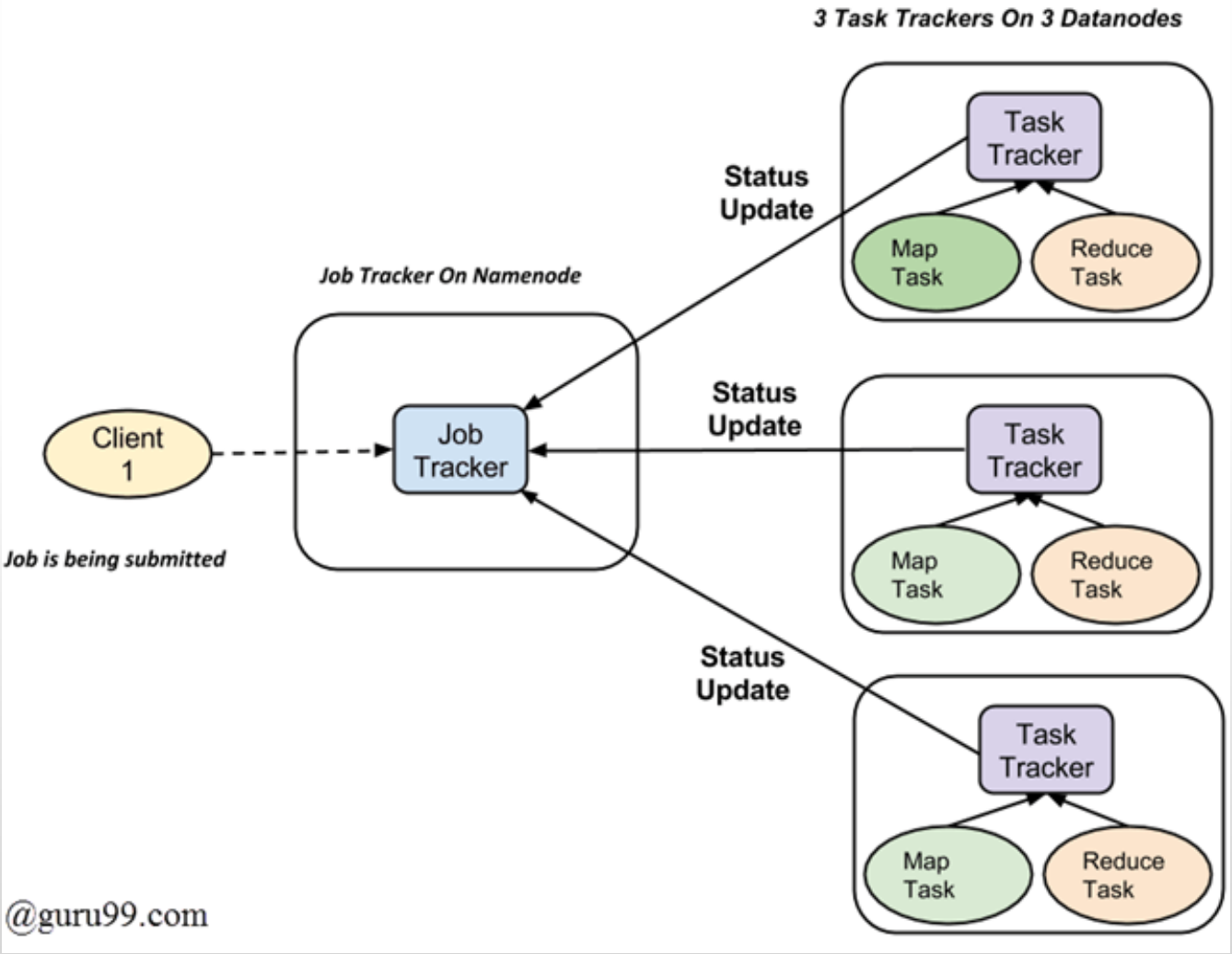
The complete execution process (execution of Map and Reduce tasks, both) is controlled by two types of entities called a

- **Jobtracker**: Acts like a **master** (responsible for complete execution of submitted job)
- **Multiple Task Trackers**: Acts like **slaves,** each of them performing the job

For every job submitted for execution in the system, there is one **Jobtracker** that resides on **Namenode** and there are **multiple tasktrackers** which reside on **Datanode**.

SLIIT
Discover Your Future

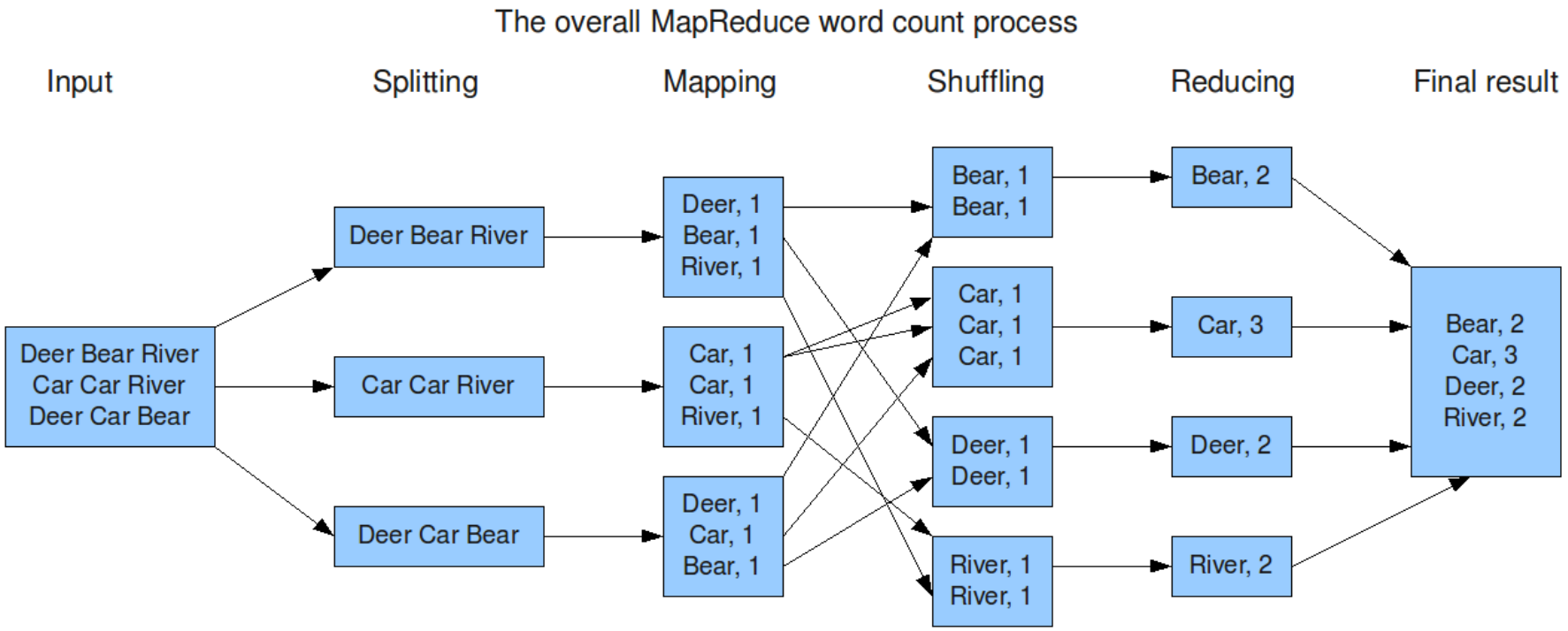
MapReduce Phases



WordCount Example

SLIIT
Discover Your Future

WordCount Example

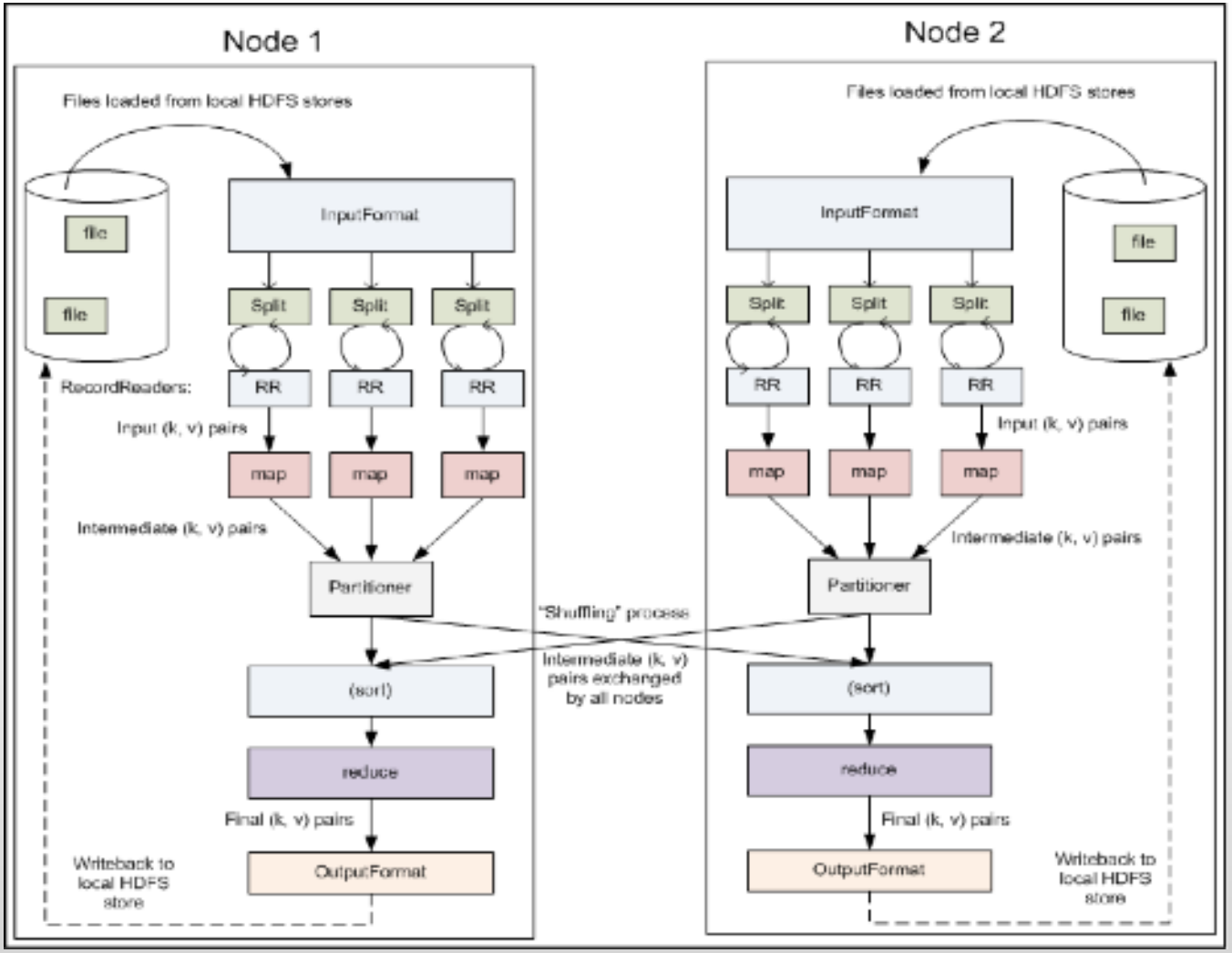


SLIIT
Discover Your Future

MapReduce FlowChart

SLIIT
Discover Your Future

MapReduce FlowChart



Thank you …