MAP REDUCE

AMRITPAL SINGH

Source : https://data-flair.training/blogs/how-hadoop-mapreduce-works/

MOTIVATION

- How to break up a large problem into smaller tasks, that can be executed in parallel?
  - How to assign tasks to machines?
  - How to partition and distribute data?
  - How to share intermediate results?
  - How to coordinate synchronization, scheduling, fault-tolerance?

Source : https://data-flair.training/blogs/how-hadoop-mapreduce-works/

INTRODUCTION

- revolution is happening right now in computing.
- Most computers shipped today use multi-core microprocessors, i.e., chips with 2 (or 4, or 8, or more) separate processing units on the main microprocessor.
- As computing with multiple processors becomes widespread, though, we're seeing a flowering of general-purpose software development tools tailored to multiple processor environments.

Source : https://data-flair.training/blogs/how-hadoop-mapreduce-works/

INTRODUCTION

- One of the organizations driving this shift is Google.
- Google is one of the largest users of multiple processor computing in the world, with its entire computing cluster containing hundreds of thousands of commodity machines, located in data centers around the world, and linked using commodity networking components.
- This approach to multiple processor computing is known as *distributed computing*;

Source : https://data-flair.training/blogs/how-hadoop-mapreduce-works/

INTRODUCTION

- MapReduce was introduced in a paper written in 2004 by Jeffrey Dean and Sanjay Ghemawat from Google.

- What's beautiful about MapReduce is that it makes parallelization almost entirely invisible to the programmer who is using MapReduce to develop applications.

Source : https://data-flair.training/blogs/how-hadoop-mapreduce-works/

INTRODUCTION

- **MapReduce** is the processing layer of **Hadoop**.
- MapReduce programming model is designed for processing large volumes of data in parallel by dividing the work into a set of independent tasks.
- Work (complete job) which is submitted by the user to master is divided into small works (tasks) and assigned to slaves.

Source : https://data-flair.training/blogs/how-hadoop-mapreduce-works/

INTRODUCTION

- A problem is divided into a large number of smaller problems each of which is processed to give individual outputs.
- These individual outputs are further processed to give final output.

Source : https://data-flair.training/blogs/how-hadoop-mapreduce-works/

MapReduce Terminologies

- A Map-Reduce program will do this twice, using two different list processing idioms-

- Map
- Reduce

- In between Map and Reduce, there is small phase called **Shuffle** and **Sort** in MapReduce.

Source : https://data-flair.training/blogs/how-hadoop-mapreduce-works/

MapReduce Terminologies : What is a MapReduce Job?

- MapReduce Job or a A “full program” is an execution of a **Mapper** and **Reducer** across a data set.
- It is an execution of 2 processing layers i.e mapper and reducer.
- A MapReduce job is a work that the client wants to be performed.
- It consists of the input data, the MapReduce Program, and configuration info.

Source : https://data-flair.training/blogs/how-hadoop-mapreduce-works/

MapReduce Terminologies :What is Task in Map Reduce?

- A task in MapReduce is an execution of a Mapper or a Reducer on a slice of data. It is also called Task-In-Progress.

- It means processing of data is in progress either on mapper or reducer.

Source : https://data-flair.training/blogs/how-hadoop-mapreduce-works/

MapReduce Terminologies :What is Task Attempt?

- Task Attempt is a particular instance of an attempt to execute a task on a node.
- There is a possibility that anytime any machine can go down.
- For example, while processing data if any node goes down, framework reschedules the task to some other node.
- This rescheduling of the task cannot be infinite. There is an upper limit for that as well. The default value of task attempt is 4.

Source : https://data-flair.training/blogs/how-hadoop-mapreduce-works/

Map Abstraction

- The **map** takes **key/value pair** as input.
- Whether data is in structured or unstructured format, framework converts the incoming data into key and value.
- Key is a reference to the input value.
- Value is the data set on which to operate.

Source : https://data-flair.training/blogs/how-hadoop-mapreduce-works/

Map Abstraction

- **Map Processing:**
- A function defined by user – user can write custom business logic according to his need to process the data.
- Applies to every value in value input.

Source : https://data-flair.training/blogs/how-hadoop-mapreduce-works/

Map Abstraction

- **Map produces a new list of key/value pairs:**

- An output of Map is called intermediate output.
- Can be the different type from input pair.

- An output of map is stored on the local disk from where it is shuffled to reduce nodes.

Source : https://data-flair.training/blogs/how-hadoop-mapreduce-works/

Reduce Abstraction

- **Reduce** takes intermediate Key / Value pairs as input and processes the output of the mapper.
- Usually, in the reducer, we do aggregation or summation sort of computation.
- Input given to reducer is generated by Map (intermediate output)
- Key / Value pairs provided to reduce are sorted by key.

Source : https://data-flair.training/blogs/how-hadoop-mapreduce-works/

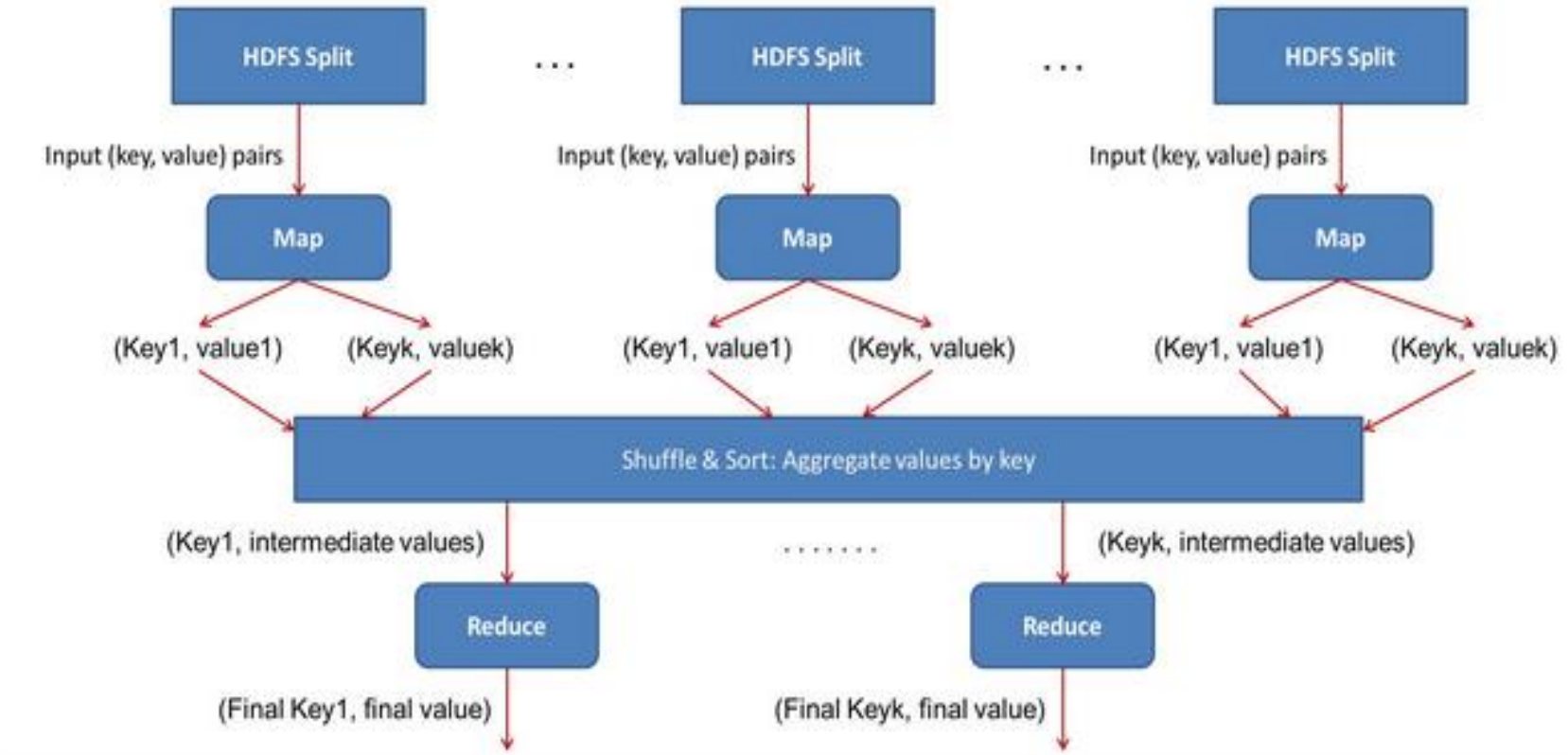
Reduce Abstraction

- **Reduce processing:**
- A function defined by user – Here also user can write custom business logic and get the final output.
- Iterator supplies the values for a given key to the Reduce function.

- **Reduce produces a final list of key/value pairs:**
- An output of Reduce is called Final output.
- It can be a different type from input pair.
- An output of Reduce is stored in **HDFS**.

Source : https://data-flair.training/blogs/how-hadoop-mapreduce-works/

How Map and Reduce work Together?



Source : https://data-flair.training/blogs/how-hadoop-mapreduce-works/

How Map and Reduce work Together?

- Why there are more number of mappers as compared to number of reducers ?

Source : https://data-flair.training/blogs/how-hadoop-mapreduce-works/

How Map and Reduce work Together?

- Input data given to **mapper** is processed through user defined function written at mapper.
- All the required complex business logic is implemented at the mapper level so that heavy processing is done by the mapper in parallel as the number of mappers is much more than the number of **reducers**.
- Mapper generates an output which is intermediate data and this output goes as input to reducer.

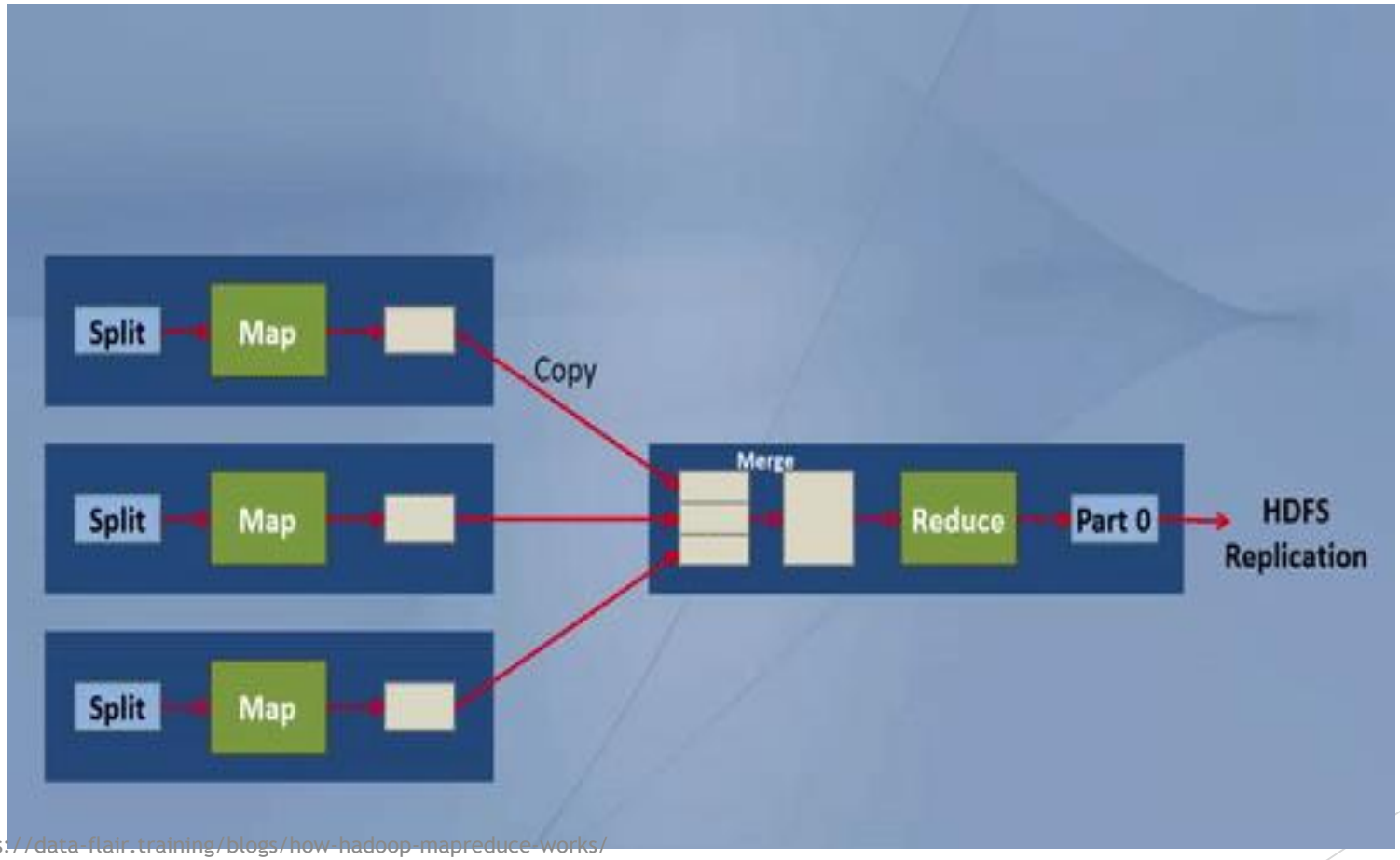
Source : https://data-flair.training/blogs/how-hadoop-mapreduce-works/

How Map and Reduce work Together?

- This intermediate result is then processed by user defined function written at reducer and final output is generated.
- Usually, in reducer very light processing is done.
- This final output is stored in **HDFS** and replication is done as usual.

Source : https://data-flair.training/blogs/how-hadoop-mapreduce-works/

MapReduce DataFlow



Source : https://data-flair.training/blogs/how-hadoop-mapreduce-works/

MapReduce DataFlow

- There are 3 slaves in the figure.
- On all 3 slaves mappers will run, and then a reducer will run on any 1 of the slave.
- An input to a mapper is 1 **block** at a time. (Split = block by default)
- **Mapper** writes the output to the local disk of the machine it is working. This is the temporary data.
- An output of mapper is also called intermediate output.

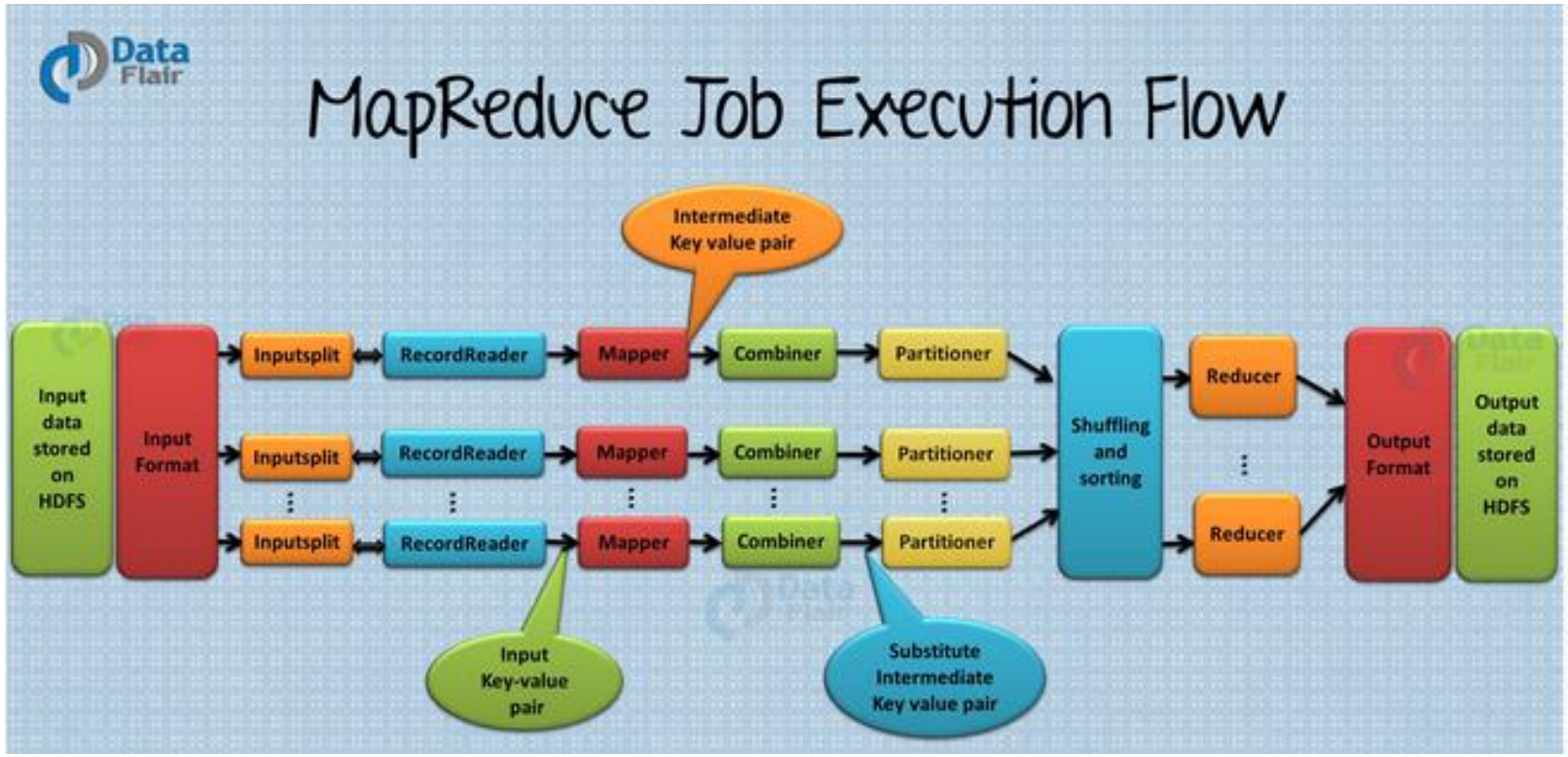
Source : https://data-flair.training/blogs/how-hadoop-mapreduce-works/

MapReduce DataFlow

- movement of output from mapper node to reducer node is called **shuffle**.
- Map and reduce are the stages of processing.
- They run one after other.
- After all, mappers complete the processing, then only reducer starts processing.

Source : https://data-flair.training/blogs/how-hadoop-mapreduce-works/

MapReduce DataFlow : Internal Working



Source : https://data-flair.training/blogs/how-hadoop-mapreduce-works/

MapReduce DataFlow : Internal Working

- There are various phases of MapReduce job execution such as **Input Files**, **InputFormat in Hadoop**, **InputSplits**, **RecordReader**, **Mapper**, **Combiner**, **Partitioner**, **Shuffling and Sorting**, **Reducer**, **RecordWriter and OutputFormat.**

Source : https://data-flair.training/blogs/how-hadoop-mapreduce-works/

MapReduce DataFlow : Internal Working

- The data for a MapReduce task is stored in **input files**, and input files typically lives in **HDFS**.
- **InputFormat** defines how these input files are split and read.
- Input Splits is created by InputFormat, logically represent the data which will be processed by an individual **Mapper.** One map task is created for each split; thus the number of map tasks will be equal to the number of InputSplits.
- The split is divided into records and each record will be processed by the mapper

Source : https://data-flair.training/blogs/how-hadoop-mapreduce-works/

MapReduce DataFlow : Internal Working (Types of Input Format in Map Reduce)

Source : https://data-flair.training/blogs/how-hadoop-mapreduce-works/

MapReduce DataFlow : Internal Working

- Record Reader communicates with the **InputSplit** in Hadoop MapReduce and converts the data into **key-value pairs** suitable for reading by the mapper.
- Mapper processes each input record (from RecordReader) and generates new key-value pair, and this key-value pair generated by Mapper is completely different from the input pair.
- The output of Mapper is also known as intermediate output which is written to the local disk.

Source : https://data-flair.training/blogs/how-hadoop-mapreduce-works/

MapReduce DataFlow : Internal Working

- The combiner is also known as ‘Mini-reducer’. Hadoop MapReduce Combiner performs local aggregation on the mappers’ output, which helps to minimize the data transfer between mapper and **reducer.**

- Hadoop MapReduce, **Partitioner** comes into the picture if we are working on more than one reducer (for one reducer partitioner is not used).

- According to the key value in MapReduce, each combiner output is partitioned, and a record having the same key value goes into the same partition, and then each partition is sent to a reducer.

Source : https://data-flair.training/blogs/how-hadoop-mapreduce-works/

MapReduce DataFlow : Internal Working

- the output is Shuffled to the reduce node (which is a normal slave node but reduce phase will run here hence called as reducer node). The shuffling is the physical movement of the data which is done over the network.

- Once all the mappers are finished and their output is shuffled on the reducer nodes, then this intermediate output is merged and sorted, which is then provided as input to reduce phase.

- Reducer takes the set of intermediate key-value pairs produced by the mappers as the input and then runs a reducer function on each of them to generate the output.

Source : https://data-flair.training/blogs/how-hadoop-mapreduce-works/

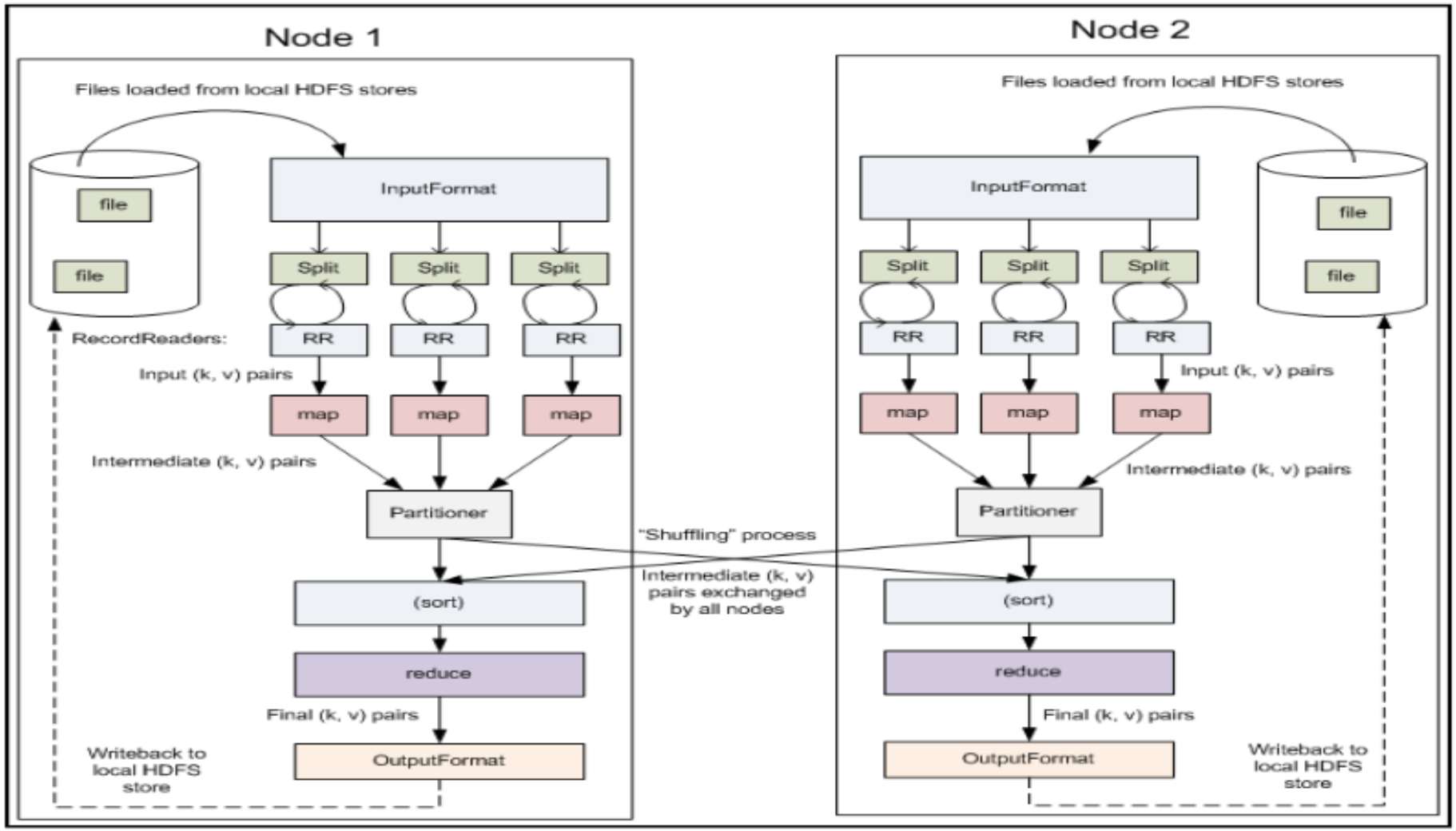
MapReduce DataFlow : Internal Working

- **RecordWriter** writes these output key-value pair from the Reducer phase to the output files.

- The way these output key-value pairs are written in output files by RecordWriter is determined by the OutputFormat.

Source : https://data-flair.training/blogs/how-hadoop-mapreduce-works/

Detailed Hadoop MapReduce data flow



Source : https://data-flair.training/blogs/how-hadoop-mapreduce-works/

Data Locality in MapReduce

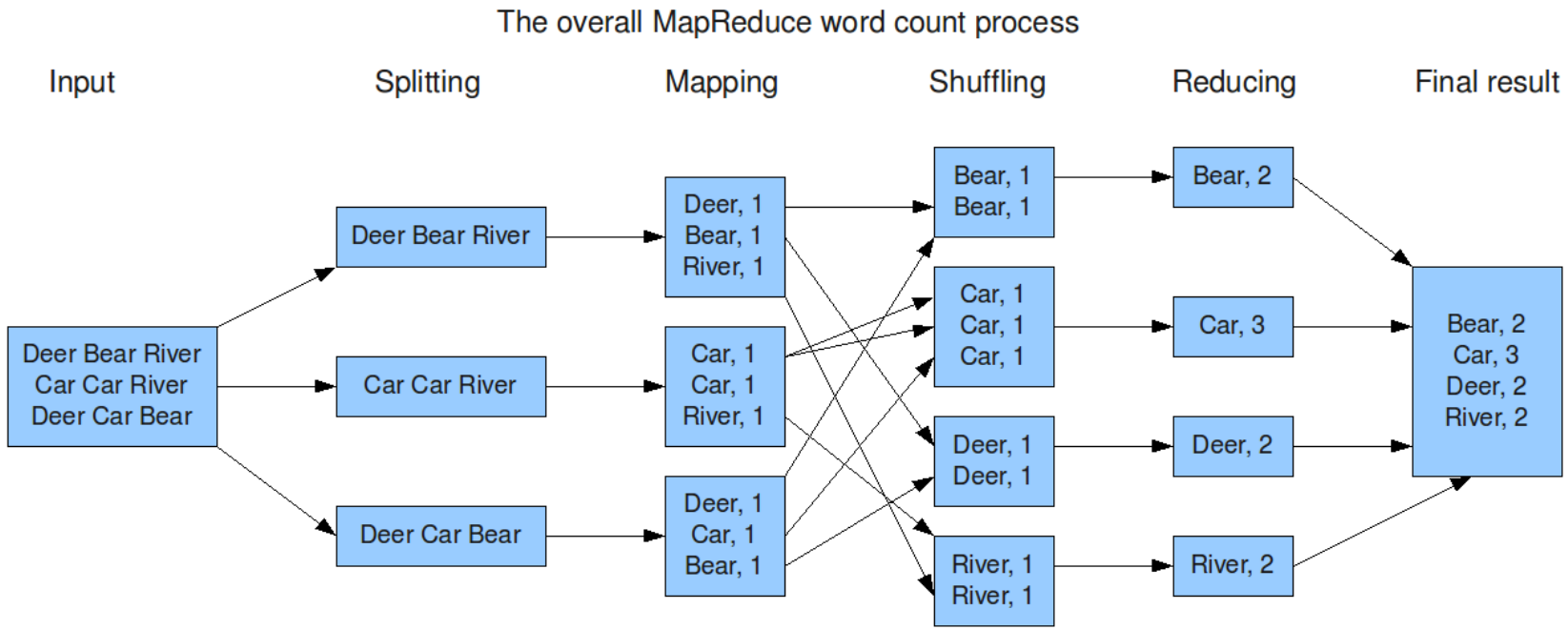
- "*Move computation close to the data rather than data to computation*". A computation requested by an application is much more efficient if it is executed near the data it operates on.

- This is especially true when the size of the data is very huge. This minimizes network congestion and increases the throughput of the system.

- Hadoop has come up with the most innovative principle of moving algorithm to data rather than data to algorithm. This is called **data locality**.

Source : https://data-flair.training/blogs/how-hadoop-mapreduce-works/

Example : Map Reduce (Word Count)



Source : https://data-flair.training/blogs/how-hadoop-mapreduce-works/

Example: Word Count (R)

- sentences<-scan("C:/Users/Amritpal/Desktop/tr.txt","character",sep="\n");
- sentences<-gsub("\\.","",sentences)
- sentences<-gsub("\\,","",sentences)
- words<-strsplit(sentences," ")
- words.freq<-table(words);
- cbind(names(words.freq),as.integer(words.freq))

Source : https://data-flair.training/blogs/how-hadoop-mapreduce-works/

Applications

What is MapReduce used for?

- **At Google**
  - Index construction for Google Search (replaced in 2010 by Caffeine)
  - Article clustering for Google News
  - Statistical machine translation
- **At Yahoo!**
  - "Web map" powering Yahoo! Search
  - Spam detection for Yahoo! Mail
- **At Facebook**
  - Data mining, Web log processing
  - SearchBox (with Cassandra)
  - Facebook Messages (with HBase)
  - Ad optimization
  - Spam detection

Source : https://data-flair.training/blogs/how-hadoop-mapreduce-works/

Map Reduce in R

- RMR package

Source : https://data-flair.training/blogs/how-hadoop-mapreduce-works/

Questions

- **What is the fundamental difference between a MapReduce Split and a HDFS block?**

- MapReduce split is a logical piece of data fed to the mapper. It basically does not contain any data but is just a pointer to the data. HDFS block is a physical piece of data.

- **When is it not recommended to use MapReduce paradigm for large scale data processing?**

- It is not suggested to use MapReduce for iterative processing use cases, as it is not cost effective, instead Apache Pig can be used for the same.

Source : https://data-flair.training/blogs/how-hadoop-mapreduce-works/

Questions

- **What is the relationship between Job and Task in Hadoop?**
- A single job can be broken down into one or many tasks in Hadoop.
- **Is it important for Hadoop MapReduce jobs to be written in Java?**
- It is not necessary to write Hadoop MapReduce jobs in Java but users can write MapReduce jobs in any desired programming language like Ruby, Perl, Python, R, etc. through the Hadoop Streaming API.

Source : https://data-flair.training/blogs/how-hadoop-mapreduce-works/

Questions

- **For a job in Hadoop, is it possible to change the number of mappers to be created?**

- No, it is not possible to change the number of mappers to be created. The number of mappers is determined by the number of input splits.

Source : https://data-flair.training/blogs/how-hadoop-mapreduce-works/

MCQ's

- What are supported programming languages for Map Reduce? | Hadoop

- A. The most common programming language is Java, but scripting languages are also supported via Hadoop streaming.
  B. Any programming language that can comply with Map Reduce concept can be supported.
  C. Only Java supported since Hadoop was written in Java.
  D. Currently Map Reduce supports Java, C, C++ and COBOL.

Source : https://data-flair.training/blogs/how-hadoop-mapreduce-works/

MCQ's

- What is the implementation language of the Hadoop MapReduce framework? | Hadoop

- A. Java
  B. C
  C. FORTRAN
  D. Python

Source : https://data-flair.training/blogs/how-hadoop-mapreduce-works/

MCQ's

- How can you disable the reduce step? | Hadoop

- A. The Hadoop administrator has to set the number of the reducer slot to zero on all slave nodes. This will disable the reduce step.
  B. It is imposible to disable the reduce step since it is critical part of the Mep-Reduce abstraction.
  C. A developer can always set the number of the reducers to zero. That will completely disable the reduce step.
  D. While you cannot completely disable reducers you can set output to one. There needs to be at least one reduce step in Map-Reduce abstraction.

Source : https://data-flair.training/blogs/how-hadoop-mapreduce-works/

MCQ's

- Why would a developer create a map-reduce without the reduce step? | Hadoop

- A. Developers should design Map-Reduce jobs without reducers only if no reduce slots are available on the cluster.
  B. Developers should never design Map-Reduce jobs without reducers. An error will occur upon compile.
  C. There is a CPU intensive step that occurs between the map and reduce steps. Disabling the reduce step speeds up data processing.
  D. It is not possible to create a map-reduce job without at least one reduce step. A developer may decide to limit to one reducer for debugging purposes.

Source : https://data-flair.training/blogs/how-hadoop-mapreduce-works/

MCQ's

- What is the default input format? | Hadoop

- A. The default input format is xml. Developer can specify other input formats as appropriate if xml is not the correct input.
  B. There is no default input format. The input format always should be specified.
  C. The default input format is a sequence file format. The data needs to be preprocessed before using the default input format.
  D. The default input format is TextInputFormat with byte offset as a key and entire line as a value.

Source : https://data-flair.training/blogs/how-hadoop-mapreduce-works/

MCQ's

- What happens if mapper output does not match reducer input? | Hadoop

- A. Hadoop API will convert the data to the type that is needed by the reducer.
  B. Data input/output inconsistency cannot occur. A preliminary validation check is executed prior to the full execution of the job to ensure there is consistency.
  C. The java compiler will report an error during compilation but the job will complete with exceptions.
  D. A real-time exception will be thrown and map-reduce job will fail.

Source : https://data-flair.training/blogs/how-hadoop-mapreduce-works/

MCQ's

- Which of the following is a valid flow in Hadoop ?

  a. Input -> Reducer -> Mapper -> Combiner -> -> Output
  b. Input -> Mapper -> Reducer -> Combiner -> Output
  c. Input -> Mapper -> Combiner -> Reducer -> Output
  d. Input -> Reducer -> Combiner -> Mapper -> Output

Source : https://data-flair.training/blogs/how-hadoop-mapreduce-works/

MCQ's

- MapReduce was devised by ...

  a. Apple
  b. Google
  c. Microsoft
  d. Samsung

Source : https://data-flair.training/blogs/how-hadoop-mapreduce-works/

MCQ's

- Which of the following is not an input format in Hadoop ?

  a. TextInputFormat
  b. ByteInputFormat
  c. SequenceFileInputFormat
  d. KeyValueInputFormat

Source : https://data-flair.training/blogs/how-hadoop-mapreduce-works/

TRUE/FALSE

- *All MapReduce implementations implement exactly same algorithm.*

- The answer is:
- False.
- For example, Google's implementation does not allow change of key in the reducer, but provides sorting for values. Hadoop does not provide values sorting, but reducer can change the key.

Source : https://data-flair.training/blogs/how-hadoop-mapreduce-works/

TRUE/FALSE

- *Each mapper must generate the same number of key/value pairs as its input had.*

- The answer is:
- False.
- Mapper may generate any number of key/value pairs (including zero).

Source : https://data-flair.training/blogs/how-hadoop-mapreduce-works/

TRUE/FALSE

- *Mappers input key/value pairs are sorted by the key.*

- The answer is:
- False.
- Mapper’s input is not sorted in any way.

Source : https://data-flair.training/blogs/how-hadoop-mapreduce-works/

TRUE/FALSE

- *Mappers output key/value must be of the same type as its input.*

- The answer is:
- False.
- Mapper may produce key/value pairs of any type.

Source : https://data-flair.training/blogs/how-hadoop-mapreduce-works/