Understanding MapReduce
UNIT III

MapReduce Framework

- Big Data can be defined as a huge dataset or collection of such huge datasets that cannot be processed by traditional systems.

- Big Data has become a whole subject in itself which consists of a study of different tools, techniques and frameworks rather than just data.

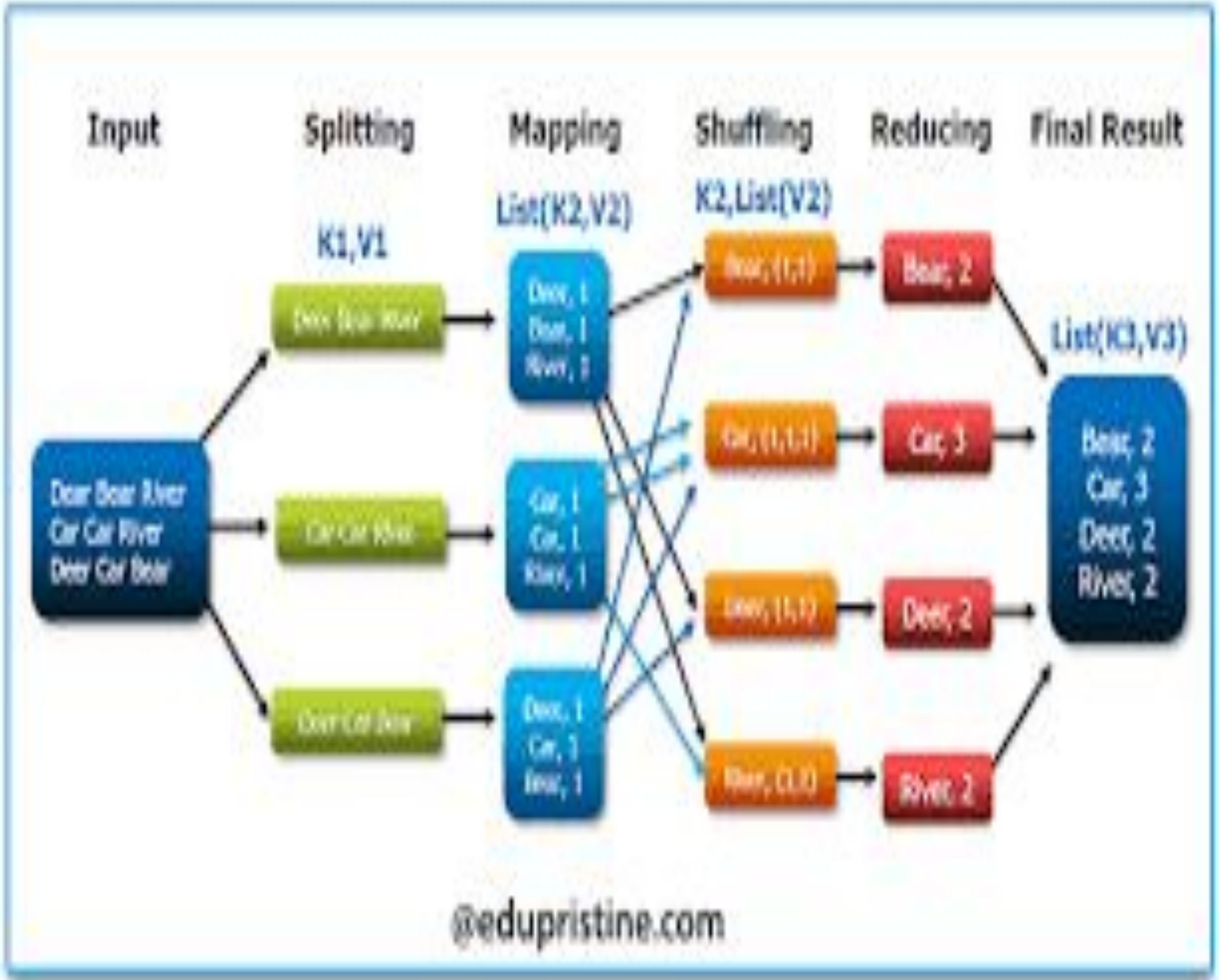
- MapReduce is a framework which is used for making applications that help us with processing of huge volume of data on a large cluster of commodity hardware.

Why MapReduce?

- Traditional systems tend to use a centralized server for storing and retrieving data.
- Such huge amount of data cannot be accommodated by standard database servers.
- Also, centralized systems create too much of a bottleneck while processing multiple files simultaneously.
- Google, came up with MapReduce to solve such bottleneck issues.
- MapReduce will divide the task into small parts and process each part independently by assigning them to different systems.
- After all the parts are processed and analyzed, the output of each computer is collected in one single location and then an output dataset is prepared for the given problem.

How does MapReduce works?

- MapReduce is a programming paradigm or model used to process large datasets with a parallel distributed algorithm on a cluster
- In Big Data Analytics, MapReduce plays a crucial role.
- When it is combined with HDFS we can use MapReduce to handle Big Data
- The basic unit of information used by MapReduce is a key-value pair.
- All the data whether structured or unstructured needs to be translated to the key-value pair before it is passed through the MapReduce model.
- MapReduce model as the name suggests has two different functions; Map-function and Reduce-function. The order of operation is always **Map|Shuffle|Reduce**
-

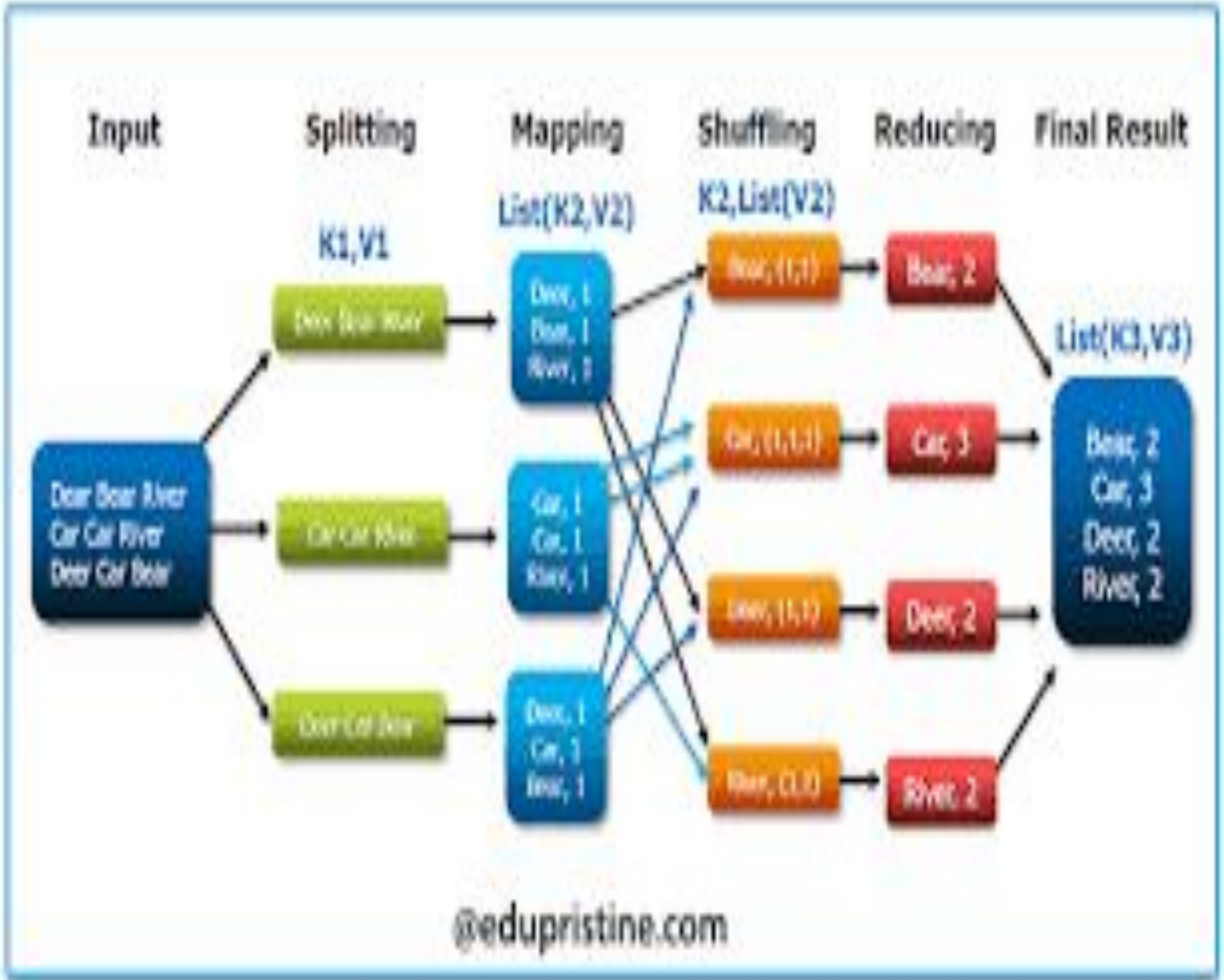

Input
Splitting
Mapping
Shuffling
Reducing
Final Result
K1,V1
List(K2,V2)
K2,List(V2)
List(K3,V3)
Deer Bear River
Car Car River
Deer Car Bear
Deer Bear River
Car Car River
Deer Car Bear
Deer, 1
Bear, 1
River, 1
Car, 1
Car, 1
River, 1
Deer, 1
Car, 1
Bear, 1
Bear, (1,1)
Car, (1,1,1)
Deer, (1,1)
River, (1,1)
Bear, 2
Car, 3
Deer, 2
River, 2
Bear, 2
Car, 3
Deer, 2
River, 2
@edupristine.com

Map stage

- Map stage is the crucial step in the MapReduce framework.
- Mapper will give a structure to the unstructured data.
- For example, if I have to count the songs and music files in my laptop as per genre in my playlist, I will have to analyze the unstructured data.
- The mapper makes key-value pairs from this dataset.
- So, in this case, the key is genre and value is the music file.
- Once all this data is given to the mapper, we have a whole dataset ready with the key-value pair structure.
- So the mapper will work on one key-value pair at a time.
- One input may produce any number of outputs.
- Basically, Map-function will process the data and make several small chunks of data.
-

Reduce stage

- The Shuffle stage and the Reduce stage together are called the Reduce stage.
- Reducer will take the output from the mapper as an input and make the final output as specified by the programmer.
- This new output will be saved to the HDFS.
- The Reducer will take all the key-value pairs from the mapper and check the association of all keys with value.
- All the values associated with a single key will be taken and it will provide an output of any number of key-value pairs.
- By understanding the Map and Reduce stages, we understand that MapReduce is a sequential computation.
- For any Reducer to work, the Mapper must have completed the execution.
- If that is not the case, the Reducer stage won't run.
- Since, the Reducer will have access to all the values, we can say that it will find all values with the same key and perform computations on them.
- So what actually happens is, since reducers are working on different keys, they are made to work simultaneously and parallelism is achieved.
- Let us understand this by an example:
-


Input
Splitting
Mapping
Shuffling
Reducing
Final Result
K1,V1
List(K2,V2)
K2,List(V2)
List(K3,V3)
Deer Bear River
Car Car River
Deer Car Bear
Deer Bear River
Car Car River
Deer Car Bear
Deer, 1
Bear, 1
River, 1
Car, 1
Car, 1
River, 1
Deer, 1
Car, 1
Bear, 1
Bear, (1,1)
Car, (1,1,1)
Deer, (1,1)
River, (1,1)
Bear, 2
Car, 3
Deer, 2
River, 2
Bear, 2
Car, 3
Deer, 2
River, 2
@edupristine.com

- Suppose we have 4 sentences for processing:
- Red, Green, Blue, Red, Blue
- Green, Brown, Red, Yellow
- Yellow, Blue, Green, Orange
- Yellow, Orange, Red, Blue
- When such an input is passed into the mapper, mapper will divide those into two different subsets.
- First will be the subset of the first two sentences and second, the subset of remaining two sentences. Now, Mapper has:
- Subset 1: Red, Green, Blue, Red, Blue and Green, Brown, Red, Yellow
- Subset 2: Yellow, Blue, Green, Orange and Yellow, Orange, Red, Blue
- Mapper will make the key-value pair for each subset.
- For our example, key is the colour and value is the number of times they have appeared.
- So, we will have key- value pairs for subset 1 as (Red, 1), (Green, 1), (Blue, 1) and so on. Similarly for subset 2.
- Once this is done, the key-value pairs are given to the reducer as input.
- So, reducer will give us the final count of all the colours in our input subsets and then combine the two outputs.
- Reducer output will be, (Red, 4), (Green, 3), (Blue, 4), (Brown, 1), (Yellow, 3), (Orange, 2).
-

Exploring the Features of MapReduce

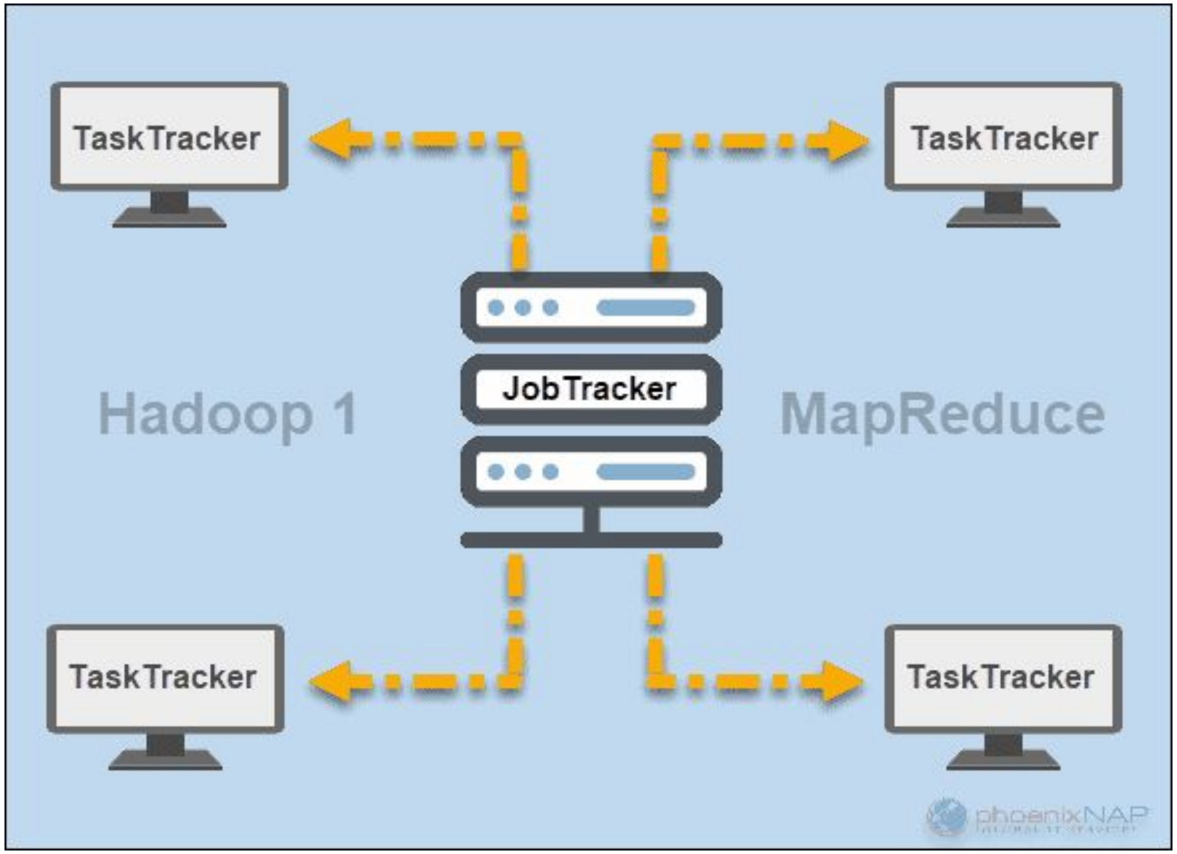
- ✔ Scheduling
- ✔ Synchronization
- ✔ Co-location of code
- ✔ Handling of errors
- ✔ Scale out architecture

Working of MapReduce

- At a high level, MapReduce breaks input data into fragments and distributes them across different machines.
- The input fragments consist of key-value pairs.
- Parallel map tasks process the chunked data on machines in a cluster.
- The mapping output then serves as input for the reduce stage.
- The reduce task combines the result into a particular key-value pair output and writes the data to HDFS.
- The Hadoop Distributed File System usually runs on the same set of machines as the MapReduce software.
- When the framework executes a job on the nodes that also store the data, the time to complete the tasks is reduced significantly.
-

Basic Terminology of Hadoop MapReduce

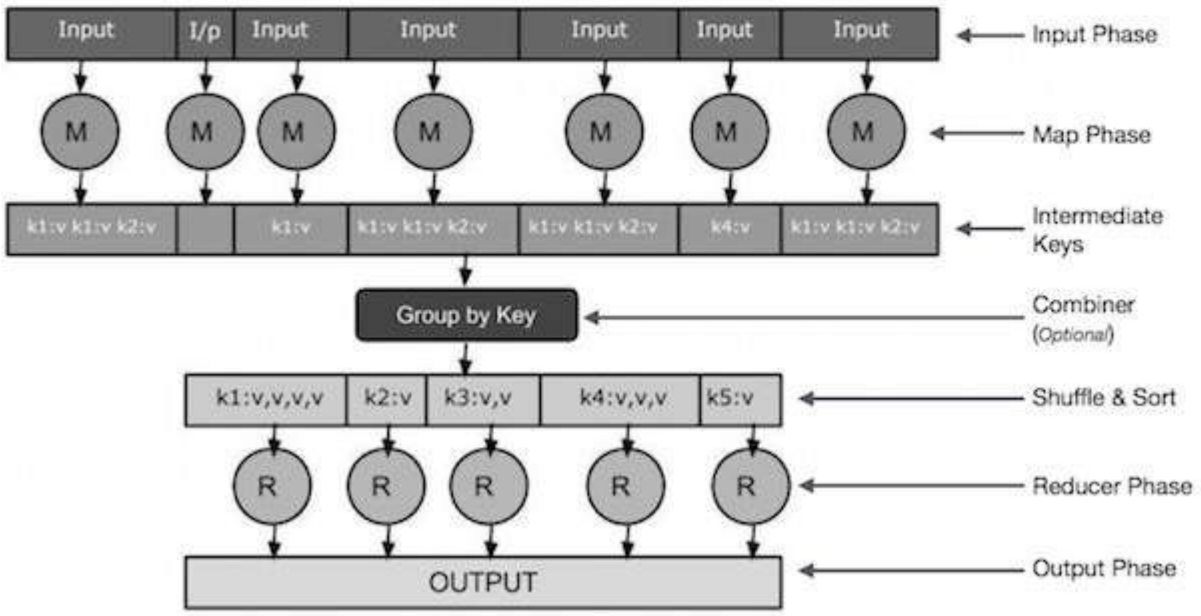
- MapReduce is a processing layer in a Hadoop environment.
- MapReduce works on tasks related to a job.
- The idea is to tackle one large request by slicing it into smaller units.
- **JobTracker and TaskTracker**
- In the early days of Hadoop (version 1), **JobTracker** and **TaskTracker** daemons ran operations in MapReduce.
- At the time, a Hadoop cluster could only support MapReduce applications.
- A **JobTracker** controlled the distribution of application requests to the compute resources in a cluster.
- Since it monitored the execution and the status of MapReduce, it resided on a master node.
- A **TaskTracker** processed the requests that came from the JobTracker.
- All task trackers were distributed across the slave nodes in a Hadoop cluster.


TaskTracker
TaskTracker
Hadoop 1
JobTracker
MapReduce
TaskTracker
TaskTracker
phoenixNAP

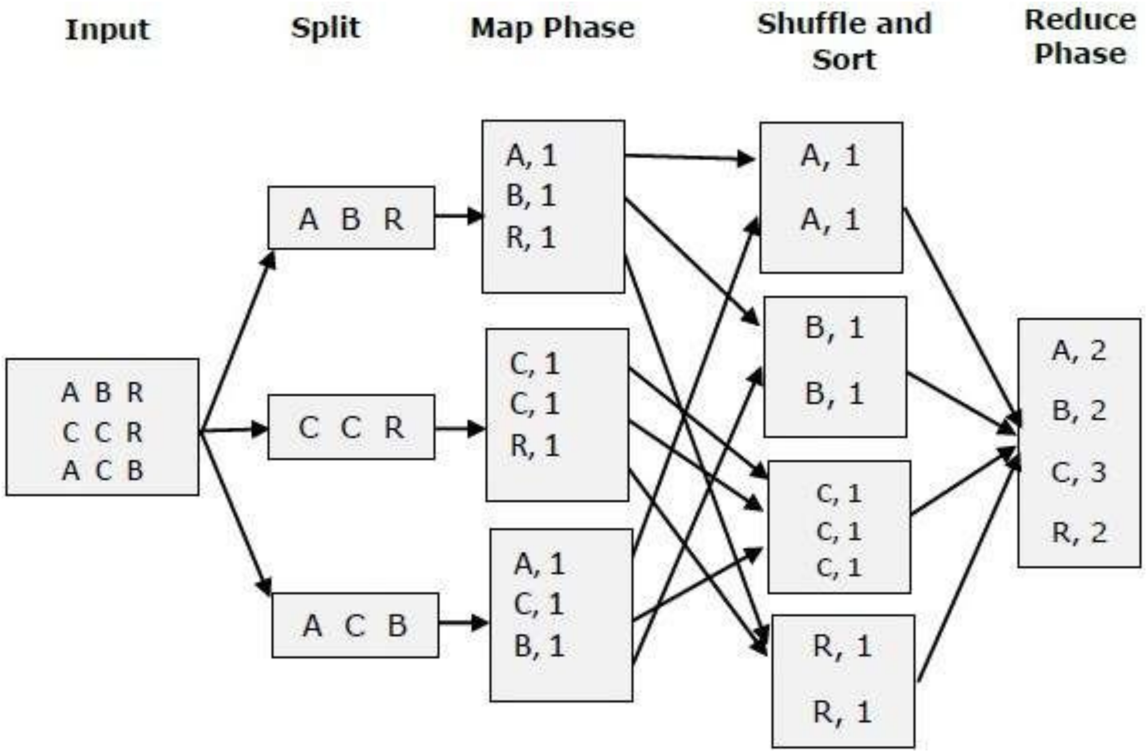
MapReduce Job

- A **MapReduce job** is the top unit of work in the MapReduce process.
- It is an assignment that Map and Reduce processes need to complete.
- A job is divided into smaller tasks over a cluster of machines for faster execution.
- The tasks should be big enough to justify the task handling time.
- If you divide a job into unusually small segments, the total time to prepare the splits and create tasks may outweigh the time needed to produce the actual job output.
- **MapReduce Task**
- MapReduce jobs have two types of **tasks.**
- A **Map Task** is a single instance of a MapReduce app.
- These tasks determine which records to process from a data block.
- The input data is split and analyzed, in parallel, on the assigned compute resources in a Hadoop cluster. This step of a MapReduce job prepares the <key, value> pair output for the reduce step.
- A **Reduce Task** processes an output of a map task. Similar to the map stage, all reduce tasks occur at the same time, and they work independently.
- The data is aggregated and combined to deliver the desired output.
- The final result is a reduced set of <key, value> pairs which MapReduce, by default, stores in HDFS.
-

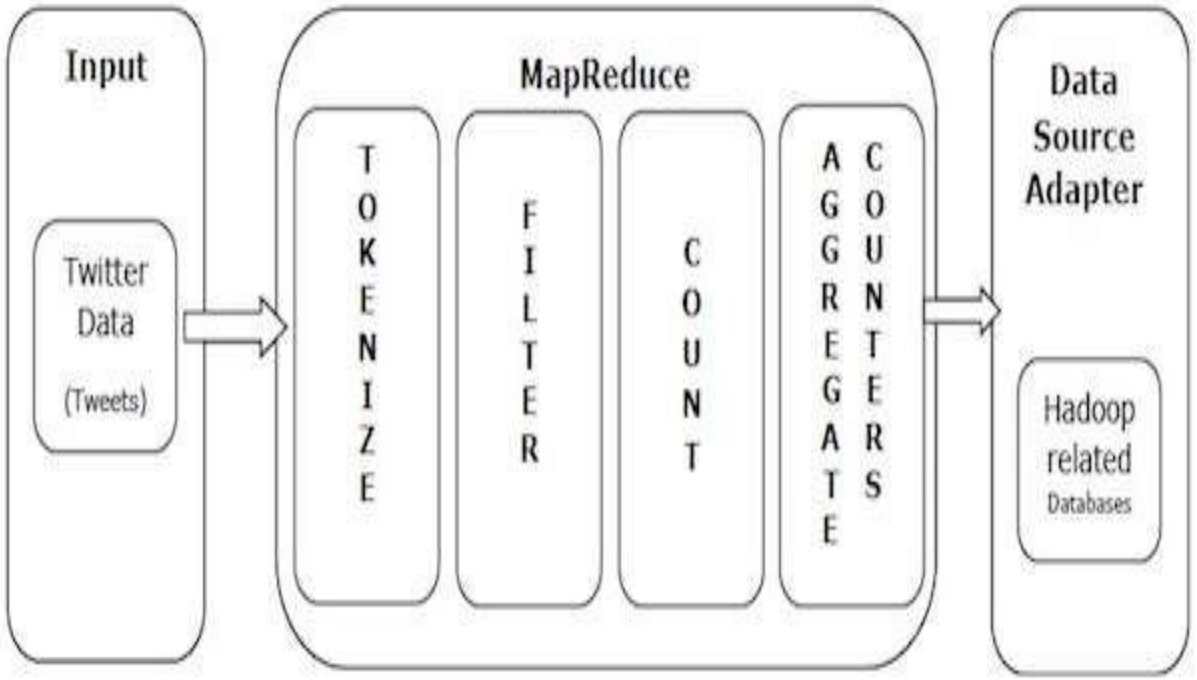
- How MapReduce Works?
- The MapReduce algorithm contains two important tasks, namely Map and Reduce.
- The Map task takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key-value pairs).
- The Reduce task takes the output from the Map as an input and combines those data tuples (key-value pairs) into a smaller set of tuples.
- The reduce task is always performed after the map job.


Input
I/p
Input
Input
Input
Input
Input
Input Phase
M
Map Phase
k1:v k1:v k2:v
k1:v
k1:v k1:v k2:v
k1:v k1:v k2:v
k4:v
k1:v k1:v k2:v
Intermediate Keys
Group by Key
Combiner (Optional)
k1:v,v,v,v
k2:v
k3:v,v
k4:v,v,v
k5:v
Shuffle & Sort
R
Reducer Phase
OUTPUT
Output Phase

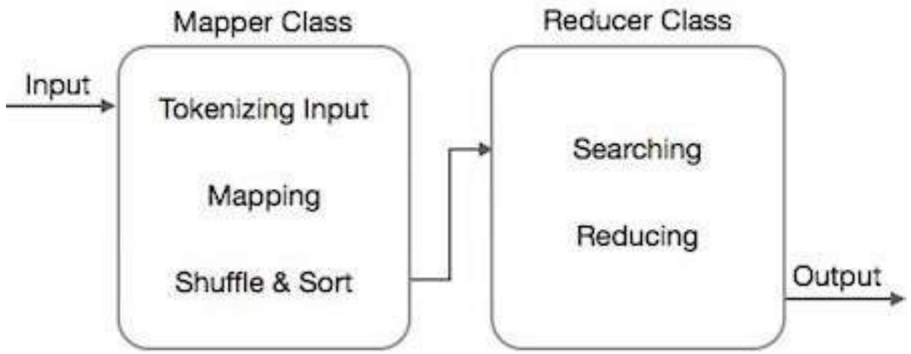
- **Input Phase** − Here we have a Record Reader that translates each record in an input file and sends the parsed data to the mapper in the form of key-value pairs.
- **Map** − Map is a user-defined function, which takes a series of key-value pairs and processes each one of them to generate zero or more key-value pairs.
- **Intermediate Keys** − They key-value pairs generated by the mapper are known as intermediate keys.
- **Combiner** − A combiner is a type of local Reducer that groups similar data from the map phase into identifiable sets. It takes the intermediate keys from the mapper as input and applies a user-defined code to aggregate the values in a small scope of one mapper. It is not a part of the main MapReduce algorithm; it is optional.
- **Shuffle and Sort** − The Reducer task starts with the Shuffle and Sort step. It downloads the grouped key-value pairs onto the local machine, where the Reducer is running. The individual key-value pairs are sorted by key into a larger data list. The data list groups the equivalent keys together so that their values can be iterated easily in the Reducer task.
- **Reducer** − The Reducer takes the grouped key-value paired data as input and runs a Reducer function on each one of them. Here, the data can be aggregated, filtered, and combined in a number of ways, and it requires a wide range of processing. Once the execution is over, it gives zero or more key-value pairs to the final step.
- **Output Phase** − In the output phase, we have an output formatter that translates the final key-value pairs from the Reducer function and writes them onto a file using a record writer.


Input
Split
Map Phase
Shuffle and Sort
Reduce Phase
A B R
C C R
A C B
A B R
C C R
A C B
A, 1
B, 1
R, 1
C, 1
C, 1
R, 1
A, 1
C, 1
B, 1
A, 1
A, 1
B, 1
B, 1
C, 1
C, 1
C, 1
R, 1
R, 1
A, 2
B, 2
C, 3
R, 2

- MapReduce-Example
- Let us take a real-world example to comprehend the power of MapReduce.
- Twitter receives around 500 million tweets per day, which is nearly 3000 tweets per second.
- The following illustration shows how Tweeter manages its tweets with the help of MapReduce.
-


Input
Twitter Data
(Tweets)
MapReduce
TOKENIZE
FILTER
COUNT
AGGREGATE
COUNTERS
Data Source Adapter
Hadoop related Databases

- As shown in the illustration, the MapReduce algorithm performs the following actions −
- **Tokenize** − Tokenizes the tweets into maps of tokens and writes them as key-value pairs.
- **Filter** − Filters unwanted words from the maps of tokens and writes the filtered maps as key-value pairs.
- **Count** − Generates a token counter per word.
- **Aggregate Counters** − Prepares an aggregate of similar counter values into small manageable units.
-


Mapper Class
Reducer Class
Input
Tokenizing Input
Mapping
Shuffle & Sort
Searching
Reducing
Output

- Mapper class takes the input, tokenizes it, maps and sorts it.
- The output of Mapper class is used as input by Reducer class, which in turn searches matching pairs and reduces them.
- MapReduce implements various mathematical algorithms to divide a task into small parts and assign them to multiple systems.
- In technical terms, MapReduce algorithm helps in sending the Map & Reduce tasks to appropriate servers in a cluster.

Techniques to optimize MapReduce jobs

- An analysis of the mapreduce program execution shows that it involves a series of steps in which every step has its own set of resource requirement
- We must avoid any bottenecks of resources in order to draw maximum benefit from the mapreduce resources
- These resources if utilized to the fullest can help to reduce the response time of mapreduce job
- In case of short jobs where user requirres quick response to his/her query the response time becomes more important.Hence we need the mapreduce job to be optimized
- We can organize mapreduce optimization techniques in the following categories
- Hardware or network topology
- Synchronization
- File system

Hardware or network topology

- The performance offered by the hardware systems that are located in the same rack where data is stored will be higher than the performance of hardware systems that are located in different rack than the one containing data
- The reason for low performance of hardware
- That is located away from the data is the requirement to move the data
- When the program is being executed we can configure the mapreduce engine to exploit the advantage of its closeness to the data.
- The best way to optimize the performance of mapreduce engine is to keep the application code and data together

Synchronization

File System

- Keep it warm
- The Bigger the Better
- The long View
- Right Degree of security

Exploring Map and Reduce Functions

Uses of MapReduce

- Web page Visits
- Web page visitor paths
- Word frequency
- Word count