MapReduce

What is MapReduce?

MapReduce is a programming model or pattern within the Hadoop framework that is used to access big data stored in the Hadoop File System (HDFS). It is a core component, integral to the functioning of the Hadoop framework.

The MapReduce component enhances the processing of massive data using dispersed and parallel algorithms in the Hadoop ecosystem. This programming model is applied in social platforms and e-commerce to analyze huge data collected from online users.

MapReduce facilitates concurrent processing by splitting petabytes of data into smaller chunks, and processing them in parallel on Hadoop commodity servers. In the end, it aggregates all the data from multiple servers to return a consolidated output back to the application.

For example, a Hadoop cluster with 20,000 inexpensive commodity servers and 256MB block of data in each, can process around 5TB of data at the same time. This reduces the processing time as compared to sequential processing of such a large data set.

With MapReduce, rather than sending data to where the application or logic resides, the logic is executed on the server where the data already resides, to expedite processing. Data access and storage is disk-based—the input is usually stored as files containing structured, semi-structured, or unstructured data, and the output is also stored in files.

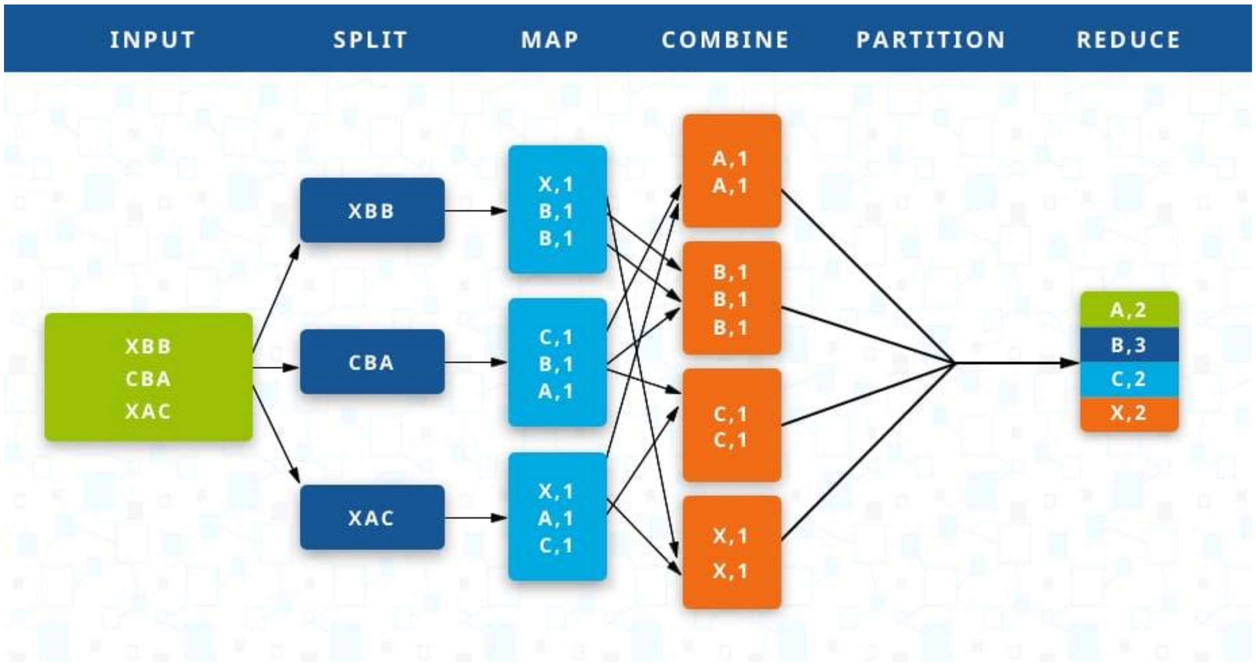
MapReduce was once the only method through which the data stored in the HDFS could be retrieved, but that is no longer the case. Today, there are other query-based systems such as Hive and Pig that are used to retrieve data from the HDFS using SQL-like statements.

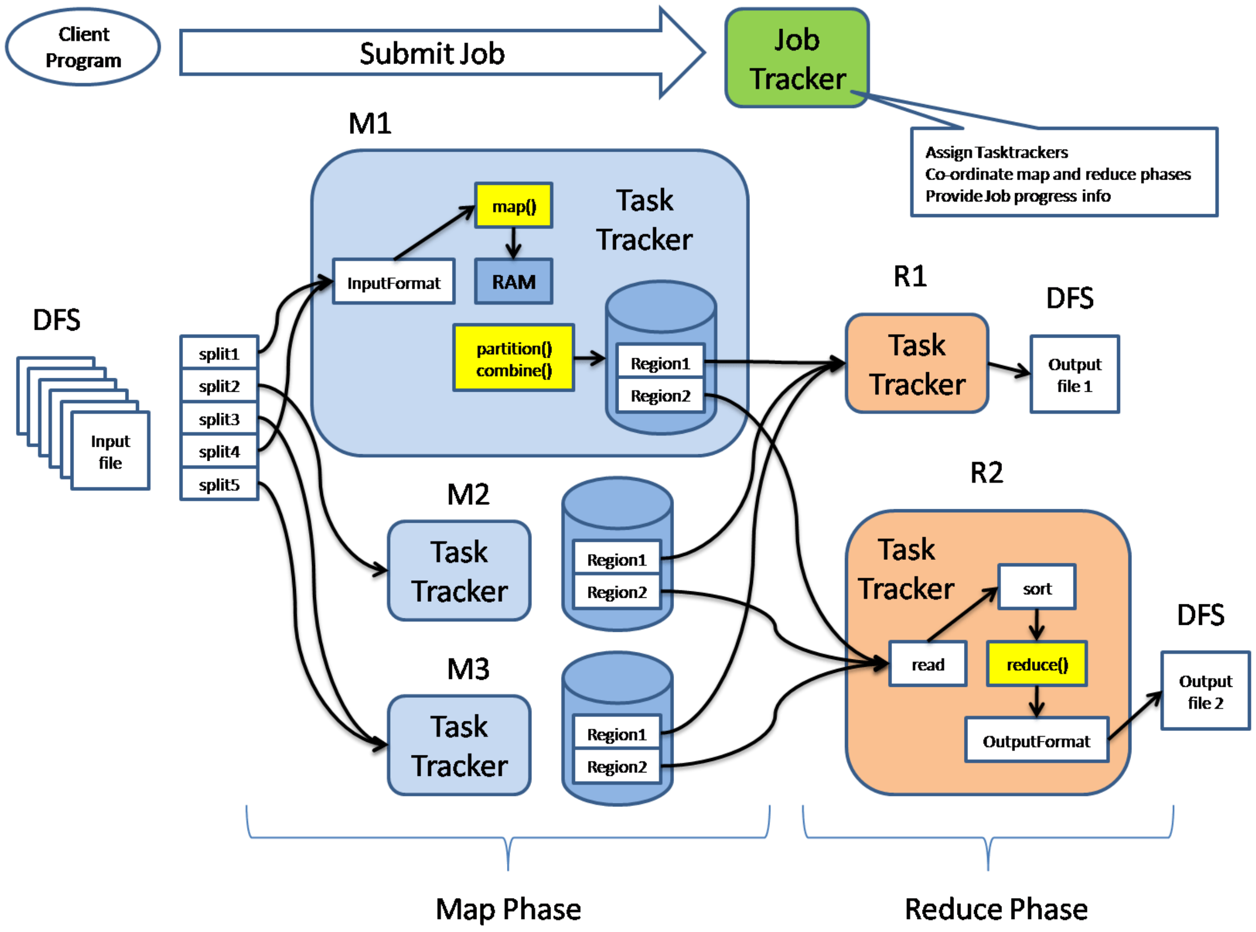
How MapReduce Works?

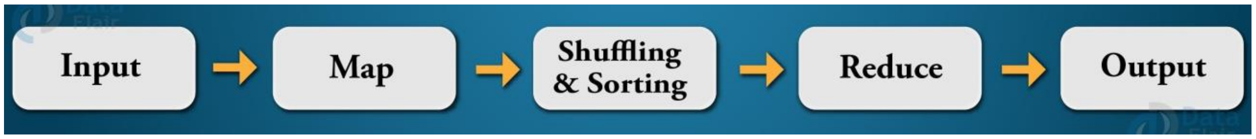
At the core of MapReduce are two functions: **Map** and **Reduce**. They are sequenced one after the other.

The **Map** function takes input from the disk as <key,value> pairs, processes them, and produces another set of intermediate <key,value> pairs as output.

The **Reduce** function also takes inputs as <key,value> pairs, and produces <key,value> pairs as output.


INPUT
SPLIT
MAP
COMBINE
PARTITION
REDUCE
XBB
CBA
XAC
XBB
CBA
XAC
X,1 B,1 B,1
C,1 B,1 A,1
X,1 A,1 C,1
A,1 A,1
B,1 B,1 B,1
C,1 C,1
X,1 X,1
A,2
B,3
C,2
X,2


Client Program
Submit Job
Job Tracker
Assign Tasktrackers
Co-ordinate map and reduce phases
Provide Job progress info
M1
Task Tracker
map()
InputFormat
RAM
partition() combine()
Region1
Region2
DFS
Input file
split1
split2
split3
split4
split5
M2
Task Tracker
Region1
Region2
M3
Task Tracker
Region1
Region2
R1
Task Tracker
DFS
Output file 1
R2
Task Tracker
sort
read
reduce()
OutputFormat
DFS
Output file 2
Map Phase
Reduce Phase



Maping Phase:

This is the first phase of the program. There are two steps in this phase: splitting and mapping. A dataset is split into equal units called chunks (input splits) in the splitting step. Hadoop consists of a RecordReader that uses TextInputFormat to transform input splits into key-value pairs.

The key-value pairs are then used as inputs in the mapping step. This is the only data format that a mapper can read or understand. The mapping step contains a coding logic that is applied to these data blocks. In this step, the mapper processes the key-value pairs and produces an output of the same form (key-value pairs).

Shuffling Phase:

This is the second phase that takes place after the completion of the Mapping phase. It consists of two main steps: sorting and merging. In the sorting step, the key-value pairs are sorted using the keys. Merging ensures that key-value pairs are combined.

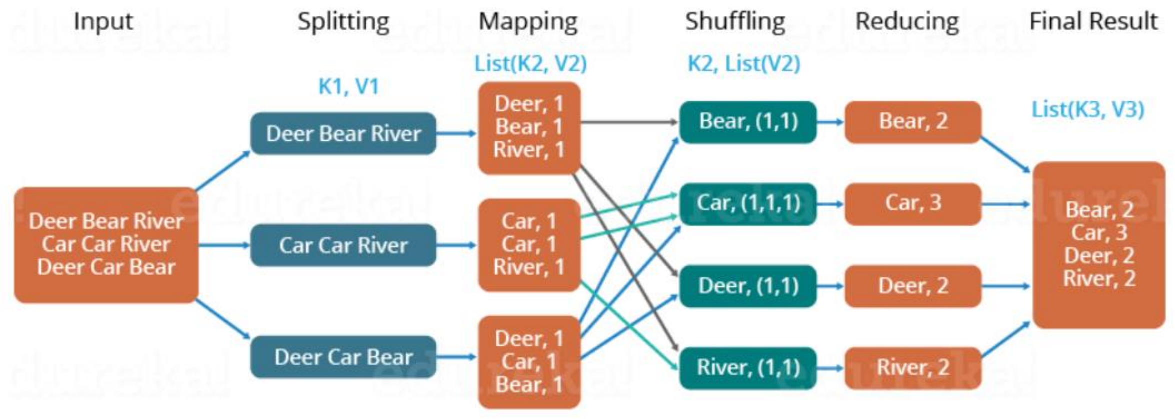
The shuffling phase facilitates the removal of duplicate values and the grouping of values. Different values with similar keys are grouped. The output of this phase will be keys and values, just like in the Mapping phase.

Reducer Phase:

In the reducer phase, the output of the shuffling phase is used as the input. The reducer processes this input further to reduce the intermediate values into smaller values. It provides a summary of the entire dataset. The output from this phase is stored in the HDFS.

The following diagram shows an example of a MapReduce with the three main phases. Splitting is often included in the mapping stage.

The overall MapReduce Word Could Process



Benefits of Hadoop MapReduce:

Speed: MapReduce can process huge unstructured data in a short time.

Fault-tolerance: The MapReduce framework can handle failures.

Cost-effective: Hadoop has a scale-out feature that enables users to process or store data in a cost-effective manner.

Scalability: Hadoop provides a highly scalable framework. MapReduce allows users to run applications from many nodes.

Data availability: Replicas of data are sent to various nodes within the network. This ensures copies of the data are available in the event of failure.

Parallel Processing: In MapReduce, multiple job-parts of the same dataset can be processed in a parallel manner. This reduces the time taken to complete a task.