MAPREDUCE

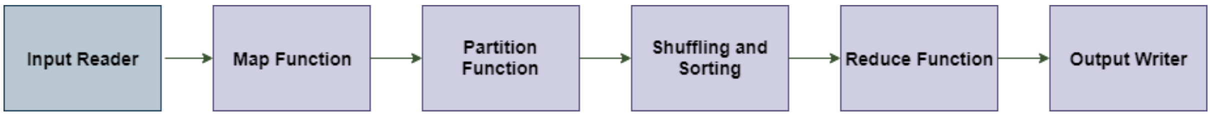
MAPREDUCE is a software framework and programming model used for processing huge amounts of data. **MapReduce** program work in two phases, namely, Map and Reduce. Map tasks deal with splitting and mapping of data while Reduce tasks shuffle and reduce the data.

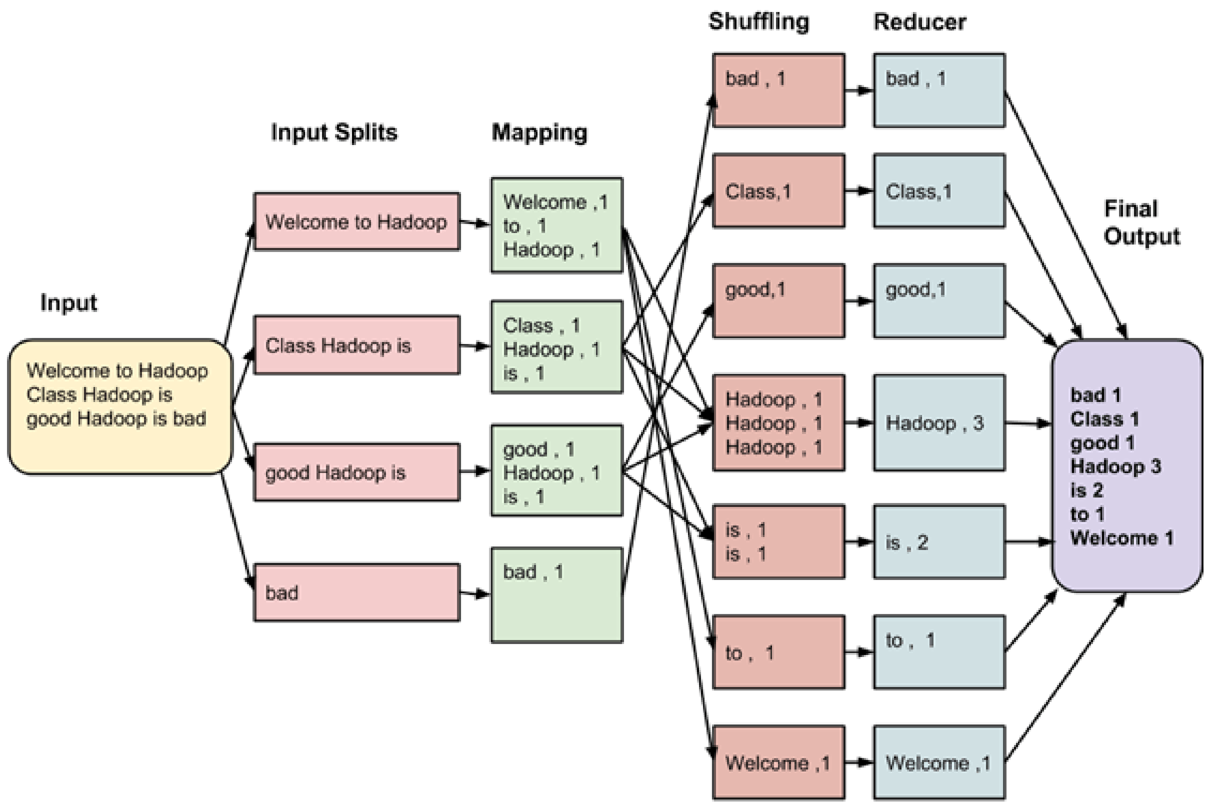
Hadoop is capable of running MapReduce programs written in various languages: Java, Ruby, Python, and C++. MapReduce programs are parallel in nature, thus are very useful for performing large-scale data analysis using multiple machines in the cluster.

The input to each phase is **key-value** pairs. In addition, every programmer needs to specify two functions: **map function** and **reduce function**.

Data Flow In MapReduce

MapReduce is used to compute the huge amount of data . To handle the upcoming data in a parallel and distributed form, the data has to flow from various phases.





Shuffling
Reducer
Input Splits
Mapping
Final Output
Input
Welcome to Hadoop Class Hadoop is good Hadoop is bad
Welcome to Hadoop
Class Hadoop is
good Hadoop is
bad
Welcome ,1 to , 1 Hadoop , 1
Class , 1 Hadoop , 1 is , 1
good , 1 Hadoop , 1 is , 1
bad , 1
bad , 1
Class,1
good,1
Hadoop , 1 Hadoop , 1 Hadoop , 1
is , 1 is , 1
to , 1
Welcome ,1
bad , 1
Class,1
good,1
Hadoop , 3
is , 2
to , 1
Welcome ,1
bad 1 Class 1 good 1 Hadoop 3 is 2 to 1 Welcome 1

Phases of MapReduce data flow

Input reader

The input reader reads the upcoming data and splits it into the data blocks of the appropriate size (64 MB to 128 MB). Each data block is associated with a Map function.

Once input reads the data, it generates the corresponding key-value pairs. The input files reside in HDFS.

Map function

The map function process the upcoming key-value pairs and generated the corresponding output key-value pairs. The map input and output type may be different from each other.

Partition function

The partition function assigns the output of each Map function to the appropriate reducer. The available key and value provide this function. It returns the index of reducers.

Shuffling and Sorting

The data are shuffled between/within nodes so that it moves out from the map and get ready to process for reduce function. Sometimes, the shuffling of data can take much computation time.

The sorting operation is performed on input data for Reduce function. Here, the data is compared using comparison function and arranged in a sorted form.

Reduce function

The Reduce function is assigned to each unique key. These keys are already arranged in sorted order. The values associated with the keys can iterate the Reduce and generates the corresponding output.

Output writer

Once the data flow from all the above phases, Output writer executes. The role of Output writer is to write the Reduce output to the stable storage.

MapReduce Mapper Class

In MapReduce, the role of the Mapper class is to map the input key-value pairs to a set of intermediate key-value pairs. It transforms the input records into intermediate records.

These intermediate records associated with a given output key and passed to Reducer for the final output.

Methods of Mapper Class

void cleanup(Context context)	This method called only once at the end of the task.
void map(KEYIN key, VALUEIN value, Context context)	This method can be called only once for each key-value in the input split.
void run(Context context)	This method can be override to control the execution of the Mapper.
void setup(Context context)	This method called only once at the beginning of the task.

MapReduce Reducer Class

In MapReduce, the role of the Reducer class is to reduce the set of intermediate values. Its implementations can access the Configuration for the job via the JobContext.getConfiguration() method.

Methods of Reducer Class

void cleanup(Context context)	This method called only once at the end of the task.
void map(KEYIN key, Iterable<VALUEIN> values, Context context)	This method called only once for each key.
void run(Context context)	This method can be used to control the tasks of the Reducer.
void setup(Context context)	This method called only once at the beginning of the task.

MapReduce Job Class

The Job class is used to configure the job and submits it. It also controls the execution and query the state. Once the job is submitted, the set method throws IllegalStateException.

Methods of Job Class

Methods	Description
Counters getCounters()	This method is used to get the counters for the job.
long getFinishTime()	This method is used to get the finish time for the job.
Job getInstance()	This method is used to generate a new Job without any cluster.
Job getInstance(Configuration conf)	This method is used to generate a new Job without any cluster and provided configuration.
Job getInstance(Configuration conf, String jobName)	This method is used to generate a new Job without any cluster and provided configuration and job name.
String getJobFile()	This method is used to get the path of the submitted job configuration.
String getJobName()	This method is used to get the user-specified job name.
JobPriority getPriority()	This method is used to get the scheduling function of the job.
void setJarByClass(Class<?> c)	This method is used to set the jar by providing the class name with .class extension.
void setJobName(String name)	This method is used to set the user-specified job name.

void setMapOutputKeyClass(Class<?> class)	This method is used to set the key class for the map output data.
void setMapOutputValueClass(Class<?> class)	This method is used to set the value class for the map output data.
void setMapperClass(Class<? extends Mapper> class)	This method is used to set the Mapper for the job.
void setNumReduceTasks(int tasks)	This method is used to set the number of reduce tasks for the job
void setReducerClass(Class<? extends Reducer> class)	This method is used to set the Reducer for the job.