Session: 9: Getting in touch with Map Reduce Framework

1. **Hadoop Map Reduce paradigm**

   A Map Reduce is a data processing tool which is used to process the data parallelly in a distributed form. It was developed in 2004.

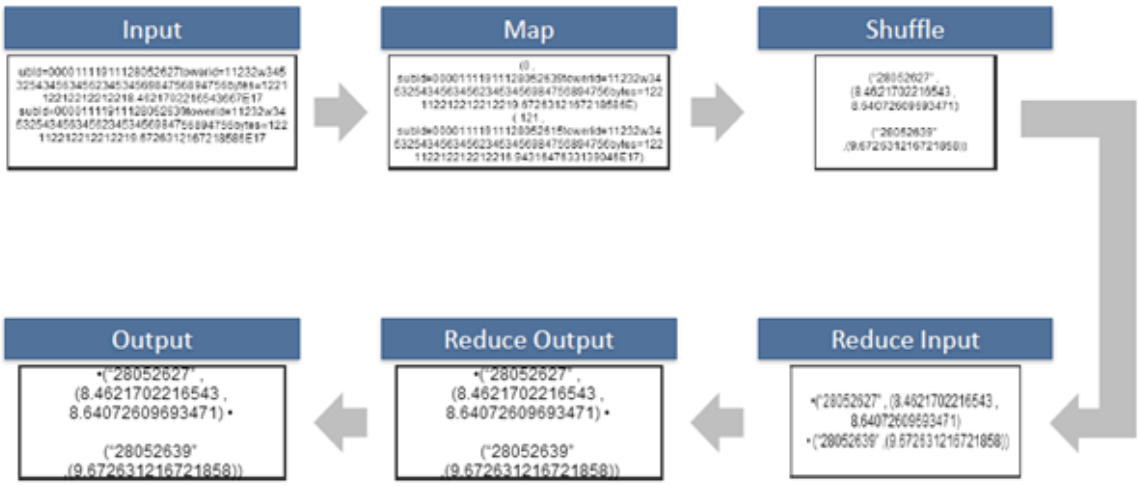
   The Map Reduce is a paradigm which has two phases:

   A. Mapper phase

   In the Mapper, the input is given in the form of a key-value pair.

   B. Reducer phase.

2. **Map and Reduce tasks**

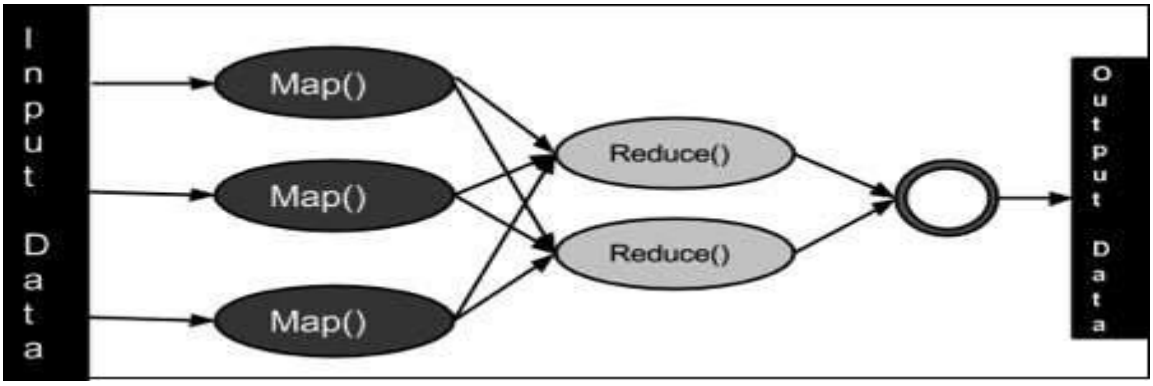


The main components of MapReduce?

- JobTracker and TaskTracker are the main components of the mapreduce.
- Job TrackerJob Tracker is a master which creates and runs the job. JobTracker that runs on name node, allocates the job to TaskTrackers.
- TaskTrackerTaskTracker is a slave and runs on data node.

3. **Map Reduce Execution Framework**

MapReduce is a processing technique and a program model for distributed computing based on java. The MapReduce algorithm contains two important tasks, namely Map and Reduce.



Different Phases of MapReduce: - Map Reduce model has three major phase.

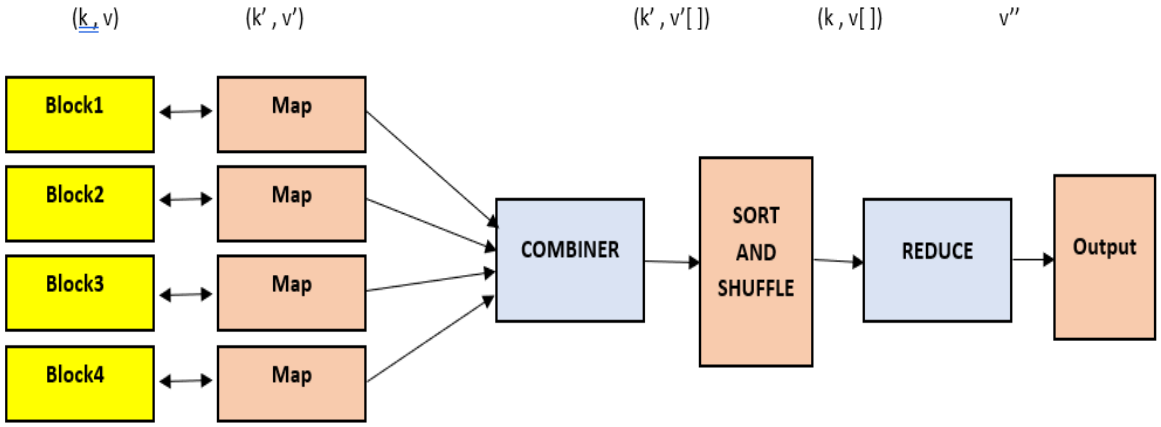
- Mapping
- Shuffling and Sorting
- Reducing
- Combining

Mapping: - It is the first phase of Map Reduce programming. Mapping Phase accepts key-value pairs as input as (k, v), where the key represents the Key address of each record and the value represents the entire record content.The output of the Mapping phase will also be in the key-value format (k', v').

Shuffling and Sorting: - The output of various mapping parts (k', v'), then goes into **Shuffling** and **Sorting phase**.

- All the same values are deleted, and different values are grouped together based on same keys.
- The output of the Shuffling and Sorting phase will be key-value pairs again as key and array of values (k, v[ ]).

Reducer: - The output of the Shuffling and Sorting phase (k, v[]) will be the input of the Reducer phase. In this phase reducer function's logic is executed and all the values are Collected against their corresponding keys.



Components of MapReduce Architecture:

A. **Client:** The MapReduce client is the one who brings the Job to the MapReduce for processing. There can be multiple clients available that continuously send jobs for processing to the Hadoop MapReduce Manager.

B. **Job:** The MapReduce Job is the actual work that the client wanted to do which is comprised of so many smaller tasks that the client wants to process or execute.
C. **Hadoop MapReduce Master:** It divides the particular job into subsequent job-parts.
D. **Job-Parts:** The task or sub-jobs that are obtained after dividing the main job. The result of all the job-parts combined to produce the final output.
E. **Input Data:** The data set that is fed to the MapReduce for processing.
F. **Output Data:** The final result is obtained after the processing.

4. Map Reduce Daemons

Map reduce is a program which is the processing layer in Hadoop framework. There are the key Map Reduce daemons:

A.JobTracker:

The JobTracker is the master daemon in a Hadoop MapReduce cluster. It manages job scheduling, resource allocation, and monitoring.

B.TaskTracker:

TaskTrackers are worker nodes in a Hadoop MapReduce cluster. They are responsible for executing Map and Reduce tasks.

TaskTrackers periodically send heartbeats and status updates to the JobTracker to report their availability and the tasks they are working on.

C.Map Task:

Map tasks are the units of work that process individual input splits. They apply a user-defined map function to the data and emit key-value pairs, which are then shuffled and sorted for subsequent reduce tasks.

D.Reduce Task:

Reduce tasks process the intermediate key-value pairs produced by the Map tasks. They apply a user-defined reduce function to group and aggregate the data, producing the final output.

E.TaskAttempt:

TaskAttempts represent individual attempts to execute a Map or Reduce task. If a task fails, it can be retried with a new TaskAttempt. TaskAttempts are tracked and managed by the TaskTracker.

F.Shuffle and Sort:

The Shuffle and Sort phase is an essential part of MapReduce processing. It involves the transfer of map output to the correct reduce task and sorting the data by keys.

G.Local Task Execution:

On a TaskTracker, map tasks can be executed locally if their input data is available on the same node. This minimizes data transfer over the network.

H. History Server:

The History Server (JobHistoryServer) stores job history information, including job configuration, status, and counters. It allows users to review the details of completed MapReduce jobs.

I.Fair Scheduler and Capacity Scheduler:

Hadoop provides advanced schedulers, such as the Fair Scheduler and Capacity Scheduler, which allow you to allocate cluster resources more efficiently based on job priorities and capacity limits.

5. Anatomy of a Map Reduce Job run

The execution of a MapReduce job in Hadoop involves several stages and processes, which can be broken down into the following key steps:

A.Client Submission:

The MapReduce job is submitted to the Hadoop cluster by the client application or user. This typically involves creating a job configuration that specifies input and output paths, input format, output format, Mapper and Reducer classes, and other job-specific settings.

B.Job Initialization:

The JobClient, running on the client machine, contacts the ResourceManager (or JobTracker in older Hadoop versions) to submit the job.

The ResourceManager (YARN) or JobTracker (in MapReduce 1) performs job initialization, which includes task assignment, resource allocation, and tracking the progress of the job.

C.Splitting and Mapping:

The input data is divided into splits, each of which corresponds to a data block in HDFS. These splits are processed by individual Mapper tasks.

Mapper tasks are responsible for applying the user-defined map function to the data within their input split and producing key-value pairs as output.

The intermediate key-value pairs generated by the Mappers are sorted and partitioned based on their keys.

D.Shuffling and Sorting:

The MapReduce framework groups and sorts the intermediate key-value pairs by key to prepare them for the Reducer tasks.

This shuffling and sorting phase ensures that data with the same key is grouped together and delivered to the same Reducer.

E.Reducing:

Reducer tasks receive the sorted key-value pairs, and they apply the user-defined reduce function to process and aggregate the data.

Each Reducer produces the final output, typically writing results to the Hadoop Distributed File System (HDFS) or another output location.

F.Job Completion and Cleanup:

- Once all Mapper and Reducer tasks have completed their work, the job is considered finished.
- Output data is stored in HDFS or another specified location.
- The **ResourceManager** or Job Tracker updates job status and cleans up any temporary resources.

G.Job History and Monitoring:

- Job history information, including job configuration, status, and counters, is stored by the **JobHistoryServe**r.
- Users and administrators can monitor the job's progress and review historical job details for analysis and debugging.

H.Output Retrieval:

Users can retrieve the final results from the output location specified during job configuration, typically by reading data from HDFS.

More Map Reduce Concepts

1. **Practitioners and Combiners**

Combiner: All the intermediate outputs are optimized by local aggregation before the shuffle/sort phase by the Combiner. The primary goal of Combiners is to save as much

bandwidth as possible by minimizing the number of key/value pairs that will be shuffled across the network and provided as input to the Reducer.

Partitioner : In Hadoop, partitioning of the keys of the intermediate map output is controlled by Partitioner. Hash function, is used to derive partition. On the basis of key-value pair each map output is partitioned. Record having same key value goes into the same partition (within each mapper), and then each partition is sent to a Reducer. Partition phase takes place in between mapper and reducer.

Default Partitioner (Hash Partitioner) computes a hash value for the key and assigns the partition based on this result

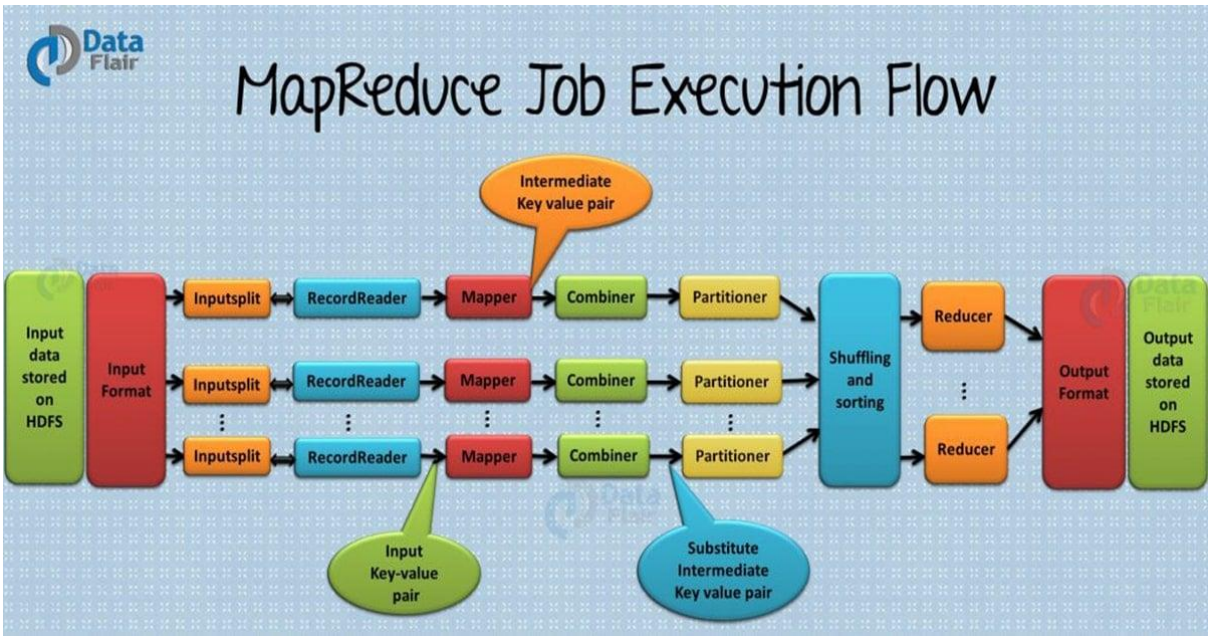
The Sequence of Execution goes as follows:

Mapper -> Combiner -> Partitioner

2. **Input Formats (Input Splits and Records, Text Input, Binary Input, Multiple Inputs)**

An Hadoop InputFormat is the first component in Map-Reduce, it is responsible for creating the input splits and dividing them into records.

Types of InputFormat in MapReduce



4.1. FileInputFormat in Hadoop

It is the base class for all file-based InputFormats. Hadoop FileInputFormat specifies input directory where data files are located. When we start a Hadoop job, FileInputFormat is provided with a path containing files to read. FileInputFormat will read all files and divides these files into one or more InputSplits.

4.2. TextInputFormat

It is the default InputFormat of MapReduce. TextInputFormat treats each line of each input file as a separate record and performs no parsing. This is useful for unformatted data or line-based records like log files.

Key – It is the byte offset of the beginning of the line within the file (not whole file just one split), so it will be unique if combined with the file name.

Value – It is the contents of the line, excluding line terminators.

4.3. KeyValueTextInputFormat

It is similar to TextInputFormat as it also treats each line of input as a separate record. While TextInputFormat treats entire line as the value, but the KeyValueTextInputFormat breaks the line itself into key and value by a tab character ('/t'). Here Key is everything up to the tab character while the value is the remaining part of the line after tab character.

4.4. SequenceFileInputFormat

Hadoop SequenceFileInputFormat is an InputFormat which reads sequence files. Sequence files are binary files that stores sequences of binary key-value pairs. Sequence files block-compress and provide direct serialization and deserialization of several arbitrary data types (not just text). Here Key & Value both are user-defined.

4.5. SequenceFileAsTextInputFormat

Hadoop SequenceFileAsTextInputFormat is another form of SequenceFileInputFormat which converts the sequence file key values to Text objects. By calling 'tostring()' conversion is performed on the keys and values. This InputFormat makes sequence files suitable input for streaming.

4.6. SequenceFileAsBinaryInputFormat

Hadoop SequenceFileAsBinaryInputFormat is a SequenceFileInputFormat using which we can extract the sequence file's keys and values as an opaque binary object.

4.7. NLineInputFormat

Hadoop NLineInputFormat is another form of TextInputFormat where the keys are byte offset of the line and values are contents of the line. Each mapper receives a variable number of lines of input with TextInputFormat and KeyValueTextInputFormat and the number depends on the size of the split and the length of the lines. And if we want our mapper to receive a fixed number of lines of input, then we use NLineInputFormat.

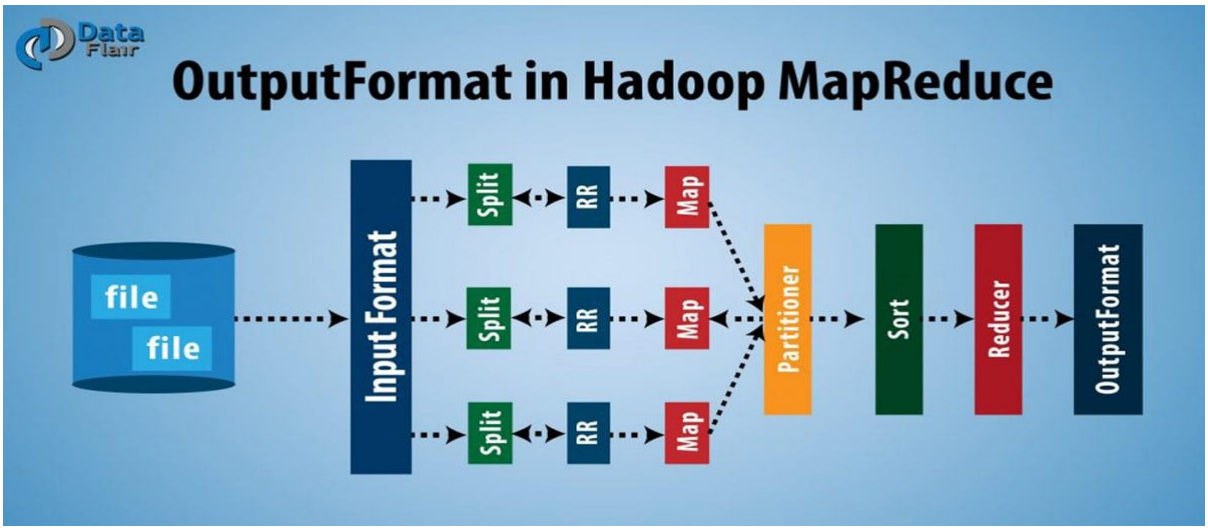
N is the number of lines of input that each mapper receives. By default (N=1), each mapper receives exactly one line of input. If N=2, then each split contains two lines. One mapper will receive the first two Key-Value pairs and another mapper will receive the second two key-value pairs.

4.8. DBInputFormat

Hadoop DBInputFormat is an InputFormat that reads data from a relational database, using JDBC. As it doesn't have portioning capabilities, so we need to careful not to swamp the database from which we are reading too many mappers. So it is best for loading relatively small datasets, perhaps for joining with large datasets from HDFS using MultipleInputs. Here Key is LongWritables while Value is DBWritables.

3. **Output Formats (Text Output, Binary Output, Multiple Output)**

The Hadoop Output Format checks the Output-Specification of the job. It determines how RecordWriter implementation is used to write output to output files.



Types of Hadoop Output Formats

i. TextOutputFormat

MapReduce default Hadoop reducer Output Format is TextOutputFormat, which writes (key, value) pairs on individual lines of text files and its keys and values can be of any type since TextOutputFormat turns them to string by calling toString() on them. Each key-value pair is separated by a tab character, which can be changed using MapReduce.output.textoutputformat.separator property. KeyValueTextOutputFormat is used for reading these output text files since it breaks lines into key-value pairs based on a configurable separator.

ii. SequenceFileOutputFormat

It is an Output Format which writes sequences files for its output and it is intermediate format use between MapReduce jobs, which rapidly serialize arbitrary data types to the file; and the corresponding SequenceFileInputFormat will deserialize the file into the same types and presents the data to the next mapper in the same manner as it was emitted by the previous reducer, since these are compact and readily compressible. Compression is controlled by the static methods on SequenceFileOutputFormat.

iii. MultipleOutputs

It allows writing data to files whose names are derived from the output keys and values, or in fact from an arbitrary string.

vii. DBOutputFormat

DBOutputFormat in Hadoop is an Output Format for writing to relational databases and HBase. It sends the reduce output to a SQL table. It accepts key-value pairs, where the key has a type extending DBwritable. Returned RecordWriter writes only the key to the database with a batch SQL query.

4. **Distributed Cache**

Distributed cache is a mechanism that stores frequently accessed data in memory to reduce the load on the database and improve the speed of data access. Distributed cache is an essential component of modern web applications, where multiple users often access data across different regions.

Distributed caches are especially useful in environments with high data volume and load. The distributed architecture allows incremental expansion/scaling by adding more computers to the cluster, allowing the cache to grow in step with the data growth.