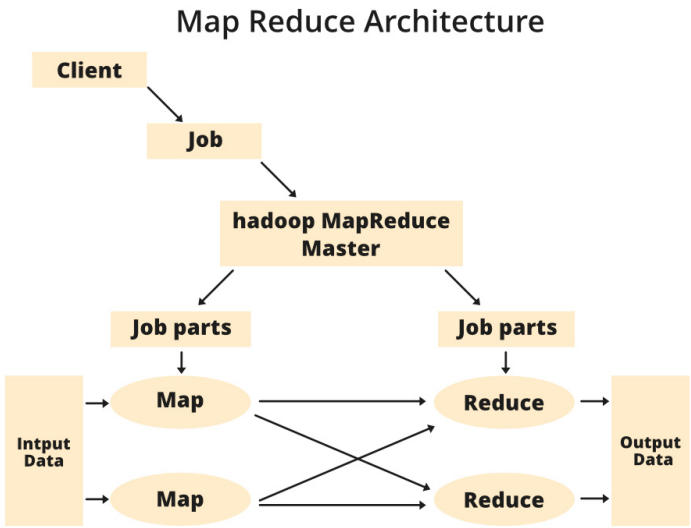
MapReduce Architecture

Difficulty Level : Easy ● Last Updated : 10 Sep, 2020

Read Discuss Courses Practice Video

MapReduce and HDFS are the two major components of Hadoop which makes it so powerful and efficient to use. MapReduce is a programming model used for efficient processing in parallel over large data-sets in a distributed manner. The data is first split and then combined to produce the final result. The libraries for MapReduce is written in so many programming languages with various different-different optimizations. The purpose of MapReduce in Hadoop is to Map each of the jobs and then it will reduce it to equivalent tasks for providing less overhead over the cluster network and to reduce the processing power. The MapReduce task is mainly divided into two phases Map Phase and Reduce Phase.

MapReduce Architecture:



Components of MapReduce Architecture:

1. **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.

2. **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.
3. **Hadoop MapReduce Master:** It divides the particular job into subsequent job-parts.
4. **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.
5. **Input Data:** The data set that is fed to the MapReduce for processing.
6. **Output Data:** The final result is obtained after the processing.

In **MapReduce**, we have a client. The client will submit the job of a particular size to the Hadoop MapReduce Master. Now, the MapReduce master will divide this job into further equivalent job-parts. These job-parts are then made available for the Map and Reduce Task. This Map and Reduce task will contain the program as per the requirement of the use-case that the particular company is solving. The developer writes their logic to fulfill the requirement that the industry requires. The input data which we are using is then fed to the Map Task and the Map will generate intermediate key-value pair as its output. The output of Map i.e. these key-value pairs are then fed to the Reducer and the final output is stored on the HDFS. There can be n number of Map and Reduce tasks made available for processing the data as per the requirement. The algorithm for Map and Reduce is made with a very optimized way such that the time complexity or space complexity is minimum.

Let's discuss the MapReduce phases to get a better understanding of its architecture:

The MapReduce task is mainly divided into **2 phases** i.e. Map phase and Reduce phase.

1. **Map:** As the name suggests its main use is to map the input data in key-value pairs. The input to the map may be a key-value pair where the key can be the id of some kind of address and value is the actual value that it keeps. The *Map()* function will be executed in its memory repository on each of these input key-value pairs and generates the intermediate key-value pair which works as input for the Reducer or *Reduce()* function.

2. **Reduce:** The intermediate key-value pairs that work as input for Reducer are shuffled and sort and send to the *Reduce()* function. Reducer aggregate or group the data

based on its key-value pair as per the reducer algorithm written by the developer.

How Job tracker and the task tracker deal with MapReduce:

across the cluster and also to schedule each map on the Task Tracker running on the same data node since there can be hundreds of data nodes available in the cluster.

2. **Task Tracker:** The Task Tracker can be considered as the actual slaves that are working on the instruction given by the Job Tracker. This Task Tracker is deployed on each of the nodes available in the cluster that executes the Map and Reduce task as instructed by Job Tracker.

There is also one important component of MapReduce Architecture known as **Job History Server**. The Job History Server is a daemon process that saves and stores historical information about the task or application, like the logs which are generated during or after the job execution are stored on Job History Server.

26

Related Articles

1. How to find top-N records using MapReduce
2. MapReduce - Combiners
3. Sum of even and odd numbers in MapReduce using Cloudera Distribution Hadoop(CDH)
4. How to Execute WordCount Program in MapReduce using Cloudera Distribution Hadoop(CDH)
5. Distributed Cache in Hadoop MapReduce
6. How MapReduce handles data query ?
7. MapReduce Job Execution
8. Job Initialisation in MapReduce
9. How Job runs on MapReduce
10. How MapReduce completes a task?