BIG DATA PROCESSING WITH HADOOP-MAPREDUCE IN CLOUD SYSTEMS

BIG DATA:

Big data is a collection of data sets so large and complex which is also exceeds the processing capacity of conventional database systems.

HADOOP:

Hadoop is a batch processing system for a cluster of nodes that provides the underpinnings of most Big Data analytic activities because it bundle two sets of functionality most needed to deal with large unstructured datasets namely, Distributed file system and MapReduce processing. Some of the current sub projects of Hadoop are Core, MapReduce, HDFS, Pig, Hive, Zookeeper, Chukka etc.

HADOOP ARCHITECTURE:

Hadoop is a Map/Reduce framework that works on HDFS or on HBase. The main idea is to decompose a job into several and identical tasks that can be executed closer to the data (on the DataNode). In addition, each task is parallelized: the Map phase. Then all these intermediate results are merged into one result: the Reduce phase. In Hadoop, The Job Tracker (a java process) is responsible for monitoring the job, managing the Map/Reduce phase, managing the retries in case of errors. The Task Trackers (Java process) are running on the different Data Nodes. Each Task Tracker executes the tasks of the job on the locally stored data.

APPLICATIONS OF HADOOP:

- Building search index at Google, Amazon, Yahoo
- Analyzing user logs, data warehousing and analytics
- Used for large scale machine learning and data mining applications
- Legacy data processing where it requires massive computational

HDFS:

An HDFS cluster has two types of node operating in a master-worker pattern: a NameNode (the master) and a number of DataNodes (workers). The namenode manages the filesystem namespace. It maintains the filesystem tree and the metadata for all the files and directories in the tree. The namenode also knows the datanodes on which all the blocks for a given file are located. Datanodes are the workhorses of the filesystem

MAP REDUCE:

MapReduce is a data processing or parallel programming model introduced by Google. In this model, a user specifies the computation by two functions, Map and Reduce. In the mapping phase, MapReduce takes the input data and feeds each data element to the mapper. In the reducing phase, the reducer processes all the outputs from the mapper and arrives at a final result. In simple terms, the mapper is meant to filter and transform the input into something that the reducer can aggregate over. Below are the failures and tolerance of Map Reduce.

Fault tolerance- MapReduce is designed to be fault tolerant because failures are common phenomena in large scale distributed computing and it includes worker failure and master failure.

Worker failure-The master pings every mapper and reducer periodically. If no response is received for a certain amount of time, the machine is marked as failed. The ongoing task and any tasks completed by this mapper will be re-assigned to another mapper and executed from the very beginning. Completed reduce tasks do not need to be re-executed because their output is stored in the global file system

Master failure- Since the master is a single machine, the probability of master failure is very small. MapReduce will re-start the entire job if the master fails. There are currently three popular implementations of the MapReduce programming model namely Google MapReduce, Apache Hadoop, Stanford Phoenix

Execution Process:

- MapReduce will first divide the data into N partitions with size varies from 16MB to 64MB
- Then it will start many programs on a cluster of different machines. One of program is the master program; the others are workers, which can execute their work assigned by master. Master can distribute a map task or a reduce task to an idle worker.
- If a worker is assigned a Map task, it will parse the input data partition and output the key/value pairs, then pass the pair to a user defined Map function. The map function will buffer the temporary key/value pairs in memory. The pairs will periodically be written to local disk and partitioned into P pieces. After that, the local machine will inform the master of the location of these pairs.
- If a worker is assigned a Reduce task and is informed about the location of these pairs, the Reducer will read the entire buffer by using remote procedure calls. After that, it will sort the temporary data based on the key.
- Then, the reducer will deal with all of the records. For each key and according set of values, the reducer passes key/value pairs to a user defined Reduce function. The output is the final output of this partition.
- After all of the mappers and reducers have finished their work, the master will return the result to users' programs. The output is stored in F individual files.