Hadoop 2x

Hadoop consists of MapReduce, the Hadoop distributed file system (HDFS) and a number of related projects such as Apache Hive, HBase and Zookeeper. **MapReduce and Hadoop distributed file system (HDFS) are the main component of Hadoop.**

Hadoop cluster has 3 components:

* 1. Client
  2. Master
  3. Slave

Masters:

The Masters consists of 3 components NameNode, Secondary Node name and JobTracker.  
  
   
  
**NameNode:**  
NameNode does NOT store the files but only the file's metadata.

NameNode oversees the health of DataNode and coordinates access to the data stored in DataNode.   
Name node keeps track of all the file system related information such as to

* + Which section of file is saved in which part of the cluster
  + Last access time for the files
  + User permissions like which user have access to the file

**JobTracker:**  
JobTracker coordinates the parallel processing of data using MapReduce.   
  
  
**Secondary Name Node:**  
The job of Secondary Node is to contact NameNode in a periodic manner after certain time interval(by default 1 hour).   
NameNode which keeps all filesystem metadata in RAM has no capability to process that metadata on to disk. So if NameNode crashes, you lose everything in RAM itself and you don't have any backup of filesystem. It is contacts NameNode in an hour and pulls copy of metadata information out of NameNode. It shuffle and merge this information into clean file folder and sent to back again to NameNode, while keeping a copy for itself. Hence Secondary Node is not the backup rather it does job of housekeeping.   
In case of NameNode failure, saved metadata can rebuild it easily. 

Slaves:

Slave nodes are the majority of machines in Hadoop Cluster and are responsible to

* + Store the data
  + Process the computation

   
  
Each slave runs both a DataNode and Task Tracker daemon which communicates to their masters. The Task Tracker daemon is a slave to the JobTracker and the DataNode daemon a slave to the NameNode 

Hadoop- Typical Workflow in HDFS:

**Client** machine does this step and loads the Sample.txt into cluster. It breaks the sample.txt into smaller chunks which are known as **"Blocks"** in Hadoop context. Client put these blocks on different machines (data nodes) throughout the cluster. 

Now **NameNode** comes into picture. The NameNode used its Rack Awareness intelligence to decide on which DataNode to provide. For each of the data block (in this case Block-A, Block-B and Block-C), Client contacts NameNode and in response NameNode sends an ordered list of 3 DataNodes.   
  
For example in response to Block-A request, Node Name may send DataNode-2, DataNode-3 and DataNode-4.   
  
   
  
Similarly for Block-B DataNodes list DataNode-1, DataNode-3, DataNode-4 and for Block C data node list DataNode-1, DataNode-2, DataNode-3. Hence

* + - * Block A gets stored in DataNode-2, DataNode-3, DataNode-4
      * Block B gets stored in DataNode-1, DataNode-3, DataNode-4
      * Block C gets stored in DataNode-1, DataNode-2, DataNode-3

Every block is replicated to more than 1 data nodes to ensure the data recovery on the time of machine failures. That's why NameNode send 3 DataNodes list for each individual block

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. Map takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs). Secondly, reduce task, which takes the output from a map as an input and combines those data tuples into a smaller set of tuples. As the sequence of the name MapReduce implies, the reduce task is always performed after the map job.

The major advantage of MapReduce is that it is easy to scale data processing over multiple computing nodes. Under the MapReduce model, the data processing primitives are called mappers and reducers. Decomposing a data processing application into mappers and reducers is sometimes nontrivial. But, once we write an application in the MapReduce form, scaling the application to run over hundreds, thousands, or even tens of thousands of machines in a cluster is merely a configuration change. This simple scalability is what has attracted many programmers to use the MapReduce model.

The Algorithm

* Generally MapReduce paradigm is based on sending the computer to where the data resides!
* MapReduce program executes in three stages, namely map stage, shuffle stage, and reduce stage.
  + **Map stage** : The map or mapper’s job is to process the input data. Generally the input data is in the form of file or directory and is stored in the Hadoop file system (HDFS). The input file is passed to the mapper function line by line. The mapper processes the data and creates several small chunks of data.
  + **Reduce stage** : This stage is the combination of the **Shuffle**stage and the **Reduce** stage. The Reducer’s job is to process the data that comes from the mapper. After processing, it produces a new set of output, which will be stored in the HDFS.
* During a MapReduce job, Hadoop sends the Map and Reduce tasks to the appropriate servers in the cluster.
* The framework manages all the details of data-passing such as issuing tasks, verifying task completion, and copying data around the cluster between the nodes.
* Most of the computing takes place on nodes with data on local disks that reduces the network traffic.
* After completion of the given tasks, the cluster collects and reduces the data to form an appropriate result, and sends it back to the Hadoop server.