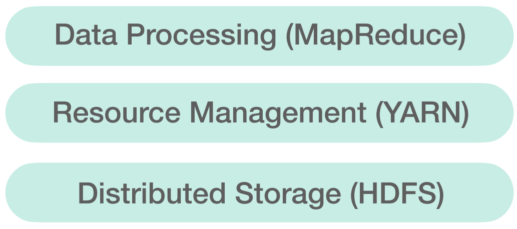
Hadoop : How does MapReduce work?

- Arushi

(To be continued)

The nature and sheer volume of data today - both structured and unstructured - demands complementary reliable, inexpensive storage capabilities: Apache Hadoop is a leading contender. Hadoop provides us with multiple servers / clusters of computers called nodes that are inter-networked to allow scalable storage and parallel computation on big data sets. It supports parallel computing, error-handling, logging, fault-tolerance, monitoring and reporting.

Hadoop has 3 main layers:


Data Processing (MapReduce)
Resource Management (YARN)
Distributed Storage (HDFS)

MapReduce

MapReduce is a programming model, or a way of structuring an application so that storage and computation can happen on multiple machines. It forces you to break what you're trying to do into three stages: Map, Shuffle, Reduce.
Map phase: It is the process of splitting input among various machines. No communication between different machines is allowed during this phase. The output from the Mapper is intermediate output, and is not stored in the HDFS.
Shuffle phase: Sort keys so that each reducer gets the same keys (since the same keys might be present across multiple intermediate output files from the Mapping phase).
Reducer: Takes results from Shuffle phase and combines together to get one result.

To put it simply, take this example: say you were a Historian, and you had to translate an entire Vedic Sanskrit text from 1000 scrolls of manuscript, while also grouping related scrolls and identifying and setting aside unrelated scrolls. If 500 or 1000 people worked together, you would be able to finish the work much faster. This is what parallel computing means. This is akin to the Mapping phase of MapReduce, and produces the intermediate output of a number of translated texts. After all the scrolls are either set aside or translated, the related data must be identified and grouped together from the intermediate output - this is analogous to the Shuffle phase. Next, the data must be put together, like in the Reduce phase where data from the Shuffle phase is aggregated to produce the final result.

However, we must consider that it wouldn't be optimal to have, say, a million people working on the translations, since each person would only translate a thousandth of a scroll (assuming the work is split equally). It would be far too difficult put the results together, and to identify which Historian is working on which type of scroll - managing so many people's results would be a hassle and may waste more time and resources than if you had only used 100 translators. Similarly, the default (and recommended) size of a HDFS data block is 128 MB, because if the size is smaller, a larger number of blocks would be required to store data; consequently, a larger number of map tasks would be required, and a larger amount of metadata would be generated.

HDFS - Hadoop Distributed File System

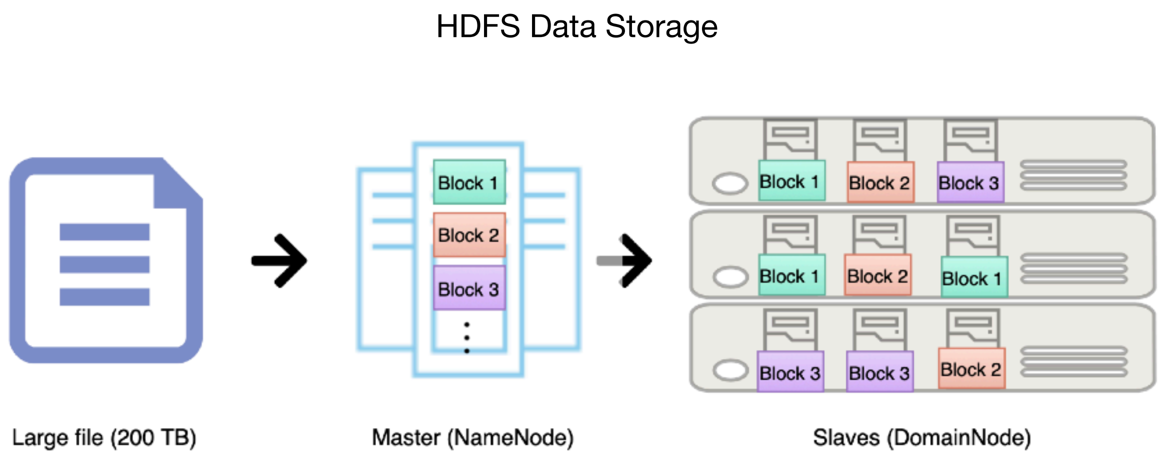
How is data stored?

On HDFS, data is divided into chunks and spread across clusters of commodity hardware. These clusters, for the most part, consist of two types of nodes: Data Nodes and Name Nodes. The data to be stored and processed enters the HDFS through input files of various formats. These input files are split into smaller blocks, which are stored on DataNodes.

What are Master nodes (NameNodes) and Slave nodes (DataNodes)?

Data Nodes are nodes which store the actual data, while a Name Node is a master node which manages the metadata. The NameNode has a book-keeping and a supervisory role; it stores the directory tree of all files, file-splits and their locations in a cluster, and is also in charge of generating reports to discern cluster-health, and for collecting reports about blocks from Data Nodes and keeping a tab on the replication factors of all files: if a block goes missing, then commands are issued to replicate the missing block of data.

How are blocks replicated?



Block replication is important so that there are multiple back-ups for blocks to ensure fault-tolerance. Files have a replication factor, which indicates the number of block-replicas stored in the file-system. To store a file, the Client application interacts with the NameNode first, in order to get the locations where the file blocks can be stored, and then can interact directly with the Domain Nodes. The Domain Nodes can interact with one another during replication of blocks. Huge HDFS instances run on clusters of computers spread across multiple racks. Communication between nodes on two different racks is through a switch, and the network bandwidth is less.

Suppose the block replication factor is 3, one block will be stored on the local rack, and the other 2 replicas will be stored on a remote (but preferably the same) rack to reduce the aggregate network bandwidth required, since risk of rack failure is significantly less than that of node failure.

YARN - Yet Another Resource Negotiator

Yarn works as the Operating system on Clusters, and does resource management and job scheduling. It keeps track of NodeManagers and available resources, allocates these resources to appropriate tasks, and monitors application masters. Yarn uses two daemons: ResourceManager (on master node) and NodeManager (on slave machines).
The Resource Manager manages resources across a Cluster. The Node Manager oversees the Containers (package of resources like CPU, RAM, HDD and Network) running on Clusters. The Application Manager negotiates with the Resource Manager for resources and runs tasks (map/reduce) on Clusters.