Basics of Hadoop

**What is Hadoop?**

* Hadoop (the proper name is ApacheHadoop) is an open-source framework that was created to make it easier to work with big data.
* It provides a method to access data that is distributed among multiple clustered computers.
* Hadoop is a framework for working with big data. It is part of the big data ecosystem, which consists of much more than Hadoop itself.
* Hadoop is a distributed framework that makes it easier to process large data sets that reside in clusters of computers.
* Hadoop is made up of four core modules that are supported by a large ecosystem of supporting technologies and products.
* They are:
  + **Hadoop Distributed File System** – Provides access to application data. Hadoop can also work with other file systems, including FTP, Amazon S3 and Windows Azure Storage Blobs (WASB), among others.
  + **Hadoop YARN** – Provides the framework to schedule jobs and manage resources across the cluster that holds the data
  + **Hadoop MapReduce** – A YARN-based parallel processing system for large data sets.
  + **Hadoop Common** – A set of utilities that supports the three other core modules

**Advantages of Hadoop**

* **Fast:** In HDFS the data distributed over the cluster and are mapped which helps in faster retrieval. Even the tools to process the data are often on the same servers, thus reducing the processing time. It is able to process terabytes of data in minutes and petabytes in hours.
* **Scalable:** Hadoop cluster can be extended by just adding nodes in the cluster.
* **Cost Effective:** Hadoop is open source and uses commodity hardware to store data so it really cost effective as compared to traditional relational database management system.
* **Resilient to failure:** HDFS has the property with which it can replicate data over the network, so if one node is down or some other network failure happens, then Hadoop takes the other copy of data and use it. Normally, data are replicated thrice but the replication factor is configurable.

**HDFS:**

* In HDFS data is distributed over several machines and replicated to ensure their durability to failure and high availability to parallel application.
* It is cost effective as it uses commodity hardware.
* It involves the concept of blocks, data nodes and node name.
* Where to use HDFS
* **Very Large Files:** Files should be of hundreds of megabytes, gigabytes or more.
* **Streaming Data Access:**The time to read whole data set is more important than latency in reading the first. HDFS is built on write-once and read-many-times pattern.
* **Commodity Hardware:** It works on low cost hardware.
* Where not to use HDFS
* **Low Latency data access:** Applications that require very less time to access the first data should not use HDFS as it is giving importance to whole data rather than time to fetch the first record.
* **Lots of Small Files:** The name node contains the metadata of files in memory and if the files are small in size it takes a lot of memory for name node's memory which is not feasible.
* **Multiple Writes:** It should not be used when we have to write multiple times.
* **HDFS Concepts:**
* **Blocks:**
  + A Block is the minimum amount of data that it can read or write.
  + HDFS blocks are 128 MB by default and this is configurable.
  + Files in HDFS are broken into block-sized chunks, which are stored as independent units.
  + Unlike a file system, if the file is in HDFS is smaller than block size, then it does not occupy full block size,
  + E.g., 5 MB of file stored in HDFS of block size 128 MB takes 5MB of space only. The HDFS block size is large just to minimize the cost of seek.
* **Name Node:**
  + HDFS works in master-worker pattern where the name node acts as master.
  + Name Node is controller and manager of HDFS as it knows the status and the metadata of all the files in HDFS.
  + The metadata information being file permission, names and location of each block.
  + The metadata are small, so it is stored in the memory of name node, allowing faster access to data.
  + Moreover the HDFS cluster is accessed by multiple clients concurrently, so all this information is handled by a single machine. The file system operations like opening, closing, renaming etc. are executed by it.
* **Data Node:**
  + They store and retrieve blocks when they are told to; by client or name node.
  + They report back to name node periodically, with list of blocks that they are storing.
  + The data node being a commodity hardware also does the work of block creation, deletion and replication as stated by the name node.

**Hadoop YARN:**

* **Y**et **A**nother **R**esource **M**anager takes programming to the next level beyond Java, and makes it interactive to let another application HBase, Spark etc. to work on it.
* **Components Of YARN:**
* **Client:** For submitting MapReduce jobs.
* **Resource Manager:** To manage the use of resources across the cluster
* **Node Manager:** For launching and monitoring the computer containers on machines in the cluster.
* **MapReduce Application Master:** Checks tasks running the MapReduce job. The application master and the MapReduce tasks run in containers that are scheduled by the resource manager, and managed by the node managers.
* **Benefits of YARN:**
* **Scalability:** Map Reduce 1 hits a scalability bottleneck at 4000 nodes and 40000 task, but Yarn is designed for 10,000 nodes and 1 lakh tasks.
* **Utilization:** Node Manager manages a pool of resources, rather than a fixed number of the designated slots thus increasing the utilization.
* **Multitenancy:** Different version of MapReduce can run on YARN, which makes the process of upgrading MapReduce more manageable.

**Hadoop MapReduce:**

* To take the advantage of parallel processing of Hadoop, the query must be in MapReduce form.
* The MapReduce is a paradigm which has two phases, the **Mapper** phase and the **Reducer** phase.
* In the Mapper the input is given in the form of key value pair.
* The output of the mapper is fed to the reducer as input.
* The reducer runs only after the mapper is over.
* The reducer too takes input in key value format and the output of reducer is final output.