International Journal of Enhanced Research in Management & Computer Applications
ISSN: 2319-7471, Vol. 6 Issue 9, September-2017, Impact Factor: 3.578

Hadoop Version 1.0

Gaurav Malik

ABSTRACT

In this Paper I have explained Hadoop Version1.0 and afterwards I am throwing light on the benefits of Hadoop And then I explained Hadoop read operation and various daemons.

Keywords: Hadoop , Hdfs.

INTRODUCTION

What is Hadoop

●A free, Java-based programming framework that supports the processing of large data sets in a distributed computing environment
●Based on Google File System (GFS)

There are 2 parts of hadoop

1. **HDFS**(Hadoop distributed File system):- It is a storage part
2. **Map Reduce**:- It is a processing part

Benefits of using Hdfs

1. **Moving Computation is Cheaper than Moving Data**
   A computation requested by an application is much more efficient if it is executed near the data it operates on. This is very true when data set is of huge. It helps in minimizing network congestion and increasing the overall throughput of the system. The assumption is that it is often better to migrate the computation nearer to where the data is located instead of moving the data to where the application is running. HDFS provides interfaces for applications to move themselves
   closer to where the data is located.

2. **Data Replication**
   HDFS is used to reliably store very large files. It stores each and every file as a sequence of blocks; all blocks in a file except the last block are the same size. The blocks of a file are replicated as we store blocks on commodity hardware so replication makes it fault tolerant. We can reset the block size and replication factor for each and every file. The replication factor can be specified at file creation time and can be changed later. Files in HDFS are write-once and read many times.

3. **Large Data Sets**
   HDFS is used to store large data sets. A typical file in HDFS is huge it terms to gigabytes to terabytes in size. It provides high aggregate data bandwidth and scale to hundreds of nodes in a single cluster. It ought to support tens of millions of files in a single instance.

4. **Portability in various Hardware and Software Platforms**
   HDFS is designed in such a manner that its easy to port it from one platform to another. This facilitates widespread adoption of HDFS as a platform of choice for a large set of applications.

5. **Rack Awareness**
   It is a knowledge how different data nodes are distributed across the racks of a Hadoop Cluster. In a large cluster of Hadoop, in order to improve the network traffic while reading/writing HDFS file, name node chooses the data

node which is closer to the same rack or nearby rack to Read/Write request. Name node gets rack information by maintaining the rack ids of each data node. This concept that chooses closer data nodes based on the rack information is called Rack Awareness in Hadoop.

Why Rack Awareness?

- Rack Awareness improves data availability and reliability.
- Rack Awareness improves the performance of the cluster.
- It improves network bandwidth.
- To avoid losing data if entire rack fails though the chance of the rack failure is far less than that of node failure.
- To keep bulk data in the rack when possible.
- An assumption that in-rack ids higher bandwidth, lower latency.

Replica Placement via Rack Awareness in Hadoop

- Placement of replica is vital for providing high reliabileness and performance of HDFS. We can optimize replica placement through rack awareness and this thing distinguishes HDFS from different Distributed File System. Block Replication in various racks in HDFS is done using a policy as follows:
- "No more than one duplicate is placed on one node and no more than two replicas are placed on the similar rack. This incorporates a constraint that the range of racks used for block replication should be less than the total number of block replicas".

Advantages of Implementing Rack Awareness

- Maximize network bandwidth and low latency
- Data protection against rack failure

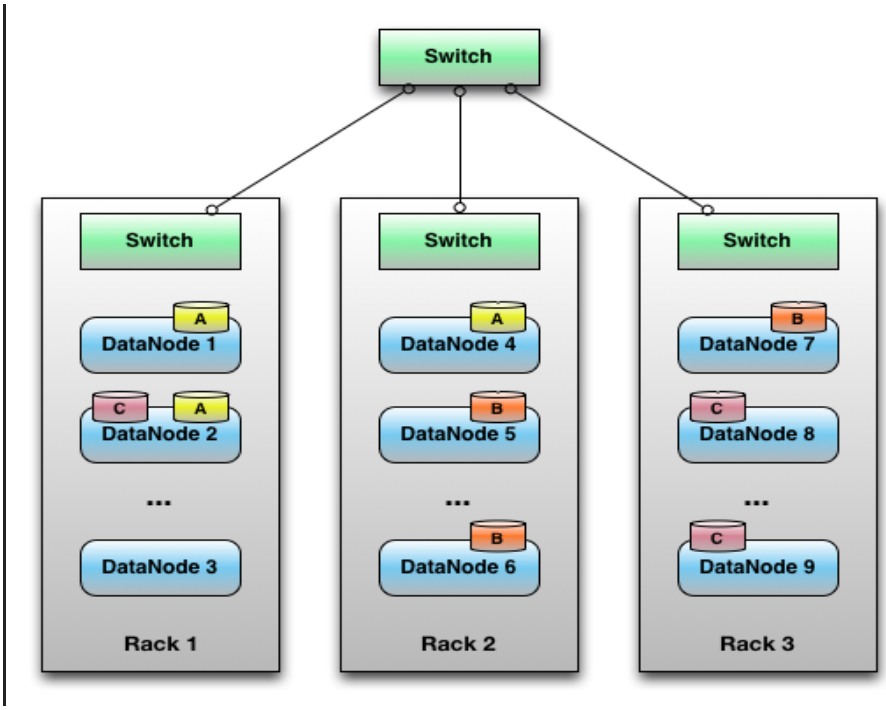

Fig. 1: Hadoop Rack

HDFS Read Operation

- Client contacts the Namenode by using Rpc(Remote procedure calls)
- Namenode check its metadata and give address of datanodes that have copy of that block
- Namenode also gave information about the proximity of datanode to the client
- Afterwards client reads data directly from datanode.

Daemons of Hdfs

Name Node:-The Name Node is the brain of an HDFS file system. It keeps the directory tree of all files in the file system, and tracks where across the cluster the file data is kept. It does not store the data of these files.

- Keeps a track of what blocks make up a file and the location of those blocks in the cluster. It does not contain any cluster data in itself.
- Directs the Clients to the Data Nodesthey need to talk to and keeps a track of the cluster's storage capacity and the condition of each Data Node.
- Ensures each block of data is meeting the minimum defined replica policy. Which is 3
- Name Node stores metadata of actual data. e.g. filename, path, No. of Blocks, Block IDs, Block location, no. of replicas, and also Slave related configuration.
- It manages Filesystem namespace.
- Name Node regulates client access to files.
- Name Node executes file system namespace operation like opening/closing files, renaming files/directories.
- As Name Node keep metadata in memory for fast retrieval. So it requires the huge amount of memory for its operation.
- Name Node does not store the actual data or the dataset. The data itself is actually stored in the Data Nodes.
- If it doesnt get heartbeat from datanode in every 3 seconds it marked that particular data node as dead
- It contains Fsimage and edits
- Data Node death may cause the replication factor of some blocks to fall below 3
- It constantly tracks which blocks need to be replicated and initiates replication whenever necessary.

fsimage –It contains the entire state of the file system at a point in time. Each and Every file system modification is assigned a unique, monotonically increasing transaction ID. An fsimage file represents the file system state after all modifications up to a specific transaction ID.

Edits – An edits file is a log that lists each file system change (file creation, deletion or modification) that was made after the most recent fsimage.

Secondary Name node:-it is used to recover metadata in case Name node fails

- It is master daemon
- It is used for check pointing process
- It uses enterprise hardware
- Stores a copy of FsImage file and edits log.
- It periodically applies edits log records to FsImage file and refreshes the edits log. And then it sends this updated FsImage file to Name Node so that Name Node doesn't need to re-apply the Edit Log records during its start up process. Thus Secondary Name Node makes Name Node start up process fast.

Data Node:-A Data Node stores data in the [Hadoop File System]. A functional file system has more than one Data Node, with data replicated across them.

- It is responsible for storing the actual data in HDFS.
- It is also known as the Slave
- Data node send heartbeat to Name node in every 3 seconds.
- When a Data Node announce itself to the Name Node along with the list of blocks it is responsible for.
- When a Data Node goes down, it does not affect the availability of data or the cluster. Name Node will arrange for replication for the blocks managed by that particular Data Node which is not available.
- It is usually configured with a lot of hard disk space. Because the actual data is stored in the Data Node.
- It uses commodity hardware(jbod)
- We can have any number of data nodes in a cluster

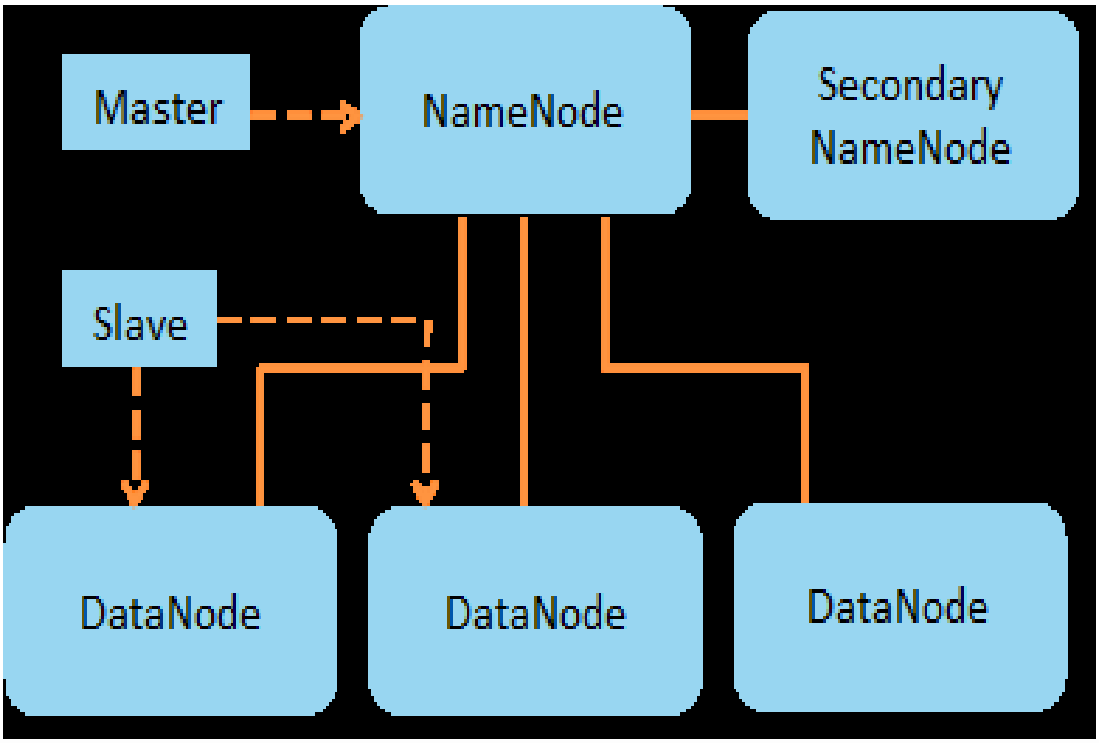

Fig. 2: Daemons of Map Reduce

Job Tracker:-:-it provides resources for job and monitors the job

- It runs on an enterprise hardware.
- Its an essential Daemon for Map Reduce execution in MRv1.
- It receives the requests from client for Map Reduce execution.
- It talks to the Name Node to determine the location of the data.
- It finds the best Task Tracker nodes to execute tasks based on the data locality (proximity of the data) and the available slots to execute a task on a given node.
- It monitors the individual Task Trackers and the submits back the overall status of the job back to the client.
- It is critical to the Hadoop cluster in terms of Map Reduce execution.
- When it is down, HDFS will still be functional but the Map Reduce execution can not be started and the existing Map Reduce jobs will be halted.

Task Tracker:- it executes the job which are given by job tracker

- It is replaced by Node Manager in MRv2.
- Mapper and Reducer tasks are executed on Data Nodes administered by Task Trackers.
- Mapper and Reducer are assigned to task trackers to execute tasks by Job Tracker.
- It remain in constant communication with the Job Tracker signalling the progress of the task in execution.
- Failure of task tracker is not considered fatal. When a Task Tracker becomes unresponsive, Job Tracker will assign the task executed by the Task Tracker to another node.
- It is a slave daemon
- It runs on commodity hardware
- Task Tracker runs on Data Node. Mostly on all Data Nodes.
- It is one per job

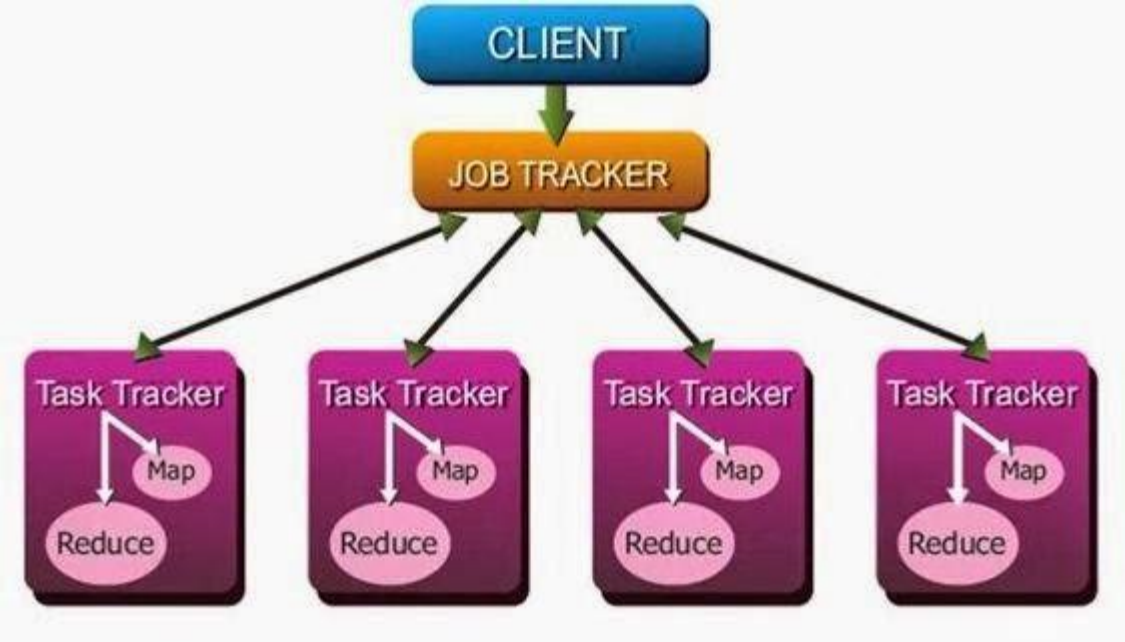

Fig. 3: Map Reduce Processing

LIMITATIONS OF HADOOP 1

- Name node is single point of failure in Hadoop Version 1.0
- Too much pressure on job tracker

REFERENCES

[1]. https://www.quora.com/What-is-NameNode-in-Hadoop
[2]. https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html#NameNode+and+DataNodes
[3]. http://hadoop.apache.org/core/docs/current/api/
[4]. http://hadoop.apache.org/hdfs/version_control.html
[5]. https://www.quora.com/What-exactly-is-a-Namespace-EditLog-FSImage-and-Metadata-in-hadoop
[6]. https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html#NameNode+and+DataNodes
[7]. http://hadoopinrealworld.com/jobtracker-and-tasktracker/
[8]. http://data-flair.training/blogs/rack-awareness-hadoop-hdfs/