BIG DATA – IMPORTANCE OF HADOOP DISTRIBUTED FILESYSTEM

Ankur Chaudhary, Pankaj Singh

Abstract - When a dataset outgrows the storage capacity of a single physical machine, it becomes necessary to partition it across a number of separate machines. Filesystems that manage the storage across a network of machines are called distributed filesystems. Since they are network-based, all the complications of network programming kick in, thus making distributed filesystems more complex than regular disk filesystems. It's not easy to measure the total volume of data stored electronically, but an IDC estimate put the size of the "digital universe" at 0.18 zettabytes in 2006, and is forecasting a tenfold growth by 2011 to 1.8 zettabytes.* A zettabyte is 1021 bytes, or equivalently one thousand exabytes, one million petabytes, or one billion terabytes. This flood of data is coming from many sources as mentioned below –

- The New York Stock Exchange generates about one terabyte of new trade data perday.
- Facebook hosts approximately 10 billion photos, taking up one petabyte of storage.
- Ancestry.com, the genealogy site, stores around 2.5 petabytes of data.
- The Internet Archive stores around 2 petabytes of data, and is growing at a rate of 20 terabytes per month.
- The Large Hadron Collider near Geneva, Switzerland, will produce about 15 petabytes of data per year.

Index Terms— Hadoop, Big Data, HDFS, Distributed Filesystem, Namenode, DataNode, dfsadmin

1. INTRODUCTION

HDFS is a distributed, scalable, and portable file system written in Java for the Hadoop framework. HDFS is the answer of storage industry for unstructured and huge amount of data which incurs huge amount of cost and fault tolerance. It is a fault tolerant file system designed to store data in a reliable manner even if failures like namenode, datanode and network occur. It works on a master slave architecture wherein a master server manages access to files and slave for storing user data via data nodes.

An advantage of using HDFS is data awareness between the job tracker and task tracker. It is used for Big Data and compliments Online Analytical Processing (OLAP) and Online Transactional Processing (OLTP).

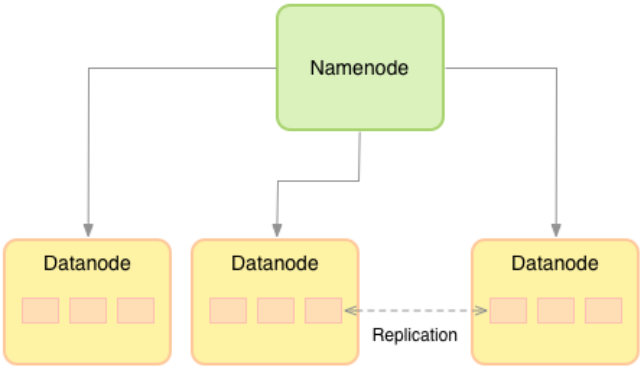
2. WHEN DO WE NEED HADOOP DISTRIBUTED FILE SYSTEM?

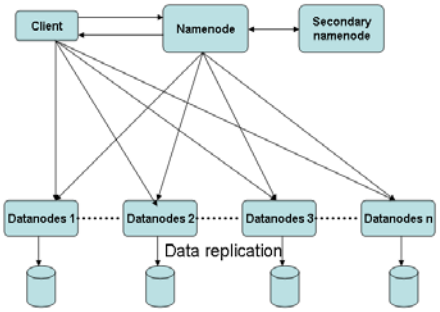
Handling small amount of data like few GB of data is possible in traditional distributed file systems .The real issue comes when we deal in so much data i.e. data in 1000s of TBs that it becomes unrealistic to be stored via traditional DFS, that's where HDFS comes into picture. It handles this data via parallel computing. It is highly reliable as it stores data via replication i.e. storing a cluster* of information on another server as a backup so that thefailure of a server does not impact the availability of that cluster of data.

HDFS can be used with commodity hardware and there is no need to share disks between different servers.

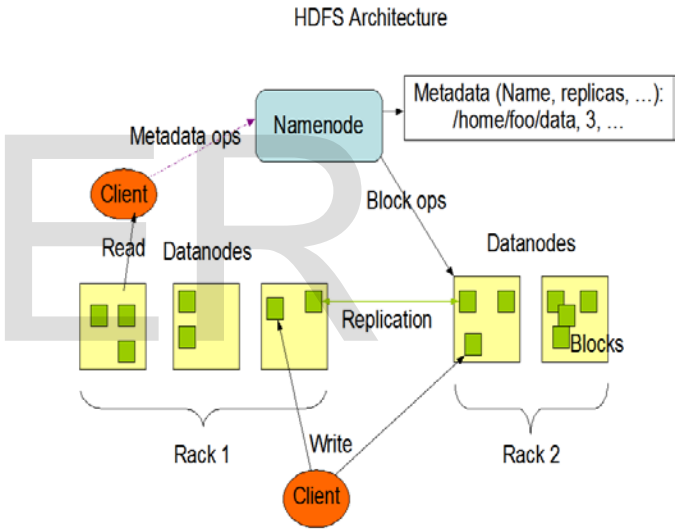
Namenode: It manages File system namespace and also handles File access by clients

Datanode: It manages storage associated with the nodes





Cluster: A Cluster is a collection of different racks in a network.
Rack: A rack is a collection of different nodes (computers) in a network. There are typically around 30 computers or nodes in a rack.



3. PREREQUISITES FOR HDFS

Following types of node setups are available:

Single Node Setup is done for first time users. Both Datanode and Namenode are on the same machine. **Cluster Setup** for large, distributed clusters. Namenode and Datanode are on different machine. Software (Java 1.6.x must be installed and ssh must be installed and sshd must be running) must be installed on all the machines.

4. HOW DOES IT WORK?

A Hadoop cluster consists of one Namenode* and multiple Datanodes*. HDFS follows Master-Slave Architecture *

NameNode: Master

DataNode: Slave

Let us take an example of a File which stores information of all employees in a company across the globe .The information is stored in different servers segregated by the countries. Now in Hadoop the entire information will be stored in a cluster and to obtain entire information individual from all the servers is needed. Now to safeguard the information against failure of invidual servers, hadoop makes copies of the data on a server on 2 additional servers. This increases data availability in hadoop.

Master-Slave Architecture: An architecture where one device has unidirectional control over other device/s .In this case Namenode has unidirectional control over DataNode.

In HDFS a File is broken into one or more blocks, which stores information by a set of datanodes. The Namenode maps datanodes to blocks and performs operations like Open,Close and Rename files/Directories. The datanodes on the other hand perform operations like block creation; Deletion as instructed by Namenode and also serves read and write requests on the file.

5. How does applications access data in HDFS?

Applications access data in HDFS using File system* Java API. A C language wrapper for this Java API is also available.

File System (FS) shell includes various shell-like commands that directly interact with the Hadoop Distributed File System (HDFS) as well as other file systems that Hadoop supports, such as Local FS, HFTP FS, S3 FS, and others.

Command	Description
bin/hadoop dfs -rmr /dir2	Remove a directory named /dir2
bin/hadoop dfs -mkdir /dir1	Create a directory named /dir1

NameNode Commands:

Used to Run NameNode of HDFS

Command	Description
hadoop namenode [-format]	Formats NameNode
hadoop namenode [-upgrade]	Upgrades NameNode to new hadoop version.
hadoop namenode [-rollback]	Rollback the namenode to the previous version.
hadoop namenode [-finalize]	Removes previous version of file system. Rollback is not available after this operation.
hadoop namenode [-importCheckpoint]	Loads image from a checkpoint directory and saves it into the current one.

DataNode Commands:

Used to Run DataNode of HDFS

Command	Description
hadoop datanode [-rollback]	Rollback the datanode to the previous version.

dfsadmin commands:

Used to run HDFS dfsadmin client

Command	Description
hadoop dfsadmin [-conf | D | fs | jt |files] [-report]	Displays basic filesystem information
hadoop dfsadmin [-conf | D | fs | jt |files] [-safemode enter | leave | get | wait]	Does not accept changes to Namespace .It is used for Namenode maintenance.
hadoop dfsadmin [-conf | D | fs | jt |files] [-refreshNodes]	Refreshes Nodes and excludes datanodes that are decommissioned.

hadoop dfsadmin [-conf | D | fs | jt |files] [-finalizeUpgrade]	Datanodes and Namenodes remove their previous versions.
hadoop dfsadmin [-conf | D | fs | jt |files] [-upgradeProgress status | details | force]	Requests the details of current upgrade or force the upgrade to proceed.
hadoop dfsadmin [-conf | D | fs | jt |files][-metasave filename]	Saves Namenode's primary data structures to filename specified by hadoop.log.dir property.
hadoop dfsadmin [-conf | D | fs | jt |files][-setQuota <quota> <dirname>...<dirname>]	Set the quota for each directory
hadoop dfsadmin [-conf | D | fs | jt |files] [-clrQuota <dirname>...<dirname>]	Clears the quota for each directory

6. LIMITATIONS OF HDFS:

HDFS was designed for mostly immutable files* and may not be suitable for systems requiring concurrent write operations. Another limitation of HDFS is that it cannot be mounted directly by an existing operating system.

**Immutable files: Immutable files prohibit changes are files which are locked (via immutable flags) i.e. cannot be modified. A file whose immutable flags have been enabled cannot be modified, despite the user having Read & Write permissions on that object.*

7. CONCLUSION:

HDFS is a distributed file system which gives high performance when used with large Clusters of data. However its performance is low when we deal with small volumes of data where smaller I/O operations are involved due to performance bottleneck. Thus HDFS is not the preferred medium of storage in case of web applications. It is optimum to work with commodity hardware, thus minimizing costs with increased performance as compared to traditional DFS. Therefore HDFS is the market leader in Distributed file systems in most of the organizations dealing with large chunks of data.

8. REFERENCES

[1] Dhruba Borthakur, the Hadoop Distributed File System: Architecture and Design, pages 137–150, 2004.
[2] Apache Hadoop Project. Available: http://hadoop.apache.org.
[3] Hadoop Distributed File System. Available: http://hadoop.apache.org/common/docs/current/hdfs design.html..
[4] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. In OSDI'04: Proceedings of the 6th Symposium on Opearting Systems Design & Implementation, pages 10–10, 2004.
[5] R. Abbott and H. Garcia-Molina. Scheduling I/O requests with deadlines:A performance evaluation. In Proceedings of the 11th Real-Time Systems Symposium, pages 113–124, Dec 1990.
[6] S. Ghemawat, H. Gobioff, S. Leung. "The Google file system," In Proc. of ACM Symposium on Operating Systems Principles, Lake George,NY, Oct 2003, pp 29–43.
[7] S. Weil, S. Brandt, E. Miller, D. Long, C. Maltzahn, "Ceph: A Scalable, High-Performance Distributed File System," In Proc. of the 7th Symposium on Operating Systems Design and Implementation, Seattle, WA, November 2006.
[8] Apache Hadoop. http://hadoop.apache.org/
[9] R. Chaiken, B. Jenkins, P. ake Larson, B. Ramsey, D. Shakib, S. Weaver, and J. Zhou. Scope: easy and efficient parallel processing of massive data sets. PVLDB, 1(2):1265–1276, 2008.