Hadoop: Addressing Challenges of Big Data

Kamalpreet Singh
Assistant Professor, CSE Department
Lovely Professional University
Jalandhar, India
kamalpreet.17706@lpu.co.in

Ravinder Kaur
Assistant Professor, CSE Department
Lovely Professional University
Jalandhar, India
ravinder.17686@lpu.co.in

Abstract— **Hadoop is an open source cloud computing platform of the Apache Foundation that provides a software programming framework called MapReduce and distributed file system, HDFS. It is a Linux based set of tools that uses commodity hardware, which are relatively inexpensive, to handle, analyze and transform large quantity of data. Hadoop Distributed File System, HDFS, stores huge data set reliably and streams it to user application at high bandwidth and MapReduce is a framework that is used for processing massive data sets in a distributed fashion over a several machines. This paper gives a brief overview of Big Data, Hadoop MapReduce and Hadoop Distributed File System along with its architecture.**

Keywords—Big Data;Hadoop;MapReduce;HDFS; Namenode; Datanode

I. INTRODUCTION AND RELATED WORK

Big data (also spelled Big Data) [1] is a general term used to describe the voluminous amount of unstructured and semi-structured data that a company generates, the data that would take too much time and cost too much money to load into a relational database for analysis. Big Data refers to large collection of data (that may be structured, unstructured or semi structured) that expands so quickly that it is difficult to manage with regular database or statistical tools. Hadoop [2] is a framework written in Java, distributed under Apache License, developed by Doug Cutting. Hadoop is basically used for analyzing and processing Big Data. Hadoop supports the running of application on Big Data, and addresses three main challenges (3V) created by Big Data-

- Volume - Hadoop provides framework to scale out horizontally to arbitrarily large data sets to address volume of data.
- Velocity - Hadoop handles furious rate of incoming data from very large system.
- Variety - Hadoop supports complex jobs to handle any variety of unstructured data.

Hadoop is a whole ecosystem of projects that work together to provide a common set of services. It transforms commodity hardware into a coherent service that stores petabytes of data reliably and also processes that data efficiently through huge distribution.

A key characteristic of Hadoop is the partitioning of both data and computation across number of hosts reliably, and then executing application computations in parallel close to their required data. The key attributes of Hadoop are that it is redundant and reliable, that is if you lose a machine due to some failure, it automatically replicates your data immediately without the operator having to do anything, it is extremely powerful in terms of data access and is preliminary batch processing centric and makes it easier to distributed applications using MapReduce software paradigm. Moreover it runs on commodity hardware which cuts off the cost of buying special expensive hardware and RAID systems.

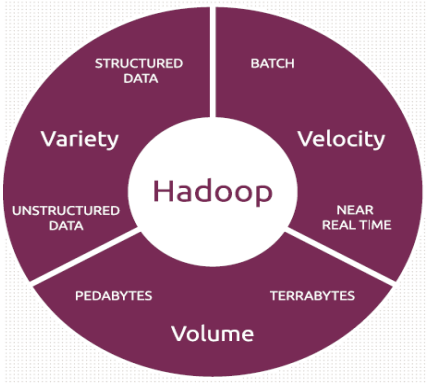

Figure 1: Various Challenges of Big Data

In traditional approach, as depicted in Figure 1, an enterprise uses a powerful computer to process the data available. But there is an upper limit to the amount of data processed because it is not scalable and Big Data grows with great velocity and variety, whereas Hadoop follows a very different approach as compared to the traditional enterprise approach. In this Big Data is first broken into smaller pieces so that large amount of data can be handled efficiently and effectively. Along with this data segregation, Hadoop breaks the computations into pieces as well that need to be performed on the data. Once all the child computations are finished, the results are combined together and sent back to the application.

.

978-1-4799-2572-8/14/$31.00 ©2014 IEEE

II. ARCHITECTURE

At a very high level, Hadoop has two main components- MapReduce and filesystem, HDFS as shown in Figure 2. MapReduce is the processing part of Hadoop and manages the jobs. HDFS refers to Hadoop Distributed File System stores all the data redundantly required for computations. Projects here refer to set of tools that are managed by Apache under the umbrella of Hadoop to provide assistance in the task related to Hadoop. Hadoop's origin come from Google File System GFS [3] and MapReduce [4] which become Apache HDFS and Apache MapReduce respectively.

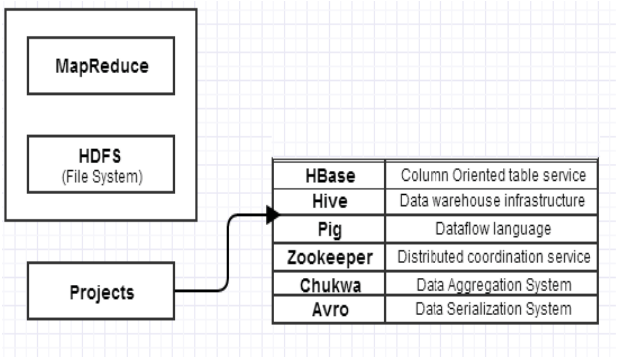

Figure 2: High Level Architecture of Hadoop

A. MapReduce

Hadoop MapReduce is a software programming model as well as an associated implementation for processing large data sets in a distributed fashion on large clusters of relatively inexpensive commodity hardware in an efficient reliable, fault tolerant manner. A MapReduce task divides the input data set across various independent chunks which are processed in a completely parallelized manner by user specified *map* function that generates a set of intermediate < key, value > pairs.

The output of these map functions, after being sorted by the framework, serve as input to the user reduce function which reduces/merges all < key, value > pairs associated with same intermediate key. The MapReduce framework takes the responsibility of splitting the input data across several machines, scheduling the task execution as well as monitoring and re-executing the failed tasks along with managing the required inter-machine communications. As the programs written in MapReduce styles are automatically parallelized, the programmers could focus on writing scale free programs thus making programming easier. These scaled programs are executed on large clusters of commodity machines which are relatively inexpensive. Thus, with help of MapReduce software framework, programmers with little knowledge of parallel and distributed systems can also easily make use of the resources of a large distributed system.

- Working of MapReduce:

The MapReduce framework maps the input dataset into <key, value> pairs and produces a set of output <key, value> pairs of merging all the pairs associated with the same intermediate key, that is, MapReduce works exclusively on <key, value> pair. The two operations in MapReduce, i.e. map and reduce come from functional programming languages which pass functions as arguments to other functions. Following is the step wise description of execution of MapReduce

1. MapReduce framework splits the input data into segments.

2. These segments are then passed different machines/clusters for computation.

3. Map script, which is written by user, runs on each machine to process portion of data given to it and produces a set of intermediate <key, value> pair.

4. The pairs with same key are grouped together by the MapReduce library and passed to Reduce function for aggregation.

5. Reduce script, which is also written by the user, takes collection of intermediate key I along with their set of values for that keys and merges them according to the user specified script into a smaller set of values. Thus, the reduce function aggregates the values of the collection of intermediate <key, value> pairs having the same key.

Below is the diagrammatic explanation of above explained scenario taking WordCount example. WordCount [5] is a MapReduce application that outputs the total number of occurrences of each word in a given input data.

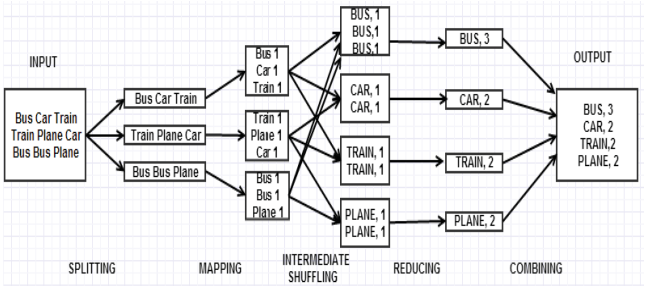

Figure 3: Workflow of MapReduce

The map function emits each word as <key> and its associated amount of occurrence as <value>. Value is taken as '1' in this WordCount example for simplification. The reduce function aggregates all count emitted for a particular word <key> together.

- Implementation

MapReduce framework is the processing part of Hadoop which manages the jobs. It has two parts - *Job Tracker* and *Task Tracker.*

Having multiple machines with Hadoop creates a cluster. For a gulf of clusters, there is a single master Job Tracker and one slave Task Tracker per cluster node. The master Job Tracker is responsible for accepting the users' jobs, dividing it into smaller components and scheduling these computations to the individual slave task trackers. It also takes the responsibility of monitoring the task tracker and re-executes the task if a task tracker fails. Thus, Job Tracker acts as a coordinator for

MapReduce. The MapReduce server on a typical machine is called Task Tracker which is responsible for running the tasks assigned by the Job Tracker and reports the Job Tracker once the task assigned to it is completed. Job tracker then combines the results and sends the final result back to the application.

- Fault Tolerance for Computation

Hadoop is built keeping in mind the hardware and software failures. It is inbuilt for fault tolerance for task tracker service running on the slave computers. If any of the clusters fails or if just the task tracker service fails, Job Tracker through regular monitoring detects the failure and delegates the same job to other working task tracker. Fault tolerance is not only limited to the slave task trackers. In order to avoid single point of failure in case if master Job tracker fails, enterprise version of Hadoop keeps two master, one as main master and the other as back up master in case the master dies.

B. HDF S

The HDFS [6] is distributed file system component of Hadoop designed to store and support large amount of data sets on commodity hardware reliably and efficiently. Like other file system, PVFS [7], Luster, Hadoop Distributed File System also has master / slave architecture, with master as Name Node and slave as Data Node. HDFS stores all the filesystem metadata on single Name Node. However unlike Luster and PVFS, HDFS uses replication of data stored on Data Node to provide reliability instead of using data protection mechanism such as RAID. The application data is stored as multiple copies on a number of slave Data Nodes, which are usually one per node in cluster.

- NameNode

The master Name Node as a coordinator of HDFS manages the filesystem namespace by keeping index of data location and regulates access to files by clients. Files and directories are represented on Name Node and it executes operations like opening, closing and renaming files and directories. NameNode itself doesn't store any data neither does any data flows through the name node. It only determines and keeps track of mapping of file blocks to Data Node, thereby acting as a repository for all HDFS metadata. Any application requiring data, first contacts the NameNode which provides locations of data blocks containing the file. While storing/writing data to HDFS, NameNode chooses a group of (by default three) to store the block replicas. The client application then pipelines the data to the Data Node nominated by Name Node.

- DataNode

Data Nodes are responsible for storing the blocks of file as determined by the Name Node. Data file to be stored is first split into one or more blocks internally. Data Nodes serve the read write requests from file system's client data. These are also responsible for creating, deleting and replicating blocks of file after being instructed by the Name Node.

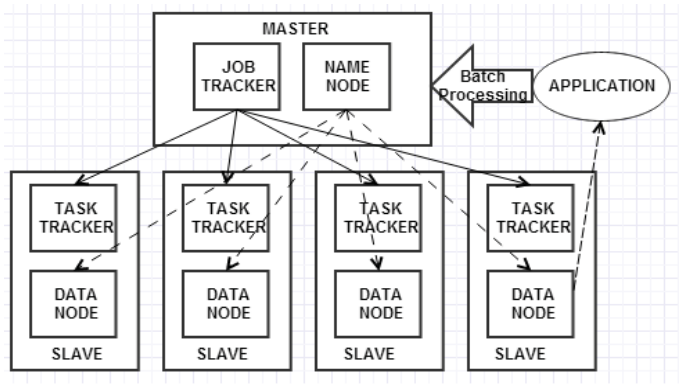

Figure 4: Master/Slave Architecture of Hadoop

- Fault Tolerance for Data

In Hadoop, failures are treated as norms rather than exception. The master Name Node makes sure that each block always has the required number of replicas through *Heartbeat* [6] and *block report* messages received from Data Nodes. Name Node also prevents Data Node from being over replicated and under replicated by evenly distributing the block of file and its replicas. Thus if a Data Node fails due to some reason, data can still be retrieved from other Data Nodes having its replica.

Name Node is the single point of availability failure. If a Name Node goes down, the Data Nodes would not be able to make any sense of the data blocks on them. So in order to avoid this single point of failure, Enterprise version of Hadoop keeps two *masters,* one as main master and the other as backup master in case the main master fails.

III. INCREASED THROUGHPUT BY MIGRATING COMPUTATIONS RATHER THAN DATA

When the size of data is very large, it is always more efficient to perform the computations requested by an application closer to where the data it needs to operate on is stored. In Hadoop, MapReduce framework and Hadoop Distributed File System run on the same set of nodes, that is, the storage and compute nodes are the same. This configuration of Hadoop framework minimizes network congestion and increases bandwidth across the clusters as it schedules the computation closer to where data (or its replica) is present rather than migrating the large data sets to where application is running. This increases the overall throughput of the system.

IV. REPLICA PLACEMENT

The placement of replica is critical to HDFS reliability and performance. HDFS acts as a self-healing system. As depicted in the figure suppose the second data node fails, we still have two other DataNodes having the required data's replicas. If a DataNode goes down, then the *heartbeat* from DataNode to NameNode will cease and after ten minutes NameNode will consider that DataNode to be dead and all the blocks that were stored on that DataNode will be re-replicated and distributed evenly on other living DataNodes.

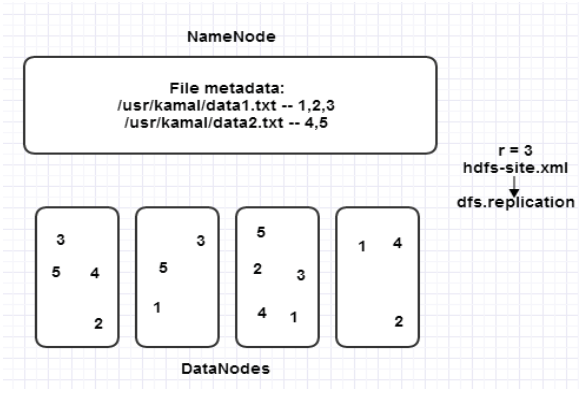

Figure 5: Replica Placement with replication value =3

V. RACK AWARENESS POLICY

An HDFS file consists of blocks. Whenever a new block is required to store the data, the NameNode allocates a block with a unique block ID and determines a list of DataNodes to host replicas of the block. Data is then pipelined from the client to the DataNodes in following the sequence of minimum distance from the client. As shown in the figure, nodes are spread across multiple racks that share a switch connected by one or more core switches. The NameNode, acting as a central place maintains the metadata that helps in resolving the rack location of each DataNode. The main motive of rack aware replica policy is to improve reliability and availability of data along with network bandwidth utilization. The default HDFS rack aware replica policy is as follows:

- No DataNode should contain more than one replica of any block of file.
- No rack should contain more than two replicas of the same block, provided there are sufficient numbers of racks on the cluster.

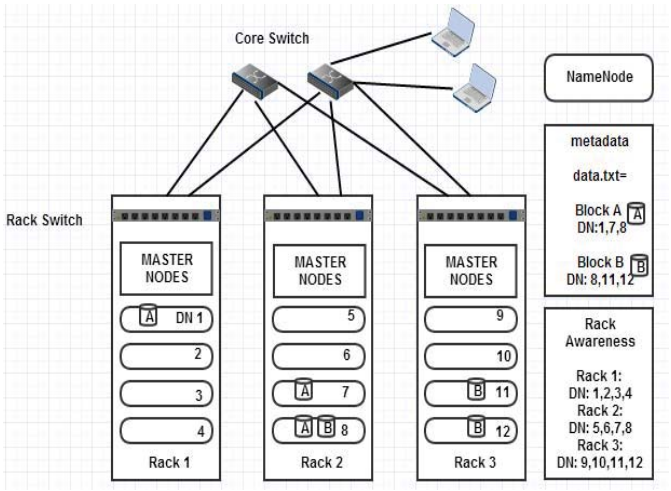

Figure 6: Rack Awareness

The default HDFS block placement policy provides a tradeoff between minimizing the write cost, and maximizing data reliability, availability and aggregate read bandwidth. When a new block is created, HDFS places the first replica on the node where the writer is located, the second and the third replicas on two different nodes in a different rack, and the rest are placed on random nodes with restrictions that no more than one replica is placed at one node and no more than two replicas are placed in the same rack when the number of replicas is less than twice the number of racks. The choice to place the second and third replicas on a different rack better distributes the block replicas for a single file across the cluster. If the first two replicas were placed on the same rack, for any file, two-thirds of its block replicas would be on the same rack.

CONCLUSION

Hadoop was originally designed for processing batch-oriented processing jobs, such as creating web page indices or analyzing log data. Hadoop is not used for OnLine Transaction Processing workloads and OnLine Analytical Processing or Decision Support System workloads where data are randomly or sequentially on structured data like a relational data to generate reports that provide business intelligence. However being reliable, (both in terms of computation and data), fault tolerant, scalable and powerful, Hadoop is now widely used by Yahoo!, Amazon, eBay, Facebook, IBM, Netflix, and Twitter to develop and execute large-scale analytics or applications for huge data sets. MapReduce framework of Hadoop also eases the job of programmers as they need not to worry about the location of data file, management of failures, and how to break computations into pieces as all the programs written are scaled automatically by Hadoop. Programmers only have to focus on writing scale free programs.

REFERENCES

[1] Zach Brown. (2014, January 22). Big Data (2nd ed.) [Online]. Available: http://technologyadvice.com/category/big-data/

[2] T. White, Hadoop: The Definitive Guide. O'Reilly Media, Yahoo! Press,June 5, 2009..

[3] 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.

[4] J. Dean, S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," In Proc. of the 6th Symposium on Operating Systems Design and Implementation, San Francisco CA, Dec. 2004,

[5] WordCount Example, June 10, 2011 [Online] http://wiki.apache.org/hadoop/WordCount

[6] Konstantin Shvachko, Hairong Kuang, Sanjay Radia, Robert Chansler. "The Hadoop Distributed File System", Yahoo!, IEEE 2010

[7] P. H. Carns, W. B. Ligon III, R. B. Ross, and R. Thakur. "PVFS: A parallel file system for Linux clusters," in Proc. of 4th Annual Linux Showcase and Conference, 2000, pp. 317–327.