A Survey: Hadoop, Managing Large Data very efficiently

Yadav Krishna R (M.Tech,) PIET, Vadodara

Ms. Purnima Singh (Assistant Professor) PIET, Vadodara

***Abstract*—Now-a-days, the amount of digital information is increasing at a high speed. The majority of this data will be "unstructured"–complex data mainly poorly-suited to management by structured storage systems like RDMS (Relational Database Management System). Unstructured data mainly come from many sources and takes many forms like web logs, text files, sensor readings, video, audio and images. Complex Unstructured data can hide important insights. The companies which are able to extract the important data from huge volume of data can better control process and costs, can better predicate demand and can better products.**

Big data mainly deals with two important things: firstly, inexpensive, reliable storage. Secondly, need new tools for analyzing unstructured and structured data. Hadoop is a powerful open source software platform which can address both the above problems. Hadoop mainly contains two important components: HDFS and MapReduce. HDFS is used for storage purposes and MapReduce is programming paradigms for distributed platform, where a cluster of computers are connected with each other through networking and useful in solving certain problems of computing cluster.

***Index Terms*--- MapReduce, Hadoop, HDFS, Big data, RDMS.**

I. INTRODUCTION

*Hadoop*is a java based computing framework which supports MapReduce paradigm. Hadoop has execute-once semantics, meaning that with iterative tasks all state information needs to be written to file and then read back in for every step of computation. *Hadoop Distributed File System* (HDFS) is used to store data for input process and store output files [9]. Many other popular tools are available for managing enterprise data like RDMS and it is mostly designed to make simple queries to run quickly and uses techniques like indexing to examine simple small portion of available data. Hadoop is little different tool. It aimed at problems which require examination of all the available data. Text analysis and image processing require that every single record to be read, and often interpreted in similar records. Hadoop having MapReduce to carries out this tedious analysis quickly.

II. HDFS

HDFS is known as reliable and fault-tolerance storage. HDFS is able to store huge amounts of data and scale up incrementally and can very easily survive the failure of significantly parts of storage infrastructure without losing data. Hadoop creates clusters of machines and co-ordinates jobs between them. Here, main advantages of Hadoop are that clusters can be built with inexpensive computers or PC.If one computer fails, Hadoop can continues its operation with cluster without losing any of data or interrupting work, by shifting work to remaining computers in the cluster.

HDFS (Hadoop Distributed File System) use to manage storage on cluster by breaking incoming files into pieces of data, called "blocks," and storing each blocks redundantly across the pool of machines or computers.

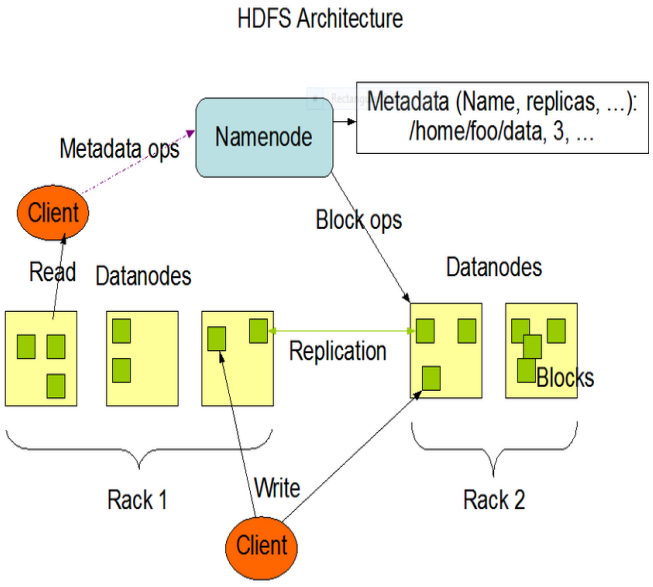

Fig. 1 HDFS Architecture[10]

Hadoop has many useful features. In fig. 1, it is clearly shown that data replicas are present and hence if any machine fails, the entire file will still be available. HDFS use to notice when a block of data or node is lost and creates a new copy of missing data and replace it.

A. NameNode and DataNodes

HDFS has a master-slave architecture. It consists of single NameNode also called master server or node that manages file system namespace and regulates access to files by clients or process. There are number of DataNodes, usually one per node in cluster, which take care of storage attached to nodes that they run on. HDFS having a file system namespace and allows data to be stored in files. Generally, a file split into one

***Yadav Krishna R**, M.Tech, Parul Institute of Engineering and Ttechnology, Vadodara, Gujarat., +91-9033-493036*
***Ms.Purnima Singh**, Assistant Professor, Parul Institute of Engineering and Technology, Vadodara, Gujarat.*

or more blocks and further, blocks are stored at DataNodes. The NameNode operates file system namespace and perform operations like opening, closing, and renaming files and directories. It also maps the blocks to DataNodes. DataNodes are responsible for read and write operations from the file system client's request. DataNodes also do operations like block creation, deletion and replication following instruction from NameNode. The DataNodes and NameNodes are software designed to run of common machines. These machines run on GNU/Linux operating system. HDFS created by using Java Language and any machine that works on java can run NameNode and DataNode software. HDFS can be deployed on a wide range of machines. A deployment has a dedicated machine that runs only NameNode software. Other machines in cluster run on instance of DataNode software [10].

The Existence of single NameNode in cluster greatly simplifies the Hadoop system. The NameNode is arbitrator and repositiory for all HDFS metadata.

B. File System Namespace

HDFS support hierarchical file organization. An application can create directories and store files in these directories. The file system namespace hierarchy is much similar to other existing file systems, where one can create and remove files and move files from one directory to another, or rename a file. HDFS does not support hard links and soft links. Any change to file system namespace or its properties is recorded by NameNode. Number of replicas of file should be maintained by HDFS.

C. Data Replication

HDFS mainly recognized for reliably storing very large files across machines in a large cluster. It maintains each file as a sequence of blocks; all blocks in a file except the last block are the same size. Fault-tolerance checked by file replication. The block size and replication factor are maintained and configurable per file. Files in HDFS are writes -once and reads many time.

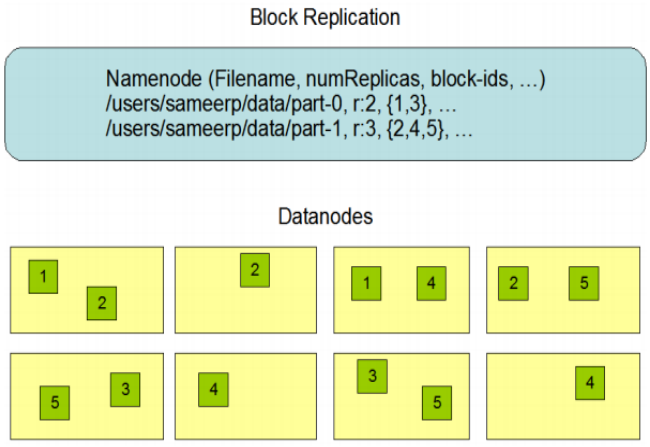

Fig. 2 Block Replication and DataNodes[10]

NameNode makes all decisions regarding replication of blocks and periodically receives a heartbeat and block report from each of DataNodes in the cluster. Heartbeat implies that DataNodes is working properly.

D. Communication Protocols

All communication within HDFS layered on top of TCP/IP protocol. A client use to establish a connection to TCP port on NameNode machine. It talks the client protocol within the NameNode. RPC (Remote Procedure call) abstraction wraps both the client Protocol and DataNode protocol. Namenode does not start RPC, but it responds to RPC issued by DataNodes and clients. It supports hierarchical file organization. An application can create directories and store files in these directories. The file system namespace

E. Data Organization

1. Data Blocks

HDFS designed to store large files. It deals with large data sets. Here, applications write their data only once but they can read it more than once times and require these reads at streaming speeds. A block size is 64 MB that being used by HDFS. Thus, HDFS file chopped into 64 MB chunks and each chunk will reside on different DataNode.

2. Staging

A client generally request for creating a file and this request does not reach NameNode immediately. Initially, HDFs caches the file data to temporary local file. The client contacts the NameNode only after the local file accumulates data over one HDFS block size. The NameNode inserts file name into file system and allocates a data block for it. The NameNode responds client request with the identity of DataNode and data blocks. Then, client used to flushes the block of data from local temporary file to specified DataNode. When file is closed, remaining unused data in local file is transfer to DataNode [10]. The client then tells the NameNode to close the file. At this, the NameNode commits file creation operation into persistent store. If NameNode dies before file is closed, the file is lost

3. Replication Pipelining

When a client deals with writing a data to HDFS file, its data must be first written to local file [11]. Let's assume that HDFS has three replication factors. When local file accumulates a full block of user data, client retrieves a list of DataNodes from NameNode. This list having the information about DataNodes, will host a replica of that block. The client then flushes data blocks to first DataNode. The first DataNode receive data in small portions (4 KB), writes each portion to local repository and transfers that portion to second DataNode in list. The second DataNode in same way receive, write to its depository and flushes that portion to third DataNode. Thus, DataNode receive and send data from previous one in pipeline.

F. Accessibility

HDFS are accessible through various applications in many different ways. HDFS provide java API for applications to use. A C language wrapper for java API is also available. HTTP browser can be used to browse the files of a HDFS instance. WebDAV protocols expose HDFS progress report.

III. MAPREDUCE

Hadoop MapReduce is framework for writing applications which process large amount of data in parallel on large

ISSN: 2278 – 1323

clusters of machines in a reliable, full-tolerant manner.MapReduce is a framework that allows certain kinds of problems particularly those involving large data sets to be computed using many computers. Problems suitable for processing with MapReduce must usually be easily split into independent subtasks that can be processed in parallel. In parallel computing such problems as known as embarrassingly parallel and are ideally suited to distribute programming [7]. MapReduce was introduced by Google and is inspired by the map and reduce functions in functional programming languages. In order to write a MapReduce program, one must normally specify a mapping function and a reducing function. The map and reduce functions are both specified in terms of data that is structured in key-value pairs. A MapReduce mainly works by splitting the input data-set into independent chunks which are processed by map tasks in a parallel manner. The framework designed in way that sorts the outputs of maps, which are then input to reduce tasks. Both input and output of the job are sorted in file system. The framework also maintains scheduling tasks, monitoring them and re-executes the failed tasks. Generally, the compute nodes and storage nodes are same; it means that MapReduce and HDFS are running on the same sets of nodes. This makes the framework to effectively schedule tasks on nodes where data is already present, resulting in very high aggregate bandwidth across the cluster [9].

The MapReduce framework having a single master JobTracker and one slave TaskTracker per cluster-node. Master node is responsible for scheduling the jobs, assign task to slave node, monitoring them and re-executing the failed tasks. The slaves execute tasks as mentioned by the master.Applications specify input/output location and supply

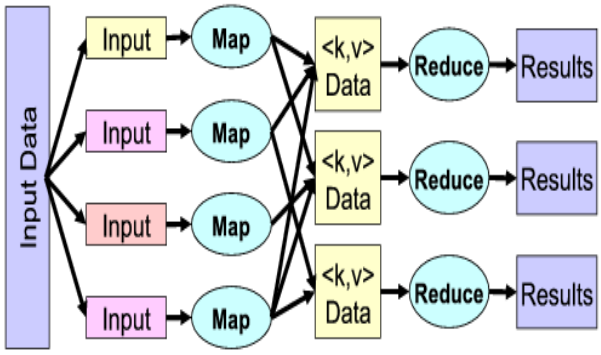

Fig. 3 MapReduce Process[11]

map and reduce functions via implementations of appropriate interfaces or abstract classes. These, and other jobs parameters, comprise of job configuration. The Hadoop job client submits the job and configuration to the JobTracker which then assumes the responsibility of distributing the software to the slaves, scheduling tasks and monitoring them, providing status and view information to the job-client. Although Hadoop framework is implemented in java, MapReduce need not be written in Java.

The MapReduce framework operates on <key, value>pairs, that is, the framework views the input as a set of <key, value> pairs and produces a set of <key, value> pairs as the output of the job, of different types. The key and value classes have to be serializable by the framework and hence need to implement the Writable interface. The keyclasses have to implement the Writable Comparable interface to facilitate sorting by the framework. Input and Output types of a MapReduce job: (input) <k1, v1> -> map-><k2, v2>-> combine-><k2, v2>-> reduce-><k3,v3> (output) [9]. This input and output are stored at HDFS.

IV. CONCLUSION

The Hadoop MapReduce architecture is easy to use, even for programmers without experience with parallel and distributed systems, since it hides the details of parallelization, fault-tolerance, locality optimization, and load balancing. A large variety of problems are easily expressible by MapReduce computations. MapReduce useto scale up large clusters of machines. MapReduce program are easy to write in many languages like java, python, perl, C++ and even in C language. Opportunity for global synchronization in MapReduce is the next big advantage between the maps and reduces phases of processing and crucially on the existence of shared global state during processing.

REFERENCES

[1] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. "The Google FileSystem." Proceedings of the nineteenth ACM symposium on Operating systemsprinciples. , pp. 29-43, ACM, New York, NY, USA, 2003.
[2] M. Zaharia et al. Improving MapReduce Performance inHeterogeneous Environments. In OSDI, pp. 29–42, 2008
[3] J. Dean and S. Ghemawat."MapReduce: A Flexible Data ProcessingTool." CACM, 53(1):72–77, 2010.
[4] C Chang, B He, Z Zhang. "Mining semantics for large scale integration on the web: evidences, insights, and challenges." SIGKDD Explorations, 2004: 6(2): pp. 67-76
[5] H. Herodotou and S. Babu. Profiling, "What-if Analysis, andCost-based Optimization of MapReduce Programs." PVLDB,4(11): pp. 1111–1122, 2011
[6] A. Floratou et al. "Column-Oriented Storage Techniques forMapReduce." PVLDB, vol. 4(7), pp. 419–429, 2011
[7] What is Hadoop? (2010), from Atbrox: http://atbrox.com/2010/02/17/hadoop/
[8] How Map and Reduce Operations are Actually Carried Out. (2009).Retrieved February 2011, from Hadoop Wiki: http://wiki.apache.org/hadoop/HadoopMapReduce
[9] Google and IBM Announce University Initiative to Address Internet-Scale Computing Challenges. (2007). Retrieved February 2011, from IBM: http://www-03.ibm.com/press/us/en/pressrelease/22414
[10] Apache Hadoop. http://hadoop.apache.org/.
[11] T. White, "Hadoop, the Definitive Guide," O'Reilly Media, May 2009.
[12] D. Jiang et al."The Performance of MapReduce: An In-depth Study." PVLDB, 3(1-2), pp. 472–483, 2010

