International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 05 Issue: 01 | Jan-2018 www.irjet.net p-ISSN: 2395-0072

Unraveling the Data Structures of Big data, the HDFS Architecture and Importance of Data Replication in HDFS

Aseema Sultana[1]

[1]*Asst.Professor, Dept. of Information Science and Engineering, HKBK CE, Bangalore, India*

Abstract – *Big Data is one of the most challenging aspects in today's world filled with emerging information. The fact that it leads us to be dealt with massive amounts of overwhelming raw data creates new opportunities in the said field. The term Big Data describes extremely large volumes of data that comes from varied sources in different formats with high velocity. Here, the amount of data is not what is important, but what can be done with the data is more important. In this process, useful information is fetched from the massive amount of data and is analyzed. The analyzing part itself is a very challenging and demanding piece of work as it demands fault forbearing, amenable and ascendible systems that work upon the said data. The ultimate goal here is to make the process of managing and maintaining data a convenient and feasible task. This paper provides a review on Big Data, its data structures, details of the HDFS architecture, and replication of data files within HDFS.*

Key Words: **Big Data, Structured, Semi-structured, Quasi-structured, Un-structured Data, Hadoop, HDFS Architecture, Data Replication, Space Reclamation.**

1. INTRODUCTION

What is Big Data? Big Data is not just large volumes of data, but also data with high variety and high velocity. In other words, Big Data is huge data in different forms being generated at high speed which is difficult to manage or process using any existing tools and architectures. Let us first see what the input data into big data systems is, it could be transactions from a bank, web server logs, chatter from Twitter or Facebook or any other social network, MP3 songs, vlogs or any broadcasted videos, content of web pages, personal documents, medical reports, and many more to count, as mentioned in [1].

According to [2], in the 1990s the data volume was measured in terabytes. In subsequent decade the data volume was being measured in petabytes. And the decade of 2010s is dealing with the exponential data volume driven by variety and many sources of digitized data. Almost everybody and everything is leaving a digital footprint. Due to the volume in data, it has been considered to start measuring data in Exabyte.

Some applications that generate a lot of data are shown in Fig -1. In the said figure we can see how the amount of information is evolving in the graph w.r.t every ten years. Data is increasing and exploding decade after decade and so are the sources for Big Data.

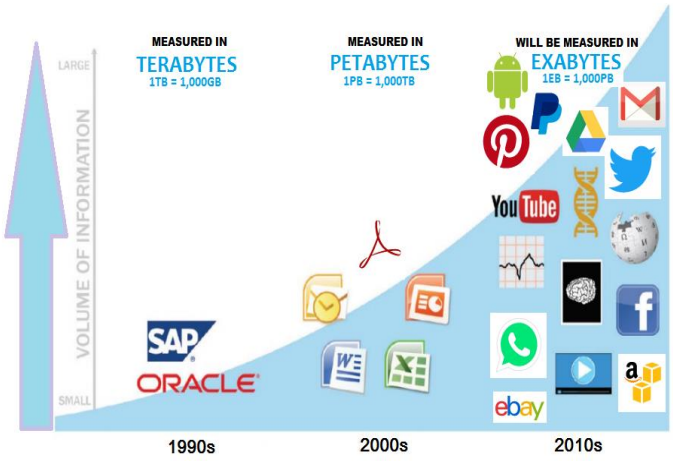

Fig -1: The rise of Big Data sources and the Evolution of data

2. DATA STRUCTURES OF BIG DATA

Big Data comes in many forms; it can be Structured, Semi-Structured, Quasi-Structured or Unstructured as in [3]. Fig -2 shows the Data structure types and the data growth accordingly.

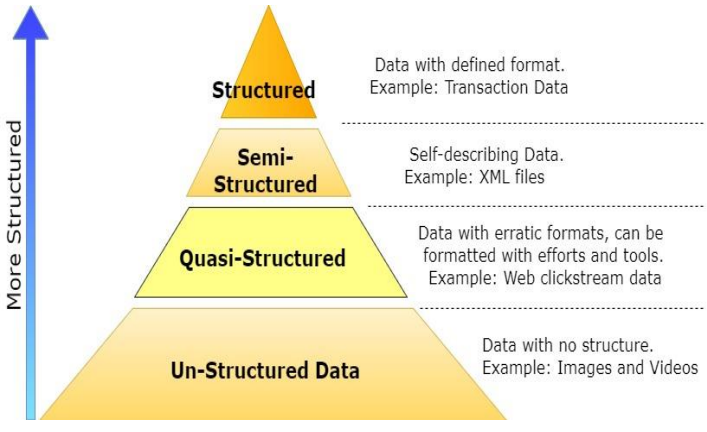

Fig -2: The pyramid of data structures of big data

2.1 Structured Data

Structured data is data that comes with clarity, description, presentation, explicate length or format. This

type of data can be easily re-organized or processed by any data mining tools. One could envision this type of data as a perfectly organized book library where books are orderly arranged, labeled and are easily accessible. Hence this type of data can be effectively used by organizations. Few examples of structured data include numbers, dates, etc., it is believed that this type of data comprises about 15-20% of data present in the world. In Fig-3, we can see one such example representing structured data. We can see data from an excel sheet that displays clearly formatted information such as First Name, Last Name and Gender of a few people.

2.2 Semi-Structured Data

This type of data is loosely defined or irregularly structured which allows a schema-less description format in which the data is less constrained than is usually in database work. Semi-structured data model allows information from several sources, with related but different properties, to be fit together in one whole, thus suitable for integration of databases and sharing information on the Web. Semi-structured data is data that may be irregular or incomplete and have a structure that may change rapidly or unpredictably. It generally has some structure, but does not conform to a fixed schema. It is Schema-less and self-describing, i.e., data carries information about its own schema (e.g., in terms of XML element tags). The main attraction for semi-structured data is that structure can be imposed upon the data as in [4]. Fig -3 shows an example for semi-structured data; here we can see the XML document that was fetched from the 'view page source' option while viewing a web page.

2.3 Quasi-Structured Data

This data is not among the commonly mentioned types of data. It is the one in between the semi-structured and unstructured data. In other words, Quasi-structured data is data which is unstructured but is in an erratic format which can be handled with special tools. For example, consider visiting the website www.google.com, the URL (Uniform Resource Locator) is tracked into log files of the webpage's server; the user's computer activity is being monitored. The URL defines a clickstream that can be parsed and mined by data scientists to discover usage patterns and uncover relationships between clicks and areas of interest on websites, as mentioned in [2]. This is illustrated in Fig -3 where the user types "HKBK College of Engineering" in the search option in Google. Upon a search, the clickstream *"https://www.google.co.in/search?q=hkbk+college+of+engineering&rlz=1C1CHBD_enIN761IN761&oq=HKBK+col&aqs=chrome.0.0j69i57j69i61l3j0.3550j0j8&sourceid=chrome&ie=UTF-8"* is generated which shall be parsed and mined.

2.4 Un-Structured Data

Un-Structured data sometimes referred to as messy or unorganized raw data is primarily in the form of text and can also be found in forms of multimedia files. This type of data does not have a formal structure, in other words the structure is unidentifiable. Word processing documents, Books, Journals, Health documents, e-mail messages, videos, presentations, webpages, photos, audio files and many other kinds of business documents are examples of unstructured data. As we know that structured data is information with a high degree of organization, unstructured data is essentially the opposite. To make use of this data, the massive data that was worthless is identified and stored in an organized fashion. Through the use of specialized software, the items can then be searched through. However, there are two aspects to be considered, firstly, the process of converting unstructured data to organized data would be costly and time consuming. Secondly, not all types of unstructured data can easily be converted into a structured model. Information aging is another factor that needs to be considered in this case where the data keeps accumulating day after day but is left unused. One example of un-structured data is illustrated in Fig -3 in which a video about Big Data from YouTube is streamed.

2.5 Heterogeneous Data

This is a combination of all 4 types of data. For example, say we have a system that stores call logs for a call-center. In this case, data such as date and time stamp could represent the structured type. On the other hand, the customer chat history could represent the quasi- or semi-structured data. So much information can be extracted from the available set of the data of different structures. Figure below shows examples of all four forms of data.

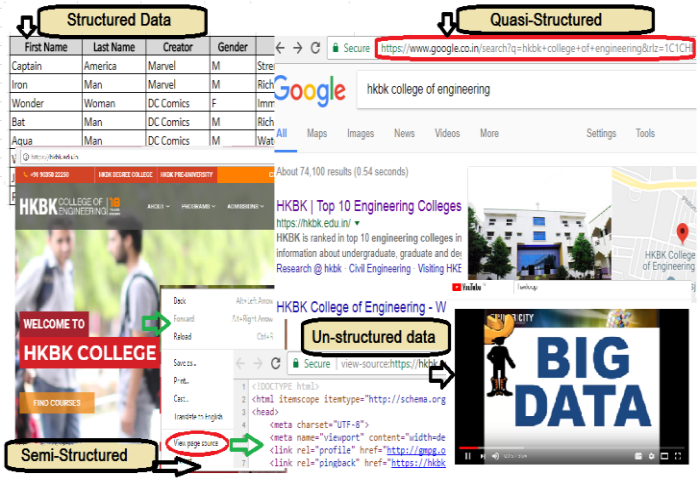

Fig -3: Illustration of data structures of big data

3. HADOOP AND HDFS ARCHITECTURE

Hadoop is an open source software framework used to process big data. It was created by Doug Cutting and Michael J. Cafarella as in [5]. Hadoop was developed for parallel and distributed processing systems and was initially motivated in

a paper by Google. As we know that managing big data is a near to impossible task for the existing traditional systems, this was made possible using Hadoop. With Hadoop, there is no limit on the storage of data and the process of computing huge data generated in high speeds is moving from traditional databases to distributed systems.

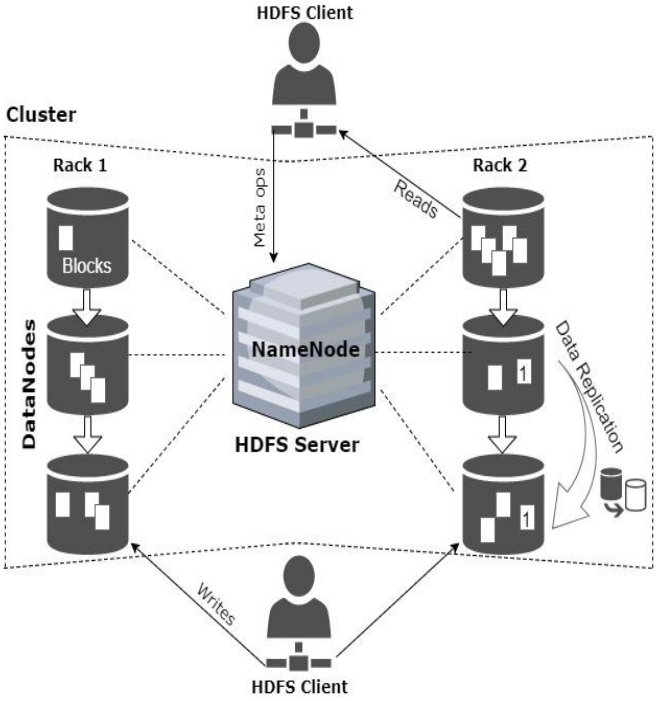

Fig -4: The HDFS Architecture and Data Replication

HDFS (Hadoop Distributed File system) is a highly fault tolerant parallel file system with a master/slave architecture whose objective includes storing and managing huge data sets, managing failures of hardware, data access and simplifying simple data coherency issues as in [6].

The HDFS architecture is as shown in Fig -4. The figure shows the HDFS master/slave architecture. Each cluster consists of exactly one NameNode which is a master server that manages accesses to files w.r.t clients and also manages file system namespace. There can be n number of DataNodes depending upon the number of nodes in the cluster. The DataNodes manage storage of files. HDFS allows user data to be stored into files usually having the files itself split into more than one blocks and in turn these blocks stored in one or more DataNodes. The responsibility of NameNode is to execute operations like Opening, Closing or renaming files. On the other hand, the responsibility of DataNodes is to serve the read and write requests from the clients. In addition, the NameNode also instructs the DataNodes to perform block creation, cloning and deletion as per the need.

The NameNode periodically receives a Heartbeat and a Blockreport from each of the DataNodes in the cluster. Receipt of a Heartbeat implies that the DataNode is functioning properly. A Blockreport contains a list of all blocks on a DataNode. The NameNode and DataNodes are designed to run on commodity hardware that runs a GNU/Linux operating system, as mentioned in [6]. HDFS is built using the Java language; therefore, any machine that supports Java can run the NameNode or the DataNode software.

3.1 Data Cloning

Why is data replication required? The answer to this question relies on the reliability and performance of HDFS. Having replicas in the DataNodes is what distinguishes HDFS from other parallel or distributed file systems. The main purpose of data replication is to better develop data availability, data reliability and fault tolerance. HDFS is designed to store files in a very reliable way using data replication in the clusters. Each file is stored as a sequence of blocks. The replication factor for each file and the block size are configurable as per each file. The above figure also shows how each file could be replicated in different DataNodes. Here we can see that the replication factor can be specified during the time of file creation. The replication factor need not be restricted since it can be changed later. All files in HDFS have strictly one writer at any point of time and can be written only once. All decisions regarding cloning of blocks are made by the NameNode.

3.2 Data Replica Placement

Instances of HDFS run on a cluster of computers that spread across a number of racks. Switches are used for communication between two nodes in different racks. The rack id to which each DataNode needs to belong to is determined by the NameNode. Usually the replicas are placed in DataNodes on unique racks. This way data loss can be prevented even if an entire rack fails since data can be fetched from another rack that holds the replica. Hence the replicas can be evenly distributed in the cluster. This policy eases balancing component failure but increases cost of writes as each write needs a transfer of replica blocks to another rack.

For instance, when a file has a replication factor of 3, HDFS's placement policy is to put one replica on one node in the local rack, another on a node in a different (remote) rack, and the last on a different node in the same remote rack. This policy cuts the inter-rack write traffic which generally improves write performance. The chance of rack failure is far less than that of node failure; this policy does not impact data reliability and availability guarantees. However, it does reduce the aggregate network bandwidth used when reading data since a block is placed in only two unique racks rather than three. With this policy, the replicas of a file do not evenly distribute across the racks. One third of replicas are on one node, two thirds of replicas are on one rack, and the other third are evenly distributed across the remaining racks. This

policy improves write performance without compromising data reliability or read performance as in [6].

3.3 Data Selection

When a read request is made, HDFS tries to satisfy it from the replica that is nearest to the reader. The replica, if exists, on the same rack as the requestor, is preferred to satisfy the request. Else, if the cluster spans multiple data centers, then the replica residing at the local data center is chosen over a remote replica.

3.4 Space Redeeming

Traditionally when we delete a file or a folder in our system on windows, the file is moved to recycle bin. This means the files space is not freed yet. File deletes in HDFS work similarly. When a file is deleted by an application or a user, it is not deleted from HDFS immediately. Upon a delete, first the file is renamed and moved to the */trash* directory. The file resides in the trash directory for a configured amount of time. After the time expires, the file is then deleted permanently from the HDFS namespace by the NameNode.

The deletion of a file causes all block spaces to be freed that are associated with the deleted file. However, the file can be restored while it still resides in the trash directory. To undelete a file, the deleted file must be restored from the trash directory. As per [6], the current default policy is to delete files permanently from trash that are more than 6 hours old.

4. CONCLUSION AND FUTURE WORK

HDFS should support use of snapshots that can be helpful in storing a copy of data at a particular instance of time. This can be particularly helpful to rollback a corrupted HDFS instance to a previously known point of time when the system worked without issues. There is a potential for making faster advancements in scientific discipline for analyzing large amounts of data .The technical challenges are most common across the large variety of application domains, therefore new cost effective and faster methods must be implemented to analyze the big data.

ACKNOWLEDGEMENT

I thank Dr. Syed Mustafa, Professor and Head, Department of Information Science and Engineering, HKBK CE for motivating me and providing their expertise that greatly assisted in writing the paper.

I am also grateful to the authors of papers and documents I have referenced. I would like to express my appreciation to the authors for sharing their pearls of wisdom.

REFERENCES

[1] Big Data Now: 2012 Edition by O'Reilly Media, Inc.

[2] Master Thesis on Tools and Methods for Big Data Analysis, by Miroslav Vozábal- 2016, University of West Bohemia.

[3] Research Paper on Big Data and Hadoop, Iqbaldeep Kaur, Navneet Kaur, Amandeep Ummat, Jaspreet Kaur, Navjot Kaur. IJCST Vol. 7, ISSue 4, Oct - Dec 2016.

[4] Adding Structure to Unstructured Data, Database Research Group (CIS), University of Pennsylvania ScholarlyCommons. 1997.

[5] HADOOP ARCHITECTURE AND FAULT TOLERANCE BASED HADOOP CLUSTERS IN GEOGRAPHICALLY DISTRIBUTED DATA CENTER, T. Cowsalya and S.R. Mugunthan, ARPN Journal of Engineering and Applied Sciences.

[6] The HDFS Architecture guide, https://hadoop.apache.org/.

[7] Big Data And Hadoop: A Review Pape, Rahul Beakta, https://www.researchgate.net/publication/

BIOGRAPHIES

Prof. Aseema Sultana was born in Bangalore, India. She received the B.E degree in ISE from HKBK CE, in 2010, and M.Tech in CSE from MVJ CE, in 2012. She worked as a Sr. Project Engineer in Wipro (2012-2016) and currently working as an Asst. Professor in HKBK CE.