Volume 4 No. 2 & Vol.5 No. 1 (July 2020 to June 2021)

ADOPTING ANALYTICS WITH BIG DATA

Ritwik Sinha
Technical Consultant Intern at Merkle Sokrati
Pune, India
E-mail : ritwiksinha53@gmail.com

ABSTRACT

The phase of extracting meaningful insights from the data, has become the top priority of any organisation. Big Data certainly helps the emphasising periods of today's era to enhance the flexibility of mining, storing and processing of data.

Thisjournal describes the punctual conduct of Big Data to successfully fulfil one's needs.

Keywords– *Algorithm, Analytics, Big Table, Cloud, Data scientists, File Systems, Hadoop, MapReduce, Structured and Unstructured data, Hadoop ecosystem, Hadoop agents*

1. INTRODUCTION

Data scientists from very first decade of 21st century, are confronting the issue of storing data. As the time passes by, the amount of data being generated has been increased in a big number. Processing of data was not kind of a big issue, in the early trends, albeit, due to high complexity, the demand for an open source big data platform increases and there comes the introduction of Hadoop in 2005. Although, it was initially developed to support distribution for the Nutch search engine project, a project of Apache Software Foundation, the Hadoop framework includes various tools such as Apache Flume, Apache Sqoop, MapReduce, Pig, Hive, Apache HBase etc. that helped a lot to overcome the issue of storage, sharing, administration, analysis and data processing. Big Data is helping the global market to realise the everlasting trends and opinions of the distinctive view of the raw data collected either in the form of text, images, videos, or audio.

Big Data find its applications in the domains like Finance, Healthcare, Telecom, Retail, E-commerce etc. Hadoop comprises of cloud technology too for storing huge data that makes storage cost pretty lower. Hadoop Distributed File System, reduces latency as well as provide fault tolerance by replicating blocks having data. This feature of HDFS is known to be as

Rack Awareness, generally used for symbolising high availability of storage.

Data is stored in blocks inside HDFS. Due to fear of losing blocks, each block has multiple copies in different racks (racks referred to servers present at different geographical locations) on the basis of replication factor. Default replication factor of HDFS is 3. That means, if certain amount of data stored inside 'Block A' at 'Rack 1', then at two other racks (say, Rack 2 and Rack 3) it will be copied, totalling to 3.

User setting up the Hadoop ecosystem can change the replication factor according to the needs. The replicas are made to ensure readily availability of blocks known as Fault Tolerance, at the time of crashing down of rack (server) having blocks that the user needs to extract and analyse data.

2. BIG DATA HADOOP

Advancement in technology leading to increasing rate of data being generated, resulted into introduction of Big Data having three major V's, later added with two more V's. V's are the characteristics of Big Data known to be as Volume, Variety, Velocity, Value and Veracity. While there are other characteristics too of Big Data, but let's discuss only these characteristics here.

1. Volume:- Itrefers to the size of data generated by different sectors inclusive to IT industry, energy industry, healthcare, national security and much more to analyse and process. It is unrealistic to define the benchmark of the size of data to be defined as big data, however, it is an assumption that data size residing inside Exabyte (EB) to Zettabyte (ZB) is considered to be Big Data.

2. Variety: - It refers to the type of data being generated to analyse by big data. Apart from old age structured data, we are facing issues to analyse unstructured data as well that we can't store in a spreadsheet or a database. These data forms can be photographs, emoticons, encrypted packets, tweet data, sensor generated data, ticker data, emails, invoice details, surveillance data, audio data, video data, weather data and uncountable more.

3. Velocity: - The pace at which data has been generating nowadays, leads to boom of big data, as velocity measure how fast is the data inflow. The social media, in especially, generates humongous amount of data every second, that needs to file it, process it and retrieve it at any time. Vast quota of data is flowing through internet across the globe and in order to reduce the risk of cyberattack, this data has tends to be forwarded in encrypted form, thus it becomes more difficult to investigate the hidden specimen, analyse the behaviour of data as there can be false data present that can harm the user's device.

4. Value: - Data analysis is done to come to some conclusion and

make a decision whether financially or something else. For this, we need to figure out the context of data driven to us, find out the usefulness of the data to reach an appropriate decision and retrieve useful insights and that's what the value characteristic of big data determines. Giant firms like Google, Amazon and Facebook have gripped the value of Big Data through analytics.

5. Veracity: - It refers to the quality of data. Due to predominance of unstructured data, the quality of data has been degraded a lot as compared to old age structured data form. It is inconsistent, includes noise, and is ambiguous or incomplete. Thus, data veracity categorize them into good, bad and undefined. Increase in data sources and variety of data, big data analytics is facing more difficulty to establish trust and maintain quality of data. Bad quality data can harm the economy of a country too.

In order to overcome the storage issue, Hadoop help us to keep on adding new nodes (similar to adding another computer), to store data as the data capacity gets increasing readily in a heterogeneous cluster. For example, if a superstore CEO, observes the busiest timings of his store and want to keep crowd crawling faster, he can increase the number of counters by increasing the number of cashiers.

Hadoop has a wonderful property of resiliency. Being resilient means, Hadoop has integrated an automatic failover management system, which replaces the subsystem in case a node fails during processing. Although, this property is not so handful of joy, when dealing with small set of data but since Hadoop is always preferred to big data, this property takes Hadoop to gain its importance deeply in an organisation. To combat the failure, Hadoop automatically realizes there is a failure, and sends another node to the place of failed node. Apart from doing this, Apache also distributes copies of stored data across all data even before any failure occurred, to prevent data loss due to system interruption. We are very close to a big data revolution and Hadoop can play the key role, as Apache has already been started partnering with Zettaset company to provide high level security. Security was a concern with Apache Hadoop, because the built in security of previous Hadoop implementation and available options are inconsistent among new release versions, albeit, it is progressively improving. Encryption of data is at foremost priority. And Hadoop never stores or have access to unencrypted data. This property satisfies at-rest encryption, which means encryption of data on persistent disk. Despite this, Hadoop requires four different approaches for its security:

(i) Authentication: - To ensure trust and protection, the first stage authenticates the user by authenticating UserId, password to the person seeking to perform an operation and hence prove their identities on claim.

(ii) Authorization: - Different users are assigned to operate different tasks, in order to control what a particular user can do to a specific file, the second stage of Hadoop security comes in i.e., authorization.

(iii) Auditing: - It tracks the authenticated and authorized user after the time they logged into the Hadoop cluster. It keeps records of what analysis has the user performed, what data was accessed, added, changed etc.

(iv) Data Protection: - It includes different techniques of data encryption and data masking to prevent data from unauthorized users.

There are two major open source projects of Apache supporting Hadoop security, Knox and Ranger. Knox is basically a Rest API base perimeter security gateway responsible for auditing, authorization management, support monitoring on Hadoop clusters. On the other hand Ranger provides or denies access to different resources of Hadoop like HDFS files, Hive tables, to the assigned users. Ranger works differently for different Hadoop components like YARN, Hive, HBase etc.

To sum up, Hadoop security can be categorised into three different levels: Firstly, Kerberos, which is an authentication protocol that is treated as a benchmark to implement authentication in the Hadoop cluster and it uses secret-key cryptography for providing authentication. Secondly, Transparent Encryption in HDFS, for fair and legal encryption, HDFS implements transparent encryption of data that is to be read or write from HDFS directories. This kind of encryption prevents cyber-attacks on the level of Operating System. Lastly, HDFS files and directory permission, every data file present in HDFS have different permission for owner, group members and other users. Additionally, the client also has to follow two step identity, one a user name and second the group list, whenever the client needs to access the data files present in HDFS directories. This completes the check of files and directory permission once the authentication of the user gets done.

3. BIG DATA IN BUSINESS

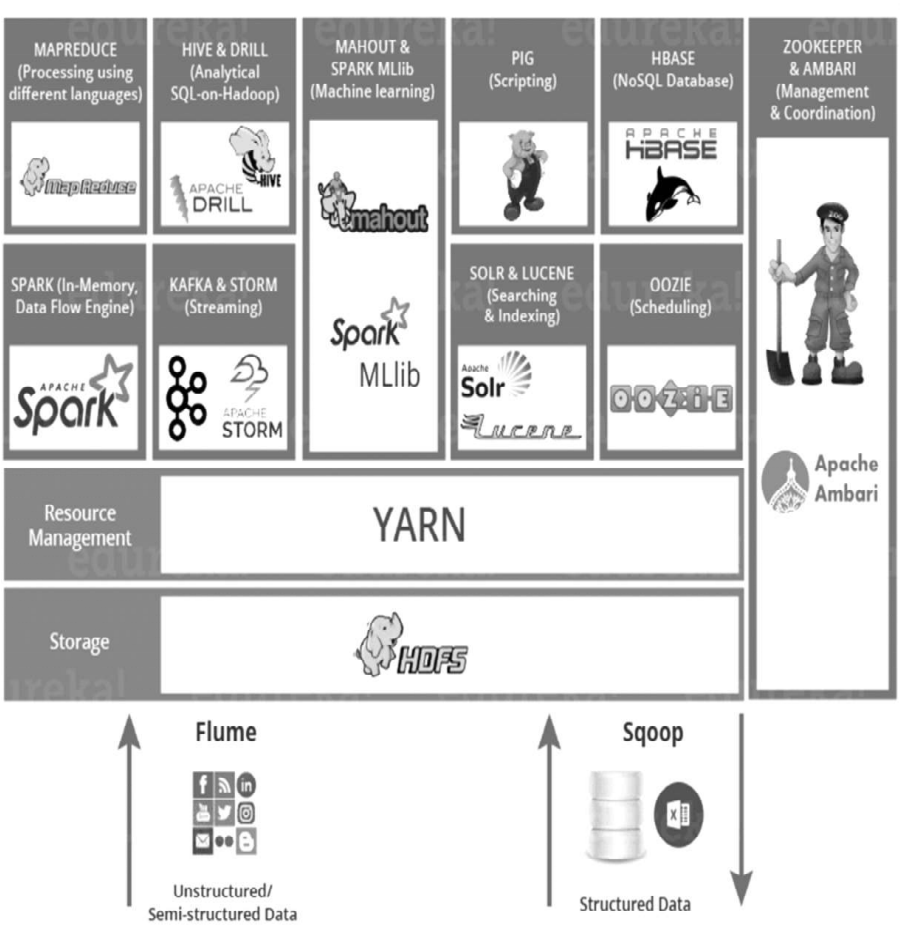
Buyer and seller are the master players of running a successful business. If there is a great relationship between these two, the business has no reason to face any kind of stoppage and well vice versa will be the scenario that business giants or fresh entrepreneurs will never want to happen. Retention of customers is the fore priority of them, and here comes the role of data analytics, digging in, big data analytics. Business works to optimize the customer churn percentage. To make all things happen perfectly to them, big data helps a lot in 21st century, as the data that is being generated is enormous and need to be analysed to decrease the customer churn. Predictive analysis using machine learning models is the first step towards the process of understanding customers. These models helps to predict which type of customers are more likely to churn from the organisation. Not only this, machine learning with deep learning can enhance the way of handling those customers too. Business can understand the existing and prospective customers' behaviour to understand why they are not showing the tendency to churn. Due to the understanding of their behaviour, business can offer promotional events and offers to the customers having a behaviour to churn. This can enhance their experience with the services provided by the business. Big data analytics not only serves customer demands, orders and complaints but also it can build loyalty within customers. Analytics finds out why customers are interacting, and concluding their implications, this helps to minimise the gap between customer and business, moreover, it will engage customers through more personalized campaigns for improved communication. The source for analytics in big data is datalake. This will help business to formulate outstanding strategies devised on the back of big data. Certainly, with

big data, business organisations can figure out valuable customers also, that will help them to enhance their products and services. What makes big data such a game changer in business? The answer is quite straightforward, this advancement in technology leads to collect and maintain real time data and it is very crucial to have real time data in order to estimate customer metrics, inclusive of customer loyalty, that was handled retrospectively earlier, and is now handled more extensively, leading to make predictive analysis better with higher percentage of accuracy in it. Also, since data is coming in huge amount, this frequency allow businesses to test models in real time.

Healthcare is the most important sector of a healthy life, and is need to be protected effectively. With inclusion of big data in healthcare industry, the evolution had occurred. Earlier there was time when testing of new pharmaceuticals outcomes consumes lots of time. But big data certainly focuses on discovering outcomes of new pharmaceuticals. It leads to better analyse the trial phase of the medicines and help to proclaim the efficiency of medicines better and faster. This reduces the time of different phases of medicines before coming to the market for public use. Patients already admitted in the hospitals are getting better treatment with the help of big data. As patient history has been managed by the hospitals as records and these records help data analysts to analyse the diseases which are having more deadly impact on patients, can analyse patients personally such as what kind of disease is more likely to happen in near future to the particular patient, as a result of which the patient must get treated accordingly. Since, lots of data are garbage too in case of healthcare, as the record of patient died 50 years ago can't predict today's trends. This old age data can harm the prediction process too. It is the sole responsibility of analysts to figure out best data to analyse and work upon because the margin of error in healthcare is very less. Big data often helps to figure out this situation too. As big data uses theory of pointers, that specifically helps analysts to point out only necessary data and finally engage those nodes only which contains this data. Problem of multiple physical nodes is also resolved with the help of big data.

4. HADOOP ECOSYSTEM

Hadoop has a hierarchical ecosystem to fulfil the needs of Big Data, hierarchy starts from ingestion of data using tools like Apache Flume, and Apache Sqoop, second stage of hierarchy is to store data using Hadoop Distributed File System (HDFS), and Apache HBase. Next step is to process the data that has come using MapReduce. To analyse data, we take the help of Apache Pig, and Apache Hive. Lastly, Apache Oozie take the control to manage the workflow of jobs assigned to Hadoop.



Before going into depth of each tool involved in Hadoop ecosystem, let's discuss Hadoop configuration, structure and daemons. Hadoop commands are similar to Linux commands just there is an addition of reference variable before that particular command. Hadoop works in three nodes:

1. Standalone (or local) Node: - In standalone node, Hadoop runs with default configuration (Empty configuration files i.e., no configuration settings in core-site.xml, hdfs-site.xml, mapred-site.xml and yarn-site.xml). If properties are not defined in the configuration files, Hadoop runs with default values for the corresponding properties.
2. Pseudo Distributed Node: - In this type of node, Hadoop daemons run on the local machine. We have knowledge of NameNode, DataNode, ResourceManager, and NodeManager.
3. Fully Distributed Node: - In this type of node, Hadoop daemons runs on a cluster of machines.

We have knowledge of Master Node and Slave Nodes.

Hadoop 2.x cluster architecture make client to follow a certain hierarchy. Client either consult HDFS or YARN initially. HDFS contains NameNode while YARN contains ResourceManager. NameNode consists of Data Nodes that are managed by NodeManagers. On the other hand ResourceManager consists of NodeManagers that looks for various DataNodes. After successfully installing Hadoop on the system we should verify whether it is running or not. Run a command "sudojps" on the terminal and look for the output. Output shows different processes which are active. These process includes DataNode, Jps. NodeManager, JobHistoryServer, NameNode, and ResourceManager. If all processes are active except NameNode, then we can process data but cannot do anything with data storage, because this is the master of DataNode or HDFS. In a simple Hadoop setup there are two master nodes and multiple slave nodes. Master Nodes are configured for reliability of the client and Slave Nodes are managed automatically by the Hadoop cluster, all data is stored across different host (usually 3). Slave nodes are built to increase speed of cluster and decrease the cost of the cluster. The two Master Node includes, Master A, which contains NameNode service and Hive Master, while Master B contains JobTracker or ResourceManager, and HBaseRegionServer.

Hadoop 2.x configuration files can be classified in four categories, Core contain "core-site.xml" file, HDFS contain "hdfs-site.xml" file, YARN contain "yarn-site.xml" file and MapReduce contain "mapred-site.xml" file. Let's discuss Hadoop daemons or processes one by one:

1. NameNode: - It runs on master node of the HDFS. It directs DataNodes to perform their low level Input / Output tasks.
2. DataNode: - It runs on each slave machine inside the HDFS. It does the low level Input / Output tasks.
3. ResoureManager: - It runs on master node of the data processing system (MapReduce).
4. NodeManager: - It runs on each slave node of data processing system and also gives platform for the data processing tasks.
5. JobHistoryServer: - It also runs on each slave node of data processing system.

Till now, how Hadoop structure or architecture looks like has been completed and it's time to dive deeper in Hadoop ecosystem. Beginning with data ingestion tool Apache Flume.

5. APACHE FLUME

To inject data into the database, we uses data injection tools. Since data can be structured and semi or unstructured, so we need different tools to inject data. For structured kind of data we use

Apache Sqoop and for semi or unstructured data we use Apache Flume. In Flume, daemons are called as Agent. By default there are three agents in Apache Flume that needs to be defined in Hadoop ecosystem in order to make Flume work. First agent is Source, from where we get data, i.e., we need to define our web servers. Second agent is Sink, it contains database i.e., HDFS. Third agent is Channel which contains Memory Channel (or MemChannel). Data collected from web servers are text, image, audio or hyperlink, which we can read and write. Since there is no need of any code in Apache Flume so execution permission is not needed in Flume. Data Flow Model for Flume initiates from Web server goes to Agents (Channel – Sink) and finally into the HDFS. Flume allows user to do the following:-

(i) Stream data into Hadoop from multiple sources.

(ii) Collect high volume web logs in real time.

(iii) It acts as a buffer when the rate of incoming data exceeds the rate at which the data can be written. Thereby preventing data loss.

(iv) Guarantees data delivery.

(v) Scales horizontally to handle additional data volume.

Scaling horizontally means connecting commodity systems in parallel, suppose hard disk space is 500 GB and data to store is 1 TB in size, then in horizontal scaling, we add another hard disk of 500 GB in the existing system. On the other hand in vertical scaling, we are not adding any peripheral, we remove the earlier 500 GB hard disk and will add 1 TB of new hard disk in the system. Flume works with stream/ online/live data. As for offline data we have Apache Sqoop and Apache Spark. Essential components involved in getting data from a live streaming source are:-

1. Event: - It is a singular unit of data that is transported by Flume (typically a single log entry).

2. Source: - It is the entity through which data enters into the Flume. Sources either actively samples the data or passively waits for data to be delivered to them.

3. Sink: - It is the unit that delivers the data to the destination. A variety of sinks allow data to be streamed to a range of destinations. For example, HDFS sink writes events to the HDFS.

4. Channel: - It is the connection between the source and the sink. The Source ingests Event into the Channel and the Sink drains the Channel.

5. Agent: - It is any physical java virtual machine running inside Flume. It is a collection of Sources, Sinks and Channels.

6. Client: - It produces and transmits the Event to the Source operating within the agent.

6. APACHE SQOOP

Apache Sqoop is a data injection tool between Hadoop and Relational Database Management System. We can use Sqoop to import (export as well) data from RDBMS such as My SQL or Oracle into HDFS, transform the data in HadoopMapReduce and then export the data back into RDBMS. Sqoop has a command line interface application and to check whether sqoop has successfully installed or not write command "sqoop import" in the terminal and start writing sqoop commands thereafter. There is a special command in sqoop "sqoopcodegen", we use this command when we want to work in sqoop, and would like it to encapsulate in jar form and move forward, so we use this convert our work into jar package. Codegen will generate code to interact with database records. Sqoop needs data, which we get from RDBMS. To check whether My SQL is working or not write command "sudo service mysql restart", after running this command in terminal mysql gets activated and from now we can use My SQL commands to do operations according to the needs. If somehow, mysql is not starting, write command "sudo service mysql start" then "mysql –u root –p" then "sudo service mysql restart". This process will ensure My SQL to work.

Java works behind every sqoop / Hadoop code. A sqoop developer, must import databases or tables from My SQL to Hadoop and there might be the chance that he/she does not know how to code in java, so codegen will help them to convert the import task into java code of jar file because if java developer wants to implement the import task so they need jar file. The code created by codegen contains java classes that encapsulates the import task in jar file. The basic syntax of codegen used by sqoop developer is:

[sqoopcodegen —connect "jdbc:mysql://localhost/<host_name> —username <user_name> —table <table_name>]

Export tool: - The export tool exports a set of files from HDFS back to RDBMS. The target table must already exist in the database. The input files are read and passed into a set of records according to the user specified delimiter. Let's take an example, suppose we have a database named "test" where data will be exported, "employee" is the table name, currently data is present in file named "row12" inside "Latest_Employees" directory and the records entered in the HDFS is terminated by 1. So, the code to export will look like:

[sqoop export —connect "jdbc:mysql://localhost/test" —table employee —export —dir "/Latest_Employees/row12" —fields-terminated-by "1";]

Sqoop is very hand full of joy for non-programmers, we can force sqoop to choose wisely which columns to show before doing inputs inside the table, this will save time as we are obliged to do an entire input and then looking for our

data. The main task doer behind the doors in sqoop is MapReduce, MapReduce job gets created behind the reach of sqoop and do importing from the external sources to HDFS.

Sqoop is preferred very often because in the field of analytics, analysts requires to load bulk amount of data from diverse sets of resources into Hadoop clusters, there can be other options too like scripting, but this approach to load huge volume of data is inefficient and time consuming. Also, giving direct access of external systems, without loading into Hadoop, to MapReduce applications not just take efficiency down but also complicates the applications associated to it. Hence, due to all these complications Apache sqoop (Apache Flume too) are built to overcome these challenges. But there is a difference between working of sqoop and flume. Sqoop uses different connectors into its architecture to connect to respective data sources (like My SQL, Oracel, PostgreSQL, SQL Server etc.) and do the data importing task. While on the other hand, Flume uses different agents as discussed earlier to fetch the data and completes the process of data ingestion.

7. HADOOP DISTRIBUTED FILE SYSTEM

The local system that an individual owns has its file system that we usually say local file system to store data into the local storage, similarly Hadoop uses HDFS to store its data as a storing file system. Due to presence of multiple nodes HDFS duty is to distribute data accordingly to every node making HDFS a distributed architecture. In Hadoop, HDFS is treated as the ultimate destination for storing data. HDFS does not take into consideration about the data source and data means, as it is least affected by these two. Likewise Linux, HDFS also has its own built in shell commands making, it a shell programming architecture, to store data into it. Another main feature of HDFS is that, it does not import live streaming data, it usually works for data already stored inside it, or simply offline data. Inside HDFS, a file is broken up into either 64 MB or 128 MB chunks and these chunks are stored on data nodes for further operation.

There are some standards specified by IEEE, one of them is POSIX semantics, which is applied to an environment of a language, for instance a signal being directed from a process to kill another process. Hadoop distributed file system provides a subset of these POSIX semantics. Due to this, HDFS supports read access to any random user and write access to non-random users. HDFS is safe as of its high security standards, one of being blocking the append access and syncing option to the user once the file has been closed successfully. There are various operations that HDFS tends to perform and they are:

(i) Executing different file system operations like opening, closing,

renaming etc. of different files and directories.

(ii) Presence of NameNode Managers to manage the DataNodes.

(iii) As a precaution to the case of data loss, HDFS ensures mapping of data blocks into different DataNodes present at different racks.

(iv) Replication factor of each block is maintained by NameNode.

(v) Suppose, a DataNode fails, the NameNode chooses automatically a new DataNode as new replicas.

The architecture of HDFS follows master-slave behaviour. Each cluster present in the HDFS comprises of one master node and multiple slave nodes and these slave nodes are responsible to store data blocks differently depending upon the replication factor setup by the Hadoop developer at the time of Hadoop installation. Master node always remain like an unsung hero, managing the metadata of data blocks. Metadata includes information of data like block locations, permissions granted to the blocks, block size, replication factor and much more.

Clients to do read and write operations in HDFS tends to follow a systematic procedure. At the time of writing a file to HDFS, client communicate to NameNode for collecting metadata. Once NameNode responds with a number of blocks location, replication factor, block size etc. to the client, the client can now directly communicate to the DataNode, to succeed the write operation. When DataNode receives the blocks from the client, it sends the write confirmation to the NameNode. On the other hand, when client needs to perform only read operation, it first communicates with the NameNode for metadata, and thereafter interacts with DataNode. The client gets the privilege to read data parallel from DataNode according to the metadata received. The data flow directly from DataNode to the client. Finally, client combines the received data blocks and converts it into the form of an original readable file.

8. APACHE HBASE

HBase is an open source, multidimensional, distributed, and scalable & a NoSQL database written in JAVA. HBase runs on top of HDFS and provides BigTable like capabilities to Hadoop.HBase is implementation of 'Google BigTable' database in open source environment. It is designed to provide a fault tolerant way of storing large collection of sparse data sets. Since, HBaseachieves high throughput and how latency by providing faster Read/Write access on huge data sets. Therefore, HBase is the choice for the applications which require fast and random access to large amount of data. HBase can store data in real time and performs analyses. Hadoop is always used for back processing, which means doing analyses on data which has some

schema and no data is inserted into them in real time. So, we need a database which is schema less and feed changes in data in real time, hence Apache HBase plays the key role. As SQL has to satisfy ACID algorithm, NoSQL has to satisfy CAP theorem. HBase follows Consistency and Partition Tolerance i.e., CP of CAP theorem. Now, CAP theorem determines that, it is impossible for a distributed computer system to simultaneously provide all three of the following guarantees:

(i) Consistency: - It refers to the situation when all nodes see the same data at the same time.

(ii) Availability: - It guarantees that every request receives a response about whether it was successful or failed.

(iii) Partition tolerance: - The system continues to operate despite arbitrary message loss or failure of part of the system.

Now, since all the three parameters can't be satisfies in a single software, so let's discuss different software on the basis of pairing these parameters. CP parameters are guaranteed in BigTable, Hypertable, HBase, MongoDB, Tenastore, Scalaris, BerkeleyDB, MemcacheDB and Redis. CA parameters are guaranteed in RDBMS (My SQL, PostgreSQL etc.), Aster Data, Greenplum and Vertica. AP parameters are guaranteed in Dynamo, Voldemort, TokyoCabinet, KAI, Cassandra, Simple DB, Couch DB and Riak.

Importance of HBase in the market was shown by Facebook. Facebook faces multiple challenges including Storage and Performance consistency, fast processing, Storing Large amount of growing data, and serving lots of requests. These challenges comes across to Facebook while using Cassandra and due to which it shifted to HBase.

To insert data in any HBase table, we need to specify three values: <RowKey, Column Family, Timestamp>. Usually in RDBMS we have row database structure that is extraction of data can be possible by making row as finding index, but HBase is a columnar database, so we need to understand why there is a need of columnar databse. In columnar database (i.e., column wise storing of table) we can compress data in order to minimise the storage usage and increase the storing capacity. Columnar database gives faster data access because only selected columns have to be read in columnar database, most of the columns contain only few distinct values hence better compression is possible. By default, table is vertically partitioned, that means operations on different columns can be processed in parallel by assigning a processor core. Now, let's understand some HBase Data Model terminologies:

1. Table: - An HBase table consists of multiple rows.

2. Row: - A row in HBase consists of a row key and one or more column with values associated

with them. Rows are sorted alphabetically by the row key.

3. Column: - A column in HBase consists of a column family and a column qualifier, which are delimited by a : (colon) character.

4. Column Family: - Column families physically collocate a set of columns and their values, often for performance reasons. Each column family has a set of storage properties, such as whether its values should be cached in memory, how its data is compressed or its keys are encoded.

5. Column qualifier: - A column qualifier is added to a column family to provide the index for a given piece of data. Though column families are fixed at table creation, column qualifiers are mutable and may differ greatly between rows.

6. Cell: - A cell is a combination of row, column family and column qualifier and contains a value and a timestamp value when we put data into cell.

HBase architecture has three major components i.e., HMaster Server, HBase Region Server, Regions and Zookeeper. HMaster, RegionServer and Zookeeper are placed to coordinate and manage Regions and perform various operations inside the Regions. A region contains all the rows between the start key and the end key assigned to that region. HBase tables can be divided into a number of regions in such a way that all the columns of a column family is stored in one region. Each region contains the rows in a sorted order. A region has a default size of 256MB which can configured according to the need. A group of regions is served to the client by a RegionServer. A RegionServer can serve approximately 100 regions to the client.

9. MAP-REDUCE

Hadoop has two steps: first is to transfer data from Linux file system to Hadoop distributed file system, and this process is known to be as data migrating, hence it comes under Map. Second step of Hadoop is to process the data like coding e.g. 3+2=5, here data gets changed or simply data manipulation. This comes under reduce task. Map refers to extract something we care about each record shuffle and sort. While, reduce refers to aggregate, summarize, filter or transform to write the results. The output of MapReduce job will be stored on HDFS by making different parts of file. Now, let's see the data flow of MapReduce:

(i) Mappers read from HDFS.

(ii) Map output is partitioned by key and sent to Reducers.

(iii) Reducers sort input by key.

(iv) Reduce output is written to HDFS.

Suppose data is present in abc.txt file, which comes to process by MapReduce, this file must have a structure of [key : value] = [Byte_Offset : Entire Line]. In different programming languages like Java, C, C++, Python [key : value] has datatype [int : varchar], as they are primitive classes (wrapper classes in Java) in these languages. But, in Hadoop, these classes are known as Box Class, where int is referred to as IntWritable, long is referred to as LongWritable and chr is referred to as Text. Hadoop framework has a default function, Record Reader that need every line and give Key Value for every line. Let's say a.txt file has size 200MB. So, in Hadoop 1, we have four splits 64+64+64+8 MB. This split is known as input splits. Number of Map functions to be used from mapper class is equal to number of input splits. The data of a.txt will be the input, to get Map out we need to tokenize these sentences and in key:value format so that Hadoop can understand. Before reducer gives output, there will be RecordWriter which will show the coming key:value in a text form and will then go back and store in HDFS. The shuffle and sort phases occur simultaneously i.e., while outputs are being fetched, they are merged.

10. APACHE PIG

Apache Pig is an open source tool, used to analyse unstructured data like textual data. For Pig. Data is collected through Flume. Pig is a Data Warehouse Tool and in this tool to work upon, the language which we use is Pig Latin. Pig is high level language and procedural language/scripting language just like SQL. To open Pig in Hadoop, run the following command in the Hadoop terminal:

[pig -x mapreduce] or simply [pig]

To open Pig in local file system mode run the following command:

[pig -x local]

Pig is not a standalone tool, it works default on Hadoop. The environment in which Pig work is Pig Shell. It is an interactive shell where we can type commands for Pig. Pig is desirable to have a high declarative language similar to SQL query, where the user specifies the 'what' and leaves the 'how' to the underlying processing engine. In Pig, Java is not required. Although, Pig favours unstructured data for analysis but we can also take any data like structured and semi-structured as well. Pig is extensible by UDF (User Defined Functions). Pig provides common data operations like filters, joins, ordering etc. and nested datatypes like tuples, bags, and maps. Collection of tuples is known as bag in Pig. Pig is an ad-hoc way of creating and executing map-reduce jobs on very large data sets. Pig is a data flow language that is present at the top of Hadoop and makes it possible to create complex jobs to process large volume of data quickly and efficiently.

There are some drawbacks of Pig also, thus we should restrict ourselves using Pig when we have really nasty data

formats, when you would like more power to optimize the code.

LOAD function is used to read data in Pig from the file system. A series of 'transformation' statements are used to process the data. A DUMP statement is used to view results and a STORE statement to save the results. LOAD is not a diagnostic operator in Pig while DUMP, DESCRIBE, ILLUSTRATE are diagnostic operators. Pig State, Pstatistics or Pig Statistics is a framework for collecting and storing sccipt-level statistics for Pig Latin. UDF's can be applied only in FOREACH statements in Pig. Logical and physical plans are created during the execution of a pig script. Big IT giants like Yahoo, Google, and Microsoft are collecting enormous data sets in the form of click streams, search logs and web crawls. Some form of ad-hoc processing and analysis of all of this information is required. Pig basic program structure consists of:

1. Script: - Pig can run a script file that contains Pig commands. E.g., 'pig script.pig' runs the commands in the local file sript.pig.
2. Grunt: - Grunt is an interactive shell for running Pig commands. It is also possible to run Pig scripts from within Grunt using run and exec (execute).
3. Embedded: - Embedded can run Pig programs from Java, much like we can use JDBC to run SQL programs from Java.

Different Pig Latin-file loaders are (out of these PigStorage is preferred, because here we have to define delimiter otherwise CSV and XML loader are made only for CSV and XML files respectively):

(i) BinStorage: - "binary" storage

(ii) PigStorage: - Loads and stores data that is delimited by something

(iii) TextLoader: - Loads data line by line (delimited by the newline character)

(iv) CSV Loader: - Loads CSV files

(v) XML Loader: - Loads XML files

There is specific implementing class for each Pig data type, they are:

(i) Bag: - org.apache.pig.data.DataBag

(ii) Tuple: - org.apache.pig.data.Tuple

(iii) Map: - java.util.Map<Object, Object>

(iv) Integer: - java.lang.Integer

(v) Long: - java.lang.Long

(vi) Float: - java.lang.Float

(vii) Double: - java.lang.Double

(viii) Chararray: - java.lang.String

(ix) Bytearray: - byte []

Out of these data types, Bag and Tuple are native data types and others are universal data types that everyone uses in context of Pig.In order to fulfil our needs accordingly that Pig faces a bit of challenge, there is another tool to

analyse data and that is Apache Hive, let's discuss it now.

11. APACHE HIVE

Hive is a tool used in data warehousing concepts including storing, managing of data. And the query language which we use to perform these tasks is Hive Query Language or HiveQL. Hive was developed by Facebook. Hive is designed in such a way that it allows easy data summarization, ad-hoc querying and analysis of big data to process structured data in Hadoop cluster. Hive is a SQL-like scripting language built on MapReduce that is used for analytics. Data per query speed is PB/sec, hence faster execution while performing analytics on huge data sets compared to SQL. In Hive, normalization is not required. In Hive, similar to Pig, have Hive shell to do analysis.

In Hive, those tables which are in HDFS but outside metastore are called external table else internal table or manage table. If we delete a managed table, both table data and metadata for that table will be deleted from HDFS. We prefer creating external table in Hive.Metastore stores information about the tables, in a central repository. It is highly essential for interpreting flat files stored in HDFS as tables. Performing joins between tables can be really tough if there are no clear schemas or information about the relations between internal tables, the metastore solved this problem. There are four Hive data models, let's discuss them:

1. Databases: - It contains Namespaces.
2. Partitions: - It means dividing a table into a coarse grained parts based on the value of a partition column such as a date. This make it faster to do queries on slices of the data. Partitions contain partition keys that determines hoe data is stored. Each unique value of the partition keys defines a partition of the table. Partitions are named after dates for convenience.
3. Tables: - It includes schemas in Namespaces.
4. Buckets or clusters: - Buckets gives extra structure to the data that may be used for more efficient queries. A join of two tables that are bucketed on the same columns including the join column can be implemented as a Map Side Join.

12. APACHE OOZIE

In context of Java, Oozie is a web user interface application and its work is to coordinate multiple jobs in any process environment. Oozie is a workflow scheduler system to manage Apache Hadoop jobs. Oozie workflow jobs are Directed Acyclical Graphs (DAG's) of actions. Oozie coordinator jobs are recurrent Oozie Workflow jobs triggered by time (frequency) and data availability. Oozie is integrated with the rest of the Hadoop stack supporting several types of Hadoop jobs out of the box (Java, Map-reduce, Pig, Hive, Sqoop and

Distep) as well as system specific jobs (Java programs and shell scripts). Oozie is a scalable, reliable and extensible system. Like Flume have three agents, Oozie also have three agents namely Oozie Workflow, oozie Coordinator, and Oozie Bundle.

14. CONCLUSION

Data is the new gold, concluded correctly as without data the computational world is as irrelevant as programming language without data structures and algorithms. It is due to the favour of Big Data, that handling this gold has become really effective and simple. In order to analyse the raw data and turn it into some meaningful insights the role of Big Data Hadoop is at the topmost priority. Hadoop ecosystem is an environment that captures big data and clarify our needs of data storage and data processing. Data security is also configured through Big Data as it provides flow of encrypted data so that there is no data loss and reduces the probability of data spoliation.

REFERENCES

[1] Tom White, book on Hadoop: The Definitive Guide, pp. 786, May 2009.

[2] Reihaneh H. Hariri, Erik M. Fredericks & Kate M. Bowers, Journal of Big Data, Springer Open, June 2019.

[3] Anand, blog on Big Data Tutorial, bigishare.wordpress.com, March 2020.

[4] Simplilearn, blog on How Big Data Can Help You Do Wonders In Your Business, simplilearn.com, March 2021.

[5] Sandra Durcevic, blog on Big Data Analytics in Healthcare, datapine.com, October 2020.

[6] Shubham Sinha, blog on Hadoop Ecosystem: Hadoop Tools for Crunching Big Data, edureka.co, November 2020.

[7] Ashish Bakshi, blog on Apache Hadoop HDFS Architecture, edureka.co, November 2020.

[8] Rakesh Ray, blog on Hadoop Streaming: Writing A Hadoop Map Reduce Program In Python, edureka.co, May 2019.

[9] Data Flair, blog on Introduction to Hadoop Security – How to secure a Hadoop Cluster?, data-flair.training

[10] Sabyasachi Dash, Sushil Kumar Shakyawar, Mohit Sharma & Sandeep Kaushik, journal on Big data in healthcare: management, analysis and future prospects, Springer Open, June 2019.