Comparision of Data Structure approaches in Big Data

Abhishek Bodas
Computer Science Department
George Mason University
Fairfax, Virginia
abodas@gmu.edu

Abstract:
The world in which a byte of data is basic and ever growing building block of life, managing the huge amount of data efficiently is of foremost significance. Big Data is that union of technologies which refers to the practice of applying modern analytical software tools to all the types of data. The main purpose of big data is to realize insights, irregularities and enhance performance and outcomes by improving decision making to develop concise projections. For organizing and collecting such large amounts of data to perform operations on it and store it in an effective way, selection of a proper data structure approach is necessary. Selecting appropriate data structure assists in proper management of data sets. Data structure is one of the the critical parameters in judging the systematic working of the system.

With the world switching to the web for every basic aspect, everyday a large amount of data is generated. To keep up with increasing data, the approaches to manage such data should be updated and made more efficient. Sectors such as E-commerce and Finance are heavily dependent on productive use of data sets. (9) One of the most common and heavily used concept of big data is the Apache Hadoop. It consist of various open source software utilities to solve computation problems with massive amount of data. One such big data approach used to handle data is Hadoop Distributed File System (HDFS). Other approach which can be taken into consideration is Apache HBase. Both of these, have their own merits and demerits when taken in comparison based on efficiency, scalability, stability, speed, memory usage and few other factors. Many such factors will be thrown light upon further in this paper.

Key words:
Big data, analytical software tools, open source software utilities, Hadoop, Hadoop Distributed File System (HDFS), HBase, Data Structure, E-commerce, Finance, massive data sets, efficiency, stability, scalability, speed, memory usage.

1. Introduction :

Big data can be viewed as enormous quantity of data on internet protocol traffic such as transactional data, multimedia, communications, documents, other internet applications. In a nutshell, big data can be represented by these three core characteristics (three V's): (1)

- Volume: The ability to hold massive quantities of datasets.
- Velocity: The ability to execute in real-time.
- Variety: The ability of being structured, semi structured or unstructured.

Some other attributes of big data includes: exhaustivity, fine-grained, relationality, extensionality, veracity, value, variability (3)

.

Big data is the concept used to illustrate the process of applying high computing resources. The latest application of big data is its use in next generation applications such as Artificial Intelligence and Machine Learning to process highly complex and massive block of information.

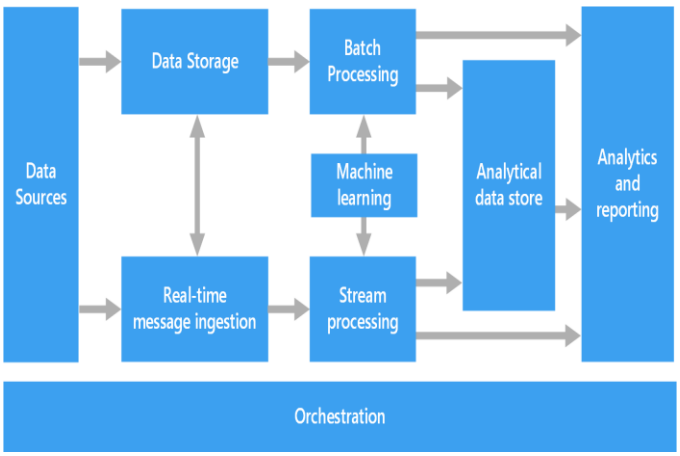

Figure 1: Big Data (5)

2. Literature Review

The Big Data concept can be bifurcated into two main technological aspects:

- **Operational Big Data**: This technology provides the system with operational abilities for interactive programs, real time environment and capturing and storing the data. MongoDB and NoSQL are the top technologies of operational Big Data. (7) It also focuses on executing request of high concurrency by exhibiting responses of low latency for responses in highly selective access criteria.
- **Analytical Big Data**: (10) This technology provides the system with sophisticated and complex analysis on majority of the data. Hadoop is one example that exhibit Analytical Big Data properties. It also focuses on high throughput and simplifying complex queries.

Both of these technologies display opposing requirements and behaviours and are well equipped to satisfy different requirements of the systems. For a complete solution, it is unwise to pick any one of these technologies, they are complementary and efforts should be made to deploy them in synchronization.

Together, these technologies operate in clusters on petabytes of data on a daily basis.

Comparing Big Data technologies with traditional data structures.

Big data can also be defined as the technology that surpasses the capabilities of conventional systems.The primary factor in development of Big Data technologies was to overcome the shortcomings of traditional Database management systems.When data size is taken into consideration, many traditional commercial fare very well. Greenplum, Vertica, Netezza, Teradata (2) are some of the analytical vendors display ability to handle multi-petabyte databases. But with current trend, it is probably enough for small scale organizations but not enough to satisfy the demands of large scale internet companies. With respect to scalability of the data, commercial systems such as MySQL and Postgres trail behind and are considered insufficient. They involve data import to maintain consistency, after that the data is queried. This process is slower and limits the ability to handle live data stream. Also, there are integration issues with the relational engines within. These shortcomings are conquered by modern technology such as MapReduce. MapReduce is able to scale huge amounts of data, but it fails to provide high-level infrastructure which is used only to process data and is unable to manage data, similar to traditional relational systems. MapReduce only provides access to collection of files. As a result, maintenance and consistency factors must be dealt by the user. It is based on processing and replicating large data sets, hence it provides responses with high latency.

Leading data structures:

Some other advanced data structures that can be used are:

- **B Trees**: B Trees are quite useful for data retrieval. But, they cannot be used for data storage. Also, B-Trees are performance problems because of its random operations that generate duplicate I/O's.
- **LSM & Fractal Tree**: The highlight of this data structure is its high performance in data storage operations, but performing read operations is a tedious task.
- **Bloomfilter**: This data structure is probabilistic in nature used to check whether the element is present in the data set. Its limitation comprises of generating false positive results and the only possible operation allowed is element addition.
- **Hyperloglog**: Hyperloglog is a data structure used to search unique elements from the data entries. It can count up to billions of distinct items but the accuracy is mere 2%

Most of the aforementioned limitations of the data structure has been successfully conquered by Big Data. Two of the most useful approaches pursued by Big Data are Hadoop Distributed File System (HDFS) and HBase.

Hadoop:

Hadoop is data processing technology designed with write once storage infrastructure. (8) As soon as the web started generating millions of pages of data the problem of indexing the pages was born. In order to sustain this heavy load of data, there was a demand for an efficient data management technique and Hadoop was created. Hadoop is designed in such a way it processes huge amount of data to work in parallel execution by using the MapReduce model to divide the query into different parts. Fault tolerance is one of the important advantages of Hadoop, even if the task fails, the data can be easily recovered.

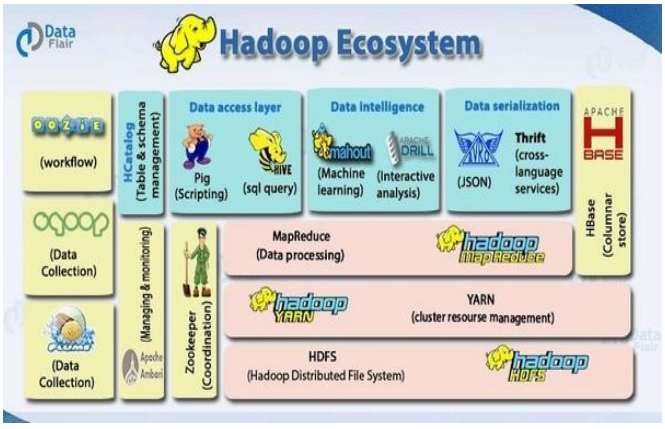

Figure 2 : Hadoop Ecosystem (6)

HDFS:
HDFS is a file system of Hadoop that allows the user to store large amounts of data throughout various nodes in the Hadoop cluster. It is developed on Java programming language and works on commodity architecture. In a distributed environment, it behaves as an underlying storage mechanism for various structures of data.

Some attractive properties of HDFS are:

- Scalability
- Fault tolerance
- Reliability
- Cost efficient data storage
- Rapid data transfer rates between nodes
- To withstand large clusters of data

The robustness of HDFS allows it to operate under various systemic and physical environments. HDFS works with wide variety of concurrent data applications. (4) With computing on distributed data among various servers, the shared storage resource can develop on a linear scale and afloat economically at any amount of storage requirements.

"HDFS has production scalability of 200 Petabytes of storage and a single cluster of 4500 servers with billions of files and blocks"

Components of HDFS:
HDFS implements on the Master-slave architecture which is useful to store data in various clusters. The job of master node is to distribute the chunk of data to the slave nodes. Every cluster in Hadoop contains a Master node or Name node to carry out file system operations and Slave node or Data node to carry out data storage process on computing node. HDFS is expandable and can fit in large amount of data on these Data nodes

Master Node / NameNode: This node is a master of the system. It maintains the file system tree and the metadata for all the files and directories present in the system. 'Namespace image' and the 'edit log' are used to store metadata information. It has knowledge of all data nodes which contains data blocks for a given file, however, it does not store block locations persistently. This information is reconstructed every time from data nodes when the system starts. This node manages all the slave nodes and assign work to slaves. It should be deployed on reliable hardware as it is the centerpiece of HDFS.

Slave Node/DataNode: These are slaves which reside on each machine in a cluster and provide the actual storage. Manages read and write requests for the clients They can be deployed on commodity hardware. If any slave node goes down, Master node automatically replicates the blocks which were present at that data node to other nodes in the cluster.

Use Case - HDFS & MapReduce
Perform batch analytics to gain SKU level insights, and gets involved recursive or sequential calculations. HDFS and MapReduce frameworks is better suited than complex Hive queries on top of Hbase. MapReduce was used for data wrangling and to prepare data for subsequent analytics. Hive used for custom analytics on top of data processed by MapReduce.

HBase
HBase is modeled after success of Google Bigtable. It is written in Java & has open source. It is in form of non relational, Not-Only-SQL distributed database. It is developed for Apache Hadoop project for Bigtable implementation. HBase runs on top of HDFS. HBase follows CP type CAP theorem (Consistency, Availability, and Partition Tolerance) theorem. It can store huge quantities of sparse data in fault tolerant manner. Features of HBase are compression, in-memory operation and Bloom Filters. HBase handles input and output for Map Reduce jobs through tables. It is accessed through the Java, REST, Avro & Thrift gateway APIs. HBase is a column-oriented key-value data store making it popular with Hadoop and HDFS.

HBase implementation are highly scalable in nature , offers sparse data, distributed across servers, and multidimensional-sorted maps. It provides very low latency access over fast-changing data.

HBase is written in Java which makes it highly scalable. It can work on variety of data types. Feature of HBase is that it is fault-tolerant and resilient. Data searching is quick owing to column-oriented table structure which provides ease on right data lookup among humongous data fields. Queries are scalable due to distributed data storage is spread across servers. Hence query processing exceed the limit of single server. HBase is best suited for transactional processing. Query response time is not interactive i.e. OLTP. HBase operations run in real-time on the database.

HBase is ideal for application which require fast & frequent random read or random write operations unlike traditional RDBMS. (12) It can handle billion rows & column, blend with variety of data types, structures, data source & schema & produces seamless output. It is integrated with Hadoop.

HBase is best suited for online real-time analytics & data query. This is done through MapReduce via API. Process is automated & involves configurable sharding for datasets or tables. Storage of data is done in NoSQL database for ease of access during real time. HBase has developed API's for data push & data pull. Hadoop MapReduce is integrated to implement bulk operations like analytics, indexing. Best

combination for getting finer results is use Hadoop as repository for static data and HBase will act like data store for that data. During processing data will get changed during real-time. HBase provides best solution for time series analysis and click stream data storage and analysis.

Nowadays companies need real time analytics, hence they deploy HBase. HBase implementation for various companies like Pinterest, Facebook, Flipboard, FINRA etc is done to store trend graph, personalize content feed for user, count likes, messaging services, store graph data. (11)

Components of Hbase

HBase HMaster : It is monitoring agent running on slave node & acts as an interface for all metadata changes. HMaster is responsible for load balancing across all Region Servers and maintenance of Hadoop cluster state.

RegionServer: These nodes are deployed on each machine and hosts data. Responsibility of RegionServer is to process I/O requests. Client send read & writes requests & RegionServer assigns the request to a specific region, where actual column family is located.

Regions: These are basic building blocks of HBase cluster. Region consists of distribution of tables and Column. For each column family there is a store. It is composed of Memstore and Hfile component.

ZooKeeper: It is centralized service to ensure reliable distributed coordination . It maintain configuration information & naming. It is an open source server responsible to provides distributed synchronization and group services. Distributed synchronization is Zookeeper ensures Distributed Synchronisation by accessing various distributed applications running across multiple cluster by providing coordination services between nodes. HBase is inoperable without ZooKeeper.

Use Case - HBase

HBase is suited for real-time environments. It helps to derive critical information from the application logs & web servers. HBase performance is superior compared to HDFS due to huge velocity.

Implementation of HDFS & HBase for an eComerce website

HDFS.-It would help an e-commerce website to store millions of customer's data in a distributed environment which grew over a long period of time. HDFS batch processing over that data will help to assess customer behaviors, pattern, requirements which enables company to strategically analyze what type of product, customer purchase in which months etc.

HBase -In an e-commerce website, , HBase would search for a product among millions of products, it optimizes the request and search process, producing the result in real time.

3. Comparative Study

Comparison: HDFS Vs HBase

1. Need: HDFS is needed for processing huge data by organizing data sets on various clusters whereas HBase is a distributed SQL data storage mechanism built on top of HDFS in the Hadoop architecture layer.
2. Latency: High latency operations on data sets are provided by HDFS whereas HBase provides low latency access to data in huge data sets.
3. System Failure: HDFS is a fault tolerant mechanism that performs rapid data transfer between clusters during system failures too. As there are plenty of nodes in the cluster, system failure is evident on few nodes whereas HBase is executed under CAP theorem within CP type and provides Consistency, Availability and Partition Tolerance (CAP).
4. Write pattern: Write pattern in HDFS in Append Only pattern, whereas write pattern on HBase is Random read and write
5. Read pattern: Some examples of scan on tables in HDFS are Full table scan, partition table scan and many others and HBase performs random read, small range scan or table scan.
6. W/R pattern: HDFS illustrates WORM (Write once and read many times) use cases whereas HBase performs random write and read of HDFS data.
7. Hive (SQL) Performance: HDFS performs faster, HBasepeforms slower.
8. Structured Storage: HDFS follows do it yourself or Sequence File whereas Sparse column family data model is HBase's storage model.
9. Maximum Data Size: For HDFS maximum data size is 30 Petabytes and for HBase maximum data size is nearly 1 Petabytes.
10. Dynamic changes: HDFS has a rigid architecture which disallows Dynamic storage, whereas for single applications HBase allows Dynamic changes.
11. Data Distribution: HDFS cluster contain nodes that stores the data which is distributed all over the system. Regions in the HBase distribute the tables on the cluster. When the data increases, these regions automatically split and data is redistributed.
12. Data Storage: HDFS follows batch processing on archieved data as the huge data sets are stored in a distributed environment. File sizes are 64MB or 128MB. HBase uses key value pairs to store data in columnar database, these columns are stored with each other in the same location. Due to this real time processing and fast reading is possible.
13. Data Modelling: Both of these technologies work on key value pairs but, HDFS makes use of MapReduce techniques and Hbase uses Google's Bigtable model.

14. Latency Operations: HDFS facilitates high latency operations whereas Latency operations are low for HBase.
15. Accessibility: MapReduce is used to access HDFS whereas HBase can be accessed by shell commands, JAVA client API, REST and others.
16. File System: HDFS uses GFS, a distributed file system used to store avalanche of data. HBase uses HDFS for multidimensional storage which enforces faster lookups and updated data in the database.
17. Query Language: HIVE Query language is applied by HDFS where as HBase does not support any query language.
18. In Memory processing: HDFS does not contain an in memory engine which makes the process slower. Presence of in memory processing engine makes the speed of HBase processing extremely faster.
19. Data Analysis: HDFS follows transparent execution. As Hbase is a NoSQL database its sorts the data with values of their key.
20. Batch Analytics: HDFS is adept for batch analytics but poor in real time analysis. HBase is not fit for batch analysis but handles real time operations pretty well.
21. Random Data Access: HDFS alone does not support Random Data Access. HBase, being a non-relational database provides Random Data Access.

4. Future Approach

Suggested Big Data Approach for future:

In this paper, I have mentioned various types of data structures that have their own way implementation. They have their own sets of merits and demerits. In my research I found out that for any approach, the speed and volume of technology is the major aspect. Faster the technology, better the technology. If the technology can cope up with ever increasing data storm with stability, the more reliable it will be. Such technologies can be implemented and level up the data processing techniques and can be more commonly used. One aspect I would like to throw light upon is the efficiency, type and structure of algorithms used. Using proper algorithm techniques will ensure faster processing speeds. Another factor that would better the functioning of big data would be more thorough information and data gathering so that unwanted and unused data can be thrown out, thereby making the data system faster and better. According to my findings, HDFS and HBase are the best systems that I came across, they both have amazing properties, but some limitations too.
As in any state of the art systems there is always a scope of improvement, even HDFS and HBase can be improved. HDFS, HBase, Bloomfilter, ADS, all of them have a key quality, if there are some integration techniques to integrate them and create a data structure, it would be a huge advancement and would solve not all but many problems of the current systems. It may also provide a unique and feasible approach.

ADS (A-train Distributed System)
ADS comprises of a collection of autonomous systems which may or may not be at different locations are connected to each other via a distribution middleware in a network which allows the complete system to work by sharing the resources amongst each other so that the user believes the system as a singular unit. Similar to the parent and slave nodes in Hadoop, ADS has Pilot Computer (PC) and Distributed Computer (DC), there could be many DC's but only one PC's. The flow of data is bidirectional. The inspiration behind its name is because that it can support heavy heterogeneous data structure to execute big data with any challenges thrown upon by the 3V's.

5. Conclusion

In overall conclusion, both HDFS and HBase have wonderful technologies in their own. Both technologies are created to store the Big Data and to provide easy access, optimize working while computing. Both technologies complement each other as HDFS stores the data the HBase establishes a schema on the data on storage and retrieval for later usage HBase is No SQL distributed database that runs on top of HDFS, hence combination of both gives us a flexibility to use the benefits of both, in a tailored solution.

6. Acknowledgment:

I (Abhishek Bodas) am extremely grateful for insights and comments of Ms. Ghada Alnifie to guide me in a proper direction and to provide the much need motivation.

7. References

Bibliography:

(1). Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.657.1513&rep=rep1&type=pdf#page=757
(2.). Retrieved from https://ieeexplore.ieee.org/document/6188576
(3.). Retrieved from https://journals.sagepub.com/doi/10.1177/2053951716631130
(4.). Retrieved from https://dl.acm.org/citation.cfm?id=2618218
(5.). Retrieved from https://docs.microsoft.com/en-us/azure/architecture/data-guide/big-data/
(6.). Retrieved from https://www.altencalsoftlabs.com/apache-hadoop/

(7.). Retrieved from https://www.mongodb.com/scale/operational-vs-analytical-big-data
(8.). Retrieved from https://dl.acm.org/citation.cfm?id=2536227
(9.). Retrieved from https://www.jstor.org/stable/41703503?seq=1#page_scan_tab_contents
(10). Retrieved from https://dl.acm.org/citation.cfm?id=2367519

(11.). Retrieved from https://www.usenix.org/system/files/conference/fast14/fast14-paper_harter.pdf
(12.). Retrieved from https://ieeexplore.ieee.org/abstract/document/6182030/