International Journal of Database Theory and Application
Vol.9, No.1 (2016), pp.45-66
http://dx.doi.org/10.14257/ijdta.2016.9.1.05

Comparative Study of Big Data Computing and Storage Tools: A Review

Bakshi Rohit Prasad and Sonali Agarwal

Indian Institute of Information Technology Allahabad
rohit.cs12@gmail.com, sonali@iiita.ac.in

Abstract

As a result of tremendous rise in internet usage like social media and forums, mail systems, scholarly and research articles, daily online transactions from multiple sources like health care systems, meteorological and environmental organizations etc., the data collected has shoot up exponentially. This vast collection of data, called Big Data, has caused the traditional tools incompetent for managing it from either of storage, computing or analytical perspective. There is an immense need of architectures, platforms, tools, techniques and algorithms to handle Big Data. The available technologies deal with two broad aspects related to Big Data that are Big Data Storage Management and Big Data Computing, focused to overcome various challenges such as scalability, faster processing speed, multiple format data processing, availability, faster response time and analytics etc. This paper reviews recent trends of storage and computing tools with their relative capabilities, limitations and environment they are suitable to work with.

***Keywords**: Big Data Computing, Big Data Computing Tools, Big Data Storage Tools, Big Data Analytics*

1. Introduction

Current world is the world of data. We have data all around us. This data is huge in volume and being generated exponentially from multiple sources like social media (Facebook, Twitter *etc.*) and forums, mail systems, scholarly as well as research articles, online transactions and company data being generated daily, various sensors' data collected from multiple sources like health care systems [1], meteorological department, environmental organizations *etc.* The data in their native form has multiple formats too. Also, this data is no longer static in nature; rather it is changing over time at rapid speed. These features owned by bulk of current data, put a lot of challenges on the storage and computation of it. As a result, the conventional data storage and management techniques as well as computing tools and algorithms have become incapable to deal with these data. Despite of so many challenges associated with these data, we cannot ignore the potentials and possibilities lying in it that can support for analytics and for hidden patterns identification. These analytics can be very effective in making business strategies and predicting effective decisions, finding various hidden patterns associated with several diseases and their attributes, in genomics to analyze thousands of genes and their associated roles in biological systems, in climate monitoring and prediction, GPS and other satellite parameters mining *etc.*

1.1 Big Data Formats and its Sources

Big data is a huge collection of data over a time frame that is so complex and difficult to process and manage using conventional database management tools [2]. Big Data and its sources can be categorized into following categories:

- Structured Data - generated from various researches efforts, CRM (Customer Relationship Management) and other such traditional databases.
- Semi-structured Data - such as XML formatted data.
- Unstructured Data – These data can be generated by humans such as social media, discussion forums and customer feedback, comments, emails *etc.* or may be generated by machine such as online transactional, satellite and environmental data collected through various sensors, web-logs, call records *etc.*

1.2 Big Data Characteristics and Big Data Challenges

There are four basic characteristics depicted in Figure 1 that Big Data shows always. These are Volume, Variety and Velocity [3-4]. Each aspect puts a challenge in handling and processing this data to extract some meaningful implications. These challenges could be in collection, integration, storage, sorting, searching, retrieval, analysis, and visualization from the various aforementioned key aspects of the Big Data.

Volume: As per current scenario, various sources of data generations throughout the world, generating the data at tremendous speed per day. Facebook and Twitter are the kind of social media that produce daily approximately 500 TB and 7 TB of data respectively [5-6]. According to a survey done by IBM [7], 2.5 quintillion bytes of data are being generated every day. A quintillion equals 1018 bytes.

Variety: The data is being collected from multiple sources in different formats already discussed - Structured data, semi-structured data and unstructured data. Out of which the unstructured data is a big hurdle in computing and analysis part as they do not have a common format, therefore a common tool or algorithm cannot be followed in variety of modalities of the data.

Velocity: This aspect of Big Data is associated with the speed at which data is being produced and processed. When we look for the real time processing and response the speed of data production becomes a critical challenge for analytical and visualization tools. If the response time of the analytical tools is not capable to cope up with speed of data arriving, the result becomes useless.

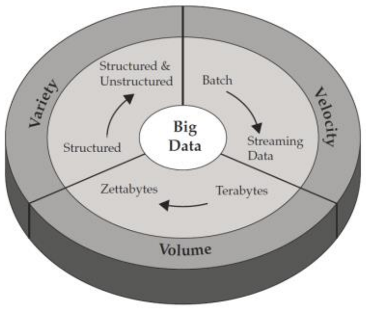


Figure 1. Varying Characteristics of Big Data Over a Period of Time

Apart from above mentioned 3Vs of Big Data, there is one more important challenge inherent in the term Big Data known as veracity and described as below:

Veracity: The last but not least challenge associated with Big Data is the veracity that means the uncertainty in data which could be due to incompleteness, ambiguity, reliability of the source of data, the deception factor involved, approximations made in various models *etc.* This puts a great challenge of genuineness and trust on the data being used for analytics.

1.3 Need for Big Data Management and Processing

There are various purposes for handling Big Data and exploring effective management and methodologies. The Big Data can be used for following purposes:

- Business Intelligence: Intelligence is incorporated in making various business strategies as listed below:
    - Business alignment strategies: It is required so that the output value and strategy may be tied up closely and may give the result after appropriate decision making.
    - Behavioral and organizational strategies: These strategies speed up the task performance and improve productivity.
    - IT strategies: It provides improved efficiency in IT at lower cost.
    - Promotion and Advertisement strategies: These are required to make intelligent and effective marketing and advertisements to raise the profit.
- Crime/ Fraud/ Fault Detection and Prediction: In this reference, the Big Data analytics can play a vital role in several aspects. For example, some of the applications may be as follows:
    - Credit card transaction: Analytics can predict the probability of a credit card holder of being fraudulent.
    - Criminal identification is possible through deep analysis of CDR (Call Detail Record).
- Querying, Searching and Indexing
    - Keyword based search
    - Pattern matching
- Knowledge discovery / Data Mining
    - Healthcare system: In healthcare system, Big Data Analytics could play a very vital role in variety of disease pattern identification, prediction and therapy suggestions such as diabetes [8], [9] heart, cancer and Parkinson disease [10], *etc.* through deeply digging Big Data using various data mining techniques [11].
    - Statistical Modeling: In various day to day life transactions.
    - Genomics: To identify new patterns and relations among the genes and other organic structures present in humans and other living beings.
    - Climate predictions and operative suggestions can be made based on the effective analytics of huge amount of climate and environmental data.
- Defect detection and prediction in software and manufacturing products [12].

1.4 Organization of Paper

The organization of the paper goes the way shown in Figure 2. The first section introduces the Big Data, different sources of their generation, their characteristics and challenges associated with it. Also, it discusses the need of handling and processing of Big Data in current scenario in different areas of applications. The second section contains a detailed description about available four well known tools and techniques for storing and four for computing Big Data with along with their advantages/disadvantages and the suitable environment they are applicable to work with. The third section gives the comparison of various tools and techniques based on their capabilities and limitations associated with them. The fourth section finally concludes this paper with some useful suggestions and recommendations.

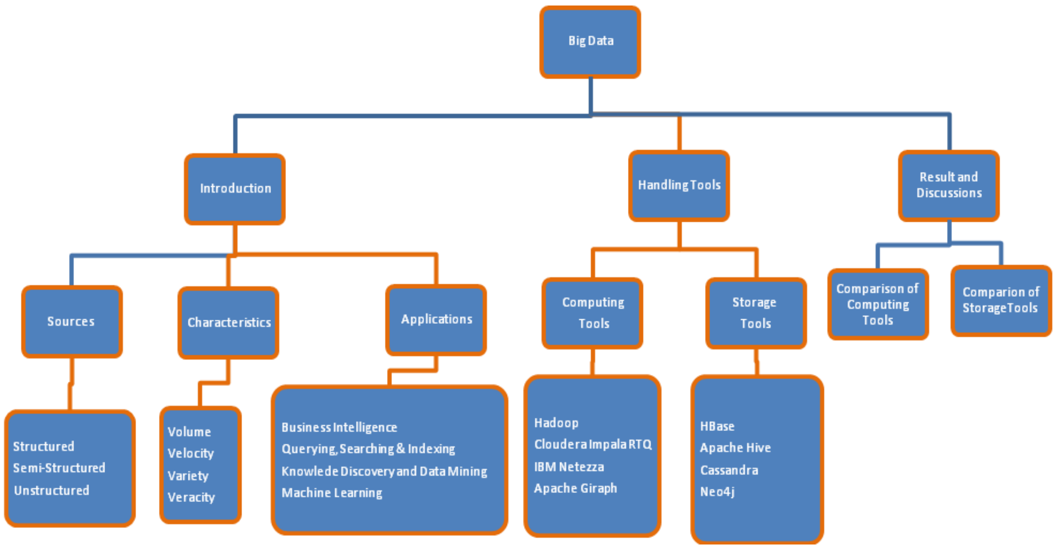


Figure 2. Organization of the Paper

2. Big Data Storage and Computing Paradigms and Tools

To draw useful implications from the Big Data, appropriate tools are required to perform data collection, data storage and processing for various analytical perspectives. The normal process flow diagram for Big Data Analytics is shown in Figure 3.

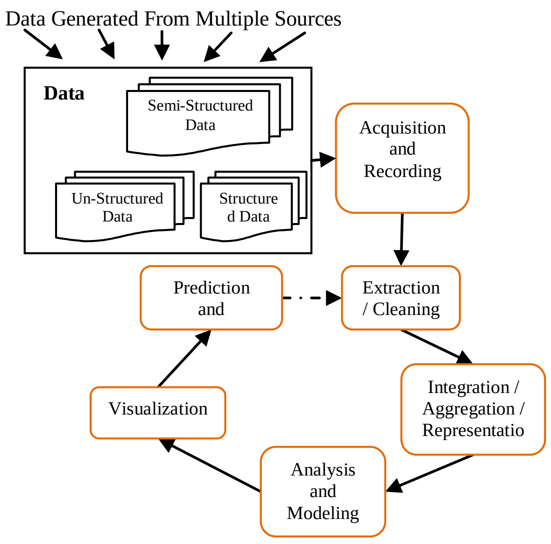


Figure 3. Process Flow Diagram for Big Data Analytics

2.1 Big Data Computing / Processing Tools

2.1.1. HADOOP MapReduce

Hadoop also known as Apache Hadoop [13-15] is an open source framework that has been provided by Apache. This framework is developed to deal with distributed and scalable computing as well as storage management of huge data, the Big Data. Hadoop platform includes two core layers; one is the distributed storage layer that is built on the HDFS (Hadoop Distributed File System) [16] inspired by the Google File System [17] and the second layer is the distributed computing layer whose key idea is MapReduce computing paradigm, initially, developed by Google.

The Hadoop framework follows the key idea of data intensive computations where it is better to transfer the computation code/program to the data rather transferring the bulk of data to the computing code. Hadoop platform involves a cluster of storage/computing nodes (or machines) out of which one node is assigned as master and other as the slave nodes. The HDFS [18] maintains each file in the chunk of same size blocks (except the last block). Also, various replicas of these blocks are maintained on various nodes in the cluster for the sake of reliability and fault tolerance. The Map-Reduce [19-20] computing technique divides the whole task of processing into smaller blocks and assign to various slave machines where the required data is available and executes computing right at that node. In this way it saves significant time and cost involved in transferring data from data server to the computing machine. Following are the advantages, disadvantages and latest version of Hadoop.

i. Advantages of Hadoop

- Open source: Being an open source, Hadoop is freely available [13].
- Cost Effective: Hadoop saves cost as it employs cheaper low end cluster of commodity of machines instead of costlier high end server. Also, distributed storage of data and transfer of computing code rather than data saves high transfer costs for large data sets [13].
- Scalable: To handle larger data, the Hadoop is capable to scale linearly by putting additional nodes in clusters [13], [14].
- Fault Tolerant and Robust: It replicates data block on multiple nodes that facilitates the recovery from a single node or machine failure. Also, Hadoop's architecture deals with frequent malfunctions in hardware. If a node fails the task of that node is reassigned to some other node [19].
- High Throughput: Due to batch processing high throughput is achieved in Hadoop [24].
- Portability: Hadoop architecture can be effectively ported [21] while working with several commodities of operating systems and hardwares that may be heterogeneous [22].
- On-Demand Service [23]: It can be set manually on lent computing nodes on cloud or can be used as on-demand service such as EMR (Elastic MapReduce) [25] provided by Amazon or AzureMapReduce or CloudMapReduce [26].

ii. Disadvantages of Hadoop

- Single Point Failure: Hadoop's (version up to 2.x) HDFS as well as MapReduce suffer from master level single points of failure [13] [28].
- Low Efficiency/ Poor Performance than DBMS [27]: Hadoop shows lower efficiency due its inability to switch to the next stage before completing the previous stage tasks causing Hadoop unsuitable for pipeline parallelism, runtime scheduling that causes degraded efficiency per node. Unlike RDBMS, it has no specific optimization of execution plans that could minimize the transfer of data among various nodes.
- Inefficient Dealing with Small Files: As HDFS is meant for high throughput optimization [24], it does not suit to random reads on small files [51].
- Not Suitable for Real Time Access: MapReduce and HDFS employ batch processing architecture hence; it does not fit for real-time accesses [18].
- Hadoop does not support iterative behavior which is common to any procedural programming paradigm.

2.1.2. Cloudera Impala and Cloudera Enterprise RTQ

Cloudera Enterprise RTQ driven by Cloudera Impala enables enterprises to exploit advantageous features of SQL tools to achieve real-time analytics potentials when working with large volumes of data, whose nature may be structured and unstructured [29]. Various business analysts and IT industries can use it over a wide range of supported data types as well as huge data volumes to interact in real time with a HBase or a HDFS data store for the sake of analytics. The Cloudera Impala's position in Hadoop stack is depicted in Figure 4.

i Advantages of Cloudera Impala and Cloudera Enterprise RTQ

- Flexible data model: It works with the same stored data and metadata in HBase or Hive, *i.e.* it supports structured as well as unstructured data [29].

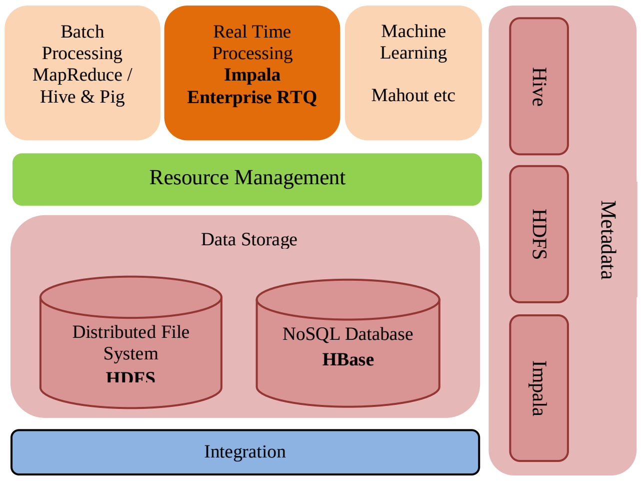

Batch Processing MapReduce / Hive & Pig
Real Time Processing **Impala** **Enterprise RTQ**
Machine Learning Mahout etc
Resource Management
Data Storage
Distributed File System HDFS
NoSQL Database **HBase**
Integration
Hive
HDFS
Impala
Metadata

Figure 4. Cloudera Impala Status in Hadoop Stack [30]

- High Performance: It executes queries at least 10 times faster than Hive/MapReduce. Pure I/O bound queries and queries with at least one join have shown 3-4 times and 7-45 times performance gain respectively. Aggregation queries have been speed-up by approximately 20-90 times as compared to HiveQL (Hive Query Language) [29].
- Real-Time Interaction Support: Cloudera Enterprise RTQ reduces response time [30] of queries to seconds unlike minutes in HiveQL or MapReduce, as shown in the comparison chart in Figure 5. Up to 90% computing cost is saved [29] spent on ETL services.
- Security: It offers Kerberos authentication support. Role based authorization is also supported in Cloudera Enterprise RTQ.

ii. Disadvantages of Cloudera Impala RTQ

- All joins operations are performed in memory capacity limited by the smallest memory node present in the cluster [29].
- It does not support querying streaming data such as streaming video or continuous sensor data *etc.* [30].
- Deleting individual rows is not possible in Cloudera Enterprise RTQ and it still does not support internal indexing for files [29].
- Single Point Failure in Query Execution: It quits the entire query if any host that is executing the query fails [30].

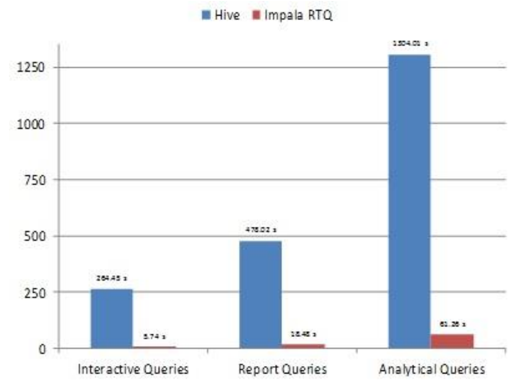


Figure 5. Comparison of Response Time of Cloudera Impala RTQ and HiveQL [29]

2.1.3. IBM Netezza

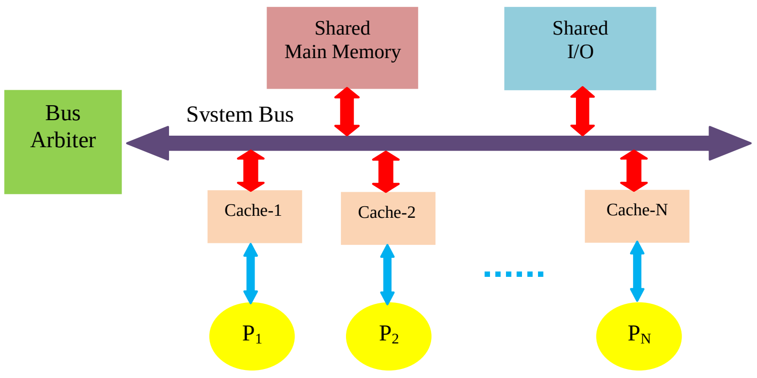
Netezza can be placed in both storage and computing category as it provides data warehouse as well as analytics appliance. Netezza is based on Asymmetric Massively Parallel Processing (AMPP) shared-nothing architecture which is basically a two-tier architecture [31,33] shown in Figure 6 and Figure 7 which handle large complex queries very quickly. The first tier employs a high performance Linux based Symmetric Multi-Processing host. This tier is responsible for compiling data query jobs and accordingly generating execution plans. It breaks down the original query task into sub-tasks suitable for parallel execution. Afterwards, these subtasks are distributed over the second tier [32]. The second tier involves hundreds of intelligent snippet processing blades called S-Blades that form the MPP engine of the appliance. These independent servers (S-Blades) contain Intel-based central processing units having multiple cores. Also, it includes multi-engine as well as high-throughput Field-Programmable Gate Arrays (FPGAs) [31,33].

i. Advantages of IBM Netezza

- Massive Parallel Processing: The load time in MPP is of order of approximately 2 TB/hour and backup and its restoration rates are of order of 4 TB/hour and above [32].
- TwinFin, an integrated component of Netezza, provides fast analysis of large data volumes of order of petabytes [32] [34] and in-database processing in Netezza causes significant reduction in terms of latency [35].
- Netezza supports models like Hadoop, Java, Python, C++, *etc.* which is programming models used majorly these days [36].
- IBM Netezza provides faster query performance using concepts of parallelism and pipelined computation [33].
- Netezza does not use indexes, table spaces. Thus, Data Definition Language becomes much simpler [36]. IBM Netezza Analytics packages are free and inbuilt in it [37].

ii. Disadvantages of IBM Netezza

- Netezza does not suit for online transactional processing [34-35].
- Netezza does not employ any query tuning mechanism [35].
- Netezza does support nested correlated queries whereas some other Business Intelligence tools leverage this aspect to fasten turnaround [35-36].



P: Independent Homogeneous Processor

Figure 6. IBM Netezza Tier-1 [31]

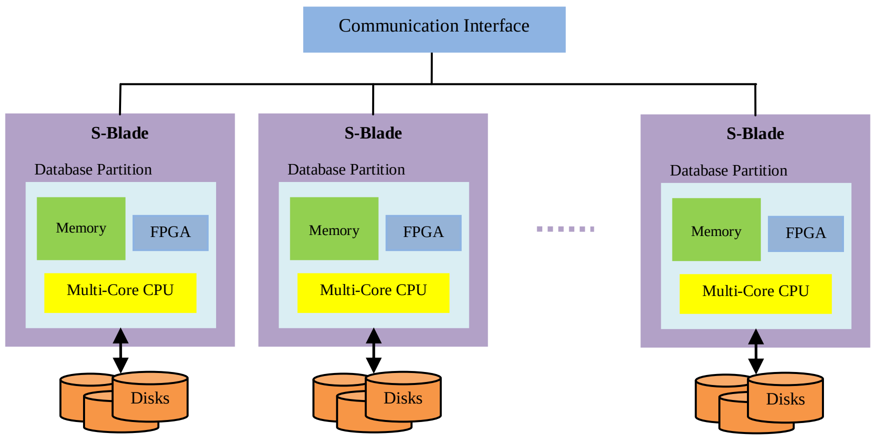


Figure 7. IBM Netezza Tier-2 [31]

2.1.4. Apache Giraph

Apache Giraph, running on top of Hadoop framework, is the open sourced version of Google's proprietary product Google Pregel [38]. It also has distributed processing structure suitable basically for large scale graph processing [39,43-44] such as in analysis of the interconnected web (for Page Ranking) or social media (Facebooks, Twitters, LinkedIn *etc.*) interaction that are nothing but a graph of interconnected vertices which may be a web page linked to another page through the edge (hyperlink) or it may be users in social media connected with each other through edges representing friendship or some kind fan or business following *etc.* The Giraph basically based on the Valient model [40] of Bulk Synchronous Parallel computation model. Usually, the Giraph is used in combination with well-known graph databases such as Infinite Graph or Neo4j or with Hadoop.

i. Advantages of Apache Giraph

- Scalable: It is used for large scale graphs' analysis involving up to trillion of edges. Giraph computing is based on the Valiant model of Bulk Synchronous Parallel computation [40].
- Fault Tolerant: It achieves fault tolerance by employing check-points technique [41].
- Simple for Graph Based Problems: Apache Giraph naturally models the graph based problems which is based on 'Think like vertex' approach which

is a vertex centric programming model as shown in Figure 8 [41]. Instead of writing several mapper/ reducer classes a vertex is implemented. Vertices can send and receive messages to each other throughout computation [42].

- Less I/O and In-memory computation: It holds the state of a graph in-memory throughout the execution of algorithm. It uses no sorting technique for computations hence time effective in query response [42].

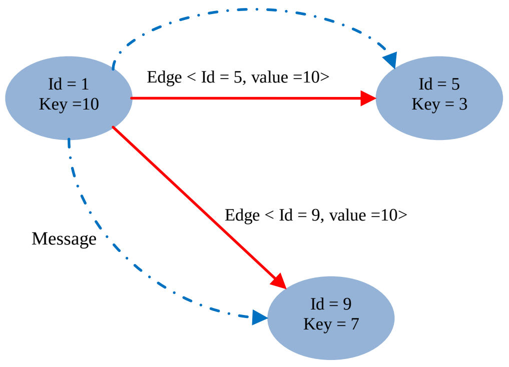


Figure 8. Vertex Centric Computing Model of Apache Giraph [43]

ii. Disadvantages of Apache Giraph

- Apache Giraph is still in a very immature phase of development [41-42].
- It lacks in providing a complete set of offered algorithms [42].

2.2. Big Data Storage Tools

2.2.1. HBase

Apache HBase [45-46] is an open source non-relational database that aims to host very large sized tables consisting of millions to billions of rows and columns. HBase allows grouping various attributes to make column families as described in Figure 9. In this way, attributes of a column family are put together in the table [47]. Apache HBase is a distributed version of the database that facilitates the same capabilities to Hadoop's HDFS as the Big Table of Google provides to the Google File System [48].

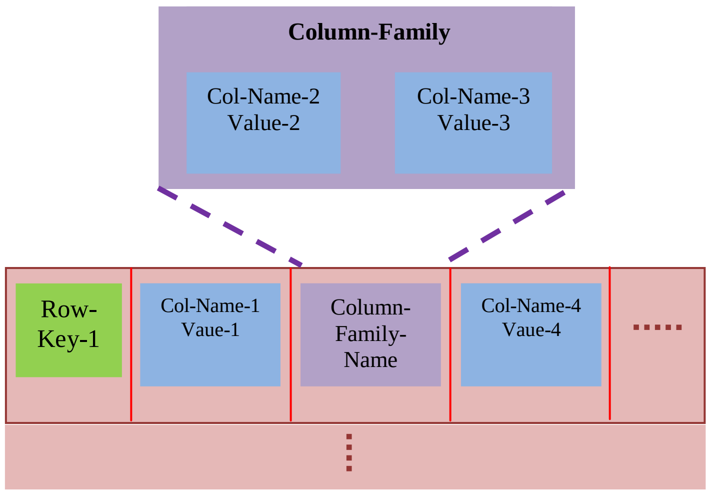


Figure 9. Column Family Containing as Attributes Columns 2 and 3 [46]

i. Advantages of HBase

Apache HBase provides following capabilities:

- Scalability: It scales horizontally, as it is a wide-column key-based data stores. Therefore, it is robust also [45].
- HBase performs consistent reads/writes on the underlying data in the database but it is optimized for performing read operations [49].
- Random and Real Time Read and Write Access: HBase stores data in MapFiles that are basically an Indexed Sequence Files. Thus, it becomes a suitable choice for streaming analysis of a MapReduce kind of style that involves occasional random look ups [51].
- It suits well to store sparse data, found usually in several data cases [49].
- Fault Tolerance: In HBase, the failovers between Region Servers are supported and handled automatically [45], [46].
- Real Time Query support: For real time interaction with data, HBase offers Bloom Filters and Block caches [45].

ii. Disadvantages of HBase

There are some technical limitations with almost all NoSQL solutions and so is the case with HBase:

- Compactions affect the consistent low latency in HBase [49].
- Single Point Failure: In HBase rows are partitioned into regions [49] and each region is allocated to a Region-Server which becomes a single point of failure. Also, HBase takes long recovery times for node failures. On the other hand, the Region Server failover takes approximately 10-15 minutes which is quite high [52].
- Operationally Inflexible: HBase's master-oriented architectural design routes all reads and writes via Region-Server, thereby causing no workload separation across different replicas in a cluster [52].
- There are no optimized classic OLTP applications or analytics support in HBase. It does not directly support SQL, however its integration with Hive supports HiveQL [49-50].

2.2.2. Apache Hive

Apache Hive, built upon Apache Hadoop, is a data warehouse tool that provides effective management of very large data which is stored in HDFS. It also provides effective query execution facility using a query language resembling to SQL. This query language is known as HiveQL. Since the language is SQL-like, hence the SQL users can easily fire their query on the database. Also, it is helpful for those programmers who know the MapReduce paradigm of computing. They can write their own mappers/ reducers and plug in them into HiveQL to achieve data analysis and data summarization that is more sophisticated and, otherwise could not have been achieved using capabilities already being provided with HiveQL [53].

i. Advantages of Apache Hive

Apache Hive facilitates following capabilities:

- Easy data ETL services: Hive provides data extract, data transform and data load operation in an easy way. Hive performs reads/writes which are independent of file formats. It uses SerDe (Serializers/Deserializers) framework libraries to support formats such as text, sequential files, control delimited or a user defined file format [54-55].

- Hive has provision for tables at external level to facilitate data processing without storing it actually on HDFS. Data partitioning in Hive, is performed at table level that improves query execution performance [54].
- The Metadata store facility introduced in the architecture of Hive enables easier look ups for query processing and analytics [54]. HiveQL can be enhanced with custom functions such as: UDF (scalar functions), UDAF (aggregation functions) and UDTF (table functions) [55].
- Scalability: It achieves scalability by dynamically adding more machines to the Hadoop node cluster and Hive has been made fault-tolerant to recover from node failures [55].

ii. Disadvantages of Apache Hive

- It does not work for OLTP, hence not suited for real-time queries [53].
- Hive is incapable of making updates and delete at row-level. Also, a single record insertion is not supported by Hive; rather it is loaded from a file in batch using LOAD command [54].
- Correlated sub queries cannot be executed in Hive as well as access control has not been implemented in Hive [54].

2.2.3. Neo4j

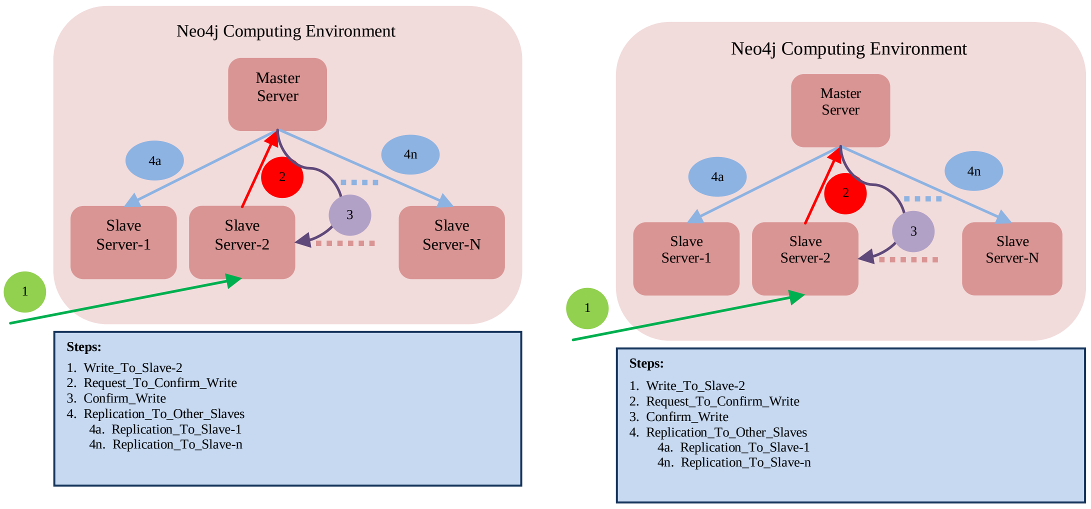
Neo4j is a graph database that is available as open source as well as commercial licensed version. It stores data modeled as a graph which is a collection of nodes (with an Id) and relationships among them represented as edges in the graph. These nodes or edges store some properties represented as key/value pairs. Neo4j is an embedded, fully transactional, a disk-based Java persistence engine.

i. Advantages of Neo4j

- Massive scalability: Neo4j can easily handle large graphs containing nodes / relationships / properties of order of billions using even a single machine. Its computation can run in parallel on multiple processors via read threads [54].
- Schema-Free Database: The schema free architecture provides for an efficient storage solution for semi structured information [55]. Since, nodes do not have fixed set of properties hence; it facilitates easy schema-changes.
- High Performance: Unlike RDBMSs, Neo4j overcomes the performance degradation problem with several joins by performing graph traversal that works at the same speed no matters how much data constitutes it [55-56]. Neo4j enables 2 million read/per second for the relationship. Calculations of shortest-paths scale far better than RDBMS [57]. Neo4j outperforms relational data stores with greater than 1000 times performance gain in many examples of deep query analytics [58].
- Neo4j exhibits support for ACID transaction properties that facilitates rollbacks and recovery from transactional failures [57].
- No O/R (object-relational) Mismatch: Neo4j naturally maps a graph structure to some Object Oriented language such as Java or Ruby and hence, does not need any complex O/R mapping tool [58].

ii. Disadvantages of Neo4j

- Single Point Failure: Neo4j has a Master-Slave model for replication as depicted in Figure 10(a) and 10(b) where all write operations are handled by the master and changes performed are reflected to the read only slaves. At master level, there can be a single point failure [57].
- Slow Online Write Transaction Speed: While committing in Neo4j, data is made permanent on disk that requires disk writes at each commit hence write speed is limited by the single server hardware's I/O capacity.


Neo4j Computing Environment
Master Server
4a
4n
2
3
Slave Server-1
Slave Server-2
Slave Server-N
1
Steps:
1. Write_To_Slave-2
2. Request_To_Confirm_Write
3. Confirm_Write
4. Replication_To_Other_Slaves
4a. Replication_To_Slave-1
4n. Replication_To_Slave-n
Neo4j Computing Environment
Master Server
4a
4n
2
3
Slave Server-1
Slave Server-2
Slave Server-N
1
Steps:
1. Write_To_Slave-2
2. Request_To_Confirm_Write
3. Confirm_Write
4. Replication_To_Other_Slaves
4a. Replication_To_Slave-1
4n. Replication_To_Slave-n

(a) When Master is Written (b) When a Slave is Written

Figure 10. Replication Model of Neo4j [64]

2.2.4. Apache Cassandra

Apache Cassandra is basically an open source column store peer to peer architecture distributed database [61]. It provides high end scalability and effective data replication that facilitates the fault tolerant feature and high availability. Despite of complex administering and data management than some other NoSQL alternatives, it has outperformed many of them [59] with its tremendous capabilities such as near real time interaction with users and streaming data analytics, *etc.* Some of the features [60-62] listed below.

i. Advantages of Apache Cassandra

- High Scalability: Cassandra provides two kinds of scalability. One is the data scalability. Second is throughput/performance scalability that enables response times in sub-seconds that scales linearly *i.e.* (two nodes double the throughput, four quadruple the throughput, and so on) [60].
- Very High Throughput for Write Operations: Provides very high throughput for write operations whereas considerably well throughput for read operations too [61]. Cassandra performs 'Per-Partition Ordering' specified while creating a table as sorting million rows is faster during development than sorting billions during production [63].
- Fault Tolerance and High Availability: It is achieved by the absence of single point failure [60] since it does not work on master/slave architecture. Data replication is done to several nodes in the data cluster centers [60], which improve the availability.
- Transaction Support: It delivers the “AID” (Atomicity, Isolation, Durability) through 'commit logs' to track each write to the database. It achieves durability through built-in redundancies [61].
- It serves streaming data analysis required in several areas like social media, stock trading, energy systems, healthcare systems, multimedia streaming systems *etc.* Also, through integration with DataStax Enterprise it achieves built-in data security *i.e.* authentication, data encryption, *etc.* [61].

ii. Disadvantages of Apache Cassandra

- A single column value may not be larger than 2GB; in practice. The number of cells allowed per row in one partition is maximum two billion [62-63].

- No join or sub query support: It does not offer join or sub query but very limited support for aggregation [63]. Join operations are implemented in the program which are expensive tasks in huge data sets [72].
- There is no in-built searching support in Cassandra architecture's. However, it supports for secondary indexes create them automatically and users must understand data model to create indexes in the absence of automatic secondary index support [72].

3. Results and Discussion

The overall management of Big Data involves storing, processing and analyzing it for various purposes, hence we can visualize the infrastructure, to handle Big Data related tasks, as a layered architecture as shown in Figure 11.

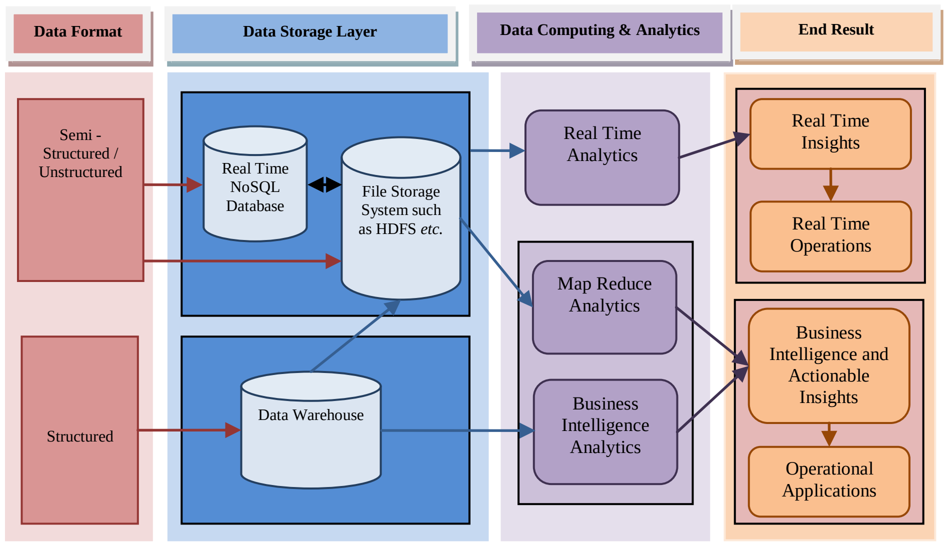


Figure 11. Layered Architecture for Big Data Handling

Through the detail analysis of various computing and storage tools, we have found several attributes that may give us a way to compare these tools. The various advantages/disadvantages of these tools let us know the suitability of various tools in various kinds of application domains.

3.1. Comparison of Computing Tools

Below is comparison Table 1, Table 2 and Table 3, consisting of various computing tools and the key features or facilities they support.

Table 1. Computing Tools Comparison Table

Computing Tools	Scala-bility	Distributed Architecture	Parallel Computation	Fault Tolerance	Single Point Failure
Hadoop	Yes	Yes	Yes	High	Yes- At master nodes)
Cloudera Impala RTQ	Yes	Yes	Yes	Yes	Yes - If any host quits query execution entire query is stopped

IBM Netezza	Yes	Yes	Yes- Asymmetric Massively Parallel Processing	Yes- Using redundant SMP hosts	At SMP server level
Apache Giraph	Yes	Yes	Yes - Bulk Parallel processing	Yes - by check points	No - Multiple master threads running

Table 2. Computing Tools Comparison Table

Computing Tools	Query Speed	Real-Time Analytics / Response Time	Streaming Query Support	ETL Required?	Data Format for Analytics
Hadoop	Slow	No	No	No	Structured/ Unstructured
Cloudera Impala RTQ	High	Yes / in seconds	No	No	Structured /Unstructured
IBM Netezza	High	Yes / in seconds	Yes	No	Structured (RDBMS)
Apache Giraph	High	Yes / very less	No	No	Graph Database

Table 3. Computing Tools Comparison Table

Computing Tools / Paradigm	I/O Optimization	Optimized Query Plans?	Efficiency	Latency Time for Query
Hadoop MapReduce	No	Not Applicable	Low for real time applications (High for batch processing)	Not Applicable
Cloudera Impala RTQ	Yes	Yes	Higher	Low Latency due to use of dedicated distributed query engine
IBM Netezza	Not Required	Yes	High	Very Low- in seconds, due to in-memory data base processing and parallelism
Apache Giraph	Not Required	Yes – In terms of graph query/ algorithms	High	Low - In memory computation

Based on the comparison Table 1, Table 2 and Table 3, we identified following set of categories on which we would like to evaluate the above tools and computing paradigm in subsequent sub-sections:

3.1.1. Distributed Computation, Scalability and Parallel Computation

As we can see from the comparison tables, all computing tools provide these facilities. Hadoop distributes data as well as computing via transferring it to various storage nodes. Also, it linearly scales by adding a number of nodes to computing clusters but shows a single point failure. Cloudera Impala also quits execution of the entire query if a single part of it stops. IBM Netezza and Apache Giraph whereas does not have single point failure. In terms of parallel computation IBM Netezza is fastest due to hardware built parallelism.

3.1.2. Real Time Query, Query Speed, Latency Time

The Hadoop employs MapReduce paradigm of computing which targets batch-job processing. It does not directly support the real time query execution *i.e* OLTP. Hadoop can be integrated with Apache Hive that supports HiveQL query language which supports query firing, but still not provide OLTP tasks (such as updates and deletion at row level) and has late response time (in minutes) due to absence of pipeline parallelism and run-time scheduling of task assignment to distributed nodes.

All other three computing tools support the real time query execution very well and have early response time in seconds. However, Cloudera executes queries at least 10 times faster than Hive/MapReduce. Hadoop has comparatively higher latency time as it targets batch-job processing. The Cloudera Impala has low latency time as it uses a dedicated distributed engine to access data. IBM Netezza and Apache Giraph also achieve very low latency time due to in-memory database processing and computation. Apart from these tools there are other frameworks that are dedicated only to big data stream computing and mining [65-66] for supporting real time analytics too but they are not discussed in this paper.

3.1.3. I/O and Query Optimization, Efficiency & Performance

Hadoop does not generate optimized query execution plans thus offers low efficiency for queries whereas Cloudera, IBM Netezza and Giraph have provision of I/O and query execution plans optimizations which results in higher efficiency and high performance in query execution. In Cloudera, purely I/O bound queries achieve approximately 3-4 times, queries of join or multiple MapReduces achieves approximately 7-45 and simple aggregation queries achieve 20-90 times performance gain over Hive/MapReduce. Giraph also provides high performance in terms of large scale graph processing for even trillion of edges.

3.1.4. ETL Requirement, Cost Effectiveness, Fault Tolerance

Since Hadoop, Giraph and Cloudera RTQ are open sourced, hence are a cost effective solution whereas IBM Netezza is proprietary to IBM, hence a costly solution for handling BigData. Also, since Clouera and Giraph perform in memory computation they do not require data input and data output that saves a lot of processing cost involved in I/O. None of the tools require the ETL (Extract, Transform and Load) service, thereby they save a major cost involved in data preprocessing. Hadoop is highly fault tolerant that is achieved by maintaining multiple replicas of data sets, and its architecture that facilitates dealing with frequent hardware malfunctions. Giraph achieves fault tolerance using barrier checkpoints.

3.1.5. Data Format, Language Support and Application Development

Hadoop HDFS is purpose built for supporting multi-structure data unlike the relational data bases whereas IBM Netezza deals strictly with the relational database. The Cloudera Impala RTQ supports both structured as well as unstructured data store. Apache Giraph is designed specially to work on graph data base such as Neo4j. Hadoop itself work on simply MapReduce paradigm but may support a range of languages for application development when integrated with other technologies such as Apache PIG that supports Python, Javascript and JRuby languages. Cloudera can be successfully integrated with various BI tools supporting various languages. IBM Netezza directly supports a wide range of languages (C, C++, Fortran, Java, Lua, Perl, PythonR) for application development. Apache Giraph builds applications implemented using Java libraries.

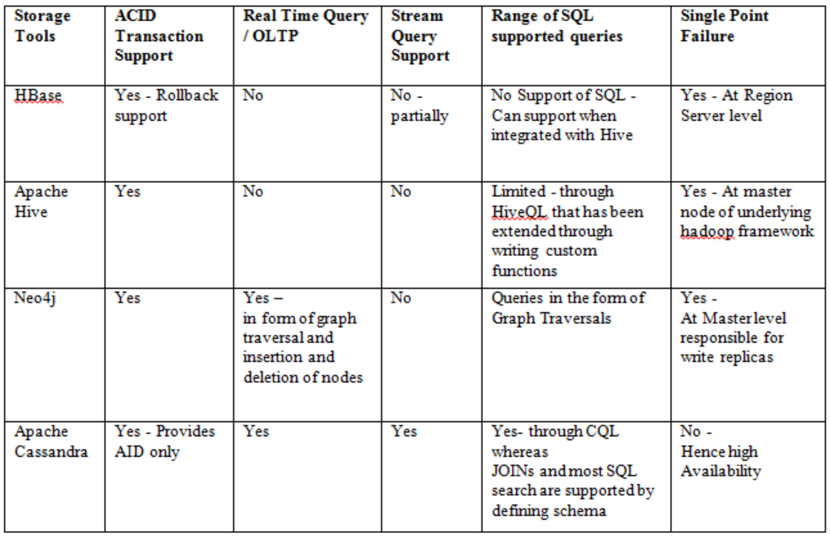
3.2. Comparison of Storage Paradigms/Tools

Below is comparison Table 4, Table 5 and Table 6, consisting of various storage tools and the key features or facilities they support.

Table 4. Storage Tools Comparison Table

Storage Tools	Open Source	Distributed	Scalable	Data Storage Format	ETL Required?
HBase	Yes	Yes	Yes	Structured *i.e.* Tabular but not exactly row-oriented Relational Table	Yes
Apache Hive	Yes	Yes	Yes - Good	Structured/ Unstructured	Yes - Hence a bit higher latency in minutes
Neo4j	Yes	Yes	Yes	Non-relational *i.e.* graph database (schema less)	No
Apache Cassandra	Yes	Yes	Yes -vast	Structured / Semi-structured / unstructured (schema less)	No

Table 5. Storage Tools Comparison Table



Storage Tools	ACID Transaction Support	Real Time Query / OLTP	Stream Query Support	Range of SQL supported queries	Single Point Failure
HBase	Yes - Rollback support	No	No - partially	No Support of SQL - Can support when integrated with Hive	Yes - At Region Server level
Apache Hive	Yes	No	No	Limited - through HiveQL that has been extended through writing custom functions	Yes - At master node of underlying hadoop framework
Neo4j	Yes	Yes – in form of graph traversal and insertion and deletion of nodes	No	Queries in the form of Graph Traversals	Yes - At Master level responsible for write replicas
Apache Cassandra	Yes - Provides AID only	Yes	Yes	Yes- through CQL whereas JOINs and most SQL search are supported by defining schema	No - Hence high Availability

Table 6. Storage Tools Comparison Table

Storage Tools	**Failover Recovery**	**Fault Tolerance**	**Meta Data Store**	**Language Interface Support**	**Access Control**
HBase	Long time at node level failure whereas 10 to 15 minutes at Region Server level	Yes	Yes	Less - Java Centric, Non-java clients are supported through REST and Thrift gateways	Yes
Apache Hive	Yes - supports node level recovery	Yes - replication mechanism to have synced with metastore	Yes	Clojure, Go, Groovy, Java JavaScript, Perl, PHP, Python, JRuby, Scala	No in-built security provisions
Neo4j	Yes - Select the new master	Yes - Supported by ACID Transaction system	Yes - Optional schema	Java, PHP, .Net, Python, Clojure, Ruby, Scala, *etc.*	Yes
Apache Cassandra	Yes - Optimized for the Recovery performance	Yes - Optional	Yes -Due to flexible schema support	Java, Python, Node.JS, *etc.*	Yes - Provided by the DataStax Enterprise

Based on the comparison tables, storage tools can be categorized and evaluated based on following subsets of characteristics that provides some insights of applicability of various tools in different application domains:

3.2.1. Distributed, Scalability and Data Format Flexibility

All storage tools provide distributed data storage and querying facility and scalable in nature. HBase can be easily scaled-up with new records up to millions of rows and billions of columns. HBase and Hive run on Hadoop data node clusters, hence exploits its scalable property to further expand the database through data partitioning over multiple cluster data nodes. Neo4j is also scaled up by simply adding new nodes if required and can model 232 billion nodes. Neo4j supports scalability in terms of parallel readings on multiple nodes. Cassandra is highly scalable NoSQL database whose throughput and query response scales linearly with machine nodes.

HBase has tabular data structure format, but it is wide-column and key-value-based data store capable of supporting a huge number of columns and flexible schema architecture. Hive also supports the unstructured database whereas Cassandra supports a full range of structured and unstructured data formats and the dynamic changes in data structure can be accommodated easily. Neo4j is extremely flexible schema less database solutions that solves graph modeled problems.

3.2.2. Availability, Fault Tolerance, Fault Recovery

The HBase and Hive run on Hadoops master/slave architecture of nodes that cause a single point failure at the master level failure. In HBase, RegionServer that manages the partitioned data into a cluster region becomes single-point-failure. Similarly, Neo4j's write-master is single point failure. All these are fault tolerant but HBase and Hive offers a bit low availability. Neo4j has a much better availability whereas Cassandra is highly available due to the absence of master/slave paradigm.

3.2.3. Real Time and Streaming Query Support, Query Performance

HBase is optimized for read operations and hence not much efficient for writes. On the other hand, Hive does not suit for OLTP due to absence of row level updates and deletes. Neo4j supports real time queries, but in the form of graph traversals. Cassandra supports OLTP very well on a full range of data formats. For stream query analysis, Cassandra is the best solution. The HBase stores data in MapFiles (that are indexed Sequence Files) thus becomes a suitable choice for streaming analysis of a MapReduce kind of style that involves occasional random look ups. The performance of query in Hive is increased through meta-store, data partitioning and external level table support that is not required to be pushed on HDFS. Neo4j overcomes the performance degradation problem in traditional RDBMS queries with several joins because a graph traversal is performed which works at the same speed, no matter how much data constitutes it. Cassandra provides very high throughput for write operation queries.

3.2.4. Open Source, Access Control, Language Support

All storage tools discussed in this paper are open source, hence free available and cost effective. Out of them, Cassandra and Neo4j does not have a requirement of ETL services that causes no extra processing cost overhead and become the cheapest choices among them. HBase, Neo4j and Apache Cassandra fully support an access control mechanism that provides security, authorized access and modification to the database whereas Hive does not provide such efficient control. HBase is java centric hence directly supports lesser languages but through REST and Thrift Gateways interface support languages. Neo4j supports several languages through various Neo4j language clients and REST APIs whereas Cassandra supports all key languages that are required to develop a variety of applications without need of any intermediate gateways.

4. Conclusion

This survey aims to find some available computing and storage paradigms and tools that are being used in current scenario to address challenges of Big Data processing. We have categorized the survey into two streams. One stream contains study and survey of existing computing paradigms and tools used to perform computation on Big Data and the other stream gives a detailed survey of storage mechanisms and tools available today. In this reference, we focused on Apache Hadoop, Cloudera Impala and Enterprise RTQ, IBM Netezza and Apache Giraph as computing tools and HBase, Hive, Neo4j and Apache Cassandra as storage tools. Based on deep and detailed analysis of their features, relative advantages and disadvantages we have made a critical comparison among these tools. The comparison is made on the most striking attributes that one looks for before choosing these tools for its application domain to handle Big Data. We have discussed various issues associated with various tools and compared them accordingly and gave critical review on the suitability and applicability of different storage and computing tools with respect to a variety of situations, domains, users and requirements. We found that Hadoop is an economic choice in many ways, but if some company or enterprise has no issue with spending money at all then high-end IBM Netezza AMPP is a better choice. Also, the world wide adoption of Hadoop has caused significant rise in the NoSQL databases that could be easily integrated with Hadoop. In this reference the HBase has supports a wide range of a community of users, multiple commercial vendors and developers and provision for cloud storage through Amazon Web Services (AWS). Also, it has shown a strong integration with Hadoop using Apache Hive. Due to strong consistency and easier application development, it is a good choice from developer point of view. Due to very high latency (in minutes), Hive is not a good solution for real time query applications and/or OLTP applications that require frequent write operations. Graphs are best suited for modeling real world situations such as computer networks, social networks, geographic pathways that calculate the shortest paths in graphs, *etc.* Hence, Neo4j and Giraph are the best choices for storage and computation respectively, to model such vertex-edge scenarios.

References

[1] S. Agarwal, Divya and G. N. Pandey, "SVM based context awareness using body area sensor network for pervasive healthcare monitoring", IITM, ACM, New York, **(2010)**, pp. 271-278.

[2] M. R. Wigan and R. Clarke, "Big Data's Big Unintended Consequences", IEEE Computer Society, DOI:http://dx.doi.org/10.1109/MC.2013.195, vol. 46, no. 6, **(2013)**, pp. 46-53.

[3] M. Kendrick, "Big Data, Big Challenges, Big Opportunities: 2012 IOUG Big Data Strategies Survey", http://www.ioug.org/p/cm/ld/fid=91, (Retrieved on September 2, 2015), **(2012)**.

[4] N. Wallis, "Big Data in Canada: Challenging Complacency for Competitive Advantage", in: T. White (Eds.), Hadoop: The Definitive Guide, third ed., O'Reilly Media, Yahoo Press, **(2012)**.

[5] J. Constine, "How Big Is Facebooks Data? 2.5 Billion Pieces Of Content And 500+Terabytes Ingested Every Day", http://techcrunch.com/2012/08/22/how-big-is-facebooks-data-2-5-billion-pieces-of-content-and-500-terabytes-ingested-every-day, (Retrieved on September 2, 2015), **(2012)**.

[6] D. Tam, "Facebook processes more than 500 TB of data daily", http://news.cnet.com/ 8301-1023_3-57498531-93/facebook-processes-more-than-500-tb-of-data-daily, (Retrieved on September 3, 2015), August **(2012)**.

[7] IBM, "What is big data?" http://www-01.ibm.com/software/data/bigdata/what-is-big-data.html, (Retrieved on September 3, 2015), **(2013)**.

[8] D. Tomar and S. Agarwal, "Predictive Model for diabetic patients using Hybrid Twin Support Vector Machine", Proceedings of 5th International Conferences on advances in communication Network and Computing, **(2014)**.

[9] B. R. Prasad and S. Agarwal, "Modeling risk prediction of diabetes—A preventive measure", Proceedings of 9th IEEE International Conference on Industrial and Information Systems (ICIIS' 14), **(2014)**, pp. 1-6.

[10] D. Tomar, B. R. Prasad and S. Agarwal, "An efficient Parkinson disease diagnosis system based on Least Squares Twin Support Vector Machine and Particle Swarm Optimization", Proceedings of 9th IEEE International Conference on Industrial and Information Systems, **(2014)**, pp. 1-6.

[11] D. Tomar, and S. Agarwal, "A survey on Data Mining approaches for Healthcare", International Journal of Bio-Science and Bio-Technology, vol. 5, no. 5, **(2013)**, pp. 241-266.

[12] S. Agarwal, Divya and Siddhant, "Prediction of Software Defects using Twin Support Vector Machine", Proceedings of 2nd IEEE International conference on Information Systems & computer Networks (ISCON), **(2014)**, pp. 128-132.

[13] J. Venner, "Pro Hadoop", a press, **(2009)**.

[14] T. White," Hadoop: The Definitive Guide", third ed., O'Reilly Media, Yahoo Press, **(2012)**.

[15] S. Ketu, B. R. Prasad and S. Agarwal, "Effect of Corpus Size Selection on Performance of Map-Reduce Based Distributed K-Means for Big Textual Data Clustering", In Proceedings of the Sixth International Conference on Computer and Communication Technology 2015, pp. 256-260. ACM, **(2015)**.

[16] W. Tantisiriroj, S. Patil and G Gibson, "Data-intensive File Systems for Internet Services", A Rose by Any Other Name (CMU-PDL-08-114). Research Centers and Institutes at Research Showcase, http://repository.cmu.edu/pdl/9. Technical report, Carnegie Mellon University, **(2008)**.

[17] M. K. McKusick and S. Quinlan, "GFS: Evolution on Fast-forward", ACM Queue, New York, vol. 7, no. 7, **(2009)**.

[18] K. Shvachko, H. Kuang, S. Radia and R Chansler, "The Hadoop Distributed File System", Proceedings of IEEE Conference, 978-1-4244-7153-9/10, **(2010)**.

[19] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters", commun. ACM, vol. 51, no. 1, **(2008)**, pp. 107–113.

[20] J. Dean and S. Ghemawat, "Mapreduce: A flexible data processing tool", commun. ACM, vol. 53, no. 1, **(2010)**, pp. 72–77.

[21] J. Shafer, S. Rixner and A. L. Cox, "The Hadoop Distributed File system: Balancing Portability and Performance", Proceedings of IEEE Conference, 978-1-4244-6022-9/10, **(2010)**.

[22] J. Xie, S. Yin, X. Ruan, Z. Ding, Y. Tian, J. Majors, A. Manzanares and X. Qin, "Improving MapReduce Performance through Data Placement in Heterogeneous Hadoop Clusters, in: Parallel & Distributed Processing", Workshops and PhD Forum, IEEE, 978-1-4244-6534-7/10, **(2010)**.

[23] T. Gunarathne, T. Wu, J. Qiu and G. Fox, “MapReduce in the Clouds for Science”, Proceedings of 2nd IEEE International Conference on Cloud Computing Technology and Science. DOI:http://dx.doi.org/10.1109/CloudCom.2010.107, **(2010)**.
[24] R. Chansler, H. Kuang, S. Radia, K. Shvachko and S. Srinivas, “The Hadoop Distributed File System”, http://www.aosabook.org/en/hdfs.html, (Retrieved on September 4, 2015), **(2014)**.
[25] Amazon, Amazon EMR, http://aws.amazon.com/elasticmapreduce, (Retrieved on September 6, 2015), **(2010)**.
[26] H. Liu and D. Orban, “Cloud MapReduce: a MapReduce Implementation on top of a Cloud Operating System”, Proceedings of IEEE International Symposium on Cluster Computing and the Grid, Newport Beach, CA, **(2011)**.
[27] A. Pavlo, E. Paulson, A. Rasin, D. J. Abadi, D. J. DeWitt, S. Madden and M. Stonebraker, “A comparison of approaches to large-scale data analysis”, Proceedings of ACM SIGMOD International Conference on Management of data, ACM, **(2009)**, pp. 165–178.
[28] V. Kumar and Vavilapalli, “Apache Hadoop YARN: Yet Another Resource Negotiator”, Proceedings of ACM Symposium on Cloud Computing, ACM, 978-1-4503-2428-1, **(2013)**.
[29] M. Kornacker, A. Behm, V. Bittorf, T. Bobrovytsky, C. Ching, A. Choi and J. Erickson, “Impala: A modern, open-source SQL engine for Hadoop”, Proceedings of the Conference on Innovative Data Systems Research, **(2015)**.
[30] J. Russell, “Cloudera Impala”, O'Reilly Media, Inc., **(2013)**.
[31] P. Francisco, “The Netezza data appliance architecture: A platform for high performance data warehousing and analytics”, IBM Redbooks, **(2011)**.
[32] L. Dignan, “Netezza's Twin Fin fuels profit surge”, ZDNet Blog, http://www.zdnet.com/ blog/btl/netezzas-twinfin-fuels-profit-surge/38539, (Retrieved on September 13, 2015), **(2010)**.
[33] Large Scale Data Management Experts, “Concurrency & Workload Management in Netezza”, in: Winter Corporation White Paper, WINTER CORPORATION, Cambridge MA, **(2009)**.
[34] E. Lai, “Netezza launches Skimmer data appliance, teases two more”, http://www. computerworld.com/s/article/9147719/Netezza_launches_Skimmer_data, (Retrieved on September 15, 2015), **(2010)**.
[35] M. Singh and B. Leonhardi, “Introduction to the ibm netezza warehouse appliance”, Proceedings of the 2011 Conference of the Center for Advanced Studies on Collaborative Research, IBM Corp., **(2011)**, pp. 385-386.
[36] T. Salomie, I. E. Subasu, J. Giceva and G. Alonso, “Database engines on multicores, why parallelize when you can distribute?", Proceedings of the 6th conference on Computer systems, ACM, **(2011)**, pp. 17-30.
[37] IBM-Corporation, IBM Netezza Analytics Release Notes, http://delivery04.dhe.ibm.com/sar/ CMA/IMA/032ig/0/IBM_Netezza_Analytics_Release_Notes.pdf., Release 1.2.4, (Retrieved on September 20, 2015), **(2011)**.
[38] A. Ching and K. Christian, “Apache Giraph”, **(2013)**.
[39] G. Malewicz, M. H. Austern, A. J. C. Bik, J. C. Dehnert, I. Horn, N. Leiser and G. Czajkowski, Pregel, “A system for large scale graph processing”, Proceedings of ACM SIGMOD International Conference on Management of data, ACM, New York, NY, USA, **(2010)**, pp.135-146.
[40] L. G. Valiant, “A bridging model for parallel computation”, commun. ACM, vol. 33, no. 8, **(1990)**, pp. 103-111.
[41] Y. Tian, A. Balmin, S. A. Corsten, S. Tatikonda and J. McPherson, “From think like a vertex to think like a Graph”, Proceedings of the VLDB Endowment, vol. 7, no. 3, **(2013)**, pp. 193-204.
[42] “Apache Giraph, Introduction to Giraph”, http://giraph. apache.org/intro.html, (Retrieved on Sep 22, 2015), **(2014)**.
[43] C. Martella, “Apache Giraph: Distributed Graph Processing in the Cloud”, FOSDEM, **(2012)**.
[44] H. Kristen, “An Introduction to Apache Giraph”, (Safari books online), http://blog. safaribooksonline.com/2014/02/10/intro-apache-giraph, (Retrieved on September 23, 2015), **(2014)**.
[45] A. Marcus, “The NoSQL Ecosystem”, the Architecture of Open Source Applications, **(2011)**, pp. 185-205.
[46] L. George, “HBase: The Definitive Guide”, first ed., O'Reilly Media, 9781449396107, **(2011)**.
[47] C. Li, “Transforming relational database into HBase: A case study”, Proceedings of IEEE International Conference on Software Engineering and Service Sciences (ICSESS), **(2010)**, pp. 683,687.
[48] “Apache HBase, Welcome to Apache HBase”, https://hbase.apache.org/index.html, (Retrieved on Sep 26, 2015), **(2013)**.
[49] M. N. Vora, “Hadoop-HBase for large-scale data”, Proceedings of International Conference on Computer Science and Network Technology (ICCSNT), IEEE, vol. 1, **(2011)**, pp. 601-605.
[50] L. Francke, “Hive HBase Integration”, https://cwiki.apache.org/confluence/display/Hive/HBase Integration, (Retrieved on September 27, 2015), **(2012)**.
[51] T. White, “The Small Files Problem”, http://blog.cloudera.com/blog/2009/02/the-small-files-problem, (Retrieved on September 27, 2015), **(2009)**.

[52] H. Doug, “Big Data Debate: Will HBase Dominate NoSQL?”, http://www.informationweek.com/big-data/software-platforms/big-data-debate-will-hbase-dominate-nosql/d/d-id/1111048?, (Retrieved on Sep 28, 2015), **(2013)**.
[53] L. Francke, “Hive HBase Integration”, https://cwiki.apache.org/confluence/display/Hive, (Retrieved on September 28, 2015), **(2012)**.
[54] B. Singhvi, “Apache Hive Review”, http://www.gise.cse.iitb.ac.in/wiki/images/2/26/Hive.pdf, (Retrieved on September 28, 2015), **(2012)**.
[55] “Neo Technology, Advantages of a Graph Database”, http://neo4j.rubyforge.org/guides/why_graph_db.html, (Retrieved on September 29, 2015), **(2014)**.
[56] M. Hunger, “Neo4j: Java-based NoSQL Graph Database”, http://www.infoq.com/news/2010 /02/neo4j-10, (Retrieved on September 29, 2015), **(2010)**.
[57] R. Sasirekha, “Neo4j, the Graph Database for high performance traversals”, http://itknowledgeexchange.techtarget.com/enterprise-IT-tech-trends/neo4j-the-graph-database, (Retrieved on September 30, 2015), **(2010)**.
[58] E. Eifrem, “Neo4j - The Benefits of Graph Databases”, http://www.oscon.com/oscon2009/public/schedule/detail, (Retrieved on September 30, 2015), **(2009)**.
[59] K. Jeff, “Cassandra Continues to Win Real-Time Big Data Converts”, http://wikibon.org/wiki/v/Cassandra_Continues_to_Win_Real-Time_Big_Data_Converts, (Retrieved on September 31, 2015), **(2012)**.
[60] A. Lakshman and P. Malik, “Welcome to Apache Cassandra”, http://cassandra.apache.org, (Retrieved on September 30, 2015), **(2011)**.
[61] DataStax Corporation, “Introduction to Cassandra -A White Paper”, **(2013)**.
[62] T. B. George and C. Bucur, “A comparison between several NoSQL databases with comments and notes”, Proceedings of 10th IEEE Roedunet International Conference (RoEduNet), **(2011)**, pp. 1-5.
[63] G. Kunz, “Cassandra Limitations”, http://wiki.apache.org/cassandra/CassandraLimitations, (Retrieved on September 1, 2015), **(2013)**.
[64] J. Webber, http://jimwebber.org/2011/02/scaling-neo4j-with-cache-sharding-and-neo4j-ha (Retrieved on September 20, 2015), **(2011)**.
[65] B. R. Prasad and S. Agarwal, “Handling Big Data Stream Analytics using SAMOA Framework - A Practical Experience”, Int. J. Database Theory and Application, vol. 7, no. 4, **(2014)**, pp. 197-208, **(2011)**.
[66] A. Murdopo, A. Severien, G. D. F. Morales and A. Bifet, “SAMOA: Developer's Guide”, Yahoo Labs, **(2013)**.

Authors

Bakshi Rohit Prasad, He is a research scholar in Information Technology Division of Indian Institute of Information Technology (IIIT), Allahabad, India His primary research interests are Data Mining, Machine Learning, Big Data Computing and Algorithms along with their applications in several domains.

Sonali Agarwal, She is working as an Assistant Professor in the Information Technology Division of Indian Institute of Information Technology (IIIT), Allahabad, India. Her primary research interests are in the areas of Data Mining, Data Warehousing, E-Governance and Software Engineering. Her current focus in the last few years is on the research issues in Big Data Computing and its application.