Journal of Multidisciplinary Engineering Science and Technology (JMEST)
ISSN: 3159-0040
Vol. 3 Issue 1, January - 2016

Adaptive Tools And Technology In Big Data Analytics

Reza Shokri Kalan, İlker Kocabaş
International Computer Institute
Ege University
İzmir, Turkey
reza.shokri@hotmail.com, ilker.kocabas@ege.edu.tr

***Abstract*—In 2015, number of people plugging to Internet rise to 3 billion. Beside human activity in social networks, billions of connected devices, sensor and instrument uninterrupted make amount of data. Irrespective of how data created, a natural question comes to mind is: How we could manage data analytics and what sort of tools should be adapted for handling this task in effective way? This paper discusses approaches and environments about big data. It revolves around three important areas of big data tools and talent, namely (i) data management (ii) infrastructure and technology, and (iii) query performance. The goal of big data management is to ensure a high level of data quality and accessibility which strongly depend on data transforming and storing. Using index improve query performance, however, many traditional indexing approaches in big data trinity involve a significant upfront cost for index creation. The key contribution of this paper is that using right tools over right data, and providing effective index to minimize upfront cost.**

Keywords—Big Data, Analytic, Cloud computing, Hadoop, MapReduce, Adaptive Indexing

I. INTRODUCTION

In the Big data-Bang theory, online potential rapidly changes social activity and transforms our lives by migrating from paper base information to electronical media such as googling, facing, twitting and texting. In a competitive market place, smart organization try to understand customer behaviors and offer customized service by mining and exploring data collecting from different resources. Data mining and knowledge discovery aims to find hidden relation between different attribute [1,2]. Mining requires integrated, cleaned, trustworthy, and efficiently accessible data [3]. However, according to the nature of big data, analyzing and managing is more complicated. Data integration focuses on collecting and combining different forms of data and reshaping all of them in new platforms. Without doubt, well designed and smaller set of data can be analyzed quickly and productively. Therefore, big data integration (BDI) process requires careful data classification. However, integration huge volume of data saved in variety file format is a big challenge. Furthermore, data velocity increases this complexity. Performing analytics on large volumes of data requires efficient methods to store, filter, transform, and retrieve the data [4]. By the way big data management is labor task and need a specific computing algorithm and infrastructure. Cloud as bedrock of big data processing enables organizations to pay only for the resources and services they use. Cloud computing is an extremely successful paradigm of service oriented computing. Although offering resources in a pay-as-you-go fashion is more cost effective and improves availability and elasticity, but it brings new challenges. Some of this challenge and solutions have been addressed in [5,6,7,8]. Since everything around big data is changing very fast, big data is more complex than as a structured database and relation between components. To overcome this complexity and find value of mine this information we need new tools and talent. In this paper, we share effective solution in order to minimize I/O costs and utilize the available resources and parallelism of large clusters for indexing. We first review the big data state-of-the-art and introduce the general background related technologies, such as could computing. We then focus on big data indexing in relational database and MapReduce technology. For each system, we introduce the general background, discuss the technical challenges, and review the latest advances. These discussions aim to provide a comprehensive overview and big-picture to readers of this exciting area.

II. BACKGROUND

The popularity of big data brings new business fashion, therefore, all market place and industry sector have started big data management in effective and valuable manners. The concept of big data analytics is successful application in different business, science and social domain. Advanced of big data analysis offers a cost-effective opportunity in critical decision making area such as healthcare, security, economic, crime, natural disaster and recourse management [9,10]. Big data is more complex ecosystem with hidden relation and unstructured data which is growing rapidly. In fact big data are a revolution in different domain. Therefore, the big data solution must be able to offer deep analytic, high agility, massive scalability with low cost and latency [11].

As shown in Fig. 1. nature of big brings new challenge in data capturing, storing and access mechanism. The first impression of big data is its volume, so the accessibility of big data is on the top priority of the knowledge discovery process [12]. Cloud computing as a bedrock provide cost effective service for storing and processing big data comes from different resources. Although the emergence of cloud computing bring new enterprise opportunity, but still there is more obstacle in use of big data leverage such as integration, confidentially, and accessibility. Furthermore, the network bandwidth capacity is the bottleneck in the cloud and distributed systems, especially when the volume of communication is large [12]. Last but not least, cloud storage also leads to data security problems [13].

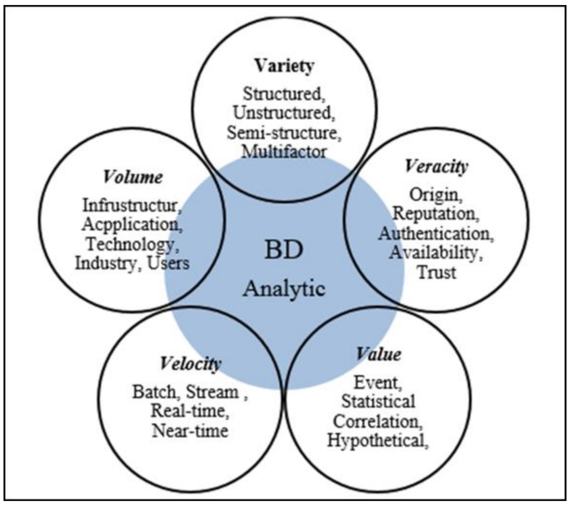

Variety
Structured, Unstructured, Semi-structure, Multifactor
Volume
Infrustructur, Acpplication, Technology, Industry, Users
Veracity
Origin, Reputation, Authentication, Availability, Trust
BD Analytic
Velocity
Batch, Stream , Real-time, Near-time
Value
Event, Statistical Correlation, Hypothetical,

Fig. 1. *Big data resources and nature*

Some of the most significant potential of big data comes from the bridge among resource and data streams. We can aggregate big data from three sources: traditional (structured data), sensory (log data, metadata), and social media (unstructured data). Instance, in the health care system, the outcome of combination multiple data comes from patient behavior, clinic, service and cost, medical research and the weather report is more desirable an cost effective. Regardless of how the data are created, data management focus on data structure and processing skills. Big data are often stored using non-relational DBMSs such as: Column based, key-value, graph, and document based.

A. *Big Data Definition and Properties*

There are many different stories about big data definition and properties. From the best knowledge of the authors, the common definition has three V's: (1) Volume, (2) Variety and (3) Velocity of data. Furthermore, big data processing lead to big (4) Value and finally the main aim of big data are (5) Veracity. While the first three V's are more technical and related to data engineering such as collection, summarizing, storage or transferring, the last two V's focused on data mining such as analytic, knowledge extraction and decision making. All of these factors covered under the complexity umbrella (Fig. 2). Organizations expect the value of these emerging techniques to soar. All the smartest organizations are embedding analytics to transform information into wisdom and then action. Organizations that know where they are in terms of analytics adoption are better prepared to turn challenges into opportunities.

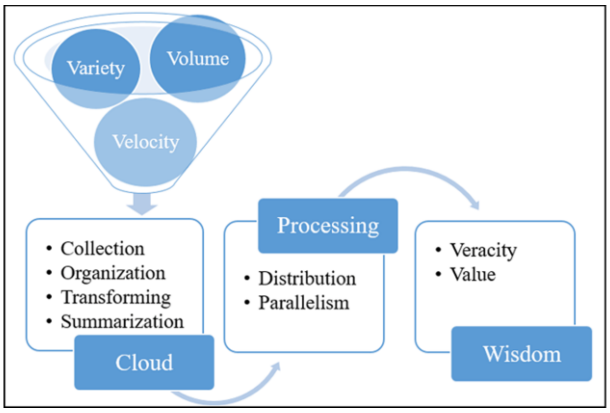


Fig. 2. *Big data life cycle and property*

B. *Big Data Infrastructure and Tools*

Big data processing requires secure and powerful computing capability. Multi core computer or multi clustering may overcome the processing lack. However, high performance computing is non-economic. The big data infrastructure must be able to support deeper and faster data mining even in distributed environments. Furthermore, data infrastructure must ensure data security and data ownership protection [14]. Because of local system limitation such as memory and CPU bottleneck, distributed system is used to store and processing big data. The cloud analytic solution can be good address to this problem, but there are still some challenges include privacy, scalability and query response time. Some of the solution such as Google file system (GFS) which are currently used in Cloud [7] attempted to provide the robustness, scalability and reliability that certain internet service need [8]. Amazon simple storage service (S3) and Windows Azure binary large object (Blob) storage are examples of object-store solution which tries by replicating file across multiple geographical sites improve redundancy, scalability, and data availability. Although these solutions provide the scalability and redundancy that many Cloud applications require, they sometimes do not meet the concurrency and performance needs of certain analytics applications [8]. Without doubt, distributed storage increase safety, but lead to redundancy problem. We will discuss about Cloud computing in next section.

C. *Big Data and Analytical Challenges*

Businesses seeking to increase profitability, improve customer retention, extend product lines and reduce risk through analytics are constrained by traditional data integration approaches that slow analytics adoption. These include three significant challenges:

- *Data agility and quality*- in dynamic businesses new data sources need to be brought on board quickly and existing sources modified to support each new analytic requirement. Furthermore, data virtualization eliminates many root causes of poor data quality.
- *Big data integration*- since data sources are more heterogeneous in content and structure and many of the data sources are very dynamic, big data integration (BDI) is the biggest bottleneck. A challenge with current big data analysis is the lack of coordination between database systems [9]. Traditional tools that extract, transform and load (ETL) are not more suitable for big data. It is possible to use built-in Hadoop tools such as Hive and Pig to extract, load, and transform data, rather than using traditional ETL tools. In the section four we will discuss about Apache Hadoop and explain that why Hadoop is not a good choose always.
- *Query performance*- since query performance is a critical success factor, most of big dataset are indexed to optimize the SQL query performance. Relational database management system (SQL) and developer-centric specialized systems (NoSQL) solutions are two classes of big data processing systems. SQL based system requires a good structure with defined attributes to hold the data in order to use effect index system. On the contrary, NoSQL databases which usually allow free-flow operations. NoSQL is a distributed, highly scalable, key-value databases provide very fast performance, and can rapidly store large numbers of transactions, but need a more complex query pattern which generally it is more difficult for end users. While index improves query performance *why general Hadoop does not use it?* And *why we need to use the MapReduce system which just expert know about*? We will answer this question in section five, by investigating advantage and disadvantage of using processing technique in both DBMSs and MapReduce systems such as Hadoop and HAIL.

III. Big Data analytics paradigms

It is proved there is no single system that would be most suitable for every need [8, 15]. Therefore, analytic paradigm can be mixed of three models:

- *Real-time:* In-memory, scale-out engines that provide low-latency,
- *Interactive:* Includes distributed MPP (massively parallel processing)
- *Machine learning*: HDFS (Hadoop distributed file system)

Such a design forward data processing toward enterprise data warehouse (EDW). This approach, compromised SQL and NoSQL data structure. As we know, data locality has impact effect in data processing. A machine learning approach based on MapReduce is new approach cover data locality problem. Hadoop, as open source Map reducer implementation use HDFS to partition and distribute big data over multi clusters. Also HDFS reduces the failure probability, by replicating data at least on three nodes. Hive-Hadoop is SQL friendly query language integrated multiple data sources.

A. *Big Data Resource Organization*

Since the volume of big data is giant, it need to more powerful computing, application and infrastructure capability rather than traditional systems. High performance computing, storage, network, and data centric service models are fundamental resources for big data. Disks are still the major bottleneck (I/O bandwidth) in query execution over large datasets even in parallel mode. As an Amdahl's Law, any new or hot data that must be accessed more frequently, is stored on faster and more expensive storage media, while less critical data is stored on slower, but cheaper media.

B. *Cost Effective, Scalable and Parallelizable System*

Every industry has its big data analytics under certain condition include (i) data volume and velocity, (ii) impacts of computing, storage, and network resources, and (iii) types of environments [16]. Cloud is complete software stack that support elasticity and provide restrict API to applications designed to run on top of a shared-nothing architecture with on demand scalability [5, 17]. Three most popular cloud paradigms include: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). The concept, however can also be extended to Data as a Service (DaaS) or Analytic as a Service (AaaS). In this section we discuss about opportunity and limitation of deploying big data on the Cloud platform. Firstly, we point to some characteristic of cloud computing and then show why current available systems are not ideally-suited for cloud deployment.

C. *Data Management in the Cloud*

Data locality is the key point of big data analysis application. Despite transferring small file to powerful computing system is the preferred option, however, this is not suitable for big data because of data volume. In such a case moving data to remote computing machine decreases efficiency due to transmission time. Since data transferring is more time costly, therefore data have to be stored as distributed manner. Thus, bringing processing unit near to data is more common and logical. Regardless of underlying infrastructure such as Cloud computing, correct tools and talent for big data mining has high priority. Any provided tools should assist customers to estimate the costs and risks of performing analytics on the Cloud.

D. Cloud Storage Locality

Nowadays, public clouds natively provide many kinds of big data analytics platforms and tools in order to speed up data analytics at an affordable cost [16]. In public clouds, customers rent and run services without exact information about the data storage location. As discussed before remote processing is not suitable for big data because of data availability. Access to data through large geographical distances is more time costly and increase response time. To address this challenge WAN optimization technology must maturing quickly to substantially reduce network latency while transmitting large amounts of data from one system to another among geographically distributed clouds.

E. Cloud Storage and Untrusted Host

In general, moving data off premises increases accessibility, confidentiality and privacy risks. Although, cloud provider loyal the privacy of its customers, but there is no 100% guarantee. Firstly, in multi talent system security risk increase, secondly, under some condition government restricts customer to access own data. Furthermore, national security allows the government to demand access to the data stored on any data storage. Without doubt, any security leakage or privacy violations in transactional databases that includes sensitive information like personal information or financial data is unacceptable. Although, data encryption increase data security, but it is more time costly. Therefore, deploying a transactional database and application on public cloud are not well suited.

IV. DATA STRUCTURES AND ARCHITECTURE

Usually selective queries that scans through the complete dataset are not efficient. This is same as looking for a needle in a haystack. Therefore, proposed approach must be far away from (i) high upfront cost which increase waiting time to run actually start querying and (ii) need for better knowledge of workload in order to choose the index to create-generally all users does not have this professionally.

A. SQL Data Modeling and Parallel DBMSs

All SQL friendly database system stores data in standard relational tables. In distributed form, all tables partitioned over the nodes in the cluster. In shared nothing architecture, the database system is deployed on multiple independent machines, each with local resources, connected together through high speed network. Data locality improves total performance by reducing the amount of data transferred over the network. In the application level user interact with the system though SQL commands and never face up with underlying technical detail. Then database system transforms and divides the SQL command and execute in distributed form. The DBMS system has a schema on write which means the user need to have knowledge about data structure.

B. Not only SQL Data Modeling

Relational modeling is typically driven by the structure of available data. NoSQL data modeling is typically driven by application-specific query patterns (what answer vs what question). Since structured database tries to eliminate duplicated data by using normalization, while aggregation and de-normalization (data duplication), cluster friendly, open source, no relational and schema-less are first-class attribute in NoSQL. In fact de-normalization, store data in query friendly manner by copying the same data into multiple documents or tables in order to simplify/optimize query processing or to fit the user's data into a particular data model [18]. It is helpful in I/O per query and processing complexity vs total data volume.

C. Hadoop and MapReduce

Hadoop works in parallel fashion which each cluster includes many nodes. The input data file distributed over all nodes in the cluster by HDFS underlying file system with at least three replication factor (Fig .3). Hadoop ecosystem is a good platform for unstructured data set. It can support the variety of components, including the HDFS, Pig, HBase (database/data store), and MapReduce (distributed sorting and hashing). Original Hadoop scans whole data set because it does not support any index. MapReduce is used for developing novel solutions on massive datasets such as web analytics, relational data analytics, machine learning, data mining, and real-time analytics. Hadoop MapReduce scales easily to very large clusters of thousands of machines. In addition, the upfront investment for using MapReduce is small: no need to use schemas, no integrity constraints, no normalization, and NoSQL. Although, HDFS provides high fault-tolerance capabilities by providing multiple replica. However, the performance of MapReduce in many cases is far from the one of an optimized structured DBMS [14].

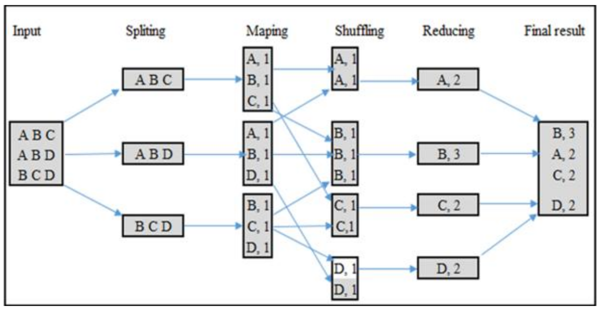


Fig. 3. *MapReduce Architecture*

V. Parallel data processing (MapReduce vs DBMS)

A database system implementation based on indexing, following three lessons [19]: (i) Schemas are good, (ii) Separation of the schema from the application is good, (iii) High-level access languages are good, but MapReduce has learned none of these lessons, it is a poor implementation based on brute force instead of indexing. The main question comes in mind if the database system is so good, why Hadoop popularity increase day to day? To answer this question we survey both systems specific architecture includes replication, indexing, complexity and flexibility.

A. *Pre-replication Layout*

A parallel processing is a good solution to big data processing. The parallelism increase performance by obtaining a high throughput for online transaction processing (OLTP) load, and low response time for online analytic processing (OLAP) to answering multi-dimensional analytical (MDA) queries. As mentioned before general Hadoop does not support any index. Since Hadoop uses three replicas, therefore, it is possible change HDFS to store each physical copy in a different data layout. This approach looks very promising, because it helpful in a lot in several query workloads, at the same time, I/O times improved [15]. Most indexes are too large to fit into memory, which means that they are going to be stored on disk. Since I/O is usually the most expensive thing you can do in a computer system, thus indexes need to be stored in an I/O efficient way.

Page attributes across (PAX) as a well-known layout solution significantly improves cache performance by grouping together all values of each attribute within each page [20]. Since the relational database system has traditionally optimized for I/O performance, PAX minimize storage penalty. Although PAX was originally invented for cache-conscious processing, but it has been adapted in the context of MapReduce [21]. Furthermore, data replication not only reduce index building time, but also increase scalability of the architecture and lead to more availability and reliability. Finally, sharing nothing architecture eliminate sharing either memory of the disk across nodes.

B. *Big Data Indexing*

While the query return similar result from database, indexing improves query response time. However, if the table is updated all the time, the overhead of the index management and re-organization may lead us to not choose the index. Every data entry of modification potentially involves updating the indexes, which can lead to slower performance of data updates. Regardless of negligible disadvantage of indexing, the main question is: when and where we need to create an index?

Indexes are the easiest way to improve the performance of long running queries with full table scans. Furthermore, using index improves replication response and network load. The major index type can be listed as row based and column based index. Since row index is more efficient in add/delete row, but read useless data. On the flip side, column index has efficient access to useful data, however inefficient in add/delete row. Therefore, columnar layout is better for big data, with lots of columns. However, the time needed to build a data series index becomes prohibitive as the data grows. Finding the most efficient index based on state-of-the-art indexing method for very large data is often hard. Major problems with big data indexing are:

- Scalability
- Expensive index creation
- High dimensional indexing
- Which attribute to create an index
- re-indexing the whole document, when indexes become corruptor or new features are added

However, the proposed solution for big data indexing must be covered:

- Size of index - Index size should be a fraction of the original data
- Speed of search- Search over billions – trillions data value in seconds
- Parallelism- should be easily partitioned into pieces for parallel processing
- Speed of index generation- index should be built at the rate of data generation
- Multi-variable queries- be efficient for combining results from individual variable search results

1) *Static and Dynamic Indexing:*

Low latency is the main advantage of static indexing. However, in lack of a suitable index creating new index is more time costly. Furthermore, inadaptability to changes in workloads without the intervention of a DBA, are big disadvantage of static index technique [22]. Since, we do not know the data access pattern, determining which attributes or criteria would be indexed during uploading data to HDFS is still problematic. Therefore, using any traditional indexing technique is non-promising, because they cannot adapt well to unknown or changing query workloads. To address this problem dynamic index was invented. The main idea behind this approach is to analyze query workload and make index based on statically observation.

2) Adaptive Indexing:

Since we try to index more data, the initialization time to build a state of-the-art data series index (Fig .4) becomes a prohibitive factor [23]. Adaptive index instead of building the complete index over the complete data set up-front and querying only later, interactively and adaptively build parts of the index, only for the parts of the data on which the users pose queries [15]. As more queries are posed, the index is continually refined and subsequent queries enjoy even better execution times. [23]. Adaptive indexing is an automatic process that is not explicitly requested by users and therefore should not unexpectedly impose significant performance penalties on users' jobs [22]. With the leverage of adaptive indexing, users can immediately start exploring the data series instead of waiting for valuable periods of time for the index creation.

This intuition is similar to soft indexes approach, where the index creation based on a single incoming query attribution [24]. The concept of adaptive indexing works with column-store databases [25]. The main idea is that, instead of building database indexes up-front, indexes are built during query processing, adapting to the workload [23, 25, 26]. However, lack of knowledge about future query is the dark side of this approach. Also, adaptive index focus on creating non-clustered indexes in the first place, and hence, it is only beneficial for highly selective queries. Furthermore, it is hard to apply existing adaptive indexing works in MapReduce systems because of (i) High I/O-costs: adaptive index considers main memory systems and thus do not factor in the I/O cost for reading/writing data from/to disk. (ii) Global index convergence: there are at least three physical data block replicas. (iii) Centralized approach: existing adaptive indexing approaches were mainly designed for single-node DBMSs.

3) Hadoop++ and Trojan Layout:

It is a buzz that Hadoop may suffer from performance issues when running analytical queries. Because, Hadoop MapReduce wastes a significant amount of time reading unnecessary data from disk as it stores data in row layout. Also, the original implementation of Hadoop does not provide index due to the lack of prior knowledge about schema and MapReduce jobs. Therefore, changing the data layout in Hadoop in order to be more suitable for analytical query processing is the biggest challenge. Unfortunately, Hadoop implements a hard-coded processing pipeline whose structure is very hard to change. Therefore, the overall goal of Hadoop++ project is to improve Hadoop's performance for analytical queries.

Hadoop++ project by focusing on user-defined functions (UDFs) such as map () and reduce () try to inject the new changes in the current framework. The idea behind Hadoop++ is, we can create appropriated index if we know schema and anticipate MapReduce jobs. Hadoop ++ use benefit of Trojan index such as optimal access path, and no more overhead for creating indexes. In this approach, all data in an HDFS block (i.e., horizontal partition of data of 64MB by default), is stored in a column layout yielding up the performance. This avoids the problems with network I/O for tuple reconstruction and still gives column-like access. Hadoop stores, three copies of an HDFS block for fault-tolerance, which all of them are byte-identical. By allowing the different copies of a logical HDFS block to have different physicals layouts, we are able to optimize the different copies for different types of queries. Furthermore, several indexes can be created on same split, but only one of them can be the primary index. In summary, Trojan layouts improves query run times both over row layouts and over PAX layouts without modifying the underlying Hadoop MapReduce and HDFS engines [26].

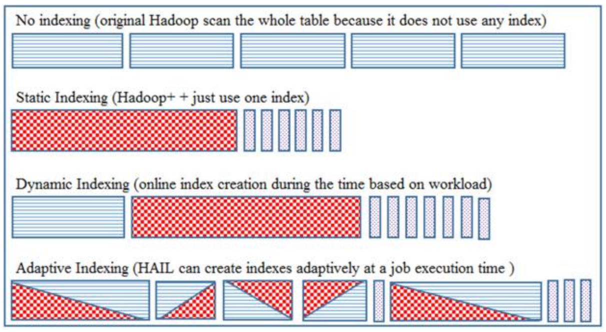

No indexing (original Hadoop scan the whole table because it does not use any index)
Static Indexing (Hadoop+ + just use one index)
Dynamic Indexing (online index creation during the time based on workload)
Adaptive Indexing (HAIL can create indexes adaptively at a job execution time)

Fig. 4. *Query pattern in different type of index*

4) Effective Cost Indexing with Zero Overhead:

Obviously the preferred index structure should have minimal overhead (e.g. Cheaper to create in main memory, cheaply to write to disk, and cheap to query from disk). A clustered index reorder the physical sort of data in table, therefore, each table can have only one clustered index. Since a non-clustered index is sorted separately from actual data, so a table can have more than one non-clustered index. Indexes use up disc space to store, and take time to create and maintain. Thus, as non-clustered indexes are dense by definition, they require more write I/O than a sparse clustered index and affect upload times [27].

Client queries run much faster on average, if indexes on the right attributes exist. By increasing the number of indices, the likelihood to find a suitable index is increasing too, but having multiple indexes is costly. Hadoop Aggressive Indexing Library (HAIL) uses the benefit of Hadoop replication factors. In HAIL, each data-node creates a different clustered index for each HDFS block replica and stores it with the sorted data [28]. HAIL can index HDFS blocks in parallel to job execution. In such a condition, the new job can already benefit from the previously indexed blocks.

Even if we did not create the right indexes at upload time, HAIL can create indexes adaptively at a job execution time without incurring high overhead [22].

C. *Flexibility and Complexity*

DBMSs database system support schema on write. This means that we need to have knowledge about database architecture when we want to write something into the database. Hadoop on the other hand has a schema on read approach which increase complexity when we want to read some data set. The good news is, both classes of system support some form of user defined function which improve flexibility. Furthermore, some of the management and analyzing tools such as Hive, Impala and Sqoop is now beginning to natively write MapReduce programs and pre-packaged schemas on read. Loading data in parallel DBMS is considerably longer, however process time is fast. On the flip side, MapReduce very suit to extract a task load. Last but not least, under some processing condition, there is no other opportunity more than using Hadoop while the main part of world information include log file, text and image are often much bigger than transaction data kept out of the parallel DBMS system. Despite of SQL adaption and popularity, DBMS systems are inefficient in big data set processing.

D. *Fault Tolerant*

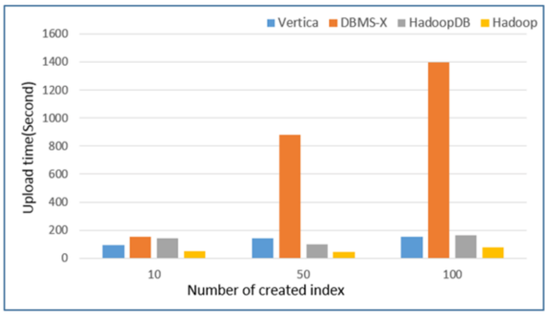
While both classes of systems use the same form of replication to deal with disk failures, architecture of the Hadoop frameworks provides a more sophisticated failure model than parallel DBMSs. Form the best knowledge of author Hadoop is more economical and scalable because of using commodity hardware.

VI. BENCHMARKS

Data loading and data processing are two main factors in data mining domain. Regardless of distribution overhead, it is adapted that parallel processing has better performance. In this section, first we will compare data loading performance in Hadoop and parallel DBMSs. Then we focus on indexing technique and compare effects of index in Hadoop, Hadoop++ and HAIL performance. Some previous works [29] compare Hadoop data loading performance over DBMS-X, HadoopDB and Vertica, which are a parallel row-store database system, a hybrid system that connects multiple single-node DBMS with MapReduce, parallel column-store database system consequently. The proposed solution report load times for two data sets (Grep and User_visits). While Grep data is randomly generated and requires no preprocessing, User_visits needs to be repartitioned by destination URL and indexed by visit Date for all databases during the load in order to achieve better performance on analytical queries (Hadoop would not benefit from such repartition).

A. *Data loading*

Obviously, most users want to start analyzing their data early. Low upload time is a crucial aspect for to adopt a parallel data-intensive system. In fact, low startup costs is one of the big advantages of standard Hadoop over DBMSs. As shown in Fig. 5 load time in the DBMS system dramatically increases while the number of nodes changes from 10 to 100. It is because DBMS are not well designed for task load. However, Vertica, which is a column-store database has good performance. Data loading depends on three main factors: (i) number of indexes, (ii) replication factor, (iii) cluster scale up and scale out.



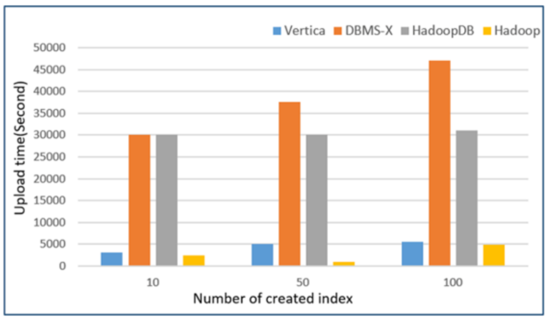


Fig. 5. *(a) Grep data load time (0.5 GB/node) [29]*
(b) User visits data load time (20GB/node) [29]

1) *Number of Indexes:*

Since Hadoop has good loading performance, then we compare it with two other projects. As discussed in section 4, general Hadoop has 0 indexes as it cannot create any index. Also Hadoop++ cannot create more than one index and finally HAIL support number of indexes. As shown in Fig .6. index creation is time costly and increase upload overhead. However, HAIL has a negligible upload overhead when creates one index per-replica. On the other hand, the result shows that HAIL improves over Hadoop++ by a factor of 5.1when creating no index and by a factor of 7.3 when creating one index [27].

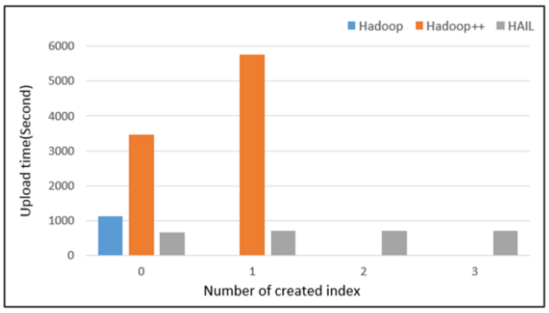


Fig. 6. *Upload times in three forms of replication datasets synthetic [27]*

2) *Number of Replica:*

As illustrated in Fig .7. we see that HAIL significantly out performs Hadoop for any replication factor. Still, when increasing the replication factor even further for HAIL, we see that HAIL has only a minor overhead over Hadoop with three replicas only. These results also show that choosing the replication factor mainly depends on the available disk space. Index space on disk is one of the main challenges in index creating. Even in this respect, HAIL improves over Hadoop. HAIL enables users to stress indexing to the extreme to speed up their query workloads.

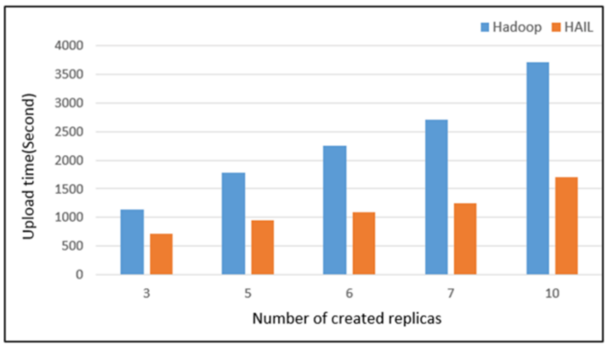


Fig. 7. *Upload times in varies form of replication datasets synthetic [27]*

3) *Resource scale:*

Resource and underlying structure has big effect on big data performance. Whenever the volume or velocity of data overwhelms current processing solution, resulting in performance that falls far short of desired [30]. Three forms of scale that help to improve performance are: (i) Scale down the amount of data processed or the resources needed to perform the processing and adjust in size relative to other things, (ii) Scale up the computing resources on a node, via parallel processing and redesigning new algorithm and data structure to take advanced of faster memory/storage technologies, and (iii) Scale out the computing to distributed nodes in a cluster/cloud or at the edge where the data resides.

VII. Conclusion

From a technical point of view, comparing parallel DBMS and Hadoop MapReduce is unfair. MapReduce is referred to as a new way of processing big data in cloud computing. Building a big data analytical under the umbrella of the MapReduce model requires great programming skills and huge resource investment. Therefore, it is not more suitable for end users. Despite lacks some of the features, however, such methods would be necessary to achieve high scalability and fault tolerance in massive data processing [31]. Therefore, the main challenge is to achieve efficiency, without losing scalability and fault tolerance. The efficiency problem is expected to be overcome in two ways: (i) using the effective index system and (ii) scale up clustering.

Providing multiple static indexes for a variety of queries still have state-of-art index techniques when it comes to unknown or changing workloads. Although creating missing indexes is the dark side of adaptive index, but automatically index creation during the job execution time is more valuable. The result shows that HAIL adaptive indexing has a very low overhead compared to Hadoop full scan.

Although MapReduce has good performance in job distribution, but reducing phase in Hadoop is time costly. Therefore, at the first glance parallel DBMSs are still a winner hours in processing battlefield. However, by increasing the number of nodes in the cluster, reducing time decrease. In other word MapReduce system has better performance on big data when it is distributed on thousands of servers. Furthermore, we need an optimum number of parallelism to reduce network overhead. Last but not least, there is no other cost effective option better than the MapReduce encounter of massive data processing.

Obviously the dynamic nature of big data goes beyond traditional tools and application. Therefore, any new approach must be suitable enough for traditional structured database and new unstructured data set. As a part of this process, the agent responsible for fast retrieval by focusing on classification, filtering and storing data. Furthermore, immigration from centralized to distributed approach improve response time because of data locality, but albeit costly.

References

[1] S. Loudcher, W. Jakawat, E. Morales, C. Favre,” Combining OLAP and information networks for bibliographic data analysis: a survey,” Scientometrics, Volume 103, Issue 2, pp. 471-487, 2015

[2] U. Fayyad, G. Piatetsky-Shapiro , P. Smyth ” The KDD process for extracting useful knowledge from volumes of data,” Communication of the ACM, Volume 39 Issue 11, pp. 27–34, 1996

[3] H.A. Labrinidis, V. Jagadish,” Challenges and Opportunities with Big Data,” Proceedings of the VLDB Endowment, Volume 5 Issue 12, August, pp. 2032-2033, 2012

[4] D.M Assuncao, N.R Calheiros ,S. Bianchi, A.S Netto, R. Buyyab,” Big Data computing and clouds: Trends and future directions,” Journal of Parallel and Distributed Computing, Volumes 79–80, pp. 3–15, 2015

[5] D.J Abadi,” Data management in the cloud: Limitations and opportunities,” IEEE Data Engineering Bulletin 32 (1), pp. 3–12, 2009

[6] S. Sakr, A. Liu, D. Batista, M. Alomari," A survey of large scale data management approaches in cloud environments," IEEE Communications Surveys Tutorials, 13 (3), pp. 311–336, 2011

[7] S. Ghemawat, H. Gobioff, S-T. Leung," The google file system," Proceedings of the 9th ACM Symposium on Operating Systems Principles (SOSP), pp.29–43, 2003

[8] J. Yu, J. Ni," Development Strategies for SME E-Commerce Based on Cloud Computing," Seventh International Conference on Internet Computing for Engineering and Science, 2013

[9] R. Tinati, S. Harford, L. Carr, C. Pope," Big data: methodological challenge and approaches for sociological analysis," Sociology 48(1), pp.23-39, 2014

[10] D.H. Shin, M.C. Choi," 2015. Ecological view of big data," perspective and issue' Telematics and Informatics, Volume 32, Issue 2, pp.311–320, 2015

[11] J. Dittrich, J. Arnulfo, Q. Ruiz, A. Jindal,Y. Kargin,V. Setty V, and J. Schad," Hadoop++: Making a Yellow Elephant Run Like a Cheetah (Without It Even Noticing)," PVLDB, Volume 3, Issue (1-2), pp.515–529, 2010

[12] C.L P. Chen, C.Y. Zhang," Data-intensive applications, challenges, techniques and technologies: A survey on Big Data," Information Sciences. Volume 275, pp. 314–347, 2014

[13] Q. Wang, C. Wang, K. Ren, W. Lou, J. Li," Enabling public auditability and data dynamics for storage security in cloud computing," IEEE Trans. Parallel and Distributed System. Volume 22, Issue 5, pp.847–859, 2011

[14] D. Agrawal, S. Das, A. Abbadi," Big data and cloud computing: current state and future opportunities. Proceedings of the 14th International conference on Extending database Technology," EDBT/ICDT11, pp.530-533, 2011

[15] A. Jindal, Q. Ruiz, J. Dittrich," Trojan data layouts: right shoes for a running elephant," SOCC, 2011

[16] C.P. Raj, S. Vanga," Use big data and fast data analytics to achieve analytics as a service (AaaS) Key analytical platforms on IBM SoftLayer Cloud," IBM developer works, 2015

[17] A. Rathi, N. Parmar," Secure Cloud Data Computing with Third Party Auditor Control. Proceedings of the 3rd International Conference on Frontiers of Intelligent Computing," Theory and Applications (FICTA), Volume 328 pp.145-152, 2014

[18]https://highlyscalable.wordpress.com/2012/03/01/nosql-data-modeling-techniques/

[19] A. Pavlo, E. Paulson, A. Rasin, D. Abadi, D. DeWitt, S. Madden, and M. Stonebraker,"A Comparison of Approaches to Large-Scale Data Analysis," Proceedings of the 2009 ACM SIGMOD International Conference on Management, pp. 165–178, 2009

[20] A. Ailamaki, J. DeWitt David, D. Hill Mark," Data page layouts for relational databases on deep memory hierarchies," The International Journal on Very Large Data Bases (VLDB), Volume 11 Issue 3, pp. 198-215, 2002

[21] S. Chen," Cheetah: a high performance, custom data warehouse on top of MapReduce," PVLDB Volume 3, Issue (1-2), pp.459–1468, 2010

[22] S. S. Richter, Q. Ruiz, S. Schuh, J. Dittrich," Towards zero-overhead static and adaptive indexing in Hadoop," VLDB, Volume 23, Issue 3, pp 469-494, 2013

[23] K. Zoumpatianos, S. Idreos, T. Palpanas," Indexing for Interactive Exploration of Big Data Series," SIGMOD '14 Proceedings of the 2014 ACM SIGMOD international conference on Management of data, pp.1555-1566, 2014

[24] M. Lühring, K.U. Sattler, K. Schmidt, E. Schallehn," Autonomous management of soft indexes," ICDE Workshop on Self-Managing Database Systems, pp. 450-458, 2007

[25] G. Graefe, F. Halim, S. Idreos, H.A. Kuno, S. Manegold, and B. Seeger," Transactional support for adaptive indexing," VLDB, Volume 23, Issue 2:pp.303–328, 2014

[26] S. Idreos, S. Manegold, H. Kuno, and G. Graefe," Merging what's cracked, cracking what's merged: adaptive indexing in main-memory column-stores," VLDB, Volume 4, Issue 9, pp.586–597, 2011

[27] J. Dittrich, S. Richter, S. Schuh, J. Arnulfo, Q. Ruiz," Efficient OR Hadoop: Why not both?," Schwerpunktbeitrag, Datenbank-Spektrum, Volume 13, Issue 1, pp.17-22, 2013

[28] J. Dittrich, J. Arnulfo, Q. Ruiz, S. Richter, S. Schuh, A. Jindal, J. Schad," Only Aggressive Elephants are Fast Elephants," Proceedings of the VLDB Endowment, volume 5 Issue 11, pp.1591-1602, 2012

[29] A. Abouzeid, K. Bajda-Pawlikowski, D. Abadil, A. Silberschatz, A. Rasin," HadoopDB: An Architectural Hybrid of MapReduce and DBMS Technologies for Analytical Workloads," Proceedings of the VLDB Endowment, Volume 2 Issue 1, pp.922-933, 2009

[30] B.P. Gibbons," 2015. Scale Down, Scale Up, Scale Out," IEEE 29th International Parallel and Distributed Processing Symposium, 2015

[31] K-H. Lee, Y-J. Lee, H. Choi, Y.D. Chung, B. Moon," Parallel Data Processing with MapReduce: A Survey," ACM SIGMOD, Volume 40, Issue 4, pp.11-20, 2011