Fast and Efficient In-Memory Big Data Processing

Babur Hayat Malik[1], Maliha Maryam[2], Myda Khalid[3], Javaria Khlaid[4], Najam Ur Rehman[5], Syeda Iqra Sajjad[6]
Tanveer Islam[7], Umair Ahmed Butt[8], Ali Raza[9], M. Saad Nasr[10]
Department of CS and IT, University of Lahore, Chenab Campus, Gujrat Pakistan

***Abstract*—With the passage of time, the data is growing exponentially and the mostly endured areas are social media networks, media hosting applications, and servers. They have thousands of Tera-bytes of data and the efficient systems, however, they are as yet confronting issue to oversee such volume of information and its size is growing each day. Data systems retrieve information with less time of In-memory. Instead of each factor data systems are required to define good usage of cache and fast memory access with help of optimization. The proposed technique to solve this problem can be the optimal indexing technique with better and efficient utilization of Cache and having less overhead of DRAM with the goal that energy can also be saved for the high-end servers.**

Keywords*—Big data processing; indexing techniques; R-tree; B-tree; X-tree; hashing; inverted index; graph query tree

I. INTRODUCTION

A. Big Data

Big Data is large datasets whose scale, diverseness, and complexity involve new strategic technologies for algorithmic computer program, and analytics to manage it and excerpt value and hidden knowledge from it. With the passage of time the data is growing exponentially and the mostly domains that hold thousands of Terabytes of data are social media networks, media hosting applications, and servers [1]. Big data is basically an umbrella term for data that are too large to handle. Characteristics of data involve 4V's.

1) Volume (Scale)
2) Varity (Complexity)
3) Velocity (Speed)
4) Veracity

Day by day data volume is increasing exponentially and it needs to be processed fast. Due to increase in the volume of data, it is also complex to handle because one application is generating different types of data containing several formats, types, and structures, text, numerical, images, audio, video, sequences, time series, social media data, multi-dim arrays, etc. [2]. The veracity of data is measured by checking its accuracy and types of data structured and unstructured and it is difficult to check the veracity of unstructured data.

Growth of data has become an issue for the past few decades because it is not easy to manage that data and store it, perform computation and extract any information from that size of data. There are many solutions for that type of data while managing it some says to classify it to make the retrieval fast. Some says to store it in efficient manner in most upgraded machines with High performance processors, Main memory and also secondary memories. In Memory Systems are better than other disk based systems because performance of main memory is better than hard drive [3]. As field of computing is very vast and progressing, each day is seen with new inventions and innovations in every domain and sub-domain, like the SSD. A secondary storage drive and PCM is very fast than the old HDD. Number of Computations per second in these No-Volatile memories is far greater than the simple hard drive. It is very integrated in size and very efficient access time [4]. Modern Servers in these days normally have hundreds of gigabytes of DRAM and tens of cores while the fastest of them have TB's of DRAM and Peta-bytes of secondary storage with hundreds of cores to process the gigantic size of data [5]. In-memory systems have been discussed in 1980s [6], and after that is has not been studied. However, recent progress in the computing era has changed the previous work entirely and developed an interest to host the entire data in Main-memory to perform faster access data analytics [7]. In-memory Data processing has been widely used to support large-scale applications totally in DRAM. Different aspects can be considered for optimizing in-memory system such as indexing, data layouts, Parallelism, Fault-tolerance, Data-overflow, concurrency control and query processing [8], [9]. Indexing deals with cache utilization and parallelism deals with data compression [10] and fast processing means packing multiple values in single processor word.

Parallelism is of two types scale-up and scale-out and both of these optimize performance by partitioning data. Concurrency Control is another aspect that has great impact on the performance of data analysis [11]. Different mechanism of concurrency control has different demerits like Heavy-weight mechanism consists of too much locks/semaphores which are the key ingredient of degrading the system performance. Similarly, other mechanisms are Light-weight Intent Lock (LIL) and very lightweight locking (VLL) simplifies the data structure by compressing all the lock states and some are based on time stamps [12], [13]. Other efficient techniques are MVCC "Multi-Version Concurrency Control" Search and Retrieval of this type of content has become a challenging task now a-days because the data volume is very high and it is not easy to manage such size of data and performing computation on that type of data.

Using Distributed Environment is one of the solution so that performance of computation on that data can be optimized [14]. In memory data storage defines performance improvement attribute because the latency of DRAM is much less than that of permanent Storage and the data in RAM can be retrieved fast because the processor can better access DRAM than the hard drive. But this still needs some kind of

optimization that can make this process space and time efficient and can retrieve data optimally.

Many Indexing techniques have been proposed such as Hash-based, Tree-based but all of them have deficiencies of any kind such as some of them have performance issues and others are not time efficient. Few enhance cache by performing huge computations while other fails to correctly compute cache. So there should be an optimal solution that can search huge data optimally. All the work about it involves in improving the indexing technique which could be to make a new and different algorithm that is best among the others implemented before or it can be the combination of any two best indexing techniques means.

B. Optimizing Big Data Processing

With the Increase of Data volume and size, it has become difficult to manage and process it efficiently. Although many techniques have been proposed in this era, as indexing, parallelism, data layouts management, concurrency control and fast query processing but it still needs optimization so that its processing time can be decreased. As the project is about In-memory big data processing so it should be reminded that in this case, a copy of database should be kept in main-memory which is DRAM all the time so that it should be processed fast but it still needs an efficient indexing technique to process more fast with less cost and time. The whole project is about optimizing indexing technique of big data to make it process fast and the end its energy should be measured because all the work overhead depends upon DRAM that's why the energy should be kept in mind.

This paper is organized as follows: Section II presents the comprehensive literature of previous work. Section III presents a comparisons based on related literature. Finally, Section IV concludes the paper and future work is also discussed.

II. Literature Review

A. Research Paper Summaries

Author of paper [1] described that for data mining, creative technologies, and analysis of data and prediction the word "Big data" appears on most of the specialized meetings around the world. Where the organization of huge sums of information is toward the best real work this word is applied in these zones. To characterize these zones, a constant rise of information brooks in the administrative procedure, whether it is an economy, savings, making, selling, broadcastings, treatment, etc. The knowledge of huge information place administration inside an equivalent treating of assets for best technology background is mainly related. The training of in memory knowledge application structures for large information groups on heavy technology which is based on SAP HANA knowledge's used as information storage and presented in this paper. SAP HANA Colum store method read information fast and data compression methods efficiency rise. The compacted information procedure lets decomposition process resources cheaper. With a similar engineering of possessions, the prices economy for large information saving and dispensation with the practice of in-memory method founded on SAP HANA let's decrease bulks of information stored in the main system information. Different to old-style files shaped for hard disk connecting, the communication of RAM and the processor is the initial aim of SAP HANA. A method [1], like a column store is applied in that situation, which moreover takes a quick action of data reading; open the ways so that data compression mechanism is applied in a good way. Deprived of outgoings wealth on the decompression procedure, SAP HANA works with compacted information in a direct way. IT facilities management that is incorporated into the IT setup of an information founded boldness, now takings residence with the service concerned with communal material schemes usage.

Author of [2], described that the use of technology is greatly improved by the installation of sensors. As time has increased the requirements have also increased. In addition to that, the number of problems has also increased which are related to the equipment of an industry. Therefore, it is required that new systems are introduced for obtaining the best performance and storage of data. Industries use a large number of sensors to monitor, process and storage of data. In the past, the systems having particular needs were used in the industries that could not have the capability to process the data in a fast way and store it. For the storage of data in the middle of working, fast processing and storing high volume the In-memory technology is the most suitable one. In this paper [2] the analysis of data at high speed and having high volume is done by developing the prototype having in-memory at the core. So, the time series data is stored continuously and simultaneously analyze the data. The case is discussed in which the memory is stored at the rate of 10000 data points/sec. Also, the analytical collection is implemented for IMDG and the data is updated in a continuous manner. The data of 3 days is stored in memory in addition to duplication in which the fault is tolerated. In this process, five terabyte of Ram is consumed [2]. While this work was performed, numerous challenges were faced like in which way the performance can be maximized. In addition to the selected data storage and processing capability, it is important to have an additional memory so that overhead and processing tasks are carried out. In the future work, it is proposed that the CEP framework will be employed which has the capability to process millions of data points/sec. Other methods like Apache storm, Hadoop file systems, and other such techniques will be implemented.

In [3] with the increase of technology, the usage of computers as well as the computation is increased. Moreover, the computation speed at higher rate is also required. For this purpose, either the approach of utilizing the memory processing or the disk-based approach is used. The main demerit of disk-based approach is that the performance is sacrificed. This paper presents a novel approach in which the scaling of graphs into sub graphs is made by RAM-Disk approach. The graphs are divided into sub graphs which are suitable for RAM. So, there is no modification in algorithm and the approach deals with small memory to large memory machines. The approach of processing in terms of out of core is identical to that of paging. However, it is applied to the sub graph which can also be thought as logical partitions. The logical partitions are formed so that they are fit in the main

memory and then combined with asynchronous push model. For the implementation of algorithms, the use of graph-processing engine has been made. It is helpful in executing the algorithms by which accessible resources can be utilized for quick as well as scalable processing of graphs. For the implementation, STAPL frame work is used. The algorithms of STAP GL are also used without any modification. The running time on various platforms having different processing capabilities is illustrated in the form of diagram in the paper. Therefore, the selected platforms include a 2-core tablet having only 1 GB Ram, another PC having 4-core with the memory of 8 GB and supercomputer of CRAY XE6. The running times for various graph mining as well as the graph analytics on PCs having memories of 4 GB and 8 GB RAM are also shown in the paper. This helps in extending the approach to the machines that are based on distributed memory.

In [4], the Big Data figuring is one of the problem areas of the web of things and distributed computing, whose exploration substance are securing, administration, handling, appear, thus on of monstrous information. In the handling join, how to process effectively on the Big Data is the key to enhancing execution. By methods for dispersed registering or memory figuring, numerous organizations and foundations give a few advancements and produce. The Hadoop innovation is run of the mill illustrative of disseminated registering. In Hadoop, MapReduce is a circulated programming system proposed by Google. It is a framework for parallel handling of extensive informational collections. In MapReduce, the undertaking can be part into subtask and the disseminated parallel registering can be acknowledged effortlessly. In memory processing, the Big Data are circulated stacked in various PCs as indicated by some standard. The SAP HANA database can possibly give execution enhancements to existing SAP applications. The circulating memory is registered in which the Hadoop innovation and memory registering innovation mix together. The PC amount of group isn't constrained, so we can stretch out group vastly to store substantial information. Yet, they are invalid in the scene in which there are continuous requests in the low-arrange group. To manage the issue, this paper gives a conveyed registering and memory processing based viable arrangement (Objectification Parallel Computing, OPC). The OPCA is made out of Client Proxy, Protest Manage Server and Object Server. There are three programming in the OPC: Object Manage Server Soft (OMSS), Object Server Soft (OSS), what's more, Client Proxy Soft (CPS). In the arrangement, the information can be organized into protest. At that point, the items are conveyed put away in the PC recollections and parallel process to finish assignments. The OPC is connected to the Electric Asset Quality Supervision Oversee System (EAQSMS) of State Grid of China, the outcome demonstrates that with PCs the framework is proficiently accessible, solid, furthermore, adaptably expansible. There are a few inquiries to investigate and settle, for instance, information pressure, checking the bunch, hot backup, et cetera. These issues require additionally look into. At present, innovation advancement changes rapidly. We require constantly center around new innovation to unravel the issue.

In [5], it is mentioned that in these days, the capacity of information and the administration of interconnected gadgets has turned into an incredible test. In this way, putting away such a lot of information and its calculation requires a unique record system known as MapReduce display which stores, process, oversees and executes the information. Apache Spark is one of the computational frameworks and its apparatuses and related systems are accessible as an open source permit. Several different associations of the world have made a utilization of Hadoop document framework. These computations are very fast in nature. But the main factor which may cause a failure is the storage or memory. Therefore, tuning of memory can be done by the Spark. For this purpose, the excessive knowledge is required. In this paper, the memory selection methods to which the first step is taken is discussed. In this way, the failure of memory and the execution which is 25 percent is decreased. The load of work to be selected is 20 gigabytes of data. The experiment is performed by the use of one thousand and four thousand on 15 gigabytes and 2 gigabytes of memory. This paper deals with the detection as well as the automation [5]. Various graphical representations are shown in the paper such as execution time against the caching strategies, a sort of algorithm which represents the selection process of cache and its configuration. Various kinds of caching modifications have been observed and are experimentally assessed. In this way, the footprints, as well as the working performance, are improved. However, the Spark's strategy for caching is observed to be inefficient in such cases where the data is not adjustable in memory. Therefore, various strategies discussed will result in some tradeoffs. Because of the reduction of footprints, a collection of garbage and its frequency is reduced.

Author of [6], addresses the problem of performing operations on bulk data is quite a time consuming and had bad impact on system performance too. There are three reasons due to which operations on bulk data reduces its latency, bandwidth and effect the system performance and energy efficiency. First, present systems perform bulk data operations on byte/line at a time due to which system has to face performance issues due to high latency. Second, these tasks require a substantial amount of information to be exchanged over the memory channel. Henceforth, they are in a roundabout way that influenced the execution of simultaneously running applications over the memory data transmission. Third, the information transfer across the memory channel speaks to a significant portion of the vitality required to perform operations on bulk data.

In this paper basic goal is too focused on optimizing two classes of bandwidth-intensive memory operations one is bulk data copy which means transferring of data in physical memory, the second is bulk data initialization to reduces the latency, bandwidth, and energy consumed by bulk data operations. Row clone is basically a simple technique to perform operations on bulk data within DRAM. This methodology takes out the need to exchange information over the memory channel to play out a bulk information activity, and subsequently can possibly moderate the related inactivity, data transfer capacity and vitality issues.

Author of [7] tells us the blast of computerized information and the regularly developing requirement for big data investigation has made in-memory big data processing progressively imperative. Due to fast-growing data, processing large graphs in main memory is still a problem. Because of memory bandwidth restrictions, it is difficult to build systems whose performance increases relatively with the increase in graphs. To accomplish a target processing-in-memory (PIM) can be a practical solution. With the end goal to take advantage of another innovation to empower memory – capacity-proportional performance Tesseract is designed for a large scale graph processing. It is made up of a new architecture that completely uses the memory bandwidth. It is a proficient strategy for correspondence between various memory partitions. It also includes two hardware pre-fetcher which works based on hints. By using this strategy author demonstrated that Tesseract works well in both performance and energy effectiveness. Tesseract accomplishes memory capacity proportional performance, which is the main objective in dealing with a large amount of data. This new plan can be a proficient and useful substrate to execute emerging data-intensive applications with memory bandwidth requests.

In [8], it is described as balanced Binary Trees (B Trees) are not optimal for indexing on modern hardware because they cannot better utilize the Cache. These shortcomings have been improved in Adaptive Radix Tress that is also a space efficient. Its lookup performance is very time efficient and supports optimal insertions and deletions as well as deals with the worst-case space consumption, which plagues most radix trees, by adaptively choosing compact and efficient data structures for internal nodes. Even though Adaptive Radix Trees performance is almost similar in time to hash tables but it stores the data in the sorted form, which requires additional computations like range finding and making it in order. But still, ART has performance issue due to a lot of computations. Later on, more work can be done like synchronizing simultaneous updates.

Author of [9] discuss that the development of unstructured information in social media gives an open challenge to cloud database network. In this era, the way we deal with big data processing becomes a serious issue. MapReduce is one of the techniques used for big data analytics. There is a number of indexing techniques like Hadoop++, HAIL, LIAH, and Adaptive Indexing. These techniques are still not efficient or an optimized way to process big data. To solve this problem author proposes a solution that is basically HDFS indexing techniques named as Low-Index and High-Index. The goal of these new approaches is to provide a platform to index in Hadoop Distributed File System and MapReduce frameworks without changing the current Hadoop structure. Low-index gives an index that contains text and it tells the Hadoop to filter only those which contain the related terms. Low-Index likewise upgrades the throughput (limits reaction time) and defeats the problem of long inactive time for list creation. This new approach is better in performance than the Lucene but not efficient in response time. To overcome this problem, High-Index is proposed which is found better than Low-Index in computation and response time. We compare the execution time of both Lucene and new suggested approaches Low-Index and High-Index. In the beginning, these two took more time for creating an index but after that, they perform better the Lucene approach. In future, we can improve these approaches by working on composite queries in a huge cluster setup.

In [10], the author discusses that for big data top-k queries are a big challenge. Top-k systems depend on positioning functions with the end goal to decide a general score for every one of the items over all the relevant attributes being examined. This positioning capacity is given by the client at query time. Bit-sliced indices (BSI) were proposed to answer these queries proficiently. MapReduce and key-values stores are strategies for investigating huge information; we set up to assess the execution of BSI. Indexing is implemented over Apache Spark for both row and column stores and appeared to beat Hive when running on Map-decrease, and Tez for top-k queries. This methodology is strong and useful for high dimensional information. Top queries are executed over a dataset with 8,000 dimensions. In performing experiments, when expanding the quantity of CPU from 24 to 48 query time reduced by around half while diminishing the number of bit-slices per measurement, the index size and the query time is also reduced. On the bases of this result, we concluded that proposed techniques performed better over Hadoop MapReduce and Hive over tez. For future work, the author intends to explore the correct measure of information rearranging done by different vertical and horizontal partition sizes and how this influences the query time. This can help on deciding the ideal number of bit-slices that ought to be gathered together during map-reduce aggregation. Additionally, it can also be a plan to research more impacts of the BSI attribute section size.

In [11], author tell us that using old strategies like storage of data on disk is now become one of the problems due to growth in data. They don't scale smoothly to address the issues of huge scale Web applications, and disk space limit have far exceeded enhancements in access latency and data transfer capacity. This paper contends for another way to deal with data center capacity called RAM Cloud, where data is kept totally in DRAM. We trust that RAM Clouds can give long-lasting and available storage with 100-1000x the throughput of disk-based systems. RAM Clouds use different techniques like storing the copy of data in DRAM for the fast retrieval of data and to provide durability for data. The two important points in this approach are that they have low latency and they have the ability to combine the resources of a large number of commodity servers. RAM Clouds also have some drawbacks like high cost and high use of energy. Later on, both innovation patterns and application necessities will manage that a bigger and bigger portion of online information is kept in DRAM.

In [12], the author described that Due to the increase in data recently we are facing emerging security and protection challenges. Huge information, since it can dig new learning for monetary development and specialized advancement. New mined data will be unconvincing; while if security isn't all around tended to, individuals might be hesitant to share their information. Since, security has been explored as another

measurement, "veracity," is enormous information. In this article, commit our consideration toward privacy in the big data era. First, formalize the general design of enormous information investigation, distinguish the relating protection prerequisites, and present a productive and security saving cosine computing protocols as an example in response to privacy requirements. There are different existing privacy-preserving techniques. One is privacy-preserving aggregation, second is operations over encrypted data and third is de-identification techniques. To evaluate the proposed PCSC protocol it is compared with direct cosine similarity computation and the HE-based protocol. Based on JAVA language, both used to evaluate results with the same output on the PC with Intel Pentium CPU B980 running at 2.40 GHz, and with 6 Gbytes of RAM. The experiment results show that the proposed PCSC protocol is also efficient along with privacy preserving. Further research can be done by addressing unique privacy issues in big data analytics.

In [13], author addresses because of different difficulties and significant issues it is difficult to manage big data. Map Reduce is the technique for handling the huge amount of information. In this 3-layer traffic, aware clustering algorithm is proposed as the best solution for traffic aware partition and aggregate to minimize the cost of network traffic. One problem that is faced in network traffic is difficult to process data in a given time. The basic goal is to reduce network traffic cost. As a result of applying this technique helps in reducing response time and simulation experiment result revealed that this proposal can reduce the network traffic.

Author in [14] is investigating a lot of healthcare information is as yet a challenge because of the absence of sufficient information management techniques that empower responsive information investigation and examination. A single patient record consists of multiple attributes. Doctor keeps this record for the future prediction about the patient. To resolve this, issue the author compares two in-memory New SQL database technologies Mem SQL and VoltDB. For doing this author uses Medicare claims synthetic data. The goal is to explore data faster and to enable real time predictions. This proposed solutions shoes that most of the queries reply back within 10 seconds. Further, it is planned to continue this research by using different new SQL databases like SQL Fire and to develop a new tool for real-time analysis of data using in-memory database system.

In [15], author address a problem that due to the increase in data it is difficult to use a DRAM for big data processing. As DRAM is facing capacity issues and become a reason of main power drain in modern computers. In this paper, the author proposed an effective page management technique for vast scale NVDIMM memory and gives an effective management technique by keeping in mind both TLB performances and page fault rates. Page fault rates become small with the increase in NVDIMM capacity. NVDIMM-based memory models are getting huge importance due to shifting of the desktop to cloud environment. The goal of this technique is, to be helpful in designing new applications for big data.

In [16], the author mentioned a problem of the large performance gap between processor computation and memory access. To solve the memory wall problem 3D stacked technology is proposed. It is combined with NVM. The proposed solution has some advantages like big capacity low cost and non-volatility. In future NVM materials can be studied along with 3D memory systems.

In [17], the author tells us that because of less expensive and faster processing in DRAM in-memory data management systems have gained a lot of attention. The author proposed an innovative in-memory data management system named as MemepiC. It brings together both online data queries and data analytics functionality permitting low-latency and proficient in-situ information analytics. For conveying message inside the MemepiC RDMA-based communication protocol is designed. Different experiments are performed by to show how efficient MemphiC is, in terms of both storage and data analytics services.

In [18], author tells us that nowadays storing and managing a big data is not only a challenge but also extracting a useful information from that data. The main purpose of this paper is to analyze unstructured data. There are many techniques to solve this problem. MapReduce in connection with the HDFS and HBase database as part of the Apache Hadoop project is a modern approach to evaluate unstructured data. Hadoop methods are used for handling big data and can be enhanced with the right approach.

Author in [19], addresses the problem of gathering a large amount of data. The organizations have to face issues of big data efficient performance and the raised infrastructure cost with the data processing. The new architecture is shifted from centralized to distributed architecture and with the help of these changes organizations are able to defeat with the problem of getting related information from a large amount of data. Apache Hadoop is a proposed technique to solve this problem. The basic goal is to facilitate user and provide them a useful information in less time with least effort.

In [20], while studying big data there are two problems that occurs one is storage of a large amount of data and second is processing speed. To solve these issues in Grid Technology is used. The Main advantages of using this are capacity abilities and the handling power. The Oracle/Cloudera approach is a successful combination of Cloudera's software tool and the Oracle built frameworks intended to give high performance and adaptable information handling for Big Data.

In [21], author said that B-tree or B + -tree is the most famous index structure in disk-based relational database systems, the T- tree has been broadly used as the good approach of index structure for in-memory databases where the entire database resides in the main memory. However, the research work on T-tree doesn't take into account the concurrency control that is a drawback one can say. Similar to B-tree index is B-link tree outperforms the T-tree if concurrency control is reduced. This is because the concurrency control over a T-tree demands more locking than that of a B-link tree, and the overhead of locking and unlocking is high which results in the performance degradation.

In [22], author describe that the data is not big in terms of their volume but also include the queries that are performed to access that data. There are many strategies through which we can search data using different keywords. In this paper new technique is introduced named as ADAM which allows a Boolean retrieval of data for structural and unstructured data. Using this technique queries are made in in the style MapReduce. Signature-based indexing strategy is supported and minimizes the access time. The accuracy and efficiency of this are measured by performing an experiment and form ImageNet 14 million images are retrieved. In the future, it can be tested on more large-scale data consist of multimedia. The author proposed indexing technique which is called hash. This technique takes the bulk of images and gives images that are same as a result.

In [23], STR (Sort-Title-Recursive) algorithm is modified for indexing technique R-trees. It will take spatial data as a data type. After the implementation of this, it is compared with the previous techniques. It is evaluated in terms of space storing index strategy. To improve STR two methods are presented. One is to collect sorted spatial objects and combine them in each axis in the form of slices. In the second strategy, each object has its own axis and then for every object connects them into suboptimum space filling. This improved STR performed well the previous methods. In future Revised R* - the tree can be implemented.

In [24], the author described that as we all know that it is difficult to handle spatial data. In memory, the database needs a method with the use of which we can easily retrieve data and then update. To fulfill this need author proposed new indexing technique named as R-tree. It is an algorithm that helps in updating and searching for data. This algorithm is implemented in Different experiments are performed in the result of which it is concluded that it is useful for spatial data.

In [25], the author presented an Inverted indexing technique to tackle the problem of processing the growing spatial data. This technique is the partition of inverted and grid indexes. This method is implemented with MapReduce many experiments are done to check the scalability of the technique. This technique is constructing time of index is very much less as compared to the other trees.it is three times faster when compared with Voronoi-based query processing.

III. Comparative Study of Related Literature

After critically analyzing all the literature, his section present the all techniques discussed. There are two types of indexing techniques. One is Artificial Intelligence Approach [23] and second is Non Artificial Approach. Non Artificial Approach is further divided into three categories [20], [21]. One is Tree Based Indexing, second is Inverted Indexing and third is Hashing [24]. Tree is further divided into more categories. Three are shown in Fig. 1, B-tree-tree and X-tree.

Different Indexing technique use different type of data for processing [25]. Every technique has different ways to perform query. All of these factors affect the complexity of indexing technique. The technical summary of indexing technique is given in Table I.

There are many factors that affect the hashing and tree indexing technique. Table II discusses these factors. A comparison of different indexing techniques is shown in Table III and also the analysis of indexing techniques on the basis of big data characteristics is given in Table IV. Furthermore, on comparing all the indexing techniques from Section II, the advantages and disadvantages of these techniques are analyzed and given in Table V.

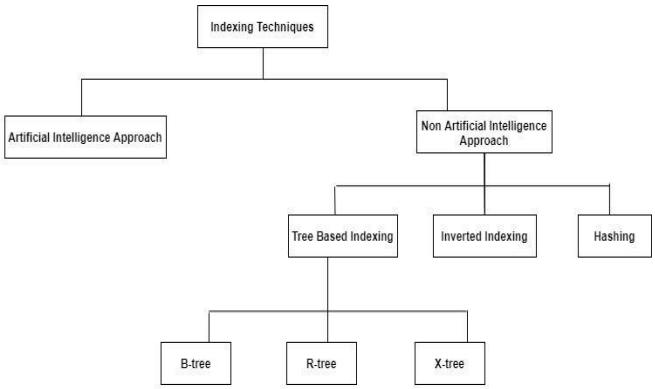

Fig. 1. Flow Chart.

TABLE I. Technical Summary

Technique	Concerns	Related Work
Indexing	Space Efficiency	Graph Query Tree
	Time Efficiency	X-Tree
	Better Use of Cache	B-Tree
		R-Tree
		Hashing
		Inverted Indexing

TABLE II. Factor Effecting Techniques

Factors	Hashing	Tree Indexing
Access Latency	Reduce Access Time	Minimum Access Time
Space	Efficient in Space Handling	High Space Consumption
Indexing Efficiency	Efficient in Balancing Access and Tuning Time	Not Powerful
Miscellaneous	N/A	Good for Random Access
Time	Hashing Function Reduce Stunning Time	Minimal

TABLE III. COMPARISON OF INDEXING TECHNIQUES

Indexing Techniques	Data Type	Query Type	Complexity
R-Tree	Multimedia and Spatial Data	2 to 3-dimensional Access Method	Worst Time Complexity and Inefficient usage of Time
B-Tree	Multimedia and Log Data	1-Dimensional Access and Range Queries	O(logn)
X-Tree	Spatial Data	Multi-Dimensional Access and Range Queries	Linear and Time Complexity O(n)
Hashing	Multimedia and Log Data	Point Query	N/A
Inverted Index Tree	Multimedia Data and Documents	Keyword Queries	N/A
Graph Query Tree	Graph	N/A	N/A

TABLE IV. ANALYSIS BASED ON BIG DATA CHARACTERISTICS

Indexing Techniques	Volume	Velocity	Variety	Veracity
R-Tree	Yes	N/A	No	N/A
B-Tree	Yes	N/A	N/A	N/A
X-Tree	Yes	N/A	N/A	N/A
Hashing	Yes	No	Yes	No
Inverted Index Tree	Yes	N/A	No	N/A
Graph Query Tree	Yes	Yes	N/A	N/A

TABLE V. TECHNICAL ADVANTAGES AND DISADVANTAGES

Indexing Techniques	Advantages	Disadvantages
R-Tree	• Less query processing cost. • Query response time depends on buffer size	• Index takes more space
B-Tree	• Faster Construction • Fast query response • Less Updating Cost • Index takes less time	• Data increase cause increase in construction cost
Hashing	• Efficient query response for large dataset	• More initial setup time
Inverted Index Tree	• Index takes less space • Fast query response • Manageable query processing cost	• Require more time to load in memory
Graph Query Tree	• Less processing query cost • Index takes less space • Fast index construction • Fast query response • Less update cost and scalable for large data	• More Computational cost for large network

IV. CONCLUSION AND FUTURE WORK

As the main memory is used as disk, In-memory data management has become interesting for industries. Shifting the data towards main-memory has improved the access time and throughput at a very great extent. Shifting of data has also developed the interest in different aspects to perform optimized results while performing computation on that size of data. Modern Systems has reduced the problem of management of that volume of data by using efficient memory and performance with cache sensitive indexing techniques to better utilize the cache and perform faster Calculations.

As big-data is very vast area of technology, the shortcomings and problems in that field are also at a large scale and all that is in case of management and processing of this size of data. Space efficiency is a factor that should be considered for hash-based indexing techniques. Hashed tree approach can be further improved by working on binary codes to save space and also creating a unique index for each component of data.

REFERENCES

[1] M. Brusakov and G. Botvin, "In-memory technology integration features for work with big data on high-tech enterprises", in Soft Computing and Measurements (SCM), 2017 XX IEEE International Conference on, IEEE, 2017, pp. 697– 698.

[2] J. W. Williams, K. S. Aggour, J. Interrante, J. McHugh, and E. Pool, "Bridging high velocity and high volume industrial big data through distributed inmemory storage & analytics", in Big Data (Big Data), 2014 IEEE International Conference on, IEEE, 2014, pp. 932–941.

[3] N. M. Amato, L. Rauchweger, et al., "Processing big data graphs on memoryrestricted systems", in Proceedings of the 23rd international conference on Parallel architectures and compilation, ACM, 2014, pp. 517–518.

[4] Z. Yang, C. Zhang, M. Hu, and F. Lin, "Opc: A distributed computing and memory computing-based effective solution of big data", in Smart City/SocialCom/SustainCom (SmartCity), 2015 IEEE International Conference on, IEEE, 2015, pp. 50–53.

[5] A. Koliopoulos, P. Yiapanis, T. Tekiner, G. Nenadic, and J. Keane, "Towards automatic memory tuning for in-memory big data analytics in clusters", 2016.

[6] V. Seshadri, Y. Kim, C. Fallin, D. Lee, R. Ausavarungnirun, G. Pekhimenko, Y. Luo, O. Mutlu, P. B. Gibbons, M. A. Kozuch, et al., "Rowclone: Fast and energy-efficient in-dram bulk data copy and initialization", in Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture, ACM, 2013, pp. 185–197.

[7] J. Ahn, S. Hong, S. Yoo, O. Mutlu, and K. Choi, "A scalable processing-inmemory accelerator for parallel graph processing", ACM SIGARCH Computer Architecture News, vol. 43, no. 3, pp. 105–117, 2016.

[8] V. Leis, A. Kemper, and T. Neumann, "The adaptive radix tree: Artful indexing for main-memory databases", in 2013 IEEE 29th International Conference on Data Engineering (ICDE), IEEE, 2013, pp. 38–49.

[9] A. B. Mathew, P. Pattnaik, and S. M. Kumar, "Efficient information retrieval using lucene, lindex and hindex in hadoop", in Computer Systems and Applications (AICCSA), 2014 IEEE/ACS 11th International Conference on, IEEE, 2014, pp. 333–340.

[10] G. Guzun, J. E. Tosado, and G. Canahuate, "Scalable preference queries for high-dimensional data using map-reduce", in Big Data (Big Data), 2015 IEEE International Conference on, IEEE, 2015, pp. 2243–2252.

[11] J. Ousterhout, P. Agrawal, D. Erickson, C. Kozyrakis, J. Leverich, D. Mazières, S. Mitra, A. Narayanan, G. Parulkar, M. Rosenblum, et al., "The case for ramclouds: Scalable high-performance storage entirely in dram", ACM SIGOPS Operating Systems Review, vol. 43, no. 4, pp. 92–105, 2010.

[12] R. Lu, H. Zhu, X. Liu, J. K. Liu, and J. Shao, "Toward efficient and privacypreserving computing in big data era", IEEE Network, vol. 28, no. 4, pp. 46–50, 2014.

[13] G. Venkatesh and K. Arunesh, "Map reduce for big data processing based on traffic aware partition and aggregation", Cluster Computing, pp. 1–7, 2018.

[14] M. Mian, A. Teredesai, D. Hazel, S. Pokuri, and K. Uppala, "Work in progressin-memory analysis for healthcare big data", in Big Data (BigData Congress), 2014 IEEE International Congress on, IEEE, 2014, pp. 778–779.

[15] S. M. Kwon and H. Bahn, "Efficient memory page management for nvdimmbased big data processing environments", in Information Science and Control Engineering (ICISCE), 2017 4th International Conference on, IEEE, 2017, pp. 283– 287.

[16] C. Qian, L. Huang, P. Xie, N. Xiao, and Z. Wang, "Efficient data management on 3d stacked memory for big data applications", in Design & Test Symposium (IDT), 2015 10th International, IEEE, 2015, pp. 84–89.

[17] Q. Cai, H. Zhang, W. Guo, G. Chen, B. C. Ooi, K.-L. Tan, and W. F. Wong, "Memepic: Towards a unified in-memory big data management system", IEEE Transactions on Big Data, 2018.

[18] K. Bakshi, "Considerations for big data: Architecture and approach", in Aerospace Conference, 2012 IEEE, IEEE, 2012, pp. 1–7.

[19] J. Nandimath, E. Banerjee, A. Patil, P. Kakade, S. Vaidya, and D. Chaturvedi, "Big data analysis using apache hadoop", in Information Reuse and Integration (IRI), 2013 IEEE 14th International Conference on, IEEE, 2013, pp. 700–703.

[20] D. Garlasu, V. Sandulescu, I. Halcu, G. Neculoiu, O. Grigoriu, M. Marinescu, and V. Marinescu, "A big data implementation based on grid computing", in Roedunet International Conference (RoEduNet), 2013 11th, IEEE, 2013, pp. 1–4.

[21] H. Lu, Y. Y. Ng, and Z. Tian, "T-tree or b-tree: Main memory database index structure revisited", in adc, IEEE, 2000, p. 65.

[22] I. Giangreco, I. Al Kabary, and H. Schuldt, "Adam-a database and information retrieval system for big multimedia collections", in Big Data (BigData Congress), 2014 IEEE International Congress on, IEEE, 2014, pp. 406–413.

[23] B. C. Giao and D. T. Anh, "Improving sort-tile-recusive algorithm for r-tree packing in indexing time series", in Computing & Communication TechnologiesResearch, Innovation, and Vision for the Future (RIVF), 2015 IEEE RIVF International Conference on, IEEE, 2015, pp. 117–122.

[24] A. Guttman, R-trees: A dynamic index structure for spatial searching, 2. ACM, 1984, vol. 14.

[25] C. Ji, T. Dong, Y. Li, Y. Shen, K. Li, W. Qiu, W. Qu, and M. Guo, "Inverted grid-based knn query processing with mapreduce", in 2012 Seventh chinaGrid annual conference, IEEE, 2012, pp. 25–32.