Volume 6, Issue 5, May 2016 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Efficient Accessing Methods of the Big Data in Cloud Environment

Vamshi Gannu*, S. Puthran
Computer Engineering & NMIMS, Mumbai,
Maharashtra, India

Abstract— Big data more often than excludes data sets with sizes past the capacity of normally utilized programming instruments to catch, oversee, and handle data inside a passable slipped by time. It comprises of data both organized and unstructured, in the field of environmental research, biology, meteorology and numerous others. It has gotten to be hard to handle, oversee and dissect designs utilizing customary databases and structures. In this way, a proper architecture should be designed to understand about the Big Data. This paper analyses different calculations essential for taking care of such vast data sets. All approaches are being reviewed based on the time complexity, parallel processing and distributed environment support. These algorithms define various structures and methods implemented to handle Big Data. Based on the comparison using various parameters it has been found that MapReduce is the effective algorithm for accessing big data over Cloud environment. Due to its ease of use, installation and implementation, Hadoop has found many uses among developers. MapReduce Joins are used to evaluate the queries easily over large datasets. And one of the most important queries are Joins. This paper explores in depth about the various existing solutions, extends them and proposes a few new ideas for joining datasets using Hadoop. Two-way Join algorithms are discussed and evaluated under both categories. These Two way Join algorithms are performed on the IMDB Dataset. The results are expected to give an insight into how good a fit Hadoop or Map/Reduce is for evaluating Joins.

Keywords— Big Data, MapReduce Technique, Hadoop, Cloud Environment, Hadoop Distributed File System, MapSide Join, ReduceSide Join, Cloudera.

I. INTRODUCTION

With the continual improvement of big data and cloud computing, it is trusted that conventional database advancements are lacking for data processing and get to, furthermore execution and flexibility requirements. In the new period of big data, NoSQL databases are more proper than social databases [8]. Key-Value store, a sort of NoSQL databases, is a proper decision for applications that utilization MapReduce model for distributed preparing. Key-Value stores offer just four hidden administrators including inserting <key, value> sets to a data collection, redesigning benefits of existing sets, discovering values connected with a particular key, and deleting sets from a data collection.

MapReduce (MR) is a prominent batch data preparing structure that empowers gigantic measures of such data to be composed and afterward broke down on a distributed group of hubs saving the standards of data region. MR processing is divided into two phases: the *map* phase and the *reduce* phase. Correspondingly every job is partitioned into two sorts of tasks: map tasks that take the crude or prepared data as the information and yield key-esteem matches, and reduce tasks that take the sets yield by the map tasks and figure the last results. In subtle element, the contribution to an occupation is isolated into altered size parts, and for every split one map errand is made. The framework sorts the yield of map tasks, and after that merges them and passes them to the reduce tasks as the info. In this manner, having numerous parts implies the preparing time for the map tasks is regularly shorter than that of the reduce tasks. Additionally, if the parts are tolerably little, the preparing is better load adjusted when the parts are handled in parallel.

Hadoop is a software framework that can be introduced on a Linux bunch to allow extensive scale distributed data examination. The underlying form of Hadoop was made in 2004 by Doug Cutting (and named after his child's full elephant). Hadoop turned into a top-level Apache Software Foundation venture in January 2008. There have been numerous donors, both scholarly and business (Yahoo being the biggest such patron), and Hadoop has a wide and quickly developing client group [6].

Hadoop gives the robust, fault-tolerant Hadoop Distributed File System (HDFS) and additionally a Java-based API that permits parallel preparing over the hubs of the group utilizing the MapReduce worldview. Utilization of code written in different dialects, for example, Python and C, is conceivable through Hadoop Streaming, a utility which permits clients to make and run occupations with any executable as the mapper and/or the reducer. Additionally, Hadoop accompanies Job and Task Trackers that monitor the projects' execution over the hubs of the group [7].

The rest of the paper is organized as follows. Section 2 provides a review of related work. Section 3 provides the Comparative study of various Algorithms for Accessing the Big Data. We discussed the MapReduce Programming model in Section 4, followed by the portable implementation of MapReduce Framework in cloud Environment called

Hadoop in Section 5. And we discussed different MapReduce Joins and analyzed their result based on the time complexity in section 6 and 7.Finally, we conclude the paper in the next Section.

II. RELATED WORK

MapReduce has been utilized at Google since February 2003, and was initially presented in 2004 by Dean and Ghemawat and in Communications of the ACM in 2008. It is utilized for preparing expansive datasets as a part of a parallel or distributed computing environment. It is a mix of map procedures and reduce forms. A map procedure is a capacity that procedures an arrangement of info <key, value> sets that is a part of a substantial information dataset to produce an arrangement of transitional <key, value> sets[4][5]. A reduce procedure with a reduce capacity consolidates all of middle of the road values created by the map forms connected with the same halfway key to frame a potentially littler arrangement of <key,value> sets, called last yield <key, value> sets. Fig. 1 is a straightforward word tallying illustration. The given string data "Apple orange Mango, Orange Grapes plum, Apple plum mango, Apple plum" is partitioned into four squares relating to every subject name isolated by commas. A Hash capacity mod (code(upper(left(key,1))),k)+1 is utilized for disseminating middle of the road <key, value> sets into reduce tasks. The left(key,1) implies taking the primary letter of key, the upper(x) implies changing x to capitalized, the code (x) implies taking ASCII code of character x, and the mod(m, k) implies giving back the rest of m is separated by k.

MR preparing is separated into two stages: the map stage and the reduce stage Fig. 2. Correspondingly every occupation is isolated into two sorts of tasks: map tasks that take the crude or handled data as the information and yield key-esteem combines, and reduce tasks that take the sets yield by the map tasks and process the last results. In subtle element, the contribution to a vocation is isolated into settled size parts, and for every split one map errand is made.

The framework sorts the output of map tasks, and afterward blends them and passes them to the reduce tasks as the

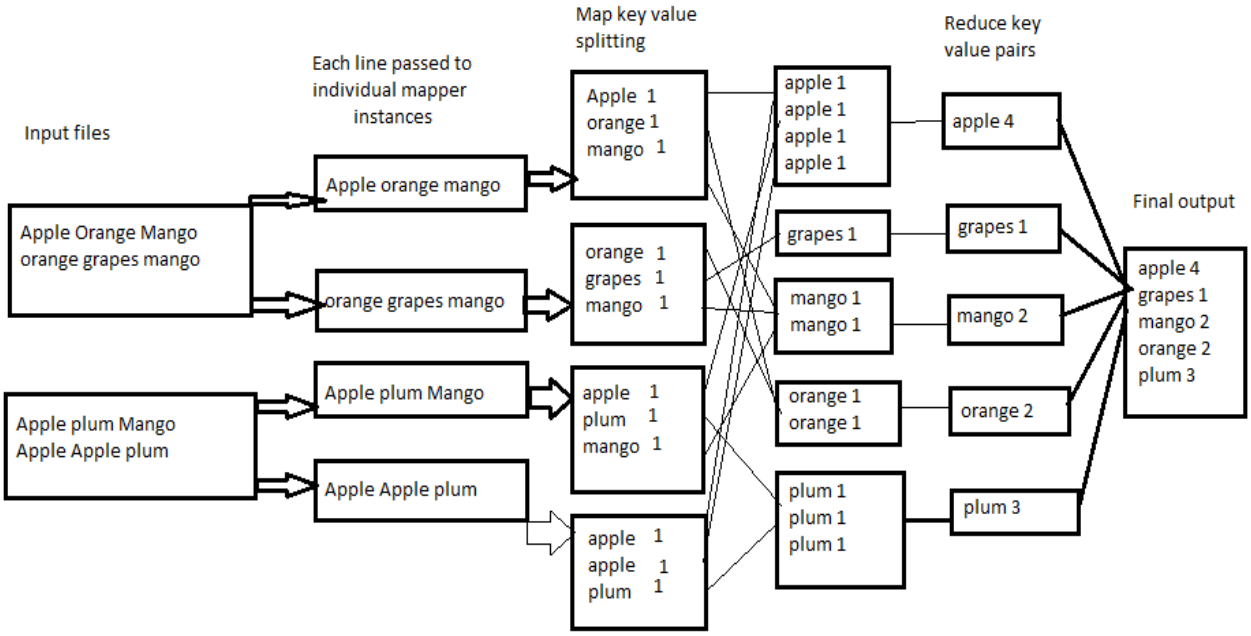

Fig. 1. MapReduce Word Counting Example

Information. Along these lines, having numerous parts implies the handling time for the map tasks is regularly shorter than that of the reduce tasks. In addition, if the parts are reasonably little, the preparing is better load adjusted when the parts are handled in parallel.

III. COMPARITIVE STUDY OF THE ALGORITHMS

Big data will be data so vast that it doesn't fit in the principle memory of a solitary machine, and the need to handle big data by proficient algorithms. The various algorithms which are available in Internet search, machine learning, scientific computing, signal processing, and several other areas can be used to analyse the big data. Many algorithms were defined earlier in the analysis of large data set. In the beginning different Decision Tree Learning was used earlier to analyze the big data this algorithm is reasonably fast and accurate. After that another method is defined an approach for forming learning the rules of the large set of training data. The approach is to have a single decision system generated from a large and independent n subset of data. And then they developed a hybrid approach combining both genetic algorithm and decision tree to create an optimized decision tree are used to improving efficiency and performance of computation[1][2].

All the algorithms reviewed in this paper. So the comparison of all the techniques can be made based on their characteristic and their Time complexity.

Based on the Time complexity and Resources required to implement the algorithm MapReduce is the efficient algorithm for accessing the large data. Because it uses the Divide and Conquer approach in parallel manner to solve the problem. The next section sums up the discussion about the MapReduce algorithm.

Table I Comparison of all Algorithms based on their characteristics and time complexity

S.No	TECHNIQUE	CHARACTERISTIC	SEARCH TIME	Parallel processing and Distributed Environment Support
1	R-tree	Minimization of both coverage and overlap is crucial	O(3D)	NO
2	R*-tree	Improved split heuristic produces pages that are more rectangular and thus better for many applications		NO
3	Nearest Neighbour Search	Optimization problem for finding closest (or most similar) points	Grows exponentially with the size of the searching space(dn log n)	NO
4	Decision Tree Learning	Performs well with large datasets, Simple to understand and interpret.	Less time Consuming	Only Parallel processing
5	Decision Tree C4.5	Handling training data with missing attribute values, Practice local greedy search throughout dataset	Less time consuming	Only Parallel processing
6	GA Tree(Decision Tree + Genetic Algorithm)	It is a search heuristic that mimics the process of natural selection.	Improved performance problems like slow memory, execution time can be reduced	Only Parallel processing
7	Hierarchical Neural Network	High accuracy rate of recognizing data, Have high classification accuracy	Less time consuming improved performance	Only Parallel processing
8	MapReduce Technique	High accuracy rate of recognizing data and it is mostly used in parallel systems.	Less time compared to all techniques	Both parallel processing and Distributed support

IV. MAPREDUCE TECHNIQUE

MapReduce is a parallel batch job programming model. Map Reduce is mildly similar to Condor where users submit jobs, the developed Map and Reduce function is applied on a large data set, and the system handles all fault tolerance, scheduling, and distribution.

A MapReduce system is made out of a Map() strategy that performs separating and sorting, Such as sorting students by first name into lines, one line for every name. The Reduce() system that performs a synopsis operation, for example, including the quantity of students every line, yielding name frequencies. The MapReduce System arranges the handling by marshalling the dispersed servers, running the different assignments in parallel. What's more, it dealing with all interchanges and information exchanges between the different parts of the framework, and providing reliable mechanisms for redundancy and fault tolerance.

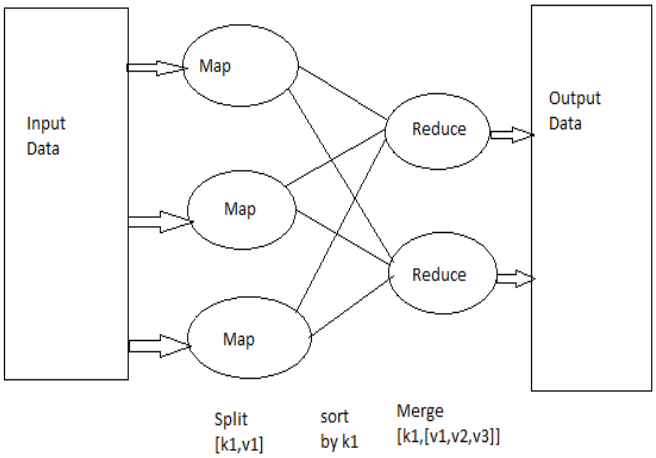

Fig 2: MR execution sequence

For implementation of MapReduce technique first we need to select Appropriate Join strategy. The selection of Join strategy depends on the application. In [4] the author has discussed the various Join strategies for Key-Value storage, based on the requirement the Join strategy is going to be selected. The All pair partition join is easy to implement and all compound partition transferred to reducers. And this is used when two datasets have more data in common. Broadcasting

join is used when one table is much smaller than the other table. Standard repartition join is easy to implement but all records from both tables will be buffered before joining that may lead to insufficient memory problem. Data in NoSQL database can be structured, semi-structured, or unstructured; and can be stored in many types of data structures such as indexed table of relational database, B-Tree, Queue, Hash.

Many methods were proposed for the implementation of MapReduce Technique. In [5] the authors proposed an efficient MapReduce technique called MR framework tells that the difference between map and reduce tasks: while most map tasks are simple and easy to parallelize, due to constraints of data locality and task complexity most reduce tasks are not easily decomposed. They assume that map tasks are preemptive and parallelizable, but none of the reduce task can be processed in parallel. And they proposed an optimal Algorithm for Makespan Minimization in Pre-emptive version but for Non-Preemptive version they are not proposed an optimal Algorithm. On total complete time minimization, for non-preemptive version they devise Non-preemptive Shortest Processing Time (NSPT) algorithm whose worst case ratio is 2−1/h, and for preemptive version they devise a heuristic named Jigsaw Shortest Processing Time (JSPT) based on extending McNaughton's wrap-around rule.

Next section sums up the discussion about the implementation of MapReduce algorithm in Distributed systems by using Hadoop Framework.

V. HADOOP AND HDFS

Hadoop is an open source usage of Google's Map/Reduce framework. It enables distributed, information intensive and parallel applications by decomposing a massive employment into smaller tasks and a massive information set into smaller partitions such that every task processes an alternate parcel in parallel. Hadoop uses Hadoop distributed File System (HDFS) which is an open source execution of the Google File System (GFS) for storing information. Map/Reduce application primarily uses HDFS for storing information. HDFS is a substantial distributed record system that uses item equipment and provides high throughput as well as adaptation to non-critical failure. Numerous enormous enterprises trust that inside a couple of years more than half of the world's information will be stored in Hadoop. HDFS stores files as a series of blocks and are reproduced for adaptation to internal failure. Strategic information parceling, processing, layouts, replication and arrangement of information blocks will increase the execution of Hadoop and a ton of research is going ahead here[9][10].

In [6] the author has examined the design and architecture of Hadoop's MapReduce framework. Particularly, there analysis has focused on data processing inside Reduce Tasks. They reveal that there are several critical issues faced by the existing Hadoop implementation, including its merge algorithm, its pipeline of shuffle, merge, and reduce phases, as well as its lack of portability for multiple interconnects[11][12]. They designed and implemented Hadoop-A as an extensible acceleration framework that can allow plug-in components to address all these issues. Because of the use of network-levitated merge algorithm, it can significantly reduce disk accesses during Hadoop's shuffling and merging phases, there by speeding up data movement. Furthermore, they have quantified the performance benefits of network-levitated merge and the RDMA protocol, respectively, on the Hadoop MapReduce. For Hadoop-A architecture see paper [6].

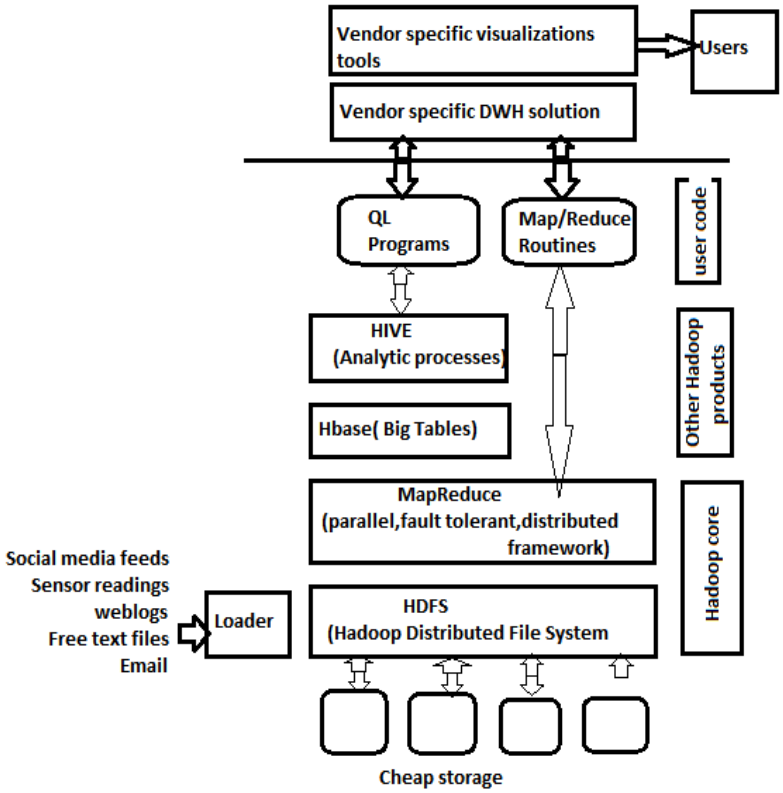

Fig 3: Hadoop Architecture

VI. ANALYSIS OF MAPREDUCE JOINS

A. Reduce-Side Join

In this algorithm, the actual join happens on the Reduce side of the framework. The "map" phase just pre-processes the tuples of the two datasets to arrange them in terms of the join key. Reduce side joins are easier to execute as they are less efficient than map-side joins because it require the information to be sorted and divided the same way. They are less productive than maps-side joins because the datasets need to experience the sort and shuffle phase[3].

B. Map-Side Join

The Reduce-Side join seems like the natural way to join datasets using Map/Reduce. It uses the framework's built-in capability to sort the intermediate key-value pairs before they reach the Reducer. But this sorting often is a very time consuming step. Hadoop offers another way of joining datasets before they reach the Mapper. This functionality is present out of the box and is arguably is the fastest way to join two datasets using Map/Reduce, but places severe constraints on the datasets that can be used for the join.

Table III Limitation of Map-Side Join

Limitation	Reason
Same comparator need to be used for all the datasets.	The sort ordering of the data in each dataset must be identical for datasets to be joined.
Same practitioner need to be used for all the datasets.	A given key has to be in the same portion in each dataset so that all parts that can hold a key are joined together.
The number of partitions in the datasets must be identical.	A given key has to be in the same partition in each dataset so that all partitions that can hold a key are joined together.

These constraints are quite strict but are all satisfied by any output dataset of a Hadoop job. Hence as a pre-processing step, we simply pass both the dataset through a basic Hadoop job. This job uses an Identity Mapper and an Identity Reducer which do no processing on the data but simply pass it through the framework which partitions, groups and sorts it. The output is compliant with all the constraints mentioned above. Although the individual map tasks in a join lose much of the advantage of data locality, the overall job gains due to the potential for the elimination of the reduce phase and/or the great reduction in the amount of data required for the reduce. This algorithm also supports duplicate keys in all datasets [5].

C. Broadcast Join

If one of the datasets is very small, such that it can fit in memory, then there is an optimization that can be exploited to avoid the data transfer overhead involved in transferring values from Mapper to Reducers. This kind of scenario is often seen in real-world applications. For instance, a small user's database may need to be joined with a large log. This small dataset can be simply replicated on all the machines. This can be achieved by simply using –files or -archive directive to send the file to each machine while invoking the Hadoop job. Broadcast join is a Map-only algorithm.

VII. RESULTS AND DISCUSSION

A. Experiment 1: Performance of two-way joins on increasing input data size

A series of experiments were conducted to compare the runtime and scalability of the various algorithms. The first of these involved running the three two-way algorithms mentioned earlier on increasing data sizes using the cluster. Both datasets involved in the joins had the following characteristics.

- Same number of tuples
- Same Key space

We tested on two kinds of key distributions. In the case of skewed data, one of the datasets had a key that was repeated 50% of the times. Which means that given n rows, n/2 of them were the same key? The other dataset had uniform key distribution concentrating first on the Uniform Key Distribution graph, it can be seen that all the three algorithms gave comparable numbers at lower input data sizes. But Map-side join always performed the best, even with large input datasets. The below table shows the working of algorithms for the input with 1 million rows when then input dataset had skewed key distribution. This can be attributed to the unequal partitioning that will occur due to the skew.

Table IIIII Comparison of Time Complexity of MapSide Join and Reduce Side Join

Tuples(*100000)	Reduce Side Joins(seconds)	MapSide Join(seconds)
1	15	10
5	24	14
10	31	18

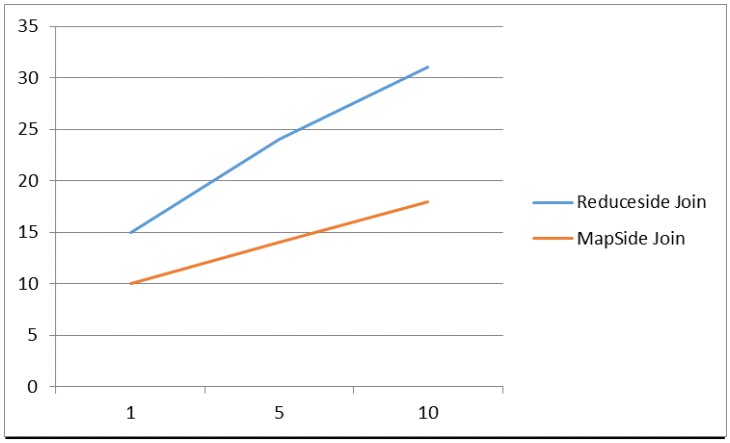

Fig. 4 Analysis of MapSide Join and ReduceSide Join

In the above graph X-axis represents the Number of tuples in multiple of lakhs and Y-axis represents the Time taken by the algorithm in seconds.

B. Experiment 2 - Performance of Broadcast Join

The major use-case of the Broadcast Join is when one of the datasets is small in comparison to the other. So we decided to test this scenario by keeping one of the datasets of constant size and increasing the size of the other. The dataset characteristics were as follows

- The larger dataset size was fixed at 1,000,000 tuples
- The size of the smaller dataset was varied from 100,000 tuples to 1,000,000 tuples

Broadcast Join gave performances that were very close to the ones by Map-Side Join, but without any pre-processing step involved. Performance of Reduce-Side Join was the worst when compared to the other two algorithms.

Table IVV Comparison of Time Complexity of MapSide Join and Reduce Side Join with Broadcast Join

Tuples (*100000)	**Reduce Side Joins(seconds)**	**MapSide Join (seconds)**	**Broadcast Join (seconds)**
1	15	10	13
5	24	14	15
10	31	18	17

Broadcast Join gave performances that were very close to the ones by Map-Side Join, but without any pre-processing step involved. Performance of Reduce-Side Join was the worst when compared to the other two algorithms.

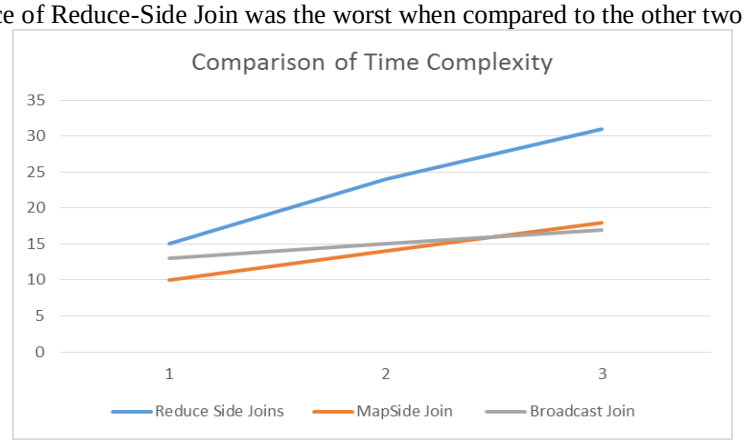

Fig. 5 Analysis of MapsideJoin and ReducesideJoin With respect to Broadcast Join

In the above graph X-axis represents the Number of tuples in multiple of lakhs and Y-axis represents the Time taken by the algorithm in seconds.

VIII. CONCLUSION

Because of Increase in the amount of information in the field of genomics, meteorology, science, natural research, it becomes hard to handle the information. In this paper we discussed about the various Algorithms like R-tree, Decision tree learning, R*-tree, Nearest Neighbor Search, Decision Tree C4.5, GA Tree(Decision Tree + Genetic Algorithm), Decision Tree C4.5,Hierarchical Neural Network and MapReduce Technique which are used for accessing the Big Data. We compared those Algorithms based on their characteristics and Time complexity. Among all the algorithms we

selected MapReduce Technique for Accessing the Data because it has advantages over all the algorithms and it uses parallel batch processing model for solving the problem. For implementation of MapReduce technique first we need to select appropriate Join strategy. In this paper we discussed about the various Join strategies. The selection of Join strategy depends on the application. And then we discussed about the open source implementation of MapReduce programming model in Distributed systems called Hadoop Framework which is most reliable framework and it is mostly used for Distributed computing of big data. We found some quite interesting results from our experiments on Different Join strategies using Map/Reduce framework. We found that Map-Side joins performs the best when the datasets have relatively uniform key distribution. But this comes at the cost of a pre-processing step. Reduce-Side Join gave almost the same performance for both uniform and skewed key distributions whereas Map-Side Join seemed to give slightly unpredictable results with skewed data.

REFERENCES

[1] ChanchalYadav, ShuliangWang ,Manoj Kumar, "Algorithm and approaches to handle large Data- A Survey", IJCSN,ISSN-2013.
[2] Edmon Begoli, James Horey, "Design Principles for Effective Knowledge Discovery from Big Data".
[3] Duong Van Hieu, Sucha Smanchat, and Phayung Meesad, "MapReduce Join Strategies for Key-Value Storage". IJCSSE-2014 Technologies & Research (IJMCTR), 2013. 1: p. 12-18.
[4] Yuqing Zhu, Yiwei Jiang, Weili Wu, Ling Ding, AnkurTeredesai, Deying Li, and Wonjun Lee. "Minimizing Makespan and Total Completion Time in MapReduce-like Systems". IEEE INFOCOM 2014 - IEEE Conference on Computer Communications.
[5] Weikuan Yu, Member, IEEE, Yandong Wang, and Xinyu Que, "Design and Evaluation of Network-Levitated Merge for Hadoop Acceleration". IEEE INFOCOM 2014 - IEEE Conference on Computer Communications.
[6] Andreas Antoniades, Clive Cheong Took, "A Google Approach for Computational Intelligence in Big Data". IEEE INFOCOM 2014 - IEEE Conference on Computer Communications.
[7] Tyson Condie, Neil Conway, Peter Alvaro, Joseph M. Hellerstein, "MapReduceOnline". Yahoo! Research.
[8] Innocent Mapanga, Prudence Kadebu, "Database Management Systems: A NoSQL Analysis", International Journal Modern Communication Technologies & Research(IJMCTR) ISSN: 2321-0850, Volume-1, Issue-7, September 2013.
[9] P Beaulah Soundarabai, Thriveni J, K R Venugopal, L M Patnaik, "AN IMPROVED LEADER ELECTION ALGORITHM FOR DISTRIBUTED SYSTEMS", International Journal of Next-Generation Networks (IJNGN) Vol.5, No.1, March 2013.
[10] Rachna Gajre, Dr. Leena Ragha, "Optimized Bully Election Method for Selection of Coordinator Process and Recovery of Crashed Process", International Journal of Scientific and Research Publications, Volume 3, Issue 5, May 2013.
[11] Mohammad Asif Khan, Zulfiqar A. Memon, Sajid Khan," Highly Available Hadoop NameNode Architecture" International Conference on Advanced Computer Science Applications and Technologies, 2012, IEEE 2013.
[12] Dhruba Borthakur, "The Hadoop Distributed File System:Architecture and Design," in Apache Software foundation,http:/ hadoop.apache.org/common/docs/r0.18.0/hdfs _design.pdf .
[13] The Apache Software Foundation. http:// http://hadoop.apache.org, Last release of March, 2012.
[14] Tom White, "Hadoop: The Definitive Guide". O'Reilly Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, 2009.
[15] Arun C. Murthy," Next Generation of Apache Hadoop MapReduce", Hortonworks Inc. 2011.
[16] Chen He, Ying Lu, David Swanson," Matchmaking: A New MapReduce Scheduling Technique", IEEE International Conference on Coud Computing Technology and Science, 2011.
[17] J. Dean and S. Ghemawat. "MapReduce: Simplified Data Processing on Large Clusters". Commun. ACM, 51(1):107–113, 2008.
[18] Fair Scheduler, http://hadoop.apache.org/mapreduce/docs/r0.21.0/fair_scheduler.htm
[19] Capacity Scheduler http://hadoop.apache.org/common/docs/r0.19.2/capacity_scheduler.html

