International Journal of Future Innovative Science and Engineering Research (IJFISER)
Volume - 2, Issue - II ISSN (Online): 2454- 1966

Research Manuscript Title

EFFICIENT QUERY HANDLING ON BIGDATA BY USING FASTRAQ

S.Hamsareka, Dr.B.Kalavathi

PG Scholar, Professor,

Dept of CSE,

K.S.R Institute for Engineering and Technology, Tamil Nadu, India

E-Mail: rekaselvam1993@gmail.com

JUNE – 2016

www.istpublications.com

EFFICIENT QUERY HANDLING ON BIGDATA BY USING FASTRAQ

S.Hamsareka, Dr.B.Kalavathi

PG Scholar, Professor,

Dept of CSE,

K.S.R Institute for Engineering and Technology, Tamil Nadu, India

E-Mail: rekaselvam1993@gmail.com

ABSTRACT

Big data is a broad term for huge data sets that traditional data processing applications are inadequate. Big data analysis can discover trends of various social aspects and their preferences of individual everyday behaviors .The main challenging factor is processing large amount of data within a time period .The query processing time is increased then the network communication cost and local files scanning cost can be increased simultaneously. In our existing system they use hive in range aggregate query but this could provide inaccurate results in big data environments .To overcome this limitations we propose a new technique called FASTRAQ- Range Aggregate Queries. FastRAQ first divides big data into different independent partitions with a balanced partitioning algorithm, and then generates a local estimation for each partition. If a range-aggregate query request arrives, FastRAQ obtains the result directly by summarizing local estimates from all partitions. Fast Range Aggregate Queries has time complexity of 0(1) for data updates. FastRAQ improve the performance by reducing the query processing time and cost of network communication and local file scanning. FastRAQ provides more accurate results in big data environments when compare to hive.

Key Terms: FASTRAQ, Hadoop, HDFS, Map Reduce, Cardinality, Hive.

I .INTRODUCTION

Big data is a buzzword or huge dataset used to describe a massive volume of both structured and unstructured data. Big data is so large and this can be difficult to process using traditional database and software techniques. The most enterprise circumstance the volume of data is too big or it moves too fast or it exceeds current processing capacity. In contempt of these problems, big data has the prospective to help in many industries or companies to improve operations and make faster, more intelligent decisions. Big data needs exceptional technologies to efficiently process large quantities of data within endurable elapsed times. The suitable technologies are crowd sourcing, data fusion and integration, genetic algorithms, natural language processing(NLP), machine learning, signal processing, simulation, time series analysis and visualization. Multidimensional big data can also be represented as tensors that can be more efficiently handle by the tensor-based computation, such as multi linear subspace learning.
Hive provides a mechanism to assignment structure onto this data and query the data by using a SQL-like language called HiveQL[1].

A.HADOOP

Hadoop is an open-source framework that can store and process big data in a distributed environment across clusters of computers by using the simple programming models. It is designed to scale up from single servers to thousands of machines, this will providing local storage and computation.

- ***Large datasets*** → Terabytes or petabytes of data
- **Large clusters** → hundreds or thousands of nodes

Hadoop framework contains two main layers

- Hadoop Distributed file system (HDFS)
- Execution engine (Map Reduce)

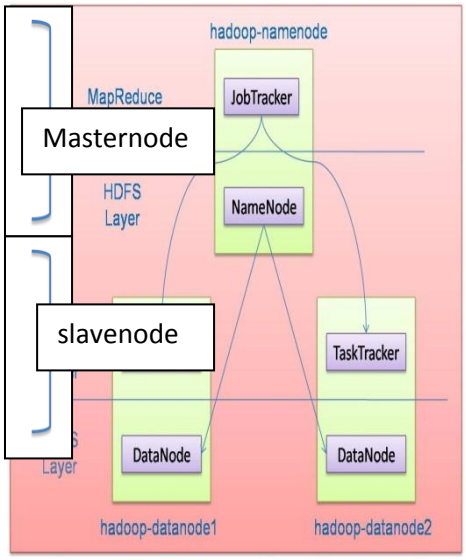

Fig 1.1: Hadoop architecture

HDFS cluster have a single **Name node**, a master server that manages the file system namespace and regulates access to files by clients. There are number of **Data Nodes** usually one per node in a cluster. The Data Nodes manages the storage attached to the nodes that they can run on. In fig [1] HDFS reveal a file system namespace and allows user to be store data in files.

B.MAP REDUCE

The Map Reduce is the programming model and a framework for data-intensive distributed computing used in batch mode processing. The key idea for the Map Reduce model is to allow the users to focus on data processing mechanisms and hide some aspect of parallel execution.

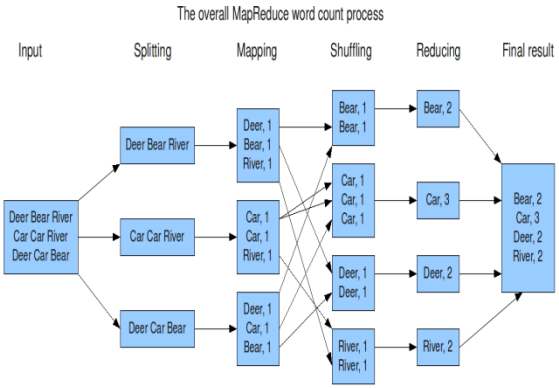

Fig 1.2: Map Reduce framework

The Map Reduce programming model has developed as a famous way to combine the large clusters of computers together. It mainly emphasis on applying transformations to sets of data records and permits the details of distributed execution, network communication, coordination and fault tolerance to be managed by the Map Reduce framework.

II.RESEARCH OBJECTIVE

Aggregation is an important operation in today's relational database systems. The data set is larger, user become more experienced, there is an increasing significance on extracting not just specific data items, but also general characterizations of large subsets of the data. Users can aggregate

information in right away, even though producing it may involve accessing and compressing huge amounts of information.

Aggregation must be performed online, it allow the users both to observe the progress of their queries, and to control execution on the fly. Online aggregation provides performance along with much more. Ending reputation is easily acquired by user in an online aggregation system, simply by canceling the process of relevant accuracy level or time. Online aggregation systems afford the user more control than sampling systems however, since stopping conditions can be modified while the query is running.

Range sum queries on data cubes are authoritative analysis tool. Existing techniques for range sum queries on data cubes, however, can acquire update costs in the order of the size of the data cube. Since the size of a data cube is expanding in the number of its dimensions, reconstructing the entire data cube can be very costly and is not practical. To cope with dynamic data cube problem, a new approach has been introduced recently, which achieves constant time per range sum query while constricting each update cost within $O(n^{d/2})$, where d is the number of dimensions of the data cube and n is the number of specific values of the domain at each dimension.

The range query sum problem on the data cubes have not been received to earn attention. The problem by presenting an exquisite algorithm for it, based on the prefix sum approach. The main idea of this method is to pre compute prefix sum of cells in the data cube, which can be used to answer the ad hoc range queries at run-time. The prefix sum approach is not applicable for large data cubes which having many dimensions, because the update time is in the order of the size of the data cube which expands exponentially with the number of dimensions ‘d’ of the data cube. Thus, it leaves us a challenging problem that is whether there is a capable algorithm which has an acceptable update time and small query time. Uniform-random sampling is impossible except for very small sample sizes. Block-level sampling is far more capable than true uniform-random sampling over a large database. However, accurate uniform-random sampling can be quite expensive.

A technique proposed where the Map Reduce concept is introduced into the MongoDB with NoSQL as a back end to implement the online aggregation. The Online aggregation uses the Map Reduce concept to get the comparative results using the MongoDB and NoSQL.

Cardinality estimation is the task of regulating the number of specific elements in a data stream. While the cardinality can be easily measured using space linear for many applications and is completely impossible and needs too much memory. Many algorithms have been proposed in the past, and the HyperLogLog algorithm is one of them.

The HyperLogLog algorithm uses randomization to approximate the cardinality of a multiset. HyperLogLog has the useful property that not the full hash code is required for the computation. HyperLogLog- Plus uses 64 bits hash function instead of 32 bits in HyperLogLog to improve the data-scale and estimated accuracy in big data environments.

III.RELATED WORK

The previous solutions mainly focus on improving performance by reducing time and file scanning cost.

A.FastRAQ framework

In FastRAQ, the attribute values can be numeric or alphabetic. One example of the range aggregate problem is shown as follows:

Select exp (AggColumn), other ColName where
li1 < ColNamei < li2 opr
lj1 < ColNamej < lj2 opr. . . ;

In the above query, expression is an aggregate function such as SUM or COUNT; AggColumn is the dimension of the aggregate operation; li1 < ColNamei < li2 and lj1 < ColNamej < lj2 are the dimensions of ranges queries; opr is a logical operator including AND, OR logical operations.

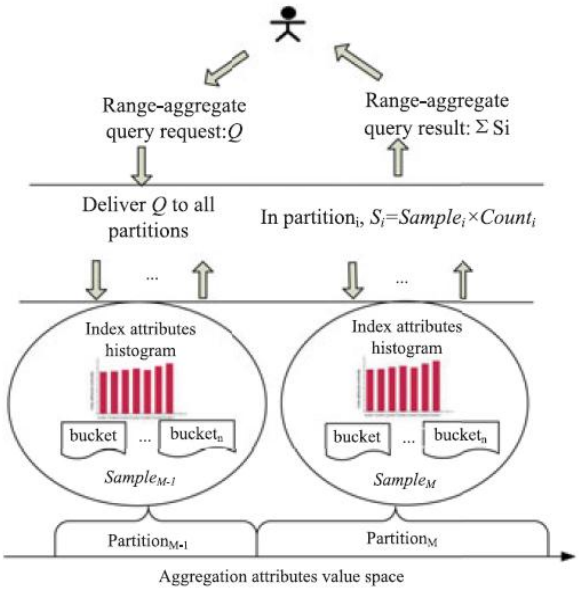

Fig 3.1: The FastRAQ framework.

The distributed range-aggregate queries cost primarily includes two parts. That is cost of network communication and the cost of local files scanning. The first cost is produced by data transmission and synchronization for aggregate operations when the selected files are stored in different servers. The second cost is produced by scanning local files on searching the selected tuples. The size of a data set increases continuously, the two types of cost will also increase simultaneously. Only when the two types of cost are minimized, can we obtain faster final range-aggregate queries results in big data environments.

B.RC tree

A hierarchical tree structure has been established to implement the histogram. A typical index structure is range cardinality Tree. RC-Tree includes three types of nodes, which are root node, internal nodes, and leaf nodes. The root node or an internal node points to its children nodes and keeps their values of spreads, such as [pi, pj]. A leaf node is for one bucket in the histogram. The parameters in a leaf node are values of spreads for each bucket, for example [pi, pi+1], the estimator CE of each bucket, and the bucket data file pointer. The leaf node only keeps the statistical information, and tuples values are stored in bucket data files. In RC tree structure obtain the query results earlier.

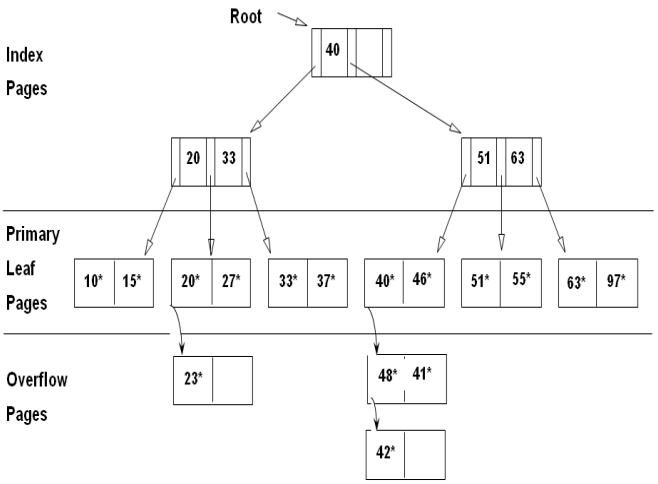

Fig 3.2: Example of RC tree structure

C. Distributed partitioning algorithm

The partitioning step has become a key determinant in data analysis to boost the query processing performance. All of these works enable each partition to be processed independently and more efficiently. Stratified Sampling is a method of sampling from independent groups of a population, and selecting sample in each group to improve the representativeness of the sample by reducing sampling error. Build the partitioning algorithm based on the idea of stratified sampling to make the maximum relative error under a threshold in each partition. At the same time, the sum of the local result from each partition can also achieve satisfied accuracy for any ad-hoc range-aggregate queries. The partition algorithm can be expressed as follows for data sets R:

$$\text{Partitioning}(R) = (g, p) = (V_e, \text{random } [1, V_r])$$

IV.EXISTING SYSTEM

In existing system, the system proposes a fast approach to range-aggregate queries in big data environments. FastRAQ first divides big data into several independent partitions with a balanced partitioning algorithm, and then generates a local estimation sketch for each independent partition. When a range-aggregate query request arrives, FastRAQ acquires the solution directly by summarizing local estimates from all partitions. It can measure the quality of tuples distributions more accurately and can support accurate multi-dimensional cardinality queries. A balanced partition algorithm is used first to divide big data into several independent partitions, then generates local estimation sketch for each partition. FastRAQ provides result by summarizing local estimation from all partitions. FastRAQ can give good starting points for real-time big data. It deals with 1: n format range-aggregate query problem, but m: n formatted problem still outside there. The sampling of data is necessary for the analysis on the big data as the data is present in huge amount.

Sampling has various methods one most famous method is stratified sampling in which sampling independent groups and select only one sample for improvement and reducing errors. It has uncontrolled chunks which are balanced with the help of a balanced partition algorithm. The system use the common K-Means clustering method to analyze the vectors set and produce K clusters. A unique Cluster ID is assigned to each cluster. We construct a list of key-value pairs from the result of K clusters. The key-value pairs are in the format of <tag, Cluster ID >. We sort the key-value pairs by tag in alphabetical order. The buckets in the histogram are built from the sorted pairs. The key idea is to merge the pairs with the same Cluster IDs into the same bucket. If some tag occurring frequency is significantly different from others, its Class ID is different after the K-Means clustering, and it will be put into an independent bucket in the histogram. FastRAQ uses approximate answering approaches, such as sampling, histogram, and cardinality estimation etc., to improve the performance of range-aggregate queries. We use relative error as a statistical tool for accuracy analysis. Relative error is widely used in an approximate answering system. Also, it is easy to compute the relative errors of combined estimate variables in a distributed environment for FastRAQ. In this section, we analyze the estimated relative error and the confidence interval of final range-aggregate query result.

V.PROPOSED SYSTEM

In my proposed work construct the private cloud with multiple nodes. When the data is send from the server system it can be replicated in multiple client systems.

Here consider four scenarios which can be implemented in Linux platform to compare and improve the performance when the dataset is too large.

A. Data fixed how the Query will change
B. Query fixed how the Data will change
C. Both are in not fixed (worst case)
D. Both are fixed

In proposed system, the system proposes a fast approach to range-aggregate queries in big data environments. FastRAQ first divides big data into independent partitions with a balanced partitioning algorithm, and then generates a local estimation sketch for each partition. When a range-aggregate query request arrives, FastRAQ obtains the result directly by summarizing local estimates from all partitions. It can measure the quality of tuples distributions more accurately and can support accurate multi-dimensional cardinality queries. The system proposed FastRAQ- big data query execution in a range-aggregate queries approach.

A balanced partition algorithm is used first to divide big data into independent partitions, then local estimation sketch generated for each partition. FastRAQ gave result by summarizing local estimation from all partitions. The Linux platform is helpful for implementing FastRAQ and performance evaluated on billions of data records. According to the authors, FastRAQ can give good starting points for real-time big data. It solves the 1: n format range-aggregate query problem, but m: n formatted problem still outside there.

VI.CONCLUSION

FastRAQ can solve the 1:n format range aggregate queries problem, there is one aggregation column and n index columns in a record. FastRAQ is now running in homogeneous environments. Here also consider the mentioned scenarios to improve the performance by reducing the query processing result and implement this in heterogeneous environments.

REFERENCES

[1]. A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, N. Zhang, S. Antony, H. Liu, and R. Murthy, “Hive—a petabyte scale datawarehouse using Hadoop,” in Proc. IEEE 26th Int. Conf. Data Eng.,2010, pp. 996–1005.

[2]. Hadoop MapReduce distribution.Available: http:// hadoop.apache.org, 2015.

[3]. W. Liang, H. Wang, and M. E. Orlowska, “Range queries in dynamic OLAP data cubes,” Data Knowl. Eng., vol. 34, no. 1,pp. 21–38, Jul. 2000.

[4].Ying Pei, Jungang Xu, Zhiwang Cen, Jian Su,”IKMC: An Improved K-medoids Clustering Method for Near-duplicated Records Detection”, 2009 IEEE conference.

[5].Mehrdad Mahdavi. Hassan Abolhassani,”Harmony K-means algorithm for document clustering”, 11 December 2008 Springer Science+Business Media, LLC 2008.

[6].Prasadkumar Kale et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 6 (3) , 2015, 2871-2875.

[7].S. Chaudhuri, G. Das, and U. Srivastava, “Effective use of blocklevel sampling in statistics estimation,” in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2004, pp. 287–298.

[8] Xiaochun Yun, Guangjun Wu, Guangyan Zhang, Keqin Li, and Shupeng Wang, ―FastRAQ: A Fast Approach to Range-Aggregate Queries in Big Data Environments‖, IEEE Transactions On Cloud Computing, Vol. 6, No. 1, January 2014.

[9] Charles L. Forgy ,―Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem‖, Artificial Intelligence ,1982.

[10] Saba Sehrish, Grant Mackey, Pengju Shang, Jun Wang,―Supporting HPC Analytics Applications with Access Patterns Using Data Restructuring and Data-Centric Scheduling Techniques in MapReduce‖, IEEE Transactions on Parallel and Distributed Systems, Vol. 24, No. 1, January 2013.

[11] Hasan M. Jamil, ―Mapping Abstract Queries to Big Data Web Resources for On-the-fly Data Integration and Information Retrieval‖, IEEE ICDE Workshops 2014.

[12] Y. Shi, X. Meng, F. Wang, and Y. Gan, "You can stop early with cola: Online processing of aggregate queries in the cloud," in Proc.21st ACM Int. Conf. Inf. Know. Manage. 2012, pp. 1223–1232.

[13] K. Bilal, M. Manzano, S. Khan, E. Calle, K. Li, and A. Zomaya, "On the characterization of the structural robustness of data center networks," IEEE Trans. Cloud Comput., vol. 1, no. 1, pp. 64–77,Jan.–Jun. 2013.

[14] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, "Integrity for join queries in the cloud," IEEE Trans. Cloud Comput., vol. 1, no. 2, pp. 187–200, Jul.–Dec. 2013.

[15] J. Dean and S. Ghemawat, ―Mapreduce: simplified data processing on large clusters, Commun. ACM, vol. 51, no. 1, pp. 107–113, Jan. 2008.

[16] BinaKotiyal, Ankit Kumar, Bhaskar Pant, RH Goudar, ―Big Data: Mining of Log File through Hadoop.