M – Age: A Framework to Process Range Aggregate Query in Big Data using MongoDB

Nandhini N
M.Tech, IT
AMC Engineering College
Bangalore,India

Pavithra Sridhar
Asst .Professor, Dept of ISE
AMC Engineering College
Bangalore, India

Abstract*—Range searching is a fairly well-structured problem in computational geometry. Big Data deals with class of problems called Range Aggregate Query problems, the aim is to deal with some composite queries involving range searching, where one needs to do more than simple range reporting or counting.A range query applies an aggregate function over all selected cells of an OLAP data cube. The essential idea is to precompute some auxiliary information that is used to answer ad hoc queries at runtime. In order to analyse and process range aggregate query M-AGE : A framework is proposed in this paper. Existing approaches were dealt only with adhoc queries and results yielded were not satisfactory. Here M-AGE is implemented on linux platform and performance is evaluated on very large park data records .M-AGE supports range queries and also runs multiple servers. When a range aggregate query arrives it is split based on the Balanced Partitioning algorithm and distributed across multiple shards(A shard is nothing but a master with one or more slaves).Queries here return specific fields of documents and also includes user defined JavaScript functions. JavaScript is used in queries ,aggregate functions(such as MapReduce)and sent directly to the MongoDB to be executed. M-AGE has O(1) time complexity for the updates of data and time complexity for range aggregate queries where N happens to be the unique tuples,P happens to be the partition number B happens to be the bucket in each of the histogram.This M-AGE framework there by reduces the cost of both network communication and local file scanning and has better performance compared to hive.

Key Words*— *Big Data, MapReduce, MongoDB, Multiple Servers, Range Aggregate Query.

I. INTRODUCTION

We are now living in the digital world.With the increase in digitization the amount of structured and unstructured data being created and stored is exploding. The data is generated from various sources – transactions, social media, sensors, digital images, videos, audios and click streams for domains including healthcare, retail, energy and utilities. It is increasing becoming imperative for organisations to mine this data to stay competitive. The volume, variety, and velocity of Big Data[1] causes performance problems when created, managed and analysed using conventional data processing techniques. Huge information examination is carried out to find out patterns of different social perspectives and inclinations of different individual practice routines. This forms a platform to investigate crucial inquiries concerning the mind boggling world. These included related works to assemble a productive speculation procedure, and investigated the enormous behavioural information sets identified with account and returned a benefit of even 326 percent higher than that of an arbitrary venture methodology. Choi and Varian[2] introduced gauge representations to figure monetary markers, for example, social unemployment, vehicles deal, and even destinations for individual voyaging. It is now vital and required to give proficient strategies and devices to enormous information investigation.So in processing this large quantities of data a main problem arises which is to make a summary which deals with approximate query answering. Random Sampling is yet another method which yields flexible summaries and supports subset-sum queries and confidence bounds. But when Classic sample-based summaries[3] are concerned which are primarily designed for arbitrary subset queries take into account the structure of the keys.So it can be understood here that the specific structure,such as hierarchy, order or product space makes range queries more relevant for Big data analysis. Range aggregate queries play an important role in OLAP(On-line analytical Processing Systems) and GIS(Geographic information Systems) in summarising information. Range aggregate queries are also used as an important tool in decision management, online suggestion, trendestimation and so on. So it is a challenging task to estimate and give accurate results for range aggregate queries in big data environments. Earlier Prefix-sum cube method[5] was used in OLAP to increase the performance of range aggregate queries along with Online aggregation. But with these approaches users cannot obtain a satisfactory answering with approximate accuracy as early as in the beginning stages.

II. RELATED WORK

The range-aggregate query problem has been studied by Sharath Kumar and Gupta[3] and Malensek [4] in computational geometry and Geographic Information Systems(GIS).The work is primarily focused on approximating range aggregate query for real-time data analysis in OLAP. Ho et al. was the first to present Prefix-Sum Cube approach to solving the numeric data cube aggregation [4] problems in OLAP. The essential idea of PC is to pre-compute prefix sums of cells in the data cube, which then can be used to answer range-aggregate queries at run-time. However, the updates to the prefix sums are proportional to the size of the data cube. Liang et al. [6] proposed a dynamic data cube for range-aggregate queries to improve the update cost. The prefix sum approaches are suitable for the data which is static or rarely updated. For big data environments, new data sets arrive continuously, and the up-to-date information is what the analysts need. The PC and other heuristic pre-computing approaches are not applicable in such applications. An important approximate answering approach called Online Aggregation was proposed to speed range-aggregate queries on larger data sets [7]. OLA has been widely studied in relational databases [8] and the current cloud and streaming systems [9], [10]. Some studies about OLA have also been conducted on Hadoop and MapReduce [10], [11], [12]. The OLA is a class of methods to provide early returns with estimated confidence intervals continuously. As more data is processed, the estimate is progressively refined and the confidence interval is narrowed until the satisfied accuracy is obtained. But OLA can not respond with acceptable accuracy within desired time period, which is significantly important on the analysis of trend for ad-hoc queries.

III. EXISTING SYSTEM

The FastRAQ system is an approximate answering approach that summarises accurate estimations quickly for range aggregate queries in environments involving big data. It first divides the data chunks into separate independent partitions by using a balanced partitioning algorithm, and by which a local estimation for each partition is generated. So on the arrival of the range aggregate query , FastRAQ obtains the result by summarising the local estimation of every individual problem. The estimation sketch is a multi-dimensional histogram which is built via the learned data distribution. In FastRAQ, the attribute values can be both numeric and alphabetic. The Key idea is so generate a local query result using the balanced partitioning algorithm with the stratified sampling model. So in FastRAQ the numerical space of the aggregation – column is divided into different groups and an estimation sketch of the group is obtained. When a new record arrives it sis sent to the partition depending on the current data distributions and the number of servers available. . It first divides the data chunks into separate independent partitions by using a balanced partitioning algorithm, and by which a local estimation for each partition is generated. So on the arrival of the range aggregate query , FastRAQ obtains the result by summarising the local estimation of every individual problem.

A brief of the FastRAQ is shown in Fig 1.

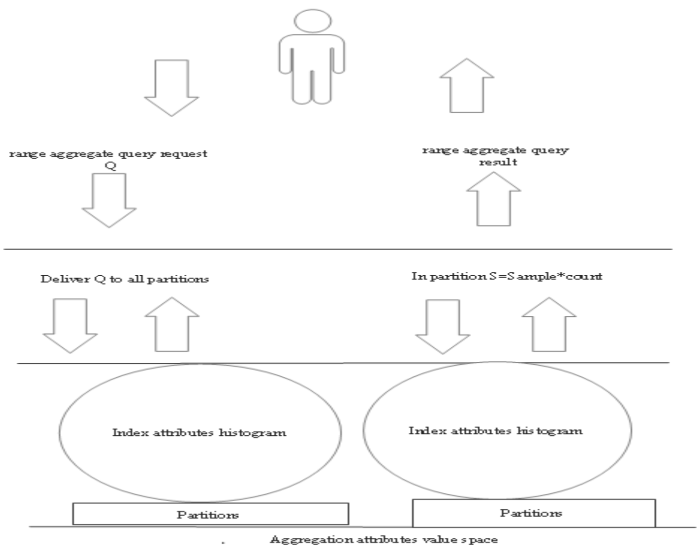
range aggregate query request
Q
range aggregate query result
Deliver Q to all partitions
In partition S=Sample*count
Index attributes histogram
Index attributes histogram
Partitions
Partitions
Aggregation attributes value space

Fig .1.FastRAQ framework.

Segment family construction for FastRAQ, which incorporates three sorts of section families identified with extent total questions. They are accumulation section family, list segment family, and default segment family. The collection segment family incorporates an accumulation segment, the file segment family incorporates different list sections, and the default column-family incorporates different segments for further augmentations. A SQL-like DDL and DML can be characterized effectively from the blueprint.

A. Disadvantages

- Cost is produced by data transmission and synchronization for aggregate operations.
- Scanning og local files to search selected tuples.
- The updating process includes delivering the record each time to the specified partition.
- Range Cardinality tree produces additional overhead.
- Cost of transmitting the local result of a partition cannot be negligible.

IV. PROPOSED SYSTEM

Here in the proposed component, Ioutline an adjusted segment calculation which works with stratified testing model. In every partition, a specimen for estimations of the accumulation section and a multi-dimensional histogram is kept for estimations of the file segments. At the point when a reach total question demand arrives, the nearby result is the result of the specimen and an expected cardinality from the histogram. This decreases the two sorts of cost all the while. Earlier FastRAQ gives a decent beginning stage to growing continuous noting strategies for huge information investigation. At the point when an inquiry demand arrives, it is conveyed into each partition. The cardinality estimator (CE) is formed for the questioned territory from the histogram in every parcel. At that point we the appraisal esteem is computed in every allotment, which is the result of the example and the evaluated cardinality from the estimator. Also I make use of the MongoDB here in the proposed component. M-AGE combines sampling, Histogram and data partition approaches to generate accurate estimations involving big data. It is designed for distributed range aggregate queries and it is shown to achieve better performance results on both query and update processing in big data. M-AGE basically comes with MongoDB. It is a cross platform document oriented database. MongoDB supports field, range queries, regular expression searches. Queries can return specific fields of documents and also include user-defined Java Script functions.MongoDB provides high availability with replica sets. A replica set consists of two or more copies of the data. Each replica set member may act in the role of primary or secondary replica at any time. The primary replica performs all writes and reads by default. Secondary replicas maintain a copy of the data of the primary using built-in replication. When a primary replica fails, the replica set automatically conducts an election process to determine which secondary should become the primary. Secondaries can optionally perform read operations, but that data is eventually consistent by default.

A. Advantages

- M-AGE can be used as a tool in DBaaS.
- It can be used to find solutions of m*n format problem. When there are m aggregation columns and n index columns of the same record.
- M-AGE achieves 26 times of performance improvement on count queries than Hive.
- Can apply on large data sets.
- M-AGE achieve better performance improvement on range aggregate queries than Hive.
- It can search different index-columns of queried ranges.
- The cost of merging due to union statements is negligible.

V. SYSTEM ARCHITECTURE OF PROPOSED SYSTEM

The data distributions are measured by the clustering values of all index-columns and they also make use of the learned knowledge to build what is known as the histogram. So here the feature vectors are extracted from the learned data set which will produce the vector set through which the final clusters are formed. The common K-means clustering method is used to produce the clusters. Each cluster is assigned with a unique ID.M-AGE supports multi-dimensional range queries which may include multiple buckets of the same histogram. It uses a unique ID for each record. The histogram is implemented as a hierarchical tree structure which is called as the range-cardinality tree(RC tree). A typical RC tree is depicted in fig 3.The RC-Tree[13] includes three types of nodes. They are root nodes, internal nodes and the leaf nodes. The root node or the internal node always points to its children nodes. A leaf node corresponds to one bucket in the histogram. The leaf node only keeps the information and the tuples values are always stored in the bucket files. Buckets are independent of each other, the RC-Tree structure and its construction process is quite similar to the B+ tree. To improve the throughput of RC-Tree, a hash table[14] for newly incoming data is introduced for incremental updating process. The hash table consists of multiple nodes which are identical to the RC-Tree's leaves nodes. If a new record is coming, it first writes into the hash table, creates node if it does not exist, and then appends the tuples values into a temporary data file. When the number of nodes in the hash table reaches a threshold, the hash table flushes nodes into the RC-Tree, and appends the temporary files to the formal bucket data files. The incremental updating process[15] will greatly improve the throughput of RC-Tree in big data environments. The following updating algorithm explains the incremental updating process in RC-Tree.

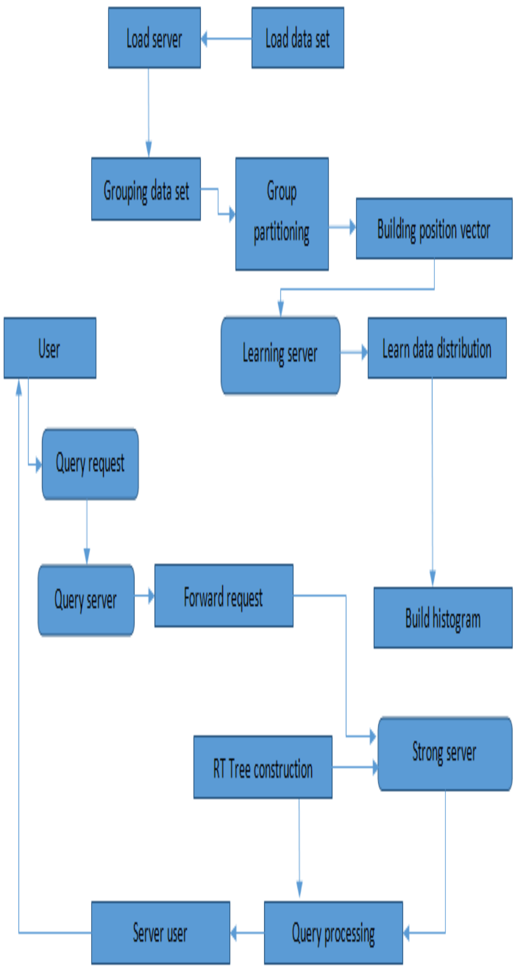

Fig.2. System Architecture of M-AGE

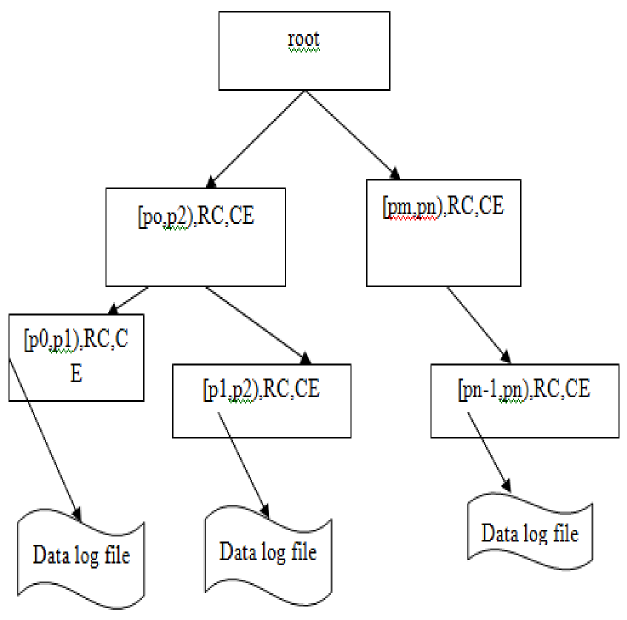

Fig.3. RC-Tree structure.

```
Algorithm 1:Grouping
Step 1: Parse value of index-columns into key-value pairs.
Step 2: Search in bucket spreads.
Step 3: Search in hash table and get the target node.
Step 4: End
```

To query cached data in hash table, the process is the same as Algorithm 2 to obtain cardinality estimator of the cached data, and then the result is merged to the estimator into CEmerge to compute the final cardinality estimation

```
Algorithm 2: Range Carnality Query algorithm.

Input:(Q,T,ho);
  Q: Select distinct count;
   T: the RC-Tree;
   ho :the edge range cardinality ratio.
Output : R;
   R:the range cardinality queried result.
Step 1: Locate the first node in RC-Tree by ColName;
Step 2:Scan the bucket data file;
Step 3:Merge into the cardinality estimator CEmerge;
Step 4: R<-h(CEmerge);
Step 5:return R.
```

VI. SYSTEM DESIGN

Partitioning is a process of assigning each record in a large table to a smaller table based on the value of a particular field in a record. It has been used in data center networks to improve manageability and availability of big data. The partitioning[13] step has become a key determinant in data analysis to boost the query processing performance[14]. The number of partitions should be kept under some threshold in an applicable system. In big data environments, a partition is a unit for load balancing and local range-aggregate queries. In each partition, a dynamic sample is calculated from the current loaded records. Currently, M-AGE uses a mean value of aggregation-column as the sample, which is Sample ¼ SUM=Counter, where SUM is sum of values from aggregation-column, and Counter is the number of records in the current partition. A detailed balanced partition algorithm is shown in Algorithm 3.

```
Algorithm 3: Partitioning.
Input:(R,VP);
   R: an input record;
   VP: the partition vector set.
Output : PID;
    PID: a partition identifier for partition p.
Step 1: Parse the input record R ;
Step 2: Compute the GID;
Step 3:Get the partition vector Vpi from VP with the GID
,and let Vpi = < GID,Vr >;
Step 4: Set the target partition identifier;
Step 5: Build the sample in partition PID;
Step 6: return PI
```

To ensure that data is balanced on each server, the partition algorithm divides each group into a number of partitions and sends to one server depending on the data distributions.The input record R is sent to a partition given by PID which is generated from its corresponding aggregation-column.M - AGE 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 M - AGE. In this section, we analyze the estimated relative error and the confidence interval of final range-aggregate query result.

Theorem:
$\Delta\widehat{S}$ *is an unbiased estimation of* ΔS *in big data environments.*

Proof: According to Algorithm 3,the range aggregate query result $\hat{S}$ in each partition is expressed as follows :

$$\hat{S} = Count * Sample, \qquad (1)$$

Where *Count* is the estimated range cardinality obtained from the histogram, *Sample* is a sample of values of aggregation-column in the queried partition. The exact range-aggregate result S is expressed as $S = \sum_{j=1}^{n} Xj$, where X is a selected tuple in the queried partition. According to Eq.(1),the expectation of $\Delta\widehat{S}$ can be expressed as:

$$E(\Delta\widehat{S}) = E(|(S - \hat{S})/S|) = E(|1-\frac{Count}{n}|) \qquad (2)$$

Now the error transformation formula is used to analyse the variance of .$\Delta\widehat{S}$ and it is expressed as follows:

$$\sigma(\Delta\widehat{S}) = \sqrt{\sigma 2(\Delta Sample + \sigma 2(\Delta Count)}, \quad (3)$$

The variance of estimated cardinality has been discussed in [15], and the $\sigma(\Delta Count)$ asymptotically equals to $\frac{1.04}{\sqrt{m}}$,where m is the number of register bit array. If m is set to m=2^{12},$\sigma(\Delta\widehat{S})$=0.026.

VII. RESULTS

The results that are shown are with FastRAQ in comparison with Hive. FastRAQ is better than Hive in terms of performance and it reduces the two types of cost significantly.

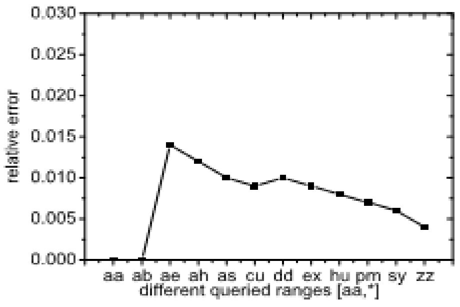

Fig 4.The different queried ranges.

Here the results are shown with M-AGE and its performance with MongoDB.M-AGE acts as a tool to boost the performance in DBaaS. It can be shown that M-AGE is significantly better than FastRAQ.

CONCLUSION

The key idea was to reduce the two types of cost which were noted earlier in the existing systems. M-AGE is a new framework deployed to ensure the same. It is mainly to process and provide accurate results to answering range aggregate queries in big data atmospheres. M-AGE solves the 1:n format range aggregate queries problem, i.e., there is one aggregation column and n index columns in a record.

REFERENCES

[1]. P. Mika and G. Tummarello, "Web semantics in the clouds," IEEE Intell. Syst., vol. 23, no. 5, pp. 82–87, Sep./Oct. 2008.

[2]. T. Preis, H. S. Moat, and H. E. Stanley, "Quantifying trading behavior in financial markets using Google trends," Sci. Rep., vol. 3, p. 1684, 2013.

[3]. H. Choi and H. Varian, "Predicting the present with Google trends," Econ. Rec., vol. 88, no. s1, pp. 2–9, 2012.

[4]. C.-T. Ho, R. Agrawal, N. Megiddo, and R. Srikant,, "Range queries in OLAP data cubes," ACM SIGMOD Rec., vol. 26, no. 2, pp. 73–88, 1997.

[5]. G. Mishne, J. Dalton, Z. Li, A. Sharma, and J. Lin, "Fast data in the era of big data: Twitter's real-time related query suggestion architecture," in Proc. ACM SIGMOD Int. Conf. Manage. Data,

[6]. 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.

[7]. J. M. Hellerstein, P. J. Haas, and H. J. Wang, "Online aggregation," ACM SIGMOD Rec., vol. 26, no. 2, 1997, pp. 171–182.

[8]. P. J. Haas and J. M. Hellerstein, "Ripple joins for online aggregation," in ACMSIGMOD Rec., vol. 28, no. 2, pp. 287–298, 1999.

[9]. E. Zeitler and T. Risch, "Massive scale-out of expensive continuous queries," Proc. VLDB Endowment, vol. 4, no. 11, pp. 1181–1188, 2011.

[10]. N. Pansare, V. Borkar, C. Jermaine, and T. Condie, "Online aggregation for large MapReduce jobs," Proc. VLDB Endowment, vol. 4, no. 11, pp. 1135–1145, 2011.

[11]. T. Condie, N. Conway, P. Alvaro, J. M. Hellerstein, J. Gerth, J. Talbot, K. Elmeleegy, and R. Sears, "Online aggregation and continuous query support in MapReduce," in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2010, pp. 1115–1118.

[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]. S. Heule, M. Nunkesser, and A. Hall, "Hyperloglog in practice: algorithmic engineering of a state of the art cardinality estimation algorithm," in Proc. 16th Int. Conf. Extending Database Technol., 2013, pp. 683–692.