Volume 4 Issue II, February 2016
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

Dynamic approach of Range Aggregate queries in Big Data environment

L. Pravin Kumar[1], Ramesh Krishnan[2]
[1]*Internship, BMW India Pvt. Ltd., Chennai, India.*
[2]*Research Scholar, Department of Electronics and Communication Engineering, Easwari Engineering College, Anna University, Chennai, India-600 089.*

Abstract: ***Range-aggregate queries are to apply a certain aggregate function on all tuples within given query ranges. Existing approaches to range-aggregate queries are insufficient to quickly provide accurate results in big data environments. In this paper, we propose FastRAQ a fast approach to range-aggregate queries in big data environments. FastRAQ approach on the Linux platform, and evaluate its performance with about 10 billions data records. Experimental results demonstrate that FastRAQ can solve the 1: n format Range aggregate queries problem, i.e., there is one aggregation column and n index columns in a record. We plan to investigate how our solution can be extended to the case of m : n format problem, i.e., there are m aggregation columns and n index columns in a same record. In this range aggregation, insertions of dynamic environments. With the geometric aggregation queries expressions.***
Keywords : big data, range-aggregate query, multidimensional histogram, Balanced partition

I. INTRODUCTION

Big data is defined as the large amount of data which requires new technologies and architectures to make possible to extract value from it by capturing and analysis process. New data sources of big data include location specific data arising from traffic management, and from the track of personal devices such as smartphones. While the word "Big" in Big Data alludes to massive volumes of data, users must understand this as a relative term. Some industries and organizations are likely to have mere gigabytes or terabytes of data as opposed to the petabytes or exabytes of data for some of the social networking organizations. Nevertheless, these seemingly smaller applications may still require the intense and complex information processing and analysis that characterize Big Data applications.

The financial services industry demonstrates this variability. When engaging in certain Big Data activities, there may be millions or billions of records to consider, but each record may only be several bytes long (such as stock ticker information). Conversely, email archives may accumulate several petabytes of data containing valuable customer suggestions or complaints, records of projects, legal records, contracts, and proposals.

The email archive usually contains the best record of pending and current business, but it needs to be sorted and mined to find out what it contains. Another good example is product design and manufacturing, where automotive and aerospace companies, for example, may evaluate hundreds or thousands of virtual prototypes to home in on the best vehicle design. The new large-scale scientific experiments that generate petabytes of mixed data every day as input into a complex simulation model are another example.

Variety in Big Data is a critical attribute. The combination of data from a variety of data sources and in a variety of formats is a key criterion in determining whether an application can be considered Big Data. Big Data applications typically combine data from a variety of data sources (typically both internal and external to an organization) and data of different types (structured, semistructured, and unstructured). This is an important facet of Big Data for both technical and potential impact reasons.

Carl W.Olofson and Dan Vesset et al Combining types of information is a complex technical challenge: What is the relative importance of a tweet versus a customer record? How do you combine a large number of changing patient records with published medical research and genomic data to find the best treatment for a particular patient? An example of this may be the mash up of internal operational data from the ERP system with semi-structured data from Web log files that identifies customers' online behavior, with sentiment analysis of unstructured text from customer comments.

Another example is advanced weather/climate modeling that draws on 100 years of weather data with new physical models of ocean water behaviors and CO level changes, mixing in satellite data feeds to create a real-time simulation.

Volume 4 Issue II, February 2016
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

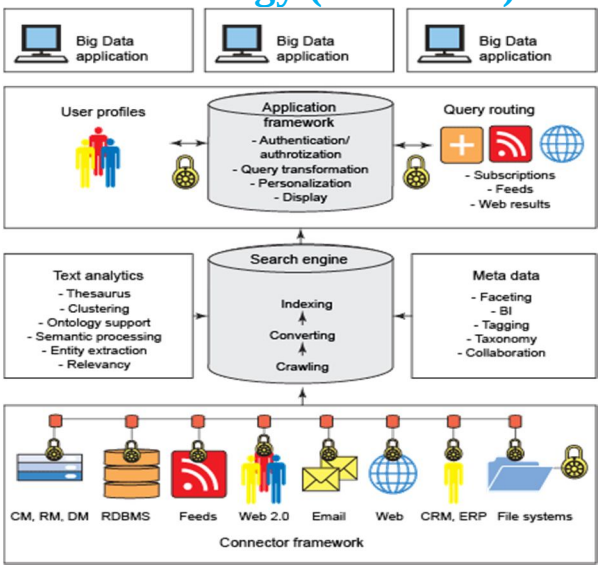

Fig.1 Big data application

A. Big data drivers

1) Ability of compute/ analyze
2) Availability of data
3) Need to deliver/ extract value

The MIT Center for Digital Business states "When it comes to big data, the 'right' governance model depends on the maturity level of the organization regarding data driven decisions." It obviously also highly depends on if big data is used to create new business or to drive more sales. To unleash the power of big data, first of all data must be available and made fit for sharing. When it comes to, for example, medical data, respect for privacy and trust is inevitable.

Standardization in governance structures, with an integrated combination of technical, organizational and legal measures and safeguards, will help to increase trust. This is especially important when integrating governmental, institutional "open" and company data. An example of this in action is seen with a European association that aims to build up a big data services platform for the health sector in their local region.

It's a unique collaboration between health institutions, government, education and knowledge institutions and major IT service providers, addressing both the clinical and research sides of the health sector.

The core solution is comprised of a "vendor neutral hub" a platform that works independently of vendors and data "owners" where data can be captured, safely and durably stored, processed and distributed and finally, shared, if permitted by the data owner. This may be the case when patients are being treated by multidisciplinary teams or having treatment in various locations, or for research purposes using large sets of anonymous data. The framework offers solutions for the immediate need to access data, to substantially lower the cost of data storage and, more importantly, to do more with the rapidly growing amount of unstructured data.

With the rapid adoption of smartphone and wearable technology, organizations are increasingly taking advantage of the new medium (e.g., developing their own apps, partnering with a third-party app, or purchasing advertising space within a third party app) in which to reach out to customers to increase their revenue. For example, health and fitness organizations have developed mobile apps that utilize the GPS function of a smartphone and motion/accelerometer to determine where you are, plot your route on a map, show your running speed, etc. Some organizations have also developed software coupled with wearable technology that monitors your sleeping patterns.

II. EXISTING SYSTEM

In a range-aggregate query problem preprocess a set S of geometric objects such that given a query orthogonal range q, a certain intersection or proximity query on the objects of S intersected by q can be answered efficiently. Although range-aggregate queries have been widely investigated in the past for aggregation functions like average, count, min, max, sum etc. there is little work on proximity problems.

Range searching is a fairly well-studied problem in Computational Geometry. In such a problem, we are given a set S of n

Volume 4 Issue II, February 2016
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

geometric objects. The goal is to efficiently report or count the intersections of the given set of objects with a given query range q. If we are required to perform queries on the geometric data set several times, it is worthwhile to arrange the information into a data structure to facilitate searching.

In a class of problems called range-aggregate query problems we deal with some composite queries involving range searching, where we need to do more than just a simple range reporting or counting. In a generic instance of a range-aggregate query problem, we wish to preprocess a set of geometric objects S, such that given a query range q, a certain aggregation function that operates on the objects of S0 = S can be computed efficiently.

In on-line analytical processing (OLAP), geographic information systems (GIS) and other applications range aggregate queries play an important role in summarizing information and hence large number of algorithms and storage schemes have been proposed. However most of these work consider functions like count, sum, min, max, average etc. range-aggregate query problems involving geometric aggregation operations like intersection, point enclosure and proximity were studied.

Geometric proximity problems arise in numerous applications and have been widely studied in computational geometry. The closest-pair problem involves finding a pair of points in a set such that the distance between them is minimal. Work on the closest-pair and some related problems are surveyed. The range-aggregate version was considered for the variant where the query range is an axes-parallel rectangle and R-tree based solution was given. In the 1-dimensional (resp. 2-dimensional) range-aggregate closest pair problems with query orthogonal rectangles were solved in O(n) space (resp. $O(n^2 \log^3 n)$ space) and O(1) (resp. $O(\log^3 n)$) query time.

The range-aggregate closest-pair problem considered in can arise in applications like GIS. For instance, consider an example borrowed from. We may be interested in finding the closest pair of post offices in a city. If the City Hall wants to move some post office to a new location, this query may help in decision making.

As another application consider VLSI design rule checking. VLSI design rules are often based on the so-called lamda (_) based design rules made popular by Mead and Conway. Design rule checking (DRC) is the process of checking if the layout satisfies the given set of rules. One problem of interest to the designer is to check whether certain features are apart at least by a required separation. To check for violations in a part of the circuit, we can check if any two points in a query range violate the minimum separation rule. This can be reduced to an instance of the range-aggregate closest-pair query.

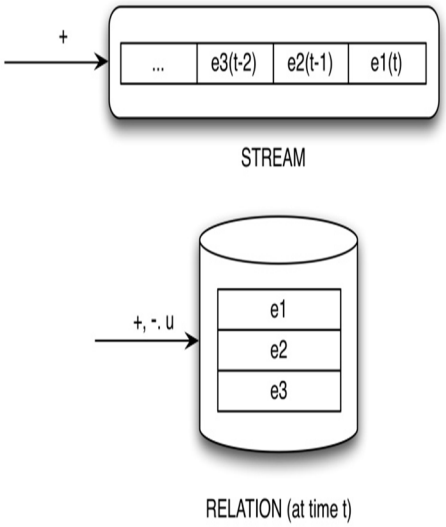

Fig.2 Series of insertion

FastRAQ a new approximate answering approach that acquires accurate estimations quickly for range-aggregate queries in big data environments. FastRAQ has data updates and ad-hoc range-aggregate queries. If the ratio of edge-bucket cardinality (h0) is small enough, FastRAQ even has time complexity for range aggregate queries. We believe that FastRAQ provides a good starting point for developing real-time answering methods for big data analysis. There are also some interesting directions.

FastRAQ can solve the 1: n format range aggregate queries problem, i.e., there is one aggregation column and n index columns in a record. We plan to investigate how our solution can be extended to the case of m : n format problem, i.e., there are m aggregation columns and n index columns in a same record. Second, FastRAQ is now running in homogeneous environments[2]. Here,

Volume 4 Issue II, February 2016
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

FastRAQ can be applied in heterogeneous context or even as a tool to boost the performance of data analysis in DBaas.

III. PROPOSED SYSTEM

A. *Semi-dynamic Range-Aggregate Point Enclosure*

Build data structures which can handle insertions efficiently for solving the Range-Aggregate Point Enclosure problem. As done previously, the one-dimensional scenario is presented first and then extended to higher dimensions. One-dimensional scenario: The preprocessing steps are as follows.

Using the segments in T, a standard segment tree ST is built. Structure ST is equipped to handle both the reporting and the counting queries (query here will be a point). Construction of the segment tree takes O(n log n) time and space. Point set S is divided into three disjoint subsets S1, S2 and S3. If a point p $\in$ S intersects with none of the segments in T, then p falls into set S1. If the number of segments of T with which p intersects lies in the range (0; log n), then it falls into set S2,S3 $\subseteq$ S contains the points which intersect with more than "log n - 1" segments of T, i.e., in the range [log n, n]. Segment tree ST is used for finding the number of segments of T being intersected by each point in S.

This partition of point set S into three subsets takes O(n log n) time. Based on the x-coordinates of the points in Si ($\forall$ i =1,2,3), we build a balanced binary search tree BTi. We pick BT2 for further augmentation while BT1 and BT3 are not augmented further.

The points in S2 are placed at the leaf nodes of BT2. With each point p $\in$ S2 present at a leaf node of T2, we store a list, Lp, of all the segments of T it intersects. Also, the size of each list Lp is maintained.

Saladi Rahul, Ananda Swarup Das et al Handling insertions. Suppose a new point p is added to the set S. It is first queried on ST to out the number of segments of T it intersects with (say kp). Based on the value of kp it is kept in one of the subsets S1, S2 or S3 and then inserted appropriately into one of the binary trees BT1, BT2 or BT3. f p is inserted into BT2, then the list Lp of the segments of T it intersects is also prepared. Thus insertion of a point p can be handled in O(log n) time.

Underlying space	Objects in S	Objects in T	Query	Space	Query time	Insert time (amortized)
$\mathbb{R}^1$	points	segments	point enclosure	$O(n \log n)$	$O(\log n + k)$	$O(\log n)$
$\mathbb{R}^d$	points	hyper rectangles	point enclosure	$O(n \log^d n)$	$O(\log^{d-1} n \log\log n + k)$	$O(\log^d n)$
$\mathbb{R}^1$	segments	segments	intersections	$O(n \log n)$	$O(\log n + k)$	$O(\log n)$

Table 1. Summary of results for *semi-dynamic* range-aggregate query problems; the query q is an orthogonal range; k is the output size. Insertion time mentioned in *amortized* time.

Now, suppose a new segment t is to be added to the set T. t is first inserted into the segment tree ST. This takes O(log n) amortized time. Next, BT2 is queried with t. For each point p 2 BT2 which intersects t, t is added to the list p. If [Lp] = log n, then p is shifted from set S2 to S3. p and its list Lp is deleted from BT2, and p is inserted into BT3. Let be the number of points in BT2 which are intersected by p and the no. of points shifting from BT2 to BT3. Then the time taken to update the lists in BT2 and the shifting process from BT2 to BT3 takes O(log n + lamda1 + lamda2 log n). Then BT1 is queried with t. For each point BT1 which intersects t, p is deleted from BT1 and inserted into BT2. In BT2, the list Lp is initialized for point p with t being the only entry in it. Let lamda3 be the number of points shifting from BT1 to BT2. This will take O(log n + lamda3log n) time. Therefore, the total time for inserting a new segment t is O(log n + lamda1 + lamda2 log n + lamda3log n)[1].

An amortized analysis is carried out to get an efficient bound. Assume that we insert n segments and points in an arbitrary order. Notice that a point in set S1 can jump only once into set S2 and a point in set S2 can jump only once into set S3. Therefore, the value of lamda2 and lamda3 where the summation is over n insertions are bounded by O(n). A point in S can remain in the set S2 till it intersects with less than log n segments. Also, the number of points in set S after n insertions is still bounded by O(n). Therefore,

Volume 4 Issue II, February 2016
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

the value of lamda1 where the summation is over n insertions is O(n log n).

Therefore, the total time taken for insertion of n segments and points is: O (log n + lamda1 + lamda2 log n + lamda3log n)) O(n log n). The amortized time turns out to be O(log n).

After n insertions of points and segments, the whole data structure is deleted and reconstructed. After n insertions, the total number of points and segments become 2n. Therefore, we shall update the criteria for a point p to enter set S2 and S3 from [0,log n) to (0,log 2n) and [log n, n] to [log 2n, 2n], respectively. Since, the preprocessing time is O(n log n) when built on n points and segments, the amortized time of insertion does not change.

IV. CONCLUSION

The solution to the semi-dynamic version of the problem is similar to that of the static solution. We shall preprocess the segments of S and the endpoints of the segments of S. An endpoint of a segment t is not considered to be intersecting with the segment it comes from (which in this case is t). This makes sense since in conditions (i) and (ii) of Lemma 1 we want an endpoint of a segment t to intersect with a segment other than t. For handling condition (iii), instead of using a static priority search tree (used in the static solution), we shall use a Dynamic priority search tree D [5]. A Dynamic priority search tree when built on m points takes up O(m) space, answers queries in O(log m + k) time and updates take O(log m) time. In our case m $\equiv$ O(n). In future higher dimensions remain important open problems.

REFERENCES

[1] P. Afshani. On Dominance Reporting in 3D Proceedings of 16th European Symposium on Algorithms (ESA), 2008, 41-51.
[2] P. Gupta. Algorithms for range-aggregate query problems involving geomteric aggregation operations. Proceedings, International Symposium on Algorithms and Computation, Springer Verlag LNCS, Vol. 3827, 2005, 892–901.
[3] Pankaj K. Agarwal, Lars Arge, Jun Yang, Ke Yi. I/O-Ecient Structures for Orthogonal Range-Max and Stabbing-Max Queries. 11th European Symposium on Algorithms, 7{18, 2003.
[4] M. de Berg, M. van Kreveld, M. Overmars, O. Schwarzkopf. Computational Geometry, Springer Verlag, 2nd ed., 2000
[5] Y. Chiang and R. Tamassia, \Dynamic Algorithms in Computational Geometry", Proceedings of the IEEE, Special Issue on Computational Geometry, G. Toussaint(Ed.), vol. 80(9), pp. 1412-1434, 1992.

AUTHORS

Pravin Kumar.L received B.E degree in computer science and engineering from Anna University,Tamil Nadu, India in 2015 respectively. Currently he is doing the full time internship in BMW India Pvt. Ltd. He has published one international journals.

His research interests focus on the area of BigData analytics, dynamic adaptation of MapReduce computations, data mining and network security.

Ramesh Krishnan received Bachelor of Engineering and Master of Engineering degrees from Anna University, India in 2007 and 2010 respectively. Currently he is pursuing the full time Ph.D Degree at Anna University. He has published papers in four international journals and three international conferences

His research interests focus on the area of low-power, high-performance VLSI architectures and circuit design for digital communications and digital signal processing.