A RAPID METHOD OF DERIVING GROUP AND SUBGROUP OF QUERIES IN BIG DATA ENVIRONMENTS

Soniya Shantilal Dadhania
Lecturer
Computer Department
RC Technical Institute, Sola, Ahmedabad

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 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. FastRAQ has O (1) time complexity for data updates and O (N/PxB) time complexity for range-aggregate queries, where N is the number of distinct tuples for all dimensions, P is the partition number, and B is the bucket number in the histogram. We implement the FastRAQ approach on the Linux platform, and evaluate its performance with about 10 billions data records. Experimental results demonstrate that FastRAQ provides range-aggregate query results within a time period two orders of magnitude lower than that of Hive, while the relative error is less than 3 percent within the given confidence interval.

1. INTRODUCTION

Big data is a broad term for data sets so large or complex that traditional data processing applications are inadequate. Challenges include analysis, capture, search, sharing, storage, transfer, visualization, and information privacy. The term often refers simply to the use of predictive analytics or other certain advanced methods to extract value from data, and seldom to a particular size of data set. Accuracy in big data may lead to more confident decision making. And better decisions can mean greater operational efficiency, cost reduction and reduced risk. Analysis of data sets can find new correlations, to "spot business trends, prevent diseases, and combat crime and so on". Scientists, business executives, practitioners of media and advertising and governments alike regularly meet difficulties with large data sets in areas including Internet search, finance and business informatics. Scientists encounter limitations in e-Science work, including meteorology, genomics, connectomics, complex physics simulations, and biological and environmental research. Data sets grow in size in part because they are increasingly being gathered by cheap and numerous information-sensing mobile devices, aerial (remote sensing), software logs, cameras, microphones, radio-frequency identification (RFID) readers, and wireless sensor networks. The world's technological per-capita capacity to store information has roughly doubled every 40 months since the 1980s; as of 2012, every day 2.5 exabytes (2.5×1018) of data were created; The challenge for large enterprises is determining who should own big data initiatives that straddle the entire organization. Big data usually includes data sets with sizes beyond the ability of commonly used software tools to capture, curate, manage, and process data within a tolerable elapsed time. Big data "size" is a constantly moving target, as of 2012 ranging from a few dozen terabytes to many petabytes of data. Big data is a set of techniques and technologies that require new forms of integration to uncover large hidden values from large datasets that are diverse, complex, and of a massive scale. Analysis of data is a process of inspecting, cleaning, transforming, and modeling data with the goal of discovering useful information, suggesting conclusions, and supporting decision-making. Data analysis has multiple facets and

approaches, encompassing diverse techniques under a variety of names, in different business, science, and social science domains. The partition problem is the task of deciding whether a given multiset S of positive integers can be partitioned into two subsets S1 and S2 such that the sum of the numbers in S1 equals the sum of the numbers in S2. Although the partition problem is NP-complete, there is a pseudo polynomial time dynamic programming solution, and there are heuristics that solve the problem in many instances, either optimally or approximately. For this reason, it has been called "The Easiest Hard Problem". There is an optimization version of the partition problem, which is to partition the multiset S into two subsets S1, S2 such that the difference between the sum of elements in S1 and the sum of elements in S2 is minimized. The optimization version is NP-hard. Histograms are a concise and flexible way to construct summary structures for large data sets. They have attracted a lot of attention in database research due to their utility in many areas, including query optimization, and approximate query answering. They are also a basic tool for data visualization and analysis. A histogram is a display of statistical information that uses rectangles to show the frequency of data items in successive numerical intervals of equal size. In the most common form of histogram, the independent variable is plotted along the horizontal axis and the dependent variable is plotted along the vertical axis. The data appears as colored or shaded rectangles of variable area. Range searching and its variants have been studied extensively in the computational geometry and database communities because of their many important applications. Range-aggregate queries, such as range-COUNT, SUM and MAX, are some of the most commonly used versions of range searching in database applications. Since many such applications involve massive amounts of data stored in external memory, it is important to consider external memory (or I/O-efficient) structures for fundamental range-searching problems. In this paper, we develop an external memory data structure for answering orthogonal range-COUNT, SUM and MAX queries. Note that from these we automatically get some other aggregates like AVE and MIN. Good histograms partition data sets into \smooth" buckets with close-to-uniform internal tuples density. In other words, the frequency variance of the tuples enclosed by such buckets is minimized, leading to accurate selectivity estimations for range queries. Unfortunately, current multidimensional histogram techniques do not always manage to produce close-to-uniform partitions of the data sets, as we discuss next. Later it reports a thorough experimental evaluation of these techniques that complements the discussion in this section. A partition of a multidimensional data domain results in a set of disjoint rectangular buckets that cover all the points in the domain and assigns to each bucket some aggregated information, usually the number of tuples enclosed. The choice of rectangular buckets is justified by two main reasons: First, rectangular buckets make it easy and efficient to intersect each bucket and a given range query to estimate selectivity. Second, rectangular buckets can be represented concisely, which allows a large number of buckets to be stored using the given budget constraints.

2. MODULES DESCRIPTION

1. FASTRAQ System Model
2. Distributed Partitioning Algorithm
3. Clustering based histogram
4. Range cardinality queries
5. Analysis of relative errors

- **FASTRAQ System Model**

In FastRAQ, we divide numerical value space of an aggregation-column into different groups, and maintain an estimation sketch in each group to limit relative estimated errors of range-aggregate paradigm. When a new record is coming, it is first sent onto a partition in the light of current data distributions and the number of available servers. In each partition, the sample and the histogram are updated respectively by the attribute values of the incoming record. When a query request arrives, it is delivered into each partition. We first build cardinality estimator (CE) for the queried range from the histogram in each partition. Then we calculate the estimate value in each partition, which is the product of the sample and the estimated cardinality from the estimator. The final return for the request is the sum of all the local estimates.

- **Distributed Partitioning Algorithm**

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 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. We build our 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. We first divide the value of numerical space into different groups and subdivide each group into different partitions according to the number of available servers.

- **Clustering Based Histogram**

We measure the data distributions by clustering values of all index-columns and use the learned knowledge to build our histogram. A feature vector of clustering is expressed as ftag; vector g, where tag is the attribute value, and vector is the frequency for the tag occurring in each dimension. For example, the feature {tag=ad, vector=<10,2>} indicates that the value of ad occurs in the first index column 10 times and the second index column 2 times. After extracting the feature vectors from learned data set, it will produce vectors set. Let it be f<tagi; vectori >j0 < i < Ng. We use the common K-Means clustering method to analyze the vectors set and produce K clusters. A unique ClusterID 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; ClusterID >. 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 ClusterIDs into the same bucket. If some tag occurring frequency is significantly different from others, its ClassID is different after the K-Means clustering, and it will be put into an independent bucket in the histogram.

- **Range Cardinality Of Queries**

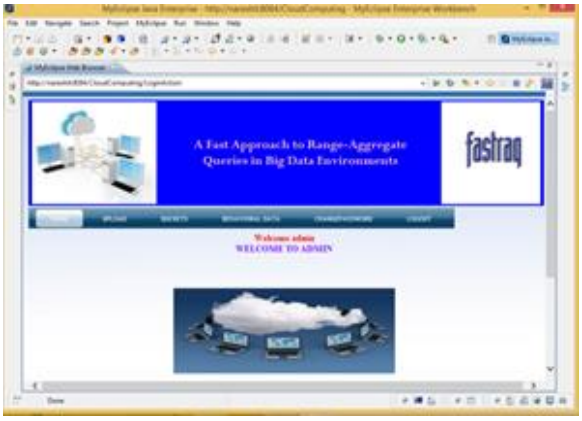
FastRAQ supports multi-dimensional ranges queries, each of which may include multiple buckets of the histogram. FastRAQ uses a unique Record ID (RID, as step 6 in Algorithm 4) to predict whether the cardinalities obtained from different buckets belonging to the same record. We adopt the Hyper Log Log Plus algorithm to estimate the cardinality in the queried range. We serialize the hash bits to bytes array in each bucket as a cardinality estimator. Hyper Log Log-Plus uses 64 bits hash function instead of 32 bits in Hyper- Log Log to improve the data-scale and estimated accuracy in big data environments. Readers can further refer to the references to learn about cardinality estimation mechanism. We establish a hierarchical tree structure to implement the histogram.

- **Analysis Of Relative Errors**

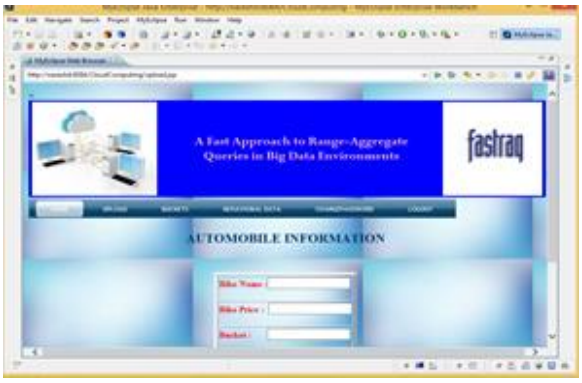
FastRAQ uses approximate answering approaches, such as sampling, histogram, and cardinality estimation etc., to improve the performance of range-aggregate queries. We use relative error as a statistical tool for accuracy analysis. Relative error is widely used in an approximate answering system. Also, it is easy to compute the relative errors of combined estimate variables in a distributed environment for FastRAQ.

3. SCRENSHOTS

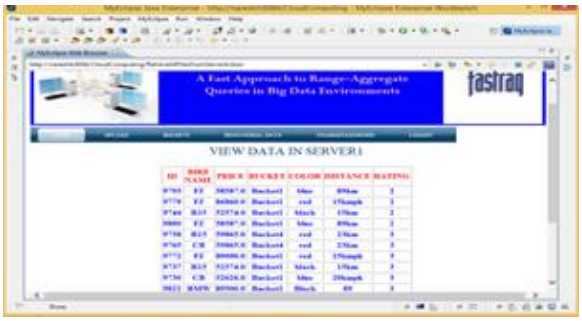
Admin



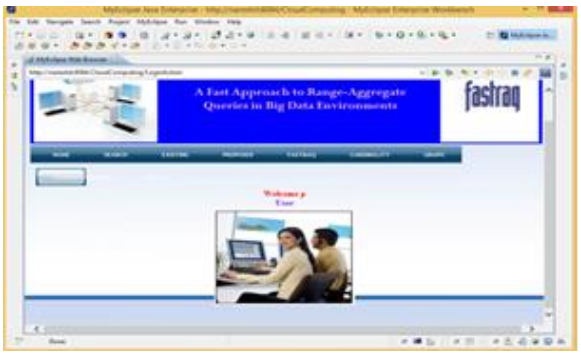
Upload Automobile Information:



View Data:



User Home



Search

4. CONCLUSION

We will further explore how Fast RAQ can be applied in heterogeneous context or even as a tool to boost the performance of data analysis in DBaas.

REFERENCES

[1] P. Mika and G. Tummarello, "Web semantics in the clouds," IEEE Intell. Syst., vol. 23, no. 5, pp.

[2] T. Preis, H. S. Moat, and E. H. 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, 2013, pp. 1147–1158.

[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 aggrega- tion,"inACMSIGMODRec.,vol.28,no.2,pp.287–298,1999.

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

[10] N. Pansare, V. Borkar, C. Jermaine, and T. Condie, "Online aggre- gation 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. Tal- bot, K. Elmeleegy, and R. Sears, "Online aggregation and continu- ous query support in MapReduce," in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2010, pp. 1115–1118.