

Black Money Check: Integration of Big Data & Cloud Computing To Detect Black Money Rotation with Range – Aggregate Queries

¹Elumalai R, ²Mathankumar G, ³Gunaseelan V, ⁴Aravind raj S, ⁵Gnanavel S
^{1,2,3,4}Department of Information Technology*

⁵Assistant Professor S.K.P Engineering College, Tiruvannamalai,

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. Fast RAQ 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, Fast RAQ obtains the result directly by summarizing local estimates from all partitions & Collective Results are provided. We deploy Big data for Banking Domain in this Project. User's banking data is partitioned into multiple Tuples and stored in different sets of Database. We have designed an Application to track multiple accounts maintained in different banks of the same user and their Transaction details. This process helps in finding out Black money Holders so that Government can track them.

Keywords: black money check, range aggregate queries, balancing partitioning algorithm, online aggregation, cloud setup, big data setup

I.INTRODUCTION

BIG data analysis can discover trends of various social aspects and preferences of individual everyday behaviors. This provides a new opportunity to explore fundamental questions about the complex world . For example, to build an efficient investment strategy, Preis et al. analyzed the massive behavioral data sets related to finance and yielded a profit of even 326 percent higher than that of a random investment strategy. Choi and Varian presented estimate sketches to forecast economic indicators, such as social unemployment, automobile sale, and even destinations for personal travelling. Currently, it is important to provide efficient methods and tools for big data analysis. We give an application example of big data analysis. Distributed intrusion detection systems (DIDS) monitor and report anomaly activities or strange patterns on the network level. A DIDS detects anomalies via statistics information of summarizing traffic features from diverse sensors to improve false-alarm rates of detecting coordinated attacks. Such a scenario motivates a typical range-aggregate query problem [4] that summarizes aggregated features from all tuples within given queried ranges. Range-aggregate queries are important tools in decision management, online suggestion, trend estimation, and so on. It is a challenging problem to quickly obtain range-aggregate queries results in big data environments. The big data involves a significant increase in data volumes, and the selected tuples maybe locate in different files or blocks. On the other hand, real time systems aim to provide relevant results within second son massive data analysis .The Prefix-sum Cube (PC) method is first used in OLAP to boost the performance of range-aggregate queries. All the numerical attribute values are sorted and any range aggregate query on a data cube can be answered in constant time. However, when a new tuple is written into the cube, it has to recalculate the prefix sums for

all dimensions. Hence, the update time is even exponential in the number of cube dimensions. Online Aggregation (OLA) is an important approximate answering approach to speeding range-aggregate queries, which has been widely studied in relational databases [8] and Cloud systems

2 OVERVIEW OF THE FASTRAQ APPROACH

2.1 Problem Statement

We consider the range-aggregate problem in big data environments, where data sets are stored in distributed servers. An aggregate function operates on selected ranges, which are contiguous on multiple domains of the attribute values. 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

$$\begin{aligned} &I_{i1} < \text{ColName}_i < I_{i2} \text{opr} \\ &I_{j1} < \text{ColName}_j < I_{j2} \text{opr} \\ &\dots; \end{aligned}$$

In the above query, exp is an aggregate function such as SUM or COUNT; AggColumn is the dimension of the aggregate operation; $I_{i1} < \text{ColName}_i < I_{i2}$ and $I_{j1} < \text{ColName}_j < I_{j2}$ are the dimensions of range queries; opr is a logical operator including AND and OR logical operations. In the following discussion, AggColumn is called Aggregation-Column, ColName_i and ColName_j are called Index-Columns.

The cost of distributed range-aggregate queries primarily includes two parts. i.e., the 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 to search the selected tuples. When the size of a data set increases continuously, the two types of cost will also increase dramatically. Only when the two types of cost are minimized, can we obtain faster final range-aggregate queries results in big data environments.

2.2 Key Idea

To generate a local request result, we design a balanced partition algorithm which works with stratified sampling model. In each partition, we maintain a sample for values of the aggregation-column and a multi-dimensional histogram for values of the index-columns. When a range-aggregate query request arrives, the local result is the product of the sample and an estimated cardinality from the histogram. This reduces the two types of cost simultaneously. It is formulated as ${}^M\text{Count}^i_{\text{Sample}^i}$, where M is the number of partitions, Count is the estimated cardinality of the queried ranges, and Sample^i is the sample for values of aggregation-column in each partition.

Column-family schema for FastRAQ, which includes three types of column-families related to range-aggregate queries. They are aggregation column-family, index column-family, and default column-family. The aggregation column-family includes an aggregation-column, the index column-family includes multiple index-columns, and the default column-family includes other columns for further extensions. A SQL-like DDL and DML can be defined easily from the schema. An example of column-family schema and SQL-like range-aggregate query statement is shown in Fig. 1.

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. A brief FastRAQ framework

PROPOSED SYSTEM

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 & Collective Results are provided. We deploy Big data for Banking Domain in this Project. User's banking data is partitioned into multiple Tuples and stored in different sets of Database. We have designed an Application to track multiple accounts maintained in different banks of the same user and their Transaction details. This process helps in finding out Black money Holders so that Government can track them

MODULES:

User Profile, Account Registration

Here first the User wants to create an account and then only they are allowed to access the Network. Once the User creates an account, they are to login into their account and request the Job from the Service Provider. Based on the User's request, the Service Provider will process the User requested Job and respond to them. All the User details will be stored in the Database of the Service Provider. In this Project, we will design the User Interface Frame to Communicate with the Server through Network Coding using the programming Languages like Java. By sending the request to Server Provider, the User can access the requested data if they authenticated by the Service Provider.

RBI server

Bank Service Provider will contain information about the user in their Data Storage. Also the Bank Service provider will maintain the all the User information to authenticate when they want to login into their account. The User information will be stored in the Database of the Bank Service Provider. To communicate with the Client and the with the other modules of the Company server, the Bank Server will establish connection between them. For this Purpose we are going to create a User Interface Frame.

Cloud Setup

Cloud Service Provider will contain the large amount of data in their Data Storage. Also the Cloud Service provider will maintain the all the User information to authenticate the User when are login into their account. The User information will be stored in the Database of the Cloud Service Provider. Also the Cloud Server will redirect the User requested job to the Resource Assigning Module to process the User requested Job. The Request of all the Users will process by the Resource Assigning Module. To communicate with the Client and with the other modules of the Cloud Network, the Cloud Server will establish connection between them. For this Purpose we are going to create an User Interface Frame. Also the Cloud Service Provider will send the User Job request to the Resource Assign Module in First in First out (FIFO) manner.

Big Data Setup

Big data is an all-encompassing term for any collection of data sets so large and complex that it becomes difficult to process using traditional data processing applications. The challenges include analysis, capture, duration, search, sharing, storage, transfer, visualization, and privacy violations. The trend to larger data sets is due to the additional information derivable from analysis of a single large set of related data, as compared to separate smaller sets with the same total amount of data, allowing correlations to be found to "spot business trends, prevent diseases, combat crime and so on. So we can implement big data in our project because every employ has an instructed information so we can make analysis on this data.

Data Split up/portioning

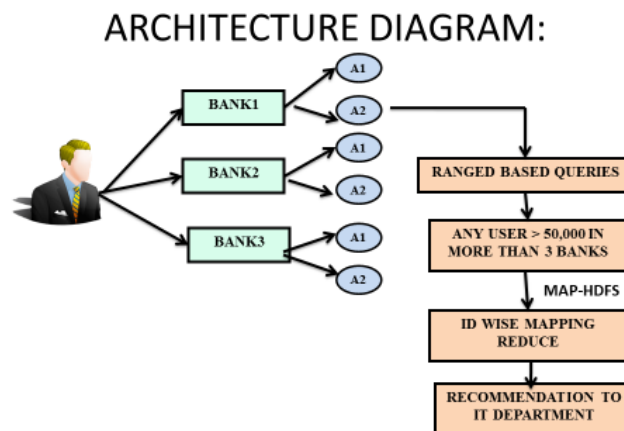
In this module we implement a concept that user in a company or organization will maintain the employee information both private and public data .so employee may contain private like employee id,employee name ,salary and the loan applied and loan go and public data like email id and phone number. But more private information like bank account number and pin number are not reveled form the company. so we split the both type information into a table.

Map reduce-Black Money Detection

In this module, output of mapping part is the input of Reduce part. Once we arrive with users with accounts more than in 3 Banks then those people is the input of this Module. In this module we will track the users with deposited more than Rs. 50000 in Annum in all three banks. Those users are identified as major threats and those people are tracked further by the Income tax department. This is output of this module.

Unique id are tracked

We will extracting new data from the merged data the implementation is all about company employee who has got loan. Employee ID plays as primary key and we can identify the list of loan obtainers. Data is analyzed only by the authorized persons



CONCLUSION

In this paper, we propose FastRAQ—a new approximate answering approach that acquires accurate estimations quickly for range-aggregate queries in big data environments. FastRAQ has $O(1/P)$ time complexity for data updates and $O(N/PB)$ time complexity for ad-hoc range-aggregate queries. If the ratio of edge-bucket cardinality (h_0) is small enough, FastRAQ even has $O(1/P)$ 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 for our future work. First, 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. We will further explore how FastRAQ can be applied in heterogeneous context or even as a tool to boost the performance of data analysis in DBaaS.





REFERENCES

- [1] 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.
- [2] H. Choi and H. Varian, “Predicting the present with Google trends,” *Econ. Rec.*, vol. 88, no. s1, pp. 2–9, 2012.
- [3] 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.
- [4] 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.
- [5] W. Liang, H. Wang, and M. E. Orłowska, “Range queries in dynamic OLAP data cubes,” *Data Knowl. Eng.*, vol. 34, no. 1, pp. 21–38, Jul. 2000.
- [6] J. M. Hellerstein, P. J. Haas, and H. J. Wang, “Online aggregation,” *ACM SIGMOD Rec.*, vol. 26, no. 2, 1997, pp. 171–182.
- [7] P. J. Haas and J. M. Hellerstein, “Ripple joins for online aggregation,” in *ACM SIGMOD Rec.*, vol. 28, no. 2, pp. 287–298, 1999.

AUTHORS



Mr. S. GNANAVEL (ASSISTANT PROFESSOR) INFORMATION TECHNOLOGY, S.K.P ENGINEERING COLLEGE-TIRUVANNAMALAI.

	R.ELUMALAI, INFORMATION TECHNOLOGY S.K.P ENGINEERING COLLEGE-TIRUVANNAMALAI
	V.GUNASEELAN, INFORMATION TECHNOLOGY S.K.P ENGINEERING COLLEGE-TIRUVANNAMALAI
	G.MATHANKUMAR, INFORMATION TECHNOLOGY S.K.P ENGINEERING COLLEGE-TIRUVANNAMALAI
	S.ARAVIND RAJ, INFORMATION TECHNOLOGY S.K.P ENGINEERING COLLEGE-TIRUVANNAMALAI