International Journal of Electrical and Computer Engineering (IJECE)
Vol. 10, No. 1, February 2020, pp. 415~420
ISSN: 2088-8708, DOI: 10.11591/ijece.v10i1.pp415-420

A time efficient and accurate retrieval of range aggregate queries using fuzzy clustering means (FCM) approach

A. Murugan, D. Gobinath, S. Ganesh Kumar, B. Muruganantham, Sarubala Velusamy
Department of Computer Science and Engineering, Faculty of Engineering and Technology, SRMIST, Chennai, India

Article Info

Article history:

Received Feb 17, 2019
Revised Aug 10, 2019
Accepted Aug 29, 2019

Keywords:

Big data
FCM algorithm
Post stratification sampling

ABSTRACT

Massive growth in the big data makes difficult to analyse and retrieve the useful information from the set of available data's. Existing approaches cannot guarantee an efficient retrieval of data from the database. In the existing work stratified sampling is used to partition the tables in terms of stratic variables. However k means clustering algorithm cannot guarantees an efficient retrieval where the choosing centroid in the large volume of data would be difficult. And less knowledge about the stratic variable might leads to the less efficient partitioning of tables. This problem is overcome in the proposed methodology by introducing the FCM clustering instead of k means clustering which can cluster the large volume of data which are similar in nature. Stratification problem is overcome by introducing the post stratification approach which will leads to efficient selection of stratic variable. This methodology leads to an efficient retrieval process in terms of user query within less time and more accuracy.

Corresponding Author:

A. Murugan,
Department of Computer Science and Engineering,
Faculty of Engineering and Technology,
SRMIST, Chennai, India.
Email: amurugan15@outlook.com

1. INTRODUCTION

Big data analysis can reveal that the cost to process the data in big data environment is too high, such as peer influence among customers, disclosed by analysing shopper's transactions, social and geographical data. It is a challenging task in big data environment to quickly acquire the result for range-aggregate queries. A range query is a common database operation that retrieves all records from database where some value exists between an upper and lower boundary. For example, list all employees with 3 to 5 years of experience. Range queries are unusual because we don't know in advance that how many rows will be returned as result. It will return any number of rows or not even any. Many other queries, such as the top ten most senior employees, or the newest employee, can be done more efficiently because these queries have an upper bound to the number of results they will return. Range query retrieval in the smaller database would be more efficient process where it will consume more time and will reduce the time complexity in case of larger database where the big volume of records are present. This problem is addresses in the existing methodology in terms of retrieving more accurate and time concerned output for the range queries from the larger databases.

In big data environment mostly the data sets are stored in distributed environment as the volume of the data is very large. The cost of distributed range-aggregate queries mainly depends on two factors. i.e., network communication cost and local files scanning cost. The first cost depends on data transmission and synchronization for aggregate operations if the files selected are in different servers. The second cost depends on local files scanning to search the selected tuples. When the data set size gets increased continuously, the above mentioned two costs will also get increase dramatically. So only when the above two

costs are minimized, we can get the final range-aggregate queries result faster in big data environments. There are various approaches to minimize the above two costs that has been studied in the past [1-4], but today analysts are progressively more focusing on efficient methods and tools for big data analysis. The sampling and histogram approaches were already used in bid data environments to support approximate answering or selectivity estimation. However, it could not obtain acceptable approximations for the underlying data sets.

2. EXISTING SYSTEM

FastRAQ method is proposed in big data environments as referred in [1]. It first divides large data into small independent partitions using a balanced partitioning algorithm, and then it generates a local estimation sketch for each and every partition. When a range-aggregate query request arrives, FastRAQ obtains the result from each partition and it gives the final result by summarizing the local estimates from each partition. FastRAQ approach is implemented on the Linux platform and its performance is evaluated using nearly 10 billion data records. The various algorithms used in [1] are stratification algorithm and k-means algorithm. The disadvantages in [1] are inefficient retrieval of results, stratification problem occurs and more time complexity.

The fast algorithms in [3, 5] are introduced to speed up the range sum and range max queries in OLAP system. The most important aim of this scenario is used to pre-compute the multi-dimensional prefix sums of the data cube. The total storage need is kept as same as the data cube along with a small increase in time for the queries of a singleton cell. Since any cell of the data cube is calculated with the same time complexity as range sum query. The aggregation operations such as max, sum are used to answer unexpected run time queries. It is efficiently used for maximum range of queries using hierarchical tree structure. The disadvantage is in few cases it has issue with lower accuracy rates.

The Prefix-sum Cube (PC) method [4, 6, 7] is first used in OLAP to improve the range-aggregate query performance. All range-aggregate queries are processed in constant time and all the numerical attribute values are sorted in order, but when a new row or tuple is added in the cube, there is a need to recalculate the prefix sums for all dimensions.Hence, the update time is even worse.

Online Aggregation (OLA) is an important approach to speed up range-aggregate queries and is widely used in relational databases [8, 9] and Cloud systems [10-12]. In OLA systems the background computing processes run for a long time. The returns are refined and the accuracy is also getting better in subsequent stages. But in early stages, the users cannot get an appropriate result with satisfied accuracy. Also it have expensive scenario, wastage of computed resources, reduced performance [13-15].

The HyperLogLog algorithm referred in [16] proves to be easy to code and efficient, being even nearly optimal under certain criteria. It is highly practical, versatile, and it conforms well to what analysis predicts. The algorithm used in [16] is “HyperLogLog with near optimal cardinality algorithm”, but it has the disadvantages of lower accuracy and slow execution time.

A new algorithm representing a series of improvements by reducing the memory requirements and significantly increase the accuracy for an important range of cardinalities is used in [17]. In single pass, it computes the large cardinalities and improves the memory usage more efficiently. The algorithm used in [17] is “Hyperloglog algorithm”. The disadvantage is it still has an issue with error values in few cases.

3. OVERVIEW OF THE FCM APPROACH

3.1. Problem statement

The existing systems still has issue with inefficient retrieval of range aggregate queries. The algorithm provides clustering inaccuracy for larger dataset. The existing system leads time complexity. Thus the overall system performance is degraded.

3.2. Key idea

The FCM approach works in the following manner. First the big data is divided into small independent partitions using a balanced partitioning algorithm as referred in [1]. After that it creates a local estimation sketch for each independent partition. When a range aggregate query request comes, FCM approach gets the result by summarizing local estimates that obtained from all partitions. The balanced partitioning algorithm works along with a poststratified sampling model. It partitions all the data in database into different groups according to their attribute values and it further separates each group into multiple partitions with regard to the current data distributions and the number of servers available.

The estimation sketch is a new type of multi-dimensional histogram which is built with regard to learned data distributions. This multi-dimensional histogram can measure the quality of rows or data set distributions more accurately. It maintains almost equivalent frequencies for different values within each bucket, even though the frequency distributions vary significantly. This concept leads to reduced time complexity by splitting the overall flow of work in multiple partitions. The partitioning is done with the use of attribute values that connects multiple databases. This mechanism leads to faster query response result for every partition that could be combined later. When a range aggregate query request arrives, it is delivered in each partition. First the cardinality estimator for the queried range is built from the histogram in each partition. After that we estimate the attribute value in each partition, which is the product of the sample and the cardinality value that is estimated using the estimator. The final output result for the query request is the sum of all the local estimates from all partitions.

3.2.1. Post stratification sampling

Partitioning is a process of dividing the large table into many smaller tables based on the value of a particular field in a table record. Data partitioning is an important step in data analysis because it improves the query performance. It reduces the size of the data to be scanned for the query result and hence the performance gets increased.

In the proposed system post stratification is introduced to overcome this limitation which will select the stratic variable for the efficient data partitioning. Stratification is sometimes introduced after the sampling phase in a process called "poststratification". This approach is implemented when the experimenter do not have the enough or necessary information to create a stratifying variable during the sampling phase. Although the method is susceptible to the pitfalls of post hoc approaches, it can provide several benefits in the right situation. Implementation usually follows a simple random sample. Poststratification can also be used to implement weighting, which can improve the precision of a sample's estimates.

Poststratification is a method for adjusting the sampling weights, usually to account for under represented groups in the population. Poststratification involves adjusting the sampling weights so that they sum to the population sizes within each poststratum. This usually results in decreasing bias because of nonresponse and underrepresented groups in the population. Poststratification also tends to result in smaller variance estimates. The svyset command has options to set variables for applying poststratification adjustments to the sampling weights. The poststrata() option takes a variable that contains poststratum identifiers, and the postweight() option takes a variable that contains the poststratum population sizes. If w_j is the unadjusted sampling weight for the jth sampled individual, the poststratification adjusted sampling weight is

$$w_j^* = \sum_{k=1}^{L_P} I_{Pk}(j) \frac{M_k}{\widehat{M}_k} w_j$$

where

$$\widehat{M}_k = \sum_{j=1}^{m} I_{Pk}(j) w_j$$

The point estimates are computed using these adjusted weights. For replication-based variance estimation, the BRR and jackknife replicate-weight variables are similarly adjusted to produce the replicate values used in the respective variance formulas: The score variable for the linearized variance estimator of a poststratified total is

$$z_j\left(\widehat{Y}^P\right) = \sum_{k=1}^{L_P} I_{Pk}(j) \frac{M_k}{\widehat{M}_k}\left(y_j - \frac{\widehat{Y}_k}{\widehat{M}_k}\right)$$

where $\widehat{Y}_k$ is the total estimator for the k^{th} poststratum,

$$\widehat{Y}_k = \sum_{j=1}^{m} I_{Pk}(j) w_j y_j$$

For the poststratified ratio estimator, the score variable is

$$z_j(\hat{R}^P) = \frac{\hat{X}^P z_j(\hat{Y}^P) - \hat{Y}^P z_j(\hat{X}^P)}{(\hat{X}^P)^2}$$

where $\hat{X}^P P$ is the poststratified total estimator for item x_j.

3.2.2. Fuzzy clustering means (FCM) approach

In our proposed system, FCM clustering can be introduced to group the similar index values by calculating the membership degree values of them. With the help of FCM clustering the similarity based grouping can be done accurately where the exact similarity of classes can be identified. The FCM clustering algorithm is given as follows:

Clustering of numerical data forms the basis of many classification and system modelling algorithms. The reason for clustering is to recognize the natural groupings of data from a large data set in order to create a concise representation of a system's behaviour. Fuzzy Clustering Means (FCM) is a method of clustering which allows one piece of data to belong to two or more clusters. It is based on minimization of the following objective function:

$$J_m = \sum_{i=1}^{N}\sum_{j=1}^{C} u_{ij}^m ||x_i - c_j||^2 \qquad 1 \leq m < \infty$$

Where m is any real number greater than 1, u_{ij} is the degree of membership of x_i in the cluster j, x_i is the ith of d-dimensional measured data, c_j is the d-dimension centre of the cluster, and ||*|| is any norm expressing the similarity between any measured data and the centre. Fuzzy partitioning is carried out through an iterative optimization of the objective function shown above, with the update of membership u_{ij} and the cluster centres c_j by:

$$u_{ij} = \frac{1}{\sum_{k=1}^{C}\left(\frac{||x_i - c_j||}{||x_i - c_k||}\right)^{\frac{2}{m-1}}}$$

$$c_j = \frac{\sum_{i=1}^{N} u_{ij}^m \cdot x_i}{\sum_{i=1}^{N} u_{ij}^m}$$

This iteration will stop when $max_{ij}\left\{|u_{ij}^{(k+1)} - u_{ij}^{(k)}|\right\} < \in$, where $\in$ is a termination criterion between 0 and 1, whereas k are the iteration steps. This procedure converges to a local minimum or a saddle point of J_m.

3.2.3. Range cardinality queries

In the existing paper [1, 18, 19], it uses a unique RecordId to find out whether the cardinalities that are get from different buckets belonging to the same record or not. We can adopt the algorithm explained in [1, 20, 21] to estimate the cardinality in the queries range.

4. RESULT AND ANALYSIS

Approximately 2,00,000 input data sets[22-27] are used to find the performance of FCM approach. The same data set is used for the existing system also and its performance measures are also calculated. After applying both the existing system and proposed system on the same input data sets the performance of proposed FCM approach is better than the existing FastRAQ approach. Sample input data sets used and the resultant graphs are described: The various performance measures are analysed in order to prove the proposed system is better than existing system. They are:i) time to execute the query, ii) accuracy and iii) error rate.

4.1. Accuracy performance

The accuracy of FCM approach is higher than the accuracy of all other existing approaches. This is shown in Figure 1.

4.2. Execution time

The time taken to execute the user entered range aggregate query is shown in Figure 2. The time taken to execute the query by FCM approach is less than the existing approach.

4.3. Error rate

The error rate graph for the user query is shown in Figure 3. The error rate of FCM approach is nearly 50% less than that of the existing approach error rate.

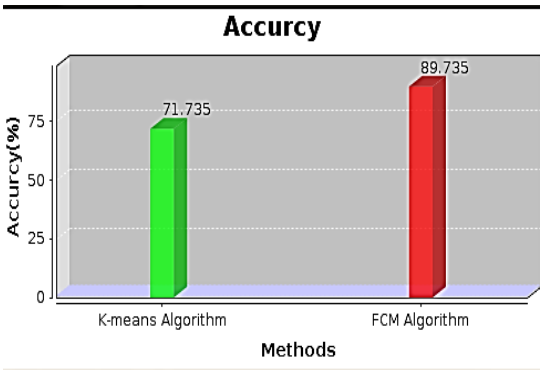

Figure 1. Comparison of accuracy

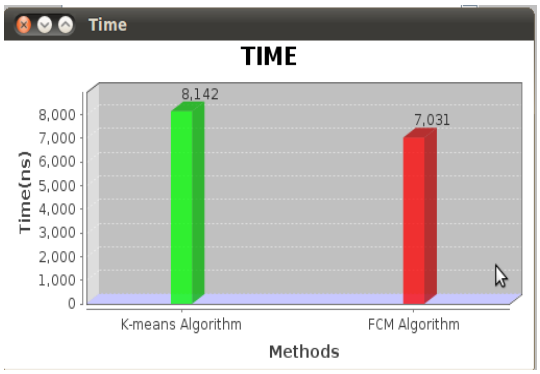

Figure 2. Comparison of execution time

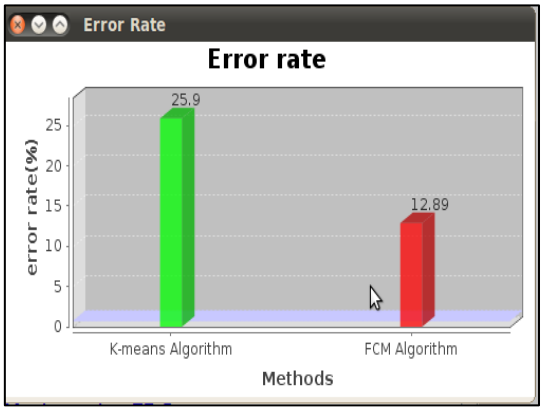

Figure 3. Comparison of error rate

5. CONCLUSIONS AND FUTURE ENHANCEMENT

In this paper, a new approach Fuzzy Clustering Means (FCM) approach is proposed that quickly acquires the accurate estimations for range-aggregate queries in big data environments. For ad-hoc range aggregate queries FCM have the time complexity of $O(nc^2p)$, where n is number of instances in the given dataset, c is cluster and p is data points. This time complexity is better than the previous existing algorithms.For future work it is planned to investigate how this solution can be extended to the case m:n format problem. Next it is planned to analyze how FCM can be used as a tool to boost the performance of data analysis in database.

REFERENCES

[1] X. Yun, *et al.*, "FastRAQ: A Fast Approach to Range-Aggregate Querues in Big Data Environments,"*IEEE Trans. Cloud Comput*, vol. 3, 2015.
[2] P. Mika and G. Tummarello, "Web semantics in the clouds," *IEEEIntell. Syst.*, vol. 23, pp. 82-87, 2008.
[3] H. Choi and H. Varian, "Predicting the present with Googletrends,"*Econ.Rec.*, vol. 88, pp. 2-9, 2012.
[4] C.T. Ho, *et al.*, "Rangequeries in OLAP data cubes," *ACM SIGMOD Rec.*, vol. 26, pp. 73-88, 1997.
[5] T. Preis, *et al.*, "Quantifying tradingbehavior in financial markets using Google trends," *Sci. Rep.*,vol. 3, pp. 1684, 2013.

[6] G. Mishne, *et al*., "Fast data in theera of big data: Twitter's real-time related query suggestionarchitecture," *Proc. ACM SIGMOD Int. Conf. Manage. Data*, pp. 1147-1158, 2013.
[7] W. Liang, *et al*., "Range queries indynamic OLAP data cubes," *Data Knowl. Eng.*, vol. 34, pp. 21-38, 2000.
[8] J. M. Hellerstein, *et al*., "Online aggregation,"*ACM SIGMOD Rec.*, vol. 26, pp. 171-182, 1997.
[9] P. J. Haas and J. M. Hellerstein, "Ripple joins for online aggregation,"*ACMSIGMOD Rec.*, vol. 28, pp. 287-298, 1999.
[10] E. Zeitler and T. Risch, "Massive scale-out of expensive continuousqueries," *Proc. VLDB Endowment*, vol. 4, pp. 1181-1188,2011.
[11] N. Pansare, *et al*., "Online aggregationfor large MapReduce jobs," *Proc. VLDB Endowment*, vol. 4, pp. 1135-1145, 2011.
[12] T. Condie, *et al*., "Online aggregation and continuousquery support in MapReduce," *Proc. ACM SIGMOD Int.Conf. Manage. Data*, pp. 1115-1118, 2010.
[13] Y. Shi, *et al*., "You can stop early withcola: Online processing of aggregate queries in the cloud," *Proc.21st ACM Int. Conf. Inf. Know. Manage*, pp. 1223-1232, 2012.
[14] K. Bilal, *et al*., "Onthe characterization of the structural robustness of data centernetworks," *IEEE Trans. Cloud Comput.*, vol. 1, pp. 64-77, 2013.
[15] S. De C. di Vimercati, *et al*., "Integrity for join queries in the cloud," *IEEETrans. Cloud Comput*, vol. 1, pp. 187-200, 2013.
[16] S. Heule, *et al*., "Hyperloglog in practice:algorithmic engineering of a state of the art cardinality estimationalgorithm," *Proc. 16th Int. Conf. Extending Database Technol*., pp. 683-692,2013.
[17] P. Flajolet, *et al*., "Hyperloglog:The analysis of a near-optimal cardinality estimation algorithm,"*Proc. Int. Conf. Anal. Algorithms*, pp. 127-146, 2008.
[18] http://research.neustar.biz/2012/12/17/hllintersections-2/, 2012.
[19] A. Thusoo, *et al*., "Hivea petabyte scale datawarehouse using Hadoop," *Proc. IEEE 26th Int. Conf. Data Eng.*, pp. 996-1005,2010.
[20] T. Yu and K.-J. Lin, "Adaptive algorithms for finding replacement services in autonomic distributed business processes," in *Autonomous Decentralized Systems*, 2005. ISADS 2005. Proceedings, pp. 427–434, April 2005.
[21] S. H. Ryu, F. Casati, H. Skogsrud, B. Benatallah, and R. Saint-Paul, "Supporting the dynamic evolution of web service protocols in serviceoriented architectures," *ACM Trans. Web*, vol. 2, no. 2, pp. 1–46, 2008.
[22] D. Mituzas,"Page view statistics for wikimedia projects," 2013. Available: http://dumps.wikimedia.org/other/pagecounts-raw/
[23] R. Bi, *et al*., "Optimizing Retransmission Threshold in Wireless Sensor Networks," *Indian Journal of Science and Technology*,vol. 16, pp. 665, 2016.
[24] Christoph Dorn, SchahramDustdar, Weighted Fuzzy Clustering for Capability-driven Service Aggregation.
[25] C. Platzer, F. Rosenberg, and S. Dustdar, "Web service clustering using multidimensional angles as proximity measures," *ACM Trans. Internet Technol.*, vol. 9, no. 3, pp. 1–26, 2009.
[26] F. Rosenberg, P. Leitner, A. Michlmayr, P. Celikovic, and S. Dustdar, "Towards composition as a service - a quality of service driven approach," 29 2009-April 2 2009, pp. 1733–1740, 2009.
[27] E. Maximilien and M. Singh, "Self-adjusting trust and selection for web services," pp. 385–386, June 2005.