International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
Volume: 03 Issue: 06 | June-2016 www.irjet.net p-ISSN: 2395-0072

Comparative Analysis of Range Aggregate Queries In Big Data Environment

Ranjanee S

PG Scholar, Dept. of Computer Science and Engineering, Institute of Road and Transport Technology, Erode, TamilNadu, India.

Abstract - *Approximated range aggregate query for real-time data analysis in OLAP presented Prefix-Sum Cube (PC) approach to solve the numeric data cube aggregation problems. 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. 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 approach to range-aggregate queries in big data environments which first divide 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, range queries obtains the result directly by summarizing local estimates from all partitions. In this paper we provide the comparative analysis between the techniques of MapReduce and Spark in Hadoop.*

Key Words: **Range-Aggregate query, data cube, balanced partitioning algorithm.**

1. INTRODUCTION

For organizations of all sizes, data management has shifted from an important competency to a critical differentiator that can determine market winners. These organizations are defining new initiatives and re-evaluating existing strategies to examine how they can transform their single technology, technique or initiative. Rather, it is a trend across many areas of business and technology. But today, new technologies make it possible to realize value from Big Data. For example, retailers can track user web clicks to identify behavioural trends that improve campaigns, pricing and stockage [1]. Utilities can capture household energy usage levels to predict outages and efficient energy consumption. Governments and even Google can detect and track the emergence of disease outbreaks via social media signals. "Big Data" describes data sets so large and complex they are impractical to manage with traditional software tools.

Specifically, Big Data relates to data creation, storage, retrieval and analysis that is remarkable in terms of volume, velocity, and variety.

In this work, defines the analytical approach to determine the efficient retrieval of the data for the purpose of analysis in big data environment using Hadoop. New technologies like NoSQL, MPP databases, and Hadoop have emerged to address Big Data challenges and to enable new types of products and services to be delivered by the business. One of the most common ways companies are leveraging the capabilities of both systems is by integrating a NoSQL database such as MongoDB with Hadoop. The connection is easily made by existing APIs and allows analysts and data scientists to perform complex, retroactive queries for Big Data analysis and insights while maintaining the efficiency and ease-of-use of a NoSQL database.

Range aggregate queries defined as by applying the given aggregation operation over the selected cells where the selection is specified as continuous ranges in the domain of the attributes. Aggregate operations are sum, count, average. A range query applies an aggregation operation (e.g., SUM) over all selected cells of an OLAP data cube. Selection is specified by providing ranges of values for numeric dimensions. Range sum queries on data cubes are a powerful analysis tool. However, the range aggregation done on dynamic data cubes require high cost on update.

2. LITERATURE SURVEY

In the era of internet Trend setting mainly focusses on how often a particular search-term is entered relative to the total search-volume across various regions of the world, and in various languages. By using big data analytics the prediction about future makes easy and it leads to improvement in businesses and improves the efficiency by recognizing the value of data and maintaining a dynamically adaptive infrastructure [2]. Due to massive data streams updation in the distributed cluster environment, there is a need for input streams to be spited into parallel sub streams for the continuous query execution. Stream splitting involves both partitioning and replication of incoming tuples, depending

on how the continuous query is parallelized which is described in [3]. Based on paper described in [3] for each input sub streams cardinalities to be estimated during the parallel execution by using an algorithm efficiently which are described in [4]. The problems associated with query processing and the cardinality estimation of the partitioned data sets results in the form called range aggregate close-pair problems. These type of problems may occur in Geographic Information Systems (GIS).The paper [5] provides the solution to the above problem by determining proximity checking for a given query.

3. PROBLEM IDENTIFICATION

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. Typical range-aggregate query problem 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 seconds on massive data analysis.

4. EXISTING SYSTEM

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 (OLA) was proposed to speed range-aggregate queries on larger data sets. OLA has been widely studied in relational databases and the current cloud and streaming systems. Some studies about OLA have also been conducted on Hadoop and Map Reduce. 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 cannot respond with acceptable accuracy within desired time period, which is significantly important on the analysis of trend for ad-hoc queries.

5. PROPOSED SYSTEM

Propose a new approximate answering approach that acquires accurate estimations quickly for range-aggregate queries in big data environments. It first divides big data into independent partitions with a balanced partitioning algorithm. When a range aggregate query request arrives range queries obtains the result directly by summarizing local estimates from all partitions.

The balanced partitioning algorithm works with a stratified sampling model. It divides all data into different groups with regard to their attribute values of interest, and further separates each group into multiple partitions according to the current data distributions and the number of available servers.

6. CONTRIBUTIONS

In this paper, the contributions to be done are as follows

- Using of Hadoop Distributed File System (HDFS) provides the distributed file system and a framework for the analysis and transformation of very large data sets using the MapReduce paradigm.
- Using of Spark speed up the Hadoop computational computing software process. Spark uses Hadoop in two ways – one is **storage** and second is **processing**.

6.1 Apache Hadoop

In a traditional non distributed architecture, whereas the data stored in one server and any client program will access this central data server to retrieve the data. The non-distributed model has few fundamental issues. In this model, mostly scale vertically by adding more CPU, adding more storage, etc. This architecture is also not reliable, as if the main server fails, you have to go back to the backup to restore the data. From performance point of view, this architecture will not provide the results faster when running a query against a huge data set.

In a hadoop distributed architecture, both data and processing are distributed across multiple servers. The following are some of the key points to remember about the hadoop:

- Each and every server offers local computation and storage.
- In simple terms, instead of running a query on a single server, the query is split across multiple servers, and the results are consolidated. This means that the results of a query on a larger dataset are returned faster.
- There is no need for a powerful server. Just use several less expensive commodity servers as hadoop individual nodes.
- High fault-tolerance

6.2 Apache Spark – RDD

Resilient Distributed Datasets

Resilient Distributed Datasets (RDD) is a fundamental data structure of Spark. It is an immutable distributed collection of objects. Each dataset in RDD is divided into logical partitions, which may be computed on different nodes of the cluster. RDDs can contain any type of Python, Java, or Scala objects, including user-defined classes.

Formally, an RDD is a read-only, partitioned collection of records. RDDs can be created through deterministic operations on either data on stable storage or other RDDs.There are two ways to create RDDs − **parallelizing** an existing collection in your driver program, or **referencing a dataset** in an external storage system, such as a shared file system, HDFS, HBase, or any data source offering a Hadoop Input Format.

7. SYSTEM ARCHITECTURE

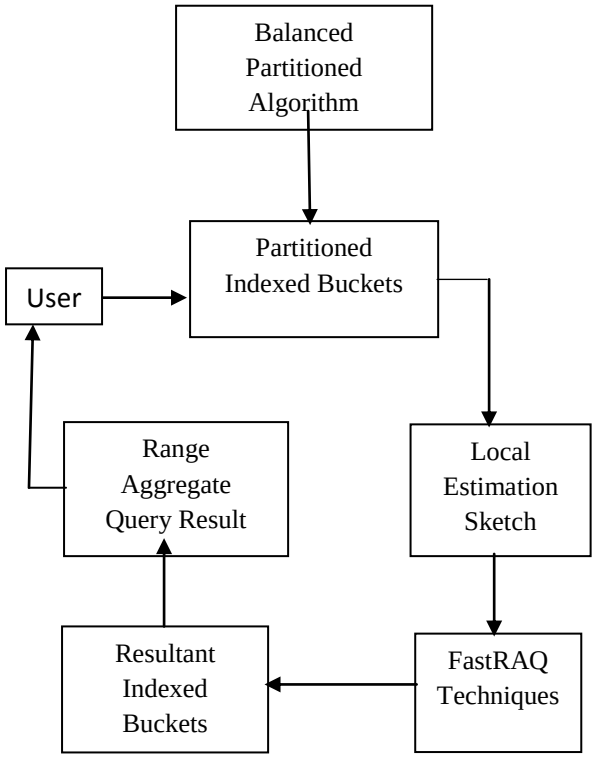

Fig -1: System Architecture

8. MODULE DESCRIPTION

8.1 Data Sets Upload Module

In this module the data is uploaded to the HDFS for processing. The data may be structured or unstructured. However in most of the cases the data used in the big data environments are unstructured. Unstructured data is data that does not follow a specified format for big data. The survey says 20 percent of the data available to enterprises is structured data, the other 80 percent is unstructured. Until recently, however, the technology didn't really support doing much with it except storing it or analysing it manually. Just as with structured data, unstructured data is either machine generated or human generated. The data sets are uploaded in HDFS (Hadoop Distributed File System).

8.2 Divide Independent Partitions Module

In this module the proposed work involves the algorithm called balanced partitioning algorithm. In this algorithm, the uploaded data sets are divides into partitions based on the index and the cardinalities are estimated. The balanced partitioning algorithm works with a stratified sampling model. It divides all data into different groups with regard to their attribute values of interest, and further separates each group into multiple partitions according to the current data distributions. 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.

8.3 Local Estimation Sketch Module

The estimation sketch is a new type of multi-dimensional histogram that is built according to learned data distributions. It can maintain nearly equivalent frequencies for different values within each histogram bucket, even if the frequency distributions in different dimensions vary significantly. When a query request arrives, it is delivered into each partition. First build cardinality estimator (CE) for the queried range from the histogram in each partition. Then 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.

8.4 Range-Aggregate Query Results Module

An aggregate function operates on selected ranges, which are contiguous on multiple domains of the attribute values. In this attribute values can be numeric or alphabetic. It 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, range queries obtains the result directly by summarizing local estimates from all partitions.

8.5 Comparative Module

Using the spark accelerates the running time of Map reduce processing. Generally the Hadoop provides the map reduce processing in two steps namely the transformation and reduction, whereas in spark the transformation step not known to the user. The user retrieve the reduced output. Therefore it reduces the space and time complexities of processing.

In the map phase of Hadoop, for example If there are 6000 (R) reducers and 2000 (M) map tasks, there will be (M*R) 6000*2000=12 million shuffle files. This is because, in Hadoop, each map task creates as many shuffle spill files as number of reducers. This caused performance degradation. In the spark processing, the shuffle files are consolidated he only difference being, the map tasks which run on the same cores will be consolidated into a single file. So, each CORE will output as many shuffle files as number of reducers. Example: If there are 6000(R) and 4 (C) Cores, the number of shuffle files will be (R*C) 6000*4=8000 shuffle files. Note the huge change in the number of shuffle files.

The reduce phase of Hadoop and Spark are same. The comparison between Apache Hadoop and spark are as follows,

Table-1: Comparison of Hadoop MapReduce and Spark

PHASES	HADOOP MAPREDUCE	SPARK
MAP	Shuffle spill Files are merged and partitioned.	Shuffle spill files are not merged and partitioned.
REDUCE	Shuffle files from cache are reduced and loaded to memory.	Shuffle files are reduced and directly loaded into memory.

Running Time of spark processing is given by the figure as follows,

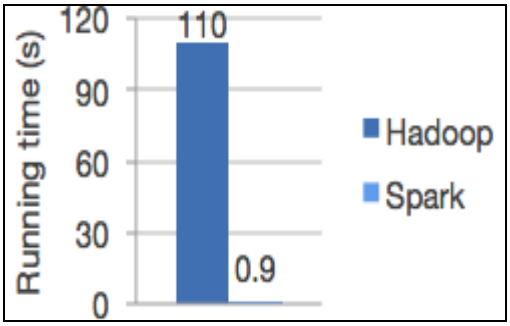

Fig-2: Time Complexity

9. RESULTS

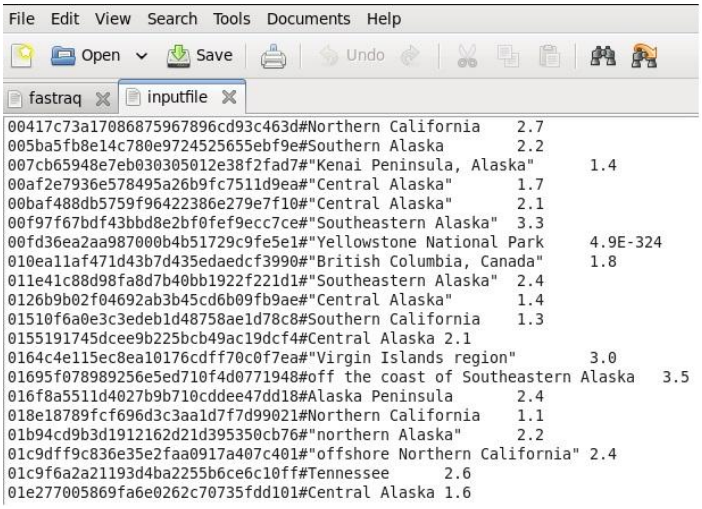

Fig -3: Input file for Hadoop and spark processing

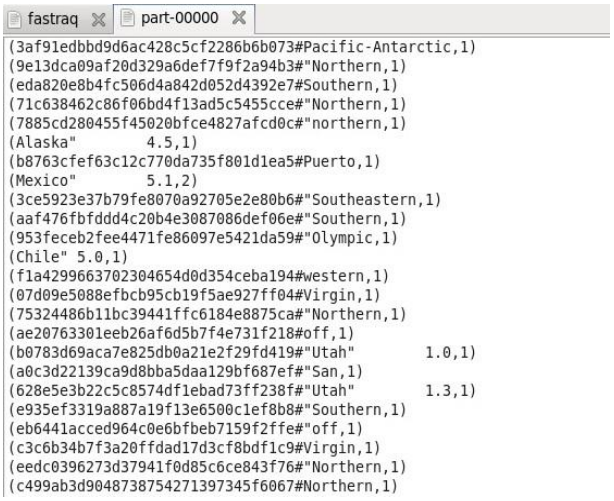

Fig-4: Hadoop result

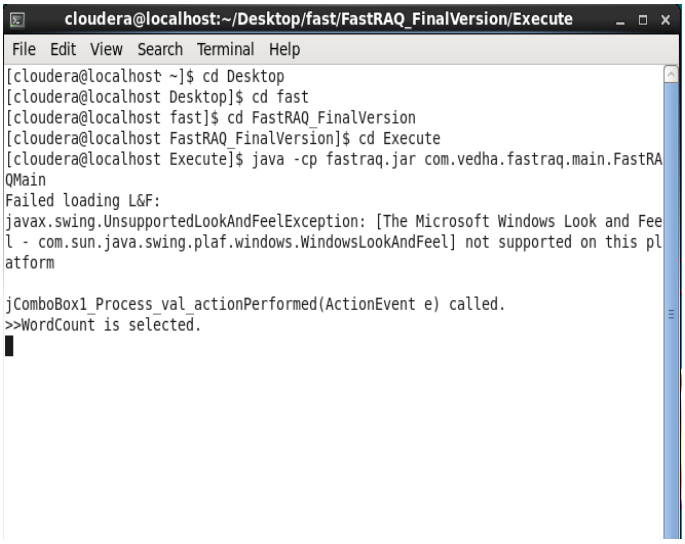

Fig-5: Process using spark

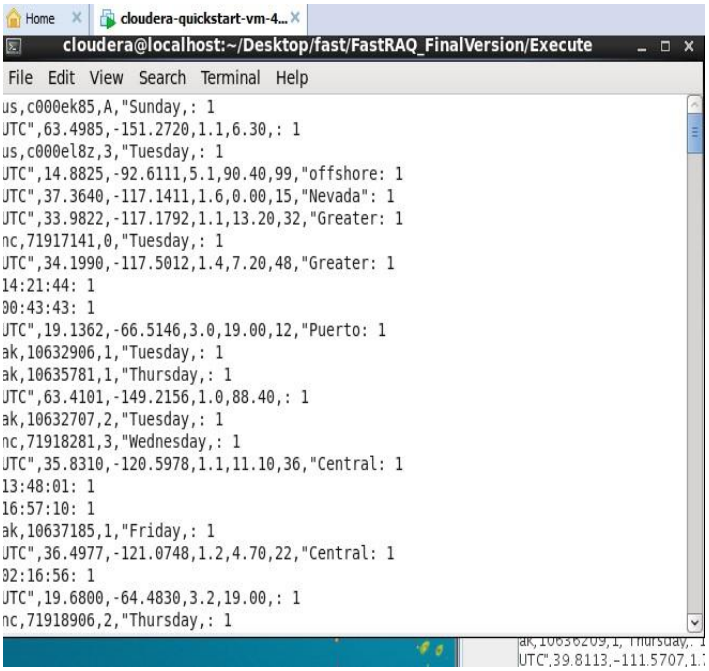

Fig-6: Spark result

10. CONCLUSION

In the proposed work, new approximate answering approach that acquires accurate estimations quickly for range-aggregate queries in big data environments. It reduces the time complexity for data updates and improves the efficiency of retrieval results.

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

[4] 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.

[5] R.Sharathkumar and P. Gupta, "Range-aggregate proximity queries," IIIT Hyderabad. VTelangana 500032, India, Tech. Rep. IIIT/ TR/2007/80, 2007.

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

[7] 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.

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

[9] 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.

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