Online Aggregation Using MapReduce in MongoDB

B Rama Mohan
Dept. of CSE, JNTUH College of Engineering Hyd
JNT University Hyderabad, India

A Govardhan
Professor, School of Information Technology
JNT University Hyderabad, India

Abstract -- In Traditional Query Processing System, Aggregation had done using the Batch mode to obtain the exact result. Online aggregation is a framework that utilizes the idea of an enhanced batch mode to obtain the approximate result of a query. To provide the approximate result for the large database in terms of terabytes or petabytes, the MapReduce concept implemented in the online aggregation. Many methodologies and algorithms designed to obtain an approximate response in less time compared to the time taken to compute an exact answer. The concept of Hadoop introduced into the MapReduce paradigm for its implementation, until now. In this paper a new methodology proposed where the NoSQL applied into the MongoDB open software that uses MapReduce paradigm along with online aggregation for implementation.

Key words -- Online aggregation, MapReduce, MongoDB, NoSQL, Hadoop

I. Introduction

A. Overview

Traditional query processing systems has concentrated exclusively on providing exact solutions to the queries by minimizing the response time and maximizing the throughput. However, there are a number of environments where the response time in producing an exact answer is often slower than the desirable. In large data, warehousing environments obtaining the exact result from the large complex queries are taking very long time. Even for the data having terabytes or more of size, the single scan of the whole database is taking millions of seconds. In case of distributed data warehousing environments when some of the data is remote or not available, the response time for such environment is very low.

A number of scenarios exist where the exact answers to the query are not essentials and the user may prefer for an approximate result. For Example the approximate results is obtained while performing the drill down operation on the ad hoc data mining to determine the interesting queries with the help of earlier queries and gives feedback on how well a query is posed. Moreover, it provides an uncertain result to the query when the database is unreachable. The other sample is when the query demands for the numerical results; the complete precision of the exact solution not needed. In this environment, the exact answer is not required until the data again becomes accessible. Thus, the environments that provide exact results with the unwanted response time motivated the study of techniques for obtaining approximate answers to the queries. To estimate the output, there is need for the techniques that gives fast approximate results, which can also use in a more systematic way within the query optimizer. Instead of exact results, these types of applications require fast response. Despite of the recent works in approximate query results it is limited to its speed, scope, and accuracy.

B. Online Aggregation

To produce an approximate result in less time compared to the time taken to compute an exact answer by reducing the number of access to the base data several methodologies came into existence. In which Query aggregation in DBMS is a traditional method performed in batch mode. In this process, initially the query submitted then the system processes on the huge volume of data over a long period and eventually returns the final solution. The need of any aggregation query is to get normal information regarding the data where the results computed with the exact precision, even though a suitably precise approximation is available very fast. Thus, the batch mode returns the results that are aggravated and lead to unnecessary delays to the user.

In order to address this problem, [1, 2] had recently proposed a system that supports aggregation processing and gives the approximate result within less time. This framework is proposed to obtain the approximate results for the queries using the aggregation where the database is scanned in random order and the results of the aggregate query is updated eventually as the scan proceeds. It is a technique for improving the interactive behavior of database systems processing with expensive analytical queries. This system performs the aggregation query in the online fashion. The graphical display shown in Fig 1.1 depicts the results with confidence intervals as the scan proceeds so that the user can stop the process at any time needed. In this system, users can both observe the improvement of their aggregation queries and control performance of these queries on the fly. The interface for the online aggregation system is more flexible and comfort mechanism for data exploration and exploitation compared to the traditional batch-mode query processing system.
C. MapReduce

The MapReduce is one of the programming model and a framework for data-intensive distributed computing used in batch mode. The key notion for the MapReduce model is to allow the users to focus on data processing mechanisms and hide the details of parallel execution. For the processing of large-scale database systems to obtain the approximate results without complete execution, the online aggregation can build into the MapReduce system. The MapReduce parallelization and Online Aggregation combined to achieve Time-to-Solutions beyond the today’s systems. To simplify fault tolerance, the output of each Map and Reduce task materialized to disk before consumed. A modified MapReduce architecture proposed to pipeline the data between operators. Apart from the traditional batch-processing model, the MapReduce programming paradigm reduces the execution time and improves the system utilization for all types of jobs. This modified MapReduce system supports Online Aggregation, and permits the user to return the results early from a job.

The MapReduce programming model has evolved as a famous way to bind the large clusters of computers together. MapReduce permits the programmers to think of a data-centric model where it mainly emphasis on applying transformations to sets of data records and allow the details of distributed execution, network communication, coordination and fault tolerance to be controlled by the MapReduce framework. This programming model eventually applied to the large batch processing computation models that concentrate mainly on time to job completion. The Google MapReduce framework [3] and the open-source Hadoop system support this model through a batch-processing application strategy. The major advantage of this programming paradigm is the ease of parallelization. The primary goal of MapReduce is its fault tolerance and scalability. Apart from its ability, the MapReduce also has some pitfalls compared to traditional DBMS.

Some of the pitfalls of the MapReduce framework are:

1) No high-level language: The Map and Reduce are the functions where the user should code their own operations into it. Thus, it does not support any high-level language any query optimization technique.

2) No schema and no index: The MapReduce works only if its input is stored in the database. It is schema-free and index-free paradigm. MapReduce requires parsing each item at reading input and transforming it into data objects for data processing, causing performance degradation [4, 5].

3) Single fixed dataflow: The MapReduce paradigm originally designed to support only single input and generate single output. It does not work for multiple inputs and output. This provides the ease of use with a simple abstraction, but in a fixed dataflow. Therefore, many complex algorithms are hard to implement with Map and Reduce functions.

4) Low efficiency: MapReduce operations not always optimized for I/O efficiency. As Map and Reduce are blocking operations, the transition to the next stage not done until all the tasks of the current stage is finished. MapReduce often shows poorer performance than DBMS [5].

The Organization of this proposed paper done in this way. Section I already discussed about the introduction of the terminologies Online Aggregation and MapReduce. Section II gives the brief discussion about the improvements in the Online Aggregation and Map Reduce, Section III gives the brief discussion of the proposed methodology with its
detailed implementation, Section IV discusses about the observed results and its analysis. Section V concludes the paper followed by References and Acknowledgement of the paper given in Section VI and Section VII.

II. Related Work

Recently several improvements have done on the query processing systems to provide an approximate result to the queries quickly. Bayardo and Miranker [6] proposed a mechanism for optimizing and executing queries using pipelines, nested loops joins to minimize the latency and obtain the results. The Oracle Rdb system [7] supports for running multiple queries simultaneously in order to retrieve the fast first query processing results. Barbar et al [8] presents a survey of data reduction techniques that are widely used for a variety of purposes that include providing approximate query answers. Matias et al [11] proposed and studied approximate data structures to obtain fast approximate answers.

The concept of online processing expanded in [12, 13, 14, 15]. Recently the details of two of the core query processing algorithms presented: ripple joins [16] and online reordering [17, 18]. Estimation and confidence interval techniques for online aggregation are presented in [19, 20] and [16]. The earliest work on approximate answers to decision support queries appears in Morgenstern’s dissertation from Berkeley [21]. The concept of online aggregation built on the work done on estimation and confidence intervals in the database environment [22, 19, 23]. The prior work has been concerned with methods for producing a confidence interval with a width and stated prior to the start of query processing. The notion of these methods is to successfully maintain a running confidence interval and stop the sampling once the length of the interval is sufficiently small. [24] The problem of producing a confidence interval of minimal length when a real time stopping condition given. The drawback of using sampling to produce approximate answers is that the end-user needs to understand the statistics. Online Aggregation been studied for some time in the context of classic, SQL databases [25, 26, 27, 28] and more recently for peer-to-peer systems [29]. However, in the context of MapReduce, the only work that considers Online Aggregation (without ignoring MapReduce’s open programmability and fault tolerance) is the Hadoop Online Prototype (HOP) system [30]. An alternative to Online Aggregation is pre computed synopsis, where the system uses summary statistics (computed prior to the execution of the query) to provide approximate answers [31, 32].

The MapReduce framework initially developed at Google [3], has recently seen wide implementation, and has become the effective standard for large-scale data analysis. Openly presented statics clearly indicates that MapReduce processes more than 10 petabytes of data per day at Google alone [33]. In recent years, several non-trivial MapReduce algorithms have emerged, from computing the diameter of a graph [4] for implementing the EM algorithm to cluster massive data sets [34]. Kang et al. [34] presented how to use MapReduce to compute diameters of massive graphs. Tsourakakis et al. [35] used MapReduce for counting the total number of triangles from a graph. Das et al. [36] implement the EM clustering algorithm on MapReduce with the motivation of personalized new results. Overall, each of these works gives practical MapReduce algorithms, but does not thoroughly define the framework to analyze. Feldman et al. [37] introduced the notion of Massively Unordered Distributed (MUD) algorithms, a model based on the MapReduce framework. While modeling the same underlying system, their approach has two crucial differences.

A. Existing Methodology

In online aggregation systems, the tuples sampled from the input relations and a continually purifying running estimate of the obtained results computed, along with the confidence interval in order to indicate the precision of the running estimate. These confidence intervals usually presented as error bars in a graphical user interface. The precision of the running estimate increases as more and more input tuples are processed. The running aggregate is a statistical estimator of the final query result. With the associated running confidence interval, one can estimate the proximity of the running aggregate of the result. The running confidence intervals are an important component of an online aggregation system.

In the Map Reduce programming paradigm, the fundamental unit of statistics is a key and value pair where each key and value are binary strings. The input to any MapReduce algorithm is a set of (key, value) pairs. Operations on these set of pairs done in three stages: the map stage, the shuffle stage and reduce stage. In the map stage, the mapper takes as input a single (key; value) pair, and produces as output any number of new (key; value) pairs. The Map operation operates only on one pair at a time and is stateless. Since different inputs to the map different machines, the parallelization done easily in the map operation can process the operation.

In the shuffle stage, all the values that are associated with the individual or a single key in the system send to the similar machines that are bearing the same key. The Shuffle stage unifies to the programmer in MapReduce. In the reduce stage, the reducer accepts all the values associated with a single key k, and gives output a multiple sets of (key; value) pairs with the same key k. This is one of the sequential aspects of MapReduce computation. All the map operations has to finish before reduce operation begins. Since the reducer has access to all the values with the same key, it can perform sequential computations on these values. Thus, the MapReduce programming paradigm consists of many rounds of different map and reduces operations performed one after another.

The primary feature of the MapReduce algorithm is that if every Map and Reduce operation is independent of all other ongoing Map and Reduce operations. These operations executed in parallel for different keys and lists of data. On a large cluster of machines, the map operation performed on several servers where the data is live. Rather than copying the data over the network, the program pushed into the machines. The output list then saved to the distributed
III. Proposed Methodology

The traditional DBMS that is being used in our daily work gives the result of the executed complex Query in thousand and millions of minutes/seconds of any large data set only after the complete execution. A methodology proposed where the MapReduce concept introduced into the MongoDB with NoSQL as a back end to implement the online aggregation. The SQL used to retrieve the data from the database that provides the result after the complete execution of all the data sets in the query. However, the inclusion of MapReduce in the Online Aggregation gives the approximate results for the data set interested by the user. The Map and Reduce operations executes in the MongoDB open software and use the NoSQL database as a back end. The approximate results obtained only for some tuples and the results updated regularly using the MapReduce along with the online aggregation.

MongoDB is a schema-free document-oriented database written in C++. It is developed in an open-source project and primarily driven by the company 10gen Inc. It also offers professional services around MongoDB. According to its developers the main goal of MongoDB is to close the gap between the fast and highly scalable key-value-stores and feature-rich traditional RDBMS relational database management systems. MongoDB name derived from the adjective humongous [39]. Prominent users of MongoDB include SourceForge.net, foursquare, the New York Times, the URL-shorted bitsy, and the distributed social network DIASPORA [40, 41, 42, 43, 44]. It is part of the NoSQL family of database systems. The classical relational database stores the data in the form of tables, apart from this in order to have the easier and faster access to data for applications, MongoDB is storing the data as JSON – like documents with dynamic Schemas. MongoDB stores the objects in a binary format called BSON. This has the added advantage of letting us efficiently retrieve a specific range of the given file. The term NoSQL was first used in 1998 for a relational database that limited the use of SQL [45]. The term was elated again in 2009 for conferences of advocates of non-relational databases such as Last.fm developer Jon Oskarsson, who structured the NoSQL meet up in San Francisco [46]. One of the Rackspace employee Eric Evans defined the motivation of the NoSQL movement as the whole point of pursuing substitutions to answer a problem than the relational databases.

Compared to the traditional RDBMS, NoSQL databases are providing significant data results. Along with this, the NoSQL databases are also providing various methods for the purpose of storage and retrieval of large data than the traditional database. Motivations for this approach comprise simplicity of design, horizontal scaling, and finer control over availability. NoSQL databases are extremely enhanced key-value stores proposed for simple retrieval and adding operations to obtain important performance benefits in terms of latency and throughput. The NoSQL databases are now a day having large and significant use in the industries where the data is growing day by day and in real time applications. Sometimes the NoSQL systems are also known as “Not only SQL!” in order to underlie that the SQL Query language can also be used into NoSQL at necessities.

To accomplish the aggregation of query results MongoDB provides the count, distinct and the group operation that may invoke via programming language libraries but executed on the database servers [47]. The count operation returning the number of documents matching a query invoked on a collection and takes selection criteria that specified in the same way as for the find operation:

\[
\text{db. < collection >. Count(<criteria>);}
\]

If the count invoked with empty criteria and returns the number of documents in the collection. If the selection criterion is used, the document fields used in it should index to accelerate the execution of the count.

MapReduce also allows implementing custom aggregation operations in addition to the predefined count, distinct and group operations. An approach to server-side code execution especially suited to batch manipulation and aggregation of data is MapReduce [48]. MongoDB’s MapReduce implementation is similar to the concepts described in Google’s MapReduce paper [4] and its open-source implementation Hadoop: there are two phases—map and the reduce—in which code written in JavaScript is executed on the database servers and results in a temporary or permanent collection containing the outcome. MapReduce architecture used to improve the performance of the aggregate calculation results like the COUNT, AVG, and SUM etc. on a NoSQL database and use this to generate the Online Aggregation results for the database in very less time. The MongoDB shell as well as programming language drivers—provides syntax to launch such a MapReduce-fashioned data processing:

\[
\begin{align*}
\text{Map\_function} & : \quad \text{"emit (this.event\_code, this.tab\_window);"} \\
\text{Reduce\_function} & : \quad \text{var\ reduce=}\text{ function (boole, values) \{return Array. sum (values);\}} \\
\text{MapReduce from MongoDB} & : \quad \text{db.collection.mapReduce (}
\end{align*}
\]

© 2013, IJARCSSE All Rights Reserved
sort: <document>, sorted order of the records
limit: <number>, limit the input
finalize: <function>, the finalize function
scope: <document>, scope of the document to perform mapreduce
jsMode: <boolean>, js Mode
verbose: <boolean> Verbosity

MapReduceCommand cmd = new MapReduceCommand (mycoll, map, reduce, null, MapReduceCommand.OutputType.INLINE, null);

A. Algorithm for MapReduce in Online Aggregation

The Algorithm 1 gives the Brief Description of MapReduce paradigm along with the online aggregation to obtain the approximate results for the large set of data batch wise. The execution of MapReduce paradigm and the online aggregation are done simultaneously. For Example if the MongoDB consists of 5 millions of records, the MapReduce paradigm in the NoSQL Query processing system applied to the first 10,000 records (for example). As the obtained reduced data set will have a less number of records. These send to the online aggregation to perform the execution and obtain the confidence interval for the reduced data set. While performing the online aggregation for the obtained reduced data set, the MapReduce paradigms applied to next 10,000 records simultaneously. The approximate results obtained for the block i.e. 10,000 records and then the approximate result updated regularly using the online aggregation. The advantage for the execution of large data set with the online aggregation and MapReduce is that it has the flexibility to stop the process whenever required. The Block Diagrams of the MapReduce paradigm and the Online Aggregation shown in the Fig 1 and Fig 2.

Algorithm 1:
1. Consider the collection of large data set in terms of terabytes or petabytes in the open software MongoDB that uses the NoSQL Query Processing.
2. In the large data set the MapReduce functions are performed only on the collection of few record/tuples (for Example: 1000 records at a time)
   i. First the map function is performed where the number of values are mapped to the same keys
   ii. Then reduce function is performed for the same keys in order to combine the values at one place.
3. The steps (i) and (ii) performed until the data set is completely reduces.
4. Then the resultant reduced data set is sent to the online aggregator to estimates the count, confidence and interval for the data set and the results are updated from the previous to new ones.
5. The step from one to four continues until all the data set in the MongoDB gives approximate results.

![Block Diagram of Online Aggregation](image)

**Fig. 2 Block Diagram of Online Aggregation**

### IV. Experimental Results

An implementation of online aggregation is done here in sensors where data is collected may be after 2 or 3 hours. Thus, the new results can compute based on previous and present values. The format of the database can be like the scoreboard in cricket, where the values run on a cricket board after every over and gradually update the score. Once the match is finished, the exact result obtained. However, if the score based on run rate predicted the result might vary based on runs in next overs.

Equal and the similar database used for both SQL Data Base and MongoDB for accessing of the results. A CSV file containing 5 million records of windows users have been imported that accessed the Mozilla Firefox database. Fig 3 shows the output result of the database containing 5 million records and the total time taken to execute the complete database is 7 seconds or 6781082834 nanoseconds.

![Results obtained from the SQL database](image)

**Fig 3: The Results obtained from the SQL database**

The fig. 4 shows the output result of the online aggregation where the complete execution of the total database not done. The start and stop buttons in the interface are used to start or stop the execution whenever needed in cases
where the exact or final result is not needed. The result shows the count of the database, with 95% confidence and the interval.

![Image of the Online Aggregation Applet]

Fig 4: The Result of Online Aggregation Applet

1. **Start and Stop Button**: Which used to start and stop the aggregation process. If the stop is not pressed or the time is allowed to elapse until the end, then the online aggregation produces accurate results which is the cause of MapReduce () execution. However, if one were to come to a conclusion about the estimate of the aggregate value, COUNT, they can stop anywhere they need.

2. **Count**: It is the estimate of the count of the values calculated on the database.

3. **Confidence**: It is the support confidence of the interval, initially taken as 95%.

4. **Interval**: It is the range by which our estimated count may differ in case of stopping the applet in the middle

V. Conclusions

The Online aggregation uses the MapReduce concept to get the approximate results using the MongoDB NoSQL. In the proposed methodology, a large set of data given to the MongoDB. The approximate results obtained by MapReduce paradigm for interested data sets and blocks of data updated regularly. The MapReduce model is simple to use. It allows scaling of applications across massive clusters of machines comprising thousands of nodes, with fault-tolerance inbuilt for ultra-fast performance. Online aggregation, which can be useful when the data collected from massive clusters, can be very advantageous when it comes to the real time applications like collecting and estimating the data from sensors, various Facebook or google search implementations. Combining two areas the study has aimed at allowing faster and accurate calculations of aggregations in real time scenarios like Sensor Data Collection using Online Aggregation.

References


B. Rama Mohan is presently an Associate Professor, Dept. of Computer Science & Engineering, Jawaharlal Nehru Technological University College of Engineering Hyderabad (Autonomous), India. He did his B.E in Electronics and Communication Engineering (ECE) from Osmania College of Engineering, Hyderabad. He did his M.Tech in Computer Science (CS) from Jawaharlal Nehru Technological University Hyderabad (JNTUH). He has 23 years of Teaching and Research experience. His areas of research are in Databases and Software Engineering.

Dr. A. Govardhan is presently a Professor of Computer Science & Engineering at School of Information Technology, Jawaharlal Nehru Technological University Hyderabad (JNTUH), India. He did his Intermediate from Andhra Pradesh Residential Junior College (APRJC), Nagarjuna Sagar in 1988, B.E. (CSE) from Osmania University College of Engineering, Hyderabad in 1992, M.Tech from Jawaharlal Nehru University (JNU), New Delhi in 1994 and Ph.D. from Jawaharlal Nehru Technological University, Hyderabad in 2003. He is the recipient of 21 International and National Awards including A.P. State Best Teacher Award, Bharat Seva Ratna Puraskar, CSI Chapter Patron Award, Bharat Jyoti Award, International Intellectual Development Award and Mother Teresa Award for Outstanding Services, Achievements, Contributions, Meritorious Services, Outstanding Performance and Remarkable Role in the field of Education and Service to the Nation. He is a Member of Executive Council, JNTU Hyderabad. He is a Chairman and Member on several Boards of Studies of various Universities. He is the Chairman of CSI Hyderabad Chapter. He is a Member on the Editorial Boards for Eight International Journals. He is a Member of several Advisory Boards & Academic Boards, a Committee Member for several International and National Conferences. He has 2 Monographs by Lambert Academic Publishing, Germany and Published in USA. He has guided 15 Ph.D. theses, 125 M.Tech projects and he has published 152 research papers at International/National Journals/Conferences including IEEE, ACM, Springer and Elsevier. He has delivered more than 50 Keynote addresses and invited lectures. He has 20 years of Teaching and Research experience. He served and held several Academic and Administrative positions including Director of Evaluation, Principal, Head of the Department and Students’ Advisor. He is a member in several Professional and Service oriented bodies. His areas of research include Databases, Data Warehousing & Mining and Information Retrieval Systems.