MC-OLA: Markov Chain-Based Multi-Table Join Online Aggregation Technique Postprint

Authors: Shi Yingjie, Du Fang

Date: 2018-10-11T00:00:00+00:00

Abstract

Multi-table join represents a critical query type in the domain of big data analytics. However, the substantial implementation cost of join queries compromises the timeliness of analytical results. Online aggregation addresses this challenge by providing statistically meaningful estimates before query completion, thereby holding significant practical importance. Existing multi-table join online aggregation algorithms employ uniform random sampling across tables, which results in low yield rates of join results and poor estimation accuracy for grouped join queries. To remedy this limitation, we propose a Markov chain-based online aggregation technique for multi-table joins that transforms the join implementation process into a random walk on a Markov chain. This approach creates stratified samples at the walk' s starting layer after determining the join order, and designs corresponding sampling strategies and result estimation methods. We implement the proposed technique on an online Hadoop platform, and experimental results demonstrate that our scheme achieves superior response time compared to existing algorithms while exhibiting good scalability.

Full Text

Preamble

MC-OLA: Multi-table Join Online Aggregation Based on Markov Chain

Shi Yingjie[1], Du Fang[2]
(1. School of Information Engineering, Beijing Institute of Fashion Technology, Beijing 100029, China;
2. School of Information Engineering, Ningxia University, Yinchuan 750021, China)

Abstract: Multi-table join is one of the most important query operations in the field of big data analysis; however, the implementation of multi-table join

is expensive, which affects the timeliness of big data analysis results. Online aggregation provides feedback of statistical significance far before the query finishes, which is of great significance. Existing work on multi-table join online aggregation conducted uniform sampling on every joining table, which results in low join result yield and estimation inaccuracy on grouping join queries. To address this problem, this paper proposed the multi-table join online aggregation technique based on Markov chain, transformed the multi-table join process into the random walk on Markov chain, constructed stratified sample on the walk start strata after determining the join order, and designed the corresponding sampling mechanism and estimation algorithm. The experiment was conducted on the online Hadoop platform, and the results demonstrate that the response time of technique proposed in this paper outperforms the existing algorithms, and it owns efficient scalability.

Key words: online aggregation; Markov chain; stratified sampling; multi-table join

0 Introduction

Social media, mobile devices, and sensors continuously generate massive amounts of data at an unprecedented rate. Exploring the value behind this data has become a major concern in both industry and academia. However, complex data analysis tasks run slowly on massive datasets, significantly diminishing the timeliness and value of analysis results, creating a bottleneck for data-driven applications. Ad-hoc interactive data analysis plays an important role in decision support, trend analysis, and data visualization, making it one of the urgent problems to be solved in the current big data analysis field. Online aggregation continuously processes partial sample data, thereby returning statistically meaningful estimated results within a short time, providing a novel solution for ad-hoc interactive data analysis.

Online aggregation was first proposed in the relational database field in the 1990s and subsequently achieved a series of research results, yet its impact on the relational database market was limited. With the emergence of big data and cloud computing platforms, new data models and management approaches have brought development opportunities for online aggregation. However, current research on online aggregation in cloud computing platforms mostly focuses on single-table operations or simple two-table joins, with relatively few studies addressing multi-table joins. Multi-table join is one of the most important operations in decision support, data mining, and analysis. In the TPC-H benchmark for big data decision support applications, 17 out of 22 query statements are join queries, involving up to 8 tables.

Compared with single-table or two-table join online aggregation, multi-table join online aggregation is more complex, and existing work cannot be directly applied. (a) Multi-table join types are diverse, including chain joins, acyclic

joins, cyclic joins, etc., and the online query processing methods and result estimation methods differ for different join types; (b) The result space of multi-table joins grows exponentially with the number of tables, while selectivity is typically low, causing existing sampling methods to result in extremely low yield of multi-table join results; (c) The overall data distribution of multi-table joins is not simply determined by one table but is the result of interactions among multiple tables, making existing algorithms for solving small-group problems inapplicable. To address these issues, this paper proposes MC-OLA, a multi-table join online aggregation technique based on Markov chain, which transforms the multi-table join processing into a random walk process on a Markov chain. Based on this model, stratified samples are created at the walk start point, and corresponding sampling strategies and result estimation methods are designed. This paper implements MC-OLA on the online Hadoop platform HOP, and experimental results demonstrate that MC-OLA outperforms random walk and ripple join algorithms in performance and possesses good scalability.

1 Related Work

Online aggregation was first proposed in the relational database field in the 1990s, with its main idea being to continuously sample from the entire dataset during query processing, estimate query results and confidence intervals based on samples, and continuously improve estimation quality as more data is processed. For single-table query online aggregation, reference [4] provides specific calculation methods for unbiased estimation and confidence intervals.

With the development of big data technology, online aggregation has attracted widespread attention in the cloud computing field. Reference [5] proposed a method for implementing online aggregation using Bayesian theory based on the MapReduce framework, considering the relationship between the aggregation value of each data block and its processing time, and statistically modeling the data block' s aggregation value, scheduling time, and processing time together. This method assumes that the longer the processing time of a data block, the larger its aggregation value, which does not hold for all aggregation operations, and the implementation is also relatively complex. G-OLA [6] is implemented based on the Spark platform, mainly oriented toward optimizing nested query estimation. G-OLA uses uniform random sampling technology for result estimation, utilizes nested query estimation results and confidence intervals to divide the final query result into certain and uncertain parts, thereby reducing data operations during updates. BlinkDB [7] is also implemented based on the Spark platform, analyzes query workloads, selects stratified sampling column sets based on query semantics, and uses stratified sampling technology to solve small data grouping problems, but does not address low selectivity issues. PF-OLA [8] proposes a framework for implementing online aggregation in distributed environments, including mechanisms for intra-node data storage, distributed data sampling, query processing, and result estimation, minimizing the impact of the distributed environment on online aggregation data streams.

Reference [9] proposes a dual-level sampling algorithm for raw data, sampling from both data blocks and within data blocks during data processing. This algorithm can adjust sample size in real-time according to computational resources, thereby reducing sampling costs. CrowdOLA [10] combines crowd-sourced entity resolution methods with online aggregation technology, solving the problem of low result estimation accuracy on duplicate data. Overall, these research efforts mainly focus on online aggregation implementation techniques for single-table queries.

The implementation mechanism for single-table query online aggregation cannot be directly applied to multi-table join query online aggregation because the results of joining random samples from each table are correlated, not completely random. Haas et al. [11] studied this problem and proposed the ripple join algorithm. Ripple join randomly samples alternately from each join table, placing sample data into memory. Whenever a new sample is read from one table, it is joined with all data already read from other tables. This process repeats until the estimation results meet user requirements. Since sample data is extracted from each table without considering data distribution or query workload information, ripple join produces estimation results very slowly when there are few results satisfying join predicates or many groups. To address the shortcomings of the basic ripple join algorithm, subsequent research in the relational database field emerged. Reference [12] parallelized the ripple join algorithm, but this method is not scalable—once data can no longer be loaded into memory, the estimation results lose statistical significance. Reference [13] applied sort-merge ideas to the ripple join algorithm, randomizing data swapped from memory to external storage to ensure statistical significance of estimation results, and implemented it in the DBO engine. Reference [14] proposed TurboDBO, which effectively utilizes intermediate results during query processing to further accelerate confidence interval convergence. COLA [15] implemented two-table join online aggregation based on the MapReduce framework, considering data storage characteristics in cloud computing platforms, using a data-block-based dual stratified sampling method for data sampling, parallelizing the ripple join algorithm, and designing MapReduce jobs for implementation. Overall, ripple join blindly randomly samples data from each join table, resulting in very low yield of join results and slow confidence interval convergence when join predicate selectivity is low or there are many join result groups. Reference [16] proposed the wander join algorithm, which performs random walks on join tables, uses indexes on join columns to determine walk directions during the walk process, and estimates results based on each walk. Wander join solves ripple join' s problem of low estimation result yield when join selectivity is low, but when there are many groups or data skew occurs, confidence interval convergence remains slow, and small-group estimation results may even be lost. This paper uses Markov chain to model the multi-table join process and performs stratified sampling based on grouping column sets, effectively addressing the problems of inaccurate result estimation and slow confidence interval convergence caused by join workload or data skew.

2 MC-OLA Overview

2.1 Problem Modeling

We use a natural join of four tables to illustrate the MC-OLA modeling process. Let the join query expression be:

$$\langle MATH_0 \rangle$$

In the above query expression, op is the specific aggregation operation, exp is the algebraic operation on tuples, and col is the grouping column set. Assuming the join order is R1-R2-R3-R4, the join process is converted into a Markov random process from R1 to R4, as shown in Figure 1 [Figure 1: see original paper]. Nodes in the figure represent tuples in each table, and if two nodes satisfy the join predicate, there is an edge between them. For example, there is an edge between t21 and t31, indicating that the condition t21.C=t31.D is satisfied. From t21, it is also possible to walk to t32 and t35, but the probability of choosing a walk direction is independent of the path before t21, thus satisfying the Markov property. A random path formed by starting from a tuple in R1 and walking to a tuple in R4 is a join result.

2.2 System Architecture

The MC-OLA system architecture consists of two phases: sample creation and online aggregation, as shown in Figure 2 [Figure 2: see original paper]. The sample creation phase creates stratified samples for the original dataset based on workload characteristics, with stratification based on the grouping column set in the query workload. The grouping column set selection uses the method proposed in reference [17] to maximize the probability of column sets appearing in the workload and the probability of grouping column sets in the workload being covered. Based on the determined grouping column set and index distribution, MC-OLA determines the join order of each table and then creates stratified samples at the walk start layer of the Markov chain. In the online aggregation phase, multi-table join query statements submitted by users are parsed, samples with the minimum query cost are dynamically selected for stratified sampling, and query results and confidence intervals are estimated.

3 Sampling Techniques

3.1 Determining Join Order

The problem modeling in Section 2.1 was introduced based on chain joins, but multi-table joins also include acyclic joins and cyclic joins. Using nodes to represent join tables and edges between nodes to represent join relationships, the join types for four tables are shown in Figure 3 [Figure 3: see original paper]. For a given multi-table join query, there are many possible join implementation orders, and different join orders have different impacts on sampling and

result estimation accuracy and convergence speed. Therefore, before creating stratified samples, MC-OLA first determines the join order based on workload characteristics and index distribution.

Taking the chain join in Figure 3(a) as an example, R1-R2-R3-R4 and R3-R4-R2-R1 are both reasonable join orders, while R3-R1-R2-R4 is not a correct join order. Lemma 1 provides the determination criteria for multi-table join order.

Lemma 1 Let there be m tables participating in the join in the query statement. The necessary and sufficient condition for the join order R1-R2-R3···Rm to be a reasonable join order is: for any table Ri in the join order, at least one of the tables preceding Ri has a direct join relationship with Ri.

Proof. We prove by mathematical induction.

a) When two tables R1 and R2 are joined, the join order includes R1-R2 or R2-R1, which obviously satisfies the condition.

b) Assume the proposition holds when k tables are joined.

Sufficiency. Let the join sequence of k tables be R1-R2-R3···Rk, satisfying the condition that "at least one of the tables preceding Ri has a direct join relationship with Ri." When adding a table Rk+1 to the join, place Rk+1 between Ri and Ri+1 in the original join sequence, and ensure that at least one table in R1-Ri has a direct join relationship with Rk+1. Then the join from R1 to Rk+1 can be completed, and the joined result can also complete the join with the sequence from Ri+1 to Rk, thus the join order is reasonable.

Necessity. Let the join order of k tables be R1-R2-R3···Rk, satisfying the condition that "at least one of the tables preceding Ri has a direct join relationship with Ri." When adding a table Rk+1 to the join, place Rk+1 between Ri and Ri+1 in the original join sequence, and the new sequence is a reasonable join sequence. Then there must be at least one table in the sequence from R1 to Ri that has a direct join relationship with Rk+1, so the new reasonable join sequence still satisfies the direct join relationship condition. Proof complete.

Based on Lemma 1, we present the join order determination algorithm. First, directions are added to the join graph according to index conditions: if there is a join edge between Ri and Rj, and Rj has an index on the join column, then the direction is added from Ri to Rj, and vice versa. Assuming the grouping column set belongs to table Ri, we then start from Ri to traverse the vertices of the directed graph to generate the join sequence, as shown in Algorithm 3.1. The algorithm produces a spanning tree of the join graph. For cyclic joins, the generated join sequence does not include all join relationships. After the walk is completed, the remaining join relationships can be used to further filter the walk results. For example, for the query in Figure 3(c), if the grouping column set is in table R3 and the generated join sequence is R3-R1-R2-R4, the join relationship R2-R3 can be used to filter the join results after the walk.

Algorithm 1 Join Order Determination Algorithm

Input: JoinArray[N][N]: two-dimensional array containing the directed join query graph;
Output: JoinList: join order list;
1: JoinSearch(int n, int m)
2: {
3: JoinList.add(n);
4: for(int i=0; i++; i<m)
...

3.2 Creating Stratified Sample Walk Start Layer

For single-table online aggregation, the estimation population is all tuples in the table, and stratified samples and layer sizes can be obtained through full table scans. Compared with single-table queries, the estimation population for multi-table joins is a subset of the join results or Cartesian product, making stratified sample creation much more complex. If the traditional ripple join algorithm is used, its population is a subset of the Cartesian product of all tables, and computational complexity grows exponentially with the number of tables. MC-OLA models the join process on a Markov chain, treating the population as random walk routes from start table tuples to end table tuples, and creates stratified samples by traversing the Markov chain.

MC-OLA places the table containing the grouping column set at the start end of the Markov chain random walk and creates the walk start layer of stratified samples based on the start table Rs. For single-table queries, the sample population is simply the original table data, so stratification can be performed directly based on the grouping column set, with the number of tuples in each sample layer being the layer size. In multi-table join online aggregation, the sample population is the multi-table join result, and stratified samples cannot be obtained by scanning any single table. For any tuple ti in Rs, MS-OLA performs a walk from ti based on the Markov chain, calculates the number of join results associated with this tuple, and further determines the start layer to which ti belongs.

Using the four-table chain join in Figure 1 as an example to illustrate the sample creation process, if the join order is R2-R1-R3-R4, then R2 is the start end for the walk, and stratified sample walk start layers are created. Assuming the walk start tuple is t21, when walking to tuple t11 in R1, it cannot continue forward, so it jumps back to t21 and continues walking in the R3 direction until reaching tuple t41 in R4. MC-OLA defines tables with degree 1 in the join graph as "edge tables," including tables like R1 that require direction jumps during the walk and tables like R4 that mark the end of the walk. Once an "edge table" is encountered during the walk, MC-OLA records the number of path branches in the current walk and changes the walk direction. The final number of join results is the product of the number of branches in each path. The specific implementation is shown in Algorithm 3.2. Given a walk start tuple t, the walk begins from all adjacent tables of t' s table (lines 4-15). If the adjacent table R'

has a degree less than 2 in the join graph, it means R' is an "edge table," and Algorithm getPathNum is called to calculate the number of branch paths (lines 8-9). Otherwise, it means R' can continue walking along the join sequence, and Algorithm getJoinSize is recursively called to obtain the number of join results (lines 10-13). Finally, the join results of each branch are multiplied to obtain the total number of join results starting from tuple t. The method for determining the number of walk branch paths is shown in Algorithm 3.3. Given the start tuple t of a branch path and the adjacent table R in the walk direction, tuples connected to t are obtained based on the index on R' s join column (line 4), and the number of tuples is accumulated to obtain the number of branch paths (lines 5-9).

Algorithm 2 Join Result Size Determination Algorithm

Input: tuple t: walk start tuple; R: table containing t; L: join order list;
Output: int joinSize: number of joined results with t as the walk start tuple;

```
1: int getJoinSize(tuple t)
2: {
3: int joinSize=1;
4: int tempSize;
5: for(int i=0; i++; i<join degree of R)
6: {
7: R' = adjacentTable(i);
8: ···
9: }
10: }
```

Algorithm 3 Walk Branch Path Number Determination Algorithm

Input: tuple t: walk start tuple; table R: the adjacent table according to the walk direction;
Output: int pathNum: number of walk paths with t as the walk start tuple;

```
1: int getPathNum(tuple t, table R)
2: {
3: int pathNum=0;
4: tuple t' = walkAccordingtoIndex(t,R);
5: while(t' )
6: {
7: pathNum++;
8: t' = walkAccordingtoIndex(t,R);
9: }
10: return pathNum;
11: }
```

4 Online Aggregation Implementation

4.1 Sampling Strategy

The mainstream algorithm currently implementing join online aggregation is ripple join, which directly performs uniform random sampling from each join table without considering the impact of data distribution on join results. Therefore, when join selectivity is low, it leads to low result yield and low estimation accuracy for small groups. MC-OLA uses Markov chain to model joins, transforming the join process into random walks across multiple tables and performing stratified sampling from the walk start data. Online aggregation updates results at a fixed frequency, and the sample size N extracted each time can be calculated based on the update frequency. MC-OLA does not use the original data of each table as the population but uses the join results as the population. Therefore, N represents the size of join results, i.e., the number of random walk paths. During sampling, N is allocated to each group' s sample layer. To minimize the variance of estimation results, MC-OLA' s sample size allocation algorithm references the class mean algorithm proposed in reference [18]. The sample size extracted from each stratum is the average value of N in L sample layers and the minimum value of remaining samples. If the total sample quantity is less than N, N is expanded to N' and the above process is repeated until N' that makes the total sampling quantity closest to N is found. Different from the algorithm in reference [18], the remaining value of each layer' s sample in MC-OLA refers to the number of join results, not the number of tuples in the walk start table.

4.2 Aggregation Result and Confidence Interval Estimation

After determining the sampling quantity for each layer, random walks begin from the sample start layer, with the number of walks being the allocated sample quantity for that layer. The query statement is based on the connection form of problem modeling in Section 2.1. Aggregation operations mainly discuss the implementation methods of SUM and COUNT, while other aggregation operations such as AVG, STD-DEV can be implemented through corresponding extensions. Using the chain join shown in Figure 1 as an example to introduce the walk method for connection results in each layer, the start data in Figure 1 is divided into three sample layers, and aggregation results and confidence intervals are estimated separately within each group. When performing random walks on sample layer S1, a start tuple is first randomly and equally probably selected from S1. Assuming t11 is selected, next, according to the index of R2 on the join key with R1, a tuple is randomly selected from the tuples adjacent to t11 in R2, and the walk continues down the Markov chain until reaching R4, finally extracting a path. Based on the join results extracted from multiple walks, aggregation results and confidence intervals are estimated. In this example, sample layer S1 contains 6 paths. If the extracted path is t11-t21-t32-t42, the probability of being extracted is 1/24, not 1/6. Using this method, each path has a different probability of being extracted, so the join result samples are not obtained through uniform random sampling.

Let the start layer samples be S1, S2, ···, Sm respectively. Given sample Si, the probability of each path being extracted is:

$$\langle MATH_1 \rangle$$

where B(t) represents the set of tuples in R2 that satisfy the join relationship with tuple t in R1. Let op( ) be the aggregation operation on the join result corresponding to path , and define the random variable X as:

$$\langle MATH_2 \rangle$$

The value of X is defined as:

$$\langle MATH_3 \rangle$$

If op is SUM operation, then:

$$\langle MATH_4 \rangle$$

If op is COUNT operation, then:

$$\langle MATH_5 \rangle$$

For a specific group to be estimated, let the sample size of this group be n. Theorem 1 provides the estimation method for aggregation results and confidence intervals.

Theorem 1 Let:

$$\langle MATH_6 \rangle$$

be the unbiased estimation of the multi-table join aggregation result. Let the confidence level of the estimation result be , then the confidence interval is:

$$\langle MATH_7 \rangle$$

Proof. The probability of path being extracted is:

$$\langle MATH_8 \rangle$$

Each path has a different extraction probability, belonging to independent biased sampling. According to the Horvitz-Thompson biased sampling estimation principle, we know that:

$$\langle MATH_9 \rangle$$

is an unbiased estimate of the population aggregation value [19], and since the n walk paths are independent, their mean remains an unbiased estimate of the population aggregation value. Construct the random variable:

$$\langle MATH_10 \rangle$$

Then the estimation of the join result aggregation value is transformed into the estimation of the population mean of the new variable. According to the Central Limit Theorem, the population mean approximately follows a normal distribution:

$$\langle MATH_11 \rangle$$

Standardizing this normal distribution yields:

$$\langle MATH_12 \rangle$$

Given confidence level , we obtain:

$$\langle MATH_13 \rangle$$

Using sample variance:

$$\langle MATH_14 \rangle$$

to replace population variance ^{2}, we can obtain:

$$\langle MATH_15 \rangle$$

Proof complete.

5 Experimental Results Analysis

5.1 Experimental Configuration

The experimental test platform is a cluster environment consisting of 11 nodes, connected via a 1 Gbit switch, with one node serving as the HDFS and MapReduce master node and the remaining 10 nodes as worker nodes. The software platform is HOP. Each node has 4GB memory and a 3.3GHz quad-core CPU, with 1TB disk space per node. The HDFS block size is set to 64M, with 2 map tasks and 1 reduce task configured on each node.

The experimental join query workload uses Q5, Q11, and Q21 from TPC-H. Q5 is a six-table acyclic join, Q11 is a three-table chain join, and Q21 is a four-table chain join. The involved tables include: customer, orders, lineitem, supplier, nation, and region. To focus on analyzing the performance of multi-table grouping join queries, the order by and having clauses were removed from the query statements during experiments. The TPC-H data generator was used to produce data, generating datasets of different sizes: 5 GB, 10 GB, 15 GB, and 20 GB based on the scale factor.

During experiments, the confidence level was set to 95%, and the aggregation result update interval was set to 2%, meaning aggregation results were updated whenever the query progress increased by 2%. The relative_error metric was used to measure estimation accuracy. In grouping aggregation queries, the relative error rate of each group' s estimation result was calculated and averaged across all data to measure the accuracy of the entire query statement. The calculation formula is:

$$\langle MATH_16 \rangle$$

The relative_interval metric was used to measure confidence interval convergence. Similar to relative error rate, the average relative confidence interval width across all groups was calculated using the formula:

$$\langle MATH_17 \rangle$$

To minimize experimental bias, all workloads were executed three times, and the average values were used for result analysis and presentation. Figures 4 [Figure 4: see original paper] and 5 [Figure 5: see original paper] show the average relative error rate and relative confidence interval width for the three queries on a 10GB dataset.

MC-OLA demonstrates significant performance advantages over random walk and ripple join. Taking the six-table join query Q5 as an example, MC-OLA' s relative error rate is already below 1% at around 25s, and the relative confidence interval width is already below 5%. Random walk and ripple join require approximately 95s and 265s respectively to achieve the same error rate and confidence interval width. Similar performance advantages of MC-OLA are observed on queries Q11 and Q21, outperforming random walk by about 3 times and ripple join by an order of magnitude. On one hand, both MC-OLA and random walk use indexes on join columns to perform random walks for multi-table joins, while ripple join performs uniform random sampling from multiple tables for joins, and the samples are not independent. Therefore, MC-OLA and random walk have lower execution complexity, stronger join "purpose," and higher join result yield than ripple join. On the other hand, random walk and ripple join perform uniform random sampling from the join start table and all join tables respectively, while MC-OLA uses stratified sampling technology and treats

each group' s join results as a separate population for estimation. Therefore, MC-OLA can obtain data for small groups early in query processing, thereby improving estimation accuracy and convergence speed for small groups.

Scalability testing was achieved by varying dataset size, using avgResTime to measure response speed. avgResTime represents the time when the relative error rate begins to be less than 1% and the relative confidence interval width begins to be less than 5%. Figure 6 [Figure 6: see original paper] shows the average response times of the three algorithms on datasets of 5 GB, 10 GB, 15 GB, and 20 GB. It can be seen that as the dataset size increases, MC-OLA' s average response time does not change significantly, while the average response times of random walk and ripple join increase slowly with dataset size. MC-OLA extracts samples from the sample start layer and then uses indexes on join columns to sample adjacent tuples for the next walk. The number of join results obtained after n walks is only related to the out-degree of nodes on the Markov chain, walk success rate, and the number of walks n, and is independent of the original dataset size. Therefore, the average response time is theoretically independent of data scale. Although random walk also implements joins through random walks, in grouping join query processing, small groups require relatively more sample data for estimation, and as the dataset size increases, the probability of extracting samples from small groups decreases, so the average response time increases with dataset size. Ripple join randomly extracts n samples from k join tables. Let the join degree between tables be d and each table size be N, then the number of join results obtained is:

$$\langle MATH_1 8 \rangle$$

Therefore, the average response time shows a trend of growing slower than linearly with dataset size.

6 Conclusion

Multi-table join query is one of the most important operations in big data analysis. However, the high implementation complexity of multi-table joins leads to long running times, affecting the timeliness and value of data analysis tasks. Online aggregation can provide statistically meaningful estimated results and confidence intervals before query completion, thus offering an effective way to reduce running time and computational cost. Existing multi-table join online aggregation algorithms are mainly based on ripple join, which performs uniform random sampling from each join table, implements joins between new and old data separately, and then performs result estimation. This algorithm has high complexity and results in extremely low result yield when join selectivity is low. Additionally, for grouping join queries, it leads to low estimation accuracy for small groups. This paper proposes MC-OLA, a multi-table join online aggregation technique based on Markov chain. Its main contributions include: proposing a system architecture for implementing multi-table join online

aggregation; using Markov chain to model multi-table joins, transforming the join process into a random walk process on the Markov chain; proposing a join order determination method and a stratified sampling algorithm for the walk start layer, along with corresponding sampling strategies and result estimation algorithms. Experimental results on the HOP platform show that MC-OLA' s average response time outperforms random walk and ripple join, and it has good data scalability. Currently, MC-OLA only supports multi-table join grouping queries. Future work will focus on nested queries, online optimization of multiple queries, and other aspects.

References

[1] TPC-H Bechmark [EB/OL]. [2018-07-20]. http://www. tpc. org/tpch/.
[2] Condie T, Conway N, Alvaro P, et al. Online aggregation and continuous query support in mapreduce [C]// Proc of ACM SIGMOD International Conference on Management of Data. New York: ACM Press, 2010: 1115-1125.
[3] Hellerstein J, Haas P, Wang H. Online aggregation [C]// Proc of ACM SIGMOD International Conference on Management of Data. New York: ACM Press, 1997: 171-182.
[4] Haas P. Large-Sample and deterministic confidence intervals for online aggregation [C]// Proc of the 9th Conference on Scientific and Statistical Database Management. 1997: 51-63.
[5] Pansare N, Borkar V, Jermaine C, et al. Online aggregation for large mapreduce jobs [J]. Proceedings of the VLDB Endowment, 2011, 4(11): 1135-1145.
[6] Kai Z, Sameer A, Ankur D, et al. G-OLA: generalized on-line aggregation for interactive analysis on big data [C]// Proc of ACM SIGMOD International Conference on Management of Data. New York: ACM Press, 2015: 913-918.
[7] Sameer A, Barzan M, Aurojit P, et al. BlinkDB: queries with bounded errors and bounded response times on very large data [C]// Proc of the 8th Eurosys Conference. New York: ACM Press, 2013: 29-42.
[8] Qin C, Florin R. PF-OLA: a high-performance framework for parallel online aggregation [J]. Distributed and Parallel Databases, 2014, 32(3): 337-375.
[9] Cheng Yu, Zhao Weijie, Florin R. Bi-Level Online Aggregation on Raw Data [C]// Proc of the 29th International Conference on Scientific and Statistical Database Management. New York: ACM Press, 2017: 10.
[10] Zhang Anzhen, Li Jianzhong, Gao Hong, et al. CrowdOLA: online aggregation on duplicate data powered by crowdsourcing [J]. Journal of Computer Science and Technology, 2018, 33(2): 366-379.
[11] Haas P, Hellerstein J. Ripple joins for online aggregation [C]// Proc of International Conference on Management of Data. New York: ACM Press, 1999: 287-298.
[12] Luo G, Curt J, Peter J, et al. A scalable hash ripple join algorithm [C]// Proc of ACM SIGMOD International Conference on Management of Data. New York: ACM Press, 2003: 252-262.
[13] Chris J, Subramanian A, Abhijit P, et al. Scalable approximate query processing with the DBO engine [C]// Proc of ACM SIGMOD International Con-

ference on Management of Data. New York: ACM Press, 2007: 725-736.
[14] Chris J, Subramanian A, Abhijit P, et al. Scalable approximate query processing with the DBO engine [J]. ACM Trans on Database Systems, 2008, 33(4): 23.
[15] Shi Yingjie, Meng Xiaofeng, Wang Fusheng, et al. You can stop early with COLA: Online processing of aggregate queries in the cloud [C]// Proc of the 21st ACM Conference on Information and Knowledge Management. New York: ACM Press, 2012: 1223-1232.
[16] Li Feifei, Wu Bin, Yi Ke, et al. Wander join and XDB: online aggregation via random walks [J]. SIGMOD Record, 2017, 46(1): 33-40.
[17] 史英杰, 杜方, 尤亚东. MSOLA: 基于多维分层采样的大数据在线聚集技术 [J]. 计算机应用研究, 2018, 35(2): 61-66. (Shi Yingjie, Du Fang, You Yadong. MSOLA: big data online aggregation based on multi-dimension stratified sampling [J]. Application Research of Computers, 2018, 35(2): 61-66.)
[18] Swarup A, Phillip B, Viswanath P. Congressional samples for approximate answering of group-by queries [C]// Proc of ACM SIGMOD International Conference on Management of Data. New York: ACM Press, 2000: 487-498.
[19] Horvitz D, Thompson, D. A generalization of sampling without replacement from a finite universe [J]. Journal of the American Statistical Association, 1952, 47(260): 663-685.

Note: Figure translations are in progress. See original paper for figures.