A Parallel Adaptive Partial Materialization Method of Data Cube Based on Genetic Algorithm

Ying Xia, Ting Ting Luo, Xu Zhang, Hae Young Bae
Research Center of Spatial Information System, Chongqing University of Posts&Telecommunications, Chongqing, China
xiaying@cqupt.edu.cn

Abstract. Due to the limitation of computing and storage resources, online analysis of massive data is usually time consuming. Data cube materialization is an effective way to improve the performance of online analysis. Considering the potential parallelism of genetic algorithm and its good global searching ability, a materialization method of data cube based on genetic algorithm is proposed. This method selects materialized views by combining the partial materialization strategy and MapReduce, while the materialized views can be adjusted adaptively according to the query log. Experimental results show that this method adapts to the big data computing environment and it can select reasonable materialized views to improve the query efficiency.

Keywords: data cube, partial materialization, genetic algorithms, MapReduce, adaptive adjustment

1 Introduction

With the rapid development of big data, many fields have urgent needs for online analytical processing of massive data. Data cube is a multidimensional data model, which is helpful for online analysis. It usually needs to be pre-computed and saved in disk in order to improve the efficiency of the queries. But materializing all data cube requires a large amount of storage space. Therefore, materialized view selection is a hot research area in data warehouse field. As the materialized view selection is a NP-hard problem [1] and genetic algorithm is suitable for solving NP-hard problem, so we transform the problem of materialized view selection to finding optimal solution with genetic algorithm, and introduce MapReduce to improve the performance. Furthermore, an adaptive update method of materialized views according to the query log is presented to optimize the materialized view dynamically.

2 Related Work

In recent years, many methods about materialized views selection have been put forward. [2] proposed a genetic algorithm based on weighted to solve the materialized views selection problem. [3] combined the ant colony algorithm and genetic

After selecting the materialized views, the views should be adjusted according to the query. [6] combined regular adjustment and query efficiency adjustment to adjust the data cube materialized views set. According to the query frequency, [7] proposed a real time adjustment strategy, but it may lead to materialized set frequent dither.

3 Parallel Selection Method of Materialized Views Based on Genetic Algorithm

3.1 Parallel Processing of Genetic Algorithm on MapReduce

Genetic algorithm generates new generation through evolution until the termination condition is reached. For parallel processing of genetic algorithm on MapReduce, each generation of evolutionary process completed within a Map/Reduce procedure. Map operation computes individual fitness value parallel. The key in the Map is the subgroup number and the value is the individual. Reduce operation merges the values of same key, and completes the selection, crossover and mutation operation of the subgroup.

3.2 Strategy of Materialized Views Selection Based on Genetic Algorithm

For the problem of materialized views selection on parallel framework, we assume that the user query task set is \( Q = \{q_1, q_2, ..., q_n\} \), the probability set corresponding to \( Q \) is \( \Phi(Q) = \{\phi(q_1), \phi(q_2), ..., \phi(q_n)\} \). \( M \) is the materialized set and \( M = \{m_1, m_2, ..., m_k\} \). \( S(m) \) is the storage space of \( m \), and \( C_{q_i}(M) \) is the query cost of \( q_i \) when there is a materialized set \( M \). Then the query cost and storage cost as shown in (1) and (2):

\[
\text{QueryCost}(Q, \Phi) = \sum_{q \in Q} \phi(q) \cdot C_{q}(M) \tag{1}
\]

\[
\text{StoreCost}(M) = \sum_{m \in M} S(m). \tag{2}
\]

Chromosome Encoding. There are several kinds of chromosome encoding method for genetic algorithm. For convenient processing of genetic algorithm, binary encoding is adopted to map materialized set to 0 and 1 list. Specific encoding steps are as follow: Firstly, the \( 2^n \) nodes of the n-dimensional data cube are numbered sequentially from 1 to \( 2^n \). Secondly, constructing one 0 and 1 list with length of \( 2^n \) (the 1 means that the node needs to be materialized while 0 is not).
Population Initialization. To initialize population, the size of the population SCALE should be determined first, which is the number of individuals in each population, and then we generate SCALE individuals randomly as the initial population.

Fitness Function. The genetic algorithm use fitness function to select the next generation. Therefore, the selection of fitness function is related to the convergence speed of the algorithm and the quality of the solution. One of the goals of data cube materialization is to reduce the query cost, but the storage overhead cannot be ignored. In order to relatively compromise query time and storage cost, here define two fitness functions respectively, and use these two fitness functions to select population.

\[
f_1(i) = \frac{1}{\text{QueryCost}(Q, \Phi)_{g=i}} .
\]

\[
f_2(i) = \frac{1}{\text{StoreCost}(M)_{g=i}} .
\]

Genetic Operation. Genetic operation includes selection, crossover and mutation. Selection operator is an evaluation method of individual adaptability. The fitness proportionate selection is used here, that is, the probability \( P \) of an individual being selected is in proportion to the value of expected fitness function.

\[
P(i) = c_1 \frac{f_1(i)}{\sum_{j=1}^{m} f_1(j)} + c_2 \frac{f_2(i)}{\sum_{j=1}^{m} f_2(j)} .
\]

When the genetic algorithm chooses the next generation of individuals, it gives weights \( c_1 \) and \( c_2 \) (which \( 0 < c_1, c_2 < 1 \), \( c_1 + c_2 = 1 \)) to \( P \) firstly. Crossover operation generates new offspring by swapping the partial gene segments of the two parent individuals. Mutation operation could expand the new search space, and the population diversity can be maintained through the variation of the local convergence of the population.

3.3 Parallelization of Materialization Views Selection Based on Genetic Algorithm

Using genetic algorithm to select materialized views, the main steps are as follows:

1. Randomly generating initial population.
2. Computing individual fitness.
3. Selecting the individuals with high fitness.
4. Performing crossover and mutation operations to generate new individuals.
5. Computing the fitness value of all the offspring.
6. Repeating 3-5 steps until the maximum number of iterations is reached.
4 The Adaptive Adjustment Algorithm of Materialized Views Set

The materialized views set should be properly adjusted according to the change of the query. As in a fixed statistical period, the query is not evenly distributed and usually leads the adjustment of the calculation to be very random. So we take a fixed number of queries as a statistical period.

Definition: Statistical period $T_n$ and the average query life period $E(T_n(q_i))[8]$.

$E(T_n(q_i))$ is the mathematical expectation of the query $q_i$ at the nth period $T_n$. If there exist the query task set Q, the probability set $\Phi(Q)$ corresponding to the Q, materialized set M, the whole query log set $L(Q)$ and the query log set $L(T_n)$ in $T_n$, we can update the mathematical expectation of $q_i$ according to the Formula (6):

$$E(T_n(q_i)) = \begin{cases} 
\alpha + (1 - \alpha)E(T_{n-1}(q_i)), & q_i \in L(T_n) \\
(1 - \alpha)E(T_{n-1}(q_i)), & q_i \notin L(T_n) 
\end{cases} \quad (6)$$

Here, $\alpha$ is the weighting coefficient.

The adaptive adjustment of the materialized views set mainly contains the following steps: Before adjusting, using the genetic algorithm select the initialization materialized set M. During the querying process, saving the number of the queries, the query statements and query logs into log $L(Q)$. Thus, we can get the query task set Q and query probabilities $\Phi(Q)$ from the $L(Q)$. Then, setting the eliminating threshold $T$ and the selecting threshold $S$. In each life period $T_n$, the adaptive adjustment algorithm is performed to adjust materialized views.

5 Experiment

In the experiment, we constructed a six-node cluster. The configuration of each node as follows: 1.8GHz CPU, 8G memory, 40G hard disk, CentOS6.5 operating system, Hadoop-1.2.1, Hive-0.10.0. The data is the Star Schema Benchmark (SSB) [9]. The SSB is widely used in academia and industry to evaluate the performance of decision support technologies. For comparison, we set the SSB data generation program SF(scale factor) to 10, 30 and 60, and get S1, S2 and S3 three data sets.

Experiment 1 compares the query response time of no materialization, random partial materialization and the proposed partial materialization method (in short, GAPWV). We select query Q1.1, Q2.1, Q3.1, Q4.1 of SSB as test cases, calculate the query response time of data set S1, and compare the results, as shown in Fig. 1.
Through the analysis, we can find that the query response time is significantly shorter after using GAPWV. This fully shows that the algorithm is effective.

In order to test performance of GAPMV when the data size increases, we test the data set S1, S2, S3 respectively ten times and calculate the average response time, the specific performance of the comparison as shown in Fig. 2.

With the growth of data set size, the query response time of no materialization and random partial materialization increased significantly. But the query response time of using GAPMV algorithm is relatively slow. That fully proves the algorithm is effective when the data size is increased.

In order to test the effect of the adaptive adjust algorithm, the number of queries is increased from 20 to 100 times to simulate the life period $T_n$. Fig. 3 shows the comparison of average response time using adaptive adjustment and no adjustment.
Fig. 3. Comparison of response time according to the number of query

Fig. 3 shows that with the increase of the number of queries, the query response time without adaptive adjustment is on the rise, and the query response time with adaptive adjustment is remained stable, indicating the materialized views to do good adaptive adjustment.

6 Conclusion

In order to improve the performance of online analytical processing, a parallel materialized views selection algorithm based on genetic algorithm is proposed, and an adaptive materialized views adjustment method is presented. Materialized views selection algorithm considers both the costs of query and storage, and apply MapReduce to improve computing performance. The adaptive adjustment method combines the query log and adjusts materialized views dynamically. The proposed methods adapt to the big data computing environment, and it can select reasonable materialized views to improve the query efficiency.

Acknowledgement. This work is funded by Chongqing Natural Science Foundation (cstc2014kjrc-qnrc40002), Scientific and Technological Research Program of Chongqing Municipal Education Commission (KJ1500431), and WenFeng Creative Foundation of CQUPT (WF2014-05).

References