The Selection and Placement Method of Materialized Views on Big Data Platform of Equipment Condition Assessment

Yan Ma¹, Jinxia Yao¹, Chao Gu¹, Yufeng Chen¹, Yi Yang¹ and Lida Zou²*

¹(State Grid Shandong Electric Power Research Institute
No. 2000 WangYue Road, Jinan, Shandong Province, 250002)
²(Shandong University of Finance and Economics
No. 7366 Bicyclic East Road, Jinan, Shandong Province, 250014)
yanpony@126.com

Abstract. With the formation of electric big data environment, more and more big data analyses emerge. In the complicated data analysis on equipment condition assessment, there exist many join operations, which are time-consuming. In order to save time, the approach of materialized view is usually used. It places part of common and critical join results on external storage and avoids the frequent join operation. In the paper we propose the methods of selecting and placing materialized views to reduce the query time of electric transmission and transformation equipment, and make the profits of service providers maximal. In selection method we design a computation way for the value of non-leaf node based on MVPP structure chart. In placement method we use relevance weights to place the selected materialized views, which help reduce the network transmission time. Our experiments show that the proposed selection and placement methods have a high throughput and good optimization ability of query time for electric transmission and transformation equipment.

1. Introduction
With the constant promotion and deepening of smart power grid construction, the data generated by grid operation and equipment monitoring are increasing exponentially. Power grid enterprises need to store and process the big data with an unprecedented magnitude and it is a huge challenge. The most important value of big data is to store and analyse the massive data. In business environment, data processing providers package their data processing into services and sell to users. With the commercial application of big data, its data processing shows the following characteristics.

1) The number of users on big data platform is huge. The big data service providers wish to serve more clients and gain more business profits.

2) Different clients have various performance requirements on big data processing. Some users have more requests on real-time analysis and some users have no special demands on query return time. For real-time data processing, users usually sign an agreement on return time with big data service providers. The earlier the return time is, the higher the profit of service provider is. On the contrary, if the return time is long, service providers gain low profits and even pay for some penalty cost.

3) Since big data platform needs to process massive analytic queries issued by multiple users, cloud computing platform is required to do various data and transaction processing [1].
Big data platform could serve multiple users through providing data analysis service and bring in profits for service providers. However, there exist many join operations in big data query, which is time-consuming. If some critical and frequently-used intermediate results are stored to external storage in advance, join operations could be avoided and return time largely reduces. We call the data of intermediate results as materialized view. Materialized views take up large storage spaces. Big data platform has large storage spaces, but it can not provide infinite spaces for materialized views. Therefore, we need to select some key and common views to materialize.

Meanwhile, on cloud computing platform with share-nothing structure, each computing node reserves certain storage space for materialized views, and some materialized views are placed on computing nodes. Different placement methods have different optimization performance. In the paper we also design a placement method of materialized views.

Big data service providers wish to achieve profit maximization. Our proposed selection and placement methods of materialized views on big data platform try to maximize the profits of service providers. The selection of materialized views is the basis and provides input for placement method. The placement of materialized views inputs the set of selected materialized views.

2. Related Work
Harinarayan et al. [2] first propose the issue how to select materialized views in 1996 and the greedy algorithm BPUS (Benefit Per Unit Space) becomes the classical selection method of materialized views. It is based on multi-dimensional data grid and the profit ratio between BPUS and the optimal solution is no less than 0.63. Shukla et al. present PBS (Pick By Size) algorithm [3] to solve the shortcoming of greedy algorithm. Greedy algorithm has a low computing efficiency and a large calculating quantity when the dimension is very high. Gupta et al. give GIA (Greedy Interchange Algorithm) algorithm [4] which is based on AND-OR graph. In 1997 J. Yang et al. use genetic algorithm to address the selection of materialized views [5]. According to Multiple View Processing Plan (MVPP), it first merges the public sub-views that could be shared to obtain the best MVPP, and then choose certain views to materialize from MVPP. After that genetic algorithm based selection method of materialized views becomes a popular research [6]. Similarly, Kirkpatrick et al. [7] propose a selection method based on simulated annealing algorithm, which is usually used in the issue of multiple dimensions and its performance relies on proper selection of parameters. Besides the above algorithms, there also exist the other algorithms such as DynaMat algorithm [8], Chunk algorithm [9] and Cache algorithm.

3. The Selection and Placement of Materialized Views
The selection and placement methods aiming for maximizing profits are closely related with profit function. We next give our profit function. Let $q_i$ denote a query, where $i$ is the subscript of the query. The profit function of a query is computed as

$$ R(t_i) = a_i - b_i \times t_i $$

In Eq. (1), $a_i$ is the initial profit, i.e., the profit instantly returned by $q_i$; $b_i$ is punishment slope, which denotes the descending rate of profits as time goes by; $t_i$ is the return time of $q_i$.

Figure 1 shows the flowchart of how to select and place materialized views.

3.1. Selecting Materialized Views
The selection method of materialized views is as follows.

1) Since big data queries change with time, the selection of materialized views is done periodically. First choose any a computing node on the clouds as the primary node and collect the set of big data queries in the last period.

2) Given a query set $Q$, use MVPP (Multi-View Processing Plan) algorithm [4] to generate MVPP structure chart. MVPP algorithm uses directed acyclic graph to describe the overall query strategy of query set.
In Figure 2 leaf node denotes the fact table in database, root node denotes the query based fact table and all the non-leaf nodes are the selection objects of materialized views. Taking Q1, Q2, Q3,Q4 and Q5 as an example, Figure 2 shows the generated MVPP structure chart. The query Q3 is obtained through joining Item, Sale and Part, i.e., (Item∞Sale)∞Part. When the number of queries is large, many query plans like this will be generated and non-leaf nodes increase quickly. Thus only materializing part of views is necessary.

Let $E$ denote the set of all the non-leaf nodes in MVPP structure chart. In MVPP, $e_j$ denotes a non-leaf node. Each non-leaf node $e_j$ corresponds to a candidate materialized view and we use $m_j$ to denote the candidate materialized view $e_j$ represents.

3) Compute the value of each non-leaf node in $E$.

The value of non-leaf node $e_j$ is denoted as $v_j$ and computed as:

**Figure 1.** The flowchart of selecting and placing materialized views
It is further deduced as
\[ v_j = \sum_{q \in Q_j} R(t_{ij}) - R(t_l) \] (2)
\[ v_j = \sum_{q \in Q_j} [a_i - b_i \times t_{ij} - (a_i - b_i \times t_l)] = \sum_{q \in Q_j} [b_i \times (t_l - t_{ij})] \] (3)
where \( t_{ij} \) is the return time of \( q_i \) when the candidate view \( m_j \) that \( e_j \) corresponds to is materialized, \( t_l \) is obtained through experimental test, and \( Q_j \) the query set that could reply on \( m_j \) to execute.

4) Last we use a greedy selection algorithm of materialized views aiming for maximize profits to obtain the set of materialized views. The proposed greedy selection algorithm is based on valuation of MVPP chart. Its input is the MVPP chart after computing the value of each non-leaf node and its output is a set of materialized views. The main steps are as follows:
   a) Let \( F \) denote a set of materialized views and initiate it as null.
   b) Compute and find the non-leaf node \( e_j \) with the largest value \( v_j / s_j \) in \( E \), where \( s_j \) is the occupied storage space of candidate materialized view that \( e_j \) corresponds to.
   c) Add the candidate materialized view that \( e_j \) corresponds to into \( F \), i.e., materialize \( m_j \). It is formalized as \( F \leftarrow F \cup m_j \), \( E \leftarrow E - e_j \).
   d) Update the value of the other non-leaf node in \( E \). After materializing \( m_j \), the return times of some queries change and the values of some non-leaf nodes also change. Next we give the concrete steps of updating operation.
      i. For each query \( q_i \) relying on \( m_j \) in query set \( Q_j \), its return time \( t_i \) is updated as the new return time when executing \( q_i \) based on \( m_j \).
      ii. Materialized \( m_j \) affects the values of all the descendant and ancestor non-leaf nodes of \( e_j \). According to Eq. (3), all the above values are re-computed.
   e) If the sum of storage spaces all the materialized views occupy in \( F \) is less than the total storage space (denoted as \( S \) ) system materialized views occupy, go to step b); else the algorithm ends, where \( F \) is the set of selected materialized views.

![Figure 2. MVPP structure chart](image-url)

3.2. Placing Materialized Views
When executing a query, two or more materialized views may be used. When these views are distributed on multiple nodes, there exist expensive data transmission cost and processing efficiency largely reduces. In the paper we present a placement method of materialized views based on relevance weight. The relevance weight is the frequency when two materialized views are concurrently accessed by a query.

Let \( N \) denote the set of computing nodes in clouds and \( |N| \) is the number of computing nodes. The storage space used for materialized views on node \( n_k \) is denoted as \( s_k \). The main steps of placing materialized views based on relevance weight are as follows.
1) Construct a matrix of relevance weights for materialized views, which is denoted as H. H has $|F|$ columns and $|F|$ rows, where $|F|$ is the number of placed materialized views. Let $h_{i,j}$ denote an element of H, i.e., $h_{i,j}$ is the relevance weight between materialized view $m_i$ and materialized view $m_j$.

2) Compute the value of each element in $H$. Taking $h_{i,j}$ as an example, its computed as

$$h_{i,j} = \sum_{q_k \in Q_{i,j}} b_k$$

where $b_k$ is the punishment slope of profit function of $q_k$ as shown in Eq. (1), $Q_{i,j}$ denotes the queries concurrently accessing $m_i$ and $m_j$ in $Q$.

3) Sort all the computing nodes in the descending order of their materialized storage spaces.

4) Select the node with the largest materialized space, $n_k$, in $N$, and obtain its placed materialized views through computation. Its computing method is as follows.

i. Choose the materialized view with the largest sum of rows from $|F|$ rows as initial clustering center.

ii. Place $m_i$ on the node $n_k$.

iii. Choose the materialized view, $m_j$, that having the largest relevance weight with $m_i$ in $F$ and place it on node node $n_k$, until the sum of storage spaces of the placed materialized views on $n_k$ is larger than $s_k$.

5) Update $N$ as $N \leftarrow N - n_k$

6) Go to step 4) until $N = \emptyset$.

4. Experimental Results

The types of queries on our big data platform of equipment condition assessment mainly include searching the relative data of a device, computing the corresponding relationship between PMS system and EMS system, computing arrester chargeability, computing capacity load rate of breaker and obtaining the relative reject rate and duration time, etc. In order to validate the performance of the proposed selection and placement algorithm, we choose the algorithms of no materialized view and random materialized view as comparison objects and use query time and throughput as comparison metrics. The algorithm of no materialized view does not materialize any view and random materialized view randomly selects several views to materialize.

We first choose one node as our placing node to validate the performance of the proposed selection method of materialized views. Figure 3 and Figure 4 show the comparison of the proposed selection method, no materialized view and random materialized view respectively on query time and throughput. From Figure 3 we see that with the increasing space of materialized views our selection algorithm has less query time, and performs 45.3% and 27.2% better than no materialized view and random materialized view. Similarly on throughput in Figure 4, our selection algorithm is 127% and 25.6% better than the other two algorithms.

![Figure 3. The query time on a node](image-url)
We next use 10 nodes on our big data platform to demonstrate the performance of both selection and placement methods. Figure 5 and Figure 6 gives the comparison on query time and throughput of three algorithms. Our proposed selection and placement algorithms also perform well on the big data platform. The query time and throughput have advantages with the increasing assigned space of materialized views. For query time, our proposed algorithms are 41.8% and 20.4% better than no materialized view and random materialized view. For throughput, our algorithms are 115% and 15.7% better than the other two algorithms.

Figure 4. The throughput on a node

Figure 5. The query time on big data platform

Figure 6. The throughput on big data platform
5. Conclusions
In the paper we propose the methods of choosing materialized views on big data platform and placing materialized views on the clouds to maximize the profits of big data service providers. Meanwhile, the placement method of materialized views could reduce the data transmission between nodes and decrease processing time. Through our extensive experiments we demonstrate that our proposed selection and placement methods perform much better than no materialized view and random materialized view on query time and throughput.

Reference
[1] Zou L D, Qing-Zhong L I, Kong L J, et al. Indexing Mechanism of Multi-Tenant Database Based on Chord [J]. Chinese Journal of Computers, 2016.
[2] Harinarayan V, Rajaraman A, Ullman J. Implementing data cubes efficiently [J]. in the Proceeding of SIGMOD, ACM Press, 1996: 205-216.
[3] Shukla A, Deshpande P M, Naughton J F. Materialized View Selection for Multidimensional Datasets[C]. Proceedings of the 24th VLDB Conference, New York, USA: ACM Press, 1998: 448-459.
[4] Gupta H, Harinarayan V, Rajaraman A. Index Selection for OLAP[C]// Gray A, Larson, Per-Ake. ICDE'97, Proceedings of the 13th International Conference on Data Engineering. Birmingham, UK: IEEE Computer Society Press, 1997: 189-255.
[5] J. Yang, K. Karlapalem, and Q. Li, "Algorithms for materialized view design in data warehousing environment," in VLDB, 1997, pp. 136-145.
[6] Hong J T, Chang Y J, Liu B J, Applying evolution algorithms to materialized view selection in a data Warehouse [J]. Soft Computing, 2003, 7(23): 574-581.
[7] Derakhshan R, Dehne F, Korn O, Stantic B. Simulated annealing for materialized view selection in data warehouse environments[M]/Hamza M H. Proc. of the 24th IASTED Int’l Conf. on Database and Applications. Innsbruck: IASTED/ACTA Press, 2006: 8-94.
[8] KotidisY, ROUSS0poulos N. DynaMat: A dynamic view management system for data warehouses. Proc. Of the 1999 ACM SIGMOD Int, 1 Coof. On Management of Data (SIGMOD, 99). NewYork:ACMPress,1999.371－382.
[9] Deshpande P, Ramaswamy K, Shukla A, Naughton J. Caching multidimensional queries using chunks. Proc. Of the ACM SIGMOD Int, 1 Conf. on Management of Data (SIGMOD, 98). Seattle: ACM press, 1998. 259-270.