A dynamic materialized view Selection in a Cloud-based Data Warehouse

Yang Kehua, Abdoullahi Diasse
Hunan University
College of computer science and communication
Changsha 410082 Hunan, China

Abstract:

A data warehouse is system which that support decision-making in production environment .Classical data warehouse management system are often optimized by improving query performance. Such improvement is casually achieved by using caches, indexes and materialized views and required selecting the best set of data structures (materialized views, indexes, etc...). In the cloud based data warehouse performing such selection become more challenging due to the complexity of a cloud computing environment .In this paper we propose a materialized view selection algorithm based on monetary which take in consideration the computation cost, storage cost and transfer cost. By using query prediction our algorithm can perform in a dynamic manner to select the best set of materialized view.

Keywords: *Cloud computing, Materialized views selection, data warehouse, OLAP, query optimization*

Introduction:

Cloud Computing is one of the most emerging technological model that supports business intelligence through fast analysis of terabytes of data in a large and very complex distributed environment. It helps to deal with huge among of data by offering On-demand Self-services.

To implement and maintain a successful decision-making system a Data Warehouse (DW) needs to be build. DW can bring together selected sources from multiple database or other resources into a single repository. Offering Data Warehouse services through cloud is very challenging due to the big among of data that we need to deal with，that involve joins between relations and aggregations of tuples. Materialized views are able to provide better performance for DW queries. However in a cloud environment these views have computing cost, storage cost and also transfer cost.

We can see in one side of the spectrum user with a big constraint in term of financial budget may accept a long query response time but in the other side user with strong budget may disregard cost and ask for a very fast response time.

Materialized view can be build to support this goal and the challenge here lies on selecting a set of views that can be materialized in the cloud to improve the query processing respond time and the cost in cloud.

Three scenarios can be considered based on the same idea in [10]:

Budget limited: Given a predefined financial budget, the challenge here lies on selecting a proper dataset views that can be materialized in the cloud to minimize the query respond time

Respond-Time limited: With limited respond time, here the challenge is to select a set of materialized view which can minimize the financial cost

Tradeoff between the budget and the time: In this scenario we have to deal with the tradeoff between the query response time and the financial budget. The objective is to select a set of views to materialize in the cloud that gives an optimal solution in both query response time and financial budget limit.

In the cloud, Data warehouse can be geographical distributed and can lead to serious performance and storage problem. User can pose queries in different data marts at different

Part	Customer	Supplier	Region	Nation	Sum
steel chiffon pink puff	Customer#002	Supplier#1191	America	Brazil	38
floral sienna bisque	Customer#006	Supplier#2150	Europe	France	37
olivegoldenrod smoke	Customer#007	Supplier#3074	Europe	Germany	28
aquamarine seashell p	Customer#001	Supplier#1534	America	Argentina	24

Table1: Sales Dataset Excerpt

times instead of a single data repository. Those data marts can also be located in different data centers in the cloud. Regarding to such issues the selection of view to materialize in the cloud has two main challenges .First select the view to materialize in a dynamic manner when queries are constantly posed in different data center at different time. Thus the materialized view selection algorithm should take in consideration the new coming views and be able to extract the optimal set to materialize in the cloud. This can also lead to the dematerialization process of some views in the cloud and store the most beneficial views .Second if the nature of queries changes over the time and result to degradation of performance, a new set of materialized views is required to regenerate.

In other words we need to constantly monitors the incoming queries and materialize the best set of views by taking in consideration the pricing constraints (storage cost, computation cost and transfer cost)

Our approach for the materialized view selection in a cloud based data warehouse is a two-phase operation: *Startup* and the *Online* phase

In the *Startup phase* which is based on an initial query stream, the system starts by selecting an initial set of views by taking in consideration the resource constraint (budget, response time). In the *Online phase* an "in use" system will be considered and an existing set of views *V* has already been selecting and materialized

However the materialized view is expensive in term of computation and storage(resource constraint) ,so materialized all possible views in relatively impossible .Thus the key challenge in the online phase becomes selecting a new set of view *V'* by discarding some views from V given a new query workload based on some query prediction.

I. Preliminary knowledge

1. Materialized view :

Materialized views have been used for a long time in both OLTP and OLAP system for performance improvement. While views virtually store a query result and can help to solve complex query, materialized views store it physically in a table so that no need to recompute a query when its corresponding view is already materialized at any time the same query is entered in the system. Thus materialized view further improves query response time which is crucial in big OLAP system.

2. Running example:

To illustrate our work we use TCP-H benchmark models. Parts are bought form the suppliers and sold to customers. Each month the business needs to analyze the total profit per part, per suppliers, per and customer, per nation and per region. The *Table 1* provides an excerpt of this dataset. Thus there are fives dimension we are interesting: Part, Suppliers, Nation, Region and Customer.

II. Cost model for view selection in a cloud environment:

The cost model is an important issue in a view selection problem, it help to achieve a multi criterion optimization (CPU utilization, bandwidth consumption and disk allocation) of materialized view under the budget constraint .In this section we introduce the cost model for view materialization in a cloud environment and it relies on the one introduced in [6]. In [6] it assumes that all queries are executed on a constant number of identical instances IC_0 $(IC_j = IC_0), \forall j = 1..n_{ic}$. Here we suppose that queries are executed on instances configuration IC with a number of instances equal to n_{ic} then $IC = \{IC_j\}_{j=1..n_{ic}}$.

1. Computation Cost

The computation cost $C_c(Q, V, IC)$ of answering a set of queries $Q = \{Q_i\}_{i=1..n_q}$ by using a set of view $V = \{V_1, \dots V_{n_V}\}$ in an instance configuration IC can be expressed by the following formula:

$$C_c(Q, V, IC) = C_{proc}(Q, V, IC) + C_{maint}(V, IC) + C_{mat}(V, IC)$$

The processing cost $C_{proc}(Q, V, IC)$ is defined by the query workload Q, the set of materialized view V, the rented cloud instances configurations and the queries frequency and it can be expressed as follow:

$$C_{proc}(Q, V, IC) = \sum_i^{n_Q} \sum_j^{n_{ic}} T_{proc}(Q_i, V, IC_j) \times c_c(IC_J) \times f(Q_i, IC_J)$$

Materialize a view require executing its associate query .In cloud such operation must be paid .Let $T_{mat}(V_i, IC_j)$ the materialization time of view V_i on a cloud instance IC_j ,Then the materialization cost can be defined as follow

$$C_{mat}(V, IC) = \sum_i^{nv} \sum_j^{n_{ic}} T_{mat}(V_i, IC_j) \times c_c(IC_J)$$

If a considerable modification is done at the source dataset an update of already materialized view set is required. The update or maintenance cost is proportional to time required for updating materialized view and the materialized views update frequency. Let the maintenance time and the update frequency of V_i be $T_{maint}(V_i, IC_j)$ and $g(V_i, IC_J)$ respectively on a cloud instance IC_j then the maintenance cost can be defined as follows

$$C_{main}(V, IC) = \sum_i^{nv} \sum_j^{n_{ic}} T_{main}(V_i, IC_j) \times c_c(IC_J) \times g(V_i, IC_J)$$

2. Data storage cost:

Storage cost is an important issue especially in a cloud based architecture where any stored data need to be paid with respect to the storage time. Using materialized view to improve queries performance implies to store selected view in the cloud and pay the corresponding price. The data storage cost is proportional to the size of data (Initial dataset (DS) and Materialized view) and the storage time and can be expressed as follow:

$$C_{storage} = \sum_{interval} c_s\big(s(DS) + s(V)\big) \times (T_{start} - T_{end}) \times \big(s(DS) + s(V)\big)$$

$s()$ return the size in GB of any parameters and c_s is the CSP storage cost function.

3. Data transfer cost

The data transfer cost $C_{transfer}$ or bandwidth consumption cost depends on the size of the inputted data (the queries workload$\{Q_i\}_{i=1..n_Q}$), outputted data (the queries result $\{R_i\}_{i=1..n_Q}$ by exploiting the set of materialized view $V = \{V_1, \dots V_{n_V}\}$), the whole dataset (initial dataset and inserted data) and the CSP's atomic transfer cost c_t

$$C_{transfer} = \Big(\sum_{i=1}^{n_Q} (s(Q_i) + s(R_i)) + (s(DT) + s(inserted\ Data))\Big) \times c_t$$

III. Materialized View Selection

In this section we propose our approach for materialized view selection. Our algorithm is based on PR_Q system predictor [12] and which can operate as a complete view management system. It predicts the next query and materializes the view for it if necessary. Our algorithm for materialized view selection is described in the figure 1 and as mentioned earlier it has two phases: Startup and Online phases.

Algorithm for materialized view selection

Input:

1. *Q :A query stream form the user*
2. *BL, the budget limit*
3. *Recurrent_pattern_th: A specified threshold to determine a pattern is recurrent*

```
    4.  THQ: A specified threshold to extract recurrent pattern of queries
Materialized_view_selection_cloud()
1: $V_{can}$2_Phase_Optimization_approach(Q,THQ);
2:     If $(Cost\ (Q,\ V_{can})\ > BL)$ Then
3:      $V_{final}$= Knapsack0/1($V_{can}$, BL, $C_C$, $C_S$, $C_t$);
4:       Else
5:           $V_{final}$= $V_{can}$ ;
6: While ($Q_i$ is entering){
7:   Answer ($Q_i$, $Vfinal$);
10:  Pattern_extraction(Q,Recurrent_pattern_th);
11:  $Q_p$ =Predict_next_query( recurrent_patterns);
12:  If(($v_p \notin V$)and
13:   $(Cost\ (Q_p, v_p)\ + Cost\ (Q,V)\ \leq\ BL)$)
14:    $Vfinal = Vfinal\ \cup\ \{v_p\}$ ;
15:      Else
16:    If ($v_i \notin Vfinal$){
17:      While($Cost\ (Q_p, v_p)\ + Cost\ (Q,V) > BL$)
18:            Select $v_{LRU} \in Vfinal$;
19:      $V = Vfinal - v_{LRU}$;
20:      }
21:     $Vfinal = Vfinal\ \cup\ \{\ v_i\}$ ;
22: }
```

1. *Startup phase:*

In the Startup phase (line 1 to 5) the system begun with an inputted query stream, the threshold of the queries stream (THQ) is specified by the database administrator. We assume that we have a substantial knowledge of the incoming query stream. A good knowledge of the query stream will allow us to make all necessary estimation we need for our algorithm. Indeed $T_{proc}(Q_i, V, IC_j)$, $T_{mat}(V_i, IC_j)$ and $T_{mat}(V_i, IC_j)$ all need to be estimated upstream (from experiments or statistical models).the startup phase in turn operate in two successive steps: ***Two-Phase Optimization approach*** and the ***knapsack 0/1 optimization.***

1.1. Two-Phase Optimization approach

Firstly it performs a static view selection based on ***Two-Phase Optimization approach (2PO)*** [7]. In [7] the ***2PO*** approach uses MVPP (Multiple View Processing Plan) [8] to express the relationship between views. It combines two optimization algorithms proposed in more previous works which is the ***Iterative Improvement (II)*** [19] and the ***Simulated Annealing (SA)***[13].It was shown that 2PO perform better than most materialized view selection algorithm compared to [8] and [9] in a Data warehouse environment.

The ***(2PO)*** has the queries stream and the specified threshold *THQ* as parameters. The output of this first step is Vcan which denotes a set of candidate views. In our case we have made some modification in the ***(2PO)***. Instead of using MVPP to express the relationship between views we decide to use data cube lattice introduced by Harinarayan and al in [15].

1.2. knapsack 0/1 optimization

Because the view selection problem in cloud based architecture is essentially an optimization problem in term of monetary cost, we found necessary to integrate in our algorithm an optimization process that take in consideration the financial budget of the user (*BL* variable in our algorithm). In our case we focus only on a budget limited scenarios, where a predefined financial budget is given so that the challenge becomes on selecting a set of views that can be materialized in the cloud to minimize the query respond time under the budget constraint. To solve this problem a dynamic programming approach have been opted calls knapsack 0/1[11] problem .As input we provide necessary elements of our cost models, the result is V_{final}= Knapsack0/1(V_{can}, BL, F_{C_C}, F_{C_S}, F_{C_T}) where F_{C_C}, F_{C_S}, F_{C_T} are functions representing parameters in our cost model.The knapsack 0/1 problem can be represented as follow:

$$Min\left(T_{proc}(V) = \sum_{i=1}^{nq}\left(t_i - \sum_{k=1}^{n_v} g_{ik} * x_{ik}\right)\right)$$

$$C_{proc} + C_{maint} + C_{mat} + C_S + C_t \leq \text{BL}$$

Where g_{ik} represent the gain in respond time by exploiting the view V_k .We use x_{ik} as decision variables: $x_{ik} = 1$ if the query Q_i exploit the view V_k and $x_{ik} = 0$ if not. t_i

denotes the query Q_i respond time without exploiting any materialized view.

The idea is to find a set of materialized view which minimize the total query respond time by maximize the gain when materialized view exploiting. So our knapsack 0/1 can be reformulated as follow:

$$Maximize\left(\sum_{i=1}^{nv} g_i * x_i\right)$$

The output of the knapsack 0/1 problem V_{final} which is used as input in the online phase if a new query is entered in the system.

2. *Online Phase*

The Online phase (from line 6 to 22) is the run-time phase of the system .During the Online phase the system materializes predicted views and moreover the system should identify insignificant old materialized view to be removed.

The Online view selection problem can be defined as follows: Given an already materialized view set $V = \{V_1, \ldots V_{n_V}\}$ and a previous query workload$\{q_i\}_{i=1..n_Q}$, find a V' (adding or discarding views form V) based on prediction of future query workload with respect to the resource constraints.

The prediction of future data query workload is done using the same technique as PR_Q system predictor which uses association rules mining and probabilistic reasoning approach.

While queries arrive this part is executed and the first operation of the online phase is *Answer(Q_i,$Vfinal$)* at line 7.

This procedure answers the incoming query by using the set of final materialized view $Vfinal$. If query definition exists in the materialized view pool the query Q_i is answered by using only one Materialized view v_i in $Vfinal$. In case of the query definition doesn't exist in materialized view pool, lets $A(v_i)$ be the smallest ancestor of v_i in the lattice [15]. If $A(v_i) \in Vfinal$ then Q_i is answer through $A(v_i)$, otherwise the query is answered by using the fact and dimension tables.

The *Pattern_extraction (figure 2)* procedure is the next operation of the online phase. This procedure extracts the recurrent patterns by giving in parameters a query stream and the recurrent pattern threshold and also by calculating the conditional probability of their last query occurrence (LQCP). To compute such a probability we apply the following conditional probability formula:

$$P(q_n = a_n | \, q_1 = a_1, q_2 = a_2, \ldots, \; q_{n-1} = a_{n-1}) = P(q_1 = a_1, q_2 = a_2, \ldots, \; q_n = a_n) \,/\, P(q_1 = a_1, q_2 = a_2, \ldots, \; q_{n-1} = a_{n-1}) = \prod(q_1 = a_1, q_2 = a_2, \ldots, \; q_n = a_n) \,/\, \prod(q_1 = a_1, q_2 = a_2, \ldots, \; q_{n-1} = a_{n-1}).$$

In this formula the current query stream is $Q_{n-1} = \{q_1, q_2, \ldots, q_{n-1}\}$, and then we use the formula to calculate the probability of entering q_n when Q_{n-1} is entering previously.

Algorithm for recurrent pattern extraction

Input: *Q, Recurrent_pattern_threshold*
Output: *recurrent_patterns, Recurrent patterns with conditional probability in their last query.*
Pseudo Code:

```
1:  Pattern_extraction () {
2:  recurrent_patterns=∅;
3:  n=2;
4:  While (n_ary patterns are recurrent){
5:      recurrent_patterns= recurrent_patterns U
6:      recurrent n_ary pattern with the conditional
7:       probability of their last query;
8:       n++;
9:       }
10: }
```

Figure 2: Pattern_extraction

The *Predict_next_query (figure_3)* has the recurrent patterns outputted by the *Pattern_extraction* procedure as parameter. This procedure predict the next query by finding the last query with maximum conditional probability from the recurrent patterns with length between 2 and the maximum length of recurrent patterns from the *Pattern_extraction* procedure.

```
Algorithm for next query prediction
Input: recurrent_patterns
Output: predicted_query
Pseudo Code:
1:  Predict_next_query () {
2:    Max_len_pattern: Maximum length of patterns
        in recurrent_patterns;
3:    n=2;
4:    While (n<= Max_len_pattern) {
5:        Find the n_ary recurrent pattern with
          maximum Conditional probability in their
          last query;
6:        n++;
7:    }
8:  predicted_query: last query with maximum
    conditional probability from founded recurrent
    patterns
}
```

Figure3: Predict_next_query()

Example:

We assume the data base administrator assigns 2 and 20 *Recurrent_pattern_threshold* and *THQ* respectively and Q is the query stream.

Q:{ $q_{11}q_5q_9q_7q_8q_{11}q_{12}q_5q_9q_{11}q_5q_9q_7q_{12}q_4q_5$ $q_9q_{11}q_3q_7q_8q_{11}q_5q_9q_7q_8q_{12}q_4$}

Q is at fist used in the startup phase and allow us to get our $Vfinal$. Now we suppose that q_7 entering in the system q_7 is now answered by using *Answer (q_7,$Vfinal$).* The next step is to call *Pattern_extraction* (Q, *2)* procedure .The table 2 show the different step of the *Pattern_extraction procedure*.

Recurrent_pattern _th(N)	**N-ary Pattern**	**LQCP**
N=2 Recurrent Binary Patterns	$q_{11}q_5$	3/5
	q_5q_9	1
	q_7q_8	2/3
	$q_{12}q_4$	2/3
	q_9q_{11}	1/2
	q_8q_{11}	2/3
N=3 Recurrent Triplicate Patterns	$q_{11}q_5q_9$	2/3
	$q_5q_9q_{11}$	1/2
	$q_7q_8q_{11}$	2/3
	$q_5q_9q_7$	3/5
N=4 Recurrent quadruple Patterns	$q_{11}q_5q_9q_7$	1
	$q_5q_9q_7q_8$	3/4
N=5 Recurrent quintuple Patterns	$q_{11}q_5q_9q_7q_8$	2/3

Now Predict_next_query () is call to predict the next query and our query stream becomes $Q+\{q_7\}$. This procedure starts by extracting the maximum length of recurrent extracted patterns. In our case this length is 5 which represent the length of the pattern $q_{11}q_5q_9q_7q_8$ .The idea here is to search below recurrent patterns (with length between 2 and 5) and find that one with maximum probability in their last query:
$\{q_7 *\}$; $\{q_5q_9q_7 *\}$; $\{q_{11}q_5q_9q_7 *\}$ with $(*)$ means the predicted query.
It appears clear that $q_5q_9q_7q_8$ has the biggest probability in its last query which is 3/4 thus q_8 is chosen as predicted query.

If the corresponding view (which we can call V_8 for simplicity) of q_8 was not materialized and we do not exceed our budget we just materialized V_8 . If V_8 was not materialized and we have already reached our budget limit we have to keep dematerializing least recently used views until materializing V_8 can be possible .

If q_8 is entered as the next query, q_8 will be answered by using V_8 through *Answer* procedure and next the query is predicted and materialized if necessary .the same process is repeated while new queries are entered in the system.

The run time performance of the system is improved by replacing old non-used materialized views with new views when a considerable change on the nature of queries stream is noticed.

IV. *Experimental Studies:*

To evaluate our work a number of experimentation have been conducted .In this section we describe our experimental setup and the result that we have obtained from our experiments. As we mention earlier we are interested only in the budget limit case where the challenge rely on selecting the best set of materialized view under a budget constraint.

1. *Experiment setup:*

In the cloud side our experiment have been conducted in a virtual cluster composed of four virtual machines running Ubuntu 10.0.4 with 8GB disk, 512 MB of RAM and 1vCPU.All nodes are feature Hadoop (Version 1.2.1)[1] and Hive (Version 0.11.0)[2].TCP-H [3] is used as dataset (scale factor 1) and all query are written in HiveQL[4] within the MapReduce[5,19] framework.

On the client side all algorithms are implemented on PC core i5-2410M 2.3GHz, with 4GB of RAM

2. Experimental results:

In order illustrate our work we experiment the budget limit use case where we need the select the best set of materialized views in order to minimize the response time under a budget constraint. We exploited the user total cost and the total query respond time for a period of 12 months with S3 [21] and EC2 [20] small instance pricing. To simulate the dynamic fashion of OLAP system, a random query generator has been use for our experimentations.

Firstly it randomly generates a query stream with 20 in length, which is going to be used in the static phase to get the first set of materialized view. In the dynamic phase it generates a number 30 of queries, it give 60 % of chance to select the predicted query, 30% of chance to select pick a query from the current query stream and 10% of chance to generate a not yet entered query.

Our results are plotted in the figure 4.It clear depicts the cost and total response time with and without materialized view for a period of 12 months. A significant improvement is showed in both cases. For example the total with and without materialized view are 6.38 USD and 11.92 USD respectively for a period of 6 months .Also a considerable improvement is noticed in the query processing time, indeed for the same period the processing time with and without materialized view are 11.13 hours and 36.88 hours respectively . Those results show that materialized views in the cloud are considerably desirable.

V. Conclusion:

In this paper we propose an approach to dynamically select materialized view in a cloud based data warehouse.

Our approach is based on PR_Q system predictor to predict the upcoming query and materialize its corresponding view by using conditional probability.

In contrast to most [3, 5, 8, 12] materialized view selection algorithm which use a cost model based on the size of tuples in the view, here we use a cost model based on monetary cloud pricing.

The results of our experimentations show that our approach perform with a considerable improvement both in term of monetary cost as well as in term of processing time.

VI. References:

[1] Hadoop. http://hadoop.apache.org/, February 2014

[2] Hive:http://hive.apache.org/, February 2014

[3] TCP-H. www.tpc.org/tpch, February 2014

[4] HiveQL, http://hive.apache.org/, February 2014

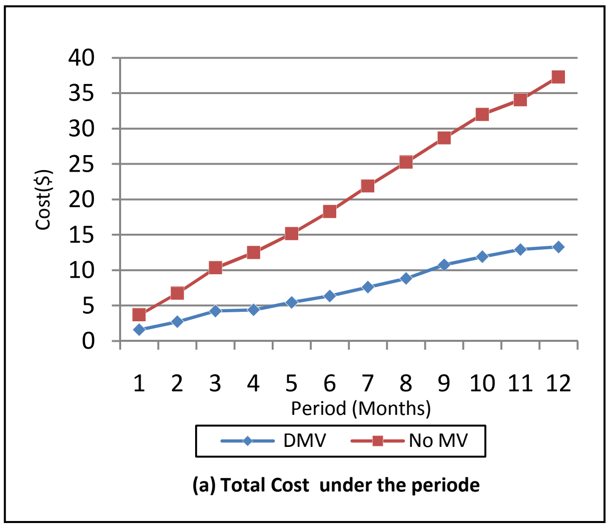
[5] Soumya Sen, Debabrata Datta, Nabendu Chaki "An Architecture to Maintain Materialized View in Cloud Computing Environment" *2012 International Conference on Computing Sciences*

[6] Thi-Van-Anh Nguyen, Laurent d'Orazio, Sandro Bimonte, Jérôme Darmont "Cost Models for View Materialization in the Cloud", *DanaC 2012, March 30, 2012, Berlin, Germany*

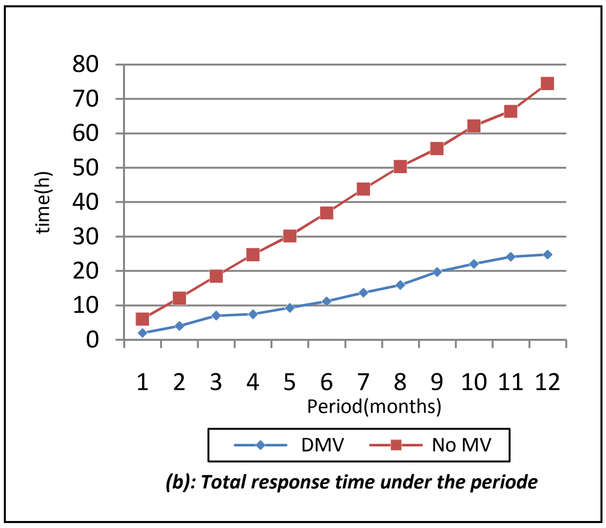
[7] Jiratta Phuboon-ob, and Raweewan Auepanwiriyakul,"Two-Phase Optimization for Selecting Materialized Views in a Data Warehouse",*World Academy of Science ,Engineering and technology 1 2007*

[8] J.Yang, K. Karlapalem, and Q. Li ,"Algorithms for materialized view Design in Data Warehousing Environement ",*VLDB conference ,1997,136-145*

[9] R. Derakhshan, F. Dehne, O. Korn and B. Stantic,"Stimulated annealing for materialized view Selection in Data Warehousing Environment"



(a) Total Cost under the periode



(b): Total response time under the periode

Figure 4: Result of Experimentations

[10] H. Kllapi, E. Sitaridi, M. M. Tsangaris, and Y. E. Ioannidis. " Schedule optimization for data processing on the cloud". *In Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 289{300, Athens, Greece, 2011}

[11] Knapsack Problem 0/1 : V .Chvatal .Hard Knapsack problems.Operations Reseach 28:1402-1411,1980.

[12] Negin Daneshpour, Ahmad Abdollahzadeh Barfourosh "Dynamic view Management System for Query Prediction to view materialization". *International journal of Data Warehouse and Data Mining*, 7(2), 67-96,April-June 2011

[13] T.V. Vijay Kumar and Santosh Kumar BDA 2012, LNCS 7678, pp. 168–179, 2012. .Materialized View Selection Using Simulated Annealing.

[14] Panos Kalnis, Nikos, Mamoulis, Dimitris Papadias. View Selection Using Randomized Search

[15] V.Harinaynan, A Rajaraman and J.D Ullman," Implementing data cubes efficiently", *in proc SIGMOD* ' 96 pp.205-216,ACM 1996.

[16] Michael Lawrence and Andrew Rau-Chaplin .Dynamic View Selecting for OLAP. *DaWak* 2006,LNCS 4081 ,pp. 33-44,2006

[17] Roozbeh Derkhshan , Frank dehne ,Othmar korn ,Bela stantic **"**Simulated annealing for materialized view selection in data warehouse environment

[18] J. Dean and S. Ghemawat. Mapreduce: simplied data processing on large clusters. *Communications of the* ACM, 51(1):107{113, 2008.

[19] T.V. Vijay Kumar and Santosh Kumar **"**Materialized view using simulated annealing" *Advances in Computing & Inf. Technology*, AISC 178, pp. 205–213. ,2013

[20] Amazon EC2 http://aws.amazon.com/ec2/, February 2014.

[21] Amazon S3 http://aws.amazon.com/s3/, February 2014.

Acknowledgment:

This work is supported by program for the growth of young teachers of Hunan University.