Temporary Materialized Views in Cloud Data Warehouses through a Web Service

Ettaoufik Abdelaziz
ESTC, CED Engineering Sciences, ENSEM, Hassan II University
Morocco
aettaoufik@gmail.com
Ouzzif Mohammed
ESTC Hassan II University
Morocco
ouzzif@est-uh2c.ac.ma

ABSTRACT: *Cloud Computing provides a flexible environment for customers to host and process their information through an outsourced infrastructure. This information was habitually located on local servers. Many applications dealing with massive data is routed to the cloud. Data Warehouse (DW) also benefits from this new paradigm to provide analytical data online and in real time. DW in the Cloud benefited of its advantages such flexibility, availability, adaptability, scalability, virtualization, etc. Improving the DW performance in the cloud requires the optimization of data processing time. The classical optimization techniques (indexing, materialized views and partitioning) are still essential for DW in the cloud. However, the DW is partitioned before being distributed across multiple servers (nodes) in the Cloud. When query containing multiple joins or ask voluminous data stored on multiple nodes, inter-node communication increases and consequently the DW performance degrades. In this paper, we propose an approach for improving the performance of DW in the cloud. Our approach is based on selection of temporary materialized views through a web service. For this purpose we use an algorithm allows to identify the queries list sent to the DW, and adds a materialized view for each new costly frequent query. This technique is based on managing temporary materialized views in order to optimize the frequent queries load by respecting the total cost. An experimental study on a cloud DW is carried out and a comparative tests show the satisfaction of our approach.*

Keywords: Cloud Computing, Data Warehouse, Performance, Materialized Views, Queries Processing

Received: 2 May 2017, Revised 14 June 2017, Accepted 20 June 2017

1. Introduction

Nowadays, data Warehouses occupy a central place in enterprise's business intelligence. They have been proposed to store heterogeneous and voluminous data [1]-[2]-[3]. They are powered from different data sources across transactional queries and propose analytical data through decision-support queries. DW is often asked by multiple users simultaneously through analytical

queries. Generally, the analytical queries execution time on large tables is very high what can degrade the performance of DW. On the other hand, having high traffic has also an impact on queries response time. Cloud Computing offers relevant solution to this kind of problems. It is a new paradigm that offers a flexible environment to host data in an outsourced infrastructure. In cloud, data are highly available; they are stored in different nodes. Optimizing the processing time and response time requires the optimization of both inter-node communication and computation time.

Several works treats data security stored in the cloud. The Confidentiality and availability of data are also handled [4]-[5]. Few works deal with data warehouse queries optimization hosted in the cloud. Previous work [6]-[7]-[8] seek to improve the query execution cost in DW hosted on the cloud. In [6] the authors proposed a cost model to select a schema of materialized views. The authors have treated the cost optimization of data warehouses queries in the cloud. From their part, the authors in [7] deal with queries optimization in cloud, they proposed a cost model to select materialized views in the Cloud. The main characteristic of the proposed cost model is that it considers the payment cost and the query processing paradigm. On the other hand, in [3] the authors presented a new mechanism for processing transactional queries. The model proposed is based on encrypting a DW and show performance results of this DW implementation. This is why we feel it is important to propose a query optimization approach in DW located in the cloud.

In this article we present a state of the art of data storage in the cloud. In section 3 we present different DW optimization techniques in the cloud. We illustrate our approach in section 4. We finish with a conclusion and perspectives.

2. State of the Art and Related Work

2.1. Cloud Computing

Cloud Computing is a new paradigm that refers to the use of computing resources on demand through a global network [7]. Cloud computing may be defined as a pay-per-use model for enabling on-demand access to reliable and configurable resources that can be quickly provisioned and released with minimal management. However, customers only pay for the resources they use [8].

1) Essential Characteristics of Cloud

• **On-demand Self-Service:** Customer may order, at any time, a service via the web. He can also quickly set up IT resources offered by the service provider without the intervention of the supplier.

• **Broad network access:** The client can access to the resources disposed on the network through any platform (mobile phone, tablet, laptop or workstation)

• **Resource Pooling:** The supplier of cloud service manages all the physical resources of his Cloud, it ensures virtualization, storage, security and distribution of resources among all its customers. The client has no control over the exact location of the provided resources but may know the location at a higher abstraction level (country, state, or datacenter).

• **Rapid Elasticity:** The customer can quickly increase or decrease the capacity of processing and storage. These capabilities seem to be unlimited and are available at any time.

2) Set of Services: The architecture of a cloud computing environment is generally based on a layered organization.

• **IaaS (Infrastructure as a Service):** Users can install and run their own applications in virtual machines proposed in cloud. The user is responsible for the installation, configuration and maintenance of the application. among the examples of cloud that provides infrastructure as a service, we quote: Amazon Elastic Compute Cloud (EC2) and Cisco Vblock;

• **PaaS (Platform as a Service):** The cloud provides a platform with multiple applications, such as operating system, database, web servers and mail servers. Other than the applications proposed by the cloud provider, the user can install other applications. Microsoft Windows Azure is an example of a cloud that provides a platform as a service.

• **Saas (Software as a Service):** Cloud that offers to multiple users the ability to use some ready-to-use applications. The access to these applications is usually multi-tenant and different users use the same installation of the application without sharing their data with others.

3) Cloud deployment:

• **Public:** The public cloud offers computing resources in a self-service mode. The use of cloud resources is willing to pay, isusually billed on the CPU, network and storage used. Amazon EC2 and Windows Azure are examples of public clouds.

• **Private:** The private cloud is an implementation of cloud computing on private networks. This implementation provides a virtualization solution for securing data hosted in the cloud

• **Community:** Several organizations with similar requirements (security requirements, policy, and compliance considerations, etc) can share an infrastructure cloud. They are hosted internally or externally and the costs are spread over the consumers.

• **Hybrid:** The cloud infrastructure is a composition of two or more distinct cloud infrastructures (private, community or public). An organization provides and manages internal resources and others are provided outside.

2.2. Data Warehouse on Cloud Environment

Several research studies have dealt with the integration of DW into cloud [4]-[5]-[10]-[12]-[13]-[14]-[3]-[20]-[21]. In [12] the authors proposed an approach for modeling ETL data management by MapReduce model. It is a question of resuming the Vassiliadis approach well known in this field and extending it to take care of specific aspects of a MapReduce. However, in [11] the authors illuminated the use of data warehouses and online analysis, they treated storage environments, users and security. To do this, they implemented a data warehouse under Hadoop and Hive and they used the Map and Reduce functions of this environment, then they retrieved the cost of loading the warehoused data and constructing OLAP cubes between a virtual and a physical cluster. Other research addresses the privacy and security of data storage in the cloud [4]-[5]. In [4] the authors treated the protection of the data warehouses stored in the cloud. The proposed approach is based on shared secret key Shamir. They showed that reliance on vendors is elaborate to construct with the usual structure of the cloud founded on a single provider. This structure threatens the confidentiality of client data on the grounds that they're hosted via a single provider of external hazard function [19]. In [5] and [14] the authors proposed an approach based on fragmentation in order to improve the confidentiality of the stored data warehouse.

3. Optimization Techniques of Queries in the Cloud

This section presents the optimization techniques used to improve the performance of DW hosted in Cloud Computing. Several previous works have dealt with this subject. In [8] and [9] the authors have proposed a cost models related to materialized views in a DW arranged in the cloud [8]-[7]. The main objective of these works is to optimize the response time of queries in the cloud. Swati et al. [6] have addressed the optimization of queries execution cost in the DW hosted in the cloud, for doing that they offered the storage structure PK-map and algorithm processing queries. In [15] the authors proposed an approach of the queries continuous optimization, they presented new technique for adapting query processing in the calculation process in the online environment from Microsoft. However, Leonidas et al. [16] proposed a query optimization framework in MapReduce paradigm, this approach is based on a novel of the query algebra and uses a small number of physical operators of higher order that are directly achievable on MapReduce systems such as Hadoop.

3.1 Materialized Views Selection Problem

Materialized views represent an optimization technique used to improve the performance in both OLTP and OLAP system. However, in the cloud environment, materialized views reduce de the cost of processing queries [7]-[8]. It compute and store a query result physically in a table only once time so it optimize all same query querying the DW. On other hand, materialized views occupy an additional memory space which has an influence on the storage cost and it requires a periodically processing which has an influence on the processing cost. For this purpose several works has proposed a cost model for view materialization in cloud [6]-[7].

3.2 Cost model for views materialization in cloud

The problem of materializing of set of views consists in identifying a set of views that optimize the execution time of queries load and that the total cost does not exceed the limit budget. However a cost model is used in order to guarantee a best optimization by respecting the budget constraint. In cloud environment, the total cost can be defined as follows:

Ctot = SC + PC + TC where

SC : total storage cost

SC = D*Cs

D : data stored in the cloud provider

Cs : storage cost

PC : total queries processing cost

PC = T * Ct

T : query processing time

Ct : query processing cost

TC : total network transfer cost

Ctot= Trr* Ctr

Trr : Size of both the request and the result

Ctr : Network transfer cost

The cost related to all materialized views is includes in total cost and it includes the following sub-costs:

- Storage cost
- Query processing cost
- Network transfer cost
- Maintenance cost

1) Storage Cost

The storage cost of all *VMi* is proportional to its size and storage period in the service provider and the price per unit of storage time [6]. In [6], the authors define the storage cost for *MVi* for a period *P* by the following formula:

$$C_{storage}(MVi) = \text{size}(MV_i) * price_{storage} * P \quad (1)$$

On other hand, A. Brighen et al. [6] proposed a storage cost of *MV* by taking into account the evolution of the size of *MV*. They supposed that the size of *MVi* is increases by an average ϕ_i each time unit and they estimated the storage cost for a period *P* by the following formula:

$$C_{storage}(MVi) = ((2\times\text{size}(MV_i)+\phi_i \times (P\text{-}1))*P/2)\times price_{storage} \quad (2)$$

The storage cost related to set of *N* materialized views is presented by the following formula:

$$TC_{storage}(MV) = \sum_{i=1}^{N} C_{storage}(MV_i) \quad (3)$$

2) Query Processing Cost

The processing cost represents the cost for executing a query Q_j in the presence of materialized view MV_i. The total processing cost is related to the set of queries *Q*, to the set of materialized views *VM*, to the rented cloud instances configurations *IC* and to the queries frequency *F* [17]. It can be presented by the following formula:

$$C_{processing} = \sum_{i=1}^{Nq} \sum_{j=1}^{Nic} T_{processing}(Q_i, VM, IC_j) \times C_{computation}(IC_j) \times F(Q_i, IC_j) \quad (4)$$

3) Maintenance Cost

The maintenance cost for a view VM_i is equal to cost of re-executing the query *Qi* related to *VMi*. However, the total cost for maintaining a set of materialized views *VM* is related to the set of queries *Q*, to the rented cloud instances configurations *IC* and to the updating frequency *F*. It can be presented by the following formula:

$$C_{maintenance} = \sum_{i=1}^{Nq} \sum_{j=1}^{Nic} T_{execution}(Q_i, IC_j) \times (Q_i, IC_j) \quad (5)$$

4) Network Transfer Cost

Network transfer cost represents the data transfer cost. This cost depends on the unit transfer cost C_{unit}, the size of the inputted data and the size of outputted data. For a set of N materialized views VM and set of queries Q, the transfer cost can be presented by the following formula:

$$C_{transfer} = \sum_{i=1}^{Nq} (Size_{qi} + Size_{VMi}) \times C_{unit} \quad (6)$$

4. Proposed Approach

We propose an approach for optimizing queries in a cloud computing environment through a web service. However, in the cloud environment, data is stored on different physical systems. Other than data processing time, the request execution requires an additional inter-nodes communication time. The objective of our approach is the reduction of inter-node communication during the execution of a query by adding temporary materialized views. The goal is detecting all new frequent query having a high cost then adding a materialized view optimizing the NFQ detected. The number of materialized views generated is controlled by the total cost. In this effect the step of queries classification is indispensable. This classification allows the web service to generate a set of materialized views optimizing set of NFQ. On other hand, the web service deletes all materialized view optimizing a non-frequent query (NonFQ). A NonFQ is a query that does not remain frequent and it is belonging to the old load of frequent queries. The task of deleting allows to optimize the storage cost and release the space for generating other temporary materialized views.

To this purpose, we propose an algorithm of queries classification (Figure 1). From the list of queries as input, this algorithm sends out the list of queries classified in order to propose a set of views to materialize.

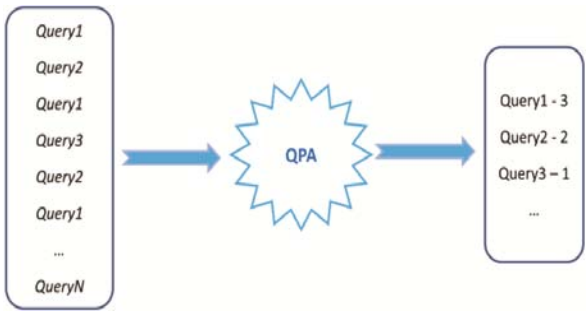

Figure 1. Query classification processing

The web service uses a module for queries management. This module is responsible for processing all requests querying the DW. This treatment takes place in the following steps:

- Detect all NFQ
- For each NFQ, determine the selection attributes list and the selection predicates list.
- Classifying queries according to selection attributes and selection predicates (matrix queries-predicates).
- Check to the materialized view selection cost
- Create new materialized view
- Establish the new frequent queries list;
- Compare the new list with the frequent queries list saved;
- Determine the list of NonFQ;
- Delete the materialized view associate to NonFQ

To compare the new list with the frequent queries list saved, the web service uses a table dedicated to saving the information

of the frequent queries load. This information will be used later to update the queries load by comparing this queries list to the current list of frequent queries. In our approach, two queries are considered identical if they use the same selection attributes with the same predicates.

Consider the following three queries Q1, Q2 and Q3:

Request Q1

```
SELECT MAX(Prix)
FROM Product P, Purchase A, Suppliers S
WHERE P.IdP = A.IdP AND P.IdF = S.IdF
      AND P.NomProduct =’P5'
      AND F.Ville = ‘Casablanca’;
```

Request Q2

```
SELECT MAX(Prix)
FROM Product P, Purchase A, Suppliers S
WHERE P.IdP = A.IdP AND P.IdF = S.IdF
      AND P.NomProduit =’P4'
      AND F.Ville = ‘Rabat’;
```

Request Q3

```
SELECT AVG(Prix)
FROM Product P, Purchase A, Suppliers S
WHERE P.IdP = A.IdP AND P.IdF = S.IdF
      AND P.NomProduit=’P5'
      AND F.Ville = ‘Casablanca’;
```

The two queries Q1 and Q3 use the same selection attributes with the same predicates. They are considered identical even if the two queries use two different aggregation functions. On the other hand, the query Q2 uses other selection predicates, it is different from the two queries Q1 and Q3. The table 1 shows an example of saving data for the three queries Q1, Q2, and Q3.

Request	**Attribute**	**Predicate**
Q1	NomProduit	P5
Q1	Ville	Casablanca
Q2	NomProduit	P4
Q2	Ville	Rabat
Q3	NomProduit	P5
Q3	Ville	Casablanca

Table 1. Queries Data Saving Table

From Table 1 data, the web service generates the domain of each selection attribute illustrated in Table 2 and create a matrix request-predicate (table 3) used to classify the set of NFQ.

When the web service detects a NFQ, it updates the queries information table. This update allows to add a new selection attributes or define a new extension of old attributes in order to trigger the optimization task.

Ville	NomProduit
Casablanca	P4
Rabat	P5

Table 2. Attributes Domains

	Casablanca	Rabat	P4	P5
Q1	1			1
Q2		1	1	
Q3	1			1

Table 2. Matrix request - predicate

The NFQ is a detected request that does not belong to the current list of frequent queries. It determines either new selection attributes or domain extensions of the old attributes.

The NonFQ is a query that does not remain frequent and it is belonging to the old load of frequent queries. For this purpose, this query must be present in the query table filled in by the web service since it participated in the current optimization process.

For determining the selection attributes list and the selection predicates list, each query is processed as a character string. Any selection attribute "AtS" of any NFQ must be presented in one of the following forms: "WHERE AtS", "HAVING AtS", "AND AtS", and "OR AtS". The selection attributes and the join attributes are distinguished by the fact that the join attributes are compared to other join attributes, while the selection attributes are compared by values. These values represent the selection predicates. The selection criterion is written as follows: "AtS opr PrS", where AtS represents a selection attribute, "PrS" represents a selection predicate and "opr" designates one of the following comparison operators {=, <,>, <>, ≤, ≥,IN, NOT IN}.

Algorithm 1: Temporary materialized views processing algorithm

```
input :
      Q : NFQ (New Frequent Query)
      FQL : Frequent Queries Load
      AS : Available space
      BL : Budget Limit
      Vs: set of temporary materialized views
output :
      Set of materialized views
Begin
      V: new materialized view
      If size(V) <AS and Total Cost d" BL Then
            Create V= {Creating materialized view}
            Declare V={ declaring in queries optimizer }
            Add V to set of materialized views
            Object = V
Else
            QCP (Queries Classification Processing)
```

```
            Delete VM associated to query Q that does not remain frequent
            Create V= {Creating materialized view}
            Add V to set of materialized views
            Object = V
      End If
End
```

Figure 2 illustrates the process of generating a set of materialized views from a queries load and metadata.

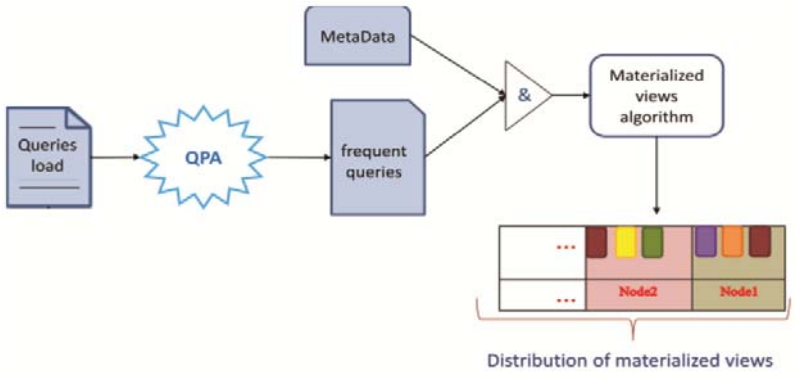

Figure 2. Queries processing

5. Test and Results

We performed tests on TPC-H benchmark [18] loaded on virtual clusters. The TPC-H benchmark is a decision support benchmark. It consists of a suite of business oriented adhoc queries and concurrent data modifications.

To do some tests we store dataset and materialized views and queries processing in the cloud. The views selection processing lives in the application server. We prepared a test environment under format of a network composed with three virtual ubunto machines with a 50 GB disk, 2 GB of RAM, and we install Oracle 11g on each machine. Then we generated 10GB from TPC-H and we distributed it to three nodes. In order to validate our approach, we use 5 queries for the performance analysis then we execute it in the TPC-H star schema. We consider that all query arrived in cloud represents a NFQ. For this effect the web service generates a materialized view optimizing the NFQ detected. In first time, we retrieve the execution time of queries without adding materialized views and after materialization of views. Whereas in second time we retrieve the total cost in both two cases.

1) Budget Limit

The objective is to add a set of materialized views *VM* that minimizes query processing time and the budget does not exceeds de budget limit.

$$\begin{cases} - Time_{execution} = \text{Minimize the execution time} \\ - Cost_{tot} \leq \text{Budget limit} \end{cases}$$

• Variation of query

In this test we execute different queries and we retrieve the execution time for each query before adding materialized views and after adding it.

• Variation of number of frequent queries

In this test we execute some queries at time and we compute the execution time for each set of queries before adding materialized views and after adding it.

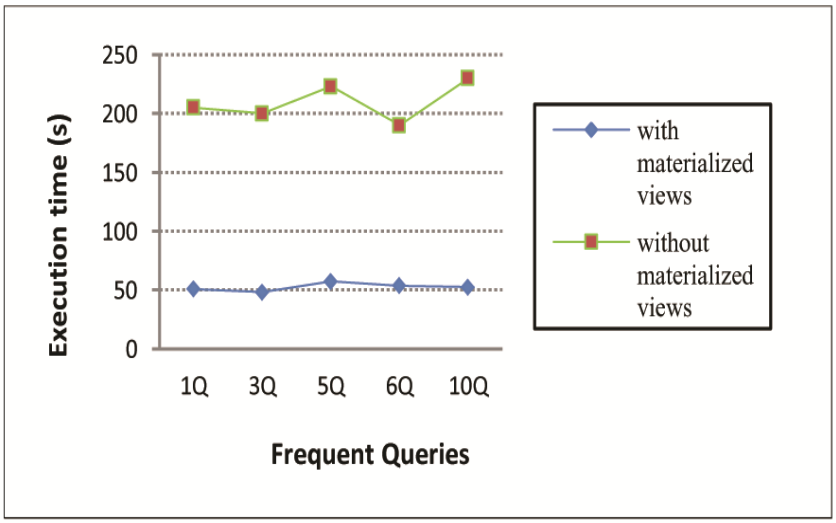

Figure 3. Execution Cost of new Queries

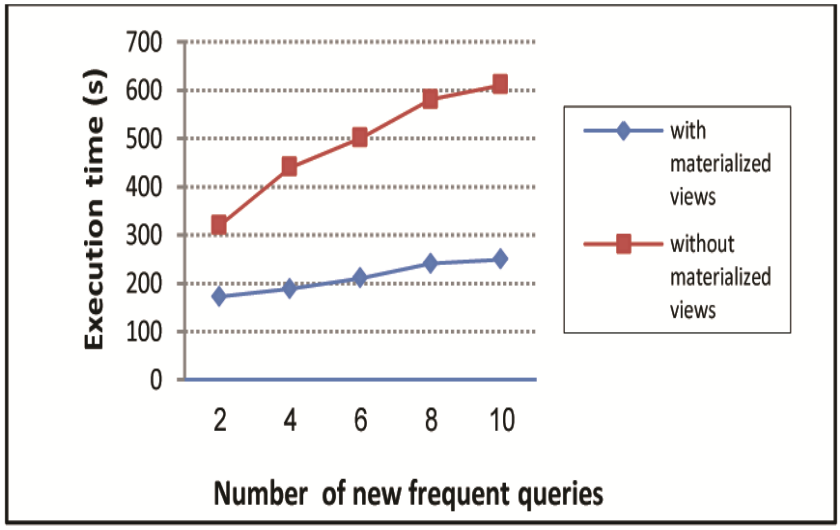

Figure 4. Execution cost for different number of new frequent queries

The results illustrated in figures 3 and 4 show the impact of materialized views on DW performance in cloud environment. Figure. 3 shows the execution time of NFQ before and after adding materialized views. Thus, Figure.4 shows the execution time of different number of queries in both two cases.

We notice that the materialized views offer a best optimization in cloud environment. This is caused by the fact that the web service generates a materialized view for each NFQ. The materialized view generated optimizes the NFQ associated to it because accessing to materialize view is less costly than accessing the data distributed in different nodes. Since materialized views generate few tuples than tables. On the other hand, executing a query in DW hosted in cloud requires a computing time and an inter-nodes communication time.

2) Time Limit

The objective is to add a set of materialized views VM that minimizes the total cost and the execution time does not exceeds de time limit.

$$\begin{cases} - Time_{execution} \leq \text{Time limit} \\ - Cost_{tot} = \text{Minimize the total cost} \end{cases}$$

In order to retrieve the total cost with and without materialized views, we execute different queries and we compute the total cost in both cases. To do this, we use "*Unit*" as currency symbol.

• Variation of query

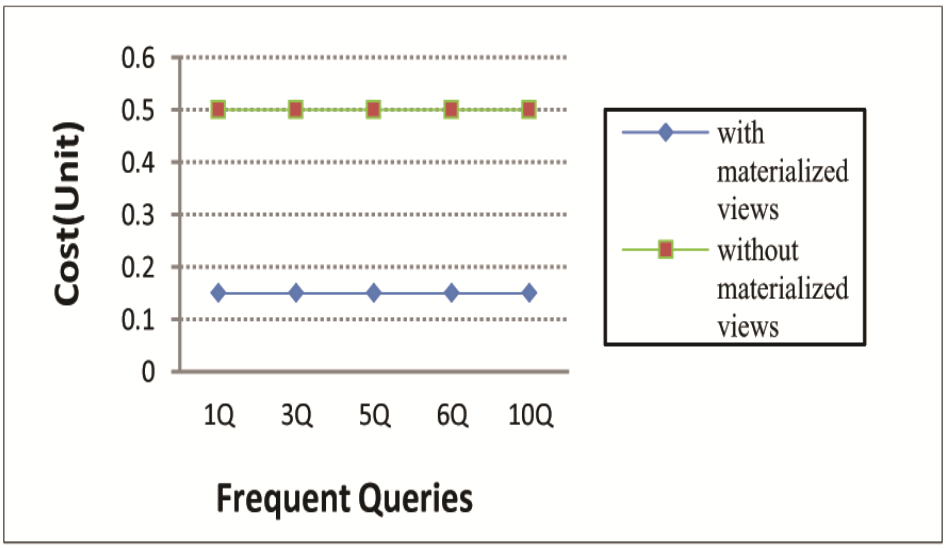

Figure 5. Execution Cost of new Queries

• Variation of number of new frequent queries

In this test we execute some queries at time and we compute the execution time for each set of queries before and after adding materialized views.

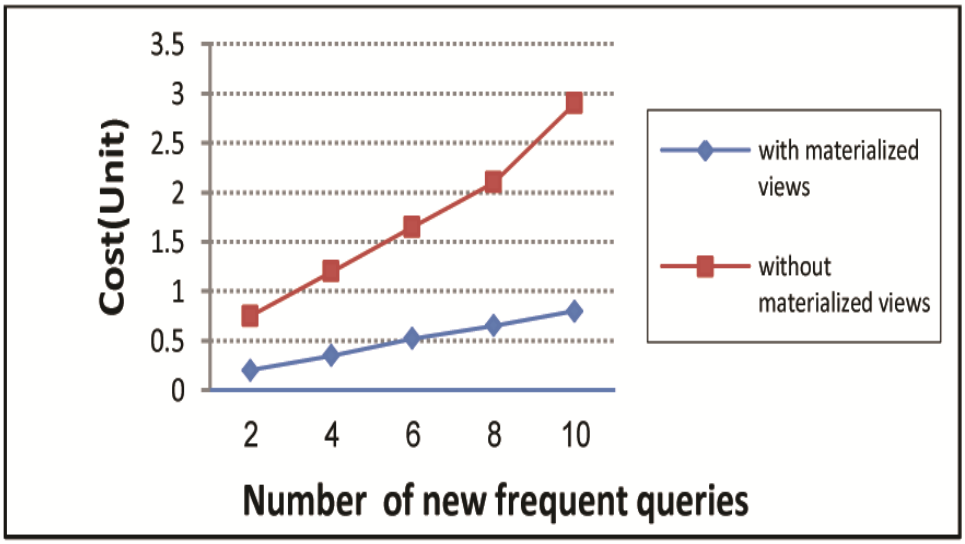

Figure 6. Execution cost for different number of new frequent queries

The results illustrated in figures 5 and 6 show the impact of materialized views on DW performance in cloud environment. Figure. 5 shows the total cost of NFQ after and before adding materialized views. Thus, Figure.6 shows the total cost of different number of queries in both two cases.

It clearly that the materialized view offer a best cost for executing a query in DW hosted in cloud environment compared to total cost for executing the some query without materialized view. This is caused by the fact that accessing to materialized view is less costly than accessing the data distributed in different nodes. However, the materialized views reduce the total transfer cost and the query processing cost because materialized views generate few tuples than tables. On the other hand, materialized views must be refreshed when data source are updated, which induces some maintenance overhead.

6. Conclusion and Perspectives

Query optimization in data warehouses has been studied extensively by industry and academic researchers. The storage space and processing times are indicators of performance of data warehouse, they represent the research focus of several previous studies. With the advent of cloud computing, data storage capacity has become very high. Improving the performance of DW in cloud is an area of research that has attracted many researchers. We proposed an approach based on adding temporary materialized views in cloud environment to improve the performance of DW. The materialized views processing is managed by a web service. This letter allows detecting all requests querying DW and optimizes it by adding a materialized view. The materialized view added is temporary because the web service deletes all materialized view none associated at any frequent query. On the other hand, although our approach allows improving DW performance on cloud environment in reasonable time, it increases the availability of DW since it will be manageable through the web. We plan to study the possibility to automate the optimization of a varied queries load by combining partitioning with indexing and materialized views in cloud environment.

References

[1] Sohrabi, M. K., Ghods, V. (2016). Materialized View Selection for a Data Warehouse Using Frequent Itemset Mining, *Journal of Computers*, 11 (2) 140-148.

[2] Nath, R., Hose, K., Pedersen, T., Romero, O. (2017). A Programmable Semantic Extract-Transform-Load Framework for Semantic Data Warehouses, *Journal of Information Systems*.

[3] Lopes, C. C., Times, V. C., Matwin, S., Ciferri, R. R., Dutra, C. (2014). *Processing OLAP Queries over an Encrypted Data Warehouse Stored in the Cloud*, *In*: International Conference on Data Warehousing and Knowledge Discovery DaWaK 2014: Data Warehousing and Knowledge Discovery, p 195-207.

[4] Karkouda, K., Harbi, N., Darmont, J., Gavin, G. (2012). Confidentialité et disponibilté des données entreposées dans les nuages, *In*: 9ème atelier Fouille de données complexes (EGC-FDC 2012), Bordeaux, France.

[5] Hudic., Islam, S., Kieseberg, P., Weippl, E. R. (2012). Data Condentiality using Fragmentation in Cloud Computing, *Int. J. Communication Networks and Distributed Systems*, 1 (3/4).

[6] Kurunji, S., Ge, T., Liu, B., Chen, C. X. (2012). Communication Cost Optimization for Cloud Data Warehouse Queries, *IEEE 4th International Conference on Cloud Computing Technology and Science*.

[7] Brighen, L., Bellatreche, H., Slimani., Faget, Z. (2013). An Economical Query Cost Model in the Cloud, *In*: The First International Workshop on Big Data Management and Analytics (BDMA), China, p. 16-30.

[8] Nguyen, T.V., d'Orazio, L., Bimonte, S., Darmont, J. (2012). Cost Models for View Materialization in the Cloud, *In*: Proceeding *Data Analytics in the Cloud* (EDBT-ICDT/DanaC 2012) Workshops, Berlin, Germany, p. 47-54.

[9] Condie, T., Conway, N., Alvaro, P., Hellerstein, J. M. MapReduce Online, Yahoo! Research

[10] Thusoo., Sarma, J. S., Jain, N., Shao, Z., Chakka, P., Zhang, N., Antony, S., Liu, H. (2010). Hive –A Petabyte Scale Data Warehouse Using Hadoop and Raghotham Murthy, ICDE Conference.

[11] Arres., Kabbachi, N., Boussaid, O. (2013). Building OLAP cubes on a Cloud Computing environment with MapReduce, Computer Systems and Applications (AICCSA), ACS International Conference, 27-30.

[12] Bala, M., Alimazighi, Z. (2013). Modélisation de processus ETL dans un modèle MapReduce, ASD' 2013, p. 1-12.

[13] Favre, C., Bentayeb, F., Boussaid, O., Darmont, J., Gavin, G., Harbi, N., Kabachi, N., Loudcher, S. (2013). Les entrepôts de

données pour les nuls. . . ou pas !", 2ème atelier aide à la Décision à tous les Etages (AIDE 13) en conjonction avec la 13ème Conférence Internationale Francophone sur l'Extraction et la Gestion des Connaissances (EGC 13), Toulouse, Janvier 2013, pp. 1-18.

[14] Kishore, T. R., Devi, D.A., Prathyusha, S., Bhagyasri, D., Naresh, B. (2013). Client and Data Confidentiality in Cloud Computing Using Fragmentation Method, *International Journal of Soft Computing and Engineering (IJSCE)*, Volume-3, Issue-2.

[15] Bruno, N., Jain, S., Zhou, J. Continuous Cloud-Scale Query Optimization and Processing.

[16] Fegaras, L., Li, C., Gupta, U. (2012). An Optimization Framework for Map-Reduce Queries, University of Texas at Arlington, EDBT.

[17] Kehua, Y., Diasse, A. (2014). A dynamic materialized view selection in a cloud-based data warehouse, *International Journal of Computer Science* 11 (1) 1, 120-127, March 2014.

[18] Transaction Processing Council. (2011). Tpc-h benchmark. http://www.tpc.org/tpch

[19] Narender, M. (2016). A New approach of Securities in cloud computing applications, *International Journal of Research*, 03 (12) 539-545.

[20] Jeba, J. R., Misbha, D. S. (2016). A Study of Data Warehousing on Cloud Environment, *International Journal of Innovative Research in Science, Engineering and Technology*, 5 (7) 13697 - 13702.

[21] Kala Bharathi, K., Sandhya Sree, K. (2015). Recent Developments on Data Warehouse and Data Mining In Cloud Computing, *International Journal of Computer Science Engineering and Technology(IJCSET)*, 5 (2) 31-34.