Self-Maintainable Views Group in P2P Environment

Sebaa Abderrazak,[a] Tari A/Kamel [b]
LIMED laboratory, Dept. of Computer Science, University of Bejaia. 06000 Bejaia, Algeria
[a] balzak.sebaa@gmail.com
[b]abdelkamel.tari@univ-bejaia.dz

Abstract. Peer-to-Peer environment is gradually more popular for sharing data for distributed applications. Peer-to-peer makes these applications more reliable, efficient, available and scalable. Also, materialized views which are derived from base relations are often used to speed up query processing and data sharing. In this paper, we are interested in materialized view maintenance in P2P environment; we propose a new approach which uses some characteristics of views in such system. Our maintenance approach consists to create virtual groups of peers. Groups will contain views that sharing maximum of common data sources, then, we will associate at each group a set of auxiliary views which will be stored in a peer chosen by our algorithm, it will be the group center. The auxiliary views set will allow the group to be self-maintainable. Then, we propose a view maintenance algorithm based on two steps, notification of modifications and computing & sending of view updates. Finally simulation experiments implemented with java, show that both maintenance cost and total message number are reduced using our approach, we compared group maintenance cost in various situations in order to find the optimal conditions that our proposal is more efficient.

Keywords: view materialized; P2P; view maintenance.

1 INTRODUCTION

Peer-to-peer (P2P) systems are in full growth since several years. This paradigm allows the design of very large systems with high availability and low cost. Indeed, an important applications class in P2P system for data sharing exists, where many users must be able to work on the same data. Often materialized views are defined over tables and\or others existing views, it's one among mechanism that implement data sharing, but consistency its main issue, when the underlying data sources changes, materialized views contain non updated data and remains until their refreshment. Most of view maintenance techniques require data sources access; these approaches are not always possible because data sources may be distributed across different sites, and sometime they are unavailable, but even if they are available, communication cost of their access may be prohibitive. For these reasons, self-maintainability of materialized views is recommended (CHEN, LIU, & Rundesteiner, 2004; Gupta, Jagadish, & Mumick, 1996; Mohania, & Kambayashi, 2000). Materialized view is said self-maintainable if any view update in response to a change can be computed without using data sources (Samtani, Kumar, & Mohania, 1999). In P2P environment materialized views self maintenance will be the most appropriate solution. In this paper, we are interested in materialized view maintenance in P2P environment. Our objective is to implement the principle of self-maintenance in this environment.

The specific contributions of our work are as follows.

- Specific steps to decompose materialized views of the system to groups are proposed. Our decomposition offers several advantages, where each group contain views that share maximum of common data sources,
- An effective designation of the center of each group, the center will serve as a location for materialized auxiliary views.
- Balancing of group sizes for an efficient management of their contents.
- Materialized view self-maintenance algorithm is proposed.
- Validation of our solution by experiments simulation, we give results and corresponding analysis.

The remaining of this paper is organized as follows. Section 2 contains a brief overview of related work. Section 3 provides motivation, describes views decomposition into groups and how materializing auxiliary views. In Section 4, self maintenance algorithm is given. Experiment results are provided in section 5.

2 RELATED WORK

Materialized view maintenance has been the subject of much research, this process has definite interest with data warehouses as in (Zhuge, Garcia-Molina, Hammer, & Widom, 1995;Agrawal, El Abbadi, Singh, & Yurek, 1997; Ding, Zhang, & Rundensteiner, 1999; Zhang, Yang, & Wang, 2010; Gupta, Jagadish, & Mumick, 1996; Quass, Gupta, Mumick, & Widom, 1997; Samtani, Kumar, Mohania, 1999; Mohania, & Kambayashi, 2000; Cui, & Windom, 2000; Laurent, Lechtenborger, Spyratos, & Vossen, 2001), other are directed towards the views maintenance in the distributed environment (Mork, 2005; Agrawal, Silberstein, Cooper, Srivastava, & Ramakrishnan, 2009 ; CHEN, LIU, & Rundesteiner, 2004) , bat little work for P2P environment (Qin, Wang, & Du, 2005; Bellahsene, Cart, & Kadi, 2010; Li, & Ishikawa, 2010).

2.1 View maintenance in centralized warehouse

The data warehouse is the most context where view maintenance is studied, In (Zhuge, Garcia-Molina, Hammer, & Widom, 1995) authors proposed view maintenance algorithm in warehouse by using compensation, called ECA (Eager Compensating Algorithm), their idea is that the warehouse formulates queries to data source in order to import tuples of other data that may be affected by the received modification, to compute view update. ECA is used specially where data source state change between moment of modification reception by warehouse by and query reception by source, in order to eliminate this inconsistency, a need to compensate is necessary, which authors call compensation query. Unfortunately, ECA takes into account only one data source.

Another work, (Agrawal, El Abbadi, Singh, & Yurek, 1997) is proposed to overcomes the drawback of ECA, authors propose SWEEP algorithm, its idea is that warehouse formulates queries and sends them sequentially to all data sources to import tuples of other tables affected by modification, first query is initialized to the modification, and progressively, it's joined with tables of other sources referenced by the view. At the end of the operation, result is the required view update, this work considers the concurrent updates, but its execution is conditioned by a FIFO communication and no message loss, between source and warehouse, the downside is that each data source must contain only one table which is not always possible. With the aim to address the SWEEP issue, authors of (Ding, Zhang, & Rundensteiner, 1999) propose MRE Wrapper, based on two layers of netting. First layer

named Wrapper, located in each source site; it's responsible for each local source compensation using a local view, which communicates with the relevant tables. Second layer is the Mediator, responsible for the global maintenance using algorithm established in warehouse such as those existing as SWEEP. This algorithm requires that data sources have information of all views, which is not always obvious. Work (Zhang, Yang, & Wang, 2010), called source compensation, its main objective is to reduce the necessary cost of local compensation, this algorithm is based on an architecture level, monitor level localized in source, it treats each table separately, records and assigns version numbers to each modification in order to discover possible anomalies, integrator level which performs maintenance multi-source by sending queries to sources.

Other studies have focused on available information for views maintenance; their objective is to eliminate data sources access (self-maintenance). For instance, work (Gupta, Jagadish, & Mumick, 1996) gives the necessary conditions for self-maintainability of several view categories, also proposes algorithms to maintain SPJ views. Authors show that view self-maintenance depends on several parameters, as modification type, updated table and information about keys; unfortunately, self-Maintenance isn't always possible only in some types of views. Authors in (Quass, Gupta, Mumick, & Widom, 1997) address auxiliary views (AV), which are an additional set of views to materialize; their objective is to make materialized views self-maintainable without querying sources , and allows to reduce exchanged messages between source and view; obviously this proposal creates more storage space inside views. To reduce the storage space for auxiliary views in data warehouses, authors of (Samtani, Kumar, Mohania, 1999) use shared plans of query execution views to reduce the number of auxiliary views to materialize by computing the shared views auxiliary once and eliminating their duplication. Other work (Mohania, & Kambayashi, 2000; Cui, & Windom, 2000; Laurent, Lechtenborger, Spyratos, & Vossen, 2001) address auxiliary views of deferent view categories, as aggregate view. These various studies are limited to maintenance in central warehouse. The case of the distributed views isn't taken account, such as in distributed systems.

2.2 View maintenance in distributed environment

In (Mork, 2005; Qin, Wang, & Du, 2005), authors propose to compute necessary and sufficient information -called booster and updategram-, to update the view, authors use a set of rules called Mork rules. Indeed, when a modification is detected in source, a compute of the boosters is triggered to obtain all necessary information of view maintenance, and then result (modification and boosters) is sent to views site. The downside is that this approach requires prior knowledge of views in each data source. In (Chen, Liu, & Rundesteiner, 2004), authors address an approach called view maintenance transaction; they give an algorithm for managing access conflicts, called TxnWrap. Authors of (Agrawal, Silberstein, Cooper, Srivastava, & Ramakrishnan, 2009) address asynchronous view maintenance in distributed systems and their principle is that to defer maintenance of the most expensive processes using two mechanisms RVT and LVT.

2.3 Materialized Views in P2P environment

In (Bellahsene, Cart, & Kadi, 2010), authors propose an approach for an effective choice of materialized views and their placement in P2P environment, the main objective is to reduce views maintenance cost and to reduce queries response time. Work (Li, & Ishikawa, 2010) proposes a mechanism of query processing by using materialized views in order to reduce the response time, without proposing a mechanism for their maintenances. In (Qin, Wang, & Du,

2005), authors propose materialized view maintenance in P2P networks, they conceived a hybrid architecture of peers which is composed of peers and super-peers, they use semantic links (mapping) for processing, version numbers, and propose an asynchronous algorithm to maintain views, but a specific architecture and a total coordination in network is necessary to maintain views.

3 SELF-MAINTAINABLE VIEWS GROUP

3.1 Motivation

View Maintenance is to update view of any inconsistency using data sources; this process is more difficult in distributed views case. Indeed, in distributed systems and specifically in P2P systems, the trend is towards materialization the most requested portions and replicate data across multiple sites to reduce response time of queries, so the probability to find views referencing one or more common sources is great, unfortunately this creates a difficulty to maintain the consistency of the views , and their replicas. In such case a common processing is desired. Our idea is to decompose peers content into several groups; each group will be composed by set of views which referencing a maximum of common source. In each group, we perform few steps for to ensure maintenance of these views, these steps well be detailed later in this work. Our motivation to decompose on group to maintain views in the P2P environment:

- Carry out a common processing for identical and semi-identical views; this will reduce the maintenance cost.

- The ability to perform updates without querying sources (self-maintenance), to avoid concurrency control with modified data.

- Reduce the number of messages between sources and views by sending once the modification to maintain several views affected by this modification.

3.2 Overview

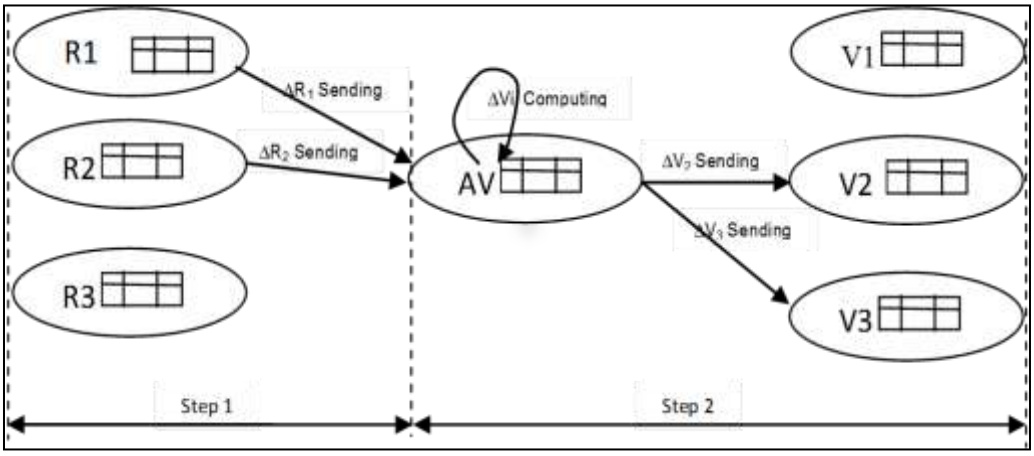

Fig. 1. Maintenance steps.

To carry out the maintenance a set of views on the P2P network, we propose an approach that spans the following steps. The first step is to decompose the set of peers storing the views in several groups; we refer to them by SMVG (Self-Maintainable Views Group). Second, we associate to each SMVG, a set of auxiliary views (AV), this will give autonomy for the maintenance process, and this set of AVs will be stored in a selected peer with optimal manner by our system in order to reduce the number of exchanged messages. The next step is self-maintenance operation, here we propose an algorithm of self-maintenance based on two steps, and as shown in Figure 1, the first step is to notify modifications, and the second is to compute update views in each SMVG. In fact, our solution is an operation similar to a materialized views factorization to produce a minimal set of shared auxiliary views, and to make the group self-maintainable.

3.3 Decomposition to SMVG

In order to decompose the set of views distributed across the peers of the network into SMVG, our proposal is carried out as follow, the first step is the choice of peers which will be centers of each SMVG, according to their views contents of referred common data sources with its neighbors, these centers (we referred them by C-SMVG) will be used as a peer for auxiliary views storage, the second step is the attachment of peers at their corresponding C-SMVG to make a group, in order to perform materialized views self-maintenance. The interest of these operations is to optimize the processing cost of views update, by computing only once views updates for shared common data sources and reduce number of transferred messages between the peers. This is carrying out by a best choice of the peers which will store auxiliary views (C-SMVG).

3.3.1 C-SMVG selection phase

The result of C-SMVG selection phase is that the selected peer as C-SMVG of each group is one that shares the largest number of source referenced with its peers neighbors. In this phase, each peer performs the following instructions:

1) Each peer P_i sends a data vector: $(R_{i,1} \dots, R_{i,n})$ to its neighbors, it contains the occurrences number of each table referenced by its views, to all its neighbors. (We have supposed that there is $\boldsymbol{n}$ tables). As shown in expression 1.

At the end of this operation each peer P_i, will know the occurrences number of all tables sources referenced by its views and its neighbors views.

Such as

$$R_{i,x}=\begin{cases}\mathbf{0} & \text{if table } R_x \text{ isn't referenced by any view of peer } P_i \\ \mathbf{1} & \text{if table } R_x \text{ is referenced by one view of peer } P_i \\ \boldsymbol{\alpha} & \text{if table } R_x \text{ is referenced by } \boldsymbol{\alpha} \text{ views of peer } P_i\end{cases} \quad \dots\dots\dots\dots\dots (1)$$

2) After reception of the data vectors from its neighbors, the peer P_i computes the note (N_{ij}), for each neighbor P_j.

It represents the views rate participation of a each peer over all the n tables of its $(m-1)$ neighbors and him, it is computed as follow (2) :

$$N_{i,j} = \sum_{x=1}^{n}\left[R_{j,x}/\sum_{y=1}^{m} R_{y,x}\right] \quad \dots\dots\dots\dots\dots\dots\dots\dots (2)$$

Such as

$R_{j,x}$: is occurrences number of table R_x referenced by views of the peerP_j.

$\sum_{y=1}^{m} R_{y,x}$: is the sum of the occurrences number of the table R_y referenced by views of peer P_j and all its (m-1) neighboring.

We note that

I. Value of $\sum_{y=1}^{m} R_{y,x}$ isn't null because it's referenced at least by one view; (Only data sources referenced by views are concerned).

II. The peer with views that reference most common tables with its neighbors, will have the largest note than the rest of peers.

3) Then, notes are sent to the concerned peers.

4) After reception of all the notes by each pair P_i, all peers identify their eligibility weight πi, based on preceding notes as follows (3):

$$\pi_i = \sum_{y=1}^{m} N_{y,i} \quad \ldots\ldots\ldots\ldots\ldots\ldots (3)$$

5) The eligibility weight are then, broadcast by each peer to all neighbors peers.

6) Each peer compares its weight with the weight of its neighbors. If its weight is the highest, it self-designates as C-SMVG. Each C-SMVG sends a notification to all its neighbors, informing them that it was elected C-SMVG.

3.3.2 Attachment phase at C-SMVG

7) After self-designation of different C-SMVG, elected peers P_i as C-SMVG compute and sent to its neighbors a load factor F_i (which is used to equilibrate the group size), as follow (4):

$$F_i = {}^{\pi_i}/_{Deg_i} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (4)$$

Deg_i: is the number of neighbors of peer pi,

π_i: is the weight of peer Pi.

8) Upon receipt of this factor by neighbors, they add the C-SMVG peer P_i to their centers selection list.

9) Each group member peer P_j select its C-SMVG by a comparison of the arrived factors (C-SMVG candidate existing in centers selection list), so the greatest value is selected as its C-SMVG peer, then informs it by its decision of attachment by sending an attach message.

3.4 Creation of an optimal set of auxiliary views

Once centers of all SMVG are selected, they launch the auxiliary views optimal set computing process (AVs). In view maintenance literature, several algorithms and methods (Quass, Gupta, Mumick, & Widom, 1997; Samtani, Kumar, Mohania, 1999; Mohania, & Kambayashi, 2000; Cui, & Windom, 2000; Laurent, Lechtenborger, Spyratos, & Vossen, 2001) are proposed to perform auxiliary views. In this work, we have used (Quass, Gupta, Mumick, & Widom, 1997) algorithm, this algorithm takes as input a view (V) and outputs the set of auxiliary views (AV), such that (V ∪ AV) is self-maintainable. Thus, in the case of common sources between two or more views, for n relations (R_1,..., R_n), referenced by views of SMVG. The set of auxiliary views of each SMGV is expressed in (5):

$$AV_{SMVG=}\{\cup A_{R1}, \cup A_{R2}, \ldots, \cup A_{Ri}, \ldots, \cup A_{Rn}\} \ldots\ldots (5)$$

Such as: $\cup A_{Ri}$: union of all auxiliary views A_{Ri} for which the relation R_i is referenced by their views

4 SELF-MAINTENANCE OPERATION

Our maintenance technique is performed in two steps, first step is an operation performed between data source peer and C-SMVG peer, and the second step is carried out inside the SMVG, between the C-SMVG peer and views peers of the same SMVG. As shown in figure 2. We detailed the role and operations performed by each peer in the following this section.

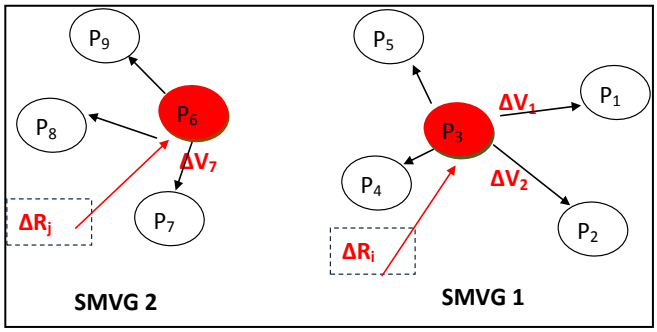

Fig 2. Views Group self-maintenance.

4.1 Data source peer

From the transaction log, table source detects and sends the modification to each group (C-SMVG) concerned by it. We assigned for every modification of source table a version number (NVRi), this same number will be assigned to the corresponding auxiliary view (NVARi). This correspondence allows detecting the lost message, this by the mismatch between the two versions of table and auxiliary view.

4.2 C-SMVG peer

This peer detects lost messages by comparing between table versions of and auxiliary view version. If any message lost detected, C-SMVG send query to table source seeking the appropriate version, it then proceeds to send the concerned modification.

Once a C-SMVG received a modification, it executes the ascending instructions to maintain views which are less composed up to most composed of data sources, after checking the received modification version. Then, our algorithm selects views affected by the current modification. After, it orders them according to the number of tables that compose them (view referenced by little source tables in first, much tables at last), thereafter it launches the computing of the deltas of each view according to the order envisaged, each time it looks for opportunities to reuse previously computed deltas views. For this computing updates, we use the rule of Mork (Mork, 2005) as follows (6):

$$\Delta R_i \bowtie^{v} A_{Ri} = \Delta V \text{.....(6)}$$

Such as A_{Ri} is an auxiliary view.

At the end, the computed updates will be sent towards the peers having the views affected by the modification△Ri.

```
Algorithm self-maintenance (ΔRi,V) : ΔV
Input
 ΔRi,   modification;
A=(A_R1 ,...,A_Rn);                    /*Auxiliary view concerned by ΔRi,*/
V=(v_1, ...,v_n);                      /* view concerned by ΔRi */
Output
ΔV=(Δv_1 ,..., Δv_n),                  /*views updates*/
BEGIN
 Repeat (  Version (ΔR_i) ≠ [Version(A_Ri)+1]   )

        Send( ( ?ΔR_i), Version(A_Ri)+1, Peer_Source)
                                       /*seek appropriate version*/
       Wait ( waiting time );          /*Waiting*/
Until (   Recive(ΔRi, Version(A_Ri)+1, Peer_Source) )
 Orderd (V);                           /*Order views */
ΔV_0 ← ΔRi;                            /*Initialization*/
 Repeat
          Compute(Δv_(j-1), A_Ri, Δv_j) ;
                        /*compute Δv_j using Δv_(j-1) , A_Ri , δ,π ,⋈ */
         Add (Δv_j, ΔV);               /*adding computed Δv_j */
          Eliminate (v_j , V);         /* eliminate v_j*/
          Send (Δv_j, pair_view) ;     /* Sendig Δv_j */
 Until  (V = ∅)
END
```

4.3 View peers

After reception of view updates, the peer view updats its view Vi by using the received △Vi, as follows (7):

$$V_{ver\ (i+1)} = V_{ver\ (i)} \cup \triangle V \text{.....(7)}$$

5 PERFORMANCE EVALUATION OPERATION

5.1 Experimentation description

The computer used for testing was a CPU Intel Core Duo T5670 with 2.0 GHz processing rate and 2GB of RAM, its operating system is Windows XP Professional SP3. Our programs developed in Java using JDK / JRE 1.6.0. We perform many iterations of simulation on a P2P network (3000 peers) generated by our program. The objective is to know what the impact of our proposal SMVG on messages number, the maintenance cost and impact of

SMVG size on two previous settings. Our simulations are carried out in three strategies: **AM**: views maintenance without SMVG. **SMVG(1)**: maintenance by groups of a 4 peers average. **SMVG(2):** maintenance by groups of a 15 peers average.

5.2 Results and analyzes

5.2.1 Number of exchanged message

To illustrate impact of peers number involved to create each SMVG, we performed several iterations by varying the average peers number that participate in SMVG, we increases groups size, as shown in Figure 3, it's represented as x axis. We note the first value (2) corresponds to a group with only two peers. This graph describes the interest of SMVG in reducing exchanged messages. When a group size increases, the number of messages required to maintain views is smaller, this trend is explained by the increasing of common sources, so modification is sending at C-SMVG only once(single shipment**).**

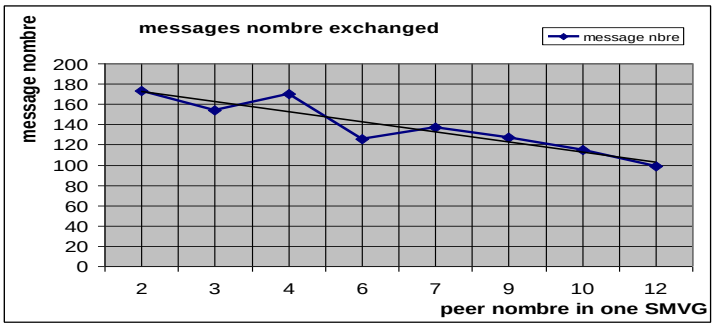

Fig 3. Messages number

The histogram Messages-number = f (strategies) on Figure 4 shows the impact on the reduction of number of messages with our solution in SMVG(1) and SMVG(2), this is explained by the single shipment of updates.

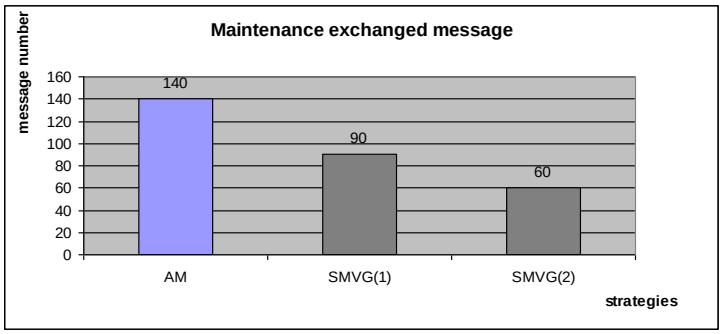

Fig 4. Messages number according to the strategy.

5.2.2 Total maintenance cost

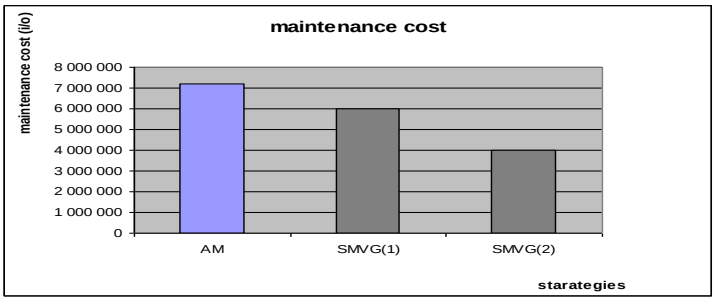

Fig 5. Global maintenance cost.

To clarify the impact of our technique on maintenance cost and processing time, we performed a comparison of maintenance cost (number of input\output disk) in the tree strategies, ie the x axis represents strategy used to maintain the views. Y axis represents the number of inputs and outputs that drive a factor indicating the processing time necessary for the maintenance of all affected views. As shown in figure 5, the trend is the reduction of cost by using our technical and whenever the group size is large. This is explained by common processing of views having common sources.

5.2.3 Average maintenance cost by group

In order to find the optimal conditions that our proposal is more efficient, we compute maintenance cost of each SMVG established. Graph below represents the variation of the average maintenance cost (reflects by processing time in C-SMVG) by one group according to the average peer number of this group.

Fig 6. Average maintenance cost by group

Graph in figure 6 illustrates three different sections, the first section where peers number in a group is more than ten (10); it's characterized by a very high group maintenance cost, which generates a long and significant processing time, which isn't always obvious for the C-SMVG which manages maintenance tasks. Second section where number of peers in the groups is between five (5) and ten (10), it's characterized by a reasonable maintenance cost or

even reduced so less processing time in the C-SMVG. Finally the third section where peers number in group is lower than 5, this section is characterized by a reduced group cost but many groups ie more significant total maintenance cost and number of messages more significant, so it isn't interesting. Then, second section (between 5 and 10 peers) is more adapted for groups' decomposition. Which justify the need to limit groups' size to keep a reduced messages number and a rational maintenance cost.

Our simulation demonstrates the reduction of total messages number and the overall maintenance cost required using SMVG, also we have noted that whenever the peers number that form a SMVG is larger, the total number of messages and the overall cost of maintenance is reduced, however, the processing load of each C-SMVG is more significant, which requires us to limit the peers number in each SMVG so reasonably.

6 CONCLUCSION

In this paper, we proposed a materialized view maintenance approach in P2P environment, so the data sources and views are geographically distributed on different peers. In this contribution, the maintenance cost reduction is our main objective. Our proposal is based on the notion of the self-maintainable views group (the views sharing maximum common data sources) for which auxiliary views will be materialized and stored in a selected peer (C-SMVG), this set of auxiliary views offers self-maintainability to group. We propose view maintenance algorithm based on two steps, notify the modification to C-SMVG and sending updates to different views affected by the modification. Simulations show the reduction in total messages number and the overall maintenance cost using our technique (SMVG), also we have noted that whenever the peers number that form a SMVG is larger, the total messages number and the overall maintenance cost is reduced, however, the processing load of the center of each SMVG is more important, which requires us to limit the peers number in each SMVG so reasonably.

References

Chen, S., Liu, B.& Rundesteiner, E A. (2004). *Multiversion-Based View Maintenance Over Distributed Data Sources, In Proceedings of the International Conference of Transactions on Database Systems, ACM, Vol: 29, No: 4 , 675–709*.

Agrawal, P., Silberstein, A., Cooper, B., Srivastava, U., & Ramakrishnan, R. (2009). *Asynchronous view maintenance in VLSD databases, In SIGMOD International Conference on Management of Data*.

Bellahsene, Z., Cart, M., & Kadi, N. (2010). *A cooperative approach to view selection and placement in P2P systems, In Proceedings of 18th International Conference on Cooperative Information Systems CoopIS, 515–522.*

Li, F., & Ishikawa, Y. (2010). *Query processing with materialized view in a traceable P2P Record Exchange Framework, The 2nd International Workshop on Web-Based Contents Management Technologies, 246–257*.

Qin, B., Wang, S., & Du, X. (2005). *Effective maintenance of materialized views in peer data management systems, In Proceedings of 1st International Conference on Semantics, Knowledge and Grid*.

Gupta, A. , Jagadish, H. , & Mumick, I. (1996). *Data integration using self-maintainable views. In Proceedings of International Conference of Advances in Database Technology, Springer, Vol:1057, 140-144.*

Quass, D., Gupta, A., Mumick, I.S., & Widom, J. (1997). *Making Views Self-Maintainable for Data Warehousing, In Proceedings of the International Conference of Parallel and Distributed Systems.*

Samtani, S., Kumar, V. , & Mohania, M. (1999). *Self Maintenance of Multiple Views in Data Warehousing. In Proceedings of international Conference on Information and Knowledge Management.*

Ding, L. , Zhang, X., & Rundensteiner, A. E. (1999). *The MRE Wrapper Approach: Enabling Incremental View Maintenance of Data Warehouses On Multi-Relation Information Sources, In Proceedings of the 2nd ACM international workshop on Data warehousing and OLAP.*

Mohania, M., & Kambayashi, Y. (2000). *Making Aggregate Views Self Maintenable. In Data Knowledge Engineering, Vol: 32, Issue 1, 87–109.*

Laurent, D., Lechtenborger, J., Spyratos, N., & Vossen, G. (2001). Monotonic Compliments for Independent Warehouses. In VLDB journal.

Zhuge, Y., Garcia-Molina, H., Hammer, J., & Widom, J. (1995). *View Maintenance in a Warehousing Environment. In Proceeding of ACM ACSIGMOD International conference on Management of data, 316 – 327.*

Mork, P. (2005). *Peer Architectures for Knowledge Sharing. PhD thesis. University of Washington.*

Agrawal, D., El Abbadi, A. Singh, A., & Yurek, T. (1997). *Efficient View Maintenance at Data Warehouses. Proceedings of ACM SIGMOD international conference on Management of data, 417 – 427.*

Zhang, X., Yang, L, & Wang, D. (2010). *Incremental view maintenance based on data source compensation in data warehouses. ICCASM, 287-291*.

Cui, Y., & Windom, J. (2000). *Storing Auxilary data for Effecent Maintenance. in 2nd DMDW, Sweden.*

Sebaa Abderrazak: Lecturer in the department of computer science and researcher in LIMED (laboratory of Medical Computing) in Bejaia University. He received his engineering and magister degree in computer science at Bejaia University.

Tari Kamel: Head of LIMED (laboratory of Medical Computing) since February 2013 and team leader of data mining. He held different positions at the university of Béjaia (Head of Computing department, Operational Rechearch department, head of doctoral school) and at the National School of Computing ESI (ex.INI), Algiers between 1986-1994. After his Diploma in Mathematics at USTHB (Algiers), he held his Master by Research in Lancaster University (England). He defended his PhD Thesis in Computer Sciences in 2008 at the university of Béjaia and his Accréditation to supervise (HDR) at ESI (ex-INI) in 2010.