Consistency-aware Evaluation of OLAP Queries in Replicated Data Warehouses*

Javier García-García
Universidad Nacional Autónoma de México
UNAM, Mexico City, CU 04510, Mexico
Carlos Ordonez
University of Houston
Houston, TX 77204, USA

***Abstract*—OLAP tools for distributed data warehouses generally assume underlying replicated tables are up to date. Unfortunately, maintaining updated replicas is difficult due to the inherent tradeoff between consistency and availability. In this paper, we propose techniques to evaluate OLAP queries in distributed data warehouses assuming a lazy replication model. Considering that it may be admissible to evaluate OLAP queries with slightly outdated replicated tables, our technique first efficiently computes the degree of obsolescence of replicated local tables and when such result is acceptable, given an error threshold, then the query is evaluated locally, avoiding the transmission of large tables over the network. Otherwise, the query can be remotely evaluated less efficiently with the master copy of tables, provided they are stored at a single site. Inconsistency measurement is computed by adapting distributed set reconciliation algorithms to efficiently compute the symmetric difference between the master and replicated tables. Our improved distributed database algorithm has linear communication complexity and cubic time complexity in the size of the symmetric difference, which is expected to be small in a replicated data warehouse. Our technique is independent of the method employed to propagate data warehouse insertions, deletions and updates. We present experiments simulating distributed databases, with different CPU and transmission speeds, showing our method is effective to decide if the query should be evaluated either locally or remotely.**

Categories and Subject Descriptors

H.2.7 [**Database Management**]: Database Administration - *Data warehouse and repository*

General Terms

Verification, Experimentation, Theory

Keywords

Set reconciliation, replica consistency, distributed databases

I. Introduction

Fault tolerance is an important reason to keep distributed data warehouses continuously available at different sites. Another important reason is the existence of multiple sources of large amounts of data in today's data warehouses. Today it is common to use distributed techniques in order to place a data warehouse in several computers under a single conceptual schema. To assure efficient response times to OLAP users increasing local access and reducing network loads and to achieve a high level fault tolerance, replication techniques are used in the implementation of distributed data warehouses. The replication has been proposed in several ways such as horizontal or vertical fact replication, complete or partial dimension table replication [3]. A data mart, for example, is a data repository that may derived from subsets of data in a data warehouse. It is common that all these techniques turn the systems complex and difficult to manage [1]. Replication brings up different problems such as storage overhead and replica inconsistency due to, for example, asynchronous updating, system failure or data corruption. Replication can be done by replacing all the dataset or by changing only updated records, however there is always a risk to lose replica consistency. Although the ETL process takes care of the integrity errors, the on going management processes can bring inconsistency to the data warehouse.

In this work we propose a strategy to be included in OLAP tools that work over distributed data warehouses that follow a lazy master replication model where the replicated tables could be outdated. A user may request a currency evaluation of replicated tables involved in an OLAP query. Also, the tool may automatically compute a query evaluation with tables that meet a certain currency threshold resulting in a local or a remote query evaluation. Our tool is useful also in active data warehouse environments where near real-time data freshness is needed but also mission critical data availability. In these environments, knowing the freshness condition of the replicas becomes critical. Our strategy is based on set reconciliation algorithms that compute efficiently the symmetric differences between the master and slave table replicas. Our techniques are especially suitable when comparing large sets, in our context large tables, with relatively few differences, as in our context, when comparing two supposedly equal replicas.

The article is organized as follows. Section II introduces basic definitions. Section III introduces our approach to compute replica consistency in a distributed data warehouse and identifies query optimization issues. Section IV provides an extensive evaluation of our technique to compute our proposal.

*This is the authors version of the work. The official version of this article was published in Proc. ACM Workshop on Data Warehousing and OLAP (DOLAP, CIKM 2009 Workshop),p.73-80, 2009

Related work is discussed in Section V. Section VI concludes the article.

II. Definitions

Let $\mathcal{D} = \{D_1, D_2, \ldots, D_n\}$ be a set of n replicas of a data warehouse at n sites, where D_1 is the primary copy at the master site, and each $D_i, i = 2 \ldots n$ is a secondary copy of the primary copy at a slave site. To have a uniform framework, all replicas have the same table names (i.e. same structure), but potentially different content (i.e. inconsistent replicas).

Let D_i be defined as $D_i(\{T_1, T_2, \ldots, T_m\})$, where each $T_p,\ p = 1 \ldots m$ is a table. Since we deal with outdated replicas we assume $D_i.T_p$ may contain rows different from $D_j.T_p$, where $i \neq j$ and $D_i, D_j \in \mathcal{D}$.

Let $D_i.T_p$ be a replica of table T in the local data warehouse D_i. Each local secondary replica table $D_i.T_p,\ i = 2 \ldots n$ is periodically refreshed according to the updates at the primary replica table $D_1.T_p$. We define replica consistency as follows:

Definition 1 Replica consistency. *Let $D_i,\ i = 2 \ldots n$ be a secondary replica of data warehouse D. D_i meets the replica consistency constraint if it is an identical copy of the primary replica D_1. That is: $\forall\ T_p \in D_i,\ D_1.T_p = D_i.T_p$. The primary replica is always (trivially) replica consistent.*

A subset of tables in D_i can meet this type of consistency. In this case, the subset of tables should be identical copies of their corresponding primary copies.

The following metric is used to know if a replica of a given table, say $D_i.T_p$, meets the replica consistency constraint (i.e. if it is current).

$$cur(D_i.T_p) = \frac{|D_i.T_p \ominus D_1.T_p|}{|D_1.T_p|} \quad (1)$$

where $D_i.T_p \ominus D_1.T_p$ is the symmetrical difference between tables $D_i.T_p$ and $D_1.T_p$. That is, the records that are in $D_i.T_p$ but not in $D_1.T_p$ with the records that are in $D_1.T_p$ but not in $D_i.T_p$:

$$D_i.T_p \ominus D_1.T_p = (D_i.T_p - D_1.T_p)\ \cup\ (D_1.T_p - D_i.T_p)$$

we denote the size of the symmetrical difference by d. That is $d = |D_i.T_p \ominus D_1.T_p|$.

The metric shown in Equation 1 gives a close idea of the table replica consistency. That is, it gives an idea of how similar the secondary replica is compared to the primary replica. Observe that this metric will be close to zero if the secondary replica is nearly identical compared to the primary replica. Also, it gives an idea of how different the involved tables are. As the quotient grows, the number of different records is greater. Observe that $D_1.T_p - D_i.T_p$ is the set of records that are not yet propagated to $D_i.T_p$. That is, the records inserted in $D_1.T_p$ not yet in $D_i.T_p$. As for $D_i.T_p - D_1.T_p$ is the set of records not yet deleted from $D_i.T_p$. As for the updated records in $D_1.T_p$, we will consider an update as two operations, i.e. a delete and an insert of a record.

We can assume then that to update/repair a replicated table, say $D_i.T_p$, we need to propagate the insert and delete operations from $D_1.T_p$ to $D_i.T_p$.

Definition 2 Repaired table with respect to replica consistency. *We say a replicated table, say $D_i.T_p$, is repaired with respect to replica consistency if*

$$D_i.T_p = (D_i.T_p \cup (D_1.T_p - D_i.T_p)) - (D_i.T_p - D_1.T_p) \quad (2)$$

III. Efficient Computation of Replica Consistency

We start by presenting alternatives to maintain consistency in replicated data warehouses. We then motivate the use of set reconciliation distributed algorithms to efficiently measure consistency. We explain step by step our approach with a small example. Our main technical contribution is a distributed algorithm that extends existing set reconciliation algorithms to work in a distributed database assuming a lazy master-slave replication model. We show our algorithm is efficient assuming replicas have a small number of missing records: communication complexity is proportional to the symmetric difference of a master table and its replica, and not to their sizes. We conclude this section discussing several practical considerations.

A. Alternatives to Maintain Consistency

Observe that the number of records in a table could be very large as in a data warehouse environment. However, the number of differences between the primary replica and a secondary replica is expected to be relatively low. That is $d << |D_1.T_p| \approx |D_i.T_p|$. An alternative to compute the operations involved in Equation 1 could be to transfer to the slave site the primary replica involved and then compute the operations. This alternative could be very expensive in terms of communication complexity since the operands could be tables with millions of records. Once the data is transferred to the slave site, computing the operations involved is also expensive. An advantage of this alternative is that it does not depend on a given propagation process. On a given time, it could the triggered in order to verify replica consistency. Other strategies to maintain consistency assume first as starting base that a given replicated table is repaired. In the master site they keep track of the updated records in a propagation set. At a transfer timepoint the master site sends its propagation set to the slave site. Upon receiving the propagation set the freshness condition may be updated taking the previous, consistent state into account. The problem with these strategies is that they assume an initial consistent state or a periodic one if the strategy includes repairing of the slave replicated table. If the system looses track of these cyclic evaluations, for example, because of a communication problem, the precision of the freshness evaluation may be compromised. To return to a consistent state, the table has to be repaired by other methods that match both master and slave replicas. In other words, these strategies are not self-contained.

B. Set Reconciliation Approach to Measure Consistency

Computing Equation 1 is challenging considering the size of the replicas in a data warehouse environment. We are assuming a lazy master replication model [5]. In each slave site, the

most important operation is the symmetrical difference. Our techniques focus on optimizing this operation. We adapted to our problem set reconciliation [8] techniques. We assume the cost to transfer a large table is high as in a data warehouse environment, and we assume the size of the symmetrical difference is low compared to the size of the tables involved, since the tables are supposed to be replicas. In our approach we are assuming that in the primary replica records are inserted, deleted and updated.

The amount of data to be transferred may be dramatically reduced and the computations may be simplified considering the ideas we will present next. The question that remains now is how can we efficiently compute the symmetrical difference shown in Equation 1 in a given slave site without transferring or broadcasting large amounts of data and/or without assuming a consistent initial state. How can we compute in site i the differences $D_1.T_p - D_i.T_p$ or $D_i.T_p - D_1.T_p$ without transferring from the master site the primary replica. To accomplish this goal, we propose to adapt techniques used to reconcile sets whose differences are small [8], that avoid the need to transfer large amounts of data. The amount of data transferred depends on the number of differences rather than on the size of the tables involved. That is, the communication complexity is $O(d)$ instead of $O(|D_1.T_p|)$. This is a great difference considering that in a data warehouse context $d << |D_1.T_p|$. On the other hand, once the data is transferred, the computational complexity of the algorithms that compute the actual different values could be cubic in the number of differences, $O(d^3)$. To the best of our knowledge, these techniques for set reconciliation have never been proposed to discover differences between replicated tables in a distributed data warehouse environment. Determining the replica consistency of tables in a distributed data warehouse where we deal with big sets with proportionally small differences is a scenario specially adequate for the use of these techniques.

Once Equation 1 is computed, replica consistency may be assessed and the user may decide at which site an OLAP query can be computed or even decide to refresh the replica.

C. Adapting Set Reconciliation to Compute Replica Consistency

To compute replica consistency, we adapted set reconciliation techniques. The following assumptions and actions were taken into consideration:

- The sets to be reconciled (the records of two replicas) are represented by their characteristic polynomials.
- A number of evaluation points mutually agreed, in our case, between the master site and the slave site must be used to evaluate the characteristic polynomials. The number of points evaluated must not be less than the maximum number of differences between the two replicas.
- Instead of transferring one of the reconciling sets from one site to the other, the evaluation points with their corresponding characteristic polynomial evaluations are transferred, this way minimizing the communication complexity.
- Both sets of characteristic polynomial evaluations are used to interpolate a rational function which will have as roots of its numerator and denominator the symmetrical difference between the sets of records of the two replicas.

The following example gives an overview of how the mentioned technique can be used to repair a table with respect to replica consistency, although it is not our intention to give here all the technical details of the set reconciliation algorithm (See [9] for details).

Example 1 *Suppose table T, with* 100 *records is replicated at two sites, say site* c *and site* r *in an asynchronous master-slave replication configuration, being* c *the master site and* r *the slave site. To simplify exposition, assume a unique attribute, the primary key K, which holds positive integers stored in a 8-bit signed integer. Suppose the number of elements in the symmetrical difference between the two sets of primary key values is no more than* 20*. To reconcile both replicas, first both sites agree in* 20 *evaluation points, one for each possible different value. At each site the characteristic polynomial, that is, the polynomial whose roots are all the primary key values of the corresponding replica, is evaluated in each one of the* 20 *evaluation points. Let $k_i, i = 1 \dots 100$ be the primary key values of the replica at site* c*. The corresponding characteristic polynomial would be then $\Pi_{i=1}^{100}(x - k_i)$. Let k_e be one of the* 20 *evaluation points. The evaluation of k_e using the mentioned characteristic polynomial is $\Pi_{i=1}^{100}(k_e - k_i)$. To avoid working with large numbers while computing the evaluations of the characteristic polynomials, we need to work in a finite field with an order not less than* v $= (2^7 - 1) + 20 = 147$ *in order to map the primary key values and the evaluation points. The selected field could be then $\boldsymbol{F}_q$,* q$=$ 149*, being the value of* q *the lowest prime greater than* v*. Observe we can fix the value of* q *since the domain of the primary key values does not change as well as the upper bound of the number of elements in the symmetrical difference. The* 20 *mutually agreed evaluation points may be the integers in the closed interval $[-20, -1]$. Since these numbers are not members of the set of possible primary key values, they will never be zeros of the characteristic polynomials. Both sites keep an updated vector, lets call it the evaluation point vector, with* 20 *pairs of values containing each evaluation point associated to its corresponding evaluation of the characteristic polynomial. Also, both sites keep the cardinality of their replica. To keep the vector updated, whenever a record with a primary key value of say k_0 is inserted/deleted, all the* 20 *current evaluations of the characteristic polynomial have to be multiplied/divided (mod* q*) by $(k_e - k_0)$, where k_e is an integer in $[-20, -1]$. The cardinality is updated as well by adding/subtracting one to the current cardinality. Suppose that in a given moment both replicas agree in all their primary key values, and suppose these values are the positive integers in the closed interval $[1, 100]$. In both evaluation point vectors the value that corresponds to the evaluation point -1 is*

$$110 = (-1-1)(-1-2)...(-1-100) \text{ mod } q$$
$$where\ q = 149$$

Suppose site c *receives three records with values* 101*,* 102*,*

103 *in its primary key* K *and the record with value* 100 *in* K *is deleted. The updated evaluations of the characteristic polynomial that correspond to the evaluation points* -1 *and* -2 *are* 15 *and* 129 *respectively.*

When the freshness evaluation is required in r*, site* c *transfers its evaluation point vector to* r *and the cardinality of its replica. Observe that the size of the transferred data depends on the number of symmetrical differences and not on the table cardinality. At site* r*, with both vectors, at each evaluation point, the two values of the evaluation of the characteristic polynomials are divided (mod* q*) and the divisions are used together with the cardinality of the functions, to interpolate a reduced rational function which will have in its numerator and its denominator characteristic polynomials whose roots are the values that are in one set but not in the other. Take for example the values of both vectors that correspond to the evaluation point* -1 *above. The division between both values gives (mod* q*)*

$$\frac{15}{110} = \frac{(-1-1)(-1-2)...(-1-99)(-1-101)(-1-102)(-1-103)}{(-1-1)(-1-2)...(-1-99)(-1-100)}$$
$$= \frac{(-1-101)(-1-102)(-1-103)}{(-1-100)} = \frac{7}{-101} = 34$$

Observe that the common factors cancel out so the result is the evaluation of a rational function whose zeros in the numeration and the denominator are the values that are in one set but not in the other.

D. Distributed Algorithm to Measure Replica Consistency

In case the maximum number of different values of one replica with respect to the other is not known and we only count with a global bound, the cardinality of the tables together with the global bound can be used to determine the bounds of the degrees of both characteristic polynomials in order to interpolate and reduce the rational function. By finding the zeros of the characteristic polynomials, the symmetrical difference between the master and the slave replicas can be determined in site *r*.

In case the keys are not integers, the key domain values are mapped to the values of a finite field. This can be done considering their binary representation. To handle tables with more attributes, the attribute values can be concatenated and the result mapped as mentioned above. If necessary, the table can be fragmented keeping in each fragment the primary key in order to rebuild the records that belong to the symmetrical difference.

Next, in Algorithm 1 we show how to use the set reconciliation techniques mentioned above in order to implement our proposal. We will refer to this method as the *Set Reconciliation Approach* - SRA. In each step we show if it is a computational (pt) or a communication (ct) task and if it is performed by the master site (c) or by a slave site (r).

E. Communication and Time Complexity

The communication complexity using the set reconciliation techniques just described, SRA approach, depends on the number of differences among the tables involved and the size of the records, $O(d'b)$, step 1 in Algorithm 1. At the slave site, the computation to interpolate the rational function using the evaluation point vectors by Gaussian elimination and the Euclidean algorithm to reduce the rational function, has complexity $O((d')^3)$ [9]. We are assuming that the tasks described in step 0 are done asynchronously. If this was not the case, we have to add to the computational complexity the cost to evaluate the evaluation point vector, $O(|T| * d')$.

Algorithm 1 Algorithm to determine symmetrical difference (c - master site, r - slave site, pt - computational time, ct - communication time).

0. (asynchronously): Sites agree in the maximum number of total different records between the two replicas, say d', and in the same number of evaluation points. At each site, the characteristic polynomial formed with the records of T is updated in a table, say E, in each one of the d' evaluation points when an insert or delete takes place. Table E is the evaluation point vector. The cardinality of T, $|T|$, is also updated.

1. (ct): Site c broadcasts $|D_c.T|$ and table $D_c.E$. $D_c.E$ has the characteristic polynomial of $D_c.T$ evaluated in each one of the d' evaluation points as explained above.

2. (pt): Site r receives $|D_c.T|$ and $D_c.E$ and together with $|D_r.T|$ and $D_r.E$ interpolates a reduced rational function and computes the symmetrical difference between the two replicas $(D_r.T - D_c.T)$ and $(D_r.T - D_c.T)$

F. Issue: Unknown Upper Bound on Symmetric Difference Set Size

There are cases where an upper bound of the maximum number of records of the symmetrical difference between any two replicas, d', cannot be determined. We present an overview of a method to apply the techniques proposed without knowledge of an upper bound, considering tables $D_c.T$ and $D_r.T$. This method is an adaptation of the probabilistic verification presented in [9].

We estimate a maximum number of differences between any two replicas above which it would be unfeasible to use this method. Observe that the real number of differences could be bigger. We determine a first set of evaluation points mutually agreed between the involved sites with this number of points. Lets call this set Q. Next, we need to determine a number of additional evaluation points that will be used to test with a given probability if an interpolated rational function is indeed the function we are looking for. To do this, we determine a low probability, p, representing the maximum probability we can tolerate for the case where we erroneously interpolate a rational function different from the correct rational function having as roots of the numerator and the denominator the values of the symmetrical difference. According to this probability we determine a second set of points mutually agreed drawn randomly from a subset of the field of the rational function we are willing to determine. We will use these random evaluation

points to find out if the computed rational function is indeed the desired one with certain probability. Lets call this set S. Now our evaluation point vector, the one that corresponds to table E in the variant presented in the last section, will hold the evaluations of the characteristic polynomials of Q and S.

When the freshness evaluation is required, the slave site, r, receives the evaluation point vector of its counterpart, site c, with $|Q| + |S|$ evaluation points and its cardinality, and compute the corresponding divisions. Next, the interpolation of the rational function takes place, assuming the number of symmetrical differences is $|Q|$ and the test to see if this function is in fact the rational function we want with a probability of failure of p. This is done by comparing the divisions that correspond to the points in S against the values of the interpolated function evaluated at those same points. If the test is successful, then the current interpolated function is reduced and the zeros are obtained. If at least one point fails, then this method is not feasible to determine the differences. Each time the set S of random points is used for this purpose, the sites involved compute another mutually agreed random set. Observe that this can be done asynchronously. Also, since normally S is small, several sets can be maintained simultaneously. Since the interpolation is done once, and $|S|$ is fixed, the computational complexity in site r is $O(|Q|^3)$ and the communication complexity is $O(|S| + |Q|)$. Bearing in mind this complexity we can better estimate the size of Q. It only remains to show how to compute the size of S from p. Observe that if the values of a key can be mapped to a field of values of size b-bits we can add an additional bit to enlarge the field and we have 2^b more values of size b-bits, where we can take the values of Q and S. Since Q is a fixed set of values, the random values may be chosen from a set of $2^b - |Q|$ values. According to Theorem 4 in [8] and considering the field described above, the probability that two monic rational functions with the sum of their numerator and denominator less than B agree in one randomly selected point although they are different is no more than

$$\overline{p} = (B - 1)/(2^b - |Q|).$$

where $B = |D_c.T| + max(|D_r.T|), 1 \leq r \leq n-1$, or another upper bound of the maximum number of different values between any two replicas, say for example, the maximum number of inserts and deletes (recall we are considering an update as if a delete and an insert had taken place) in a period of time.

Observe that since the rational function is generated from the division of the characteristic polynomials the way it was described, two different rational functions cannot agree in more than the number of points equal to the size of the symmetrical difference of the involved sets minus one. Observe that if the two rational functions agree at more than this number of points, then they are equivalent. With these ideas in mind, we can determine that the probability that two rational functions are different after agreeing at e consecutive randomly selected evaluation points is no more than $B\overline{p}^e$. Let this probability be p, then the size of S is

$$|S| = \lceil log_{\overline{p}}(p/B) \rceil \geq \lceil log_{\overline{p}}(p/d) \rceil$$

where d is the real number of the maximum number of total different values between the two replicas.

G. Technical Considerations

Our approach offers a compromise among the following criteria: replica freshness, modern requirements to compute OLAP queries with updated data, communication efficiency and self containment in the computation of replica consistency. The user of an OLAP tool or a data warehouse application wishing to perform the computation of the replica consistency, only needs to determine the threshold d', the maximum size of the admissible symmetrical difference. This value is transmitted to the master and slave replicas in order to determine the number of evaluation points, according to Algorithm 1.

All the arithmetic operations to compute the characteristic polynomials and to obtain the roots of the rational function are done in a finite field to avoid working with large numbers. In case that the concatenation of the binary representation of the attribute domain values is too big to be computed, the table can be vertically fragmented. Each fragment should keep the primary key value in order to eventually assemble the records that belong to the symmetrical difference. Each fragment is treated as a different table and the computations needed to obtain the characteristic polynomials and the roots can be done in parallel in the corresponding site. Note that the computations to update the evaluations of the characteristic polynomials are done asynchronously locally in each of the corresponding site limiting the need to communicate with other sites.

An important contribution of our work is the proposal of a technique to measure replica consistency in a distributed data warehouse context with a communication complexity that depends on the size of the symmetrical difference between the replicas. Efficient computations to compare data warehouse tables have been proposed, for example, in the context of the snapshot differential problem [6]. However, these techniques in the distributed context have a communication complexity that depends on the size of the tables. Our computations lead also to a method to repair the slave replica in a way that is independent to the propagation method used by the distributed data warehouse system. A distributed data warehouse where we deal with big sets with proportionally small differences is a scenario specially adequate for the use of these techniques. The computational complexity in the slave site using the set reconciliation techniques to obtain the symmetrical difference between replicas, although it is cubic in the number of differences, which, however, is expected to be low, it does not depend on the size of the tables.

IV. Experimental Evaluation

The main goal of our experimental evaluation is to compare our proposal to measure replica consistency based on set reconciliation techniques computing the characteristic polynomials (we will refer to this method as the *Set Reconciliation Approach* - SRA), with other methods that require the transmission of the master replica to the slave site, here the

Algorithm 2 General algorithm for complete transfer approach (c - central site, r - remote site, pt - computational time, ct - communication time).

1. (ct): c broadcasts $D_c.T$.
2. (pt): r receives $D_c.T$, and computes $D_r.T \ominus D_c.T$ (this could be done with, for example, an SQL expression inside the DBMS, or a Java function outside the DBMS), and generates tables $D_r.T - D_c.T$ and $D_c.T - D_r.T$.

Complete Transfer Approach - CTA algorithms. Algorithm 2 shows the basic framework of the CTA algorithms.

With the CTA algorithms, computing the symmetrical difference in the slave site could be useless since in the lazy master replication model, the master replica is the correct replica so the slave one simply could be substituted. However, determining the symmetrical difference could be useful for quality assurance purposes. Nevertheless, the CTA algorithms have a communication complexity that depends on the size of the replica.

In our experiments we want to show how both approaches, SRA and CTA, behave differently. We stress that we are not using the best CTA algorithms that require at least a table scan to obtain the evaluation of the replica consistency. However, what is important here is that the communication and computational complexity of these algorithms depend on the size of the evaluated table. We present here our experiments done with sizes of tables that highlight the efficiency of our SRA methods even in scenarios where tables are not so large.

A. Setup

We conducted our experiments on four database servers. Two of them with one Intel Core 2 Duo CPU at 1.66 GHz with 1 GB of main memory and 120 GB on disk. The other two servers with two Intel Core Duo CPU at 1.66 GHz with 2 GB of main memory and 160 GB on disk. We identified both types of servers as SS and FS servers respectively (slow and fast servers). We used two networks one with a transmission speed of 0.1 MB/s, and the other one with a transmission speed of 10 MB/s. We identified both types of networks as SN and FN respectively (slow and fast networks). We installed Fedora 8 in all servers with gcc 3.6 and java 1.6. We adapted the reconciliation functions available from [14]. We simulated a Master-Slave configuration with the relational DBMS PostgreSQL 8.2.

Our synthetic databases were generated by the TPC-H DBGEN program, [15], with scaling factors 1 and 2. The results we present in this section, unless stated otherwise, use a default scalefactor 1. We decided to present these results in order to highlight that even with data warehouses with tables not so big, our techniques still perform better compared with techniques that require the transmition of an amount of data proportional to the size of the evaluated table. The tables we replicated without loss of generality were projections of the primary key of the tables $customer$, $orders$ and $lineitem$, denoted here by $pkcustomer$, $pkorders$ and $pklineitem$ respectively, with the following sizes: 150k, 1.5M and 6M respectively and the following primary keys: $c\_custkey$, $o\_orderkey$ and a compound primary key ($l\_orderkey$, $l\_linenumber$) respectively. Each experiment was executed five times, we eliminated the fastest and slowest execution and reported the average of the rest. Elapsed times are indicated in seconds.

In every experiment, we recorded separately the total elapsed time to execute our Algorithms in the servers (computational time - pt) and the total time for the communication (communication time - ct).

B. Complete Transfer Approach vs. Set Reconciliation Approach

In this section we compare the CTA approach against the SRA approach. To compute the symmetrical difference in the slave site with the CTA approach first the master replica was transmitted to the slave site. Once in the slave site, we used a Java function that computed an antijoin operation between the two replicas. For the SRA approach set reconciliation techniques were adapted to the replica comparison.

Figure 1 shows several executions of the CTA and the SRA variants with the FS servers and the SN network. We can see that the computational time of the SRA variant depends on the maximum number of differences between any two replicas (d'). See also that with the SRA variant the computational time is not affected by the size of the replica since all three lines overlap in Figure 1, meaning the performance is similar in the three cases. However, see how the time increases as d' increases. Since the evaluations of the characteristic polynomials in each evaluation point is done asynchronously, this time is not considered in this experiment. On the other hand, the computational time of the CTA variant depends on the size of the tables. Since the sizes of the symmetrical differences in the experiment are marginal compared to the size of the tables, the computational times of each table is almost constant. Observe that the points where the lines that correspond to the SRA variant cross with the ones that correspond to the CTA variant, indicate the borderline values of d' where one variant turns to be better than the other. Specifically, we can see that for table $pklineitem$, for values above $2,000$ differences, the CTA variant turns to be better compared to the SRA variant.

Table I shows the computational and communication times of the CTA and the SRA variants computed considering several d' values. All the computations are done with replicas of table $pklineitem$ executed with the FS servers and the FN networks. The purpose is to show the specific elapsed times of several tasks. The first three columns refer to the specific tasks of the SRA variant. The columns show the time for the evaluation of the characteristic polynomials (CP), for the transfer of the evaluation point vector, (Com.), and for the interpolation of the reduced rational function to obtain the roots of the numerator and denominator (Roots). As for the CTA variant, the next two columns show the time to transfer table $D_c.T$ (Com.), and the time to evaluate the Java function. The CP time is shown even though these values may

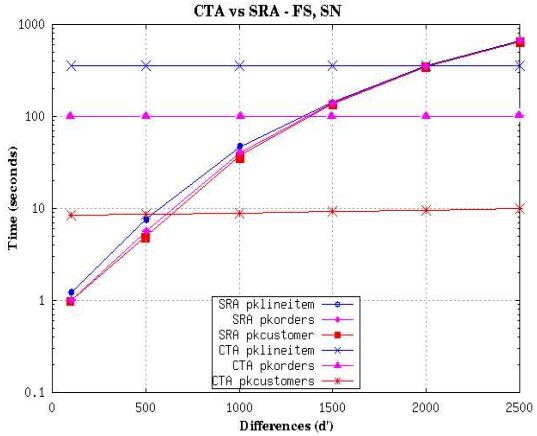

Fig. 1: Comparison between CTA variant vs. SRA variant.

TABLE I: Computational and communication times of CTA and SRA variants.

	SRA			CTA	
d'	CP	Com.	Roots	Com.	Java funct. (antijoin)
100	42.0	0.2	1.0	7.8	5.7
200	**77.8**	**0.2**	**1.5**	**8.1**	**6.0**
300	117.0	0.2	2.2	8.2	6.5
500	188.2	0.2	7.4	8.4	6.5
1,000	**374.0**	**0.2**	**46.2**	**8.6**	**6.8**
1,500	559.7	0.2	144.8	8.7	7.4
2,000	745.5	0.2	352.9	8.9	10.7
2,500	810.4	0.2	661.9	9.2	11.4

CP: characteristic polynomial; Roots: interpolation of reduced rational function; Java funct.: computation of Java function - antijoin; Com: communication

be computed asynchronously while updating the replica, not necessarily when the replica consistency is evaluated.

We highlighted the approximate value of d' where it is better to use one or the other approach (not all times are considered). The two lines that are highlighted indicate that for d' values below 200, the SRA variant is better than the CTA variant if the elapsed times to evaluate of the characteristic polynomials (CP) are considered. However, if this last time is not considered, then the approximate value of d' that bounds a better performance of the SRA variant is $1,000$. Contrast this result with the past experiment. We can see that if we have a faster network, the results change.

Contrast in Table I the communication times. Observe that the SRA variant has a much lower communication time. However, contrast the computational times. With the SRA variant, computing the roots of the rational function becomes very much expensive as d' grows.

V. Related Work

Replica consistency has received much attention in recent years. In [13] the authors propose two update propagation strategies that improve freshness, a concept that supposes that replica consistency in a distributed database can be relaxed. These strategies are based on immediate propagation, without waiting for the commitment of the update transaction in Master-Slave configurations. In [12] the authors propose a refreshment algorithm to maintain replica consistency in a lazy master replicated database based on specific properties of the topology of replica distribution across nodes. Both works propose strategies towards maintaining replica consistency in a database in a Master-Slave configuration. Our work supposes an a posteriori scenario where replica inconsistency violations are probably present and the user wants to measure the problem. Our work is oriented towards highlighting the benefits to use our methods in a distributed data warehouse, in a Master-Slave configuration. In [2] the authors propose two lazy update protocols that can be used in a distributed data warehouse, that guarantee serializability but require that the copy graph be a directed acyclic graph. The authors propose a solution to prevent the lazy replication inconsistency problems in a particular distributed configuration. In [10] the authors propose a new class of replication systems called TRAPP (Tradeoff in Replication Precision and Performance). Instead of storing stale exact values in the slave site, the system stores ranges that are guaranteed to bound the updated data values. Users give a quantitative precision constraint along with their query and the system selects a combination of locally cached bounds and current data stored in the master site. The system delivers a bounded answer consisting of a range that is no wider than the specified precision constraint that contain the precise answer and is computed as quickly as possible. In contrast, our system delivers imprecise values that meet the user precision constraints or exact precise values, where a low cost is paid in terms of data transmition.

In [4] the authors introduced a coherency index to measure replica consistency (coherency). They examine the trade off between consistency and performance, and show that in many situations a slight relaxation of coherency can increase performance. In our work, we focus on efficiently diagnosing replica consistency in a data warehouse scenario. In [16] and [17] the authors proposed several metrics to measure the quality of replicated services where the access to a replicated database is included. They show a middleware layer that enforces consistency bounds among replicas allowing applications to dynamically trade consistency for performance based on the current service, network, and request characteristics. They measure availability while varying the consistency level, the protocol used to enforce consistency, and the failure characteristics of the underlying network. However they measure the quality of the service (access) and not the quality of, as in our case, the replicated data. In [7] the authors propose an approach to repair a crashed site in a distributed data warehouse that uses data replication to tolerate machine failures. Their approach uses timestamps to determine which records need to be copied or updated. Our strategy is an on demand technique, that also queries sites but does not require timestamps. It measures the quality of the data warehouse when the user needs a diagnosis of the inconsistencies. It is based in the efficient computation of the symmetrical differences among replicas.

A related problem deals with detecting and extracting modi-

fications from information sources, in particular, detecting differences between snapshots of data. In [6] the authors propose several algorithms in order to solve this problem, known as the snapshot differential problem. The algorithms perform compression of records and one of them, the window algorithm, works specially well when the number of differences is small. However, the time complexity of the algorithms depends on the size of the snapshots. In particular, the window algorithm, needs to read the snapshots once. In contrast, our method using set reconciliation techniques based on the computation of the characteristic polynomials depends on the size of the symmetrical differences, considering that the computation of the evaluation points can be done asynchronously.

To close our discussion on related work, in [11] we proposed referential integrity metrics in distributed databases.

VI. Conclusions

We proposed techniques to efficiently measure replica consistency in distributed data warehouses. We identified research issues on evaluating replica consistency in a lazy master replication model. Specifically, we studied how to quickly compute the symmetrical difference between the master table and the slave replica, adapting set reconciliation techniques to a distributed database. Our techniques are applicable when the symmetric difference between the master table and a replica is small, since communication complexity is proportional to the number of different records; computational complexity is relatively high since it is cubic in the number of different records. However, in a distributed data warehouse where tables are large and the symmetric differences between replicas are expected to be small, our techniques represent a good alternative to measure replica consistency. Thus an OLAP tool can efficiently evaluate replica consistency at a remote slave site before evaluating a query as follows. If replica freshness is acceptable, the OLAP tool can continue with local query evaluation, avoiding a replica refresh computation, saving time and computer resources. Our experiments compared the set reconciliation strategy against the Complete Transfer Approach, which is based on full table scans, simulating distributed databases. In contrast to the set reconciliation approach, the Complete Transfer Approach is inefficient for large tables because both communication and time complexity are linearly dependent on the size of replicated tables.

There are several research issues for future work. We need to analyze time and communication complexity in more depth considering the relational database system operators used to compute the symmetric difference in SQL. We would like to develop new database algorithms to maintain consistency measures up to date, as new records are inserted. We need to develop a cost model that considers a non-uniform distributed database with computers running at different speeds and different network speeds between pairs of nodes. We plan to explore methods where the user gives thresholds related to the precision of aggregate functions over measure attributes, together with the size of the symmetrical difference between replicas. We would like to explore how set reconciliation techniques can solve other problems in distributed data warehouses, beyond measuring consistency.

Acknowledgments. The first author was sponsored by the UNAM information technology project "Macroproyecto de Tecnologías para la Universidad de la Información y la Computación". The second author was partially supported by NSF grants CCF 0937562 and IIS 0914861. We would like to thank Sergio Rajsbaum for an inspiring conversation that derived on an earlier version of this article. Also, we would like to thank Claudia Morales-Almonte and Rogelio Montero-Campos for programming several of our experiments.

References

[1] J. Albrecht and W. Lehner. On-line analytical processing in distributed data warehouses. In *IDEAS '98*, page 78. IEEE Computer Society, 1998.

[2] Y. Breitbart, R. Komondoor, R. Rastogi, S. Seshadri, and A. Silberschatz. Update propagation protocols for replicated databates. In *SIGMOD '99*, pages 97–108, 1999.

[3] M. Costa and H. Madeira. Handling big dimensions in distributed data warehouses using the DWS technique. In *DOLAP '04*, pages 31–37, 2004.

[4] R. Gallersdörfer and M. Nicola. Improving performance in replicated databases through relaxed coherency. In *VLDB '95*, pages 445–456, 1995.

[5] J. Gray, P. Helland, P. O'Neil, and D. Shasha. The dangers of replication and a solution. In *SIGMOD '96*, pages 173–182, 1996.

[6] W. Labio and H. Garcia-Molina. Efficient snapshot differential algorithms for data warehousing. In *VLDB '96*, pages 63–74, 1996.

[7] E. Lau and S. Madden. An integrated approach to recovery and high availability in an updatable, distributed data warehouse. In *VLDB '06*, pages 703–714, 2006.

[8] Y. Minsky, A. Trachtenberg, and R. Zippel. Set reconciliation with nearly optimal communication complexity. Technical report, Ithaca, NY, USA, 2000.

[9] Y. Minsky, A. Trachtenberg, and R. Zippel. Set reconciliation with nearly optimal communication complexity. *IEEE Transactions on Information Theory*, 49(9):2213– 2218, Sept. 2003.

[10] C. Olston and J. Widom. Offering a precision-performance tradeoff for aggregation queries over replicated data. In *VLDB*, pages 144–155, 2000.

[11] C. Ordonez, J. García-García, and Z. Chen. Measuring referential integrity in distributed databases. In *ACM CIMS*, pages 61–66, 2007.

[12] E. Pacitti, P. Minet, and E. Simon. Replica consistency in lazy master replicated databases. *Distrib. Parallel Databases*, 9(3):237–267, 2001.

[13] E. Pacitti and E. Simon. Update propagation strategies to improve freshness in lazy master replicated databases. *The VLDB Journal*, 8(3-4):305–318, 2000.

[14] D. Starobinski and A. Trachtenberg. Boston University Laboratory of Networking and Information Systems, http://ipsit.bu.edu/nislab/projects/cpisync /download.htm, 2008.

[15] TPC. *TPC-H Benchmark*. Transaction Processing Performance Council, http://www.tpc.org/tpch, 2005.

[16] H. Yu and A. Vahdat. Design and evaluation of a continuous consistency model for replicated services. In *OSDI'00*, pages 21–21, 2000.

[17] H. Yu and A. Vahdat. The costs and limits of availability for replicated services. *ACM Trans. Comput. Syst.*, 24(1):70–113, 2006.