THE ABORTION RATE OF LAZY REPLICATION PROTOCOLS FOR DISTRIBUTED DATABASES *

Luis Irún-Briz, Francesc D. Muñoz-Escoí, Josep M. Bernabéu-Aubán
Instituto Tecnológico de Informática
Universidad Politécnica de Valencia
46071 Valencia, SPAIN

Keywords: Replication, fault tolerance, reliability, distributed systems, databases, middleware, systems integration

Abstract: Lazy update protocols have proven to have an undesirable behavior due to their high abortion rate in scenarios with high degree of access conflicts. In this paper, we present the problem of the abortion rate in such protocols from an statistical point of view, in order to provide an expression that predicts the probability of an object to be out of date during the execution of a transaction.
It is also suggested a pseudo-optimistic technique that makes use of this expression to reduce the abortion rate caused by accesses to out of date objects.
The proposal is validated by means of simulations of the behavior of the expression. Finally, the application of the presented results to improve lazy update protocols is discussed, providing a technique to theoretically determine the boundaries of the improvement.

1 INTRODUCTION

Nowadays, many distributed (i.e., networked) applications have to manage large amounts of data. Despite the increasing ubiquitousness of information, the access patterns to distributed data often feature a noticeable degree of geographical locality. Moreover, many applications require a high degree of availability, in order to satisfy the need of offering services at any time, to clients that are either internal or external to the networked application. A predominance of locality of access patterns usually suggests a partitioning of the database (Gray et al., 1996). In many scenarios, it may be convenient or even necessary to replicate the information in a set of servers, each one attending its local clients. The different replicas must then be interconnected, a WAN being usually the best-fit architectural option.

The typical kind of applications we have in mind are, for instance, widely distributed databases in wide area intranets of medium and large enterprises, for data warehousing or resource management, as well as extranet service provisioning of such enterprises. Examples where such intranets of databases are deployed advantageously are enterprises with several branch offices (e.g., banks, chains of retailers, super- and hypermarkets), telecommunication providers, travel businesses, logistics, etc. Examples of extranet services which benefit from replication protocols as discussed in this paper are customer relationship management, e-banking, virtually all kinds of e-business as well as most e-government applications. Common to all of these applications is that potentially huge amounts of data are maintained and replicated on distributed sites, while access patterns (or at least significant contingents thereof) are highly local. Efficiency and high availability of such services is key to their acceptance and success.

The GlobData project (Instituto Tecnológico de Informática, 2002; Rodrigues et al., 2002) strives to provide a solution for the kinds of applications just outlined. It does so by defining a specific architecture for replicated databases, together with an API and a choice of consistency modes for data access. In the GlobData architecture, a number of consistency protocols can be implemented and plugged, each one providing particular guarantees and behaviors.

During the development of the project, we encountered that the protocols necessities of our target scenario (i.e. geographical distributed applications with a high degree of locality) could be well-fitted with a

*This work has been partially supported by the EU grant IST-1999-20997 and the Spanish grant TIC2003-09420-C02-01.

Irún-Briz L., D. Muñoz-Escoí F. and M. Bernabéu-Aubán J. (2004).
THE ABORTION RATE OF LAZY REPLICATION PROTOCOLS FOR DISTRIBUTED DATABASES.
In *Proceedings of the Sixth International Conference on Enterprise Information Systems*, pages 130-135
DOI: 10.5220/0002623301300135

family of replication protocols classified as *lazy update protocols*. However, we encountered that this kind of protocols introduces a high number of abortions in the system, because they propagate the updates beyond the commit phase (in contrast to *eager update protocols*, where the whole system is updated inside the commit phase), making it possible for a transaction to read out of date data, and thus, there are consistency violations.

In (Muñoz et al., 2002; Irún-Briz, 2003), we presented two consistency protocols, using different approaches for the update propagation.

Lazy update protocols have proven to have a critical inconvenience in contrast to eager approaches: the dramatical increase of the abortion rate in scenarios with a high degree of access conflicts. This inconvenience makes unusable the traditional lazy update protocols in certain scenarios, because an unacceptable number of started transactions will terminate with an undesirable abort.

To understand the problem, this work presents a set of expressions describing the abortion rate. In the presentation, we model a complete system including nodes, the sessions executed, and the objects accessed by the sessions. Moreover, we use our LOMP algorithm as an example of a lazy update protocol where all these expressions may be directly applied. A full description of this protocol may be found in Section 3.2 of (Irún-Briz, 2003).

Section 2 includes the description of such modeled system, in order to formalize an analysis of the abortion rate in section 3. In section 4, an empirical validation of the model is presented, and section 5 will provide a theoretical analysis of an improvement of the lazy approach. Finally, section 6 describes some related work, and section 7 includes some conclusions about the applicability of the expressions.

2 THE MODELED SYSTEM

One of the problems solved by the GlobData architecture(Muñoz et al., 2002) (called COPLA) is to ensure the consistency among transactions being executed in different database replicas. Multiple implementations of the consistency management are possible, thus allowing that a GlobData system could change its consistency protocol at will. So, we use the COPLA architecture as a platform to experiment with different consistency algorithms working on the top of a transactional system (in fact, a database) and replicated over a distributed system.

The target system of our analysis follows a number of considerations, designed to configure a scenario as close as possible to a general environment that, although simplified, is able to fit the requirements of the kind of environment we are centered in. This environment was described in the introduction, and has considerations about client applications, system load, pattern of accesses, interconnection network, etc.

In summary, these adopted assumptions are the following:

- There are K COPLA managers running in the system. Each one can be considered as a "node" $N_{k=1..K}$.
- Each node in the system manages a complete replica of the database. This database contains N objects.
- We assume that the objects stored in the database have a identifier that is unique in the entire system. In addition, each write operation performed in a node includes an storage for each written object of the local time the operation is performed, as well as a version number of the written object.

3 PROBABILITY OF ABORTION

We can define the probability for a session S to be aborted as: $PA(S) = 1 - (PC_{conc}(S) \cdot PC_{outd}(S))$ where:

- $PC_{conc}(S)$ is the probability that the session concludes without concurrency conflicts.
- $PC_{outd}(S)$ is the probability that the session concludes without accessing to outdated objects.

The goal of this section is to determine the value of $PC_{outd}(S)$, in order to predict the influence our LOMP has into the abortion rate in the system. To this end, we can calculate this probability in terms of the probability of a session to conclude with conflicts produced by the access to outdated objects ($PA_{outd}(S)$):

$$PC_{outd}(S) = PC_{outd}(r_i)^{nr}$$

taking nr as the number (in mean) of objects read by a session,

$$PC_{outd}(S) = PC_{outd}(r_i)^{\frac{\sum_k nr_k}{K}} \quad (1)$$

moreover, $PC_{outd}(r_i)$ is the probability for an object r_i to have an updated version in the instant the session accesses it. This probability can be expressed in terms of the probability for an object to be accessed in an outdated version ($PA_{outd}(r_i)$) as:

$$PC_{outd}(r_i) = 1 - PA_{outd}(r_i)$$

now, let's see the causes of these conflicts: we took r_i as an asynchronous object in the active node that has not been updated since $ut(r_i)$; the outdated time for r_i satisfies $\delta(r_i) = t(r_i) - ut(r_i)$; it can be seen that $PA_{outd}(r_i)$ depends on the number of sessions that

write r_i having the chance to commit during $\delta(r_i)$. Let PC_{T,r_i} be the probability for another concurrent session T (that has success in its commit) to finalize with $r_i \notin W(T)$. Then,

$$PA_{outd}(r_i) = 1 - (PC_{T,r_i})^C$$

where C depends on the number of write-sessions that can be committed in the system during $\delta(r_i)$ ...

$$PA_{outd}(r_i) = 1 - (PC_{T,r_i})^{\sum_k wtps_k \times \delta(r_i)} \quad (2)$$

now, we can reformulate PC_{T,r_i} as $PC_{T,r_i} = P[r_i \notin W(T)]$ and, considering $W(T) = \{w_1, w_2, \dots w_{nw(T)}\}$, then in mean, it will be satisfied that:

$$PC_{T,r_i} = \left(P[r_i \neq w_{j\in\{1..nw\}}]\right)^{nw}$$

taking nw as the mean of $|W(T)|$ for every write-session in the system.

$$PC_{T,r_i} = \left(P[r_i \neq w_{j\in\{1..nw\}}]\right)^{\frac{\sum_k nw_k}{K}} \quad (3)$$

The next step consists of the calculation of $P[r_i \neq w_{j\in\{1..nw\}}]$. To do this, we must observe the number of objects in the database (N). The probability that an accessed object is a given one is $\frac{1}{N}$, thus: $P[r_i \neq w_{j\in\{1..nw\}}] = 1 - \frac{1}{N}$

Finally, the complete expression can be rewritten as follows:

$$PA_{outd}(r_i) = 1 - \left(1 - \tfrac{1}{N}\right)^{\frac{\sum_k nw_k}{K} \times \sum_k wtps_k \times \delta(r_i)} \quad (4)$$

This expression provides a basic calculation of the probability for an object access to cause the abortion of the session by an out-of-date access.

The expression can be calculated with a few parameters. Only nw_k and $wtps_k$ must be collected in the nodes of the system in order to obtain the expression. Thus, it becomes possible for a node to estimate the convenience for an object to be locally updated before being accessed by a session. This estimation will be performed with a certain degree of accuracy, depending on the "freshness" of the values of nw_k and $wtps_k$ the node has. The way the expression can be used, and an adequate mechanism for the propagation of these parameters has been presented in (Irún-Briz, 2003).

4 EXPERIMENTAL VALIDATION OF THE MODEL

We have validated the algorithm presented above by implementing a simulation of the system. In this simulation, we have implemented nodes that concurrently serve sessions, accessing to different objects of a distributed database. We have also modeled the concurrency control, and the lazy update propagation used by LOMP.

4.1 Assumptions

The assumptions for the implementation of the simulation (Chandy and Misra, 1979; Bagrodia et al., 1987) are compatible with the ones taken for the model calculation, and the values have been established to increase the number of conflicts produced by the transactions executed in the system (i.e., this configuration shows a "worst-case" scenario for our system):

- There are 4 nodes in the system, each holding a full replica of the database, that contains 20 objects. Each node executes transactions, accessing the database.
- For every object, a local replica holds the value of the object, and the version corresponding to the last modification of the object. That is, the only synchronous replica for each object is its owner node.
- There are three kinds of transactions, with a probability to appear of 0.2, 0.4, and 0.4 respectively: "Type 0" (read-only): reads three objects; "Type 1" (read-write): reads three objects, then writes these three objects; "Type 2" (read&read-write): reads six objects, then writes three of the objects read.
- The model supports the locality of the access by means of the probability for an accessed object to be owned by that node (i.e. the node where the transaction is started). For read-only transactions, this is 1/4, (this models no locality for read-only transactions). For "type 1" and" type 2" transactions, the probability is 3/4.
- The simulation time has been set at 1,422 t.u., discarding the first 2 t.u. as stabilization time for the simulation. This allows to start up to 60,000 transactions.

4.2 Accuracy of the Prediction

When the expression exceeds the established threshold for an object, and an update request is sent, it is possible for the response of this request to contain the same version for the requested object (e.g. when the object was, in fact, up to date). We name this situation "Inaccurate prediction".

The more accurate the predictions are, the less overhead the algorithm introduces in the system. This accuracy of the predictions will be given by the set threshold: higher values for the threshold should provide more accurate predictions. In general, we observed that higher values for the threshold increase the accuracy of the prediction.

The evolution of the inaccuracy in respect to the amount of LocalUpdate messages is shown in the figure 1. The optimum line is also shown, and corresponds with the diagonal. The more accurate the prediction is, the closest the curves are. The studied im-

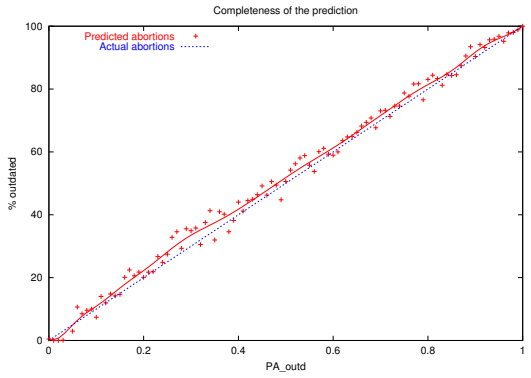

Figure 1: Evolution of the inaccuracy for different P[update]

plementation differs from the ideal line with a lower bound pattern, and it is shown that is quite proximal to the ideal.

5 THEORETICAL BOUNDARY OF AN IMPROVEMENT

The first sections of this paper has been dedicated to the study of an statistical expression determining the probability for a particular object access, to obtain an out to date value, thus causing the abortion of the requesting transaction.

We can make use of such expression, in order to determine the achievable improvement for a transaction, in terms of abortion rate, when the average outdate time of the objects is decreased.

Unfortunately, this decrement in the outdate time will cause a degradation in the service time of the executed transactions, and this must also be taken into account.

In this section, we present a theoretical boundary for the reduction of the abortion rate that an adequate exploit of the expression can provide.

5.1 Preliminaries

The used expression for the probability of an access to be an stale-access was presented in 4. In the expression, the elapsed time between two consecutive updates of the object r_i is expressed as $\delta(r_i)$.

We can perform a serial analysis in order to determine the mean value for $PA_{outd}(r_i)$. The analysis can be easily performed applying differentiate calculus to the expression. The obtained expression is showed to be:

$$PA_{outd} = 1 - \left(1 - \frac{1}{N}\right)^{\frac{\sum_k nw_k}{K} \times \sum_k wtps_k \times \delta} \quad (5)$$

where δ is the mean value for the $\delta(r_i)$ in a system execution, and PA_{outd} is the mean value for the $PA_{outd}(r_i)$ in the same system execution.

Now, we can obtain $PC_{outd} = 1 - PA_{outd}$, being:

$$PC_{outd} = PC = \left(1 - \frac{1}{N}\right)^{nwwt \times wtps \times \delta} \quad (6)$$

5.2 Average Outdate Time

To simplify, let's suppose that transactions are distributed homogeneously along the system history.

Imagine the system execution history as a line, where a number of transactions are sequentially executed. For a certain object o_i, the probability for a executed transaction to read o_i will be $\frac{nr}{N}$, where nr is the number, in mean, of objects read by any executed transaction (either read or write transactions are included here), and N is the number of objects in the database.

The probability of the object o_i to be updated by a lazy replication protocol depends on the probability for a transaction that read o_i to be aborted by an stale-access (i.e. $PA_T = PA^{nr}$). Thus the probability for an object to be updated by a generic transaction will be:

$$PA^{nr} \times \frac{nr}{N}$$

Now, let tps be the number of transactions executed in the system per second. Thus, there will be $up(o_i) = tps \times PA^{nr} \times \frac{nr}{N}$ updates of o_i per second. Finally, we can express $\delta(o_i)$ as $\frac{1}{up(o_i)}$, and, in mean:

$$\delta = \frac{1}{tps \times PA^{nr} \times \frac{nr}{N}} \quad (7)$$

If the accessed objects are updated along the transaction, the value for δ will be decreased proportionally to the amount of updates performed during the transaction execution.

To model this, a simple approach can be expressed with the following expression:

$$\delta' = PC \times d'_T + (1 - PC) \times \delta \quad (8)$$

where d'_T is the duration of a transaction when the updates are performed along its execution. For the aborted transactions, (i.e. $(1 - PC)$), the mean outdate time is unchanged (δ). In contrast, for committed transactions, the new outdate time is decreased to d'_T (i.e. the duration of the transaction).

Now, the duration of a transaction when the updates are performed will depend on the number of requested accesses that are actually updated along the transaction execution ($nr \times P_{UPD}$), and the cost of each of these updates (K_{UPD}). Note that P_{UPD} is the probability for a requested object to be updated. Thus, if $P_{UPD} = \frac{1}{2}$, there will be forced to be updated the half of the objects requested by a transaction.

In summary, the expression for d'_T can be composed by:

$$d'_T = d_T + nr \times P_{UPD} \times K_{UPD} \quad (9)$$

Replacing 9 in 8, the new outdate time will follow the expression:

$$\delta' = \delta \times (1 - PC) + PC \times d'_T \quad (10)$$

This result will be useful in section 5.3, where the achievable abortion rate is specified in terms of δ'.

5.3 Abortion Rate

In mean, we can say that the achievable commit rate will be, observing equation 6:

$$PC' = \left(1 - \frac{1}{N}\right)^{nwwt \times wtps \times \delta'} \quad (11)$$

Replacing δ' in the expression, we obtain:

$$PC' = \begin{cases} \left(1 - \frac{1}{N}\right)^{nwwt \times wtps \times \delta \times (1-PC)} \\ \times \\ \left(1 - \frac{1}{N}\right)^{nwwt \times wtps \times PC \times d'_T} \end{cases} \quad (12)$$

That can be rewritten as:

$$PC' = PC^{1+PC\left(\frac{d'_T}{\delta} - 1\right)} \quad (13)$$

Now, we can replace δ with the expression obtained in (7), the resulting expression is:

$$\frac{PC'}{PC} = PC^{PC\left(d'_T \times (tps \times (1-PC)^{nr} \times \frac{nr}{N}) - 1\right)} \quad (14)$$

From the equation 14 we obtain that the improvement of the probability for an object to be accessed in an adequate way (i.e. not an stale access), is determined by $\frac{PC'}{PC}$, and it will be benefitted from the decrease of the established value or any of the following expressions:

- d_T, the duration of the transactions.
- $nr \times P_{UPD} \times K_{UPD}$, the number of updated objects, and the computational cost of each of these updates.
- tps, the amount of committed transactions per second (including both read-only and read-write transactions).
- $\frac{nr}{N}$, the relation between the amount of objects accessed per transaction (read-only or read-write transactions) and the total number of objects contained in the database.

To simplify the expression 13, we can denote as Δ to the existing relation between d'_T and δ (i.e. $\Delta = \frac{d'_T}{\delta}$). The resulting expression is:

$$PC' = PC^{1+PC\times(\Delta-1)} \Longrightarrow \frac{PC'}{PC} = PC^{PC\times(\Delta-1)} \quad (15)$$

Let's see an example for the improvement achievable in an extremely simple system, where $nr = 1$. In such system, we can establish as a parameter the probability for a requested object to be previously updated (i.e. P_{UPD}), and then study the achieved improvement for different values of Δ (and, consequently, different computational overheads).

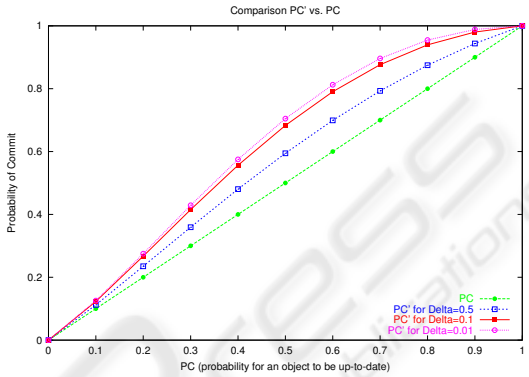

Figure 2: Evolution of the improvement varying Δ

Figure 2 shows the commit probability for a session accessing **only one object**. There is shown how the commit probability can be improved, when the update-time is decreased to the half, up to the 120% of the original commit time. When the update-time is decreased ten times, the improvement reaches the 140%. Lower values of Δ provides marginal improvements, at a higher computational costs.

When more than just an object is accessed in a session, the results shows higher differences between the commit promise, due to the factorization of such probabilities.

These results points to the convenience, in the scenarios fitting the parameters described above, to apply the techniques postulated by the presented discussion.

6 RELATED WORK

Current work in consistency protocols for replicated databases can be found using either eager (Agrawal et al., 1997) and lazy protocols (Breitbart and Korth, 1997; Holliday et al., 1999) using either optimistic and pessimistic consistency control.

Each one has its pros and cons, as described in (Gray et al., 1996). Eager protocols usually hamper the update performance and increase transaction response times but, on the positive side, they can yield serializable execution of multiple transactions without requiring too much effort. On the other hand, lazy protocols may allow a transaction to read outdated versions of objects, hamper the abortion rate, but they can improve transaction response times.

Pessimistic consistency control for distributed databases (Bernstein et al., 1987) is based on the principle of "locks" in order to avoid concurrent transactions to access to the same object in an inadequate mode. The use of "locks" minimizes the number of aborted transactions, but degrades the performance of the system, because the complexity introduced by the management of the locks.

On the other hand, the traditional approach for optimistic consistency control was presented in (Kung and Robinson, 1981), and its main advantage consists on the reduction of the blocking time of the transactions, using "versions" (or "timestamps") as the basis for its implementation. The main disadvantage of optimistic consistency protocols consists of the increase in the abortion rate.

The presented approach uses a optimistic consistency control, with lazy replication, and has been implemented in a real environment (COPLA).

7 CONCLUSIONS

Lazy update protocols have not been widely exploited due to its excessive abortion rate on scenarios with high probability of access conflicts. Nevertheless, such protocols can provide important improvements in the performance of a distributed system, when the abortion rate can be kept low, and the locality of the accesses is appreciable.

We have presented an statistical study of the abortion rate (as disadvantage of lazy protocols), in order to provide an expression for the probability for an accessed object to be out of date ($PA_{outd}(o_i)$), and cause a further abortion of the accessing transaction.

An statistical analysis has been performed in this paper, in order to provide an expression for the probability for a requested access to obtain a stale value of the required object.

The application of the expression has been also discussed, in order to determine the convenience, using a general algorithm, to update along the execution of a transaction the objects predicted to be stale. This discussion has provided a set of conditions, in base to a number of parameters, where these generic algorithms can improve the abortion rate of a lazy update protocol.

Consequently, the improvement has also studied, in base to the established decrement of the update-time of the accessed objects, giving as conclusion that such reductions may considerably improve the probability for an object to be updated, thus reducing dramatically the abortion rate of lazy update protocols.

Thus, this work theoretically validates the implementation of not-so-lazy update protocols based on statistical prediction of stale accesses to reduce the abortion rate. These protocols conform possible implementations of the mentioned generic algorithm, as a real improved update algorithm.

REFERENCES

Agrawal, D., Alonso, G., El Abbadi, A., and Stanoi, I. (1997). Exploiting atomic broadcast in replicated databases. *Lecture Notes in Computer Science*, 1300:496–503.

Bagrodia, R. L., Chandy, K. M., and Misra, J. (1987). A message-based approach to discrete-event simulation. *IEEE Transactions on Software Engineering*, SE-13(6).

Bernstein, P. A., Hadzilacos, V., and Goodman, N. (1987). *Concurrency Control and Recovery in Database Systems*. Addison Wesley, Reading, MA, EE.UU.

Breitbart, Y. and Korth, H. F. (1997). Replication and consistency: being lazy helps sometimes. In *Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of Database Systems*, pages 173–184. ACM Press.

Chandy, K. M. and Misra, J. (1979). Distributed simulation: A case study in design and verification of distributed programs. *IEEE Transactions on Software Engineering*, SE-5(5):440–452.

Gray, J., Helland, P., O'Neil, P., and Shasha, D. (1996). The dangers of replication and a solution. In *Proc. of the 1996 ACM SIGMOD International Conference on Management of Data*, pages 173–182, Canada.

Holliday, J., Agrawal, D., and Abbadi, A. E. (1999). Database replication: If you must be lazy, be consistent. In *Proceedings of* 18^{th} *Symposium on Reliable Distributed Systems SRDS'99*, pages 304–305. IEEE Computer Society Press.

Instituto Tecnológico de Informática (2002). GlobData Web Site. Accessible in URL: *http://globdata.iti.es*.

Irún-Briz, L. (2003). *Implementable Models for Replicated and Fault-Tolerant Geographically Distributed Databases. Consistency Management for GlobData.* PhD thesis, Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, Valencia, Spain. Available at *http://www.iti.upv.es/~lirun*.

Kung, H. T. and Robinson, J. T. (1981). On optimistic methods for concurrency control. *ACM Transactions on Database Systems*, 6(2):213–226.

Muñoz, F., Irún, L., Galdámez, P., Bernabéu, J., Bataller, J., and Bañuls, M.-C. (2002). Globdata: A platform for supporting multiple consistency modes. *Information Systems and Databases (ISDB'02)*, pages 244–249.

Rodrigues, L., Miranda, H., Almeida, R., Martins, J., and Vicente, P. (2002). The globdata fault-tolerant replicated distributed object database. In *Proceedings of the First Eurasian Conference on Advances in Information and Communication Technology, Teheran, Iran*.