Simplified Performance Models of the Reordering Issue in Timestamp Ordering Concurrency Control in Distributed Databases

by

Christos J. Bouras[1] and Paul G. Spirakis[1,2].

Σ2

1. Computer Technology Institute, Greece
   and Department of Computer Science and Engineering
   P.O. Box 1122, 261 10 Patras, Greece
2. Courant Institute of Mathematical Sciences, U.S.A.

16

Abstract

One important problem in distributed database research is that of Concurrency Control. Although many timestamp-based algorithms have been proposed to protect the distributed database from incosistency during concurrent accesses, only a few analytic models have been developed to compare their performance. This paper concentrates on modelling the essential features of the conflicts between transactions that arise due to the distributed environment which provides the reordering phenomenon. As an example, we study the basic and the conservative timestamp ordering algorithm and determine the probability of restarts and other interesting performance measures.

1. Introduction

Database concurrency control is concerned with the problems that arise when several users access and update a database simultaneously. Concurrency control techniques try to maintain the consistency of the database. Since the semantics of transactions are embedded in application programs, it is not easy to design concurrency control methods that take advantage of semantic knowledge. Most techniques therefore try to preserve (syntactic) serializability among accesses. A sequence of interleaved transactions is serializable if it produces the same effect on the database as a serial execution of those same transactions. Since a serial execution preserves consistency, a serializable execution also preserves consistency.

In a distributed system, one way to serialize accesses is based on using unique transaction identifiers for determining the order of execution of transactions. These are called timestamps and numerous timestamp algorithms have been proposed, all guaranteeing serializability (e.g. [8], [9], [12], [13], and a nice survey of some of these techniques can be found in [3]. The comparison of the performance of these techniques is a challenging issues due to the amount of detail that must be taken into account in realistic situations and also due to the mathematical difficulties of the analysis, which must consider the distortion of the relative order of events due to the distributed environment. In paper [10] developed approximation techniques to analyse timestamp-ordering in fully redundant databases but ignored the reordering issue.

* This research was partially sponsored by the Ministry of Education of Greece, and by the EEC RACE project TELEMED.

This phenomenon was studied in [2], [5] and [6] but all these works assumed total ordering. This assumption is not realistic in distributed databases. The partial ordering which analyzed in [11] is more convenient for distributed databases. Papers [7] and [14] developed a performance model of timestamp- ordering concurrency control which takes many aspects of the problem into account. Our work is motivated from works [7], [11] and [14]. We believe that our approach is simpler and slightly more general.

This paper concentrates on the analysis of the conflicts between transactions that arise in a distributed database. Its emphasis is on simplicity. We focus on the few important events that affect the conflict phenomenon and develop a mathematical model to access their relative significance.

2. Basic architecture of distributed databases

A database is a collection of shared data objects. In a database, certain relationships hold among its data objects. The set of these relationships for a database is called the consistency assertions of the database. A database is in ,a consistent state if the current values of its data objects satisfy all of its consistency assertions. In such a system, a transaction is a program with read, write and other operations. A distributed database management system may be viewed as a collection of nodes connected to a network. Each node consists of a computer running either a Transaction Manager (TM) or a Data Manager (DM) or both. The nodes communicate by sending messages over the network. The network is assumed to be completely reliable, i.e., if node A sends a message to node B, it is guaranteed that B will receive the message error-free. The architectute of the system is shown in Fig. 1a. A Transaction Manager coordinates the execution of the transaction. A Data Manager manages a local database. From the viewpoint of a single transaction, the system consists of a single TM and a number of DMs, Fig. 1b. Neither TMs nor DMs intercommunicate.

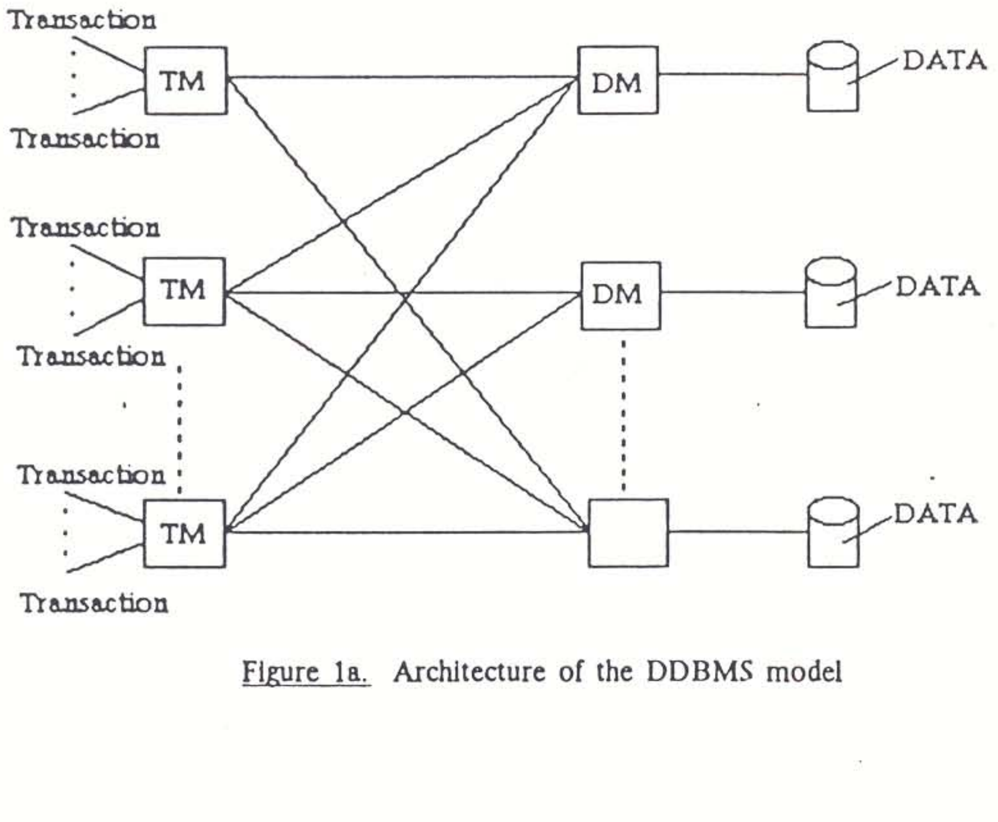

Figure 1a. Architecture of the DDBMS model

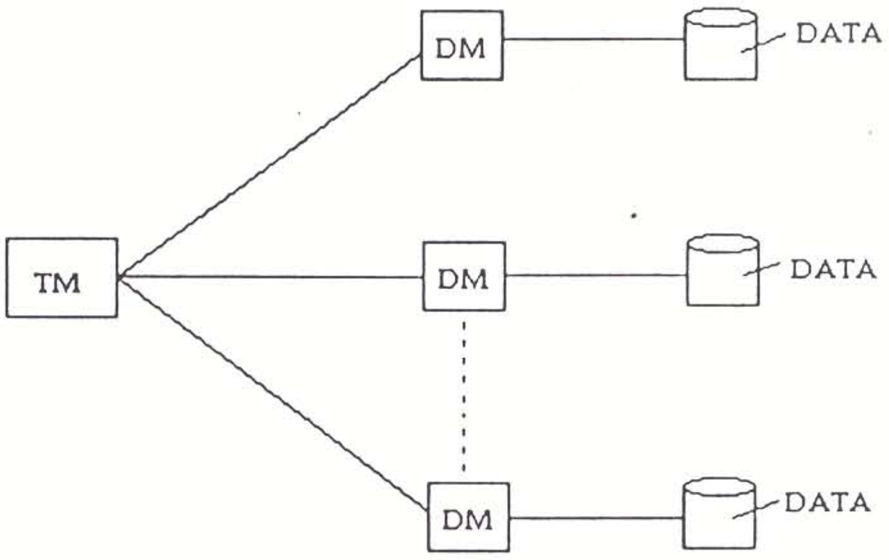

Figure 1b. DDBMS model for a single transaction

There are several reasons why distributed databases are developed. Many organizations are decentralized and a distributed database approach fits more naturally the structure of the organization. Distributed databases are the natural solution when several databases already exist in an organization and the necessity of performing global applications arises. The recent development of small computers, providing (at a lower cost) may of the capabilities which were previously provided by large mainframes, constitutes the necessary hardware support for the development of distributed information systems. The technology of distributed databases is based on two other technologies which have developed a sufficiently solid foundation during the last years computer networks technology and databases technology.

3. Performance model

Our model of distributed database is a collection of K sites interconnected by a network. Network connections are assumed to be perfectly reliable. We assume that the distributed database is full data replication. This means that the number of copies is equal to number of nodes and that at each site there is the same database. The number of data objects at each site is equal to N.

4. The essentials of the conflict model

4a. Causes

It is known from [3] and [4] that
(a) Two or more transactions may access the same data object and at least one of the attempts to write, and
(b) Intersite transmission delays may cause requests originating from different sites to arrive in reverse of their timestamps.
Any timestamp ordering algorithm must reorder these requests and process them in timestamp order.

4b. The issue of pipelined delays

The requests originating from the same site are assumed to be pipelined i.e. they will arrive at the same site in the order of their timestamps. This can be ensured by communication network protocols. The pipelined transmission requirement introduces resequencing delays. The distribution of such delays was analyzed, for example, in [1], [2], [6] and [11], mostly by modelling the network connecting two sites as an M|M|∞ or an M|6|∞ or an M|M|k queueing system. Even in such a case, the PDF of the pipelined transmission delay does not have a convenient form. In our analysis, the PDF of this delay is an input parameter.

4c. Static Conflicts

It is obvious that causes (a), (b) of the conflict phenomenon are independent. Cause (a) is a prerequisite for conflict. Without it cause (b) would not matter. The percentage of potential conflicts (static conflicts) due to cause (a) can be estimated by combinatorial techniques based on assumptions about the creation of readsets and writesets of transactions.

Let each transaction access M data objects (out of N), selected uniformly. Then the probability of two transactions having at least one common data object is

$$\Phi = 1 - \frac{\binom{N-M}{M}}{\binom{N}{M}} \tag{1}$$

If we assume that M>>N, which is usually the case in practice, then the (1) gives

$$\Phi = \frac{M^2}{N} \tag{2}$$

Furthermore, if the fraction of READ operations per transaction is a $(0 \leq a \leq 1)$ and the fraction of WRITE operations is b $(0 \leq b \leq 1)$ and the reads and writes are uniformly spread out among the M data objects, then the percentage of static conflicts (probability of static conflict) is

$$f = (2ab + b^2) \cdot \Phi . \tag{2a}$$

if Thomas's rule is ignored and

$$f = (2ab)\, \Phi \tag{2b}$$

otherwise.

4d. The phenomenon of order reserves

Cause (b) of conflicts, i.e. the possibility that requests originating from different sites may arrive in reverse order of their timestamps, is mainly due to the distributed nature of the database. A detailed modeling of this phenqmenon would have to take into account the time intervals shown in Fig. 2:

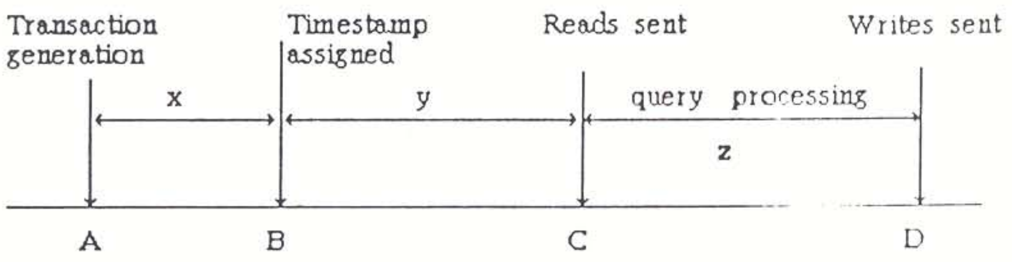

Figure 2: Time Ordering of events related to Local Transaction Processing

Times x,y (and z in most cases) are insignificant compared to the network delays. Thus, a simplified look at the phenomenon of order reverses may assume all those times to be negligible. (An analysis with x=y=0 and z exponential of known mean was conducted in [7]).

Hence, the order-reverse is essentially indicated in Fig. 3, for two transactions, T_i and T_j.

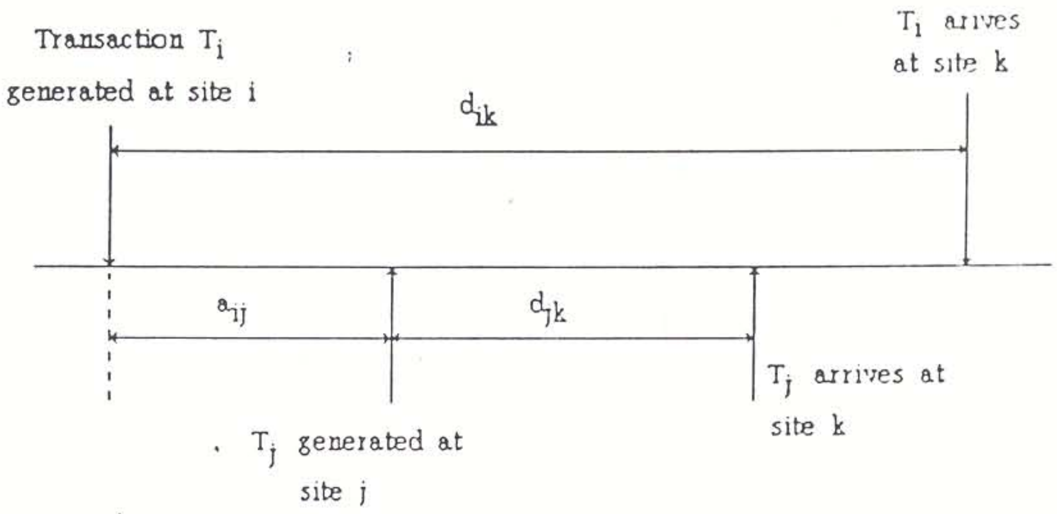

Figure 3: Order reverse

If

a_{ij} = the time interval between the generations of T_i at site i and T_j at site j

and

d_{ik}, d_{jk} = network delays (transmission plus pipelining) transactions T_i, T_j to site k

then the order-reverse is represented by the inequality:

(3) $a_{ij} + d_{jk} < d_{ik}$

Let R^k_{ij} = probability of the above inequality happening (i.e. the percentage of cases of order-reverse).

The time difference a_{ij} depends on the absolute times of generation of T_i and T_j. Let T_i be the nth transaction generated at site i (since time zero) and T_j the mth at site j. Let the corresponding generation times be $t_{i,n}$ and $t_{j,m}$.
Then

$$a_{ij} = t_{j,m} - t_{i,n} \tag{4}$$

Order reverse can happen only when m=n. In such case $a_{ij} = t_{j,n}-t_{i,n}$ and this may be a short, fixed, interval.

Let $A_{ij}(x)$ be the PDF of a_{ij} (with density $a_{ij}(x)dx=\text{prob}(a_{ij} \in (x, x+dx))$.
Let $D_{ik}(\ )$, $D_{jk}(\ )$ and $d_{ik}(\ )$, $d_{jk}(\ )$ be the PDFs and pdfs of the r.v.'s d_{ik}, d_{jk}.
From the architecture of the system, d_{ik}, d_{jk} are independent r.v.'s. Then

$$R^k_{ij} = \int_0^\infty \text{prob}\big(a_{ij} \in (x,x+dx)\big)\, dx \cdot \text{prob}\big(x+d_{ik} < d_{jk}\big)$$

where

$$\text{prob}\big(x+d_{ik} < d_{jk}\big) = \int_0^\infty \text{prob}\big(d_{ik} \in (y,y+dy)\big)\cdot dy \cdot \text{prob}\big(d_{jk} > x+y\big)$$

$$R^k_{ij} = \int_0^\infty a_{ij}(x)dx \int_0^\infty d_{ik}(y)dy \cdot \big(1-D_{jk}(x+y)\big)$$

The above yields that

$$R^k_{ij} = 1 - \int_0^\infty a_{ij}(x)dx \int_0^\infty d_{jk}(y)\cdot D_{ik}(x+y)dy \tag{5}$$

If we assume that transactions are generated at any site as independent Poisson processes of rate λ, the same for each site and the network delay for each transaction is exponentially distributed of rate μ, then

$$R^k_{ij} = 1 - \int_0^\infty \lambda\, e^{-\lambda x}\, dx \int_0^\infty \mu\, e^{-\mu y}\, dy \cdot \big(1-e^{-\mu(x+y)}\big)$$

After some algebra the above expression can be written:

$$R^k_{ij} = \frac{\lambda}{2(\lambda+\mu)} \tag{6}$$

The above probability under our assumptions is the same for each pair of transactions at any site. From now on we will call it R.

5. Performance measures

5a. Basic Timestamp Ordering

The simplest timestamp ordering technique, basic timestamp ordering, employs only transaction rejects and restarts in order to maintain serializability. Details can be found in [3] and [4].

The probability that a transaction is rejected if only two sites are present, is clearly

r = prob(rejection probability in case of two sites) =
= prob(to have static conflict) · prob(out of order at one site at least) =

$$f \cdot \frac{\lambda}{2(\lambda+\mu)} \qquad (7)$$

In the case of K TM sites, our performance model, for each transaction to be rejected it is enough to conflict and be in reverse order with at least one transaction of another site, generated later than this transaction. This, the total rejection probability P_r, is (assuming size independence)

$$P_r = 1 - (1-r)^{K-1} \qquad (8)$$

If Thoma's rule is ignored, this probability is

$$P_r = 1 - \left[1-\left(2ab+b^2\right)\frac{N^2}{M}\frac{\lambda}{2(\lambda+\mu)}\right]^{K 1} \qquad (8a)$$

and

$$P_r = 1 - \left[1-(2ab)\ \frac{N^2}{M}\frac{\lambda}{2(\lambda+\mu)}\right]^{K 1} \qquad (8b)$$

otherwise.

5b. Conservative Timestamp Ordering

In this technique, a transaction which contains READ arriving at any site has to wait until all transactions which contain WRITE to same data objects with it and have smaller timestamps, arrive and be processed. Also, in the case which a transaction which contains WRITE arriving at any site has to wait until all transactions which contain READ to same data objects with it and have smaller timestamps arrive and are processed.

In the context of our simplified model, we have that the probability of waiting each transaction, in any site is

$$P_w = 1 - (1-r)^{K-1} \qquad (9)$$

and the waiting time of each transaction in any site is

$$W(j,k) = \max_i \left(d_k - \left(a_i + d_k\right)\right) \qquad (10)$$

(gives that $d_{ik} \geq d_{ij} + d_{jk}$)

Under our assumptions about Poisson arrivals and exponential delays, we have that the probability of $d_{ik}-(\alpha_{ij} + d_{jk})=z\geq 0$ is:

$$\text{Prob}\{d_{ik}-(\alpha_{ij}+d_{jk}) = z\geq 0\} =$$

$$= \text{Prob}\{z \leq d_{ik}-(\alpha_{ij}+d_{jk}) \leq z+dz\} =$$

$$= dz \int_0^\infty \lambda e^{\lambda x}\, dx \int_0^\infty \mu e^{-\mu y}\, dy \left(\mu e^{-\mu(z+x+y)}\right) = \frac{\mu\lambda}{2(\lambda+\mu)}\, e^{-\mu z}\, dz$$

Now, $\text{Prob}\{W(j,k) \leq w\} = \text{Prob}\{\text{for all } i\neq j.\ d_{ik}-(\alpha_{ij}+d_{jk}) \leq w\}$

So, due independence,

$$\text{Prob}\{W\leq w\} = \frac{\lambda}{(2\mu)^{K-2}\,(\lambda+\mu)^{K-1}}\left(1-e^{-\mu w}\right)^{K-1} \qquad (11)$$

From the derived distribution of the waiting time of a transaction per site, we may derive the distribution of the total waiting time of the transaction in the database, since the total waiting time is the maximum, over all sites K of the $d_{jk}+W(j,k)$. Also it is easy to derive the total number of waiting transactions, using Little's law. All these formulas are cumbersome but easy to derive.

6. Corollary & further work

Our simplified conflict model allows a tractable mathematical analysis of timestamp-ordering concurrency control techniques, while, at the same time, the model focuses on the essential factors of the conflict phenomenon. We conjecture that the Poisson and exponential assumptions of the examples can be replaced by testable operational conditions without altering the analytical results.

References

[1] Agrawal S., Ramaswamy R., "Analysis of the Resequencing Delay for M|M|∞ systems", ACM SIGMETRICS 1987, pp. 27-35.

[2] Baccelli F., Gelenbe E., Plateau B., "An end to end approach to the resequencing problem", JACM Vol. 31, No. 3, July 1984.

[3] Bernstein P., Goodman N., "Concurrency Control in Distributed Database Systems", Computing Survey, Vol. 13, No. 2, June 1981.

[4] Ceri S., Pelagatti G., "Distributed databases. Principles and Systems", McGraw-Hill, 1984.

[5] Kamoun F., Ben Djerad M., LeLann G., "Queueing Analysis of the Ordering Issue in a Distributed Database Concurrency Control Mechanism: A General Case", Proceedings of the 3rd International Conference on Distributed Computing Systems, 1982 pp. 447-452

[6] Kamoun F., Kleinrock L., Muntz R., "Queueing Analysis of the Ordering Issue in a Distributed Database Concurrency Control Mechanism", Proceedings of the 2nd International Conference on Distributed Computing Systems 1981, pp. 13-23.

[7] Li V., "Performance Models of Timestamp-Ordering Concurrency Control Algorithms in Distributed Databases", IEEE Transactions on Computers, Vol. C-36, No. 9, September 1987, pp. 1041-1051.

[8] Reed D., "Naming and Synchronization in a decentralized computer systems", Ph.D. dissertation, Dep. Elec. Eng. Comput. Sci., MIT, Sept. 1978.

[9] Silberschatz A., "A multiversion concurrency control scheme with no rollbacks", in Proc. ACM Symp. Principles Distributed Comput., Aug. 1982, pp. 216-223.

[10] Singhal M., Agrawala A., "Performance Analysis of an Algorithm for Concurrency Control in Replicated Database Systems", ACM SIGMETRICS 1986, pp. 159-169.

[11] Stafylopatis A., Gelenbe E., "Delay Analysis of Resequencing Systems with partial Ordering", PERFORMANCE '87, pp. 433-400.

[12] Stearns R., Rosenkrantz D., "Distributed database concurrency control using before values", in Proc. SIGMOD Conf. Management Data, 1981, pp. 74-83.

[13] Thomas R., "A majority consensus approach to concurrency control for multiple copy databases", ACM Trans. Database Syst., vol. 4, 1979, pp. 180-209.

[14] Wang C., Li V., "Queueing analysis of the conservative timestamp-ordering concurrency control algorithm", in Proc. IEEE Int. Comput. Symp., 1986, pp. 1450-1455.