中華民國八十四年 全國計算機會議論文集
Proceedings of National Computer Symposium 1995

分散式資料庫主拷貝策略之模擬模式
A SIMULATION MODEL FOR THE PRIMARY COPY STRATEGY OF DISTRIBUTED DATABASES

黃胤傅 毛貞傑
Yin-Fu Huang and Chien-Chie Mao

國立雲林技術學院 電子與資訊工程研究所
Department of Electronic Engineering, National Yunlin Institute of Technology
123 University Road, Section 3, Yunlin, Taiwan 640, R.O.C.

摘要

在分散式 INGRES 中，是以兩段式鎖法爲基礎的主拷貝策略來解決協同控制的問題。爲了觀察分散式 INGRES 中交易的行爲，在這篇論文提出了一套模擬模式並做了模擬。在另一方面，爲了觀察資料片斷的主拷貝位置對交易反應時間的影響，我們嘗試設計一個粗略方法，其能夠產生較佳反應時間的主拷貝位置。實驗結果顯示我們方法選出的主拷貝位置在低衝突率和交易到達的長間隔下，的確會產生較佳的結果。

關鍵詞: 分散式資料庫, 模擬, 效能, 兩段式鎖法, 協同控制

Abstract

In distributed INGRES, the primary copy strategy based on two-phase locking is used to solve the concurrency control problem. To observe the behaviors of the transactions in distributed INGRES, a simulation model is proposed and a simulation is made in this paper. On the other hand, to observe how the primary copy sites for fragments influence the overall-response time for the transactions, we try to give a rough method for selecting the primary copy sites that can result in better overall-response time. The experimental results obtained from the simulation show that the primary copy sites selected by our method generate better results under low conflict ratio or long interval among transaction arrivals.

Keywords: Distributed database, Simulation, Performance, Two-phase locking, Concurrency control

1 Introduction

Distributed INGRES was developed at the University of California at Berkeley, as a distributed version of the relational database system INGRES [5,11]. It consists of multiple copies of University INGRES (INteractive Graph and REtreieval System), running on multiple interconnected machines [3]. It is designed to operate on both a local (Ethernet-like) network and a geograghical(ARPANET-like) network [11]. It supports **location transparency** (like SDD-1 and R*); it also supports data fragmentation (via restriction but not projection), with **fragmentation transparency**, and data replication for such fragments with **replication transparency**. The distributed database in such an environment may be called a **Highly Replicated DataBase** (HRDB) [12], which is partially or completely replicated in a large number of sites connected by a communication network. The transactions in Distributed INGRES may be issued at any site in the network. Replication provides ease of access, availability, and reliability in these transactions.

In a dynamic database, the update operations in a transaction are very frequent. In order to maintain consistency of the replicas, these updates have to be propagated to all replicas eventually. Thus, update propagation strategies have a significant impact on the performance of a HRDB. Distributed INGRES provides two algorithms: a "performance" algorithm, which works by updating a primary copy and then returning control to the transaction (leaving the propagated updates to be performed in parallel by a set of slave processes), and a "reliable" algorithm, which updates all copies immediately [3,9,11]. Furthermore, the applicability of certain update propagation strategies is strongly influenced by the choice of concurrency control algorithms. The concurrency control of Distributed INGRES is based on two-phase locking

[11].

To solve the concurrency control problem, **the primary copy strategy** based on two-phase locking is used in Distributed INGRES [9]. Under this strategy, for a given logical object x, the site holding the primary copy of x will handle all lock requests on x. In our environment, a logical object called a fragment is a horizontal subset of a relation. Till now, though there are so many papers discussing the performance in centralized databases [1,2], [6-8], [10], [13-16], few ones investigated the same issue in distributed databases. To observe the behaviors of the transactions in Distributed INGRES, a simulation model is proposed and a simulation is made. On the other hand, to observe how the primary copy sites for fragments influence **the overall-response time** for the transactions, we try to give a rough method for selecting the primary copy sites that can results in better overall-response time. In addition, the simulation model proposed here can also be used to simulate other distributed database systems as long as parts of the model are modified.

In Section 2, a system model is presented and the behaviors of transactions are analyzed. In addition, a rough method for selecting the primary copy sites for fragments is also proposed. In Section 3, a simulation model is proposed. Then the simulation and some experiments are made in Section 4. Finally, we make a conclusion in Section 5.

2 The System Analysis

2.1 The System Model

We assume that the sites in the model are connected by a **point-to-point** communication network. One site in the network is connected to the other with a logical communication link. Each site in the model may contain either a complete replica of the database called **fully-replicated database** or some fragment copies of the database called **partially-replicated database**. Such a database partially or fully replicated in the sites connented by the communication network is called a **Highly Replicated DataBase** (HRDB) [12]. A transaction can issue at any site in a HRDB and access the fragments in the sites that contain those replicas. Moreover, we assume that the communication among sites is jammed free, and there are no failures in sites and communication links.

The system can be viewed from four viewpoints, which are Sites, Network, Fragments and Transactions. The relationships among them are defined as follows:

S - set of sites in the system.

G = (S, E) - a point-to point communication network, where $E \subset S \times S$ is the set of logical communication links. In other words, $(n_1, n_2) \in E$ if there is a logical communication link between the site n_1 and n_2.

FC(S_i) - set of fragment copies resident at the site S_i.

T_i - a transaction contains a serial of operations which have the form **(T_i, op_type, f_j)**, where op_type represents the type of operations, which could be "read" or "write", and $f_j \in F$ represents the fragment which the operation accesses.

T(S_i) - set of transactions issued at the site S_i.

$$T = \bigcup_i T(S_i)$$ - set of transactions in the system.

F - set of fragments which the transactions in T access.

Based on the system model described above, the simulation can be made smoothly.

2.2 Analysis of Transaction Behaviors

In order to simulate Distributed INGRES and solve the problem of selecting primary copy sites for fragments in Distributed INGRES, the behavior of an operation initiated in a transaction is analyzed. The analysis is divided into two parts, which are the Analysis of Locking and the Analysis of Access.

2.2.1 The Analysis of Locking

The concurrency control of Distributed INGRES is based on two-phase locking. Before accessing a data object, the transaction must acquire a lock on that object. After releasing a lock, the transaction should not issue any lock requests.

The problem with locking in a distributed environment is the amount of message traffic it generates [3]. Consider a transaction T that needs to update an object which exists at n remote sites. If every site has its own Lock Manager to control incoming lock requests, then a straightforward implementation of the two-phase locking protocol will require 5n messages: n lock requests, n lock grants, n updates, n acknowledgments, and n unlock requests. A better approach is to adopt the primary copy strategy. For a given logical object x, the Lock Manager at the site containing the primary copy of x will handle all lock requests on x. Under this strategy, the total number of messages will be reduced from 5n to 2n+3 (one lock request, one lock grant, n updates, n acknowledgments, and one unlock request). In Distributed INGRES, the primary copy strategy based on two-phase locking is used [9].

Under the primary copy strategy, before executing a read or an update operation on an object, the transaction concerned must acquire a lock from the

Lock Manager at the site containing the primary copy of the object. Once it has acquired a lock, the transaction concerned must not release that lock until COMMIT. After releasing a lock, the transaction should not issue any lock requests. For each fragment accessed by a transaction, the transaction must issue a lock request and transmit it to the primary copy site. Then the lock request is processed by the Lock Manager. After a successful locking, the primary copy site will respond with a grant for the lock back to the issuing site and then the transaction can continue accessing the fragment. If the transaction no more accesses the locked fragment, it will issue an unlock request to the primary copy site, and similarly the unlock request is processed by the Lock Manager. If an unsuccessful locking happens, the transaction is blocked and waits for the unlock request of the transaction holding the object. Under this situation, the extra blocked cost is taken.

2.2.2 The Analysis of Access

A transaction issued at any site in the network contains a serial of operations. The type of operations could be "read" or "write". When the operation type is "write", the problem of data consistency must be taken into account. Owing to data replication, there are replicas resident at some sites for a logical data object. In order to maintain consistency of the replicas, an update to a logical data object must be propagated to all replicas of that object.

In Distributed INGRES, two update propagation algorithms are provided [3,9,12]. One is a **"performance"** algorithm, which works by updating a primary copy and returning control to the transaction (leaving the propagated updates to be performed in parallel by a set of slave processes). Another is a **"reliable"** algorithm, which updates all copies immediately. In the "performance" algorithm, update operations are directed to the primary copy in the first instance. An update is considered completed as soon as it has been applied to the primary copy (Control is returned and the transaction can continue execution). The site holding the primary copy is responsible for broadcasting the update to all other sites after applying the initial update; those other sites can then update their copies in parallel with the continuing transaction execution. The "performance" algorithm processes updates with the shortest response time. However a problem with this approach may happen [4]. A transaction may update an object whose primary copy is at some remote site, and after a while it may issue a read for an object, which is directed to a secondary copy that has not yet been updated. The "reliable" algorithm is intended to overcome this drawback. Because the "reliable" algorithm is dependable and simple, it is used in our simulation model. The type of an operation may be "read" or "write", and the fragment accessed by an operation may be local or remote. Therefore, to analyze the access behavior of an operation (T_i, op_type, f_j), based on the "reliable" algorithm, the following four cases must be considered.

Let **IS(T_i)** be the issuing site for a transaction T_i.

Case 1: op_type = "read" and $f_j \in$ FC(IS(T_i))

In this case, the operation only spends a Local Processing Cost (LPC) in reading the fragment fj at the issuing site.

Case 2: op_type = "read" and $f_j \notin$ FC(IS(T_i))

In this case, the read request should be transmitted to one of the sites holding the replicas of the fragment f_j, called the candidate sites. Here three costs are taken: Read Request Cost (RRC) from the issuing site to one of the candidate sites, Local Processing Cost (LPC) at the candidate site, and Transmitting Result Cost (TRC) from the candidate site to the issuing site. For each candidate site, the sum of the above three costs are calculated and compared with each other. Then a candidate sit with the minimum cost is chosen to be the site to which the read operation will be directed. After the read request is transmitted to the chosen site, the fragment f_j will be read there. Then the results about the fragment f_j will be transmitted back to the issuing site.

Case 3: op_type = "write" and $f_j \in$ FC(IS(T_i))

In this case, the fragment f_j is updated at the issuing site. At the same time, the update requests should be propagated to the sites holding the replicas of the fragment f_j. After updating, those sites will respond with acknowledgments for the updates to the issuing site. Here three costs are taken: Update Request Cost (URC), Local processing Cost (LPC), and ACKnowledgment Cost (ACKC). For each site holding the replicas, the sum of the above three costs is calculated and compared with each other. Then the write operation spends the maximum cost to finish the updates for the fragment f_j.

Case 4: op_type = "write" and $f_j \notin$ FC(IS(T_i))

The update behavior in this case is similar to that in Case 3, but the fragment f_j is not updated at the issuing site.

2.3 A Rough Method for Selecting the Primary Copy Sites for Fragments

In distributed INGRES, the primary copy sites for fragments are randomly selected. To observe how the primary copy sites for fragments influence **the overall-response time** for transactions, we try to give a rough method for selecting the primary sites for fragments that can results in better overall-response time. In other words, given a set of transactions **T** whose issuing site is described by **T(S_i)** and a set of fragments **F** whose copy allocation is described by **FC(S_i)**, we try to find the primary copy site function **PCS** to determine how the primary copy of each fragment f_j in **F** should be assigned to a site in **S** such that the overall-response time of the transactions in **T** is better.

Since the time and the order of transaction arrivals are not known in advance, the wait-for sequences among transactions for the conflicted locking are not determined. As a result, the blocked costs cannot be estimated. Therefore, the blocked costs are difficult to be involved in our method. Leaving the blocked costs behind, how to select the primary copy sites for fragments is not affected by the costs taken in the access, but is only affected by the costs taken in the locking from the analysis of transaction behaviors in Section 2.2. In other words, no matter how the primary copy sites for fragments are changed, the costs taken in the access are not changed. Next, some input parameters used in our method are introduced first, and then the method is proposed.

2.3.1 Parameters

There are two types of input parameters used in our algorithm. One is the type concerned with the costs taken in the locking, and the other is the type concerned with the transaction properties. The former includes five input parameters listed as follows.
(1) **RLC(s_i,s_j)** indicates the transmission cost of a lock request from s_i to s_j.
(2) **LC(s_i)** indicates the processing cost of a lock request at s_i.
(3) **LGC(s_i,s_j)** indicates the transmission cost of a grant for lock from s_i to s_j.
(4) **RULC(s_i,s_j)** indicates the transmission cost of an unlock request from s_i to s_j.
(5) **ULC(s_i)** indicates the processing cost of an unlock request at s_i.
The latter includes two input parameters listed as follows.
(1) **O(T_i,s_j)** indicates the number of Occurrence for a transaction T_i issued at s_j.
(2) **W(T_i)** indicates Weight factor, which is used to represent the significance of a transaction T_i.

The input parameters described above are given values according to the actual environment. Through these input parameters, the problem of selecting primary copy sites for fragments in Distributed INGRES will become more general.

2.3.2 Description of the Rough Method

There are three variables used in our algorithm. **SFPC(f_j)** indicates Set of Feasible Primary Copy sites, which is the set of the candidates for the primary copy site of the fragment f_j. **F(T_i)** is the set of the fragments which a transaction T_i accesses. **PC_Cost(f_j,s_i)** is the cost of selecting s_i to be the primary copy site of the fragment f_j.

In the beginning of our algorithm, SFPC(f_j) for each fragment f_j in F is determined by checking FC(s_k) for each s_k in S. Then F(T_i) for each transaction T_i is determined by scanning those operations in the transaction T_i. No matter how many operations are issued to access a fragment in a transaction, only a lock and an unlock are needed for the fragment in a transaction. Thus, the five costs taken in the locking are summed up for the fragment in a transaction. However, the five costs not only depend on the issuing site of a transaction, but also on the feasible primary copy site for the fragment. Therefore, the costs taken in the locking are calculated for each site in SPFC(f_j) and for each fragment f_j in F(T_i), and then PC_Cost(f_j, s_i) can be derived. Here the two input parameters W(T_i) and O(T_i, s_j) are considered. Finally, PC_Cost(f_j, s_i) for all sites s_i in SPFC(f_j) are compared and the site with the minimum cost is chosen to be the primary copy site of the fragment f_j. Then the primary copy site of each fragment f_j in F is selected. The detailed algorithm is shown as follows.

```
/*Beginning of the algorithm*/
Begin
    For each $f_j$ in F Do
    Begin
        SPFC($f_j$) = $\phi$ ;
        For each $s_k$ in S Do
        If $f_j \in$ FC($s_k$) Then
        Begin
            SFPC($f_j$) = SFPC($f_j$) $\cup$ {$s_k$}
            PC_Cost($f_j$, $s_k$) = 0;
        End
    End;
    For each $s_k$ in s Do
    For each transaction $T_i$ in $T(s_k)$ Do
    Begin
        $F(T_i) = \phi$ ;
        For each operation ($T_i$, op_type, $f_j$) in $T_i$ Do
        $F(T_i) = F(T_i) \cup \{f_j\}$;
        For each $f_j$ in $F(T_i)$ Do
          For each $s_m$ in SFPC($f_j$) Do
  $PC\_Cost(f_j, s_m) = PC\_Cost(f_j, s_m) + [RLC(s_k, s_m) + LC(s_m)$
  $+ LGC(s_m, s_k) + RULC(s_k, s_m) + ULC(s_m)] * \frac{O(T_i, s_k)}{W(T_i)}$
    End;
  For each $f_j$ in F Do
  Begin
  $Min\_Cost = \underset{s_m \in SPFC(f_j)}{MIN} PC\_Cost(f_j, s_m)$;
      If Min_Cost = PC_Cost($f_j$, $s_1$) Then
          PCS($f_j$) = $s_1$;
  End;
End.
/*End of the algorithm*/
```

3 The Simulation Model

3.1 The Queuing Model

To make the simulation as close to a real situation as possible, a general and closed queuing model is designed. There are two main components for each site in the queuing model. One is **Concurrency Control (CC)** Component and the other is **Database Access Component (DAC)**. The CC Component includes **Concurrency Control (CC) Queue, Concurrency Control (CC) Server** and **Blocked Queue**. The DAC includes **Object Queue** and **Object Server**. The queuing model can reflect the behavior of a transaction very close to a real distributed database management system.

Each transaction has the requests of LOCK, UNLOCK , and ACCESS which may be READ or UPDATE. When a transaction begins to access the database, it enters the queuing model as shown in Fig. 1. The transaction first issues a LOCK request to the lock manager in the primary copy site for the fragment which it requires to access, and enters the CC queue of the primary copy site. Before the data objects in the fragment can be accessed by the DAC, the LOCK request must be granted by the CC server. When a LOCK request is made, there are two possible outcomes. The first outcome is that the LOCK request is **granted**, and the data objects in the locked fragment could then be accessed by the DAC. The second outcome is the fragment to be locked is unavailable, and the transaction is **blocked**. The blocked transaction enters the blocked queue until the blocked fragment is unlocked by an UNLOCK request, and a blocked transaction resident in the blocked queue is waked up by the UNLOCK and is ready to enter the object queue.

When a LOCK request is granted, a grant for the LOCK request is transmitted back to the issuing site ,and then the data objects in the locked fragment could be accessed by ACCESS requests safely. Assume that each site in the network owns a Database Management System, and the ACCESS request is serviced by the DAC of it. When the operation type is "read", the READ request is issued to the DAC of the determined site as shown in Case1 and Case2 of Section 2.2.2. As the operation type is "write", the UPDATE requests are issued to the DAC of each site holding the replica of the fragment as shown in Case 3 and Case 4 of Section 2.2.2. After receiving the ACCESS request, the DAC first puts it into the object queue, and the data objects in the locked fragment are then accessed by the object server. Then the access results about the locked fragment for a read request or acknowledgments for the update requests are transmitted back to the issuing site.

If the transaction is not over, there are two possible outcomes. The first outcome is that the next request of the transaction is ACCESS request. In this case, the transaction enters the DAC of the simulation model to access the needed fragment. The second outcome is that the next request of the transaction is CC request including LOCK or UNLOCK request. In that case, the transaction enters the CC component to lock or unlock the fragment. As the transaction has finished its operations, it is issued again after a time interval if necessary. We assume that the UNLOCK requests for all the locked fragments in the transaction are not issued until END-OF-TRANSACTION (EOT). Although a transaction may issue a number of LOCK, ACCESS, and UNLOCK requests, each fragment accessed by a transaction can only be locked once by the transaction.

3.2 Workload Parameters

There are two workload parameters used in the simulation. One is **conflict ratio** and the other is **interval among transaction arrivals**. A fragment accessed by a transaction is called a **conflicted** fragment if the fragment is also accessed by other transactions. The conflict ratio represents the number of conflicted fragments in a transaction against the number of fragments in the transaction. The transaction with a higher conflict ratio may have greater possibility to be blocked and need more blocked time. The interval among transaction arrivals represents the interval among the time of transaction arrivals. With the increase of the interval among transaction arrivals, the blocked time in which an operation is blocked is less.

3.3 Cost Parameters

To reflect the simulation environments as close to a real situation as possible, the cost parameters are included. The cost parameters can be classified into two categories. Since the speed of machines in sites is different, the first category, **the machine oriented parameters**, is given. The machine oriented parameters includes 1) Local Processing Cost (LPC), 2) Lock Cost (LC), the processing cost of a lock request, 3) Unlock Cost(ULC), the processing cost of an unlock request.

In addition, there are different communication cost among sites for different media used in communication links; the second category, **the communication oriented parameters**, is given. The communication oriented parameters includes 1) Request Lock Cost(RLC), the transmission cost of a lock request, 2) Lock Grant Cost (LGC), the transmission cost of a grant for a lock, 3) Request Unlock Cost (RULC), the transmission cost cf an unlock, 4) Read Request Cost (RRC), the transmission cost of a read request, 5) Transmission Result Cost (TRC) ,the transmission cost of a read request, 6) Update Request Cost (URC), the transmission cost of an update request, 7) ACKnowledge Cost (ACKC), the transmission cost of the acknowledgment. For different cost parameters, the different overall-response time is obtained in the simulation.

3.4 Measure Parameters

The goals for the transactions issued by users in Distributed INGRES is to reduce **the response time** per transaction. Moreover, **the blocked time** has an influence upon the response time. Therefore, there are two parameters measured in the simulation. The first paramters, **response time**, measures the average time in which the system responds to each transaction in T. The second parameter, **blocked time**, measures the average time in which an operation is blocked in the blocked queue. According to the two parameters, the performance of our rough method for selecting the primary copy sites for fragments is measured in the experiments.

4 The Experimental Results

To observe how the primary copy sites for fragments influence the response time, some experiments are done. In the experiments, the estimated average time in which the system responds to a transaction in T (T_{es}), regarding the interval among transaction arrivals (INT), is estimated without considering the blocked time. Because we assume that the UNLOCK request for all locked fragments in the transaction are not issued until END-OF-TRTANSACTION (EOT), the blocked transaction often has to wait in the blocked queue for about a period of $T_{es}/2$ in average cases. Thus, we use $T_{es}/2$ as the base line of the interval among transaction arrivals and increase gradually the interval among transaction arrivals in our observation.

4.1 The Experiments

We consider a system with 5 sites. There are 6 transactions existing in the system and these transactions access 10 fragments totally. A description of the entire system is given as follows.

$S=\{s_1,s_2,s_3,s_4,s_5\}$

$T=\{T_1,T_2,T_3,T_4,T_5,T_6\}$

$F=\{f_1,f_2,f_3,f_4,f_5,f_6,f_7,f_8,f_9,f_{10}\}$

The issuing site of transactions:

$T(s_1)=\{T_3\}$; $T(s_2)=\{T_1\}$; $T(s_3)=\{T_4\}$; $T(s_4)=\{T_2,T_5\}$; $T(s_5)=\{T_6\}$;

Allocation of fragment copies:

$FC(s_1)=\{f_3,f_4,f_6,f_8,f_9\}$; $FC(s_2)=\{f_2,f_4,f_{10}\}$; $FC(s_3)=\{f_3,f_5,f_6,f_7\}$; $FC(s_4)=\{f_1,f_7,f_8,f_{10}\}$; $FC(s_5)=\{f_1,f_2,f_5,f_9\}$;

Communication cost:

RLC=LGC=RULC=RRC=TRC=URC=ACKC

	s_1	s_2	s_3	s_4	s_5
s_1	0	200	300	500	100
s_2	200	0	100	400	300
s_3	300	100	0	600	500
s_4	500	400	600	0	200
s_5	100	300	500	200	0

Lock/Unlock cost:

$LC(s_1)=ULC(s_1)=1.5$;$LC(s_2)=ULC(s_2)=2.5$; $LC(s_3)=ULC(s_3)=2$;$LC(s_4)=ULC(s_4)=3$; $LC(s_5)=ULC(s_5)=1$;

Local processing cost

$LPC(s_1)=3$; $LPC(s_2)=5$; $LPC(s_3)=4$; $LPC(s_4)=6$; $LPC(s_5)=2$

From $FC(s_i)$ of the system described above, there exist 2 feasible primary copy sites for each fragment.

Thus, there are all 1024 feasible solutions of primary copy sites for 10 fragments. In the section, three experiment results according to different conflict ration are summarized. The optimal solution is found by searching exhaustively through all feasible solutions.

Experiment 1: conflict ratio=0

The pattern of transactions is given as follows:
T_1:[(T_1,w,f_5),(T_1,w,f_6),(T_1,r,f_6)]
T_2:[(T_2,r,f_4),(T_2,w,f_1)]
T_3:[(T_3,w,f_5),(T_3,r,f_{10}),(T_3,r,f_5)]
T_4:[(T_4,w,f_2),(T_4,r,f_2)]
T_5:[(T_5,r,f_7),(T_5,w,f_9)]
T_6:[(T_6,r,f_8),(T_6,w,f_8)]

The experimental results are listed in Table I. Table I shows that the primary copy sites selected by our method is validated to be the optimal solution because the response time generated by them is the minimum one .

Experiment 2: conflict ratio =50%

The pattern of transactions is given as follows:
T_1:[(T_1,w,f_{10}),(T_1,w,f_1),(T_1,r,f_6),(T_1,w,f_3)]
T_2:[(T_2,r,f_4),(T_2,w,f_1)]
T_3:[(T_3,w,f_5),(T_3,r,f_{10})]
T_4:[(T_4,r,f_9),(T_4,w,f_2)]
T_5:[(T_5,r,f_7),(T_5,w,f_9)]
T_6:[(T_6,w,f_{10}),(T_6,w,f_8)]

After calculation, T_{es} is equal to 3463.6166. The experimental results are listed in Table II. Table II show that the primary copy sites selected by our method can generate better response time as the interval among transaction arrivals is greater than $T_{es}/2$. With the increase of the interval among transaction arrivals, the primary copy sites selected by our method improve.

Experiment 3: conflict ratio =100%

The pattern of transactions is given as follows:
T_1:[(T_1,w,f_1),(T_1,w,f_{10}),(T_1,r,f_3),(T_1,r,f_6)]
T_2:[(T_2,w,f_8),(T_2,w,f_5),(T_2,r,f_6),(T_2,w,f_7)]
T_3:[(T_3,r,f_1),(T_3,r,f_5),(T_3,w,f_{10})]
T_4:[(T_4,w,f_3),(T_4,r,f_2),(T_4,w,f_9),(T_4,r,f_8)]
T_5:[(T_5,w,f_2),(T_5,r,f_4),(T_5,w,f_5),(T_5,w,f_7)]
T_6:[(T_6,w,f_9),(T_6,w,f_4),(T_6,r,f_7)]

After calculation, T_{es} is equal to 5679.4166. The experimental results are listed in Table III. Table III show that the primary copy sites selected by our method can still generate better response time as the interval among transaction arrivals is greater than $T_{es}/2$. The response time generated by the primary copy sites selected by our method ranks at least 21th under 1024 possible solutions. In other words, its rank lies at least within the rank of top 2%. Although the response time generated by the primary copy sites selected by our method is not optimal, the difference between the optimal response time and the response time generated by our method is small.

Unfortunately, the primary copy sites selected by our method do not necessarily improve with the increase of the interval among transaction arrivals. The reason for the unexpected results is that the wait-for time sequences among transactions for the conflicted fragments may be different for different primary copy sites of a fragment. As a result, the different wait-for sequences may result in different blocked time. Disregarding the blocked time, the response time generate by our method is minimum. Considering the blocked time, however, the blocked time generated by our method may be high, so that the response time is no longer minimum. What is worse, the primary copy sites selected by our method may generate worse response time as the blocked cost is high enough to dominate the response time.

4.2 Summary

As the conflict ratio is equal to zero, the primary copy sites selected by our method generate optimal response time. With the increase of the conflict ratio and the interval among transaction arrivals, the primary copy sites selected by our method have greater possibility to generate more blocked time and result in worse response time. The wait-for sequences among transactions for the conflicted fragments may be different for different primary copy sites of a fragment, so the primary copy sites selected by our method generate better response time only under low conflict ratio or long interval among transaction arrivals.

5 Conclusions

In this paper, a simulation model is proposed and the simulation is made for Distributed INGRES. To build the simulation model, some characteristics in Distributed INGRES are analyzed. Besides, to observe how the primary copy sites for fragments influence the overall-response time for the transactions, we try to give a rough method for selecting the primary sites for fragments. Through the analysis of transaction behaviors, we find that how to select the primary sites for fragments depends on the cost entailed in the locking on condition that there be no blocked cost.

Since the wait-for sequences among transactions for the conflicted fragments is not determined until the run time, the blocked cost is difficult to be involved in our method. The experiment results obtained from the simulation show that the primary copy sites selected by our rough method generate the optimal overall-response time as the conflict ratio is equal to zero and generate better overall-response time under low conflict ration or long interval among transaction arrivals. The simulation model proposed in the paper is used to simulate Distributed INGRES. In addition, it can also be used to simulate other distributed database systems as long as parts of the model are modified.

References

[1] R. Agrawal, M. J. Carey, and M. Livny, "Models for studying concurrency control performance alternatives and implications," in Proceedings ACM SIGMOD Int. Conf. on Management of Data, 1985, pp. 108-121.

[2] M. J. Carey and M. R. Stonebraker, "The performance of concurrency control algorithms for database management systems," in Proceedings 10th Int. Conf. on VLDB, 1984, pp. 107-118.

[3] S. Ceri and G. Pelagatti, Distributed Databases: Principles and Systems, New York, McGraw-Hill, 1984.

[4] C. J. Date, An Introduction to Database Systems, 6th ed., Addison-Wesley, 1995.

[5] C. J. Date, An Introduction to Database Systems, vol. 2, Addison-Wesley, 1983.

[6] C. Hasse and G. Weikum, "A performance evaluation of multi-level transaction management," in Proceedings 17th Int. Conf. on VLDB, 1991.

[7] Y. F. Huang and Y. H. Chin, "A new methodology to evaluate locking protocols," IEEE Trans. Knowledge and Data Engineering, vol. 2, no. 4, pp. 431-435, 1990.

[8] D. Mitra and P. J. Weinberger, "Probability models of database locking: solutions, computational algorithm, and asymptotics," J. ACM, vol. 31, no. 4, pp. 855-878, Oct. 1984.

[9] T. P. Ng, "Propagation updates in a highly replicated database," in Proceedings 6th Int. Conf. on Data Engineering, 1990, pp. 529-536.

[10] D. Potier and Ph. Leblanc, "Analysis of locking policies in database management systems," Comm. ACM, vol. 23, no. 10, pp. 584-593, Oct. 1980.

[11] M. Stonebraker and E. Neuhold, "A distributed data base version of INGRES," in Proceedings 2nd Berkeley Workshop on Distributed Data Base and Networks, May 1977.

[12] M. Stonebraker, "Concurrency control and consistency of multiple copies in distributed INGRES," IEEE Trans. Software Engineering, vol. 5, no. 3, May 1979.

[13] Y. C. Tay, R. Suri, and N. Goodman, "A mean value performance model for locking in databases: the waiting case," in Proceedings 3rd ACM SIGACT-SIGMOD Symposium on Principles of Database Systems, 1984, pp. 311-322.

[14] Y. C. Tay and R. Suri, "Choice and performance in locking for databases," in Proceedings 10th Int. Conf. on VLDB, 1984, pp. 119-128.

[15] Y. C. Tay, R. Suri, and N. Goodman, "A mean value performance model for locking in databases: the no-waiting case," J. ACM, vol. 32, no. 3, pp. 618-651, July 1985.

[16] A. Thomasian and I. K. Ryu, "Performance analysis of two-phase locking," IEEE Trans. Software Engineering, vol. 17, no. 5, pp. 386-402, May 1991.

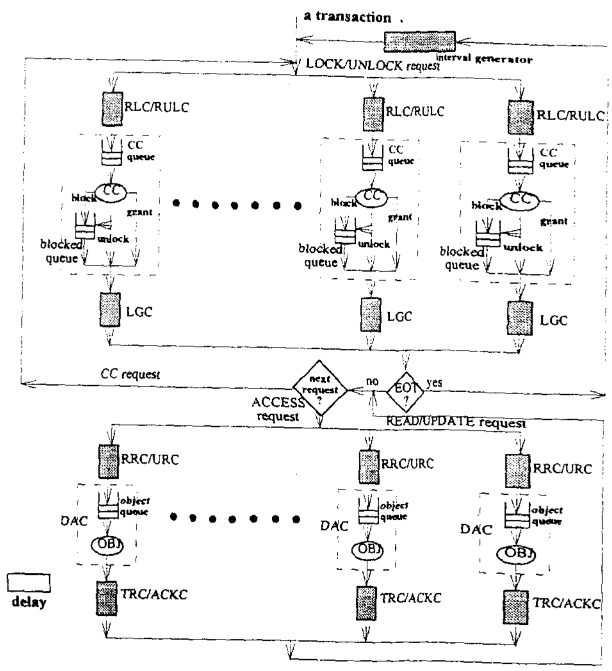

Fig. 1 Logical queuing model

Table I Results of Experiment 1

The optimal primary copy sites:

The primary copy sites selected by our method:

f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}
4	2	3	2	5	3	4	1	5	2

	response time	number of possible solutions
*.#	1699.166626	1
	1748—1800	12
	1800—1900	35
	1900—2000	74
	2000—2100	116
	2100—2200	148
	2200—2300	165
	2300—2400	162
	2400—2500	131
	2500—2600	98
	2600—2700	55
	2747—2800	22
	2847—2899	5

* the optimal result

the result selected by our method

Table II Results of Experiment 2

INT		primary copy sites										blocked time	response time	rank
		f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}			
Tes/2	*	5	2	3	2	5	3	4	1	5	2	121.41	2935.29	1
	#	4	2	3	2	5	3	4	1	5	2	143.12	2937.29	2
Tes*	*	5	2	3	2	5	3	4	1	5	2	100.79	2887.19	1
7/12	#	4	2	3	2	5	3	4	1	5	2	122.51	2889.19	2
Tes*	*	5	2	3	2	5	3	4	1	5	2	80.18	2839.09	1
2/3	#	4	2	3	2	5	3	4	1	5	2	101.89	2841.09	2
Tes*	*	5	2	3	2	5	3	4	1	5	2	59.56	2742.88	1
3/4	#	4	2	3	2	5	3	4	1	5	2	81.28	2744.88	2
Tes*	*	5	2	3	2	5	3	4	1	5	2	38.95	2742.88	1
5/6	#	4	2	3	2	5	3	4	1	5	2	60.66	2744.88	2
Tes*	*	5	2	3	2	5	3	4	1	5	2	18.33	2694.78	1
11/12	#	4	2	3	2	5	3	4	1	5	2	40.05	2696.78	2
Tes	*	4	2	3	2	5	3	4	1	5	2	19.43	2648.68	1
	#	4	2	3	2	5	3	4	1	5	2	19.43	2648.68	1
Tes*	*	4	2	3	2	5	3	4	1	5	2	0	2603.33	1
13/12	#	4	2	3	2	5	3	4	1	5	2	0	2603.33	1
Tes*	*	4	2	3	2	5	3	4	1	5	2	0	2603.33	1
7/6	#	4	2	3	2	5	3	4	1	5	2	0	2603.33	1

* the optimal solution

the solution selected by our method

Table III Results of Experiment 3

INT		primary copy sites										blocked time	response time	rank
		f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}			
Tes/2	*	5	2	3	1	5	1	4	1	1	2	198.03	5220.97	1
	#	5	2	3	1	5	1	4	4	1	2	294.70	5776.43	11
Tes*	*	5	2	3	1	5	1	4	1	1	2	54.97	5095.41	1
7/12	#	5	2	3	1	5	1	4	4	1	2	127.77	5164.33	17
Tes*	*	5	2	3	1	5	1	4	1	1	2	33.46	5016.53	1
2/3	#	5	2	3	1	5	1	4	4	1	2	106.26	5085.45	21
Tes*	*	5	2	3	1	5	1	4	1	1	2	11.95	4937.65	1
3/4	#	5	2	3	1	5	1	4	4	1	2	84.74	5006.57	21
Tes*	*	5	2	3	1	5	1	4	1	1	2	17.84	4860.27	1
5/6	#	5	2	3	1	5	1	4	4	1	2	63.23	4927.69	15
Tes*	*	5	2	3	1	5	1	4	1	1	2	0	4794.83	1
11/12	#	5	2	3	1	5	1	4	4	1	2	41.72	4848.81	11
Tes	*	5	2	3	1	5	1	4	4	1	2	20.20	4769.93	1
	#	5	2	3	1	5	1	4	4	1	2	20.20	4769.93	1
Tes*	*	5	2	3	1	5	1	4	4	1	2	0	4695.83	1
13/12	#	5	2	3	1	5	1	4	4	1	2	0	4695.83	1
Tes*	*	5	2	3	1	5	1	4	4	1	2	0	4695.83	1
7/6	#	5	2	3	1	5	1	4	4	1	2	0	4695.83	1

* the optimal solution

the solution selected by our method