データベースシステム 119−63
（1999. 7. 23）

移動体計算環境における断線を考慮したデータベース更新制御方式

Loh Yin Huei　原 隆浩　塚本 昌彦　西尾 章治郎
大阪大学大学院工学研究科情報システム工学専攻
〒565-0871 吹田市山田丘 2-1
{yhloh,hara,tuka,nishio}@ise.eng.osaka-u.ac.jp

近年，無線通信技術の急速な発展により，移動体計算環境が普及しつつある．移動体計算環境では，無線の通信範囲の制限や移動体の省電力などのために，サイト間の断線が頻繁に発生する．断線した複数のサイトにおいて，同一データの複製を更新するトランザクションが同時に実行されると，そのデータの一貫性が損なわれてしまう．そこで本稿では，サイトの断線時に，トランザクションの発生確率とホスト間の断線時間により，データベースの更新制御法をトークン手法と楽観手法のいずれかから動的に選択する手法を提案する．トークン手法では，断線時に，トランザクションの実行権利を唯一のサイトに与えることで複製間の一貫性を保証する．一方，楽観手法では，複数の断線したサイトで同時にトランザクションを実行でき，再接続時に更新操作の衝突を検出すれば，一部のトランザクションをロールバックする．
キーワード：断線操作，データベース同時実行，移動体計算環境

Controlling Database Updates in Mobile Computing Environments with Frequent Disconnection

Yin-Huei Loh　Takahiro HARA　Masahiko TSUKAMOTO　Shojiro NISHIO
Department of Information Systems Engineering, Graduate School of Engineering, Osaka University
2-1 Yamadaoka, Suita, Osaka 565-0871, Japan
{yhloh,hara,tuka,nishio}@ise.eng.osaka-u.ac.jp

With the rapid advancement in wireless networks, mobile computing has become more and more popular. However, the limitations of mobile computers and wireless networks cause frequent disconnection among the hosts. Consequently, transactions on databases on disconnected sites cannot be executed smoothly without creating problems of inconsistency among copies of data. In this paper, we propose an algorithm to minimize this problem. Our approach chooses the appropriate method to control database udpates before disconnection based on the probability that transactions occur and the duration of disconnection time between the sites. Token method enables a single site to execute transactions during disconnection, and thus ensure no conflicts between the transactions. Optimistic method lets multiple disconnected sites execute transactions simultaneously. Conflicts are checked upon reconnection and rollback of transactions is performed if necessary. We show the formula to choose the appropriate method depending on the situation.
keywords: disconnected operations, database concurrency, mobile computing environments

1 Introduction

In mobile computing environments, small portable computers are carried by users who can move flexibly everywhere. These computers, also known as PDA (Personal Digital Assistants) or MH (mobile hosts), have the wireless connection capability which enables users to travel freely everywhere without being limited by the length of the wired cable. However, there are quite some limitations on these mobile hosts. Generally, mobile hosts run on battery power with limited life time; the wireless network, though convenient and require no physical connection, is costly and has limited bandwidth; and the connection has lower quality with more interference compared to conventional wired network. When a mobile host is out of the range of the wireless network, communication becomes impossible, and disconnection is said to have taken place. In other cases, in order to conserve energy, save the network cost and reduce network traffic in the network with limited bandwidth, a mobile host may be disconnected intentionally even though its location is within the reach of wireless network.

During periods of disconnection, necessary data is replicated in the mobile hosts to allow access. However, updates to these data create problems as inconsistency may occur when many copies of updated data exist in many mobile hosts, each with a different version. For this reason, conventional approach disallows updates to the copies of data during periods of disconnection. This limits the update ability of mo-

bile hosts and causes great inconvenience especially to mobile hosts with high update frequency. Hence, there emerges a need to be able to effectively and concurrently run updates to these data, even during disconnection.

As an example of an application in which data needs to be updated during disconnections, consider the schedule plan of a manager. When the manager is out from the office, he may meet with clients who would like to fix a time to have a meeting with him the week after. At the same time, his secretary sitting in the office may receive phone calls from other clients who would also like to have an appointment with the manager the week after. In this scenario, an effective way is needed to allow both parties to update the schedule without having to contact each other, especially if the manager is overseas.

So far, many works have been done to solve the problem mentioned earlier. [2], [3] and [4] focus on the file systems. For database systems, several approaches have been proposed in [1] and [10]. However, rollback of the transactions frequently happens as conflicts may occur among them, resulting in heavy workload.

For data items which can be partitioned and the transactions executed on the data items are commutative as discussed in [5], [7], [8] and [9], strategies which partition the data value to different sites in a distributed environment to allow concurrent updates were proposed. This was further extended for mobile computing environments in [6]. One drawback about this approach is that it can only be applied on partitionable data items and thus the usage is limited.

In this paper, we propose a method which effectively controls the database updates in mobile computing environments with frequent disconnection. It adaptively chooses the appropriate strategy to ensure the best performance, balancing the trade-off between the number of transactions succeed and the number of rollback. Basically our approach applies either the token method or the optimistic method depending on the situation.

The remainder of the paper is organized as follows. In section 2, we describe the system architecture. In section 3, we explain the approach of our proposed algorithm in detail. We evaluate and discuss the proposed method in section 4, and conclude the paper with some discussions about future work in section 5.

2 System Architecture

In this section, we describe the system architecture of our proposed algorithm.

We assume an environment which consists of mobile hosts with or without fixed hosts. We do not differentiate between mobile host and fixed host. Each host carries a copy of the database and can be connected to any other hosts. As the example in Figure 1 shows, even if a mobile host is in the cell within the range of possible wireless communication of the fixed host (mobile support station), it can voluntarily disconnect from the fixed host. The mobile hosts can also communicate freely among themselves. In another case shown in Figure 2, the network may be ad hoc where there exist only mobile hosts which move freely and communicate freely with any other mobile hosts around.

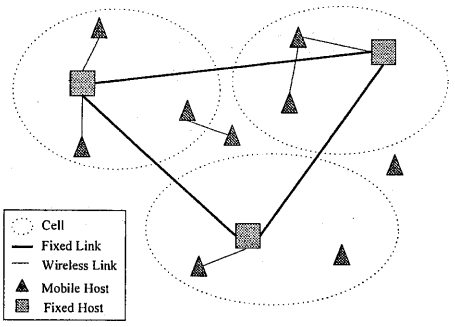

Figure 1: An example of system architecture consisting of fixed and mobile hosts.

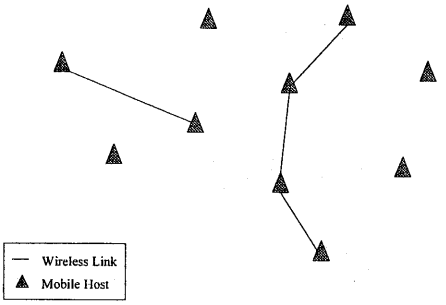

Figure 2: An example of system architecture: ad hoc network consisting of only mobile hosts.

We assume that all disconnections between the hosts are voluntary and the length of the disconnection time is fixed and known. Further, the probability of the occurrence of transactions at each host is known before hand by referring to historical data or planned schedule.

Assume that the database is partitioned into one or more clusters according to the data access patterns of the transactions. Data items which are often accessed together in the same transaction are clustered together. Our algorithm focuses on only one cluster. This is to enable independent transactions to be handled separately so that the number of transactions which can be executed at the same time is higher. The whole database can be handled by running the algorithm on all the data clusters. All the descriptions below are meant for only one data cluster.

In a network, many hosts may exist. Disconnection is assumed to happen one at a time, sequentially, but not simultaneously. Thus, upon disconnection, the network is separated into two different networks, which may be further separated into two different networks again and again, recursively.

The two different separated networks are considered as two sites, each consists of one or more mobile hosts. For the purpose of simplicity, we consider the case in which there are no two disconnections coexist at the same time. Thus, there are at most two sites in the whole network.

Assume that in 1 very small unit of time, at most 1 transaction can occur at 1 site. Thus, the number of transactions, n, which can happen in 1 unit of time is such that $0 \leq n \leq 1$ where

n = probability of the occurrence of 1 transaction per 1 unit of time.

Let $P(k)_i^t$ denotes the probability that i transactions occur at site k in t units of time. Since at most 1 transaction can occur at 1 site in 1 unit of time, the probability that 1 transaction occurs in 1 unit of time at site k is

$P(k)_1^1 = n$,

and the probability that no transactions occur at site k for 1 unit of time is

$P(k)_0^1 = 1 - P(k)_1^1$.

In general, the probability that i transactions happen at site k in the duration of time t is:

$$P(k)_i^t = {}^tC_i (P(k)_1^1)^i (P(k)_0^1)^{(t-i)}. \quad (1)$$

(There are ${}^tC_i$ ways to arrange i transactions in t units of time.)

To decide whether the two disconnected sites are allowed to run transactions on the data or not during disconnection, before the disconnection, we use an algorithm to choose the best method between token and optimistic method. The definition of each method is as follows:

- Token Method: A "token" is used to represent the right to execute transactions. Upon disconnection, this token is given to the site with higher probability of transactions, and only this site with the token is granted the permission to run transactions on the data (followed by a commit action), while the other site without token is prohibited from running transactions on the data.
- Optimistic Method: Both sites can run transactions on the data, but it is not guaranteed that these transactions can be committed. In other words, during the disconnection, if transactions occur at only one of the sites, these updates will succeed when the sites reconnect. On the other hand, if conflicts occur, i.e. transactions occur at both sites, rollback is forced to be performed at the site where the number of transactions occurred is smaller.

Note that in this paper, the term "conflict of transaction" does not just mean conflicts caused by "write" transactions, but also "read" transactions. We do not intend to discuss further details about it, but in principle conflict is defined by the serializability of the transactions.

3 Algorithm Description

In this section, we discuss the way to choose the appropriate method.

Let us consider the number of transactions succeed in a range of time (i.e., the expected disconnection time) by using each method. By "succeed", we mean that the transactions are committed either on the spot or upon reconnection. Even though it is predicted that the number of transactions happen at the site with higher probability of transactions would outnumber the one at the site with lower probability, in real life, there is always a possibility that the reverse happens. When this happens, using optimistic method will ensure higher number of transactions that succeed because it decides the transactions that succeed/fail only after they occurred whereas token method decides before the transactions occur. However, using optimistic method requires much more work to execute all the transactions on both sites and later perform rollback on some of these transactions. Thus, when the expectation of the number of transactions succeed for both methods are the same, using token method is obviously more efficient as it requires less work.

Since the optimistic method requires much more work than the token method, we introduce a function $f(x)$ to define the satisfaction level of each successful transaction. The satisfaction level of each successful transaction decreases with the increase of x, where x is the duration of time between the time the transaction happens and the time it commits. For example, when a transaction happens at time 1 and commits at time 4, $x = 4 - 1 = 3$. Thus, using parameter K (< 1) as a constant, we define $f(x)$ as:

$$f(x) = \begin{cases} 1 - Kx & x < \frac{1}{K} \\ 0 & otherwise. \end{cases} \quad (2)$$

Integrating this function $f(x)$, we define the evaluation function as the multiplication of (i) the number of transactions which happen, and (ii) the satisfaction level of each transaction. We represent this evaluation function with the symbol $F(T)$ for token method and $F(O)$ for optimistic method.

For token method, since transactions always commit at the time they happen, $x = 0$ and $f(x) = 1$ is always true. Thus, $F(T)$ equals the number of transactions succeed in the corresponding disconnection time. This is equal to the number of transactions happen at the site with higher probability of transaction (which is the site that will receive the token), which equals the summation of multiplication of (i) number of transactions, and (ii) the probability that this number of transactions happen. When $P(A)_1^1 > P(B)_1^1$,

$$\begin{aligned} F(T) &= 1P(A)_1^t + 2P(A)_2^t + ... + tP(A)_t^t \\ &= \sum_{i=1}^{t} iP(A)_i^t \\ &= tP(A)_1^1. \end{aligned}$$

(We omit the proving here.)

Thus, we conclude the following equation:

$$F(T) = \begin{cases} tP(A)_1^1 & P(A)_1^1 > P(B)_1^1 \\ tP(B)_1^1 & otherwise. \end{cases} \quad (3)$$

We now derive $F(O)$. For optimistic method, the satisfaction level is always less than 1, since a successful transaction only commits when the sites reconnect. Transactions that happen at different time have different satisfaction level. Thus, we need to consider each of them separately.

From equation (1), we know that:

$$P(A)_i^t = {}^tC_i (P(A)_1^1)^i (P(A)_0^1)^{(t-i)}.$$

Thus, there are ${}^tC_i$ possible patterns that i number of transactions happen in the duration of disconnection time t. Thus, the probability that each way happens is $P(A)_i^t / {}^tC_i$.

The formula of $P(A)_i^t$ can be expanded into ${}^tC_i$ terms, in which each term equals $(P(A)_1^1)^i (P(A)_0^1)^{(t-i)}$, and thus contains i number of $P(A)_1^1$ and $(t-i)$ number of $P(A)_0^1$. Hence, if we consider the total of all ${}^tC_i$ terms, there are $i \times {}^tC_i$ number of $P(A)_1^1$. These $i \times {}^tC_i$ number of $P(A)_1^1$ is also evenly distributed over the time $1, 2, ..., t$. So, for each time $1, 2, ..., t$, the total number of $P(A)_1^1$ is $(i \times {}^tC_i)/t$.

As an example, consider the case of $P(A)_2^4$ ($t = 4$ and $i = 2$). To arrange 2 transactions in 4 units of time, there are ${}^4C_2$ (= 6) ways to arrange them, as shown in Figure 3. The probability that each of these 6 ways happens is $P(A)_2^4/6$. For each of these 6 ways, there exist 2 $P(A)_1^1$, which add up to the total

Way \ Time	1	2	3	4	Total $P(A)_i^1$
1	1	1	0	0	2
2	1	0	1	0	2
3	1	0	0	1	2
4	0	1	1	0	2
5	0	1	0	1	2
6	0	0	1	1	2
Total $P(A)_i^1$	3	3	3	3	12

Figure 3: An example: expanding $P(A)_2^4$.

of 2×6 (= 12) $P(A)_1^1$. These 12 $P(A)_1^1$ are evenly distributed over the time $1, 2, 3, 4$. So, for each time $1, 2, 3, 4$, there are $12/4$ (= 3) $P(A)_1^1$.

If we consider the actual number of transactions happen at each time $1, 2, ..., t$, it is equal to the number of transactions multiplies the probability that this number of transactions happen (as in token method):

$$\frac{i \times {}^tC_i}{t} \times \frac{P(A)_i^t}{{}^tC_i} = P(A)_i^t \left(\frac{i}{t}\right).$$

Now we consider the satisfaction level of the transactions happen at each time $1, 2, ..., t$. For disconnection time t, if disconnection happens at time 1, then reconnection will happen at time $t+1$. Recall from equation (2) that x is the duration of time between the time the transaction happens and the time it commits. In this case, we know that when a transaction happens at:

$$\begin{aligned} &\text{time } 1, \quad x = t, \\ &\text{time } 2, \quad x = t-1, \\ &\qquad \vdots \\ &\text{time } t, \quad x = 1. \end{aligned}$$

Hence, by referring to equation (2), the satisfaction level $f(x)$ for transactions happen at time 1 to t is:

$$\begin{aligned} f(t) &= \begin{cases} 1 - tK & t < \frac{1}{K} \\ 0 & otherwise, \end{cases} \\ f(t-1) &= \begin{cases} 1 - (t-1)K & t - 1 < \frac{1}{K} \\ 0 & otherwise, \end{cases} \\ &\vdots \\ f(1) &= 1 - K. \end{aligned}$$

Thus, the total satisfaction function, $S(A)_t^i$, is the number of transactions multiplies the satisfaction level as follows:

$$S(A)_i^t = P(A)_i^t \left(\frac{i}{t}\right) \times \sum_{j=1}^{t} f(j)$$

$$= \begin{cases} P(A)_i^t \left(\frac{i}{t}\right) \times \sum_{j=1}^{t}(1-jK) & t < \frac{1}{K} \\ P(A)_i^t \left(\frac{i}{t}\right) \times \sum_{j=1}^{\frac{1}{K}}(1-jK) & otherwise \end{cases}$$

$$= \begin{cases} P(A)_i^t \left(\frac{i}{2}\right)(2-(t+1)K) & t < \frac{1}{K} \\ P(A)_i^t \left(\frac{i}{2Kt}\right) & otherwise. \end{cases} \quad (4)$$

For $F(T)$, the evaluation function is $iP(A)_i^t$ for each $P(A)_i^t$, $(i = 1, 2, ..., t)$. For $F(O)$, the satisfaction level is integrated and the evaluation function is $S(A)_i^t$.

Thus, when transactions occur at only one site,

$$F(O) = P(B)_0^t \sum_{i=1}^{t} S(A)_i^t + P(A)_0^t \sum_{i=1}^{t} S(B)_i^t.$$

When transactions occur at both sites, as mentioned earlier, transactions happen at the site with the higher number of transactions will succeed, while transactions happen at the other site will have to experience rollback. Thus, for this case, when the number of transactions happen at site A exceeds that of site B,

$$F(O) = \sum_{i=1}^{t}\sum_{j=1}^{i} S(A)_i^t P(B)_j^t.$$

When the reverse happens,

$$F(O) = \sum_{i=1}^{t}\sum_{j=i}^{t} P(A)_i^t S(B)_j^t.$$

Thus, we get the following result:

$$\begin{aligned} F(O) = \; & P(B)_0^t \sum_{i=1}^{t} S(A)_i^t \\ & + P(A)_0^t \sum_{i=1}^{t} S(B)_i^t \\ & + \sum_{i=1}^{t}\sum_{j=1}^{i} S(A)_i^t P(B)_j^t \\ & + \sum_{i=1}^{t}\sum_{j=i}^{t} P(A)_i^t S(B)_j^t. \end{aligned} \quad (5)$$

When $F(T) > F(O)$, token method is used, and vice versa.

4 Evaluation and Discussion

It is important that for both token and optimistic method, we consider the trade-off between the work and the results.

The good thing about token method is that the success/failure of transactions is known on the spot without any delay. Only the sites with token are allowed to execute transactions. In other words, only transactions which will succeed are executed, while transactions which will not succeed are rejected straight away. Thus, there is no waste of work at all in executing the transactions. The efficiency is 100%.

However, there is no chance at all for the site without token to execute a single transaction. Thus the sites with lower probability of transactions always have no right to execute transactions during disconnection.

In optimistic method, the transactions are tentatively executed with the hope that eventually they will succeed. In some cases, they may succeed if no conflicts occur. This happens mostly when the probability of transactions are not high and/or the disconnection time is short.

However, the transactions executed may fail if conflicts occur, mostly in the case that the probability of transactions are high and/or the disconnection period is relatively long. Further, if the disconnection time is long, many uncommitted transactions accumulate and eventually they may all fail, resulting in a lot of waste of time and work.

It is hard to quantify the work of transactions rollback in optimistic method and compare it with the loss of not being able to execute transactions in token method. There is no exact barrier between these two methods in which we can say that "token method is better compared to optimistic method" and vice versa. It all depends on the circumstances.

However, by understanding the above phenomena, i.e., the advantages and disadvantages of both methods, a user can set his own priority and decide a barrier to define his own evaluation function. This is where the satisfaction level $f(x)$ and the constant parameter K play the role.

Depending on the priority, whether the number of transactions succeed is more important or the efficiency (result over work) is more important, the value of the constant K may be changed. The higher the value of K, the higher the probability that token method is applied. The user may set his own K to suit his own demand.

As an example, in the graph shown in Figure 4, the probability of transactions at site A and B, and the disconnection time t is fixed, and K is changed to reflect the change in the value of the evaluation function for token and optimistic method (refer to equation (3) and (5)). In this example, these values are set: $P(A)_1^1 = 0.3$, $P(B)_1^1 = 0.2$, $t = 5$. As can be seen in the graph, when $K \leq 0.095$ (approximately), the evaluation function chooses optimistic method, while when $K > 0.095$, it chooses token method.

Comparing our approach with the approach which

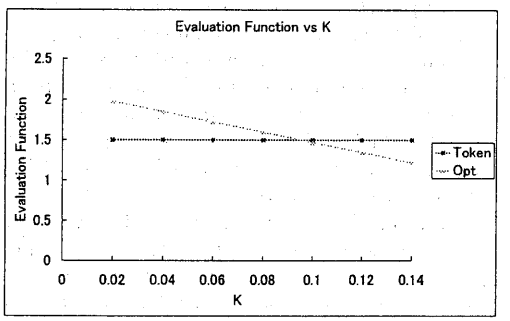

Figure 4: Change of value in the evaluation function with varying K with the following parameters: $P(A)_1^1 = 0.3$, $P(B)_1^1 = 0.2$, $t = 5$.

uses only one method, whether token or optimistic, certainly our approach is better as it balances off the advantages and disadvantages of both methods and gives the users a choice to decide the appropriate method to be used by considering the probability that transactions occur and the duration of disconnection time between the sites.

5 Conclusions

In this paper, we have proposed two approaches, namely token method and optimistic method to handle database updates during periods of disconnection. The former predicts the number of transactions which will happen based on the probability of the transaction and then decides the site which is allowed to execute transactions. The latter uses an optimistic approach where all the sites are allowed to run transactions, but rollback may happen if any conflicts occur. The latter always ensures that the number of transactions succeed is maximum, but it requires more work to be done.

We then proposed an algorithm to choose the most effective method between these two methods to be used during disconnection depending on the probability that transactions occur and the disconnection time. In order to choose the best method, the evaluation function in our algorithm uses the number of transactions succeed and the satisfaction level of each transaction. The users can set the appropriate parameter K in the formula of our proposed algorithm to suit their own demand depending on their priority.

In this paper, we only consider the case in which a network is disconnected into two sites. When they are further separated into more sites, the same algorithm can still be applied as long as the probability of transaction and the disconnection time is known. However, when reconnection happens in a random way, the situation becomes more complicated. We wish to look into this matter in the future.

On top of that, we would like to include the Escrow method ([5], [6], [7], [8], [9]) besides token and optimistic methods for partitionable data in our algorithm to make it more robust in the future.

Acknowledgement

This research was supported in part by the Research for the Future Program of Japan Society for the Promotion of Science under the Project "Advanced Multimedia Content Processing (Project No. JSPS-RFTF97P00501)" and by Grant-in-Aid for Scientific Research on Priority Areas "Advanced Databases" of the Ministry of Education, Science, Sports and Culture of Japan (Grant No. 08244103).

References

[1] Y. Breitbar and H.F. Korth, "Replication and Consistency: Being Lazy Helps Sometimes", Proc. of 16th ACM SIGACT-SIGMOD-SIGART Symposium on PODS 1997, pp.173–184, May 1997.

[2] A. Demers, K. Petersen, M. Spreitzer, D. Terry, M. Theimer, and B. Welch, "The Bayou Architecture: Support for Data Sharing among Mobile Users", Proc. of the Workshop on Mobile Computing Systems and Applications, December 1994.

[3] J.S. Heidemann, T.W. Page, R.G. Guy, and G.J. Popek, "Primarily Disconnected Operation: Experiences with Ficus", Proc. of the Second Workshop on Management of Replicated Data, IEEE, November 1992.

[4] J.J. Kistler and M. Satyanarayanan, "Disconnected Operation in the Coda File System", ACM Transactions on Computer Systems, Vol.10, No.1, February 1992.

[5] N. Krishnakumar and A.J. Bernstein, "High Throughput Escrow Algorithms for Replicated Databases (Extended Abstract)", Proc. of 18th International Conference on VLDB, pp.175–186, August 1992.

[6] N. Krishnakumar and R. Jain, "Escrow Techniques for Mobile Sales and Inventory Applications", Wireless Networks 3, Vol.3, No.3, pp.235–246, August 1997.

[7] A. Kumar, "An Analysis of Borrowing Policies for Escrow Transactions in a Replicated Data Environment", Proc. of 6th International Conference on Data Engineering, pp.446–454, February 1990.

[8] A. Kumar and M. Stonebraker, "Semantics Based Transaction Management Techniques for Replicated Data", Proc. of ACM SIGMOD'88, pp.117–125, June 1998.

[9] P.E. O'neil, "The Escrow Transactional Method", ACM Transactions on Database Systems, Vol.11, No.4, pp.405–430, December 1986.

[10] E. Pitoura and B. Bhargava, "Maintaining Consistency of Data in Mobile Distributed Environments", Proc. of 15th International Conference on Distributed Computer System, pp.404–413, May 1995.