Broadcasting Group Information to Ensure Consistency and Correctness in Mobile Computing Environments*

Daein Kim and Buhyun Hwang

Department of Computer Science Chonnam National University,
300 Yongbong-dong, Gwang-ju, Korea,
{dikim,bhhwang}@chonnam.ac.kr

Abstract. Recent advances in wireless communication technology and popularity of portable computers have rendered the mobile computing environments that users access the information via wireless channel, regardless of the situation or the location. In this environments, a mobile host have a cache to avoid the bandwidth usage and improve the response time of transactions. However, the cache mechanism does not guarantee the serializable execution of mobile transactions. In this paper, we propose the BGI-MT method that mobile hosts can make a commit decision by using the group information. Also our method can improve the cache efficiency in the case that the disconnection of mobile hosts is longer than the broadcast interval of the window report.

1 Introduction

In wireless mobile network, the server may exchange the data items and the control information with mobile hosts using broadcast strategies, but frequent broadcast technique requires high communication costs and causes a bottleneck due to the limited channel[1]. Therefore, caching technique is especially important to use the bandwidth efficiently and improve the response time of transactions in mobile computing environments which are characterized by narrow bandwidth, limited battery, host's mobility, and frequent disconnection from the server[3, 7]. In this strategy, a server broadcasts an invalidation report to ensure the cache consistency. However, it becomes difficult for the mobile hosts to know whether their cached data items are still valid or not before the arrival of an invalidation report. Thus, this method defers the commit decision of mobile transactions to ensure theirs serializable execution until the invalidation report is arrived[5]. In this paper, we find out the problems of cache invalidation methods and propose the method to solve them while ensuring the serializable execution of mobile transactions. Our method makes a commit decision by using the group information before receiving an invalidation report, and avoids discarding the entire cache in the case that the disconnection of mobile hosts is longer than the broadcast interval of an invalidation report.

* This work was supported by KOrea Science and Engineering Foundation(KOSEF R05-2003-000-10532-0)

2 Related Work

In [1], three strategies that use invalidation reports to fulfill the obligations has been suggested. In these methods, a server periodically broadcasts update information to ensure the cache consistency of a mobile host. But, these methods have a problem that the response time of mobile transactions can be long since mobile transactions is answered after receiving the next report[1].

In [4], UFO(Update First Order) detects inconsistency between cached data items in mobile hosts and the updated data items in the server. UFO protocol checks whether there is any overlap between the broadcast data set and the write set of update transactions during the current data broadcast period. If there is an overlap, the server knows that any conflicting data items have been existed and rebroadcasts the overlapped data items. But, this protocol may require additional cost since the broadcast schedule of conflicting data items should be reconstructed at the server. Also, if all conflicting data items are not rebroadcasted in one broadcast period, the similar conflict can be happened.

In [5], OCC-UTS^2(Optimistic Concurrency Control with Update TimeStamp Span) method is devised to guarantee the serializability of mobile transactions. In OCC-UTS^2 method, mobile hosts locally ensure the serializable execution of mobile transactions without server's intervention. Also a server does not handle the commit requests for mobile transactions issued by mobile hosts and thus the uplink traffic does not affect overall performance seriously. But, this method can make an incorrect commit decision of mobile transactions, as shown in Fig.1.

3 Problem Statement

Fig.1 shows the example that a mobile transaction reads inconsistent data items by uncontrolled data broadcast.

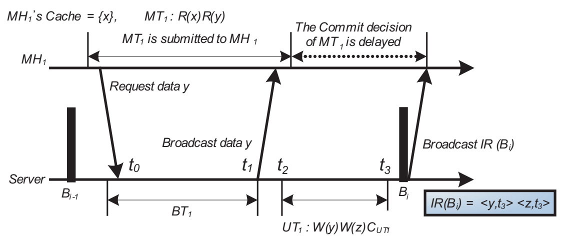


Fig. 1. Incorrect abort decision of a mobile transaction

In Fig.1, the mobile transaction MT_1 wants to read x and y, is submitted to the mobile host MH_1. As MH_1 has only data x in its cache, it sends the request message for y to execute MT_1. In the server, the broadcast transaction BT_1 is executed between t_0 and t_1, and makes the data broadcast schedule containing

y. The server broadcasts y at time t_1 and the update transaction UT_1 which updates y and z is executed between t_2 and t_3. Therefore the execution of MT_1 is serializable since MT_1 reads y which is not updated in this broadcast period. However, in the previous methods, the commit decision of MT_1 is deferred until received an invalidation report from the server. At time B_i, MT_1 receives the invalidation report and makes an abort decision since MT_1 accesses y which is involved in the invalidation report. But, it is an incorrect abort decision.

The biggest problem of invalidation report broadcast methods is that mobile transactions can read data items from theirs cache at any time, even if the data is being updated at the server. But mobile hosts do not know the update occurrence until receiving the invalidation report from the server. Therefore, the execution of mobile transactions have a potential possibility that mobile transactions access the different timing data, namely inconsistent data. In this paper, we propose the BGI-MT(Broadcast Group Information-for Mobile Transaction) method. In our method, a broadcast transaction is executed several times during an invalidation report broadcast period to improve the response time of mobile transactions. Also, a mobile host uses the group information to make a commit decision before receiving the invalidation report.

4 BGI-MT Algorithm

4.1 Discussion on Grouping

In [9], there are many issues that concern data grouping, namely, how data items should be grouped and the size of each group or the number of groups, and so on. In [9], entire database is basically grouped into four categories according to the update rate and the demand rate as shown in Fig.2.

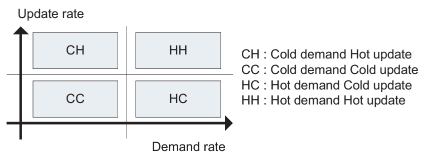

Update rate
CH
HH
CC
HC
Demand rate
CH : Cold demand Hot update
CC : Cold demand Cold update
HC : Hot demand Cold update
HH : Hot demand Hot update

Fig. 2. Basic data group

In Fig.2, the validity of data including in Hot update group, e. g. CH group and HH group is lower than data including in other groups since it is frequently updated. Also, the cache hit ratio of data including in Hot demand group, e. g. HC group and HH group is higher than data including in other groups. Similar to [9], we assume that entire database is basically grouped according to the update rate and the demand rate. Relatively, to improve the cache efficiency of our scheme, data items including in lower update rate group or higher demand rate group are divided into a number of groups. That is to say, the number of groups including HC category is the most, but the number of groups including CH category is the least. Also, additional data groups can be achieved by analyzing information about the past usage patterns with data mining techniques, e. g. association rule[2].

4.2 BGI-MT Method

In BGI-MT, a server broadcasts four types of reports, e. g. the invalidation report, the window report, the data report, and the group report. A server broadcasts the invalidation report $IR(B_i)$ to maintain the cache consistency of mobile hosts where B_i is a time point when the report is broadcast. The invalidation report $IR(B_i)$ is defined as follows.

Definition 1. *Invalidation Report* $IR(B_i)$*,*

$$IR(B_i) = \{[j, TS(j)]\,|j \in D, and\ B_i - L < TS(j) < B_i\}$$

D is a set of data items in the database, j is an identifier of data item that has been updated in the server during a broadcast period L, and $TS(j)$ is the most recent update timestamp of data j.

A server maintains the update information of data items to prepare for the disconnection of mobile hosts. This update information of data items is defined as the window report. If the window report broadcast interval is N times of the invalidation report interval L, the window report includes the update information of data items during $N \times L$. The window report $WR(B_i)$ is defined as follows.

Definition 2. *Window Report* $WR(B_i)$*,*

$$WR(B_i) = \{[j, TS(j)]\,|j \in D, and\ B_i - W < TS(j) < B_i\}$$

D is a set of data items in the database, j is an identifier of data item that has been updated in the server during a broadcast period W, and $TS(j)$ is the most recent update timestamp of data j.

In a server, a broadcast transaction collects the data information requested by mobile hosts, and makes a schedule for the next data broadcast. A server broadcasts the data report $DR(B_i)$ where B_i is a time point when data are broadcast. The data report $DR(B_i)$ is defined as follows.

Definition 3. *Data Report* $DR(B_i)$*,*

$$DR(B_{di}) = \{[j, TS(j)]\,|j \in D\}$$

D is a set of data items in the database, j is a data item that has been requested by mobile hosts during the execution of a broadcast transaction, and $TS(j)$ is the most recent update timestamp of data j.

In our scheme, data items of database are grouped according to the update rate, access rate, and update pattern, and so on. The group report $GR(B_i)$ is defined as follows.

Definition 4. *Group Report $GR(B_i)$,*

$$GR(B_i) = \{[k, TS_l(k)]\,|k \in G\}$$

G is a set of data groups, k is a group identifier of data items that have been updated in the server, and $TS_l(k)$ is the most recent update timestamp about data items of group k.

In our scheme, if the current available bandwidth is sufficient, a server broadcasts the group report with the data report to improve the response time of a mobile transaction. Also, the group report is broadcast with the window report that is composed the most recent update timestamp of entire group to prevent the entire cache dropping. Also, a mobile host MT keeps its read data set $ReadSet(MT)$ as follows, to decide a immediate commit decision of MT.

Definition 5. *$ReadSet(MT)$,*

$$ReadSet(MT) = \{d_1, d_2, ..., d_i \in D(1 \leq i \leq n)\}$$

MT is the mobile host identifier, D is a set of data items in the database, and $d_i(1 \leq i \leq n)$ is the identifier of data items accessed by MT.

If one of the following three types of condition is satisfied, our algorithm makes a commit decision and a mobile host commits its mobile transaction immediately. We assume that B_{IL} is the last broadcast time of an invalidation report and G_{MT} is the list of group including $ReadSet(MT)$.

Immediate Commit Decision Condition
Type 1.All data items in $ReadSet(MT)$ have a equal timestamp.
Type 2.The Timestamps of all data items in $ReadSet(MT)$ are less than B_{IL}.
Type 3.The Timestamps of all data items in $ReadSet(MT)$ are greater than $MAX\{TS_l(j), j \in G_{MT}\}$ or not involved in the group report.

If all data items accessed by a mobile transaction MT have a single timestamp TS_c, it has accessed the snapshot of database at time TS_c. We think that a snapshot of database is consistent. Thus, in the case of Type 1, MT can be immediately committed. In the case of Type 2, MT sees the snapshot of database at time B_{IL}. This means that one of the $ReadSet(MT)$ is cached in the current invalidation report broadcast period, but it is not updated in this period. Thus, in the case of Type 2, MT can be immediately committed. In the case of Type 3, MT sees the latest snapshot of data items. That is to say, MT cached the data items after update transactions have been completed. If each data item in $ReadSet(MT)$ has been cached after the commit of the most recent update transaction, its cache timestamp is bigger than the $MAX\{TS_l(j), j \in G_L\}$. Thus, if

```
Input:    $B_{IL}$ : The last broadcast time of an invalidation report
          ReadSet(MT) : The read set of a mobile transaction MT
          $G_{MT}$ : List of groups including ReadSet(MT)
          TSc(x) : Timestamp of data x in mobile host

Output:   Commit decision of MT

begin
  Step 1: MT executes its all operations
  if ReadSet(MT) has a single timestamp then
     Commit MT;
  else
     Delay the commit decision of MT until receiving the group report;
  endif
  Step 2: MT received the group report from the server
  if $G_{MT}$ is not included in the group report then
   Commit MT;
  else if TSc(x) of every data x in ReadSet(MT) < $B_{IL}$ then
     Commit MT;
  else if TSc(x) of every data x in ReadSet(MT) > MAX{[$TS_l(G_{MT})$]} then
     Commit MT;
  else
     Delay the commit decision of MT until receiving an invalidation report;
  endif
end
```

Fig. 3. Commit decision algorithm of BGI-MT

Type 3 condition is satisfied, MT sees a consistent snapshot of a database. In BGI-MT, the commit decision of a mobile transaction is shown in Fig.3.

If the execution of a mobile transaction does not satisfy at least one of three types of condition, the commit decision of a mobile transaction is delayed until receiving the invalidation report from the server. Assume that a mobile host MH only has a data item x included group j in its cache. If MH is disconnected and reconnected, MH waits for listening a window report from its server to verify the consistency of cache content. When MH receives a window report, it compares its disconnection period with the broadcast period of the window report. In [1, 5], if the disconnection period of MH is greater than the broadcast period of a window report W, MH drops the all cache contents. However, in BGI-MT, if the data timestamp of included group j is bigger than the timestamp $TS_l(j)$ of group report, we can ensure the validity of data items included group j since $TS_l(j)$ means the last update time of data items of included group j. Fig.4 shows an example that the disconnection period of a mobile host is longer than the broadcast period of a window report, but a mobile host avoids discarding the entire cache contents in our scheme.

We assume that data x and y are included in group G_1 and G_2, respectively. Also All update timestamps of G_1 and G_2, e. g. $TS_l(G_1)$ and $TS_l(G_2)$ are t_0 initially. In Fig.4, a mobile host MH_1 sends the request message of data x and y and the server broadcasts the data report containing data x and y at time t_1 and t_2, respectively. Thus, timestamps of x and y in MH_1's cache are all t_0 which is the initial timestamp in a server. Suppose MH_1 disconnects at time t_3 and

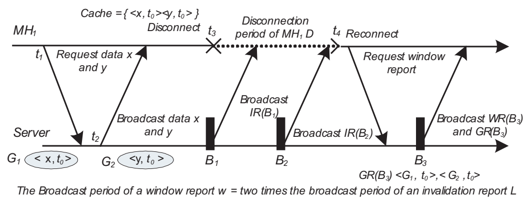

Cache ={ <x, t0><y, t0> }
MH1
Disconnect
t3
Disconnection period of MH1 D
t4
Reconnect
t1
Request data x and y
Request window report
Broadcast IR(B1)
Server
t2
Broadcast data x and y
Broadcast IR(B2)
Broadcast WR(B3) and GR(B3)
G1 < x, t0>
G2 <y, t0 >
B1
B2
B3
GR(B3) <G1, t0>,< G2 , t0>
The Broadcast period of a window report w = two times the broadcast period of an invalidation report L

Fig. 4. After disconnection, cache management example

reconnects at time t_4. Also, a server broadcast the invalidation report $IR(B_1)$ and $IR(B_2)$ at time B_1 and B_2(where $t_3 < B_1 < B_2 < t_4$) during disconnection of MH_1(between t_3 and t_4), respectively. Let the period of a window report W is two times the period of an invalidation report L. After reconnection time t_4, MH_1 requests the window report to verify the its cache contents. In our scheme, a server broadcasts the window report $WR(B_3)$ and $GR(B_3)$ at time B_3, and MH_1 verifies its cache validation In BGI-MT, timestamp $TS_l(G_{id})$ of group report means the last update time of data items included group G_{id}. In Fig.4, if data items of group G_1 and G_2 are not updated in the disconnection of MH_1, update timestamps of G_1 and G_2 are t_0. Thus, even if the disconnection period of MH_1 D exceeds the period of a window report W, MH_1 does not drop x and y since their timestamps are equal to the timestamps of group report $TS_l(G_1)$ and $TS_l(G_2)(t_0 = t_0)$, respectively.

5 Analytical Model

In this section, we analyze the response time and the bandwidth usage of mobile transactions. As a comparison baseline, we choose UFO method and OCC-UTS^2 method since two methods ensure the serializable execution of mobile transactions and maintain the cache consistency by receiving the invalidation report and they are similar to our method. Similar to [1, 5], we begin by stating some assumptions of our model:

- The server broadcasts an invalidation report per every L seconds.
- Each mobile transaction accesses to n data items in average.
- The access rate of each data item is λ(follow an exponential distribution).
- The update rate of each data item is μ(follow an exponential distribution).
- P$_{OCC}$ and P$_{BGI}$ are the probability that the execution of a mobile transaction meets the immediate commit condition in OCC-UTS^2 and BGI-MT, respectively.
- The average data caching time $MT_{caching}$ for a mobile transaction accessing n data items can be computed with the following equation, is $n \times (1 - h) \times (data\ transfer\ time)$.

5.1 Average Response Time of a Mobile Transaction

If the invalidation report is broadcast per every L seconds, the average delay time of a mobile transaction is $\frac{L}{2}$ seconds. In UFO method, the average response time of mobile transactions RT$_{UFO}$ is computed as:

$$RT_{UFO} = MT_{caching} + \frac{L}{2} \tag{1}$$

In OCC-UTS^2 method, we can compute the probability that a mobile transaction leads to a successful commit by immediate validation. In order for a mobile transaction to be committed by immediate validation, data items accessed by a mobile transaction should not be updated, even if cached in current invalidation report period. In OCC-UTS^2 method, the average response time of mobile transactions RT$_{OCC}$ is computed as:

$$RT_{OCC} = MT_{caching} + \frac{L}{2} \times (1 - P_{OCC}) \tag{2}$$

In BGI-MT method, if the execution of a mobile transaction MT meets at least one of the three types of condition, MT is committed immediately. Suppose data items of groups including $ReadSet(MT)$ are not updated at least E_1 seconds or E_2 seconds such as Fig.5.

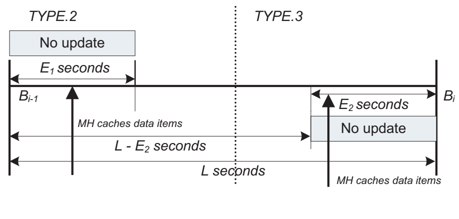


Fig. 5. The reason for response time in BGI-MT

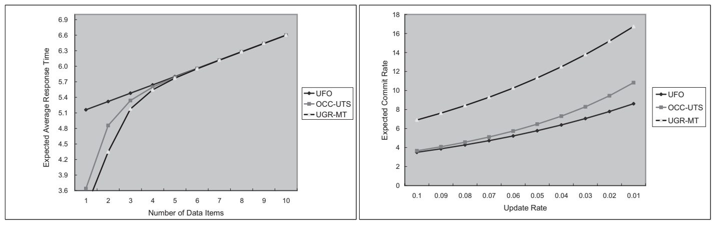
In BGI-MT method, immediate commit condition Type 1 is equal to the immediate commit condition of OCC-UTS^2 method. Fig.5 shows the case that the execution of a mobile transaction meets the immediate commit condition Type 2 or Type 3. In order for a mobile transaction to meet condition Type 2 or Type 3, a mobile transaction should cache data items during E_1 seconds or E_2 seconds, respectively. Assume that cached data items are not updated during E_1 seconds or E_2 seconds, respectively. Thus, we can compute the average response time of mobile transactions in BGI-MT method, RT$_{BGI}$ is computed as:

$$RT_{BGI} = MT_{caching} + \frac{L}{2} \times (1 - P_{BGI}) \tag{3}$$

Based on these analysis, we compute the response time and the commit rate of mobile transactions on various methods. Similar to the parameter setting of [1, 5], our analysis is performed based on the scenarios as follows.

Scenario.1
A mobile client access the information, such as weather, stock, location, or traffic information at the rate of $0.1(\lambda = 0.1)$ and this information is changed at the rate of $0.005(\mu = 0.005)$.

Scenario.2
A mobile client access the information at the rate of $0.01(\lambda = 0.01)$ and a mobile transaction access to three data items.



(a) The response time of a mobile transaction (b) The commit rate of a mobile transaction

Fig. 6. The response time and the commit rate on various schemes

From the results shown in Fig.6, we observe that the smaller the number of data items accessed by a mobile transaction n or the update rate μ, the better performance of BGI-MT. This improvement comes from the two points. First, our scheme can make a commit decision before receiving the invalidation report against other schemes. This decision can reduce the possibility of false invalidation and may well improve the response time of a mobile transaction. Second, immediate commit decision case in our scheme will be a great opportunity for transaction waited its verification to reduce the possibility of the incorrect abort decision, such as Fig.1. Incorrect abort decision is a cause that the restart of mobile transactions and limited bandwidth usage are increased and system performance is lower.

5.2 Bandwidth usage of our scheme

In our scheme, a server will broadcast the additional control information e.g. the group report to improve the response time of mobile transactions and the cache efficiency of mobile hosts, against other schemes. This control information may cause the bottleneck of wireless network by increasing the limited bandwidth usage. However, in our scheme, this information is selectively transmitted from the server to mobile hosts if only the current available bandwidth is sufficient. Also, if the size of group report is large, a server will broadcast partially the control information which is particular group-centered e.g. the group of HC category. Thus, the overhead of additional broadcast will not be crucial to the system performance since it is optional control information. Also, our scheme can reduce the false invalidation caused by the disconnection of a mobile host. In previous schemes, these drawbacks lead to transmit the additional data and increase bandwidth usage, one the other hand our scheme can minimize these problems.

6 Conclusion and Future Work

In this paper, we have proposed a new method for guaranteeing the serializable execution of mobile transactions, called BGI-MT, using the data group information. BGI-MT method for improving the response time of mobile transactions makes a commit decision by using the group information of data items before receiving an invalidation report from the server. Also, our scheme can prevent the entire cache dropping, even though the disconnection of a mobile host is longer than the broadcast interval of a window report. Through the analytical model, we have shown that our method is superior to other methods, in terms of the average response time and the commit rate of mobile transactions. For the future work, we have to study a data grouping method which is an important issue in our scheme, particularly to improve the response time of mobile transactions and to reduce the communication.

References

1. D. Barbara and T. Imielinski, "Sleepers and Workaholics: Caching Strategies in Mobile Environments(Extended Version)," VLDB Journal Vol.4, No.4, pp.567-602, 1995.
2. J. Han, J. Pei, and Y. Yin, "Mining Frequent Patterns without Candidate Generation," In Proc. ACM SIGMOD Conference on Management of Data, pp.1-12, May 2000.
3. S. Kim, S. Yang and S. Lee, "Maintaining Mobile Transactional Consistency in Hybrid Broadcast Environments," ACTA Information Vol.41 pp.65-81, Aug. 2004.
4. K. Lam, M. Au and E. Chan, "Broadcasting Consistent Data to Read-Only Transactions from Mobile Clients," The Computer Journal, Vol.45, No.2, pp.129-146, 2002.
5. S. Lee, "Caching and Concurrency Control in a Wireless Mobile Computing Environments," IEICE Transaction on Information System, Vol.E85-D, No.8, pp.1284-1296, Aug. 2002.
6. J. B. Lim and A. R. Hurson, "Transaction Processing in Mobile, Heterogeneous Database Systems," IEEE Transactions on Knowledge and Data Engineering, Vol.14, No.6, pp.1330-1346, Nov. 2002.
7. C. Lin, H. Hu, and D. Lee, "Adaptive Data Delivery in Wireless Communication Environment," Wireless Networks, Vol.10, pp.103-120, March 2004.
8. N. Prabhu, V. Kumar, I. Ray and G. Yang, "Concurrency Control in Mobile Database Systems," Proc. AINA2004 18th International Conference, Vol.2 pp.83-86, March 2004.
9. K. L. Tan and J. Cai, "Broadcast-Based Group Invalidation : An Energy-Efficient Cache Invalidation Strategy," Information Sciences, Vol.100, pp.229-253, Aug. 1997.