ISSN 2278-3091

International Journal of Advanced Trends in Computer Science and Engineering, Vol. 3 , No.3, Pages : 01 - 05 (2014)
Special Issue of ICIITEM 2014 - Held during May 12-13, 2014 in PARKROYAL on Kitchener Road, Singapore

Distribution of Updated Data in Wireless Broadcasting Environment

Ahmad al-Qerem*
* Department of Computer Science, Zarqa University, Jordan *E-mail:ahmad_qerm@zu.edu.jo

Abstract: *Wireless data broadcast is a popular data dissemination method in mobile computing environments because of its capability of concurrently disseminating data to multiple users. When an application involves wireless mobile clients that run multiple-operation transactions and dynamically update the server database, those updates have to be appear in the next broadcast cycle. Earlier show up of such updates is highly improving the performance. In this paper we aim to decompose the main broadcast cycle into sub cycles which only contain a subset of data items in the database. This decomposition contains both the original data to be broadcast and the updates come from the committed transactions on these data items. Perfecting and Back off at sub cycle level will be used for further improvements. At sub cycle level, it is more powerful for both update and read-only transactions which allow more than one conflicting transactions to be committed on the same broadcast cycle. Moreover, by using sub cycle the currency of the data may be higher since the delays in performing the updates are shorter.*

Key words: data broadcast, mobile transaction, concurrency control, cycle decomposition, back off**.**

INTRODUCTION

Data broadcast is becoming a promising way to disseminate information to a large population of mobile clients by mean of transaction. Unlike the conventional client server approach, where a data item have to be send many times to deliver the requested data even in the case of read –only transactions. Broadcast has the potential to satisfy all outstanding requests for the same data with a single response. It increases the efficiency of shared bandwidth and improves the system throughput. On the other hand, the presence of update transactions make the management of transactions over such environment is non trivial and revisiting the conventional concurrency protocol is must. In more specific, the data broadcast model requires the minimal usage of the uplink communication from mobile clients to the server. This makes conventional schemes such as 2-phase locking and time stamp ordering concurrency control are not applicable to control concurrency in broadcast-based transaction processing because all these methods require extensive communication among clients and server. There have been many research efforts reported in the literature that tackle the concurrency problems in wireless broadcast environments, such as Update First Ordering (UFO) [2], Multi-Version Broadcast [3], Serialization Graph [5], Broadcast Concurrency Control with Time Stamp Interval (BCC-TI) [2], F-Matrix [1], and Certification Report [4, 6]. The drawbacks of these methods have been analyzed in [9, 7]. In general, some of these methods only support client read-only transactions, and some of them could have substantial processing overhead.

To the best of our knowledge, this is the first effort on the study of cycle decomposition to support concurrency control in presence of update transactions over a broadcast environment. The major contribution of this research is two-fold. First of all, we have identified and analyzed the performance problem, namely one mobile update transaction is allowed to commit during one broadcast cycle problem. Furthermore, combining back-off technique with cycle decomposition approach called CCBR-SL in scheduling the mobile transactions to perform an effective scheduling in terms of up-link bandwidth utilization and power consumption. According to the extensive analysis and comprehensive performance evaluation, the proposed approach shows a satisfactory performance in transactions processing on broadcast environments.

The rest of this paper is organized as follows: Section II summarizes the related works. Section III motivate our work by evaluating the random back –off at different broadcast cycle length. Section IV describes how broadcast cycle decomposition could be achieved. Section V proposes a new CCBR-SL approach and its algorithm. The simulation model and the performance evaluation metrics. The experimental results discussed in Section VI demonstrate the problem suffered by existing algorithms. Finally, we conclude the paper in Section VII.

RELATED WORKS

Concurrency control using random back-off at sub cycle level (CCRB-SL) method is based on the optimal concurrency control method. Optimistic concurrency control method well suits for wireless broadcast environments. Usually, there are two ways of validating transactions in optimistic concurrency control methods, forward validation and backward validation. In a backward validation, the validating process is done against committed transactions. If there is a conflict, the validating transaction has to be aborted. In a forward validation, the validating process is done against the currently running transactions. If there is a conflict among them, either the validating transaction or the currently running transactions relevant to the conflict have to be aborted. Normally, it is more expensive to abort the validating transaction than to abort

ISSN 2278-3091

International Journal of Advanced Trends in Computer Science and Engineering, Vol. 3 , No.3, Pages : 01 - 05 (2014)
Special Issue of ICIITEM 2014 - Held during May 12-13, 2014 in PARKROYAL on Kitchener Road, Singapore

the currently running transactions. Thus, it is better to abort the currently running transactions rather than the validating transaction. CCRB-SL method uses the partial backward validation and the forward validation for the concurrency control of mobile transactions. In CCRB-SL method, all transactions at the mobile clients including read-only transactions and update transactions must perform partial backward validations by accessing the Ucast Cycle. In wireless broadcast environments, the Ucast consists of the set of data items that was updated during the previous subcycle with its new values. If a transaction fails the partial backward validation, it will be aborted and restarted. The partial backward validation is performed against the committed update transactions at the server during the previous broadcast cycle. At the server, the forward validation of a validating update transaction is carried out against currently running transactions. The validating update transaction at the server is either a server transaction or a mobile update transaction submitted by the mobile clients. Forward validation is suitable for wireless broadcast environments with the following two reasons. First, a validating mobile transaction should not be aborted and restarted as much as possible when its validation is performed at the server. This is because the abortion and restart of the validating mobile transaction would consume a large amount of the computing resources of the corresponding mobile client. With forward validation, the server can avoid selecting the validating mobile transaction for abort if there is any conflict. Second, a mobile client only has to send a small amount of information to the server for the forward validation of mobile transaction. The write data set of the validating transaction is enough for the forward validation process. As read-only mobile transactions do not have any write data set, they can be locally committed without sending any validation requests to the server. Therefore, using forward validation helps reducing the uplink bandwidth usage for mobile clients. To improve the response time of mobile transactions, CCRB-SL uses data prefetching technique [8] where mobile client keeps a consistent data items accessed in its cache while it is executing in order to reuse the unchanged data if its force to be restart. Data perfecting improves the response time of the mobile transaction by reducing the access time for the required data items. This is because the mobile transaction can retrieve the required data items directly from its cache rather than waiting for them to appear on the broadcast channel. Since CCRB-SL utilizes the cache for data prefetching, it is necessary to maintain the consistency between server data and the client's cached data. CCRB-SL make use of Ucast data cycle to maintain the consistency as we will explain later. CCRB-SL method uses a random back-off technique at subcycle level to allow more conflicting transactions to be committed in one broadcast cycle.

To prevent the repetitive aborts and restarts, a random back-off technique has been introduced in [7]. In this technique, a server provides the update contention degree on the data item x to the mobile clients and the mobile clients use that information to execute the random back-off technique. With such random back-off technique, restarting schedules of the mobile update transactions are distributed as shown in Fig. 1b instead of repetitive aborts resulting in more consumption of uplink and downlink bandwidths of mobile clients as well as their battery powers Fig. 1 (a). Unfortunately, still there is only one mobile update transaction is allowed to commit during one broadcast cycle.

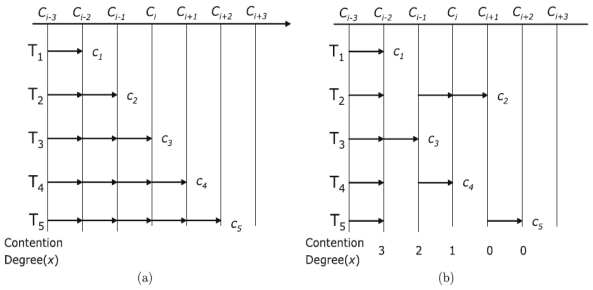

Fig. 1: Random back-off at broadcast cycle level [9]

Fig. 1 (a) illustrates this problem by using five mobile update transactions, T1; T2; T3; T4 and T5 where they are competing to update data item x during the broadcast cycle Ci_3. T1; T2; T3; T4 and T5 may be locally aborted due to the overlapping of the read set of one transaction and the write set of the other. Assume that the mobile transactions have a high access invariance and the mobile clients for T1; T2, . . ., T5 prefetches the necessary data items into their caches. As a result, restarting transactions can read all required data in one broadcast cycle. Though many factors can cause some delay, let's assume every restarting transactions read all data in one broadcast cycle by accessing the prefetched data items in the caches. Since only one mobile update transaction is allowed to commit during one broadcast cycle, T1 commits at broadcast cycle Ci_2, and the others (i.e., T2; T3; T4; T5) are aborted and restarted. At the next broadcast cycle Ci_1; T2 commit and the other 3 transactions has to abort and restart again. After five broadcast cycles, all 5 transactions are finally committed resulting in 10 restarts of 4 transactions. This continuous aborting problem gets more serious as their access patterns get more skewed. That is, it consumes more uplink and downlink bandwidths of mobile clients as well as their battery powers.

In Fig. 1b, the server detects an update conflict on data item x by five mobile update transactions during the broadcast cycle Ci_3. At the beginning of Ci_2, the server allows for T1 to commit at broadcast cycle Ci_2, and the others (i.e., T2; T3; T4; T5) are aborted and restarted. Then, at the beginning of broadcast cycle Ci_2, the server sends 3 as the contention degree on the data item x to each mobile clients with the aborted transactions T2; T3; T4, or T5. Each mobile client generates its random back-off time number w between 0 and 3. Each mobile client restarts its transactions after passing w broadcast cycles. For example, in Fig. 1(b), T4 generates a random back-off time number of 1. It waits for a single broadcast cycle, and restarts at Ci_1, and

ISSN 2278-3091

International Journal of Advanced Trends in Computer Science and Engineering, Vol. 3 , No.3, Pages : 01 - 05 (2014)
Special Issue of ICIITEM 2014 - Held during May 12-13, 2014 in PARKROYAL on Kitchener Road, Singapore

commit at Ci. This technique can effectively reduce the repetitive aborts and restarts of mobile update transactions.

RANDOM BACKOFF AT SUB CYCLE LEVEL

Random back off is best known in the context of the Ethernet access-control framework. In the context of mobile transaction in wireless broadcasting environment, it is a technique used to alleviate repeated transaction conflicts on a same data item thereby saving scarce uplink bandwidth. It can also be used even in read-only mobile transactions to defer an access prior to actual conflict and thereafter elide the conflict entirely. There seems to be a general consensus that back off is useful as it's wildly used in different context for the same reason (i.e. conflict avoidance) a little works on back-off technique in the context of broadcasting environment have been done. In this section, we motivate our work by evaluating random-back off in this context; we study three simple contention scenarios, high, moderate and low for different broadcast cycle length. In this workload there are different transactions lengths, long transactions aborted by short transactions may help to increase the validity of the data within the broadcast cycle allowing more read only transaction to be locally committed. With prefetching technique long transactions need to be restarted on the data items being in conflict with previously commited transactions which may provide early in the next sub cycle resulting in more concurrency for both read only and update transactions.

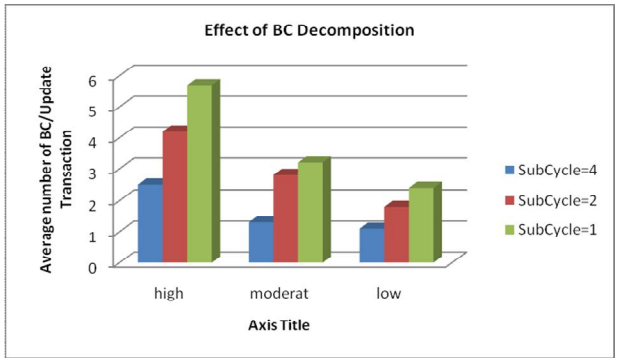

Fig.2. Average Number of broadcast cycle till the update transaction gets committed.

For transaction that detected the conflict just loses the conflicted data items and restarted locally. At sub-cycle level, those data items may appear directly in the same broadcast cycle which means that there is no need for both read-only and update transaction to be delayed to the next broadcast cycle. Further more; back-off just prior to a commit at broadcast cycle level cannot evade write-write conflicts between backed-off transaction and the newer one since by the commit point both transactions have already performed their respective updates. In this situation, one of them has to restarted and back-off again. If the broadcast cycle is large enough, random back off technique at the cycle level become substantially inefficient as the transaction need to span a multiple large broadcast cycle in order to get their update being reflected and available to other transactions. Random-back-off at sub cycle level helps avoid both convoy and starvation problems. Fig.2. shows the performance plots. Y-axis plots average number of broadcast cycle till the Update transaction gets committed. Each bar in the plot represents number of sub cycle in the main broadcast cycle. Performance results show that the overhead of broadcast cycle decomposition can be kept low while providing a considerable increase in concurrency. Besides increasing the concurrency of client transactions, decomposition of broadcast cycle provides clients with the possibility of accessing multiple server states in a single broadcast cycle. Furthermore, decomposition always supports the up to date data items for client transactions where ever it's come in the broadcast period.

This study is important because in broadcast data model maximizing data currency (minimizing staleness) and decreasing transaction's aborts are of equal importance as providing consistent data items to transactions. Integrating back off into concurrency protocols at sub cycle level is not straightforward. Its require the main broadcast cycle to be lengthen which can be coped over by using a proper indexing method. In the next section we will explain how such decomposition could be achieved.

BROADCAST CYCLE DECOMPOSITION

As we see in previous section, an efficient way to improve the performance of broadcast data model is to relax the requirement that each data item has to be broadcast at least once in each cycle. We may decompose the main broadcast cycle into sub cycle which may only contain a subset of data items in the database. More over, by using sub cycle the currency of the data may be higher since the delays in performing the updates are shorter. The broadcast cycle is divided into multiple sub-cycles of two types alternatively see Fig.3: read cast called Rcast and update cast called Ucast in alternative way. The former contains the data items which were predefined scheduled for the main broadcast cycle but distributed along many Rcast where as the Ucast content is dynamics and changed based on the data being updated by the committed transactions in the previous sub cycles.

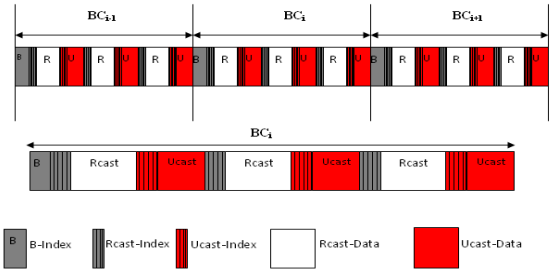

Fig.3: Broadcast Cycle Decomposition

Between any two Rcast cycles, there is a reserved space for Ucast to accommodate identities for all the data objects which are updated by transactions in the server after the first sub-cycle begins. A mobile transaction can validate its prefetched data consistency autonomously by accessing the

ISSN 2278-3091

International Journal of Advanced Trends in Computer Science and Engineering, Vol. 3 , No.3, Pages : 01 - 05 (2014)
Special Issue of ICIITEM 2014 - Held during May 12-13, 2014 in PARKROYAL on Kitchener Road, Singapore

Ucast cycle. Our goal is to reveal the write set of the committed transactions as soon as possible. Unfortunately, this may jumble the original schedule. We can tradeoff the immediate show up of the updated data items from the previous sub cycle and delaying all those updates until the beginning of a next sub cycle which will be accommodated in front of pre assigned Rcast index. As a result, the identities of all the data items which are updated by any transactions are included in the Ucast space after the current broadcast sub-cycle. The identities of all the extra data items with no rooms at the current Ucast are included in the next Ucast based on commit sequence of their transactions. At the same time, since all information of latest committed update transactions is dynamically loaded at the beginning of some Rcast index segment, a mobile transaction only needs to tune to the channel at specific periods to retrieve the requested data item and the control information associated with it.

Data broadcast usually requires a client to be active all the time in order to monitor the data units that go by. This leads to unacceptable energy consumption on wireless mobile equipments, for which power saving is a very essential issue. To save power in data broadcast models, indexing schemes are proposed in [9]. The Basic idea of indexing is to insert pointers for data broadcast in a future schedule into a broadcast cycle. Consequently, a client application can go to doze mode after it accesses this pointer, and only wakes up at the time the requested data unit is on the air. Several index schemes have been proposed in [8]. In all indexing schemes, an index tree of all data in a broadcast cycle is inserted to the schedule. Pointers to each real data units are located at the leaves of the index tree while a route to a specific leaf can be found following the pointer from the tree root. In our work the B-index contains only information about the starting time of both Rcast-index and Ucast-index based on the number of sub cycles and the sizes for each of them; this is easy know as all Ucast are of equal size to and the Rcast cycles is previously determined at the beginning of each broadcast cycle. In addition to the index information provided by Rcast-index and Ucast-index, Rcast-index contains a pointer for the next Rcast. the Ucast-Index contains the actual index of the data items to be broadcasted and each index is associated with control information to validate its pre fetched data items such as sub cycle number as well as the conflicting degree which will be used in back-off procedure as we will be explain later in this paper. Fig.4 shows the structure of index information for each element in Ucast-index in addition to the pointer indicating the next Rcast-index segment as the client may arrive at any time during the major broadcast cycle.

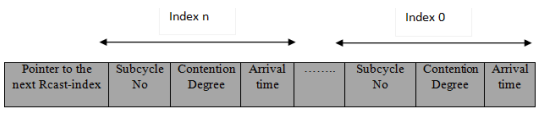

Fig.4: Information associated with each index in Ucast cycle

CCBR-SL APPROACH

As we see in section 3, the random back-off will be a promising approach if we applied at the sub cycle level. A new back-off algorithm with appropriate distribution of updated data items based on conflicting degree along the Ucast cycles called CCBR-SL Fig.5 will be introduced. conflicting degree also important for read-only transaction that is the transaction need to read the same data item multiple times equal to the number of updated transactions so back off work as locking technique although that there is no need for such read only transactions to communicate with the server (i.e. self locking).

Each mobile transaction reads all its requested data items directly by tuning into Rcast cycle, the mobile transaction may need to check different Ucast cycles in order to collect their data items. The up coming Ucast-index also needs to be checked in order to ensure that its previously accessed data items are consistent. No transaction can be committed in a certain sub cycle unless its data could be placed on a certain Ucast cycle. This can increase the probability of read only transaction to commit as we defer the commit of conflicting transaction to the next sub cycle if there is no space to accommodate the updated data items in the next Ucast, the commit is deferrable to Ucast with available space to publish the updated data items.

Input:

$UD^i = \{ d_1^i, d_2^i, \ldots, d_n^i \}$ be a set of updated data items in $Rcast^i$

Conflict degree (d_j^i) = the number of failed to commit transactions because of d_j^i trying to update d^i at Rcast $_j$.

$Ucast^{NumSc}$ =Ucast cycle number within the major broad cast cycle.

Output: The data broadcast schedule for updated data items over Ucast in BC

Step1: Sort the data d_j^i in UD by conflict degree (d_j^i) in descending order;

Let d_j^i represent i^{th} sorted data in descending order of Rcast $_j$.

Step 2: while ($UD^i \neq \Phi$ & k < total number of Ucast in BC) **do**
J=0
k= (current $Ucast^{NumSc}$ + 1) **to** remaining Ucast cycles in BC {
For i= 1 to i<= n, i++ do { //n: the Ucast Size
J=j+1
If j>n break; n: size of Ucast exceeded
Allocate d_j^i in $Ucast^{NumSc}$ =k
} **End For i**
K=k+1 **}**
End while

Fig .5: DUD algorithm

CONCLUSION

Earlier show up of such updates is highly improving the performance. In this paper we propose a decomposition approach of the main broadcast cycle into sub cycles which only contain a subset of data items in the database. This decomposition contains both the allow us to apply a concurrency protocol with back-off at sub cycle level. At sub cycle level, it is more powerful for both update and read-only

transactions which allow more than one conflicting transactions to be committed on the same broadcast cycle. Moreover, by using sub cycle the currency of the data may be higher since the delays in performing the updates are shorter. We are working now on developing a simulator to compare the result with FBOCC protocol. This will be the future work of this study.

ACKNOWLEDGEMENT

This research is funded by the Deanship of Research and Graduate Studies in Zarqa University /Jordan"

REFERENCES

[1] Lam, K. Y., Au, M. W., and Chan, E., "Broadcast of consistent data to read-only transactions from mobile clients," Proc. of the Second IEEE Workshop on Mobile Computing Systems and Applications, 1999.

[2] Pitoura, E., "Supporting read-only transactions in wireless broadcasting," Proc. of the DEXA98 International Workshop on Mobility in Databases and Distributed Systems, pp. 428-422, 1998.

[3] Pitoura, E., and Chrysanthis, P. K., "Scalable processing of read-only transactions in broadcast push," Proc. Of the 19th IEEE International Conference on Distributed Computing System, 1999.

[4] Lee, SangKeun, Hwang, Chong-Sun, Kitsuregawa, Masaru,. Efficient, energy conserving transaction processing in wireless data broadcast. IEEE Transactions on Knowledge and Data Engineering 18 (9), 1225–1237. September 2006.

[5] Cho, H. Concurrency control for read-only client transactions in broadcast disks. IEICE Transactions on Communications E86-B (10), 3114–3122. 2003

[6] Lee, Victor C.S., Lam, Kwok Wa, Kuo, Tei-Wei, Efficient validation of mobile transactions in wireless environments. Journal of Systems and Software, 183–193. 2004.

[7] Sunggeun Park, Sungwon Jung; An energy-efficient mobile transaction processing method using random back-off in wireless broadcast environments The Journal of Systems and Software vol 82 pp 2012–2022, 2009

[8] Vikas Goel, Ajay Kumar Anil Kumar Ahlawat; A Comparative Study of Energy Efficient Air Indexing Techniques for Uniform Broadcasting International Journal of Computer Applications COMNET-2011

[9] Huang, Y., and Lee, Y. H., "Concurrency control protocol for broadcastbased transaction processing and correctness proof," ISCTA PDCS 2001,in press, August 2001.

Ahmad Alqerem obtaining a BSc in 1997 from JUST University and a Masters in computer science from Jordan University in 2002. PhD in mobile computing at Loughborough University, UK in 2008. He is interested in concurrency control for mobile computing environments, particularly transaction processing. He has published several papers in various areas of computer science. After that he was appointed a head of internet technology Depts. Zarka University