Using Predeclaration for Efficient Read-only Transaction Processing in Wireless Data Broadcast

SangKeun Lee
Next Generation Handsets Lab.
LG Electronics Inc.
yalphy@lge.com

Masaru Kitsuregawa
Institute of Industrial Science
The University of Tokyo
kitsure@tkl.iis.u-tokyo.ac.jp

Chong-Sun Hwang
Dept. of Computer Science and Engineering
Korea University
hwang@disys.korea.ac.kr

Abstract

Wireless data broadcast allows a large number of users to retrieve data simultaneously in mobile databases, resulting in an efficient way of using the scarce wireless bandwidth. The efficiency of data access methods, however, is limited by an inherent property that data can only be accessed strictly sequential by users. This paper addresses the issue of ensuring consistency and currency of data items requested in a certain order by wireless read-only transactions. To properly cope with the inherent property of data broadcast, we explore a predeclaration-based query optimization and devise three predeclaration-based transaction processing methods.

1. Contributions of the Paper

In a traditional pull-based (i.e. client-initiated) data delivery, predeclaration technique has often been used to avoid deadlocks in locking protocols. In the push-based data delivery, however, predeclaration in transaction processing has a novel property that each read-only transaction can be processed successfully with a bounded worst-case response time.

For practical protocols exploring predeclaration, three transaction processing methods are devised: \( P \) (Predeclaration), \( PA \) (Predeclaration with Auto-prefetching) and \( PA^2 \) (PA/Asynchronous). With \( P \), the correctness of read-only transactions can be easily guaranteed only by requiring the server to broadcast consistent data values in each broadcast cycle. Alternatively, by further requiring the server to broadcast invalidation information, a client can use either \( PA \) or \( PA^2 \) to reduce the number of accesses to broadcast channel for its transaction. The conducted analytical study confirms the applicability of our methods in that all the three methods exhibit a considerable reduction in response time without sacrificing serializability or currency of reads. The unique contributions of our work are three-fold:

1. To the best of our knowledge, our work is the first approach to query optimization (on a client side) for reducing transaction response time significantly in wireless data broadcast. This is based on the philosophy that a client should take a more active role in maintaining its transactions consistency in an asymmetric communication environment.

2. Our work is able to balance between average response time (i.e. overall system performance) and worst-case response time (i.e. individual performance), which is one important challenge in wireless transaction management [3]. With our methods, in the general case, the performance gap between average and worst-case response time of a transaction is only half of a broadcast cycle.

3. We observed that serializability is not expensive to achieve in the proposed concurrency control techniques. This is in contrast to the argument that any potential protocol for ensuring serializability would be very expensive in broadcast environments, thus leading to poor performance [2].
2. Proposed Methods

The central idea is to employ predeclaration of readset in order to minimize the number of different broadcast cycles from which transactions read data.

We define the predeclared readset of a transaction $T$, denoted by $Pre_{RS}(T)$, to be a set of data items that $T$ reads potentially. For all methods, each client processes $T$ in three phases: (1) Preparation phase: it gets $Pre_{RS}(T)$, (2) Acquisition phase: it acquires all data items belonging to $Pre_{RS}(T)$ from the broadcast(s) or its local cache. During this phase, a client additionally maintains a set $Acquire(T)$ of all data items that it has acquired so far, and (3) Delivery phase: it delivers data items to its transaction according to the order in which the transaction requires data.

2.1 Method $P$

For method $P$, we assume that the following always hold:

Server Requirement for $P$: The server broadcasts only serializable data values in each broadcast cycle.

One obvious way to satisfy this requirement is to make each broadcast cycle represent the state of the database at the beginning of the cycle. However, if all updates of an update transaction are made solely to the items not yet transmitted over the broadcast channel at the moment, the updated values would be installed into the current broadcast, thereby improving data currency without violating consistency of items within a broadcast cycle. With the above requirement, the execution of each read-only transaction is clearly serializable if a client can fetch all data items within a single broadcast cycle. Since, however, a transaction is expected to be started at some point within a broadcast cycle, its acquisition phase may therefore be across more than one broadcast cycle.

To remedy this problem, in $P$, a client starts the acquisition phase synchronously, i.e. at the beginning of the next broadcast cycle. Since all data items for its transaction are already notified, the client will complete the acquisition phase within a single broadcast cycle. More specifically, a client processes its transaction $T_i$ as follows:

1. On receiving $Begin(T_i)$ {
   get $Pre_{RS}(T_i)$ by using preprocessor;
   $Acquire(T_i) = \emptyset$;
   wait for the next broadcast cycle to begin;
}

2. While ($Pre_{RS}(T_i) \neq Acquire(T_i)$) {
   for any $d_j \in Pre_{RS}(T_i) - Acquire(T_i)$ {
     tune in and read $d_j$ when available from the broadcast;
     put $d_j$ into local storage;
     $Acquire(T_i) \leftarrow d_j$;
   }
}

3. Deliver data items to $T_i$ according to the order in which $T_i$ requires, and then commit $T_i$.

The main advantage of $P$ is that it achieves a considerable reduction of transaction response time in an update-intensive environment without sacrificing serializability or currency of reads. In particular, each transaction can be successfully committed within two broadcast cycles even in an extreme case where all data items in a database are updated during a broadcast cycle. This is because the acquisition phase is completed within a broadcast cycle. The disadvantage of $P$ is that local storage is not fully utilized. In the following section, we devise two variants of $P$ which utilize local storage as local caches.

2.2 Methods $PA$ and $PA^2$

Clients can cache data items of interest locally to reduce access latency. Caching reduces the latency of transactions since transactions find data of interest in their local cache and thus need to access the broadcast channel for a smaller number of times. Clients use their available hard disks as local caches and caching technique is employed in the context of transaction processing. Two practical transaction processing methods in the context of local caching technique, referred to as $PA$ and $PA^2$, can be found at the work [1].

References

