TRANSACTIONAL SUPPORT FOR MOBILE DATABASES

INTRODUCTION

With the widespread deployment of wireless communication infrastructure in the past decade, accessing information online while a client is on the move becomes a concrete possibility. Such a computing environment is often referred to as a mobile environment (Imielinski & Badrinath, 1994). A typical group of applications that deserve strong support under the mobile environment would be database access. Database systems that support operations initiated from mobile clients are referred to as mobile databases (Leong & Si, 1997). We have witnessed a tremendous growth in mobile database research in the past ten years. Yet only the most primitive results have been incorporated in real applications. This is due to the additional dimensions of complexity that the mobile environment has introduced, beyond standard client/server computing environment.

Besides traditional issues to accessing a database system under a client/server setting, a mobile environment also suffers from problems such as low communication bandwidth and unreliable communication with occasional disconnection. To combat for the disconnection problem of mobile clients, database server will allow clients to cache data items that they may need in future, either performing caching on demand or prefetching cache in an anticipatory manner (Jing, Helal & Elmagarmid, 1999). In the context of the mobile file system Coda, this latter approach is called hoarding (Mummert, Ebling & Satyanarayanan, 1995).

There are claims that a mobile environment does possess a high broadcast bandwidth. It is this peculiar feature of asymmetric communication that leads to a good number of research works. For instance, it is quite intuitive to utilize the high bandwidth downlink channel to schedule useful data items to be broadcast to a large collection of mobile clients, thereby improving the scalability of database access, since the same bandwidth can serve a large client population (Si & Leong, 1999).

With downscaling of mobile clients from laptop computers to PDAs and even to smart phones, the relatively weak processing power of mainstream mobile clients cannot be solved in a satisfactory manner without considering a certain tradeoff. To reduce the processing stress on the small mobile clients, it is necessary to move the computation from the clients back to the server. Under such environment, the clients often only implement the interface, passing the data back to the server for operation. However, one may not be willing to give up the autonomy of mobile clients by relinquishing the processing control to the server. The most acceptable variant that still favors client control is through the use of mobile agents (Yau, Leong & Si, 2003), which can act as surrogates on behalf of the mobile clients in performing possibly complex operations on the database and to prepare for handling processing results upon client disconnection. The captured results would only be conveyed back to the mobile clients upon reconnection.

BACKGROUND

Access needs from mobile clients to database server in a mobile environment should be backed by efficient mechanisms, which are specially designed for those mobile databases. To further complicate matters, clients in a mobile environment can physically move around. As a result, the database applications should be enriched by taking into consideration the client location as a special form of data in the database. As such, that gives rise to the need of providing client location management to be tracked through the use of a special kind of databases, namely, moving object databases (Wolfson et al., 1999). Location tracking and management performance can be improved by providing approximate moving object locations, which often suffice for most applications (Lee, Leong & Si, 2003). With object location in light, a new class of database queries whose result sets would depend on the client location need to be handled. These are often referred to as location-dependent queries (Madria et al., 2000).

Most common applications accessing mobile databases often access a collection of related data items for information purpose, that is, for inquiry. There are often only few updates to the databases. In such cases, it may be easy to provide a consistent view of data items to mobile clients, since the updates can be batched and installed to the databases in the background. This is also the philosophy behind many mobile file systems (such as Coda) that only provide the session semantics for remote accesses. However, there are certain applications that require a strong level of consistency to access-
Transactional Support for Mobile Databases

...ing a related set of data items, satisfying the referential integrity (accessing no phantom tuple) or other integrity constraints such as the mutual consistency of the item set (e.g., the total number of seats sold plus remaining seats should be constant in a show). This can be guaranteed via the execution of transactions, satisfying the four conditions of Atomicity, Consistency, Isolation, and Durability (Bernstein, Hadzilacos & Goodman, 1987). The widely accepted correctness criterion of executing concurrent transactions is the serializability of transactions. Transactions initiated by mobile clients that are executed on mobile databases are called mobile transactions (Dirckze & Gruenwald, 2000; Mok, Leong & Si, 1999).

Extending standard ACID mobile transactions further, there are occasions that one needs to access data not only from one database, but also from multiple databases residing on different sites, perhaps spanning different administrative domains. It becomes more difficult to ensure consistency, especially when the databases are managed by different organizations. The extension of serializability in such a context is called the global serializability (Breitbart, Garcia-Molina & Silberschatz, 1992), and the transaction spanning across different organizations is known as a global transaction. Tesch and Wäsch (1997) presented an implementation of global transactions on the ODMG-compliant multidatabase systems. Preserialization techniques were proposed to improve performance of global transactions in the mobile environment (Dirckze & Gruenwald, 2000).

Although global transactions ensure strong consistency, the cost of global transaction execution is very high, so they are not widely adopted in practice. Instead, a global transaction is often split into multiple smaller subtransactions for execution on individual databases. Each subtransaction can commit by itself under the Saga concept (Garcia-Molina & Salem, 1987). Such a collection of a logically related global transaction can be derived from the concept of a nested transaction (Moss, 1987). Alternatively, compensating transactions (Chrysanthis & Ramanritham, 1994) can be executed to undo the effect of failed subtransactions. It has been argued that execution of compensating transactions is often more appropriate in a mobile environment to reduce blocking (Tesch & Wäsch, 1997), since mobile transactions are relatively long-lived.

SUPPORTING MOBILE TRANSACTIONS

Transactional semantics to access a database satisfying the serializability of transactions can be enforced by means of concurrency control protocols, such as two-phase locking and timestamp ordering (Bernstein et al., 1987). Most practical database systems employ the strict two phase locking protocol as their concurrency control mechanism. In the context of a mobile database, it is often more common to implement a variant of the optimistic concurrency control protocol (Bernstein et al., 1987) and two-phase locking with lock caching (Franklin, Carey & Livny, 1997) in view of the high communication overhead. The benefits of optimistic concurrency control protocol lie in the reduction of communication overhead in acquiring locks, thereby deferring the detection of conflicts to the end of a transaction. This is appropriate under the assumption that data conflict is rare.

To improve the performance of variants of optimistic concurrency control protocol as applied in the mobile environment, especially when the degree of conflict is not very low, the server may issue a certification report periodically to mobile clients (Barbara, 1997). The certification report contains data items belonging to the read set and write set of active transactions that have successfully been certified for commitment. In other words, the report conveys information about conflicts in data items from transactions that have been validated. Mobile clients that execute transactions conflicting with those in the certification report can choose to abort those transactions that are destined to fail. To further facilitate early detection of conflicts, relevant information can be conveyed to clients when the latter issue requests for data items. This enables early partial validation to be performed at the server (Lee & Lam, 1999) since the read set of the requesting mobile transaction is known. Another way to reduce the number of mobile transactions being aborted is to reduce the time interval between the availability of updated values of data items to the time the corresponding update transactions commit. This is resolved by introducing the pre-write operation that allows early availability of updated values (Madria & Bhargava, 2001). The corresponding pre-read operation allows return of the value updated by a pre-write operation.

A very special property in a mobile environment is the availability of the high bandwidth broadcast media. As a result, database items can be scheduled to be broadcast over the downlink channel, thereby allowing efficient processing of transactions, especially those read-only transactions (Pitoura & Chrysanthis, 1999). However, one must ensure that the set of data items constituting a broadcast cycle are mutually consistent by taking a consistent snapshot of the database. Furthermore, an invalidation report can be broadcast at the beginning of a broadcast cycle to allow a mobile client to abort transactions that have read an old data value, which has subsequently been overwritten (i.e., reading...