Semantic Reliability on the Database State Machine (Extended Abstract)*

António SOUSA José PEREIRA Rui OLIVEIRA Francisco MOURA
{als,jop,rco,fsm}@di.uminho.pt
Departamento de Informática
Universidade do Minho
Braga, Portugal

1 Introduction

Database replication protocols based on group communication primitives have recently been the subject of a considerable body of research [1, 11, 13, 6, 8, 4]. The reason for this stems from the adequacy of the order and atomicity properties of group communication primitives to implement *synchronous replication* (i.e., strong consistent) strategies. Unlike database replication schemes based on traditional transactional mechanisms, group-based replication mechanisms use *atomic broadcast* primitives to broadcast transactions to all replicas of the database. The approach allows the delegation of much of the synchronization complexity to the group communication layer and can accommodate different replication strategies.

We have been working on extending the Database State Machine protocol (DBSM) [8] to large-scale networks. In [12] we extended the protocol to handle partial replication and to use an optimistic atomic broadcast protocol in order to mitigate the higher communication latency of large-scale settings.

Optimistic atomic broadcast exposes preliminary message delivery orders to the application allowing it to tentatively antecipate processing before it is provided with a final and definitive order — an idea that generalizes the broadcast protocol presented in [7]. In the DBSM, the optimistic atomic broadcast is most useful during low workload periods where processor idle time is used to obtain faster transaction execution times. During heavy workloads however it is of little help as the processor has no idle periods.

To improve DBSM responsiveness during high workloads this paper proposes the use of a semantically reliable atomic broadcast protocol [9] in DBSM.

2 Database State Machine

In this section we briefly describe the Database State Machine protocol. For a full description of the protocol refer to [8].

2.1 System Model

We consider a system $S = \{s_1, \ldots, s_n\}$ of database sites. Sites communicate through message passing (i.e., no shared memory). The system is asynchronous in that there are no assumptions about the time it takes for a site to execute a step nor the time it takes for messages to be transmitted.

Sites may only fail by crashing (i.e., we do not consider Byzantine failures), and we do not rely on site recovery for correctness. A site that never fails is said to be correct. We assume that our asynchronous model is augmented with a Failure Detector Oracle [3] so that Atomic Broadcast can be solved.

A database $\Gamma = \{x_1, \ldots, x_n\}$ is a finite set of data items and each database site s_i holds a copy of the database.

*Research supported by FCT, ESCADA project (POSI/33792/CHS/2000).

A transaction is a sequence of read and write operations followed by a commit or abort operation, issued by a client on behalf of the transaction. Every transaction belongs to the set T of all possible transactions. For each transaction $t \in T$, $Items(t)$ is defined as the set of data items read or written by t. $RS(t)$ denotes the set of data items read by t and $WS(t)$ the set of data items written by t.

2.2 Protocol

A transaction t is entirely executed in some database site $s_i \in S$. From the time t starts until it finishes, a transaction passes through three well-defined states. The starting state is the *executing state*, a state where all operations are executed locally at the database site s_i. When the client that initiates the transaction requests its commitment, transaction t passes to the *committing state*. At this point, transaction t, is sent to all database sites. A transaction received by a database site s_j is in the committing state until its fate is known. The transaction then evolves to one of its final states *committed* or *aborted*.

Using the deferred update replication technique, a transaction t is locally synchronized during its execution at the database where it initiated according to some concurrency control mechanism [2] (e.g., two-phase locking). Interaction with other database sites on behalf of t only occurs when the client requests the transaction commitment. At this time, t is propagated, i.e., the updates of transaction t and some control structures are propagated to all database sites using an atomic broadcast protocol. Each database site will then *certify* and, if possible, commit the transaction.

In order for a database site to certify a committing transaction t, it must be able to determine which transactions conflict with t. A transaction t' *conflicts with* t if: (*i*) t and t' have conflicting operations and (*ii*) t' does not *precede* t.

Two operations conflict when they are issued by different transactions, access the same data item and at least one of them is a write operation. The precedence relation between transactions t and t' is denoted $t' \rightarrow t$ (i.e., t' precedes t) and defined as: (1) if t and t' execute at the same database site, t' precedes t if t' enters the committing state before t; or (2) if t and t' execute at different sites, for example s_i and s_j respectively, respectively, t' precedes t if t' commits at s_i before t enters the committing state at s_i.

After receiving a transaction t delivered by the atomic broadcast protocol, the certification process is started. The certification of a transaction involves a deterministic *certification test* which when terminated allows the database site executing the certification to immediately commit or abort the transaction.

The certification test of t at database site s_i involves every data item accessed by t. Database site s_i commits t if all committed transactions at s_i precede t, or if there is no committed transaction t' at s_i that conflicts with t; s_i aborts t otherwise. The certification test of site s_i on transaction t is formally described as a transition from the committing to the committed state as follows:

$$Committing(t, s_i) \rightsquigarrow Committed(t, s_i) \equiv \left[\begin{array}{l} \forall t',\ Committed(t', s_i) : \\ \\ t' \rightarrow t \vee (WS(t') \cap RS(t) = \emptyset) \end{array} \right]$$

3 Semantically Reliable Multicast in the Database State Machine

In DBSM the certification test is assumed to be deterministic and thus, when a database site aborts a transaction then all database sites also abort it. Moreover, when the result of certifying a transaction t is abort then all the information received with t may be discarded. This means that if a s_i site knows in antecipation that some transaction t will abort, then s_i can simply discard t and save itself from processing the certification of t. Being able to notify in antecipation other database sites which transactions will abort can be most valuable for overloaded database sites as they can simply discard those transactions.

To implement this optimization efficiently we use a semantically reliable atomic broadcast protocol to dissiminate commiting transactions. Semantic reliability [9] is a correctness criterium for broadcast protocols based on the concept of *message obsolescence*. A message becomes obsolete and can be readily discarded by the protocol whenever its purpose is superseded by a subsequent message. This information is expressed in the protocol obsolescence relation encoded by the application.

In the DBSM the obsolescence relation is given by the knowledge a database site has about the transactions that aborted. That is, whenever a site s_i broadcasts a message m to propagate a commiting transaction, s_i encodes in m that m obsoletes all previous messages propagating commiting transactions that have aborted at s_i. With this information encoded in m, the broadcast protocol might be able to prevent those messages carrying transactions that will not pass the certification test from overloading some database sites.

The obsolescence relation is a strict partial order and the fact that a message m is obsoleted by a message m' is expressed as $m \sqsubset m'$. A semantically reliable multicast protocol [10] is an algorithm fulfilling the following properties:

Validity If a correct database site multicasts a message m and there is a time after which no database site multicast some m'' such that $m \sqsubset m''$, then it delivers some m' such that $m \sqsubseteq m'$.

Agreement If a correct database site delivers a message m and there is a time after which no database site multicasts m'' such that $m \sqsubset m''$, then all correct database sites deliver some m' such that $m \sqsubseteq m'$

Integrity For every message m, every database site delivers m at most once and only if m was previously multicast by some database site.

FIFO Order If a database site multicasts a message m before it multicasts a message m', no database site delivers m after delivering m'.

FIFO Completeness If a database site multicasts a message m before it multicasts a message m' and there is a time after which no database site multicasts m''' such that $m \sqsubset m'''$, no correct database site delivers m' without delivering some m'' such that $m \sqsubset m''$.

The DBSM correctness criterium relies on an atomic multicast protocol to ensure that all database sites process the same set of transactions in the same order. An atomic multicast protocol is a reliable multicast protocol satisfying a total order requirement [5]. Informally, an atomic multicast protocol must ensure that if a database site delivers two messages m and m' in that order then no other database site delivers them differently. This is described by the following total order property:

Total Order if correct database sites s_i and s_j both deliver messages m and m' , then s_i delivers m before m' if and only if s_j delivers m before m'.

4 Conclusions and Future Work

In [12] we extended the DBSM to support partial replication and the use of an optimistic atomic broadcast to improve performance mainly during low/moderate load periods.

This paper describes ongoing research on improving DBSM performance during heavy load periods, allowing better responsiveness during those periods. This seems a promising research direction as DBSM presents the required pattern for benefiting from semantically reliable protocols [9].

Currently our obsolescence relation is based on aborted transactions. We believe that with results derived from the experiences we are conducting with this approach along with further analysis of the transactions we could enrich our obsolescence relation making it more effective. By reducing the processing requirements at each database site, we might improve responsiveness to the client even in normal workloads.

In this paper we have focused in high workloads and dropped the use of an optimistic broadcast protocol. We are studying how to combine semantic and optimistic protocols in order to improve DBSM performance in all possible workloads.

References

[1] Y. Amir, D. Dolev, P. Melliar-Smith, and L. Moser. Robust and efficient replication using group communication. Technical Report CS94-20, The Hebrew University of Jerusalem, November 1994.

[2] P. A. Bernstein, V. Hadzilacos, and N. Goodman. *Concurrency control and recovery in Database Systems*. Addison-Wesley, 1987.

[3] T. D. Chandra and S. Toueg. Unreliable failure detectors for reliable distributed systems. *Journal of the ACM*, 43(2), March 1996.

[4] U. Fritzke and P. Ingels. Systéme transactionnel pour donnés partiellement dupliqués, fondé sur la communication de groupes. Technical Report 1322, INRISA, Rennes, France, April 2000.

[5] V. Hadzilacos and S. Toueg. Fault-tolerant broadcasts and related problems. Technical Report 94-1425, Department of Computer Science, Cornell University, May 1994.

[6] B. Kemme and G. Alonso. A suite of database replication protocols based on communication primitives. In *Proceedings of the 18th International Conference on Distributed Computing Systems*, Amsterdam, The Netherlands, May 1998.

[7] B. Kemme, F. Pedone, G. Alonso, and A. Schiper. Processing transactions over optimistic atomic broadcast protocols. In *19th IEEE International Conference on Distributed Computing Systems (ICDCS '99)*, pages 424–431. IEEE, May 1999.

[8] F. Pedone, R. Guerraoui, and A. Schiper. The database state machine approach. Technical Report SSC/1999/008, École Polytechnique Fédérale de Lausanne, Switzerland, March 1999.

[9] J. Pereira, L. Rodrigues, and R. Oliveira. Semantically reliable multicast protocols. In *Proceedings of the 19th IEEE Symposium on Reliable Distributed Systems (SRDS'19)*, pages 60–69, Nurnberg, Germany, October 2000.

[10] J. Pereira, L. Rodrigues, and R. Oliveira. Semantically reliable multicast: Definition implementation and performance evaluation. *Special Issue of IEEE Transactions on Computers on Reliable Distributed Systems*, 2003. to appear.

[11] A. Schiper and M. Raynal. From group communication to transactions in distributed systems. *Communications of the ACM*, 39:84–87, April 1996.

[12] A. Sousa, F. Pedone, R. Oliveira, and F. Moura. Partial replication in the database state machine. In *IEEE International Symposium on Network Computing and Applications*, pages 298–309, Cambridge, MA, October 2001. IEEE Computer Science.

[13] I. Stanoi, A. Agrawal, and E. Abbadi. Using broadcast primitives in replicated databases (abstract). In *Proceeding of the Sixteen Annual ACM Symposium on Principles of Distributed Computing*, page 283, Santa Barbara, USA, August 1997.