EDRC:
An Early Data Lending-Based Real-Time Commit Protocol

Sarvesh Pandey
https://orcid.org/0000-0002-3014-9792
MMM University of Technology, India

Udai Shanker
https://orcid.org/0000-0002-4083-7046
MMM University of Technology, India

INTRODUCTION

The advanced Distributed Database System (DDBS), with the integration of 'real-time' attribute to each transaction, makes sure that results are produced within a stipulated deadline while maintaining the consistency. Such systems are broadly studied under the umbrella of the research area named Distributed Real Time Database System (DRTDBS). In simple, they are time constrained DDBSs (Pandey & Shanker, 2018a). In the DRTDBS, the distributed real-time transaction (DRTT) is invoked to perform changes at multiple sites atomically. Here, the correctness of the result depends on two things: logical computation performed, and the time when the result is produced. Even if the result produced is functionally right, it may lead to tragic repercussions, be unusable, or has less value if it is not produced in time (Shanker, Misra, & Sarje, 2001).

Based on the consequences of their deadline misses, DRTTs may be categorized as soft, firm and hard. The soft DRTT is not killed/aborted in case of its deadline miss because the result has some value (obviously degrading) even after the deadline miss. The firm DRTT is killed in case it misses its deadline because its outcome has no value after the deadline miss. In addition, allowing the execution of firm DRTT after its deadline expiry may also lead to deadline expiration of other concurrently executing firm DRTTs requiring access to resources already locked by it. The hard DRTT must be finished before its deadline; otherwise, it may lead to a potentially catastrophic consequence.

Scheduling concurrent execution of DRTTs is extremely complex as the goal of database management algorithms is not only to maintain database consistency but also to ensure that timing constraints are met. Issues such as data conflicts, site failures, communication delays, interaction through the underlying operating system and I/O subsystem are major obstacles in meeting the deadlines of concurrently executing DRTTs (Shanker U., 2008). Among all these, the data conflict problem is a key factor that adversely affects system performance; it can be categorized as Execute-Execute Conflict and Execute-Commit Conflict. The Execute-Execute conflict may occur amongst executing transactions. A considerable research investigation has been carried out to resolve this conflict. For detailed knowledge about the Execute-Execute conflict and strategies proposed for its resolution, the readers may go through (Shanker, Misra, & Sarje, 2008). The Execute-Commit conflict may occur between executing and committing transactions. It occurs only if one of the transactions involved in data conflict is in a PREPARED state.

DOI: 10.4018/978-1-7998-3479-3.ch055

The real-time concurrency control protocol resolves execute-execute conflict while the real-time commit protocol, an optimistic implementation of 2-Phase Commit (2PC) protocol with relaxed ACID property, resolves execute-commit conflict.

To overcome execute-commit conflicts, the lender-borrower approach is commonly used in next generation commit protocols — which are not only of optimistic nature, but also intended to satisfy real-time constraints. Such real-time commit protocols permit prepared lender cohorts to lend their modified and uncommitted data items to executing borrower cohorts in case of occurrence of Execute-Commit conflict between them. This reduces data inaccessibility. Thus, they improve DRTDBS performance. They, however, usually do not permit the borrower cohort to send WORKDONE/PREPARED message, and thus, increase the borrowing transactions' commit time. Hence, designing efficient real-time commit protocol is a vital issue as it largely affects the DRTDBS performance.

BACKGROUND

The research on developing an efficient real-time variant of the classical 2PC is still an open question in the study of the DRTDBS (Pandey & Shanker, 2016). Ramesh Gupta et al. first proposed a real-time variant of classical 2PC protocol named OPT (Gupta, Haritsa, Ramamritham, & Seshadri, 1996); this protocol is specifically designed to fulfill the consistency and deadline requirements of the DRTDBS by reducing the negative impact of Execute-Commit conflict and inherent priority inversion problem. The OPT protocol permits a high priority cohort to borrow/access the uncommitted data item(s) held by prepared low priority cohort. Such lending by a prepared cohort creates a dependency between conflicting DRTTs. The fate of borrower DRTT depends on the final outcomes of DRTTs from whom it had borrowed the data items — if final outcomes of all the lender DRTTs are 'commit', then only the borrower DRTT can start its commit processing. Thus, the OPT makes the borrower to be blocked till lenders complete their execution; this optimism is advantageous only when lenders successfully complete their executions. While the policy of using uncommitted data items may result in the chain of dependencies responsible for the cascading aborts, the OPT protocol restricts the length of chain to only one by not allowing the borrowers to simultaneously become lenders. Therefore, it doesn't suffer from a cascading abort problem. It provides a considerable performance improvement over 2PC.

Moreover, two variants of the OPT is also suggested: Healthy-OPT and Shadow-OPT (Gupta, Haritsa, & Ramamritham, 1997). In the Healthy-OPT, every executing transaction is assigned a health factor (HF), and a transaction can become lender only when its HF value is greater than or equal to some threshold value. Obviously, the performance of the system heavily varies with change in the chosen threshold value. In the Shadow-OPT, the cohort originates a shadow (replica of itself) at the instance it borrows some uncommitted data item. Moreover, the original version of the borrower cohort continues its execution as usual even after borrowing uncommitted data item while the shadow cohort is blocked right after being forked off as a separate cohort. In case lenders successfully commit, the shadow cohort is discarded since optimistic borrowing done its job well. Otherwise, if any of the lenders (who have a dependency with this cohort) aborts, the shadow cohort is activated, and the original cohort is aborted. Thus, the Shadow-OPT saves the borrower cohort from restarting its execution in case of an unsuccessful lending-borrowing event and provides a feature so that the borrower cohort can resume its processing from the point the borrowing is made. The detailed simulation results showed that the Healthy-OPT performs reasonably fairer than the Shadow-OPT.

Related Content