On High Performance Distributed Transactional Data Structures

Technical Report

Aditya Dhoke
Virginia Tech
adityad@vt.edu

Roberto Palmieri
Virginia Tech
robertop@vt.edu

Binoy Ravindran
Virginia Tech
binoy@vt.edu

Abstract

We present three protocols for developing high performance distributed transactional data structures. Our first protocol, QR-ON, incorporates the open nesting transactional model into QR, a quorum-based protocol for managing concurrency on distributed transactions. The open nesting model allows nested transactions to commit independently of their parent transaction. This releases objects in the transaction read-set and write-set early, minimizing aborts due to false conflicts and improving concurrency. We then introduce Optimistic Open Nesting, QR-OON, in which open-nested transactions commit asynchronously so that subsequent transactions can proceed without waiting for the commit of previous transactions. Finally, we propose an early release methodology, QR-ER, in which objects that do not affect the final state of the shared data are dropped from the transaction read-set, avoiding false conflicts and improving performance.

Categories and Subject Descriptors D.1.3 [*Programming Techniques*]: Concurrent Programming; H.2.4 [*Systems*]: Transaction processing

Keywords Data Structures, Distribution, Transactions, Performance, Nesting

1. Introduction

The growing promise of Software Transactional Memory has led to the extension of well-known concurrent data structures with transactional support, in both multiprocessor and distributed contexts. While the benefits of transactional data structures under multiprocessor settings are well-known, they can be equally useful in distributed systems.

A transactional conflict occurs among transactions when at least one of them is writing to the same object. One of the transactions must be aborted to resolve the conflict. In addition, systems based on transactional data structures also suffer from another type of conflict, called *false conflict* [2], which occurs among transactions performing seemingly independent operations. Consider an example of a *set* implemented using a sorted list. Here, insertion of an element in the *set* can be viewed as a high level operation, while insertion of an object in the sorted list can be viewed as a low level operation. To insert an object O_1 between objects O_2 (lower) and O_3 (greater), a transaction T must traverse from the head of the list, and read all objects prior to O_1. Ideally, any invalidation due to concurrent writes on objects prior to O_1 would not compromise T's correctness and should not create any conflicts. However, high level operations, even though semantically independent, traverse the same set of objects during their execution, causing false conflicts. False conflicts can degrade performance, especially in distributed data structures, where repeated aborts can significantly increase transaction execution time, as transaction execution in this setting includes expensive network communication.

Past research has proposed solutions to the problem of false conflicts [1, 2]. However, all these works focus on centralized systems (i.e., shared memory multicores/ multiprocessors), but not distributed. Transaction execution characteristics in distributed settings are significantly different from multiprocessor settings. Unlike centralized systems, in distributed systems, the cost of communication dominates overall transaction execution time.

Motivated by these observations, with the goal of reducing false conflicts, we propose three protocols: QR-ON (Section 2), QR-OON (Section 3), and QR-ER (Section 4). The protocols use QR [3], an efficient quorum-based distributed concurrency control protocol guaranteeing fault-tolerance, as the baseline transactional protocol. In QR, objects are fully replicated across all nodes, and transaction execution is divided into two independent phases with orthogonal responsibilities: 1) read/write phase, in which a transaction obtains the latest copies of objects, and 2) commit phase, in which objects are validated and updates are committed. QR uses the quorum intersection property for obtaining the latest copy of objects and for detecting conflicts.

We implemented the protocols, and report preliminary experimental results obtained on three (distributed) data structures (linked-list, hashmap, and BST). The experiments were conducted on a 13 node cluster, where each is an 8-core AMD machine, interconnected using a 1Gbps network.

2. QR-ON

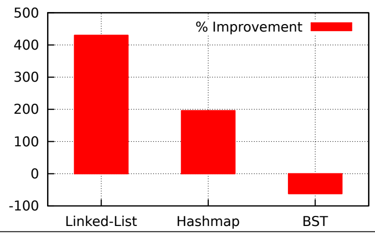

Figure 1. QR-ON vs QR.

QR-ON incorporates the open nesting model [2] into QR. In open nesting, only the objects accessed within the (open) nested transactions are validated and (globally) released after successful commit. This early release increases the potential for improving concurrency: two parent transactions that have read or written the same set of objects in their inner transactions will not detect any conflict during their commit. Since each nested transaction directly commits to the shared (or possibly distributed) state, other transactions can also immediately access the just committed data. Figure 1 shows QR-ON's throughput improvement over QR of 4.2× for linked-list and 1.95× for hashmap. Conversely, BST's transactions access a small subset of shared objects, and therefore does not suffer from false conflicts.

3. QR-OON

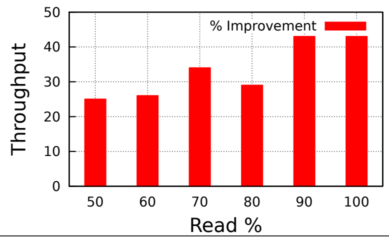

Figure 2. QR-OON's speed-up over QR-ON for linked-list.

Exploiting open nesting for speeding up the validation of parent transactions, as done in QR-ON, has a drawback. For a distributed transactional data structure whose commit phase is inherently slower than rest of the transactional execution due to remote validation and locking, repeating the commit as many times as there are (open) nested transactions can negate open nesting's potential for increased concurrency (due to early object release). Motivated by this observation, QR-OON makes QR-ON's commit phase non-blocking: when an open-nested transaction commits, it starts the classical open-nested commit phase. Besides, the transaction is also locally committed, allowing subsequent transactions to start their execution without waiting for its commit. This causes an overlap between the commit of an open-nested transaction and the read/write phase of subsequent transactions. The approach pays off when subsequent open-nested transactions are likely to access the data written by the previous, still committing, transaction. Figure 2 shows QR-OON's throughput improvement over QR-ON (up to 43%) for linked-list, with 5 nested transactions and 500 objects.

4. QR-ER

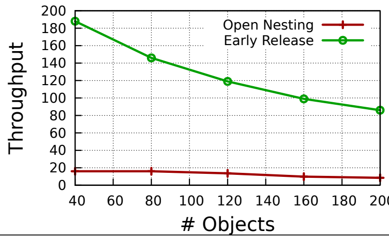

Figure 3. QR-ER vs QR-ON with linked-list.

Both QR-ON and QR-OON exploit open-nested transactions for reducing the size of the read-set when the parent transaction commits. Even though in QR-OON, the expensive cost of a commit is alleviated by the asynchronous implementation, distributed protocols extensively use the network during transaction execution. In such cases, the overlapping time may be limited, because, few nested transactions may optimistically execute concurrently with the commit phase. Motivated by this observation, our third protocol, QR-ER, does not rely on open nesting, but instead, use an early release mechanism to resolve false conflicts. QR-ER drops those objects from the transaction read-set that do not need to be validated because, even in case of invalidation, they do not compromise correctness of the execution. Figure 3 compares QR-ER's throughput with QR-ON's for linked-list, by varying the number of objects in the data structure, for 3 nested transactions (for QR-ON) and 50% write transactions. Results reveal an improvement over QR-ON up to 7×. We selected linked-list and excluded QR-OON in this comparison because we already reported a comparison between QR-ON and QR-OON in Figure 2.

Acknowledgements

This work is supported in part by US National Science Foundation under grants CNS 0915895, CNS 1116190, CNS 1130180, and CNS 1217385.

References

[1] M. Herlihy and E. Koskinen. Transactional boosting: a methodology for highly-concurrent transactional objects. In *PPoPP '08*.

[2] J. E. B. Moss. Open nested transactions: Semantics and support. In *WMPI '06*.

[3] B. Zhang and B. Ravindran. A quorum-based replication framework for distributed software transactional memory. In *OPODIS '11*.