On High Performance Distributed Transactional Data Structures

Aditya Dhoke, Roberto Palmieri, Binoy Ravindran
Department of Electrical and Computer Engineering, Virginia Tech, Virginia, USA
{adityad, robertop, binoy}@vt.edu

Transactional Data Structures suffer from false conflicts

The impact of those conflicts is significant in distributed data structure

❖ 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.

MAIN CHALLENGE

- Reduce false conflicts.
- Improve performance of distributed data structures.

False Conflicts

❖ They are generated by conflicting transactions performing seemingly independent operations.
❖ They do not compromise transactions' correctness.
❖ They do not break data consistency.

False Conflicts by example

❖ 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 O1 between objects O2 (lower) and O3 (greater), a transaction T must traverse from the head of the list, and read all objects prior to O1. Ideally, any invalidation due to concurrent writes on objects prior to O1 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 in- crease transaction execution time, as transaction execution in this setting includes expensive network communication.

Contributions:

QR-ON: Open Nesting

QR-ON incorporates the open nesting model. 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.

QR-OON: Optimistic Open Nesting

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.

QR-ER: Early Release

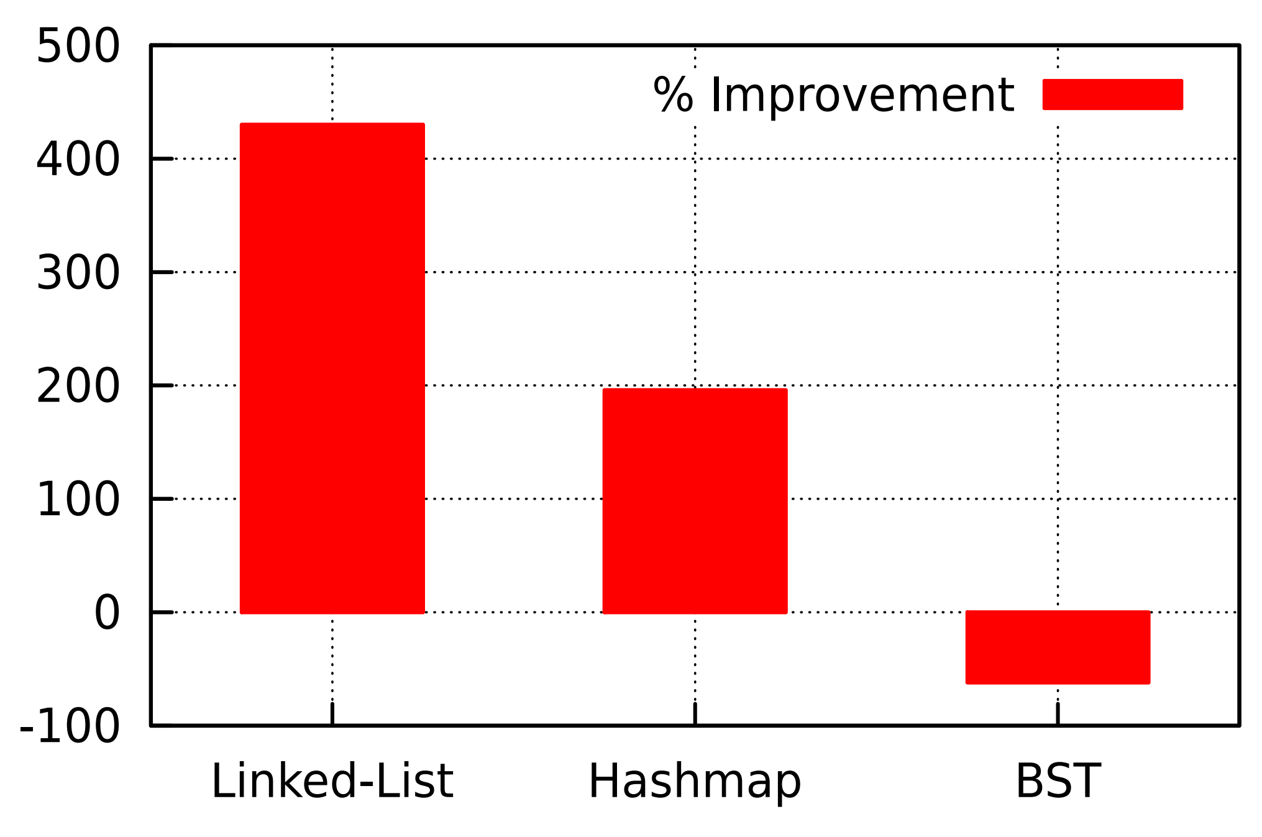
QR-ER, does not rely on open nesting, but instead, use an early release mechanism to resolve false conflicts. QR- ER drops those objects from the transaction read-set that do not need to be validated because, even in case of invalidation, they do not compromise correctness of the execution.

Preliminary Evaluation

Configuration

❖ The experiments were conducted on a 13 node cluster, where each is an 8-core AMD machine, interconnected using a 1Gbps network.

QR-ON vs QR

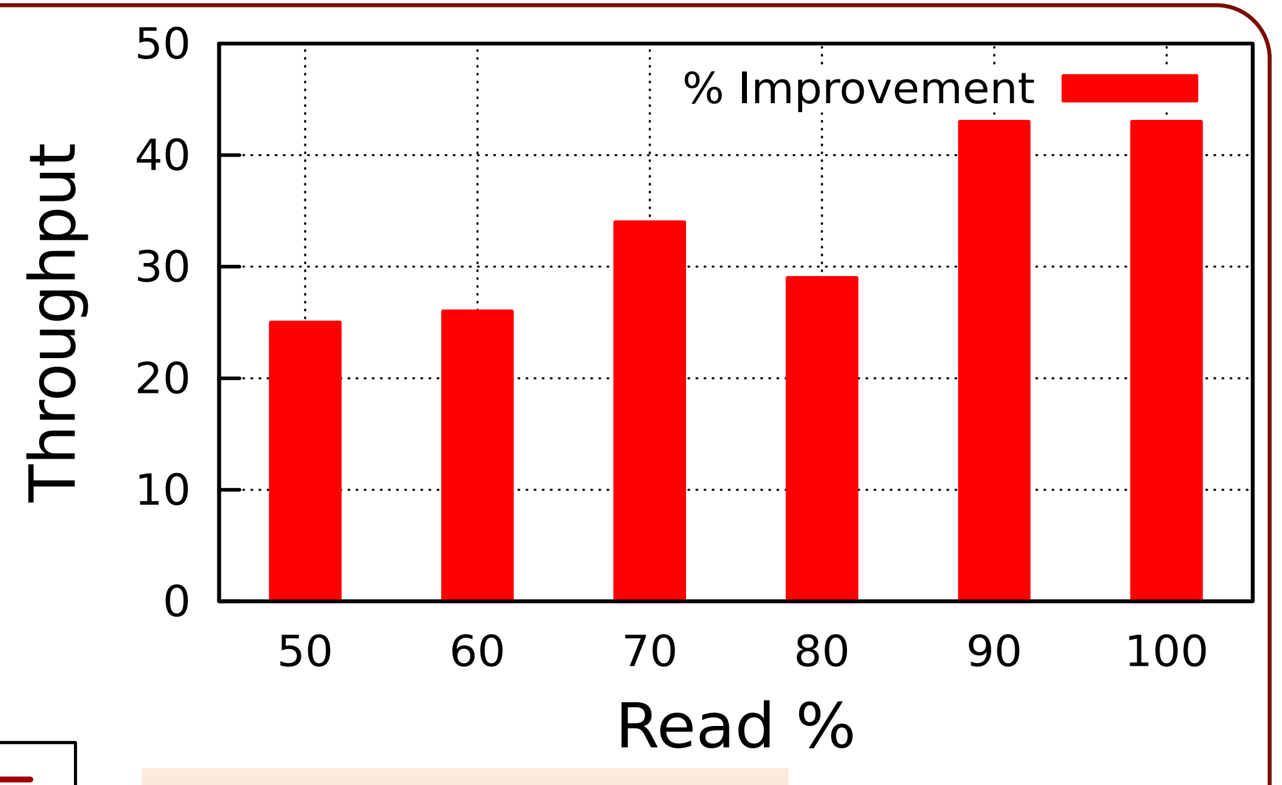


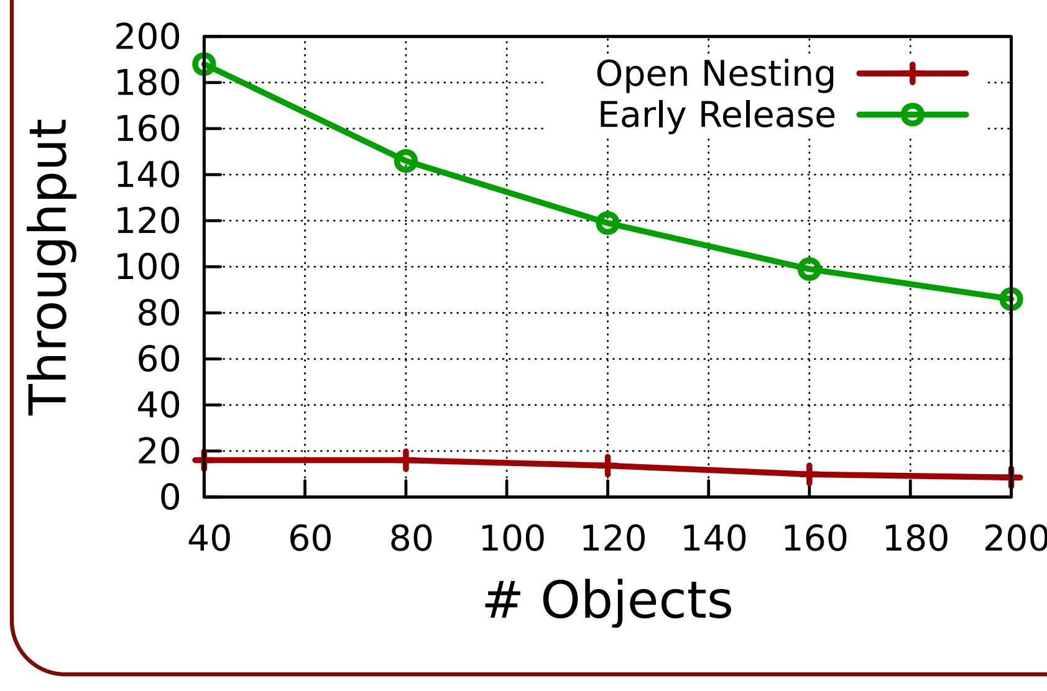
❖ The Figure shows QR-ON's throughput improvement over QR of 4.2× for linked-list and 1.95× for hash-map.
❖ Conversely, BST's transactions access a small subset of shared objects, and therefore does not suffer from false conflicts.

QR-OON vs QR-ON

Hash-Set:

❖ Figure shows QR- OON's throughput improvement over QR-ON (up to 43%) for linked-list, with 5 nested transactions and 500 objects.





QR-ER vs QR-ON

❖ Figure 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.

Finally...

Open nesting is the typical solution for solving false conflicts, but we determined that it has significant commit overhead in fault-tolerant DTM. We showed that optimistic open nesting can outperform open nesting in low contention scenarios. Additionally, we showed that early release can provide substantial performance improvement -- up to an order of magnitude -- over its open nesting counterparts.

ACKNOWLEDGMENTS

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