SMASH: Speculative State Machine Replication in Transactional Systems

Sachin Hirve, Roberto Palmieri, Binoy Ravindran
Department of Electrical and Computer Engineering, Virginia Tech, Virginia, USA
{hsachin, robertop, binoy}@vt.edu

Transactional Systems based on the Active Replication paradigm, an Optimistic Atomic Broadcast layer and a speculative concurrency control.

❖ Fault-tolerance is a strongly desirable property of transactional systems. Object replication provides fault-tolerance guarantees, but suffers from some trade-offs. On the one hand, partially replicating shared objects across remote nodes helps reducing the storage requirement at each node, but it also burdens transactions with expensive network communication steps, resulting in long execution times and possibly low performance. On the other hand, full replication allows transactions to execute fully locally, which yields low execution times, but requires an ordering protocol that ensures a total order among the transactional requests issued by clients.

Main Challenge

- **Building a high-performance transactional system based on Active Replication.**

Atomic Broadcast

❖ Atomic Broadcast (AB) is a total order layer based on consensus. All nodes receive the same set of messages in a unique order, even in presence of failures.
❖ *ABcast(m):* used by clients to broadcast a message m to all the nodes.
❖ *Adeliver(m):* event notified to each replica for delivering a message m.

Optimistic Atomic Broadcast

❖ *Odeliver(m)*, which is used for early delivering a previously broadcast message *m* before the *Adeliver(m)* is issued.

System Model

We consider a classical distributed system model consisting of a set of processes (replicas) that communicate via message passing. Processes may fail according to the fail-stop (crash) model. In order to reach consensus, we assume that the majority of nodes are always correct. We assume the full replication model where each replica maintains the whole shared data set. Clients wrap transactions in transaction requests. They are broadcast using the OAB service to all the replicas.

SMASH

Optimized OAB service

❖ The key idea of our optimized OAB service is exploiting the observation that, during a crash-free execution, while the nodes are establishing the consensus of messages, the probability of a mismatch between the optimistic and relative final order of the message is minimal.
❖ Implemented on top of S-Paxos.
❖ When the leader sends the proposed order for a batch, replicas use it for triggering the optimistic delivery. In order to minimize inversions, replicas trigger an optimistic delivery only when:
1) they receive a propose message;
2) all request batches of the propose message have been received; and
3) all previous instances have already been optimistically delivered.

Speculative Concurrency Control (SCC)

❖ SCC exploits multi-versioned memory for activating read- only transactions in parallel to write transactions that are, in contrast, executed in a single thread.
❖ Single-thread processing ensures that when a transaction completes its execution, all the previous transactions are executed in a known order.
❖ Additionally, no atomic operations are needed for managing locks or critical sections. As a result, write transactions are processed faster and read-only transactions do not suffer from otherwise overloaded hardware bus (due to CAS operations and cache invalidations caused by spinning on locks).
❖ When the final order of a transaction is established, if it is completely executed by then, it is validated for detecting the equivalence between its actual serialization order and the final order. If the processing order is equivalent to the OAB order, then the transaction is committed; otherwise it is aborted and restarted.

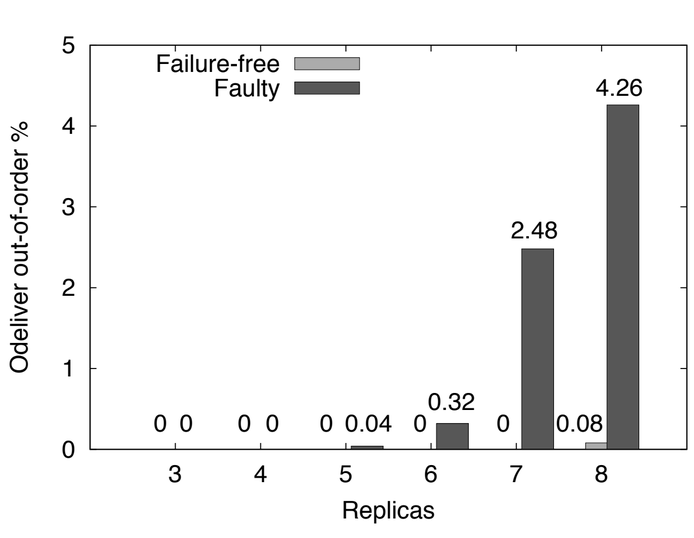
Preliminary Evaluation

Configuration

❖ We implemented the two components of SMASH in Java.
❖ Our test-bed consists of 8 nodes, each of which is a 64-core AMD Opteron machine, interconnected using a 1Gbps network.

OAB service

❖ We conducted experiments measuring the percentage of reordering between optimistic/final deliveries.
❖ We tested failure-free and faulty executions where we explicitly crash the leader.

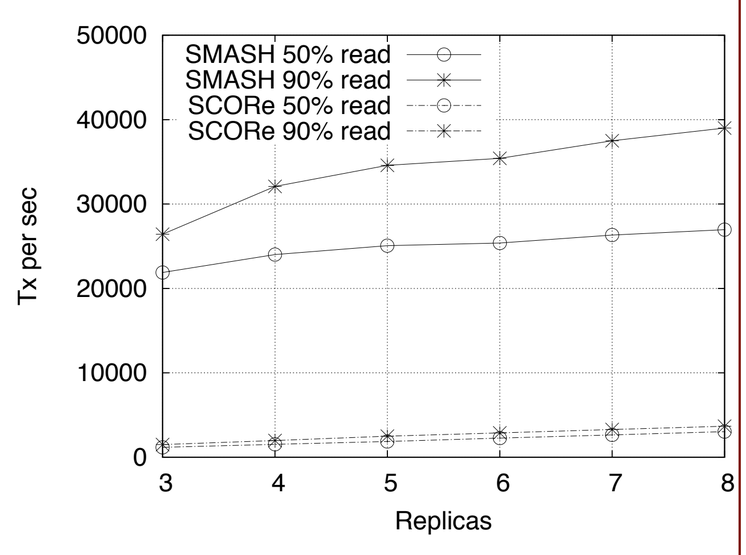


SMASH

❖ We contrast SMASH with SCORe, a state-of-the-art partial replication approach.

❖ We used TPC-C benchmark, configured to generate different percentages of read-only transactions (50% and 90%).

❖ SMASH outperforms SCORe with 8 replicas by up to 10x.



❖ SMASH's effectiveness is particularly evident here, as SCORe pays the cost for looking-up remotely accessed objects, unlike SMASH, which executes locally.

Finally…

At its core, our work shows that optimism pays off: speculative transaction execution, started as soon as transactions are optimistically delivered, allows hiding the total ordering latency, and yields performance gain. Single-communication step is mandatory for fine-grain transactions.

ACKNOWLEDGMENTS

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