SMASH: Speculative State Machine Replication in Transactional Systems

Sachin Hirve
Virginia Tech
hsachin@vt.edu

Roberto Palmieri
Virginia Tech
robertop@vt.edu

Binoy Ravindran
Virginia Tech
binoy@vt.edu

Abstract

We present SMASH, a high performance active replication approach for fault-tolerant distributed transactional systems. The active replication paradigm allows transactions to execute locally, costing them only a single network communication step during transaction execution. Shared objects are replicated across all sites, avoiding remote object accesses. Replica consistency is ensured by an optimistic atomic broadcast layer that total-orders transactional requests, while guaranteeing minimal message reordering. Additionally, a local multi-version concurrency control protocol efficiently enforces a commit order equivalent to transactions' delivery order, while read transactions are executed in parallel.

Categories and Subject Descriptors C.2.4 [*Computer Communication Networks*]: Distributed databases; H.2.4 [*Database Management*]: Transaction processing

Keywords Transactions, Memory, Fault-Tolerance

1. Introduction

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 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 (Atomic Broadcast [3] or AB) that ensures a total order among the transactional requests issued by clients. The latter approach, also called *State Machine Replication*, is usually not adopted in transaction processing due to the high overhead of globally ordering transactions before their execution. AB offers two primitives for broadcasting a message (used by clients) and for delivering an ordered transaction (used by replicas). An improved version of AB, called Optimistic Atomic Broadcast (OAB) [2], has been recently proposed. OAB enriches AB with an additional notification, called *optimistic delivery*, which is sent by the ordering layer to the replicas after the transaction's broadcast, but prior to the notification of its final order. OAB's order is generally not reliable. However, the optimistic delivery can be exploited by the replicas' concurrency control mechanism for activating the transactions by guessing the final order (speculatively), anticipating possibly useful work, in case the ordering layer does not reorder any transaction during the coordination phase.

In this paper we present *SMASH*, a state-machine replication approach for building high performance fault-tolerant transactional systems (Section 2). We leverage two main building blocks. The first is an optimized ordering layer that implements OAB, which ensures high throughput and minimal reordering. The second is a speculative concurrency control mechanism that uses a single thread for processing write transactions, as well as a pool of threads for executing read-only transactions in parallel to write transactions, exploiting object multi-versioning. Both these blocks exploit multiple nodes for replicating the transactional state. Due to the multiple committed versions that are stored for each shared object, read-only transactions run fully locally at each node, yielding high performance. Write transactions are executed fast, without blocking operations, assuming that the optimistic order is likely confirmed by the OAB service.

2. SMASH

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 [3], 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.

Middleware 2013 Posters and Demos Track, December 9-13, Beijing, China.
Copyright © 2013 ACM 978-1-4503-2549-3…$15.00.

Clients wrap transactions in transaction requests. They are broadcast using the OAB service to all the replicas. After that, clients simply wait until the feedback of the submitted transaction is issued by the system. A client's request is first sent to the ordering layer (Section 2.1), and subsequently to the replica's concurrency control mechanism (Section 2.2). At each node, transactions are processed and committed according to the final order notified by the OAB service.

2.1 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. A recent effort has proposed a high throughput AB layer [1]. Here, a replica creates a batch of client requests and distributes it to other replicas. The receiver replicas store the batch of requests and send an *ack* to all other replicas. When the replicas observe a majority of *ack*s for a batch, it is considered as stable. The leader (i.e., the node currently constructing the ordering) then *proposes* an order (containing only the batch IDs) for the non-proposed stable batches, for which, the other replicas reply with their agreement i.e., *accept* messages. When a majority of agreement is reached, each replica considers it as *decided*.

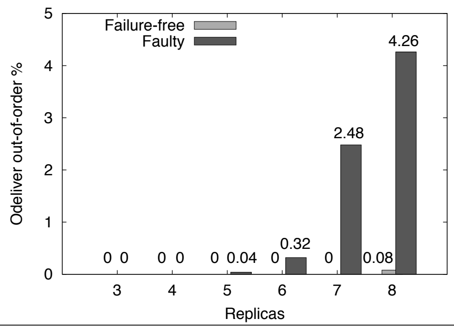

Figure 1. % of out-of-order optimistic w.r.t. final delivery.

We designed an architecture that minimizes the likelihood of message reordering by extending [1]. 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. Figure 1 shows results for failure-free and faulty runs.

2.2 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.

3. Preliminary Results

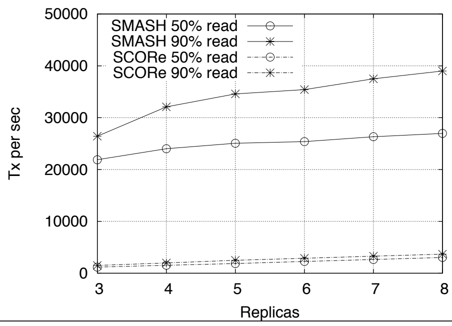

Figure 2. Performance with read workload.

We implemented the two components of SMASH in Java. Figure 2 reports initial results using the TPC-C benchmark. Our test-bed consists of 8 nodes, each of which is a 64-core AMD Opteron machine, interconnected using a 1Gbps switched network. We contrast SMASH with SCORe [4], a state-of-the-art partial replication approach. We configured TPC-C to generate different percentages of read-only transactions. Results reveal SMASH's effectiveness in overlapping transaction execution with OAB's actions. SMASH outperforms SCORe with 8 replicas by up to 10×. SMASH's effectiveness is particularly evident here, as SCORe pays the cost for looking-up remotely accessed objects, unlike SMASH, which executes transactions locally.

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. Biely, Z. Milosevic, N. Santos, and A. Schiper. S-Paxos: Offloading the Leader for High Throughput State Machine Replication. In *SRDS '12*.

[2] B. Kemme, F. Pedone, G. Alonso, A. Schiper, and M. Wiesmann. Using optimistic atomic broadcast in transaction processing systems. *IEEE TKDE*, 15(4), 2003.

[3] L. Lamport. The part-time parliament. *ACM Trans. Comput. Syst.*, pages 133–169, 1998.

[4] S. Peluso, P. Romano, and F. Quaglia. SCORe: A scalable one-copy serializable partial replication protocol. In *Middlewar 12*.