High Performance State-Machine Replication

Parisa Jalili Marandi
University of Lugano
Switzerland

1 Introduction

State-machine replication is a well-known approach to making systems fault tolerant. The idea is to replicate the state of a service across several failure-independent servers with each server implementing the state-machine. In state-machine replication by starting from the same initial state and executing the same set of commands in the same order all the servers evolve to the same states [5, 10]. Following this direction the focus of this research has been on two of the components of state-machine replication, the ordering of the commands and their execution. In Section 2 a new atomic broadcast protocol to address the ordering is discussed. In Section 3 the execution of the commands are considered and two optimizations for improving its performance are briefly investigated. Finally some ideas for future work are mentioned in Section 4.

2 Command Ordering

Ensuring consistency among the replicas of a replicated system is an important issue that can be achieved through the ordering provided by atomic broadcast protocols [2, 9]. Therefore and based on the focus of this research as explained in Section 1 the primary goal has been to propose a high performance atomic broadcast protocol. The result is Ring Paxos, briefly explained in the following section.

2.1 Ring Paxos

Ring Paxos is a high performance atomic broadcast protocol derived from Paxos [6] capable of exploiting 90% bandwidth of a 1G network with an approximate latency of 4 mili seconds. Similar to Paxos, nodes play the acceptor, proposer, and learner roles. Proposers propose values, acceptors decide the ordering of values, and learners are the nodes interested in the outcome of the protocol. In Ring Paxos acceptors are positioned in a unidirectional ring decided by the coordinator who itself is an acceptor as well. Proposers submit their values to the coordinator only. For each value to be decided there is a single message propagated along the ring and each acceptor adds its relevant data to this message. Acceptors receive messages only from their immediate predecessor in the ring and send messages only to their immediate successor. Thus they do not directly communicate with the coordinator. The Coordinator is the only one who uses multicast to send proposed values or decisions to the nodes. Consensus in Ring Paxos is reached over value IDs rather than the values, thus the communication along the ring is very light. On the other hand only a majority of the nodes are placed in the ring. These two properties, besides efficient communication patterns, lead to high throughput and low latency in Ring Paxos. By adding more acceptors to the ring, throughput remains constant with a negligible increase in the latency. Ring Paxos is explained in detail in [8] and the code is available at [1].

3 Command Execution

The need for totally ordering commands in state-machine replication to ensure consistency increases the response time regardless of any specific ordering protocol used. Furthermore by adding more replicas
to the system, throughput can not be improved after reaching some maximum point [3]. To address these issues two optimizations have been considered regarding the execution of the commands in state-machine replication and thoroughly evaluated in [7]. Speculative execution is the approach used to address latency and state partitioning is considered to improve the throughput. These optimizations have been implemented and assessed using a replicated parallel Btree.

3.1 Speculative Execution

Based on an optimistic viewpoint, speculative execution aims at improving the response time by parallelizing the time required for executing commands in replicas with the time required to decide the order of the commands [4]. In the normal behavior once a command multicast by the coordinator of Ring Paxos reaches a replica, it will be stored in the replica’s internal buffers until the ordering is known after which the replica can safely execute the command. The idea in the speculative execution is to execute a command upon its arrival but to delay responding to the client until the order is known. If the order in which one or more commands were executed is not confirmed, the replica must rollback and re-execute them in the proper order. Although Ring Paxos is highly optimized in terms of the latency, by applying speculative execution we could get up to 16% of improvement.

3.2 State Partitioning

In the state-machine replication as each replica stores full copy of the state and receives and executes every command, no gain in throughput can be expected by adding more replicas [3]. The idea beyond state partitioning is to partition the state of the service among the servers and replicate each partition rather than the full copy. Accordingly, only a subset of the commands are forwarded to each server for execution, saving the processing capacity and bandwidth thus leading to improvements in both scalability and throughput. In [7] state partitioning is evaluated in the context of a replicated parallel Btree whereas the key range is partitioned among the replicas. State partitioning improves the throughput by almost 3 times for reads and 4 times for updates.

4 Future Work

Investigating the generality of state partitioning for other applications besides parallel Btree is one of the future works to be considered. The other intended work is to see how the protocol’s performance is infected by dynamic addition and removal of learners.

References