Scalable State-Machine Replication

Eduardo Bezerra Parisa Jalili Marandi Fernando Pedone
University of Lugano
Switzerland

Many current online services have stringent availability and performance requirements. High availability entails tolerating component failures and is typically accomplished with replication. For many online services, "high performance" essentially means the ability to serve an arbitrarily large load, something that can be achieved if the service can scale throughput when provided with additional resources (e.g., processors). In light of the requirements of modern online services, combining high availability and high performance without sacrificing consistency is a challenging endeavor of utmost importance.

Replication has been extensively studied by the research community. Early work in the database community dates back to the late 1970s and early 1980s, addressing both theoretical and practical concerns (e.g., one-copy serializability [1], primary copy replication [10]). In the distributed systems community, foundational work on object replication (i.e., non-transactional behavior) dates back to the late 1970s (e.g., state machine replication [7]). Although the topic is well-established, general-purpose replication techniques are mostly used for high availability, not performance—in fact, a replicated service is usually outperformed by its single-server implementation. Replication for performance typically sacrifices consistency: the behavior of the replicated service does not correspond to the intuitive behavior of a single-server service. Although some services can settle for weak guarantees (e.g., [4]), weak consistency renders the design of services more complex, if possible at all.

Scalability, the ability to increase performance by upgrading existing resources (scale up) or aggregating additional resources to the existing ones (scale out) is fundamental to meeting the performance requirements of modern services. Unfortunately, existing replication techniques cannot benefit from either approach. Modern servers increase processing power by aggregating multiple processors (e.g., multicore architectures), therefore scaling up a replicated service requires a replication technique that supports parallelism (multithreading) but existing techniques have at least one sequential part in their execution. Some replication techniques provide performance improvements by aggregating additional replicas (scale out), but the improvements are limited. The reason is that each new replica contains the complete service state and must execute every request, thus adding replicas will not result in improved performance, although it will lead to higher availability.

In this talk, I will present our research agenda towards techniques for *scalable general-purpose replication*. This work is motivated by the observation that guarding service designers against the complexity of replication allows them to focus on the service's inherent complexity. We aim at state-machine replication [7, 9], a fundamental approach to replication, used in many current systems (e.g., Google's Chubby [3], Spanner [6], Scatter [5]). Scaling state-machine replication is a formidable challenge since the technique relies on the assumption that execution at each replica is deterministic, which usually requires sequential execution.

We have recently proposed two execution models to scale state-machine replication: P-SMR, a model for multithreaded execution (scale up) [8], and S-SMR, a model for distributed execution (scale out) [2]. The idea behind P-SMR is to allow multiple threads to execute commands concurrently when possible and synchronize their execution when needed. In S-SMR, the server's state is decomposed into replicated partitions and multi-partition commands are executed transparently to the clients. I will detail how these systems work and discuss several open issues.

References

[1] P. Bernstein, V. Hadzilacos, and N. Goodman. *Concurrency Control and Recovery in Database Systems*. Addison-Wesley, 1987.

[2] C. E. Bezerra, F. Pedone, and R. van Renesse. Scalable state-machine replication. In *DSN*, 2014.

[3] M. Burrows. The chubby lock service for loosely coupled distributed systems. In *OSDI*, 2006.

[4] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels. Dynamo: Amazon's highly available key-value store. In *SOSP*, 2007.

[5] L. Glendenning, I. Beschastnikh, A. Krishnamurthy, and T. Anderson. Scalable consistency in scatter. In *SOSP*, 2011.

[6] J. Dean J. C. Corbett and M. Epstein et al. Spanner: Google's globally distributed database. In *OSDI*, 2012.

[7] L. Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7):558–565, 1978.

[8] P. J. Marandi, C. E. Bezerra, and F. Pedone. Rethinking state-machine replication for parallelism. In *ICDCS*, 2014.

[9] F. B. Schneider. Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM Computing Surveys*, 22(4):299–319, December 1990.

[10] M. Stonebraker. Concurrency control and consistency of multiple copies of data in distributed Ingres. *IEEE Transactions on Software Engineering*, SE-5:188–194, May 1979.