Linearizability and State-Machine Replication

Franz J. Hauck
Alexander Heß
franz.hauck@uni-ulm.de
alexander.hess@uni-ulm.de
Ulm University
Germany

SHORT ABSTRACT

Linearizability is a well-known and popular correctness criterion for concurrent and distributed systems [3, 6]. It combines atomic execution of invocations or request executions with real time, i.e. if an invocation is completed, its effect has to be seen by freshly submitted invocations. For distributed systems, linearizability guarantees that the system behaves like a non-distributed but likely concurrent system. This makes linearizability a good candidate for proving distributed systems correct.

State-machine replication (SMR) was first sketched by Lamport [4], but the tutorial written by Schneider [5] is probably the most cited fundamental work about SMR. SMR achieves fault tolerance by replicating a service on multiple nodes in a distributed system. Each replica gets the same client requests—typically in the same total order—and executes them deterministically. This ensures that all correct replicas have eventually the same state, and can replace each other in case of faults. SMR is even suitable for the Byzantine failure model (BFT) that allows for almost any faulty behaviour in a defined subset of the total number of replicas.

Linearizability was extensively used in the past to prove SMR system correct. For example, the seminal work about the first practical implementation for BFT SMR by Castro [2] takes linearizability as the correctness property of choice.

In this presentation, we argue that linearizability is too strict. Services that need to use conditional waits to coordinate internal resources and block their threads are no longer linearizable. Likewise, nested invocation, i.e. replicated services that call other replicated or non-replicated services, are no longer linearizable—at least as long as a harsh set of preconditions is not fulfilled. If concurrently executing threads communicate in both directions, the implementation is not linearizable. Although linearizability is perfect for data-oriented systems that basically comprise read and write operations, it is not for many others.

We will suggest a different correctness criterion for SMR namely interval linearizability [1] and show that it can cover all those applications that linearizability cannot. We further sketch how SMR systems can be proved correct given that the application code is already proven to be linearizable or interval linearizable.

REFERENCES

[1] Armando Castañeda, Sergio Rajsbaum, and Michel Raynal. 2018. Unifying Concurrent Objects and Distributed Tasks: Interval-Linearizability. *J. ACM* 65, 6 (Nov. 2018), 45:1–45:42. https://doi.org/10.1145/3266457

[2] Miguel Castro. 2001. *Practical Byzantine Fault Tolerance.* Ph.D. MIT.

[3] Maurice P. Herlihy and Jeannette M. Wing. 1990. Linearizability: a correctness condition for concurrent objects. *ACM Transactions on Programming Languages and Systems* 12, 3 (July 1990), 463–492. https://doi.org/10.1145/78969.78972

[4] Leslie Lamport. 1978. Time, clocks, and the ordering of events in a distributed system. *Commun. ACM* 21, 7 (July 1978), 558–565. https://doi.org/10.1145/359545.359563

[5] Fred B. Schneider. 1990. Implementing fault-tolerant services using the state machine approach: a tutorial. *Comput. Surveys* 22, 4 (1990), 299–319. https://doi.org/10.1145/98163.98167

[6] Gal Sela, Maurice Herlihy, and Erez Petrank. 2021. Linearizability: A Typo. In *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing*. 561–564. https://doi.org/10.1145/3465084.3467944 arXiv:2105.06737 [cs].