Contents

1 Verification and Checking

- *Verdi: A Framework for Implementing and Formally Verifying Distributed Systems* [12]

  Verdi is a framework for practically verifying distributed systems. Often implementations of distributed systems are too complex to be exhaustively tested, so, Verdi attempts to choose an appropriate fault model to more effectively enumerate bugs and faults.

- *Teaching Rigorous Distributed Systems with Efficient Model Checking* [8]

  While exhaustively determining bugs in a distributed system can be incredibly effective, it can, at the same time, be incredibly costly for developers. This paper purposes a model that allows students, or developers with fewer resources at their disposal to efficiently verify their systems and visually debug them. Also included, are methods to reduce the search space for potential faults in the system and to detect errors in realtime.

- *A Generalised Solution to Distributed Consensus* [2]

  This paper attempts to simplify the general consensus problem. It looks at the general consensus problem, and considers how it may be simplified in universal terms with respect to immutable state. They look specifically at the Paxos algorithm as an example. It is synomomous with consensus, though, can be incredibly difficult to understand. This generalized solution to consensus hopes to quelle some of this confusion. In analysis, they find that quorum requirements of many algorithms could in fact be weakend.

2 Consensus

- *SDPaxos: Building Efficient Semi-Decentralized Geo-replicated State Machines* [13]

  The distributed systems attemping geo-replication have run into multiple notorious problems: mainly load imbalance. SDPaxos proposes an alternative algorithm that is based on Paxos, that separates consensus into two distinct phases, replicating the commands to the nodes, and enforcing a consistent order on the nodes. This is done in an attempt to curb workload imbalance by maintaining optimal one-trip latency in two steps.

- *Mencius: building efficient replicated state machines for WANs* [7]

  Traditional consensus algorithms, like Paxos, are effective in local contexts, but in WANs, they often suffer the consequence of geographic separation. Often when Paxos, or a version of it, is implemented in a WAN there is a definite increase in network latency, decrease in network throughput and much more prevelant problems with load distribution. Mencius however, is the proposed algorithm that attempts to lessen problems traditionally associated with WANs and consensus. It does this by partitioning sequences (of commits) across multiple nodes in the network, slowly reducing the load on any one specific participant. Mencius adaptively allows nodes with less load to skip their turns and propose changes.

- *Fast Paxos* [5]
- Generalized Consensus and Paxos [4]
- MDCC: Multi-Data Center Consistency [3]
- On the correctness of Egalitarian Paxos [10]
- There Is More Consensus in Egalitarian Paxos [9]
- The FuzzyLog: A Partially Ordered Shared Log [6]

3 Databases and Implementations

- Spanner: Google's Globally-Distributed Database [1]
- Calvin: fast distributed transactions for partitioned database systems [11]

References

[1] Corbett, J. C., Dean, J., Epstein, M., Fikes, A., Frost, C., Furman, J., Ghemawat, S., Gubarev, A., Heiser, C., Hochschild, P., Hsieh, W., Kanthak, S., Kogan, E., Li, H., Lloyd, A., Melnik, S., Mwaura, D., Nagle, D., Quinlan, S., Rao, R., Rolig, L., Saito, Y., Szymaniak, M., Taylor, C., Wang, R., and Woodford, D. Spanner: Google's globally-distributed database. In *10th USENIX Symposium on Operating Systems Design and Implementation (OSDI 12)* (Hollywood, CA, 2012), USENIX Association, pp. 261–264.

[2] Howard, H., and Mortier, R. A generalised solution to distributed consensus. *CoRR abs/1902.06776* (2019).

[3] Kraska, T., Pang, G., Franklin, M. J., Madden, S., and Fekete, A. Mdcc: Multi-data center consistency. In *Proceedings of the 8th ACM European Conference on Computer Systems* (New York, NY, USA, 2013), EuroSys '13, ACM, pp. 113–126.

[4] Lamport, L. Generalized consensus and paxos. Tech. Rep. MSR-TR-2005-33, March 2005.

[5] Lamport, L. Fast paxos. *Distributed Computing 19* (October 2006), 79–103.

[6] Lockerman, J., Faleiro, J. M., Kim, J., Sankaran, S., Abadi, D. J., Aspnes, J., Sen, S., and Balakrishnan, M. The fuzzylog: A partially ordered shared log. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)* (Carlsbad, CA, Oct. 2018), USENIX Association, pp. 357–372.

[7] Mao, Y., Junqueira, F. P., and Marzullo, K. Mencius: Building efficient replicated state machines for wans. In *Proceedings of the 8th USENIX Conference on Operating Systems Design and Implementation* (Berkeley, CA, USA, 2008), OSDI'08, USENIX Association, pp. 369–384.

[8] Michael, E., Woos, D., Anderson, T., Ernst, M. D., , and Tatlock, Z. Teaching rigorous distributed systems with efficient model checking. In *EuroSys* (Dresden, Germany, Mar. 2019).

[9] Moraru, I., Andersen, D. G., and Kaminsky, M. There is more consensus in egalitarian parliaments. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles* (New York, NY, USA, 2013), SOSP '13, ACM, pp. 358–372.

[10] Sutra, P. On the correctness of egalitarian paxos. *CoRR abs/1906.10917* (2019).

[11] Thomson, A., Diamond, T., Weng, S.-C., Ren, K., Shao, P., and Abadi, D. J. Calvin: Fast distributed transactions for partitioned database systems. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data* (New York, NY, USA, 2012), SIGMOD '12, ACM, pp. 1–12.

[12] Wilcox, J. R., Woos, D., Panchekha, P., Tatlock, Z., Wang, X., Ernst, M. D., and Anderson, T. Verdi: A framework for implementing and formally verifying distributed systems. In *Proceedings of the 36th ACM SIGPLAN Conference on Programming Language Design and Implementation* (New York, NY, USA, 2015), PLDI '15, ACM, pp. 357–368.

[13] Zhao, H., Zhang, Q., Yang, Z., Wu, M., and Dai, Y. Sdpaxos: Building efficient semi-decentralized geo-replicated state machines. In *ACM Symposium on Cloud Computing 2018 (SoCC)* (October 2018), ACM.