Optimized BFT Replication from Authenticated Logging

Hanish Gogada
University of Stavanger

Christian Berger
Friedrich-Alexander-Universität Erlangen-Nürnberg

Leander Jehl
University of Stavanger

Hans P. Reiser
Reykjavik University

Hein Meling
University of Stavanger

1 MOTIVATION

Many BFT protocols improve scalability using optimization strategies effective under specific favorable conditions (e.g., [3, 5, 6]). In adverse scenarios, these optimizations become less effective or even defunct, necessitating a switch to a more resilient but less effective protocol like PBFT [2]. It is non-trivial to detect whether or not the operating conditions are favorable or adversarial. Often approaches rely on repeated trial-and-error to find a working system configuration and ignore the actual operating conditions, such as the latency between replicas and prior misbehavior, thus resulting in poor performance.

Various systems that consider the operating conditions rely only on local measurements to select a system configuration (e.g., [4]). This is problematic because local measurements across the replicas may be inconsistent, making it challenging to make global configuration decisions. In the Byzantine fault model, there is a lack of transparency and trust if a single replica, e.g., the leader, can make such decisions based only on its local measurements because it is impossible to verify that the decision is based on real measurements.

In this presentation, we introduce SmartLog, a logging framework that collects and analyzes metrics for smart configuration decisions to improve performance despite the presence of faults. SmartLog presents local measurements in global data structures, to enable consistent decisions and hold replicas accountable if they do not perform according to their reported measurements.

2 DESIGN

We advocate for a holistic, measurement-based approach to accurately identify the current operating conditions, promoting aggressive use of efficient protocols and reducing fallback to less efficient ones. We accomplish this through a shared append-only log of measurements.

SmartLog is an integrated shared log for recording various metrics and computing efficient system configurations from these metrics. SmartLog extends a generic replicated state machine with sensors and monitors to capture and evaluate different metrics (see Figure 1). Individual replicas instrumented with sensors record measurements in the log, and corresponding monitors at replicas collate these measurements into data structures that are used to and deploy efficient configurations. Furthermore, SmartLog enables replicas to make consistent configuration decisions based on the same information. The log also provides transparency, allowing all replicas to verify decisions and recognize faulty behaviors.

Moreover, some optimization techniques can be costly to (deterministically) evaluate for large configuration sizes. Thus, SmartLog allows for heuristic optimization techniques, where the resulting non-deterministic configurations are logged, such that the replicas

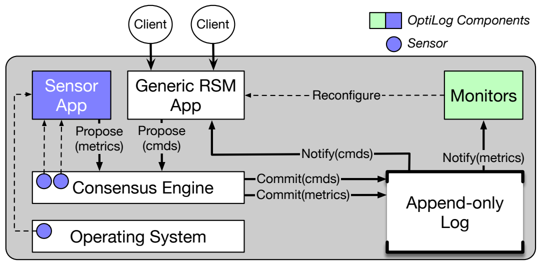

Figure 1: SmartLog's component architecture.

can consistently determine a global ranking of configurations. Also, SmartLog supports collaborative optimization techniques where the search space is partitioned and distributed across replicas.

3 IMPLEMENTATION & EVALUATION

To showcase the potential of SmartLog, we apply our implementation to the BFT protocol Kauri [5] to boost its performance in heterogenous networks by selecting a fast tree configuration which is decided based on logged replica latency measurements and indications about faulty replicas. This implementation is based on an open-source [1] BFT framework based on HotStuff [7]. Our experimental evaluations reveal that in a geo-replicated deployments, Kauri enhanced with SmartLog can select tree configurations with up to 33% lower latency. In particular, the estimate on actual faults in the system, which can be provided by SmartLog, allows the implementation to better balance performance and robustness.

REFERENCES

[1] 2024. Relab HotStuff. https://github.com/relab/hotstuff. (2024).

[2] Miguel Castro and Barbara Liskov. 1999. Practical Byzantine Fault Tolerance. In *Proceedings of the Third Symposium on Operating Systems Design and Implementation (OSDI '99)*. USENIX Association, USA, 173–186.

[3] Guy Golan Gueta, Ittai Abraham, Shelly Grossman, Dahlia Malkhi, Benny Pinkas, Michael Reiter, Dragos-Adrian Seredinschi, Orr Tamir, and Alin Tomescu. 2019. SBFT: A Scalable and Decentralized Trust Infrastructure. In *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. 568–580. https://doi.org/10.1109/DSN.2019.00063

[4] Michael G. Merideth, Florian Oprea, and Michael K. Reiter. 2009. When and How to Change Quorums on Wide Area Networks. In *2009 28th IEEE International Symposium on Reliable Distributed Systems*. 12–21. https://doi.org/10.1109/SRDS.2009.35

[5] Ray Neiheiser, Miguel Matos, and Luís Rodrigues. 2021. Kauri: Scalable BFT Consensus with Pipelined Tree-Based Dissemination and Aggregation. In *Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles (SOSP '21)*. Association for Computing Machinery, New York, NY, USA, 35–48. https://doi.org/10.1145/3477132.3483584

[6] Chrysoula Stathakopoulou, Tudor David, Matej Pavlovic, and Marko Vukolić. 2022. [Solution] Mir-BFT: Scalable and Robust BFT for Decentralized Networks. *Journal of Systems Research* 2, 1 (2022). https://doi.org/10.5070/SR32159278

[7] Maofan Yin, Dahlia Malkhi, Michael K. Reiter, Guy Golan Gueta, and Ittai Abraham. 2019. HotStuff: BFT Consensus with Linearity and Responsiveness. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing (PODC '19)*. Association for Computing Machinery, New York, NY, USA, 347–356. https://doi.org/10.1145/3293611.3331591