Blazingly Fast BFT Consensus with MERCURY

Christian Berger[†] Lívio Rodrigues[⋆] Hans P. Reiser[‡,†] Vinícius Cogo[⋆] Alysson Bessani[⋆]

[⋆]LASIGE, Faculdade de Ciências Universidade de Lisboa, Portugal
[†]University of Passau, Germany [‡]Reykjavik University, Iceland

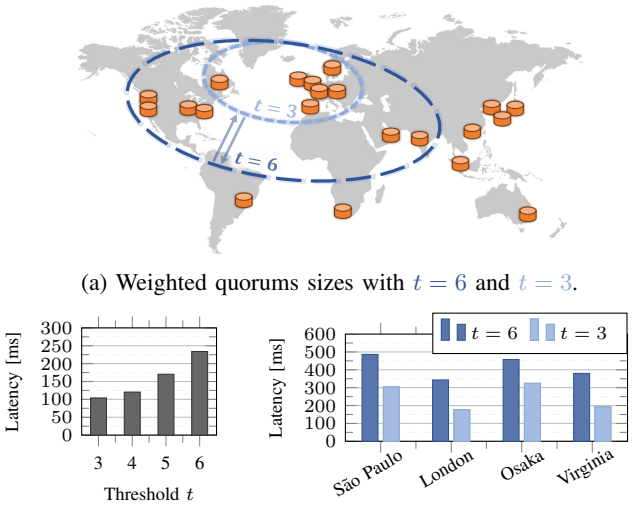
***Abstract*—Blockchain technology has renewed interest in planetary-scale Byzantine fault-tolerant (BFT) state machine replication (SMR). While recent works focus on scalability and throughput, few address latency. We present the idea of MERCURY, a transformation for quorum-based BFT consensus that uses an adaptive resilience threshold. MERCURY employs *adaptive weighted replication* to assign high voting power to specific replicas, thus yielding smaller quorums that speed up consensus. To maintain SMR safety and liveness guarantees with optimal resilience, MERCURY employs two modes of operation and BFT forensics. Experiments with replicas worldwide show MERCURY finalizes client requests in less than 0.4 s, half the time needed by a PBFT-like protocol with optimal consensus latency.**

I. INTRODUCTION

State machine replication (SMR) is an approach to tolerate faults in distributed systems by coordinating client interactions with a set of n independent replicas [1]. Recently, many scalable (BFT) SMR protocols have emerged for blockchain infrastructures, such as HotStuff [2], SBFT [3], Tendermint [4], Mir-BFT [5], RedBellyBC [6], Kauri [7], IA-CCF [8], and the Dumbo family [9], [10]. These protocols employ some dynamically elected leader [2]–[4], [7], [11], use multiple leaders [5], [12], or are leaderless [6], [10], [13], [14].

Nevertheless, the consensus in all these cases requires communication involving a quorum of replicas under the assumption that the adversary controls no more than a *fixed resilience threshold* of $t = \lfloor \frac{n-1}{3} \rfloor$ replicas. Often, the quorum size for proceeding to the next protocol stage depends on this threshold, a Byzantine *t-dissemination quorum* with $\lceil \frac{n+t+1}{2} \rceil$ replicas [15]. This size equals roughly $\frac{2}{3}$ of all replicas if an *optimal* resilience threshold is used. For geo-replicated or planetary-scale systems like permissioned blockchains (e.g., [6], [16]) with tens of nodes distributed worldwide, optimizing end-to-end client latency is challenging for two reasons. First, theoretical lower bounds define that at least three communication steps are required for reaching consensus without giving up the optimal resilience [17], [18]. Second, there are physical limits that bound link transmission speed to a fraction of the speed of light (e.g., $0.67c$ [19]). Improving throughput, on the other hand, is a much more popular objective that can be achieved by parallelizing/distributing tasks (e.g., [6], [14]), improving bandwidth usage (e.g., [20]), or simply by using a better infrastructure (e.g., better links). Nonetheless, the goal of globally ordering transactions in a fraction of a second is still far from existing systems [21].

Using smaller quorums of closer replicas can significantly decrease SMR latency [22], [23]. The challenge lies in ensuring these smaller, faster quorums intersect in sufficiently many replicas with all other quorums of the system. Such smaller, intersecting quorums can be built using *weighted replication*, giving faster replicas more voting power.



(a) Weighted quorums sizes with $t = 6$ and $t = 3$.

(b) Consensus latency *vs.* resilience threshold.

(c) End-to-end transaction latencies observed by clients in different regions.

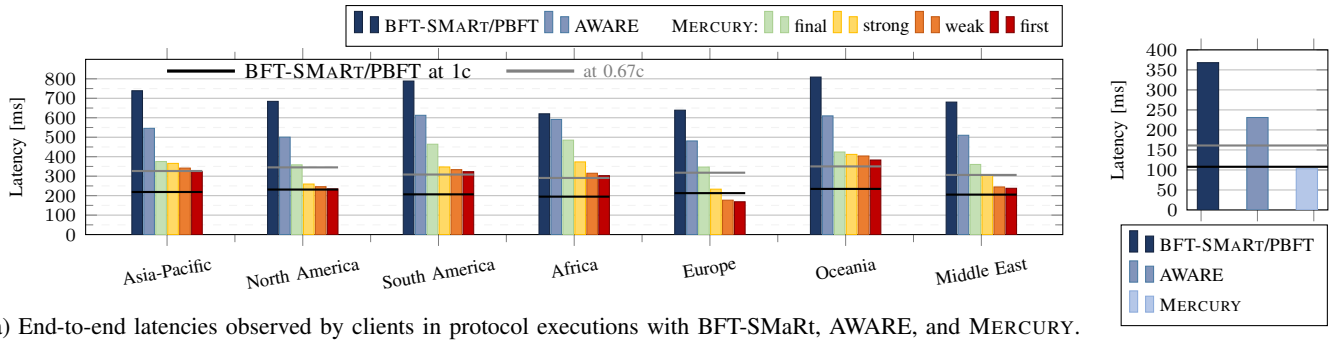
Fig. 1: Weighted quorums composition and resulting BFT SMR latency for different resilience thresholds (t) in our $n = 21$ AWS setup.

II. IMPROVING LATENCY WITH COMPACT QUORUMS

Figure 1 illustrates how a geo-replicated system can progress faster with smaller quorums. It considers a weighted system [23] with $n = 21$ replicas dispersed across all 21 AWS regions (see Figure 1a). When configured for maximum resilience, this system tolerates up to $t = 6$ Byzantine replicas (the highest integer satisfying $t < \frac{n}{3}$) with $\Delta = 2$ spare replicas. The smallest weighted consensus quorum Q_v^6 contains 13 replicas, only one replica less than using non-weighted replication. Alternatively, when configured for $t = 3$ failures, the smallest weighted quorum Q_v^3 contains only 7 replicas, with $\Delta = 11$. This quorum can comprise closer replicas that can exchange votes with each other faster, accelerating the consensus protocol stages (see Figure 1b) and resulting in end-to-end latency improvements around the globe (see Figure 1c).

III. MERCURY DESIGN

MERCURY offers dynamic self-optimization during runtime by adjusting the resilience threshold and modifying weights



(a) End-to-end latencies observed by clients in protocol executions with BFT-SMaRt, AWARE, and MERCURY. Client results are averaged over all regions per continent.

(b) Consensus latency.

Fig. 2: Achievable latency gains for the $n = 21$ AWS setup.

to facilitate the formation of smaller consensus quorums, resulting in lower latencies. To accomplish this, it pursues the best of both worlds: preserving the system's maximum resilience through the use of diagnosis and repair, ensuring system consistency, while consistently striving to expedite consensus instances by optimizing the system for the expected common scenario with minimal or no failures. This dual approach is realized through two modes of operation, inspired by related methods for enhancing performance [24] and resource efficiency [25] of replicated state machines.

The system starts in *conservative mode*, where it runs instances of a quorum-based consensus protocol capable of tolerating t failures. After a specific number of successfully completed consensus instances, the system transitions to the *fast mode*, characterized by a more optimistic setup that can tolerate only t_{fast} failures. In this fast mode, as long as the leader remains correct and the actual failures stay within the threshold t_{fast}, MERCURY continues to use smaller quorums, expediting the completion of consensus. However, should the observed latency gains fall short of expectations, if the leader is found to be faulty, or if correct replicas identify inconsistencies, MERCURY switches back to the conservative mode. For instance, SMR liveness can be endangered if the protocol operates with a lower threshold $t_{fast} < t$ and there are $f > t_{fast}$ Byzantine replicas that stay silent. To avoid this scenario, we utilize two distinct modes of operation: If the system blocks or equivocates, we abort the execution of the fast mode of MERCURY and start a more resilient protocol instance, which uses the maximum resilience threshold t.

A key insight of our work is that we can utilize *BFT protocol forensics* [26] in a novel way. Instead of using it as a forensic tool to investigate the "day after", we leverage it as a protective countermeasure against equivocations. In BFT protocol forensics, the client is the one who detects conflicting values based on replicas' logged signed messages and pinpoints equivocating replicas. In MERCURY, the responsibility of detecting faulty replicas is imposed on all correct replicas so the system can autonomously detect and expel equivocating parties. Using $t_{fast} = \lceil \frac{t}{2} \rceil$ guarantees that audits are always supported for up to t faulty replicas. The system addresses violations by removing identified violators from the system and restoring correct replicas' divergent decisions to a consistent state. This continuous auditing serves as both a means of recovery and a deterrent against attacks, as it identifies and expels perpetrators from the system.

However, the possibility of rolling back decisions on replicas introduces a challenge. It may result in the undoing of operation outcomes observed by clients, impacting the durability of these operations. To resolve this, we increase the matching reply requirements of clients so that clients can determine when an operation is considered completed in the system (and survives a rollback), guaranteeing *linearizability* [27].

We further extend the programming model of BFT SMR with *Byzantine Correctables* that allow client applications to use incremental consistency guarantees [28], which simplifies and abstracts client-side speculation. Clients can lower their operation latency even further using this abstraction.

IV. PRELIMINARY RESULTS

We use *c5.xlarge* instances on the AWS cloud for deploying a client and a replica in each of the $n = 21$ AWS regions (depicted in Figure 1a). All clients send 400-bytes requests simultaneously and continuously. Finally, *request latency* is the average end-to-end protocol latency computed by a client after completing all operations. For a better exposition, we group the 21 clients' results by the continent they are located in, reporting only their regional averages (see Figure 2).

First, we observe that MERCURY significantly accelerates consensus, leading to a speedup of $3.57\times$ faster decisions. Second, accelerating consensus decisions also leads to faster request latencies observed by clients worldwide (see Figure 2). Averaged over all client regions, MERCURY leads to a speedup of $1.87\times$ over BFT-SMART for clients' observed end-to-end request latencies with finalization (a competitive protocol AWARE [29], which uses only leader and weight optimization but no threshold adaption leads to $1.33\times$ only). Our results also show that even higher speedups can be achieved by employing the incremental consistency levels of MERCURY's implementation of *Byzantine Correctables*. Furthermore, MERCURY can often achieve a similar latency as running its base protocol on theoretically optimal internet links (transmitting at 67% of the speed of light).

References

[1] F. B. Schneider, "Implementing fault-tolerant services using the state machine approach: A tutorial," *ACM Computing Surveys (CSUR)*, vol. 22, no. 4, pp. 299–319, 1990.

[2] M. Yin, D. Malkhi, M. K. Reiter, G. G. Gueta, and I. Abraham, "Hotstuff: BFT consensus with linearity and responsiveness," in *Proc. of the 38th ACM Symp. on Principles of Distributed Computing (PODC)*. New York, NY, USA: Association for Computing Machinery, 2019, pp. 347–356.

[3] G. G. Gueta, I. Abraham, S. Grossman, D. Malkhi, B. Pinkas, M. Reiter, D.-A. Seredinschi, O. Tamir, and A. Tomescu, "SBFT: a scalable and decentralized trust infrastructure," in *Proc. of the 49th Annual IEEE/IFIP Int. Conf. on Dependable Systems and Networks (DSN)*. IEEE, 2019, pp. 568–580.

[4] D. Cason, E. Fynn, N. Milosevic, Z. Milosevic, E. Buchman, and F. Pedone, "The design, architecture and performance of the Tendermint blockchain network," in *Proc. of the 40th IEEE Int. Symp. on Reliable Distributed Systems (SRDS)*. IEEE, 2021, pp. 23–33.

[5] C. Stathakopoulou, T. David, and M. Vukolić, "Mir-BFT: High-throughput BFT for blockchains," 2019.

[6] T. Crain, C. Natoli, and V. Gramoli, "Red Belly: A secure, fair and scalable open blockchain," in *Proc. of the 42nd IEEE Symp. on Security and Privacy (S&P)*. IEEE, 2021, pp. 466–483.

[7] R. Neiheiser, M. Matos, and L. Rodrigues, "Kauri: Scalable BFT consensus with pipelined tree-based dissemination and aggregation," in *Proc. of the 28th ACM SIGOPS Symp. on Operating Systems Principles (SOSP)*. New York, NY, USA: Association for Computing Machinery, 2021, pp. 35–48.

[8] A. Shamis, P. Pietzuch, B. Canakci, M. Castro, C. Fournet, E. Ashton, A. Chamayou, S. Clebsch, A. Delignat-Lavaud, M. Kerner, J. Maffre, O. Vrousgou, C. M. Wintersteiger, M. Costa, and M. Russinovich, "IA-CCF: Individual accountability for permissioned ledgers," in *Proc. of the 19th USENIX Symp. on Networked Systems Design and Implementation (NSDI)*. Berkeley, CA: USENIX Association, 2022, pp. 467–491.

[9] Y. Lu, Z. Lu, and Q. Tang, "Bolt-Dumbo transformer: Asynchronous consensus as fast as pipelined BFT," in *Proc. of the 29th ACM Conference on Computer and Communications Security (CCS)*. New York, NY, USA: Association for Computing Machinery, 2022, pp. 2159–2173.

[10] Y. Gao, Y. Lu, Z. Lu, J. X. Qiang Tang, and Z. Zhang, "Dumbo-NG: Asynchronous consensus with throughput-oblivious latency," in *Proc. of the 29th ACM Conference on Computer and Communications Security (CCS)*. New York, NY, USA: Association for Computing Machinery, 2022, pp. 1187–1201.

[11] X. Sui, S. Duan, and H. Zhang, "Marlin: Two-phase bft with linearity," in *Proc. of the 52nd Annual IEEE/IFIP Int. Conf. on Dependable Systems and Networks (DSN)*. IEEE, 2022, pp. 54–66.

[12] S. Alqahtani and M. Demirbas, "BigBFT: A multileader Byzantine fault tolerance protocol for high throughput," in *Proc. of the IEEE Int. Performance, Computing, and Communications Conference (IPCCC)*. IEEE, 2021, pp. 1–10.

[13] K. Antoniadis, A. Desjardins, V. Gramoli, R. Guerraoui, and I. Zablotchi, "Leaderless consensus," in *Proc. of the 41st IEEE Int. Conf. on Distributed Computing Systems (ICDCS)*. IEEE, 2021, pp. 392–402.

[14] H. Zhang and S. Duan, "Pace: Fully parallelizable bft from reproposable byzantine agreement," in *Proc. of the 29th ACM Conference on Computer and Communications Security (CCS)*. New York, NY, USA: Association for Computing Machinery, 2022, pp. 3151–3164.

[15] D. Malkhi and M. Reiter, "Byzantine quorum systems," *Distributed computing*, vol. 11, no. 4, pp. 203–213, 1998.

[16] E. Androulaki *et al.*, "Hyperledger Fabric: a distributed operating system for permissioned blockchains," in *Proc. of the 13th European Conference on Computer Systems (EuroSys)*. New York, NY, USA: Association for Computing Machinery, 2018.

[17] J.-P. Martin and L. Alvisi, "Fast Byzantine consensus," *IEEE Transactions on Dependable and Secure Computing (TDSC)*, vol. 3, no. 3, pp. 202–215, 2006.

[18] P. Kuznetsov, A. Tonkikh, and Y. X. Zhang, "Revisiting optimal resilience of fast Byzantine consensus," in *Proc. of the 40th ACM Symp. on Principles of Distributed Computing (PODC)*. New York, NY, USA: Association for Computing Machinery, 2021, pp. 343–353.

[19] K. Kohls and C. Diaz, "Verloc: Verifiable localization in decentralized systems," in *Proc. of the 31st USENIX Security Symp. (USENIX Security)*. Berkeley, CA: USENIX Association, 2022, pp. 2637–2654.

[20] C. Stathakopoulou, M. Pavlovic, and M. Vukolić, "State machine replication scalability made simple," in *Proc. of the 17th European Conference on Computer Systems*, 2022, p. 17–33.

[21] V. Gramoli, R. Guerraoui, A. Lebedev, C. Natoli, and G. Voron, "Diablo: A benchmark suite for blockchains," in *Proc. of the 18th European Conference on Computer Systems*, 2023, p. 540–556.

[22] F. Junqueira, Y. Mao, and K. Marzullo, "Classic Paxos vs. fast Paxos: caveat emptor," in *Proceedings of USENIX Hot Topics in System Dependability (HotDep)*. Berkeley, CA: USENIX Association, 2007.

[23] J. Sousa and A. Bessani, "Separating the wheat from the chaff: An empirical design for geo-replicated state machines," in *Proc. of the 34th IEEE Int. Symp. on Reliable Distributed Systems (SRDS)*. IEEE, 2015, pp. 146–155.

[24] P.-L. Aublin, R. Guerraoui, N. Knežević, V. Quéma, and M. Vukolić, "The next 700 BFT protocols," *ACM Transactions on Computer Systems (TOCS)*, vol. 32, no. 4, pp. 1–45, 2015.

[25] T. Distler, C. Cachin, and R. Kapitza, "Resource-efficient Byzantine fault tolerance," *IEEE Transactions on Computers (TC)*, vol. 65, no. 9, pp. 2807–2819, 2015.

[26] P. Sheng, G. Wang, K. Nayak, S. Kannan, and P. Viswanath, "BFT protocol forensics," in *Proc. of the 28th ACM Conference on Computer and Communications Security (CCS)*. New York, NY, USA: Association for Computing Machinery, 2021, pp. 1722–1743.

[27] M. Castro and B. Liskov, "Practical Byzantine fault tolerance," in *Proc. of the 3rd Symp. on Operating Systems Design and Implementation (OSDI)*. Berkeley, CA: USENIX Association, 1999, pp. 173–186.

[28] R. Guerraoui, M. Pavlovic, and D.-A. Seredinschi, "Incremental consistency guarantees for replicated objects," in *Proc. of the 12th USENIX Symp. on Operating Systems Design and Implementation (OSDI)*. Berkeley, CA: USENIX Association, 2016, pp. 169–184.

[29] C. Berger, H. P. Reiser, J. Sousa, and A. N. Bessani, "AWARE: Adaptive wide-area replication for fast and resilient Byzantine consensus," *IEEE Transactions on Dependable and Secure Computing (TDSC)*, vol. 19, no. 3, pp. 1605–1620, 2022.