Poster: Optimizing Tree-based Quorum Certification for BFT Systems at Planetary-Scale

Christian Berger
University of Passau
Passau, Germany

Hans P. Reiser
Reykjavik University
Reykjavik, Iceland

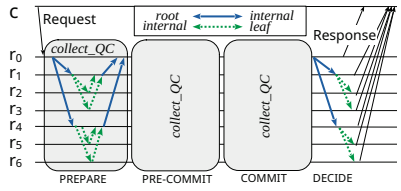

Fig. 1: Tree-based dissemination and aggregation of QCs in Kauri [1] is a load balancing technique for improved protocol scalability.

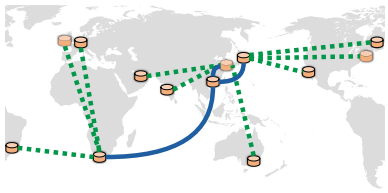

Fig. 2: The latency of *collect_QC* depends on the chosen tree configuration, because the latencies of communication links vary.

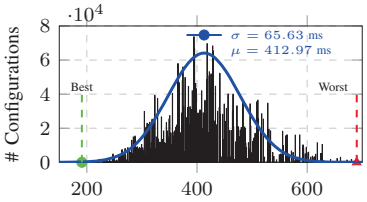

Fig. 3: Distribution of all tree configurations in respect to their *collect_QC* latency.

***Abstract*—Recent research work on Byzantine fault-tolerant (BFT) protocols focuses on the question of how to improve scalability. A particular scalability-enhancing technique used in several BFT protocols such as ByzCoin, Motor, and Kauri is load balancing the communication through a tree-based dissemination and aggregation of quorum certificates. However, current evaluations of these algorithms often overlook how heterogeneous network latencies impact system performance. In this work, we first explore the influence of heterogeneous latencies as found in planetary-scale networks towards tree-based BFT protocol performance. Subsequently, we propose an optimization method which searches for an optimal communication tree by adapting tree configurations to given network conditions, thus minimizing system latency. Simulation-based experimentation reveals that the margin of latency improvements is substantial: An optimized configuration is up to $2.16\times$ faster than the mean. We further discuss required modifications to support reconfigurations in Kauri with fast configurations, thus assisting automated deployments.**

I. Introduction

While BFT protocols are newcomers in the blockchain landscape, they mark a significant paradigm shift as blockchain systems transition from Proof-of-Work [2] schemes to BFT consensus to benefit from energy-efficiency, higher throughput, or proven system properties. Despite these promises, an open challenge to BFT protocol design is how scalability can be improved to allow for larger system sizes [3]. There are many new ideas of how this goal can be achieved [4], and a particular promising idea relies on solving the *leader bottleneck* problem by introducing a *load balancing* technique. Load balancing means shifting the communication burden from the leader to a more evenly and thus more efficient utilization of bandwidth, thus optimizing resource allocation [5]. Recent solutions propose tree structures to achieve load balancing and examples include ByzCoin [6], OmniLedger [7] Motor [8] and Kauri [1].

In a tree-based BFT protocol, the leader resides at the root position of the tree and invokes a method to collect a *quorum certificate* (QC) with communication happening only between adjacent layers of the tree. Information (such as a proposal) is disseminated "top-down" from root to internals and eventually to leaf nodes while votes are aggregated "bottom-up": Cryptographic primitives like *multi-signatures* [9] allow to iteratively combine votes into single signatures which ascend the tree until reaching the root (leader) which, as soon as sufficiently many votes are received, forms a QC and proceeds to the next stage of the consensus algorithm (see *collect_QC* in Fig. 1).

The adoption of tree-based communication represents a load balancing technique and contributes significantly to increasing scalability in BFT protocols. The number of nodes contacted by each node is within $\mathcal{O}(\sqrt{n})$ and remains manageable even in large-scale networks with n nodes. For instance, in Kauri and assuming a system size of 100 nodes, the leader would only need to communicate with 10 (the *internal* tree) nodes.

II. Open Problem

BFT algorithms that introduced tree-based quorum certification have been extensively evaluated in (emulated) homogeneous networks, in which the latency between all possible pairs of nodes is a single, specified value [1], [6]–[8]. We argue that a realistic deployment scenario for a blockchain use case should assume a geographic dispersion of nodes which implies heterogeneous network latencies between them[1]. In this case, the latency of a leader collecting a quorum certificate depends on how the tree is structured, and the overall performance of the algorithm varies across different tree configurations. Selecting an optimal tree configuration by hand is difficult in practice and requires automation that is not yet provided

[1]Kauri evaluated a single heterogeneous network scenario in which 60 nodes were distributed across 6 regions (Sect. 7.9 in [1]). The limited amount of regions made a manual configuration easy as the leader and all internal tree nodes could be deployed within the same, well-connected region (Oregon).

2833-292X/24/$31.00 ©2024 IEEE
DOI 10.1109/DSN-S60304.2024.00044

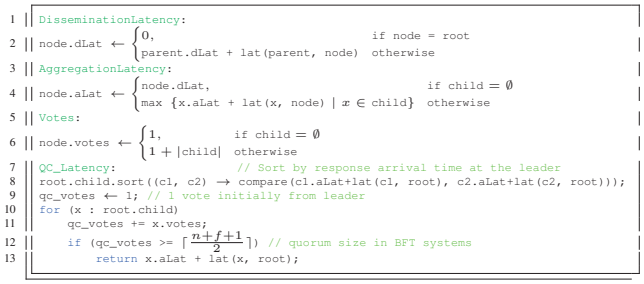

Fig. 4: Compute the latency for quorum certification over a tree.

by existing solutions. To illustrate this problem, consider a "small" deployment scenario of 13 nodes as shown in Figure 2: The figure shows the structure of the fastest tree to collect a QC in just 191 ms while the slowest tree (not pictured) would require 690 ms, which is 3.6 times slower. Simulation-based experimentation reveals that picking a tree configuration at random is not ideal either, considering that most tree configurations display latencies close to the mean of 412 ms as shown in Figure 3. The problem becomes increasingly complex the more regions are available for deployment. This leads to our first research question: *Given a set of network characteristics (or: geographic regions), how can we automatically adjust the employed tree configuration to optimize the latency of the system?* A second research question emerges when a subset of nodes might become unavailable due to failures and we still intend to find a system reconfiguration that performs fast.

III. Proposed Solution

We present our solution by extending the Kauri protocol [1] to equip it with mechanisms that (1) automate finding a fast tree configuration for a given deployment and (2) support efficient reconfiguration in the system with fast configurations.

A. Prerequisites for Automated Optimization

1) Latency Matrix: To implement our optimization scheme, we first require knowledge that can be exploited to search for fast system configurations. For this purpose we must know the latency, represented by a function $\texttt{lat}(x, y)$ between all pairs of nodes (x, y). In practice, this can be implemented by crafting a self-monitoring mechanism into the protocol where nodes periodically ping each other, and record and exchange their measurements. Another (albeit less decentralized) solution would be to rely on an external monitor such as CloudPing[2] which collects statistics for inter-regions latency of a large cloud infrastructure provider. We assume each node knows or can look up the regions of any other node through some fault-tolerant directory service and can read an available *latency matrix* through a function $\texttt{lat}(x, y)$.

2) Deterministic Optimization: Once a fast-performing tree configuration is found, all nodes need to deterministically switch to it. To achieve this, a mechanism for deterministic optimizations is implemented into Kauri, which periodically attempts to optimize the system configuration at a logical point in time, i.e., after a certain interval of protocol rounds passed.

B. Finding a Fast Tree Configuration

1) Assessing Tree Configurations: We first define a fitness function *QC_Latency* that defines the quality of a tree configuration by computing the time a leader needs to collect a quorum certificate (see Figure 4).

This function can be implemented in three steps: In the first step, we traverse the tree downwards, calculating the dissemination times at every node by adding the link latency between node and parent to the dissemination time of the parent (Line 2). In the second step, we proceed backward by computing the aggregation times of multi-signatures bottom-up at each node when all votes (signatures) from children could be collected (Line 4). Finally, we calculate the latency for quorum certification: It is a Byzantine dissemination quorum [10] of the *fastest responses* that allows the leader to form a QC and thus determine the overall speed (Lines 6-13).

2) Navigating the Configuration Space: Because the exploration space of trees quickly grows for larger n, it is not feasible to evaluate all possible configurations. To solve this problem, we employ a heuristic method to find a reasonably fast configuration efficiently. In particular, we currently use *simulated annealing* [11] which employs *QC_Latency* as a fitness function to evaluate configurations. Simulated Annealing is a randomized search strategy that remembers and slightly modifies good solutions to proactively find improvements. It may also accept a temporary worsening with a certain acceptance probability to avoid getting stuck in a local optimum.

C. Supporting Reconfigurations with Fast Configurations

In the case of faulty nodes, Kauri is the first tree-based BFT protocol to support *quick recovery*, a guarantee that a *robust tree*[3] is found after at most $f + 1$ reconfigurations assuming that $f < m$ where m is the tree's fanout. This is achieved by a partitioning of all nodes into $m + 1$ disjoint sets of size $\frac{n}{m+1}$, implying one of these sets must exclusively consist of correct nodes which can be used to form the top of a robust tree (leader and internal nodes). To improve this idea and outline it with our goal of achieving fast latency in heterogeneous networks, we propose to adopt the partitioning algorithm to account for network characteristics. For instance, the algorithm can adapt k-means clustering using geographic proximity (for instance haversine formula) as a distance metric to put nodes that are close together into the same cluster. The algorithm partitions regions into $k = m+1$ clusters by iteratively assigning regions to the nearest cluster centroid and then updates the centroids based on the mean of the regions assigned to each cluster. This follows the intuition that a leader and its internal nodes should be well connected by residing in the same cluster. For every of the k clusters, we use *QC_Latency* to determine a suitable cluster leader and for the assignment of leaf nodes to the internal nodes of the cluster, yielding a fast configuration.

[2] See https://www.cloudping.co/grid.

[3] *Robust tree*: The leader is correct and all correct nodes are connected to the leader over an existing path that only consists of correct nodes.

ACKNOWLEDGEMENTS

This work has been funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) grant number 446811880 (BFT2Chain).

REFERENCES

[1] 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)*, 2021, pp. 35–48.

[2] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[3] M. Vukolić, "The quest for scalable blockchain fabric: Proof-of-work vs. BFT replication," in *International Workshop on Open Problems in Network Security*, 2015, pp. 112–125.

[4] C. Berger, S. Schwarz-Rüsch, A. Vogel, K. Bleeke, L. Jehl, H. P. Reiser, and R. Kapitza, "Sok: Scalability techniques for bft consensus," in *2023 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. IEEE, 2023, pp. 1–18.

[5] M. J. Amiri, C. Wu, D. Agrawal, A. E. Abbadi, B. T. Loo, and M. Sadoghi, "The bedrock of BFT: A unified platform for BFT protocol design and implementation," *preprint arXiv:2205.04534*, 2022.

[6] E. K. Kogias, P. Jovanovic, N. Gailly, I. Khoffi, L. Gasser, and B. Ford, "Enhancing bitcoin security and performance with strong consistency via collective signing," in *25th USENIX Security Symposium (USENIX Security 16)*, 2016, pp. 279–296.

[7] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford, "Omniledger: A secure, scale-out, decentralized ledger via sharding," in *2018 IEEE Symposium on Security and Privacy*, 2018, pp. 583–598.

[8] E. Kokoris-Kogias, "Robust and scalable consensus for sharded distributed ledgers," *Cryptology ePrint Archive*, 2019.

[9] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," in *International conference on the theory and application of cryptology and information security*. Springer, 2001, pp. 514–532.

[10] D. Malkhi and M. Reiter, "Byzantine quorum systems," in *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, 1997, pp. 569–578.

[11] S. Kirkpatrick, C. D. Gelatt Jr, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983. [Online]. Available: https://science.sciencemag.org/content/220/4598/671