# Unleash Full Potential Of Scalability In Leaderless BFT By Execution Attestation

Anonymous Author(s)

## ABSTRACT

The work is awesome.

## 1 INTRODUCTION

GD: Scope. What problem this paper works on. Why the problem is important. BFT [5]. Demand high-throughput permissioned BFT protocols. Goal: be scalable in performance. The throughput should scale up with the number of nodes.

GD: Current status. What are the previous works on this problem. Leaderless BFT protocols resolve leader bottleneck, parallelize block proposal and validation. Autobahn [3]. Bullshark [8]. Narwhal and Tusk [2].

Stateless BFT protocols outsource the state and the execution to storage nodes and parallelize the execution among them. SlimChain [9].

Sharded BFT protocols shard the transaction processing, parallelize the whole consensus and execution. Elastico [6]. Omniledger [4]. RapidChain [10].

GD: Gap. What feature that previous works failed to secure and why they give up on it. Why the feature matters for the problem. Leaderless BFT loses its effectiveness as the execution becomes the throughput bottleneck, and sharded BFT enforces unconventional security model. Observation: consensus takes bandwidth resource and execution takes computation resource, and data center per-host bandwidth increases much faster than computation. Moreover, bandwidth consumption of block dissemination can be optimized *globally/systematically*. Chop chop [1]. Kauri [7]. Local execution parallelism can be leveraged, but it won't provide the scalability we desire.

Stateless BFT and sharded BFT reduce execution overhead systematically, but their approaches adopt unconventional security model and introduce performance penalty. Stateless BFT employs off-chain storage nodes. Although it claims that off-chain nodes are untrusted and are allowed to behave arbitrarily, it can only improve performance based on sensible execution of the storage nodes, e.g., their execution should be rarely conflicted with each other, or they should execute at all. Furthermore, stateless BFT gives up the responsibility of ordering as well, losing control of fairness. Pompē [11]. From performance aspect, stateless BFT suffers from additional latency overhead of communicating with the storage nodes. Even more performance penalty when the execution of storage nodes conflicts with each other.

Sharded BFT adopts all kinds of customizations to the security model. Reduced failure rate. Pre-determined failure clusters. Synchronous network model. Trusted hardware. Also pay performance penalty for cross-shard coordination.

Problem settings: permissioned BFT with conventional security model and be scalable in execution. Match the security model of conventional BFT protocols gives us practicality and wider adoption. Match the scalability of stateless and sharded BFT to maximize performance.

GD: Approach. What we have noticed and how it leads to our solution. Intuitively, how the solution different from previous approaches so not missing the gapping feature, and how it covers the other features as previous works despite being different. Observation from leaderless BFT and reliable dissemination works: block and quorum certificate dissemination can be systematically optimized because they feature *transparent authentication.* Received blocks and certificates can be independently attested by the receiver. Verifiable computation and authenticated data structure bring this opportunity of remote attestation to the execution as well. Thus, in stateless BFT, every transaction only need to be executed once without compromising safety, superior to the sharded approach.

However, in data center, nodes need not be stateless. Storage is abundant, each node can afford to store full state. Having the nodes themselves collaboratively execute the transactions instead of relying on completely untrusted external collaborators brings us closer to conventional security model, because we can rely on a (super)majority of the nodes being correct.

Key insight: the *executor* role in stateless BFT can be merged with the *proposer* in leaderless BFT. In stateless BFT, each executor concurrently executes disjoint transactions. While in leaderless BFT, each proposer concurrently proposes disjoint transactions. In stateless BFT, the execution outcome of the executors need to be remotely attested by a majority of the nodes. While in leaderless BFT, the proposals also need to be endorsed by a majority of the nodes. Most importantly, the execution of the storage nodes form a partially-ordered causal relation based on the based blocks of each execution. While in DAG-based leaderless BFT, the blocks also form a causal relation based on the links between rounds, and only the blocks with sufficient causal relations can be committed. This correspondence inspire us to pair a stateless BFT with a DAG-based leaderless BFT as its atomic broadcast solution, colocate the storage nodes with the consensus nodes, extend the semantics of the broadcast primitives of leaderless BFT to further disseminate and attest the execution outcomes, and re-propose the causal links to respect the causal relation of the execution of the blocks.

Approach. Nodes speculatively execute blocks before proposing and propose blocks along with their execution outcomes. The execution must *happen after* a supermajority of the accepted blocks proposed by disjoint nodes from last round. Remote nodes can attest for the execution and install the write-sets without replicated execution. The proposal is only accepted if the execution is correct and does not conflict with other accepted proposals. The commit rule is the same as the original leaderless BFT. GD: Need to further revise the design details very carefully.

## REFERENCES

[1] M. Camaioni, R. Guerraoui, M. Monti, P.-L. Roman, M. Vidigueira, and G. Voron. Chop chop: Byzantine atomic broadcast to the network limit. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, pages 269–287, 2024.

[2] G. Danezis, L. Kokoris-Kogias, A. Sonnino, and A. Spiegelman. Narwhal and tusk: a dag-based mempool and efficient bft consensus. In *Proceedings of the Seventeenth European Conference on Computer Systems*, pages 34–50, 2022.

[3] N. Giridharan, F. Suri-Payer, I. Abraham, L. Alvisi, and N. Crooks. Autobahn: Seamless high speed bft. In *Proceedings of the ACM SIGOPS 30th Symposium on Operating Systems Principles*, pages 1–23, 2024.

[4] 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 (SP)*, pages 583–598. IEEE, 2018.

[5] L. LAMPORT, R. SHOSTAK, and M. PEASE. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, 1982.

[6] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena. A secure sharding protocol for open blockchains. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 17–30, 2016.

[7] R. Neiheiser, M. Matos, and L. Rodrigues. Kauri: Scalable bft consensus with pipelined tree-based dissemination and aggregation. In *Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles*, pages 35–48, 2021.

[8] A. Spiegelman, N. Giridharan, A. Sonnino, and L. Kokoris-Kogias. Bullshark: The partially synchronous version. *arXiv preprint arXiv:2209.05633*, 2022.

[9] C. Xu, C. Zhang, J. Xu, and J. Pei. Slimchain: Scaling blockchain transactions through off-chain storage and parallel processing. *Proceedings of the VLDB Endowment*, 14(11):2314–2326, 2021.

[10] M. Zamani, M. Movahedi, and M. Raykova. Rapidchain: Scaling blockchain via full sharding. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, pages 931–948, 2018.

[11] Y. Zhang, S. Setty, Q. Chen, L. Zhou, and L. Alvisi. Byzantine ordered consensus without byzantine oligarchy. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, pages 633–649, 2020.