Poster: Sharding SMR with Optimal-size Shards for Highly Scalable Blockchains

Jianting Zhang
Purdue University
zhan4674@purdue.edu

Zhongtang Luo
Purdue University
luo401@purdue.edu

Raghavendra Ramesh
Supra Research
r.ramesh@supraoracles.com

Aniket Kate
Purdue University / Supra Research
aniket@purdue.edu

***Abstract*—Blockchain relies on State Machine Replication (SMR) to enable trustless nodes to uphold a consistent ledger while tolerating Byzantine faults. With the rapid growth of decentralized web3 platforms and applications, a central challenge of blockchain systems is scalability, which can be evaluated with two metrics: high performance and large network. However, existing blockchain systems struggle to simultaneously achieve both scalability metrics while requiring to guarantee the underlying security properties of SMR—safety and liveness. In this poster, we present a novel blockchain architecture addressing this dilemma by sharding the SMR. Our architecture builds upon two core insights: ordering-processing sharding scheme and safety-liveness separation. Specifically, the ordering-processing sharding scheme securely accommodates a large number of nodes by dividing them into multiple shards, enhancing the network scale. Additionally, the safety-liveness separation allows each shard to consider the security properties of SMR against Byzantine failures separately, by which the system can create more optimal-size shards to process transactions in parallel, enhancing performance. The preliminary experiments show the efficacy of our architecture in scaling blockchains.**

I. Motivations and Contributions

Blockchain systems rely on a Byzantine Fault Tolerant (BFT) State Machine Replication (SMR) process to ensure *safety* where any two honest nodes store the same prefix of a ledger, and *liveness* where all transactions are eventually handled. While the SMR problem can be solved using Proof-of-X (PoX), such as Proof-of-Work and Proof-of-Stake, or an atomic broadcast (ABC) protocol, they only achieve one of the two scalability metrics, i.e., they either scale the network or enhance performance. Specifically, the PoX protocol adopted by early blockchain systems can accommodate thousands of nodes but suffers from poor performance. For instance, the mainnet of Ethereum has 900,000+ validators but can only process 12 transactions per second and require minute-level confirmation latency as of January 2024 [1]. On the other hand, many subsequent blockchain systems, such as Diem [5] and Sui [2] adopt the ABC protocol to solve the SMR problems, achieving thousands of transaction throughput and second-level confirmation latency. However, running the ABC protocol with even hundreds of nodes will drop the throughput exponentially due to the high communication overhead required by the ABC protocol. Therefore, existing deployed blockchain systems suffer from the scalability problem where performance cannot scale well with more nodes joining.

Recently, sharding technology has been proposed to solve the scalability problem for blockchains by combining the PoX and ABC protocols [6]. A blockchain sharding system initiates a PoX protocol to create identities for nodes and divides nodes into multiple committees (a.k.a. shards) based on their identities randomly. After that, shards can run the ABC protocol to handle transactions in parallel. Intuitively, with more nodes joining, a blockchain sharding system can create more shards to handle transactions, thus enhancing scalability. Unfortunately, existing blockchain sharding systems face a size-security dilemma. Specifically, a small shard size enhances efficiency but increases the probability of forming insecure shards whereas a large shard size is less efficient but more secure. Since security is a more important property for a system, sharding systems have to set every shard large enough to ensure security, leading to only a few shards being created to handle transactions and each of them suffers from high communication overhead. Therefore, sharding only brings limited improvement to the scalability of blockchains and is far away from practicality.

This poster targets designing a novel architecture to securely enhance both performance and network scale for blockchains. The main idea of our architecture is to shard an SMR based on its three tasks (i.e., data dissemination, ordering, and execution) while achieving optimal shard sizes. The optimal shard sizes enable a blockchain system to create more shards to perform SMR tasks in parallel, making it highly effective and scalable. We first summarize the contributions of our work here and then elaborate on our architecture in Section II.

- We propose a novel blockchain architecture that can: 1) scale the blockchain network to support thousands of nodes to participate in the SMR process, and 2) achieve competitive performance compared to the state-of-the-art high-performance blockchain systems.
- Our architecture frees most nodes of the system from running consensus protocol, allowing them to asynchronously process transactions without any timing assumptions for avoiding FLP impossibility [4]. This makes our architecture more robust to the realistic network.
- Our architecture is designed to be agnostic to the BFT consensus protocol. Since the communications between nodes do not rely on any timing assumptions, any BFT consensus protocol is compatible with our architecture,

making our solution highly flexible.

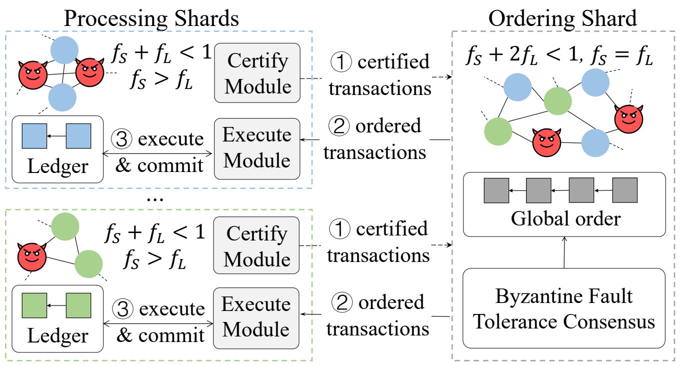


Fig. 1. Overview of our architecture. The transaction handling workflow in our architecture involves three key tasks within the SMR: ① Data dissemination: each processing shard disperses transactions and forwards the certified transactions to the ordering shard; ② Ordering: the ordering shard runs a BFT consensus to order the received transactions; ③ Execution: processing shards execute and commit the ordered transactions received from the ordering shard.

II. Architecture Description

Fig. 1 shows the overview of our architecture, which consists of two building blocks: ordering-processing sharding scheme and safety-liveness separation.

(1) Ordering-processing sharding scheme. There are two types of shards: one ordering shard and multiple processing shards. An ordering shard performs the ordering task of SMR. It runs a BFT consensus protocol to establish a *global* transaction order for the system. Processing shards perform the data dissemination/availability and execution tasks of SMR. They work as the ledger maintainer and transaction executor. At a high level, our sharding architecture *shards ledger maintenance and transaction execution but not consensus*. By decoupling transaction processing from ordering, our architecture enables processing shards to be free from consensus. Without consensus, each processing shard can tolerate up to $f < 1/2$ ratio of Byzantine nodes, allowing a smaller shard size. Note that the ordering shard running the consensus protocol is designed to always guarantee both safety and liveness and thus still requires a security threshold $f < 1/3$.

(2) Safety-liveness separation. Our architecture considers safety and liveness against Byzantine nodes separately in processing shards. Similar to a state-of-the-art sharding protocol GearBox [3], we guarantee the safety property of processing shards all the time, while allowing a processing shard to violate liveness for a while but can be recovered eventually by the sharding reconfiguration mechanism. Specifically, we appropriately increase the safety threshold f_S and decrease the liveness threshold f_L (thus $f_S > f_L$). When forming processing shards, we use a larger fault-tolerance threshold (i.e., f_S) to obtain smaller shard sizes. Note that since processing shards do not run a consensus protocol, the threshold parameters satisfy $f_S + f_L < 1$, thus achieving $f_S \geq 1/2$. This is a significant improvement for achieving a small shard size while guaranteeing the liveness of processing shards in a high probability (e.g., 99.99%). In contrast, GearBox asks each shard to independently run a consensus protocol, requiring $f_S + 2f_L < 1$ in each shard. Therefore, when setting the same f_S as our architecture, GearBox have to set smaller f_L than ours, which leads to more frequent liveness violations of shards and shards recovery, thus compromising performance.

III. Preliminary Evaluation

In our preliminary evaluation, we evaluate the scalability of our architecture compared to GearBox. When calculating the shard sizes, we want to make sure that the shards always satisfy the safety but only guarantee liveness with 99.99% probability. TABLE I presents the results of this experiment (where m represents the shard size and k represents the number of shards), showing that our architecture can create more shards with smaller sizes and thus achieves better scalability.

TABLE I
Scalability comparison (where total Byzantine ratio $s = 15\%$, security parameter $\lambda = 20$), shown in [GearBox, Ours]

	The total number of nodes n					
	50	100	200	300	400	500
m	[20, 13]	[38, 18]	[49, 20]	[57, 22]	[60, 24]	[63, 24]
k	[2, 3]	[2, 5]	[4, 10]	[5, 13]	[6, 16]	[7, 20]

IV. Conclusion and Future Directions

This poster presents a novel sharding scheme for achieving highly scalable blockchains. Some interesting directions could be further explored based on our sharding scheme. First, this approach allows us to shard any unsharded blockchain infrastructure without weakening the system resilience to Byzantine failure or making network synchrony assumptions. Second, utilizing the global order to efficiently handle cross-shard transactions. Since the ordering shard establishes a global order for all transactions from all shards, shards can consistently handle transactions without locking any states. Last but not least, integrating the smart contract execution engine into blockchain sharding systems to support more complex logic business. Since transaction execution and ordering are performed separately and asynchronously, our architecture may facilitate such an integration.

References

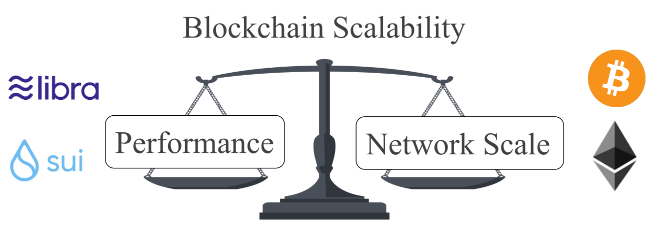
[1] beaconcha.in. History of daily active validators in ethereum. https://beaconcha.in/charts/validators. Accessed: 2024.
[2] Same Blackshear, Andrey Chursin, George Danezis, Anastasios Kichidis, Lefteris Kokoris-Kogias, Xun Li, Mark Logan, Ashok Menon, Todd Nowacki, Alberto Sonnino, et al. Sui lutris: A blockchain combining broadcast and consensus. *arXiv preprint arXiv:2310.18042*, 2023.
[3] Bernardo David, Bernardo Magri, Christian Matt, Jesper Buus Nielsen, and Daniel Tschudi. Gearbox: Optimal-size shard committees by leveraging the safety-liveness dichotomy. In *ACM CCS*, pages 683–696, 2022.
[4] Michael J Fischer, Nancy A Lynch, and Michael S Paterson. Impossibility of distributed consensus with one faulty process. *JACM*, pages 374–382, 1985.
[5] The DiemBFT Team. State machine replication in the diem blockchain. https://developers.diem.com/docs/technical-papers/state-machine-replication-paper/. Accessed: 2024.
[6] Jianting Zhang, Wuhui Chen, Sifu Luo, Tiantian Gong, Zicong Hong, and Aniket Kate. Front-running attack in sharded blockchains and fair cross-shard consensus. In *NDSS*, 2024.

Sharding SMR with Optimal-size Shards for Highly Scalable Blockchains

Jianting Zhang[1], Zhongtang Luo[1], Raghavendra Ramesh[2], Aniket Kate[1,2]

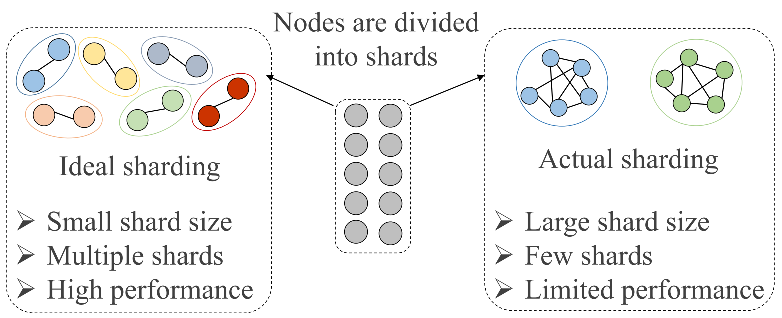
[1]Purdue University, [2]Supra Research

Introduction



- Blockchain scalability is evaluated by performance and network scale.
- With more nodes joining in, a scalable blockchain system is expected to handle more transactions.

Blockchain Sharding



- Sharding scales a blockchain by dividing nodes into shards for parallel execution.
- Efficiency-security dilemma: large shards are required to guarantee security.

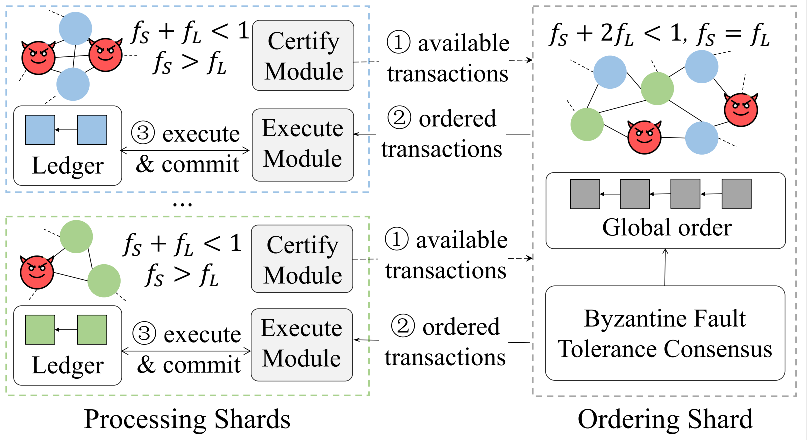
Key Observations

Obs1: Blockchains rely on state machine replication to maintain a ledger securely, performing the repeated tasks:
① Dissemination (data availability); ② Ordering; ③ Execution

- Tasks ①③: resource-intensive but 1/2 fault tolerance
- Tasks ②: resource-saving but only 1/3 fault tolerance

Obs2: The larger the fault tolerance a shard achieves, the smaller the size of the shard is needed.

Our Solution



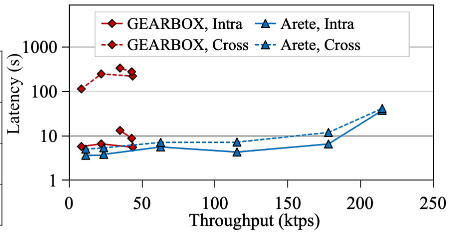
Idea: deconstructing SMR to create more lightweight shards.

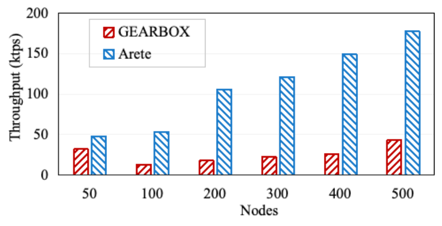
- Ordering-processing sharding scheme: one ordering shard performs the ordering task and multiple processing shards perform the dissemination and execution tasks.
- Safety-liveness separation: trade liveness threshold f_L for larger safety threshold f_S, create much smaller shards.

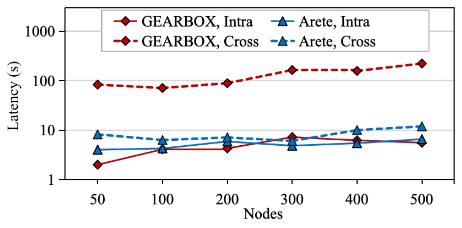
Evaluations

m-shard size; k-shard number;

	The total number of nodes n [SOTA, Ours]					
	50	100	200	300	400	500
m	20, 13	38, 18	49, 20	57, 22	60, 24	63, 24
k	2, 3	2, 5	4, 10	5, 13	6, 16	7, 20







Future Directions

- Shard any non-sharded blockchains without weakening the system resilience to Byzantine failure or making synchrony network assumptions.
- Efficient cross-shard coordination: with a global order, shards could handle cross-shard transactions consistently without locking states.
- Programmability for sharding: Since transaction execution and ordering are performed asynchronously, this may facilitate the integration of smart contract execution engines.