Poster: Unanimous-Majority - Pushing Blockchain Sharding Throughput to its Limit

Xu, Yibin; Slaats, Tijs; Düdder, Boris

Published in:
CCS '22: Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security

DOI:
10.1145/3548606.3563506

Publication date:
2022

Document version
Peer reviewed version

Document license:
Unspecified

Citation for published version (APA):
ABSTRACT
Blockchain sharding protocols randomly distribute nodes to different shards. They limit the quantity of shards to ensure that the adversary remains a minority inside each shard with a high probability. There can exist only a small number of shards. In this article, we propose a new sharding protocol that links the number of shards with the adversary population in real-time instead of a fixed upper-bounded population. The protocol is a two-phase design. First, several committee shards are constructed where the majority of nodes inside each are honest with high probability: then, each committee shard randomly splits into several worker shards with a high likelihood that at least one honest node is inside each. Each worker shard handles different transactions. Worker shard blocks that did not pass the unanimous voting are collected and voted by the committee shard using the majority voting.

We show that (1) in the worst case (extremely unlikely) when all the transactions need to be handled by the committee shards, the transaction throughput and the data requirement only deteriorate to the same level as classical sharded blockchain; (2) when the worker shards handle most transactions, the overall transaction throughput is zoomed by two magnitudes securely while the data requirement for nodes remains at the same level.

CCS CONCEPTS
- Networks → Network architectures; Security and privacy:

KEYWORDS
Blockchain Sharding, Performance improvements

ACM Reference Format:

1 INTRODUCTION
Blockchain sharding [1–6] is a type of blockchain performance optimization methodology that increases the number of transactions per second while reducing the workload on each node. Technically, it creates many blockchains (shards) that run in parallel and randomly assign nodes to them, with cross-shard transactions connecting shards. Each shard employs voting instead of the longest-chain mechanism to obtain consensus. Researchers in this domain are particularly interested in how cross-shard transactions, bootstrap designs, and intra-shard consensus procedures are carried out. However, an upper-bound adversary population is assumed because of the inability to identify honest nodes from adversaries. It limits the shard number and performance overall.

Blockchain consensus is validated consensus, and nodes reach consensus on the sequence of the transactions but not the acceptability. The transaction acceptability (e.g., violations like overspending and double-spending) is determined solely by checking the previous blockchain records but not by the consensus decision. A majority honesty of nodes is required only for determining the mainchain of the blockchain, avoiding forking as well as the cases of reversing consensus. Thus, for a sharded blockchain, we may use a veto system for only a few nodes to reject faulty blocks and a majority vote system involving more nodes for conflict resolution.

In this article, we deploy a two-phase-voting design. In the first phase, the nodes in a tiny group (we refer to it as a worker shard) that deploy the unanimous voting generate and vote for a worker block. The worker block contains transactions associated with the accounts under the government of the worker shard. When at least one honest node is in the worker shard, a worker block containing unacceptable transactions will not be appended to the blockchain of that worker shard. To avoid faulty or adversary nodes disrupting a unanimous verdict, a shard of greater size (committee shard) is summoned in the second phase to reach a majority verdict on the worker block that did not pass unanimous voting in the first phase. The blockchain in the worker shard is only then synced by the members in the committee shard for verification.

The classical sharded blockchain can be seen as the above design in the scenario where, in each blockchain epoch, all the worker shards fail to generate unanimously approved worker blocks within the first phase. Thus, the lower-bound performance of this design is allied to the performance of the classical approaches. Furthermore, the performance of our design is dynamically adjusted with the number of adversary nodes in real-time, which intend to disrupt the protocol by voting for faulty blocks.

1.1 What are the cost and challenges?
Though the two-phase design can increase the transaction per second, it must overcome some challenges.

(1) Constraints on the number of shards. Once a worker block fails to obtain the unanimous approvals within the first phase, the block is synced and verified by the corresponding committee shard. Thus,
We have the following designs:

1. Among the nodes, properties for reaching consensus inside a worker/committee shard:
   - Adversary nodes will not vote for correct blocks.
   - We provide three nodes with an upper-bounded communication delay among them.

2. Close-membership synchronous communication protocol, has
   - Contains a number of worker shards.
   - A committee shard governs

Our protocol is a sharded replicated service of two layers, the first

2.1 How is the performance improved?

The theoretical performance of the classical approach: Assume there are $s$ shards, each sized $m$. The shard membership does not overlap, $N = s \times m$. The failure probability of a shard describes the probability that the adversary with $t$ nodes overall has more than half of the nodes in a shard [6].

$$Pr_{ghs} = \sum_{X \in \{m/2\} \times (m/2)} \binom{m}{X} \left( \frac{t}{X} \right) \left( \frac{N-X}{N-t} \right)$$ (1)
When considering the system as a whole, a loose upper bound for any shard to be compromised is

\[ Pr_1 \leq \min(1, Pr_{h_g} \times s) \]  

(2)

\( Pr_1 \) with some parameters is displayed in Table 1. If the security threshold is set to be \( Pr < 10^{-6} \), the blockchain can only split into at most ten shards when it has 2000 nodes. If each block contains TXN transactions, it can process \( Overall_{TXN} = TXN \times s \) transactions at maximum in each blockchain epoch.

### Table 1: \( Pr, Pr_{h_g}, N = 2000, t = \lfloor (N - 1)/3 \rfloor = 666 \)

<table>
<thead>
<tr>
<th>( m )</th>
<th>( s )</th>
<th>( Pr_{h_g} )</th>
<th>( Pr_1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>125</td>
<td>16</td>
<td>3.35E-05</td>
<td>0.000556</td>
</tr>
<tr>
<td>200</td>
<td>10</td>
<td>8.66E-08</td>
<td>8.66E-07</td>
</tr>
<tr>
<td>250</td>
<td>8</td>
<td>1.67E-09</td>
<td>1.36E-14</td>
</tr>
<tr>
<td>400</td>
<td>5</td>
<td>2.72E-15</td>
<td>1.36E-14</td>
</tr>
<tr>
<td>500</td>
<td>4</td>
<td>7.80E-20</td>
<td>3.12E-19</td>
</tr>
</tbody>
</table>

The theoretical performance of our approach: When a two-phase is deployed where the nodes are divided into cs committee shards, with ws worker shards inside each, the probability for an unacceptable worker block to be accepted at a particular worker shard of a particular committee shard is

\[ Pr_{h_g} = \sum_{i=0}^{m_1} \left( \frac{N}{m_1} \right) \left( \frac{m_1-i}{m_2} \right) \left( \frac{i}{m_2} \right) \]  

(3)

where \( m_1 = \lfloor N/cs \rfloor \) and \( m_2 = \lfloor m_1/ws \rfloor \). When considering the system as a whole, the probability for any worker shard of any committee shard to be compromised is

\[ Pr_2 \leq \min(1, Pr_{h_g} \times ws \times cs) \]  

(4)

When a worker block is not being accepted, it should be verified and voted on by the whole committee shard, which should maintain a majority honesty. Thereby, \( Pr_1 \) (for constructing committee shards) and \( Pr_2 \) should simultaneously be smaller than the threshold. Table 2 shows some result for \( Pr_2 \).

### Table 2: \( Pr_2, Pr_{h_g}, \) when \( N = 2000, t = \lfloor (N - 1)/3 \rfloor, Pr_{h_g} \leq 10^{-6} \)

<table>
<thead>
<tr>
<th>( m_1 )</th>
<th>( cs )</th>
<th>( m_2 )</th>
<th>( ws )</th>
<th>( Pr_{h_g} )</th>
<th>( Pr_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>10</td>
<td>13</td>
<td>15</td>
<td>1.12E-08</td>
<td>7.49E-06</td>
</tr>
<tr>
<td>250</td>
<td>8</td>
<td>13</td>
<td>20</td>
<td>2.15E-10</td>
<td>1.44E-07</td>
</tr>
<tr>
<td>400</td>
<td>5</td>
<td>11</td>
<td>36</td>
<td>3.46E-16</td>
<td>2.31E-13</td>
</tr>
<tr>
<td>500</td>
<td>4</td>
<td>11</td>
<td>45</td>
<td>6.98E-21</td>
<td>6.59E-18</td>
</tr>
<tr>
<td>1000</td>
<td>2</td>
<td>9</td>
<td>111</td>
<td>1E-59</td>
<td>3.48E-57</td>
</tr>
</tbody>
</table>

Our approach’s maximum performance per blockchain epoch is \( Overall_{TXN} = ws \times cs \times TXN \). Thus, \( ws \) times more than the classical ones in theory when \( cs = s \).

**Data requirement** Assuming the approach has the same \( Overall_{TXN} \). In the best case, the data complexity for an individual node per blockchain epoch is

\[ O(\text{Block}_{\text{worker}} + \text{Block}_{\text{committee}} + 2 \times m_1 \times \text{Vote} \times \frac{Overall_{TXN}}{cs \times \text{trans}}) \]  

(5)

The votes consist of the votes for all the worker shards within the same committee shard (\( m_1 \) votes in total), and the votes for the committee block (\( m_1 \) votes in total).

In the worst case, the data complexity for an individual node per blockchain epoch is approximately

\[ O(ws \times \text{Block}_{\text{worker}} + \text{Block}_{\text{committee}} + 2 \times m_1 \times \text{Vote} \times \frac{Overall_{TXN}}{cs \times \text{trans}}) \]  

(6)

The data complexity for the classical approach is

\[ O(\text{Block} + m \times \text{Vote} + \frac{Overall_{TXN} \times \text{trans}}{2000}) \]  

(7)

Depending on the real-time situation, our approach may or may not outperform the classical approach in data requirements.

### 3 PERFORMANCE SIMULATION

We randomly assign nodes with different adversary populations into 222 worker shards. Each has at least 9 nodes inside. \( N = 2000 \). The shard membership does not overlap. The simulation was repeated 1000 times. The blue bars in Figure 2 show the average number of worker shards with no adversary nodes inside. These shards will always generate a unanimously approved block within the first time-bound. The corresponding committee shards will handle the worker blocks from the remaining worker shards. Our approach outperforms the classical approach when the real-time adversary is lower. There is a trade-off between \( m_1 \) (committee shard size), the data burden with different real-time adversary populations (how many malfunctioning worker shards a committee shard needs to process), and the frequency of cross-committee-shard-transactions. These are omitted due to page limits.

**Figure 2: Performance comparison.** \( N = 2000 \). There are 222 worker shards. The blue bars show the number of worker shards that can generate the unanimously approved worker blocks (no adversary nodes inside the shard) with different adversary percentages in overall VS the max shard num in the classical design (orange bar), which is only ten in constant.

### 4 CONCLUSION

This paper proposes a new blockchain sharding approach deploying a unanimous-then-majority design. The performance is automatically adjusted with the adversary nodes in real-time instead of an upper-bounded population. It increases the performance of the sharding blockchain significantly.

### REFERENCES