Evaluating the Performance Limits of Sharded Blockchain Architectures through Simulations

1st Om Amit Gandhi
College of Computing
Illinois Institute of Technology
Chicago, United States of America
ogandhi1@hawk.illinoistech.edu

2nd Ioan Raicu
College of Computing
Illinois Institute of Technology
Chicago, United States of America
iraicu@illinoistech.edu

***Abstract*—The most widely used blockchains by market share, such as Bitcoin and Ethereum, suffer from low transaction throughput, which limits their effectiveness as digital currencies and drives up transaction costs. Sharding has emerged as a promising technique to parallelize block creation and increase theoretical throughput. This work evaluates several sharding strategies and examines how factors such as the number of shards, block time, block size, and communication overhead influence system-wide transaction throughput. To validate our simulator, we model well-known blockchains both without sharding (Bitcoin, Bitcoin Cash, Litecoin, DOGE) and with sharding (NEAR), replicating their performance under different configurations. We then introduce a new blockchain design, MEMO, which leverages sharding to significantly accelerate transaction processing. Using our simulation framework, we identify parameter settings that achieve peak throughput (50723.1754 transactions per second), minimal transaction finality (0.23 second block time), and hybrid configurations that provide strong throughput while maintaining short finality times (42853.7178 transactions per second with 1.42 second block time). These results represent an 14-fold improvement in throughput and a 10-fold reduction in finality compared to NEAR and offer performance many orders of magnitude higher than non-sharded blockchain systems.**

***Index Terms*—Blockchain Sharding, Winner-based Sharding, Blockchain Simulation**

I. Introduction

Public blockchains like Bitcoin, Bitcoin Cash, Litecoin and DogeCoin were designed with conservative parameters that prioritize security and decentralization over raw throughput. In these systems, every node verifies every transaction, and performance is dominated by how often blocks are produced and how many transactions each block can safely carry. Simply increasing block size or reducing block time appears to offer easy path to higher throughput, but this changes the operational regime of the network. Blocks become heavier to verify and propagate, and the system has to tolerate more frequent, larger updates to global state. In parallel, a different line of work explores sharded architectures, where responsibility for transaction processing is split across multiple subsets of validators that cooperate to maintain one logical chain.

In this work, we build a SimPy-based simulation tool that lets us explore both directions within a single framework. First, we configure the simulator to reproduce non-sharded, Bitcoin-style settings, and sweep block sizes and block times to see how far a monolithic chain can be pushed in terms of Transactions Per Second (TPS). We then enable sharding and use the same tool to model our MEMO configuration, where multiple shards produce work in parallel and their outputs are merged into a single logical chain. By varying the number of shards and the total block capacity, we measure how MEMO's throughput scales with shard count, and we compare these results against the best non-sharded configurations to quantify how much additional TPS can be gained by increasing the number of shards rather than only tuning block size and block time.

II. System Design

We build a discrete-event "winner-based sharding" simulator in SimPy that can model both monolithic and sharded proof-of-work-style blockchains under a unified network. The core entities are nodes, miners, and wallets.

A. Simulator Architecture

The nodes form an undirected overlay graph. Each node maintains a neighbor list and relays blocks via a simple flooding protocol. Miners are attached to this network and draw exponential samples based on a configurable global hash-rate and difficulty to decide when each shard finds a block. Wallets generate transactions over time at a fixed interval and add them to a shared transaction pool. Each block is represented by a `Block` object with a header and a per-transaction payload (`size = HEADER_SIZE + tx * 256`), so we can tie network and storage costs directly to the number of confirmed transactions. At the end of a run, the simulator reports aggregate metrics such as number of blocks, total confirmed transactions, average block time, effective TPS, and total coordination messages in a CSV-friendly format.

B. Round Timing and Coordination Modes

The timing model for each round is built from four components: mining, verification, coordination, and network throughput. Given a target baseline block time and a shard count `S` the simulator derives an effective per-shard difficulty using a harmonic-number correction so that the maximum of `S` exponential shard times has the desired expectation. For each round, `S` shard times are sampled, and the round's mining time is the maximum of these samples. The coordinator

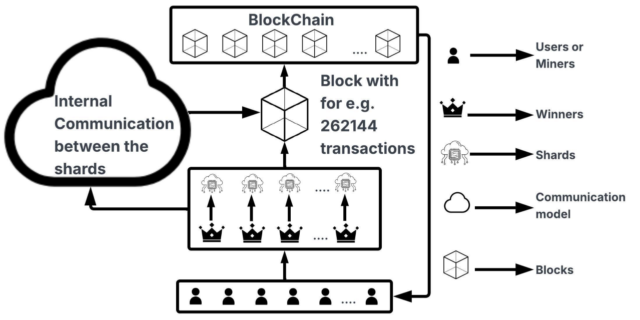
BlockChain
....
Users or Miners
Block with for e.g. 262144 transactions
Winners
Internal Communication between the shards
Shards
Communication model
Blocks

Fig. 1. Winner-based sharding system architecture.

then pulls up to `total_blocksize` transactions from the global pool (if available), splits them evenly across the shards, and computes a verification cost based on `tx_cost_ms` per transaction. The slowest shard's verification time gates this phase. Block rewards and halving are modeled explicitly: every block mints the current reward, and the reward is halved after a configurable number of blocks, though for most of out MEMO and NEAR experiments we turn halving off to focus on throughput rather than long-term supply.

Coordination between winners is modeled at the message and bandwidth level, with three distinct modes. In metronome mode, each round has `S` winners that coordinate via a logical metronome. The function `metronome_messages(S,N)` counts winner $\leftrightarrow$ metronome, winner $\leftrightarrow$ winner, and network broadcast messages, and the simulator converts these into time using `msg_size`, `control_bw_mbps`, and a per-message processing cost `msg_proc_ms`. In no-metronome mode, winners announce themselves to the entire network and then run pairwise coordination. We explicitly model a sequential announcement phase and a combinatorial $S(S-1)/2$ pairwise phase, each gated by control-plane bandwidth and CPU cost. In leader-metronome mode, inspired by NEAR-style protocols, the first shard to finish becomes the leader, announces to all nodes, receives compact metadata from the other winners, and eithe verifies all transactions itself or relies on per-shard verification plus attestations. In all modes, block header/body broadcast is modeled with a separate `brodcast_bw_mbps` knob, and we optionally overlap the block body with the next round via `overlap_broadcast`, so the round's effective time is the maximum of mining, verification, coordination, and broadcast-blocking terms.

C. Network Topology and Cost Assumptions

Our default network and cost assumptions are chosen to approximate a well-provisioned US (or global) validator cluster rather than a consumer-grade home node. Unless otherwise stated, we use $1,000$ nodes with 500 random neighbors each, 1,000 miners sharing a total hashrate of 10^6 abstract units, and $1,000$ wallets that each generate 1,000 transactions at an interval of 0.01 s. This corresponds to a heavy offered load, and we pre-fill the mempool (`prefill=true`) in all experiments so that every configuration is evaluated under sustained stress rather than in a warm-up regime. Control-plane messages are assumed to be 300 bytes on average and traverse a 200 Mbps "control link," while blocks are broadcast over a 2 Gbps "data link." We assume an 80 ms round-trip time with one coordination round per block (`coord_rounds = 1`), and we assign a 0.2 ms verification cost per transaction (`tx_cost_ms = 0.2`) and a 0.05 ms CPU cost per control message (`msg_proc_ms = 0.05`).

To model per-node bandwidth constraints, we use a simple NIC-token abstraction: each node is given separate token buckets corresponding to its 200 Mbps control NIC and 2 Gbps data NIC, and sending a message or block consumes tokens in proportion to the number of bytes transmitted. When a bucket is empty, further traffic on that link is delayed until tokens are replenished at the configured rate. This ensures that our simulator never injects more traffic than a realistic validator NIC could sustain, and it lets us attribute throughput limits and scalability breaks to concrete compute- or network-side bottlenecks rather than to an unrealistically "infinite" network.

1) No-Metronome Communication: In the no-metronome sharded configuration, we partition validators into S shards but allow each shard to emit blocks as soon as a PoW winner is found, without any global timing signal. Each shard therefore behaves like an independent Bitcoin-style chain running on the same underlying network and workload. As we increase the shard count, the aggregate TPS rises because multiple shards mine in parallel, but block times remain stochastic and can vary significantly across shards. Our "no metronome" graphs plot TPS, average block time, and message volume versus S under fixed block-size settings to show both the raw throughput gains and the cost in terms of higher message load and more variable finality.

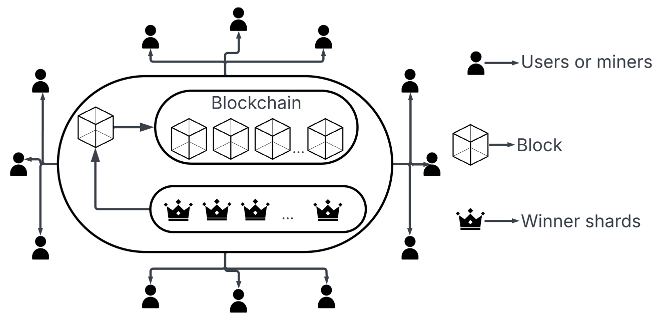

Fig. 2. No Metronome communication protocol, purely just broadcast

2) Leader Metronome Communication: In the leader-based metronome configuration, MEMO augments sharding with a global timing signal driven by the first shard winner in each round. At the beginning of a round, all shards mine in parallel; the earliest winner immediately announces itself as the round leader by broadcasting a short control message to the entire network, and the remaining shard winners send their shard summaries only to this elected leader. The leader then aggregates the shard-level results into a single logical block and triggers a global commit at the configured period, which smooths block times while still exploiting parallelism across shards. Using the same network and workload as above, we sweep over shard counts and block sizes and report TPS, average block time, and message volume. In the results, we highlight three representative block-size settings: (i) a throughput-optimized configuration where MEMO achieves its highest TPS at large shard counts, (ii) a latency-optimized configuration with sub-second average block times and still-high TPS, and (iii) an intermediate "balanced" configuration that trades a modest loss in peak throughput for substantially better finality. Across these regimes, the leader-based metronome consistently delivers higher or comparable throughput to the no-metronome sharded baseline while substantially reducing and stabilizing block time.

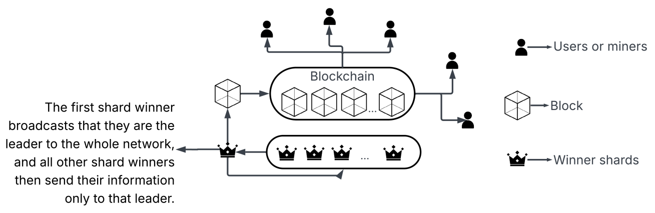
Blockchain
Users or miners
Block
Winner shards
The first shard winner broadcasts that they are the leader to the whole network, and all other shard winners then send their information only to that leader.

Fig. 3. Leader Metronome communication protocol, only the first winner broadcasts and the rest communicates with that one

D. Model Validation

Before using our simulator to explore MEMO-style sharding, we first validate that it reproduces realistic behavior for well-known blockchain designs. In particular, we calibrate it against two classes of systems: (i) monolithic, non-sharded proof-of-work chains modeled after Bitcoin, Bitcoin Cash, Litecoin, and Dogecoin, and (ii) a NEAR-style sharded configuration with leader-based coordination. In all cases, we reuse the same network and cost parameters described in the system design section, so that any differences in throughput or latency stem from the consensus and sharding mechanics rather than from changes in hardware or workload.

1) Non-Sharded Validation: First, we validate the simulator in a monolithic setting by matching the behavior of four widely-deployed UTXO chains: Bitcoin (BTC), Bitcoin Cash (BCH), Litecoin (LTC), and Dogecoin (DOGE). We configure each run with the published target block interval and block size: BTC and BCH use 10-minute blocks, with BTC at 1 MB and BCH at 32 MB; Litecoin keeps the same 1 MB limit but targets 2.5 minute blocks, and Dogecoin uses roughly 1 minute blocks with Bitcoin-style block sizes. We keep the network model, node count, miner hash-rate, and workload identical across runs so that only the configured chain parameters differ. This lets us treat these experiments as a sanity check: if we cannot produce the relative behavior of these four chains, our subsequent sharding results would be hard to trust.

The average block time graph confirms that the simulator tracks these targets closely. Bitcoin and Bitcoin Cash converge to approximately 610 seconds per block, consistent with a 600 second configuration plus stochastic variation. Litecoin's simulated average of 154 seconds is near is 150 second target, and Dogecoin's 68 second aligns with its 60 second goal once variance is accounted for. In other words, given the same hash-rate and difficulty calibration logic that we later use for MEMO and NEAR, the simulator reproduces the expected ordering BTC, BCH, LTC, and DOGE in terms of block interval. This is important because effective throughput of a monolithic proof-of-work chain is essentially, `transactions per block/block time` so any mistuned block time model would distort the TPS we report later.

We next look at messages and TPS for the same four chains. With the sharded gossip overlay and identical node count, the simulator produces roughly the same order of magnitude of messages for BTC, LTC, and DOGE, while BCH's much larger blocks slightly reduce the total number of blocks (and thus coordination rounds) required to clear the same workload. TPS follows the expected pattern from the literature: Bitcoin stays near single-digit TPS under realistic assumptions, Bitcoin Cash reaches into hundreds of TPS with 32 MB blocks, and Litecoin and Dogecoin occupy the middle ground thanks tio shorted block times and smaller block. The fact that our simulated TPS ordering and approximate magnitudes align with these known capacities gives us confidence that the mining, verification, and broadcast components of the model are all behaving reasonably in the non-sharded regime.

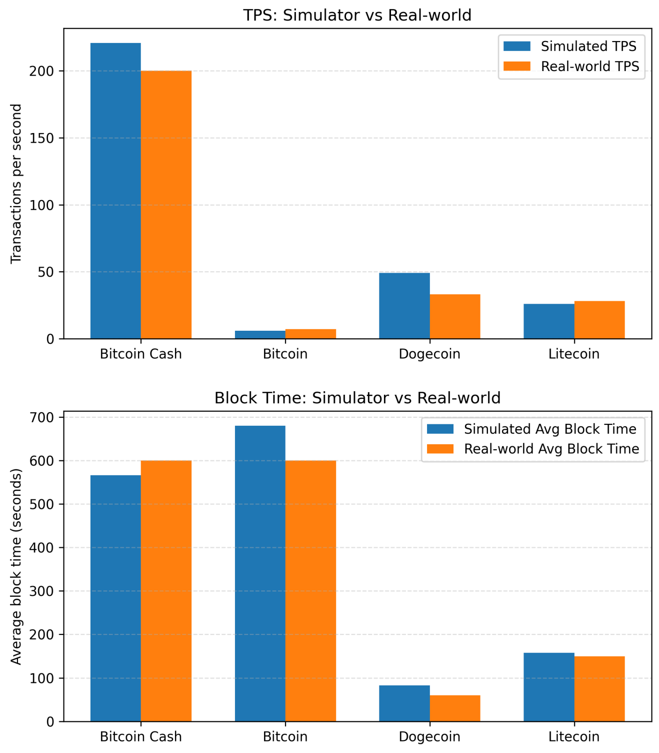

Fig. 4. Non-sharded BTC/BCH/LTC/DOGE validation, comparison of TPS and block-time with real world values

2) Sharded Validation: To validate the shared side of the simulator, we construct three NEAR-like configurations with shard counts $S \in \{4, 6, 9\}$. For each configuration we keep the same network and workload settings as in the monolithic experiments (1,000 nodes, 1,000 miners, 1,000 wallets, and 1,000 transactions), and we tune the target block time and difficulty so that the simulated average block time converges to roughly 0.6 seconds. The "NEAR: average block time vs shards" graphs shows that the measured block time stays tightly clustered between 0.59 and 0.61 seconds across 4, 6, and 9 shards, which is consistent with NEAR's design goal of maintaining a short, stable block interval while scaling capacity via sharding rather than by slowing down blocks.

The NEAR messages-versus-shards plot provides a complimentary view of control-plane overhead. As we increase the number of shards from 4 to 9, the total number of control and broadcast messages needed to clear the fixed workload actually decreases, because each round can carry more transactions and the systems completes in a fewer rounds. At the same time, the NEAR TPS-versus-shards graph shows throughput rising from roughly 2600 TPS at 4 shards to over 4100 at 9 shards. This matches the qualitative behavior expected from state sharding: additional shards primarily increase throughput by allowing more work to be done in parallel, while coordination costs remain bounded under realistic bandwidth and latency assumptions.

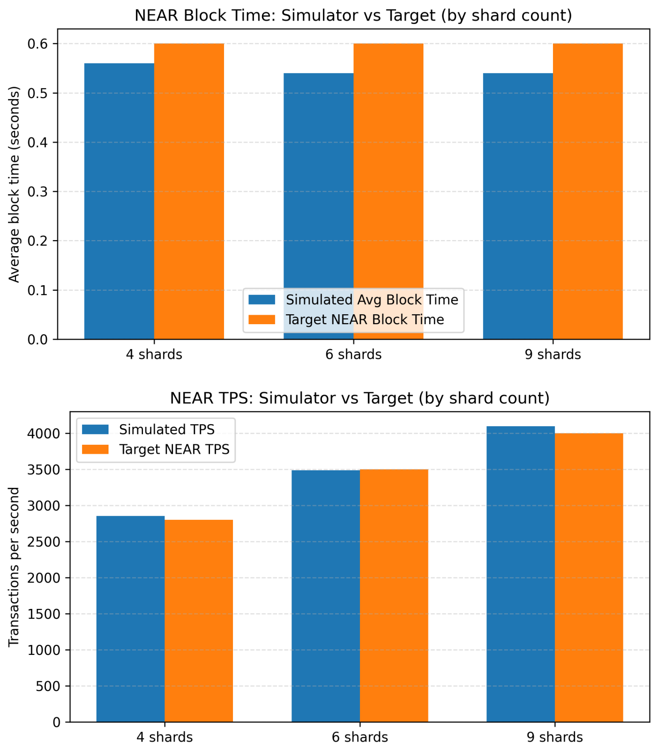

Fig. 5. NEAR-like validation, comparison of TPS and block-time with real-world values

III. Performance Evaluation

We evaluate our simulator and the proposed MEMO design along three main axes: effective throughput (transactions per seconds, TPS), average block-time and control-plane message volume. All experiments share the same underlying environment described in the system design: 1,000 nodes arranged in a random overlay, 1,000 miners sharing a global hash-rate, and 1,000 wallets generating a high offered load. For each experiment, we run the simulator until either a fixed number of blocks is produced or the prefilling workload has been fully drained, and we log aggregate metrics in a CSV-friendly format. This setup lets us compare a monolithic configuration with a sharded configuration on equal footing, while only changing protocol-level knobs such block time, block size and the number of shards.

In the previous section, we showed that the simulator reproduces the qualitative behavior of the non-sharded Bitcoin-style chains (BTC, BCH, LTC, and DOGE) and NEAR-like sharded configurations, giving us confidence that the timing and message model is reasonable. In this section we therefore focus on the performance results rather than validation: how TPS scales with shard count, how coordination overhead grows with S, and how MEMO compares against both the monolithic baselines and the NEAR-like design under the same network and cost assumptions.

A. *Scaling Limits of Non-Sharded Chains*

In the non-sharded experiments, we fix the network and workload and vary only the block-level knobs: the configured block interval and the block size. For each of BTC, BCH, LTC, and DOGE experiments, we use 1,000 nodes, 1,000 miners, 1,000 wallets, and 1,000 transactions, so that the system is driven by a high offered load so that blocks are almost always full. We then sweep the block size from a few thousand up to 262,144 transactions per block while tightening the configured block interval in line with each currency's design (for example, BTC-style runs around 600 seconds, LTC around 150 seconds, DOGE around 60–75 seconds). The TPS-versus-block-size curves show the expected behavior: starting from small blocks and long intervals, throughput rises sharply as we either increase the number of transactions per block or shorten the target block time. Across all four chains, the simulator reports low hundreds of TPS in the conservative regime and climbs up to roughly 5000 TPS in our aggressive non-sharded runs with larger blocks and shorter intervals.

B. *Sharding*

Sharding offers a way to push beyond these single-chain limits by splitting the validator set and workload across multiple parallel shards while still producing a single logical chain of finalized blocks. Instead of forcing every node to download and verify every transaction, the network is partitioned into S shards, each of which processes a disjoint subset of transactions and proposes a shard-local block for each round. A lightweight coordination layer (e.g., a metronome-style leader) then merges these shard blocks into a single "global" block

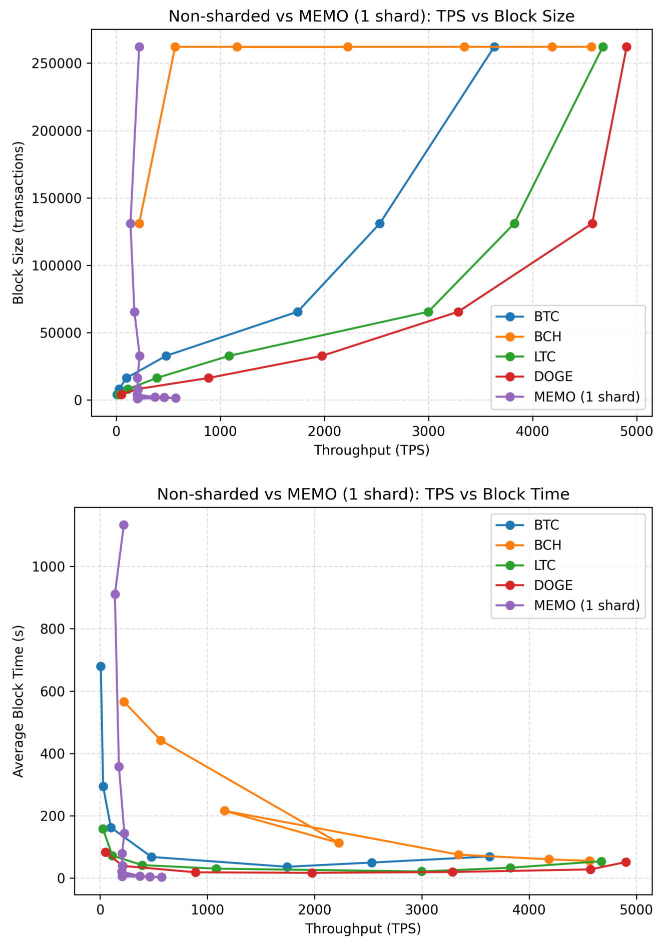

Fig. 6. Non-sharded BTC/BCH/LTC/DOGE chains compared to the MEMO single-shard configuration, illustrating how tuning block size and block time eventually hits scaling limits.

header, so that from the application's perspective there is still one canonical chain, but the underlying execution has been parallelized. This lets us increase aggregate throughput roughly with the number of shards without needing to grow individual block sizes to extreme values or drive block intervals to unrealistically small targets on a single chain. In our simulator, this means we can sustain thousands to tens of thousands of TPS at sub-second block times by adding shards, rather than by pushing a monolithic chain into regimes where network bandwidth, verification cost, and propagation delays would become prohibitive.

C. Throughput–Finality Tradeoffs in Sharded MEMO

Across the full MEMO sweep, we hold the network and workload fixed and vary only the block size and shard count, allowing us to see how far sharding can push throughput before other bottlenecks dominate. Starting from the single-shard baseline, where large blocks and long intervals yield only a few hundred TPS at multi-minute block times, increasing the shard count steadily drives both higher throughput and much shorter finality. For moderate configurations (e.g., 4–9 shards with 1.6–4K transaction blocks), MEMO already sustains on the order of 10^3–3×10^3 TPS at sub-second to a few-second average block times, comfortably beating the aggressive non-sharded runs while staying in a realistic latency regime. As we continue to scale out to 32–256 shards and larger blocks, aggregate throughput climbs into the 10^4–4×10^4 TPS range while average block time remains well below tens of seconds; at this point, the curves in our plots start to show diminishing returns as coordination overhead, NIC bandwidth, and verification cost become the dominant constraints rather than the raw shard count itself.

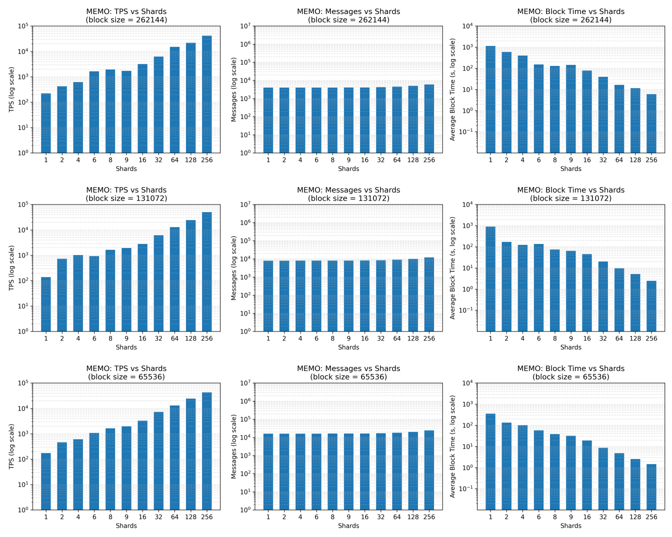

Fig. 7. Sharded scaling behavior of MEMO under a fixed high-load configuration. Increasing the shard count from 1 to 16 yields near-linear throughput gains (reaching multi–10k TPS) while preserving sub-second block times, demonstrating that MEMO can achieve high throughput without pushing any single shard into unrealistic block sizes or verification costs.

1) Best TPS for Memo: For the NEAR-like configuration, the highest throughput in our sweep occurs at a block size of 131,072 transactions with 256 shards, which pushes the system toward the extreme end of our sharding design space. At this setting, the aggregate TPS is maximized by fully exploiting parallelism across shards, while individual shards still operate with per-block workloads that are barely within the capacity of a well-provisioned validator. However, we do not claim that this 256-shard point is an ideal operational target: coordination overheads, implementation complexity, and practical deployment constraints would likely make such a configuration challenging to sustain in a real network. Instead, we treat it as an upper bound that demonstrates how far a NEAR-style sharded architecture can be pushed under optimistic assumptions about network bandwidth and validator hardware. In the rest of our evaluation, we focus on more moderate shard counts (e.g., 4–64 shards) where the trade-off between throughput, latency, and system complexity is more representative of a deployable configuration, and use this 256-shard result primarily as a reference for the maximum attainable throughput in our simulator.

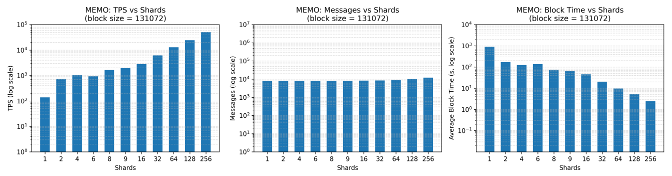

Fig. 8. Best throughput for MEMO at 256 shards for the block size of 131072.

2) Best finality for MEMO: For the NEAR-like configuration, the lowest confirmation latency in our sweep occurs at a block size of $1,600$ transactions with 64 shards. At this setting, the average block time drops to well below a second, bringing finality into a regime that is suitable for interactive applications rather than just bulk settlement. Because each shard only needs to assemble and verify a relatively small block, per-shard computation and propagation are cheap, allowing the global metronome to advance quickly without saturating the network. The trade-off is that aggregate throughput at this point is lower than in our largest-block, many-shard configurations, since we are explicitly giving up per-block payload to gain faster rounds. We therefore treat the 64-shard, $1,600$-transaction configuration as a "latency floor" reference point: it shows how far a NEAR-style design can push finality times when optimized for responsiveness rather than raw TPS.

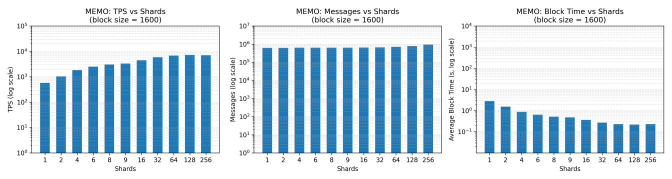

Fig. 9. Best finality for MEMO at 64 shards for the block size of 1600.

3) Sweet Spot for MEMO blockchain: In between the pure throughput and pure latency extremes, we find a useful "sweet spot" at a block size of $65,536$ transactions with 256 shards. At this configuration, the simulator sustains on the order of 4.3×10^4 TPS while keeping the average block time around 1.5 seconds, which is fast enough to feel responsive for many end-user workloads. Larger blocks or more shards can push raw throughput higher, but they begin to incur diminishing returns as coordination overhead and propagation delays grow. Smaller blocks, in contrast, reduce latency further but leave substantial capacity on the table. The 256-shard, $65,536$-transaction regime thus illustrates how a NEAR-style sharded design can simultaneously deliver high TPS and sub-two-second finality without pushing any individual block or shard into an unrealistic operating regime. In practice, this configuration leaves headroom for transient load spikes while keeping validation and networking costs within the reach of well-provisioned validators. More broadly, it highlights how sharding exposes a tunable design space where protocol designers can trade modest increases in confirmation time for large multiplicative gains in effective throughput.

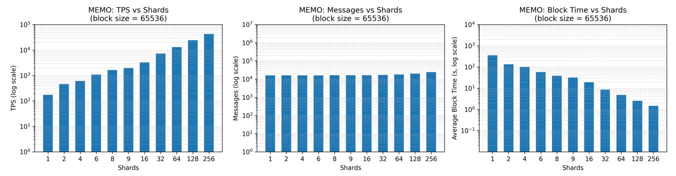

Fig. 10. The sweet spot for the MEMO configuration to have a nice throughput with a valid finality.

IV. Conclusion

This work presented a configurable blockchain simulator capable of capturing both conservative, Bitcoin-style non-sharded chains and aggressive sharded designs such as NEAR and our proposed MEMO configuration. By holding the underlying network and workload fixed and sweeping only block-level and sharding parameters, we showed how traditional single-chain systems quickly run into scaling limits: throughput initially grows with larger blocks and shorter intervals but saturates in the low-to-mid thousands of TPS once propagation delay, verification cost, and bandwidth constraints become dominant. In contrast, our sharded MEMO configurations demonstrate that we can push well beyond these limits: by increasing the shard count and keeping per-shard blocks modest, the simulator reaches peak throughputs on the order of 5×10^4 TPS (e.g., $k = 131{,}072$ at 256 shards) and still maintains multi-second finality, a regime that is unreachable for a single monolithic chain under the same network assumptions.

Sharding also gives us a direct handle on finality. Instead of relying only on ever-shorter block intervals on a single chain, we can dial up shard parallelism while keeping per-shard block times in a safe range, and then choose operating points along a throughput–latency trade-off curve. In our experiments, we see configurations with extremely low latency (e.g., ≈ 0.25 s at 64 shards and small blocks) that still sustain several thousand TPS, and "sweet-spot" configurations (e.g., 256 shards and medium block sizes) that deliver tens of thousands of TPS at around 1–2 s block times. These results highlight that sharding is not just a way to get more raw throughput; it is a mechanism to jointly optimize throughput and finality under realistic network and verification costs. Looking forward, we see this simulator as a platform for systematically exploring that design space, validating sharded protocols against real-world conditions, and helping engineers choose shard counts and block parameters that hit their desired balance between TPS, finality, and robustness.

V. Future Work

a) Real-world validation of block-level behavior.: A natural next step is to validate our simulated block-level behavior against detailed traces from deployed blockchains such as Bitcoin, Bitcoin Cash, Litecoin, Dogecoin, and NEAR. In this work we tuned block sizes, target intervals, and hashrates to roughly match published protocol parameters and aggregate performance, but we did not attempt a fine-grained replay of real chains. Future work will incorporate historical block headers, transaction counts, and mempool data to check whether our distributions of block sizes, inter-block times, and orphaned blocks track those observed on mainnet and long-running testnets. This will help quantify how much realism we gain or lose from our current abstractions and where the simulator needs additional fidelity.

b) Calibration against on-chain throughput and fee dynamics.: Beyond matching block times and sizes, we would like to calibrate our simulator against end-to-end application-level metrics such as realized throughput, fee levels, and waiting-time distributions for transactions. Public blockchain explorers expose time series of confirmed transactions per second, fee spikes during congestion, and tail latencies for inclusion in a block; we can use these measurements to tune our transaction arrival processes, fee models, and mempool policies. By fitting these higher-level curves, we can check whether a given MEMO or NEAR-like configuration would have produced behavior consistent with what users actually experience on contemporary chains, rather than just matching theoretical capacity.

c) Cross-chain generality and robustness of MEMO-style sharding.: A third direction is to test how robust our MEMO-style sharding gains are when we port the same design choices to other economic and protocol environments. For instance, we can instantiate MEMO-like configurations under Bitcoin-style proof-of-work economics, proof-of-stake validator sets, or hybrid committee-based protocols and compare the resulting throughput and finality trends to those we observe under our current parameter choices. This cross-chain validation will help distinguish which of our results are artifacts of a particular configuration versus general consequences of winner-based sharding and metronome-style coordination, and it will clarify how portable our proposed settings are to future L1 designs.

d) Network-model validation using real latency and bandwidth measurements.: On the network side, our default experiments assume a stylized but fixed latency and bandwidth model (for example, an 80 ms RTT between validators and separate “control” and “data” links). In future work, we plan to validate and refine this model using empirical measurements from geographically distributed validator deployments and public measurement platforms. By fitting our latency matrix and bandwidth assumptions to real-world measurements between regions and cloud providers, we can evaluate how sensitive MEMO and NEAR-like sharded configurations are to realistic path asymmetries, congestion, and heterogeneity in validator connectivity. This will help us understand whether our optimistic US-style cluster assumptions understate the network overhead in more diverse global settings.

e) Modeling heterogeneous, dynamic, and adversarial networks.: Finally, we intend to extend the network model to capture dynamic and potentially adversarial conditions that current experiments abstract away. This includes modeling churn in validator participation, temporary bandwidth reductions, and regional outages, as well as adversarial behaviors such as targeted eclipse-style delays on specific shards or leaders. By injecting these perturbations into the simulator and comparing the resulting block times, orphan rates, and cross-shard consistency to what has been observed during real incidents on production networks, we can stress-test the resilience of MEMO’s leader-based metronome and NIC-token mechanisms. Such network-centric validation will be essential for arguing that the throughput and finality gains we obtain from sharding do not come at the cost of unacceptable fragility under realistic Internet conditions.