From Topology to Performance: Workload-Driven Scalability Limits in the NEAR Sharded Blockchain

Md. Ahsan Habib and Lu Peng

***Abstract*— Sharded blockchain architectures promise to overcome scalability limitations of decentralized networks, yet their behavior in production remains poorly understood. We present a measurement study of the NEAR Protocol across 4, 6, 8, and 9 shard configurations. Using trace-driven analysis, we find that throughput improves by approximately 5–8.7$\times$ over the 4 shards baseline, although gains are non-monotonic, while latency increases in the 6 shards era but decreases in later expansions. To explain these trends, we construct three graph representations—Value Flow Graph (VFG), Contract Interaction Graph (CIG), and Account Control Graph (ACG)—capturing economic activity, contract execution, and account management. All exhibit highly skewed structures with a few dominant hubs, leading to significant shard-level load imbalance. We develop a data-driven, queueing-inspired interpretation relating computational load (throughput $\times$ execution cost) to latency, showing that high-load shards experience increased latency and persistent congestion. Jensen–Shannon divergence reveals that user-level activity varies significantly across shard configurations, while contract-level behavior remains relatively stable, and cross-shard communication increases with shard count. These results show that throughput scaling is non-monotonic and depends not only on shard count but also on workload distribution, cross-shard communication, and protocol design.**

***Index Terms*— Sharded Blockchain, Scalability, Graph-based Analysis, Queueing-inspired Metric**

I. Introduction

Blockchain technology was introduced with the advent of Bitcoin in 2008, establishing a decentralized paradigm for trustless value transfer. Since then, it has attracted substantial attention from both academia and industry, with proposed applications spanning finance [1], healthcare [2], education [3], and industry [4]. Despite this broad interest, the widespread deployment of blockchain-based systems remains constrained by inherent scalability limitations, particularly in terms of transaction throughput and latency.

Sharded blockchain architectures have emerged as a promising approach to improving scalability by partitioning the blockchain system into multiple *shards*, allowing transactions and state updates to be processed in parallel while aiming to preserve security and decentralization [5]. However, although a large body of research has explored sharding-based designs [6]–[9], only a limited number of production-grade systems such as NEAR Protocol [10] and Zilliqa [11] have been successfully deployed in practice.

The authors are with the Department of Computer Science, Tulane University, New Orleans, LA, USA (e-mail: mhabib@tulane.edu; lpeng3@tulane.edu).

This work used computational resources provided by the Louisiana Optical Network Infrastructure (LONI). The artifact is available at https://github.com/mahsanhabib/near-sharding-artifact.

NEAR Protocol is a production-grade sharded blockchain designed for scalable smart contract execution and decentralized applications. It employs the Nightshade architecture to enable parallel transaction processing while maintaining a single logical chain [12]. The platform supports low-latency execution and flexible account models, with its native token, NEAR Ⓝ, serving as both a medium of exchange and a governance mechanism.

Despite the deployment of production sharded blockchains, there is limited empirical evidence on how real-world workload structure influences shard-level performance. This study asks: **(RQ1)** how throughput and latency evolve as NEAR expands from 4 to 9 shards, **(RQ2)** how transaction topology and workload skew differ across value-flow, contract-interaction, and account-control activities, and **(RQ3)** how workload concentration, cross-shard communication, and shard-level load imbalance are associated with congestion and scalability. To answer these questions, we present a comprehensive measurement study of the NEAR Protocol, combining graph-based analysis with a data-driven, queueing-inspired interpretation to explain observed performance behavior. This paper makes the following contributions:

1) To the best of our knowledge, we present the first large-scale empirical analysis of NEAR sharded blockchain.
2) We construct three activity-driven graphs—Value Flow Graph (VFG), Contract Interaction Graph (CIG), and Account Control Graph (ACG)—to capture economic activity, contract execution, and account management.
3) We show that shard-level performance is strongly associated with topology-induced workload skew, characterized by super-hubs and long-tail nodes.
4) We develop a queueing-inspired framework that helps explain how activity concentration is associated with shard-level congestion and performance imbalance.
5) We quantify workload evolution across shard configurations using Jensen–Shannon divergence, showing that user-level activity varies significantly while contract-level behavior remains relatively stable.

II. Background and Related Work

NEAR is a deployed sharded blockchain which shards state and computation while maintaining a single logical chain [12]. Each block contains shard-specific chunks produced by stake-weighted validators assigned per shard and rotated every epoch. Validators participate under a Proof-of-Stake (PoS) mechanism, where they are responsible only for

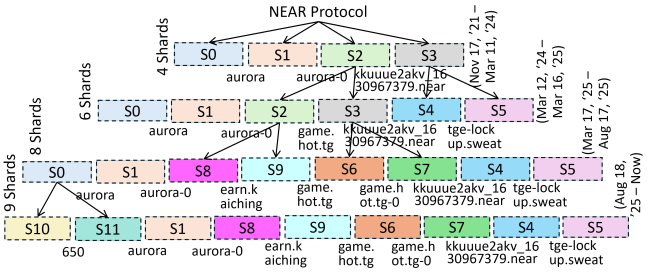


Fig. 1. Shard creation observed over time in the NEAR Protocol.

validating their assigned shards, reducing hardware requirements and improving decentralization [13], [14].

Account Model. NEAR uses a unified account model for both users and contracts, supporting flexible key management and hierarchical subaccounts [15]. Transactions consist of *actions* (e.g., transfers, contract calls, staking, deployment) that generate *receipts*, which are the basic units of execution. Smart contracts are written in Rust or AssemblyScript, compiled to WebAssembly, and deployed for execution.

Shard Dynamics. NEAR's sharding roadmap progressed from Simple Nightshade (Phase 0), which introduced state sharding without execution sharding in late 2021, toward full state and execution sharding under Nightshade, with shard-specific validation and chunk-only producers introduced in Phase 1. Subsequent upgrades, including Nightshade 2.0 in Aug 2024, enabled stateless validation and significantly improved shard scalability. In production, NEAR evolved from a single-shard mainnet to 4 shards in late 2021, briefly to 5 shards in Mar 2024, then to 6 shards, and subsequently expanded to 7, 8, and finally 9 shards by Aug 2025 using fast single-block resharding. While live shard splits are now supported, fully automatic dynamic resharding remains a future objective [16].

Shard Creation and Assignment. Shard membership in NEAR is determined by lexicographically ordered *boundary accounts* [17], where each shard corresponds to a contiguous range of account identifiers. When increasing the shard count, the protocol splits existing shards by introducing new boundary accounts, deterministically repartitioning the namespace while preserving locality. This enables incremental scaling without global rebalancing (Fig. 1).

Shard Evolution. NEAR evolved from 4 to 9 shards through successive splits driven by changing system bottlenecks. The 4 shards configuration (Nov 2021 – Mar 2024) became constrained by CPU-intensive workloads from high-frequency accounts. The transition to 6 shards alleviated compute pressure by distributing execution across more validators. The 8 shards expansion (Mar–Aug 2025) was primarily a technical upgrade to enable stateless validation (Nightshade 2.0) and reduce memory requirements. Finally, the 9 shards configuration addressed state bloat caused by a large number of low-activity accounts, shifting the bottleneck from CPU to memory.

A. Related Works

Prior works have analyzed Ethereum transaction activity using graph-based methods, including transaction networks, temporal modeling, anomaly detection, and ERC20 ecosystem analysis [18]–[20]. Other studies examine contract behavior or cross-platform comparisons [21], [22]. Blockchain sharding studies have explored trust-aware node allocation [23] and load-balanced account migration [24]; however, these works primarily propose sharding mechanisms rather than empirically characterizing workload-driven scalability limits in a production sharded blockchain. In contrast, our work connects real NEAR workload topology to shard-level throughput, latency, cross-shard communication, and congestion behavior.

TABLE I
HIGH-LEVEL GRAPH CONSTRUCTION METHODOLOGY.

Graph	Purpose	Action Kinds
VFG	Track token flows	*transfer*, *stake*
CIG	Contract deployment & usage	*deploy_contract*, *deploy_global_contract*, *deploy_global_contract_by_account_id*, *function_call*, *use_global_contract*, *use_global_contract_by_account_id*
ACG	Account/key management	*create_account*, *delete_account*, *add_key*, *delete_key*, *delegate_action*

III. Proposed Methodology

A. Dataset

We obtained the entire transaction and contract history of the NEAR Protocol, spanning from the genesis block up to Dec 31, 2025, directly from the NEAR Public Data Lake on Google BigQuery using SQL [25]. The earliest block in the NEAR mainnet corresponds to block height 9,820,210, marking the genesis block following the re-genesis event around Aug 2020. While the network has continuously produced blocks since then, the dataset does not always contain perfectly consecutive block heights. For example, some blocks such as 9,820,211 are missing, likely due to blocks being proposed but not finalized (orphaned) or reorganized. On the mainnet, each epoch spans 43,200 blocks, defining the cadence of validator rotations and shard reconfigurations.

B. Graph Construction

We model NEAR activity using three interaction graphs capturing complementary aspects of system behavior. The Value Flow Graph (VFG) represents economic activity through token transfers and staking operations. The Contract Interaction Graph (CIG) captures contract deployment and execution dynamics, while the Account Control Graph (ACG) models account lifecycle and key management actions. Together, these graphs provide a structured view of user and contract behavior in the system. Table I summarizes the corresponding action types and their mapping to each graph.

1) VFG Modeling: We model VFG as a directed, weighted graph $VFG = (V, E, W)$, where V denotes the set of nodes representing accounts (users or contracts), E denotes the set of directed edges, and W denotes the set of edge weights. Formally, each edge $(v_i, v_j) \in E$ represents a native token transfer from node v_i to node v_j, with weight

$w_{ij} \in W$ capturing the interaction frequency between v_i and v_j. Nodes correspond to NEAR accounts, while edges capture economic interactions such as *transfer* and *stake*.

2) CIG Modeling: CIG is represented as a directed, weighted graph $CIG = (V, E, W)$, where V is the nodes (users and contracts), E is directed edges representing interactions between nodes, and W is the edge weight set. Each edge $(v_i, v_j) \in E$ is assigned a weight $w_{ij} \in W$, capturing the number of times node v_i interacts with node v_j through contract-related actions. This graph provides a structured view of how contracts are deployed and invoked across the network and the identification of highly active contracts.

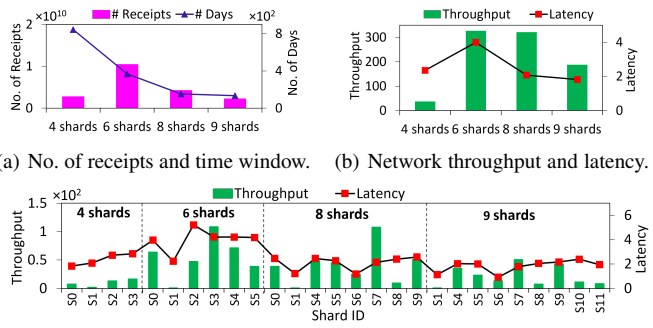
3) ACG Modeling: ACG is modeled as a directed, weighted graph $ACG = (V, E, W)$, where V represents user and contract accounts, E is the set of directed edges representing account management and delegated interactions, and W contains the corresponding weights. Each edge $(v_i, v_j) \in E$ has a weight $w_{ij} \in W$, reflecting the number of actions initiated by node v_i affecting node v_j. This graph enables analysis of account lifecycle behaviors, delegation patterns, etc. providing insights into user activity and protocol-level authorization within the NEAR network.

C. Queueing-Based Interpretation of Shard Congestion

We model each shard s_i as a single-server queue, where receipts arrive at rate $\lambda_i(t)$ and require service time $S_i(t)$. The arrival rate is estimated as $\lambda_i(t) = n_i(t)/\Delta t$, where $n_i(t)$ is the number of executed receipts over interval t. In classical queueing theory, utilization is defined as $\rho_i(t) = \lambda_i(t)\mathbb{E}[S_i(t)]$, and latency increases rapidly as utilization approaches system capacity. However, the service time is not directly observable from on-chain data. To bridge this gap, we approximate $\lambda_i(t) \approx \text{Throughput}_i(t)$ and use the average gas per execution $\overline{G}_i(t)$ as a proxy for service demand, i.e., $\mathbb{E}[S_i(t)] \propto \overline{G}_i(t)$. This approximation relies on a stable-queue assumption; under congestion, $\text{Throughput}_i(t)$ reflects observed completions rather than the true offered arrival rate. We therefore define an effective shard load $L_i(t) = \text{Throughput}_i(t) \times \overline{G}_i(t)$, capturing the combined effect of arrival intensity and computational cost. Since $L_i(t)$ is unnormalized, it is not bounded by 1; larger values indicate higher queueing pressure and are expected to correspond to increased latency.

D. Performance and Workload Metrics

We evaluate NEAR Protocol using throughput and latency at both network and shard level. Throughput is measured as receipts/s and latency as execution time (s), as receipts (not transactions) are the true execution units and capture cross-shard execution. To characterize workload structure, we categorize transaction actions into four types: user-to-user (U2U), user-to-contract (U2C), contract-to-user (C2U), and contract-to-contract (C2C). We further analyze user and contract activity distributions using statistical measures such as degree, activity frequency, and inequality (e.g., Gini coefficient). To quantify distributional differences across shard



(a) No. of receipts and time window. (b) Network throughput and latency.

(c) Per-shard throughput and latency.

Fig. 2. Observed performance across different shard configurations: (a) workload (receipts and observation window), (b) network-level throughput and latency, and (c) shard-level throughput and latency.

configurations, we use Jensen–Shannon (JS) divergence [26], which ranges from 0 to 1 (log base 2), with higher values indicating greater divergence. We measure cross-shard action ratio for communication overhead and define shard load to analyze its relationship with latency.

IV. NEAR Protocol Analysis

We organize the analysis around the research questions: Section IV-A examines performance evolution across shard configurations (RQ1), Section IV-B analyzes workload topology and skew using VFG, CIG, and ACG (RQ2), and Sections IV-C–IV-E study workload evolution, cross-shard communication, and load–latency behavior associated with congestion and scalability (RQ3).

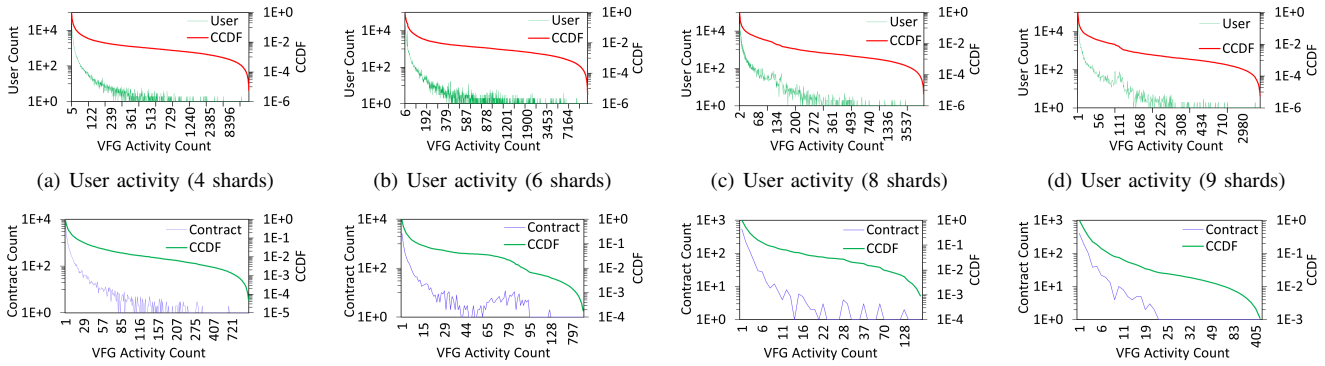
A. Throughput and Latency

Network-level Trends. The NEAR Protocol shows substantial throughput gains with sharding while maintaining relatively low latency (Fig. 2(b)). Compared to the 4 shards baseline (37.5 receipts/s, 2.36 s), the 6 shards configuration achieves 327 receipts/s (8.7× increase) with higher latency (4 s), reflecting increased coordination overhead. At 8 shards, throughput remains high at 321 receipts/s (8.6×), while latency drops to 2.08 s, indicating improved execution efficiency. However, at 9 shards, throughput declines to 188 receipts/s (5×), despite achieving the lowest latency (1.82 s).

As shown in Fig. 2(a), total receipts and observation windows vary across configurations, indicating differences in workload intensity and system conditions. While throughput

Summary of Main Findings

- **Performance:** Throughput rises from 37.5 rec./s at 4 shards to 327 rec./s at 6 shards, remains high at 321 rec./s at 8 shards, and drops to 188 rec./s at 9 shards; latency peaks at 4.0 s and falls to 1.82 s.
- **Topology:** Workloads are highly skewed: top 1% users account for 96.6–98.0% of VFG activity, top 1% contracts for 93.0–99.5% of CIG activity, and top 1% users for 97.3–99.96% of ACG activity.
- **Scalability:** Cross-shard actions increase from 0.57 to 0.79 as shard count grows, while higher effective shard load is associated with higher latency.



(a) User activity (4 shards) (b) User activity (6 shards) (c) User activity (8 shards) (d) User activity (9 shards)

(e) Contract activity (4 shards) (f) Contract activity (6 shards) (g) Contract activity (8 shards) (h) Contract activity (9 shards)

Fig. 3. VFG activity distributions (log scale) for users (top) and contracts (bottom) across shard configurations, showing heavy-tailed workload skew.

may appear to increase with total receipts, it does not scale proportionally (e.g., 8 and 9 shards), suggesting that system performance is not determined by workload volume alone. The observed non-monotonic scaling reflects the combined influence of increased shard, workload distribution, cross-shard communication, and system design. From a queueing perspective, skew in arrival intensity and computational demand leads to imbalanced effective load $L_i(t)$ across shards, resulting in localized congestion that constrains overall throughput.

Shard-level Trends. Figure 2(c) shows shard-level throughput and latency across configurations. In the 4 shards setup, throughput ranges from 1.4 to 16.2 receipts/s. With 6 shards, some shards exhibit substantial increases, reaching up to 107.9 receipts/s, while others remain lower. In the 8 shards configuration, throughput becomes highly skewed (0.2–107.4 receipts/s), indicating uneven load distribution. With 9 shards, throughput is less skewed (0.26–50.2 receipts/s), suggesting improved distribution across shards. However, increasing shard count boosts throughput for heavily loaded shards, though some shards remain underutilized.

On the other hand, in the 4 shards setup, latency ranges from 1.8 to 2.8 s, reflecting uneven load distribution. With 6 shards, latency increases for some shards (up to 5.2 s), indicating localized congestion. In the 8 shards configuration, latency becomes more balanced (1.2–2.6 s), suggesting improved load sharing. With 9 shards, latency further decreases and stabilizes (0.9–2.4 s), demonstrating reduced congestion and more uniform performance across shards.

B. Graph-based Analysis

Value Flow Graph (VFG) Analysis. Table II shows that U2U interactions dominate VFG activity, comprising approximately 91-99% of all actions across shard configurations. Contract-related flows (C2U, U2C, C2C) account for only a small fraction. A summary of user activity statistics

TABLE II

FLOW DISTRIBUTION (%) IN VFG ACROSS SHARD CONFIGURATIONS.

Config	U2U	C2U	U2C	C2C	Config	U2U	C2U	U2C	C2C
4 shards	90.8	7.9	1.3	0.1	6 shards	96.7	3.2	0.04	–
8 shards	99.95	0.03	0.01	–	9 shards	99.96	0.02	0.02	–

– indicates values $< 0.01\%$.

TABLE III

VFG (USER) TOPOLOGY AND DEGREE DISTRIBUTION.

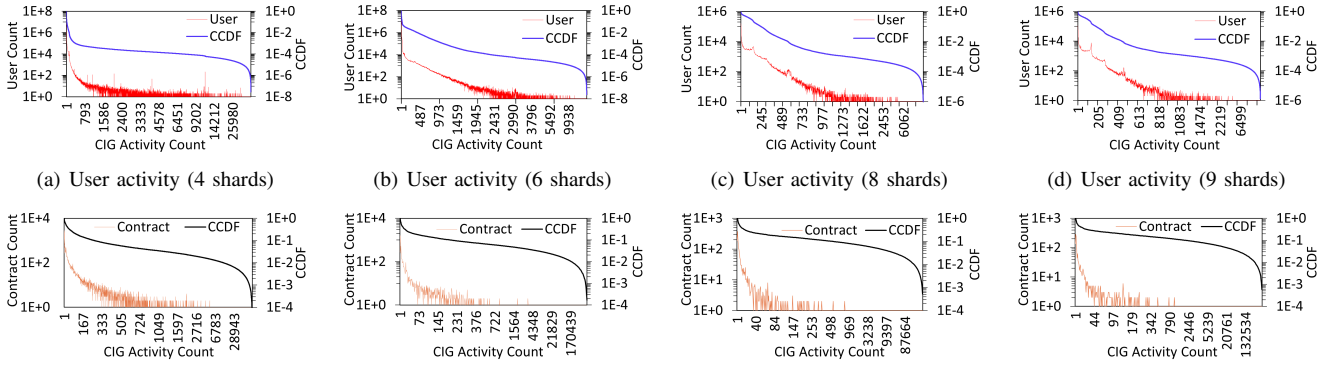
Metric	4 shards	6 shards	8 shards	9 shards
Periphery users (Retail activity)				
Users with degree $k = 1$	0	0	0	55,282
Users with degree $k = 2$	0	0	68,116	67,449
Users with degree $k \leq 10$	127,166	128,146	153,149	165,621
Core users (Liquidity/Exchange hubs)				
Max user degree (k_{max})	39,503,587	86,466,619	40,841,898	26,408,881
2nd highest degree	23,672,749	18,966,411	1,584,432	143,026
Hubs with $k > 100{,}000$	35	17	5	3
Distribution and concentration				
Mean user degree	583.60	560.82	260.99	157.23
Median degree (P50)	7	10	3	2
99th percentile degree (P99)	182	309	127	97
Skew ratio (P99 / P50)	26.0	30.9	42.33	48.5
Gini coefficient	0.9885	0.9825	0.9883	0.9885
Top 1% activity share	0.9803	0.9666	0.9740	0.9737

are provided in Table III and Fig. 3(a-d), whereas contract activity statistics are presented in Table IV and Fig. 3(e-h).

Economic Centrality and Liquidity Hubs (User): VFG exhibits a highly skewed, scale-free structure with a small number of *economic super-hubs* dominating value transfers (Table III). This is evident from the high Gini coefficient ($\sim$0.98) and the top 1% of users accounting for over 96–98% of activity. The skew ratio further increases across configurations, indicating growing disparity between typical users and high-activity hubs. At the extreme, a single user processes tens of millions of transfers (e.g., 86.4 million in 6 shards), highlighting strong concentration of economic flow.

The Retail Periphery (User): In contrast, most users remain weakly active, with low median degree (P50 = 2–10) and a limited number of degree $k = 1$ accounts (e.g., 55,282 in 9 shards). Although low-degree users increase with shard count, their contribution to total activity is negligible. This indicates that VFG user activity is driven by a few high-frequency hubs rather than widespread P2P interactions.

Supply-Side Concentration and Liquidity Sinks (Contract): VFG from the contract perspective exhibits a highly centralized *Hub-and-Spoke* topology (Table IV). In the 4 shards setup, this skew is extreme: the top contract processes over 1.25 million transfers, while thousands of contracts remain nearly idle. This concentration is reflected in a high Gini value (0.97) and top 1% activity share exceeding 91%,



(a) User activity (4 shards) (b) User activity (6 shards) (c) User activity (8 shards) (d) User activity (9 shards)

(e) Contract activity (4 shards) (f) Contract activity (6 shards) (g) Contract activity (8 shards) (h) Contract activity (9 shards)

Fig. 4. CIG activity distributions (log scale) for users (top) and contracts (bottom) across shard configurations, showing heavy-tailed workload skew.

TABLE IV

VFG (CONTRACT) TOPOLOGY AND INTERACTION DISTRIBUTION.

Metric	4 shards	6 shards	8 shards	9 shards
Contract periphery (Idle or Low-usage contracts)				
Contracts with degree $k = 1$	7,556	2,599	548	405
Contracts with degree $k = 2$	3,332	976	221	193
Contracts with degree $k \leq 10$	17,227	4,915	1,040	848
Contract core (Service hubs)				
Max contract degree (k_{max})	1,258,497	28,378	701	759
2nd highest degree	19,472	4,324	309	405
Contracts with degree $k > 1{,}000$	18	5	0	0
Distribution and concentration				
Mean contract degree	74.88	14.08	5.10	5.51
Median degree (P50)	2	2	2	2
99th percentile degree (P99)	134	88	43	56
Skew ratio (P99 / P50)	67	44	21.5	28
Gini coefficient	0.9692	0.8717	0.7036	0.7099
Top 1% activity share	0.9142	0.6073	0.3778	0.4118

TABLE VI

CIG (USER) TOPOLOGY AND ACTIVITY DISTRIBUTION.

Metric	4 shards	6 shards	8 shards	9 shards
Retail engagement (Retention gap)				
Users with degree $k = 1$	18,283,919	16,920,757	162,604	109,244
Users with degree $k = 2$	3,527,791	5,500,747	72,063	55,856
Users with degree $k \leq 10$	29,082,840	34,779,055	370,277	305,479
Automated actors (High-Frequency / Bots)				
Max user activity (k_{max})	90,070,059	95,240,633	46,930,390	31,877,900
2nd highest activity	26,008,087	24,488,155	2,865,670	2,316,705
Hubs with $k > 100{,}000$	177	62	68	140
Distribution and concentration				
Mean activity	18.11	23.42	157.34	138.23
Median activity (P50)	1	2	49	30
99th percentile (P99)	122	420	562	440
Skew ratio (P99 / P50)	122.00	210.00	11.47	14.67
Gini coefficient	0.8872	0.8894	0.7769	0.8159
Top 1% activity share	0.6307	0.5405	0.4588	0.5367

indicating strong dominance of a few service hubs.

Long-Tail Contract Activity: Across all configurations, most contracts remain weakly active, with median degree fixed at 2 and a large fraction handling only a few interactions (e.g., 2,599 contracts with $k = 1$ in 6 shards). Although concentration decreases in later phases (Gini drops to ∼0.70 and top 1% share to ∼40%), activity remains highly skewed, with a small set of contracts sustaining the majority of execution.

Insight 1. A small fraction of users and contracts dominate VFG activity, creating persistent hotspots across shards.

Contract Interaction Graph (CIG) Analysis. Table V shows that for CIG, the majority of actions are contract-related flows (U2C and C2C), unlike VFG. A summary of user activity statistics are provided in Table VI and Fig. 4(a-d), whereas contract activity statistics are presented in Table VII and Fig. 4(e-h).

Degree Distribution and Structure (User): CIG exhibits a highly skewed, heavy-tailed structure with a pronounced core-periphery pattern (Table VI). This is reflected in high inequality (Gini = 0.78–0.89) and the top 1% of users accounting for 45–63% of total activity. The skew ratio (P99/P50) further highlights the large gap between typical users and high-activity nodes.

TABLE V

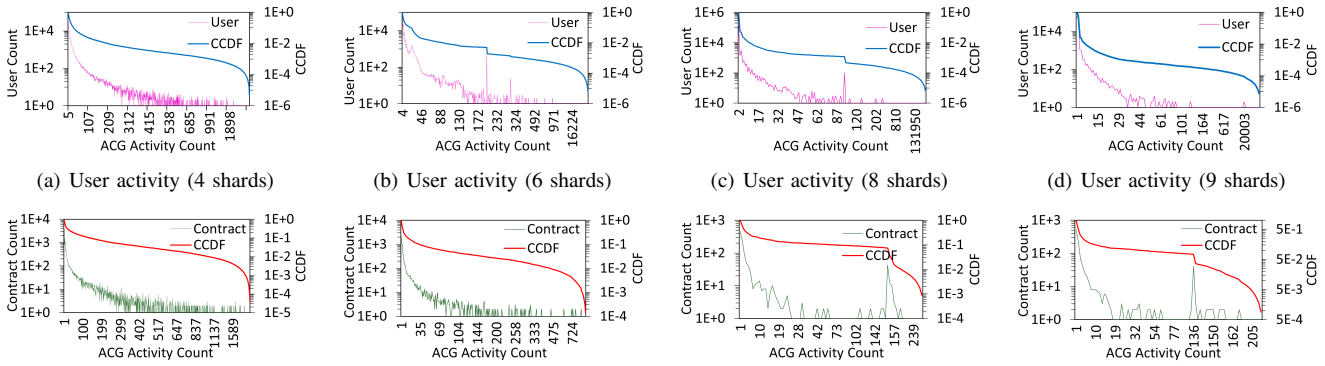
FLOW DISTRIBUTION (%) IN CIG ACROSS SHARD CONFIGURATIONS.

Config	U2U	C2U	U2C	C2C	Config	U2U	C2U	U2C	C2C
4 shards	89.8	9.6	0.6	0.02	6 shards	0.7	0.02	85.6	13.8
8 shards	1.6	0.02	92.3	6.1	9 shards	1.8	0.05	86.1	12.1

The Super-Hubs (Compute Saturation): The head of the distribution contains a small number of extremely high-activity nodes, with maximum activity reaching up to 95.2 million actions. These super-hubs dominate execution workload and are primary sources of cross-shard contention and CPU saturation. Their presence implies that uniform sharding is insufficient, as workload is concentrated on a few high-degree nodes rather than evenly distributed.

Dust Nodes and State Bloat: The tail of CIG contains a massive number of low-activity nodes, with millions of users having degree $k = 1$ (e.g., 18.2 million in 4 shards and 16.9 million in 6 shards). While these nodes contribute little to execution load, they dominate the storage footprint, leading to significant state growth. This shift in workload–from compute-intensive hubs to storage-heavy periphery–motivates shard expansions aimed at managing memory pressure.

Mega-Application Phenomenon (Contract): CIG exhibits stronger centralization than VFG, with a few *mega-applications* dominating execution load (Table VII). In the 6 shards era, the most active contract executed over 602 million interactions, indicating that computational demand is



(a) User activity (4 shards) (b) User activity (6 shards) (c) User activity (8 shards) (d) User activity (9 shards)

(e) Contract activity (4 shards) (f) Contract activity (6 shards) (g) Contract activity (8 shards) (h) Contract activity (9 shards)

Fig. 5. ACG activity distributions (log scale) for users (top) and contracts (bottom) across shard configurations, showing relayer-driven dominance.

TABLE VII

CIG (CONTRACT) TOPOLOGY AND ACTIVITY DISTRIBUTION.

Metric	4 shards	6 shards	8 shards	9 shards
Contract periphery (Low-activity contracts)				
Contracts with degree $k = 1$	267	269	188	137
Contracts with degree $k = 2$	2,840	1,596	376	286
Contracts with degree ≤ 10	9,807	3,569	1,185	972
Contract core (Systemic hubs)				
Max contract activity (k_{max})	148,292,092	602,457,645	94,958,635	49,990,543
2nd highest activity	119,643,232	144,422,971	46,930,390	31,877,900
Hubs with $k > 100{,}000$	80	78	35	43
Distribution and concentration				
Mean activity	27,270.95	189,323.7	94,606.28	701,44.53
Median activity (P50)	18	6	5	6
99th percentile (P99)	11,887	189,289	384,440	525,073
Skew ratio (P99 / P50)	660.38	31,548.17	76,888	87,512.17
Gini coefficient	0.9986	0.9987	0.9968	0.9939
Top 1% activity share	0.9946	0.9943	0.9744	0.9304

concentrated in a small set of high-impact applications. So, scalability depends on isolating these hotspots, motivating compute-aware sharding rather than uniform load balancing.

Deploy-and-Initialize Pattern: Unlike user-centric graphs where the mode is $k = 1$, CIG consistently peaks at $k = 2$. For example, in the 4 shards era, significantly more contracts exhibit exactly two interactions than one. This reflects a typical contract lifecycle–deployment followed by initialization–indicating that the effective working set of active contracts begins at $k = 2$.

Account Control Graph (ACG) Analysis. Table VIII shows that ACG activity is overwhelmingly dominated by U2U interactions, comprising approximately 97–98% of total actions across all shard configurations. Contract-related flows remain very small, each below 3%, reflecting that ACG graph primarily captures user-driven activity. A summary of user activity statistics are provided in Table IX and Fig. 5(a-d), whereas contract activity statistics are presented in Table X and Fig. 5(e-h).

Degree Distribution and Structure (User): ACG exhibits

TABLE VIII

FLOW DISTRIBUTION (%) IN ACG ACROSS SHARD CONFIGURATIONS.

Config	U2U	C2U	U2C	C2C	Config	U2U	C2U	U2C	C2C
4 shards	96.7	2.2	0.02	1.1	6 shards	97.5	2.4	0.01	0.01
8 shards	97.9	2.1	0.003	–	9 shards	98.4	1.5	0.003	–

– indicates values $< 0.01\%$.

TABLE IX

ACG (USER) TOPOLOGY AND ACTIVITY DISTRIBUTION.

Metric	4 shards	6 shards	8 shards	9 shards
Periphery Users (User onboarding patterns)				
Users with degree $k = 1$	0	0	0	52,943
Users with degree $k = 2$	0	0	133,932	80,694
Users with k_{min} activity	9,839 ($k = 5$)	70,838 ($k = 4$)	133,932 ($k = 2$)	52,943 ($k = 1$)
Core Users (Relayers & onboarding services)				
Max user activity (k_{max})	67,917,085	490,032,606	302,568,459	271,660,322
2nd highest activity	46,329,804	489,361,240	117,517,736	60,191,985
Hubs with $k > 100,000$	10	10	10	8
Distribution and concentration				
Mean activity	593.10	10,057.98	4,742.47	3,582.04
Median activity (P50)	9	5	2	2
99th percentile (P99)	203	94	10	6
Skew ratio (P99 / P50)	22.56	18.80	5	3
Gini coefficient	0.9853	0.9995	0.9996	0.9996
Top 1% activity share	0.9734	0.9992	0.9996	0.9996

Note: Gini values near 1.0 indicate extreme centralization via delegate-action relayers.

extreme centralization, with activity heavily concentrated in a small number of accounts (Table IX). This is reflected in near-maximal Gini coefficients (> 0.98) and the top 1% of users accounting for over 97–99.9% of total activity across all configurations. The low median activity (P50 = 2–9) combined with high maximum values indicates a highly skewed, hub-dominated structure.

Core Relayers (User): The head of ACG is dominated by a few infrastructure gateways acting as relayers. In the 4 shards configuration, a single account executes tens of millions of control actions, with similar dominance persisting across all phases. These relayers concentrate permissioning and gas payment, creating localized *hot-key* bottlenecks due to frequent state updates and nonce management.

Sticky User Base (User): In contrast, most users exhibit minimal control activity. The median remains low (P50 = 2–9), and many users perform only a few actions after onboarding (e.g., create account and add key). The spike of users with exactly two actions in the 8 shards configuration suggests standardized onboarding workflows, reinforcing that the majority of users remain largely inactive after initialization.

Degree Distribution and Structure (Contract): ACG from the contract perspective exhibits moderate concentration with a heavy-tailed structure (Table X). Inequality remains high (Gini ≈ 0.83–0.88), though significantly lower than the user-side ACG. The top 1% of contracts account for 15–47% of activity, indicating that control operations are distributed

TABLE X

ACG (CONTRACT) TOPOLOGY AND ACTIVITY DISTRIBUTION.

Metric	4 shards	6 shards	8 shards	9 shards
Onboarding patterns (Setup phase)				
Contracts with $k = 1$	4,328	2,198	451	535
Contracts with $k = 2$	3,358	1,080	2,230	2,237
Contracts with $k \leq 10$	14,511	5,633	919	967
Core infrastructure (Control hubs)				
Max activity (k_{max})	25,690	41,328	2,709	486
2nd highest activity	9,215	2,371	1,012	265
Contracts with $k > 10{,}000$	3	1	0	0
Distribution and concentration				
Mean activity	63.48	27.43	22.07	15.76
Median (P50)	7	3	2	2
99th percentile (P99)	1,050	427	160	171
Skew ratio (P99/P50)	150.00	142.33	80.00	85.50
Gini coefficient	0.8444	0.8787	0.8448	0.8290
Top 1% activity share	0.2834	0.4686	0.2565	0.1521

across multiple endpoints rather than dominated by a single entity.

Registrar and Control Hubs: The head of the distribution contains a small number of infrastructure contracts responsible for account management. In the 4 shards configuration, the most active contract processes up to 25,690 control actions, with only a few contracts exceeding 10K actions. These hubs act as entry points for account creation and key management, creating localized write pressure, although far less pronounced than user-side relayer dominance.

Setup-Phase Signature: A distinct onboarding pattern emerges in the 8 and 9 shards configurations, where the mode shifts to $k = 2$ rather than $k = 1$. For example, in the 9 shards setup, 2,237 contracts receive exactly two actions, compared to 535 with one. This indicates a consistent two-step onboarding process (e.g., *create_account* followed by *add_key*). After this setup phase, contracts become largely inactive in ACG, confirming that account management is primarily a setup workload rather than a sustained runtime workload.

Insight 2. ACG is dominated by relayers, creating centralized control and hot-key bottlenecks, while contract-activity shows structured onboarding with more distributed control.

C. Distributional Divergence

To quantify how distributions evolve across configurations, we compute the JS divergence between all pairs of shard settings. Figure 6 shows that user-level activity exhibits substantially higher divergence than contract-level behavior. In particular, ACG user activity shows the highest sensitivity, with divergence reaching up to 0.95 (4 vs 9 shards) and remaining high across configurations (e.g., >0.92 for 4–8 and 6–9), while dropping to 0.23 between 8 and 9 shards, indicating partial stabilization in distributional patterns, despite continued increase in cross-shard communication.

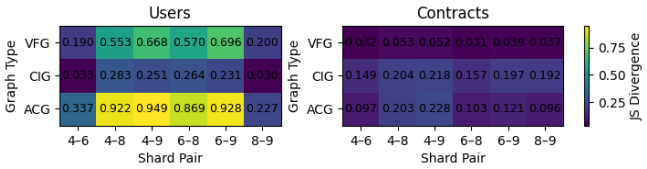
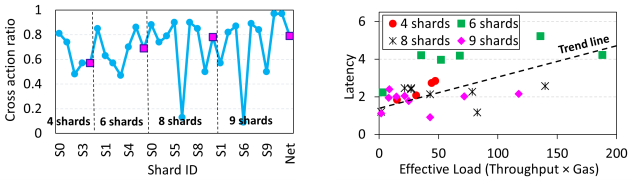


Fig. 6. JS divergence across shard configurations. User-level activity shows higher divergence than contract-level behavior across VFG, CIG, and ACG.



(a) Cross-shard action ratio. (b) Shard load vs. latency.

Fig. 7. Cross-shard communication and load–latency relationship. (a) Cross-shard action ratio capturing communication overhead (square marker represents cross-shard action ratio for network). (b) Shard load (throughput × gas) versus latency, where higher load is associated with increased latency.

In contrast, contract-level distributions remain relatively stable. For example, VFG contract divergence stays below 0.06 across all configurations, while CIG contract divergence remains within 0.15–0.22. User-level VFG and CIG exhibit moderate divergence (e.g., 0.55–0.69 for VFG and 0.23–0.28 for CIG), indicating that user behavior evolves more significantly than contract execution patterns.

D. Cross-Shard Communication

Figure 7(a) shows cross-shard action ratios for the network (square markers) and individual shards (circle markers). At the network level, the ratio increases steadily from 0.57 (4 shards) to 0.79 (9 shards), indicating that a larger fraction of actions require cross-shard coordination as the number of shards grows. At the shard level, the ratios exhibit substantial variability within each configuration. For example, in the 8 shards setup, values range from 0.13 to 0.90, while in the 9 shards configuration they span 0.09 to 0.97, highlighting significant heterogeneity across shards. These results indicate that increasing shard count not only raises overall overhead but also amplifies imbalance in cross-shard interactions across shards.

E. Queueing Interpretation of Load–Latency

We examine how shard-level workload imbalance affects performance within each configuration. Figure 7(b) shows a moderate positive correlation between load and latency across shards (Pearson correlation $r = 0.51$, with statistical significance $p < 0.01$). For example, in the 6 shards configuration, heavily loaded shards (load > 135) exhibit latency above 5.2 s, while lightly loaded shards (load $\sim$3) remain near 2.2 s. Similar patterns are observed in other configurations, where higher-load shards consistently experience higher latency. Importantly, load varies significantly

Key Takeaways

1) Throughput scaling is non-monotonic and does not depend on shard count alone.
2) Workload skew creates uneven shard utilization and persistent hotspots.
3) User activity varies across configurations, while contract behavior remains stable.
4) Cross-shard communication increases with shard count and varies across shards.
5) Higher effective load corresponds to higher latency.

across shards within the same configuration (up to an order-of-magnitude difference), leading to persistent hotspot shards that dominate overall system performance. It indicates that intra-configuration load imbalance is strongly associated with latency variation, and that shard-level load is a key driver of latency, particularly under high imbalance (e.g., 6 shards).

V. Discussion

• **Design Implications:** These findings suggest that future sharded blockchains should adopt workload-aware shard assignment rather than relying only on static or namespace-based partitioning. Shard placement should account for high-activity accounts, contract hotspots, and cross-shard locality to reduce compute bottlenecks and coordination overhead. Runtime monitoring of shard load, execution cost, and latency can further support adaptive rebalancing or hotspot isolation.

• **Generalizability:** Some observations are specific to NEAR, including its Nightshade architecture, account-based shard assignment, receipt-based execution model, and the 4–9 shard evolution studied here. However, the broader findings are likely relevant to sharded blockchains more generally: heavy-tailed workloads can create hotspot shards, cross-shard interactions can increase coordination overhead, and load imbalance can limit throughput and increase latency.

• **Limitations:** This study is observational and does not establish causality. The queueing perspective is interpretive rather than predictive; future work includes predictive modeling and adaptive sharding design.

VI. Conclusion

This paper presents a large-scale empirical study of the NEAR Protocol, showing that sharding improves throughput while maintaining low latency, but with uneven gains across shards. Using graph-based analysis (VFG, CIG, and ACG), we find that network activity is highly skewed, with a few high-degree nodes dominating execution and many low-activity nodes contributing to state growth, leading to both compute and memory bottlenecks. We further observe that user-level activity varies significantly across shard configurations, while contract-level behavior remains stable, and that cross-shard communication increases with shard count. A queueing-based interpretation shows that skewed workloads lead to non-uniform utilization and persistent hotspot shards, with higher load generally associated with increased latency. These results indicate that even as NEAR scales from 4 to 9 shards, performance gains remain limited by workload structure, motivating topology-aware and workload-aware sharding strategies.

References

[1] Philip Treleaven, Richard Gendal Brown, and Danny Yang. Blockchain technology in finance. *Computer*, 50(9):14–17, 2017.

[2] Mohammad Rifat Ahmed, Abdul Aziz, Md Motaleb Hossen Manik, and Md Ahsan Habib. Blockchain-aided comparative study of heart disease detection using machine learning-based approaches with an expanded dataset. *Computers in Biology and Medicine*, 195:110611, 2025.

[3] Md Ahsan Habib, Md Motaleb Hossen Manik, and Saklain Zaman. A blockchain-based technique to prevent grade tampering: A university perspective. In *2023 International Conference on Electrical, Computer and Communication Engineering (ECCE)*, pages 1–6. IEEE, 2023.

[4] Zhuowen Chen, Joseph Sarkis, and Abdullah Yildizbasi. Digital transformation for safer circular lithium-ion battery supply chains: a blockchain ecosystem-data perspective. *International Journal of Production Research*, 63(5):1585–1606, 2025.

[5] Mahdi Zamani, Mahnush Movahedi, and Mariana Raykova. Rapidchain: Scaling blockchain via full sharding. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 931–948, 2018.

[6] Hung Dang, Tien Tuan Anh Dinh, Dumitrel Loghin, Ee-Chien Chang, Qian Lin, and Beng Chin Ooi. Towards scaling blockchain systems via sharding. In *Proceedings of the 2019 International Conference on Management of Data (SIGMOD)*, pages 123–140, 2019.

[7] Zicong Hong, Song Guo, Peng Li, and Wuhui Chen. Pyramid: A layered sharding blockchain system. In *In Proceedings of the 2021 IEEE Conference on Computer Communications (INFOCOM)*, pages 1–10. IEEE, 2021.

[8] Vitalik Buterin. A roadmap for scaling rollups with data sharding. https://notes.ethereum.org/@vbuterin/data_sharding_roadmap, 2021. Accessed: 2025-12-31.

[9] Cong T Nguyen, Dinh Thai Hoang, Diep N Nguyen, Yong Xiao, Dusit Niyato, and Eryk Dutkiewicz. Metashard: A novel sharding blockchain platform for metaverse applications. *IEEE Transactions on Mobile Computing*, 23(5):4348–4361, 2023.

[10] NEAR Protocol. https://near.org/, 2025.

[11] The Zilliqa Team. The Zilliqa project: A secure, scalable blockchain platform. 2018.

[12] Alex Skidanov and Illia Polosukhin. Nightshade: Near protocol sharding design. https://nearprotocol.com/downloads/Nightshade.pdf, 2019.

[13] Yuming Huang, Jing Tang, Qianhao Cong, Richard TB Ma, Lei Chen, and Yeow Meng Chee. The last survivor of pos pools: Staker's dilemma. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 9(1):1–29, 2025.

[14] Fahad Saleh. Blockchain without waste: Proof-of-stake. *The Review of financial studies*, 34(3):1156–1190, 2021.

[15] Data Structures. https://nomicon.io/DataStructures/Account.html.

[16] NEAR Protocol. Nightshade 2 launches on NEAR mainnet. https://www.near.org/blog/nightshade-2-launches-on-near-mainnet-introducing-stateless-validation, 2024. Accessed: 2026-04-13.

[17] NEAR Shard Layout. https://github.com/near/nearcore.

[18] Ting Chen, Zihao Li, Yuxiao Zhu, Jiachi Chen, Xiapu Luo, John Chi-Shing Lui, Xiaodong Lin, and Xiaosong Zhang. Understanding ethereum via graph analysis. *ACM Transactions on Internet Technology*, 20(2):1–32, 2020.

[19] Dan Lin, Jiajing Wu, Qi Yuan, and Zibin Zheng. Modeling and understanding ethereum transaction records via a complex network approach. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 67(11):2737–2741, 2020.

[20] Weili Chen, Tuo Zhang, Zhiguang Chen, Zibin Zheng, and Yutong Lu. Traveling the token world: A graph analysis of ethereum erc20 token ecosystem. In *Proceedings of The Web Conference (WWW)*, pages 1411–1421, 2020.

[21] Lucianna Kiffer, Dave Levin, and Alan Mislove. Analyzing ethereum's contract topology. In *Proceedings of the Internet Measurement Conference (IMC)*, pages 494–499, 2018.

[22] Christian Cachin, Angelo De Caro, Pedro Moreno-Sanchez, Björn Tackmann, and Marko Vukolic. The transaction graph for modeling blockchain semantics. 2021.

[23] PeiYun Zhang, WeiFeng Guo, ZiJie Liu, MengChu Zhou, Bo Huang, and Khaled Sedraoui. Optimized blockchain sharding model based on node trust and allocation. *IEEE Transactions on Network and Service Management*, 20(3):2804–2816, 2023.

[24] Mingzhe Li, Wei Wang, and Jin Zhang. LB-Chain: Load-balanced and low-latency blockchain sharding via account migration. *IEEE Transactions on Parallel and Distributed Systems*, 34(10):2797–2810, 2023.

[25] NEAR BigQuery Public Dataset. https://docs.near.org/data-infrastructure/big-query, 2025. Accessed: 2026-04-13.

[26] Jianhua Lin. Divergence measures based on the Shannon entropy. *IEEE Transactions on Information theory*, 37(1):145–151, 2002.