Svedberg Open
DISSEMINATION OF KNOWLEDGE

International Journal of Artificial Intelligence and Machine Learning

Publisher's Home Page: https://www.svedbergopen.com/

Research Paper Open Access

Empirical Evaluation of Hybrid Ethereum-Ipfs Architectures Under High-Throughput Workloads: Storage Expansion, Memory Utilization, and Access Latency

Kshitiz Saxena[1], Dhowmya Bhatt[2*]

[1,2*]Department of Computer Science and Engineering, Faculty of Engineering and Technology, SRM Institute of Science and Technology, Delhi-NCR Campus, Delhi-Meerut Road, Modi Nagar, Ghaziabad, Uttar Pradesh, India

ABSTRACT

Scalability issues in Ethereum-based blockchain systems can be attributed to bloat in the global state, where every node is required to host a local copy of the account state, smart contracts and their respective storage in the state tries. The idea of using a combination of on-chain and off-chain approaches limits the growth of the global state by storing a portion of the payload in decentralized off-chain stores, such as the InterPlanetary File System (IPFS), and only committing to the ledger an immutable, fixed-size Content Identifier (CID). While the performance and system-level resource trade-offs of this approach have been investigated, it would benefit from quantifying these under real-world conditions with high-throughput, write-heavy loads. The performance characteristics of a hybrid, dual-layer system consisting of a Go-Ethereum (Geth v1.13.14) archive node coupled with an IPFS Kubo (v0.26.0) node and running a Hyperledger Caliper test benchmark workload with a high frequency transaction rate composed of writes, submitted from four concurrent workers with hard resource limits imposed on the Docker containers (1.5 CPUs, 2048 MB RAM), are described. Throughput, network I/O and storage I/O metrics are measured and analyzed using native HTTP RPC latencies and hardware counters. Results show that while limiting the growth of on-chain state using raw data payloads reduces the overall size from 77 MB to 95 MB over the course of the benchmark run. The latency of off-chain IPFS store is 57.1 ms in the worst case (95th percentile; P95) when operating at full saturation for four workers. On the other hand, the average access time for retrieving payloads via IPFS fetch is almost instantaneous (sub-5 ms). Container memory consumption starts out at 3500 MB, which peaks initially before flattening out around 1700-1780 MB (depending on the garbage collector mode of the Go runtime). These results offer definitive upper bounds for using a hybrid decentralized storage approach in edge scenarios with tight resource constraints.

Keywords: Blockchain storage, Ethereum, InterPlanetary File System (IPFS), Off-chain storage, State bloat mitigation, Hyperledger Caliper, Empirical performance evaluation, Container resource utilization.

INTRODUCTION

Blockchain networks executing EVM (Ethereum Virtual Machine) state transitions maintain cryptographic verifiability by requiring all validating nodes to record state modifications permanently within underlying persistent key-value datastores (such as LevelDB or Pebble) [4],. As transaction volume scales, this design induces severe state bloat, requiring validating nodes to maintain gigabyte-to-terabyte-scale Merkle-Patricia Trie (MPT) structures in persistent storage and memory [5, p. 1]. State bloat increases the hardware barrier to entry for full validating nodes, leading to centralization risks and performance degradation during block execution [6, p. 1],[5, p. 1].

To combat state expansion while preserving data immutability and verifiable integrity, modern decentralized applications adopt a hybrid storage architecture. In this paradigm, large application payloads (e.g., IoT sensor telemetry, medical records, supply chain manifests, media assets) are offloaded to an off-chain distributed content-addressed network—most notably the InterPlanetary File System (IPFS) [7]. Rather than embedding raw payload byte arrays into EVM contract storage slots, the client application ingests the payload into IPFS, receives a cryptographic Multihash Content Identifier (CID), and

submits only this reference string to the blockchain transaction envelope [7],[8, p. 115].
This hybrid approach offers considerable state reduction potential in theory. However, it also brings new challenges at the systems level when dealing with long-running high-throughput write workloads:
Network & RPC Bottlenecks: Local or remote submissions of thousands of concurrent write requests to IPFS daemons may lead to network or RPC bottlenecks that can cause saturation of local I/O operations and thus performance degradation due to resolving and downloading operations [9];
Resource Competition: If the blockchain node and IPFS daemon run on the same underlying compute resources as part of an edge or container deployment, they will compete for resources, which may impact storage performance [10];
Storage Amplification: Generating off-chain UnixFS blocks and building block-level DAGs require processing files into chunks and then hashing them [11, p. 161]. An extensive experimental study investigates the behavior of a hybrid Geth-IPFS system under a 250,000-transaction benchmark using Hyperledger Caliper. This enables us to analyze the system's behavior in a realistic setting.

I.1 Research Questions

This paper addresses three primary research questions:
Latency Profiles: How does an off-chain IPFS storage model affect end-to-end read and write latencies under sustained heavy write workloads across parallel worker threads?
Storage Expansion: What is the exact LevelDB storage growth rate when offloading payloads versus full on-chain state logging across a 250,000-transaction trajectory?
Resource Utilization: What are the memory (RAM) and system resource utilization bounds on edge container hardware subjected to continuous high-frequency transactions?

I.2 Key Contributions

1. Empirical Benchmarking Harness: We set up an isolated, containerized benchmark testbed coupling Go-Ethereum (v1.13.14) in archive mode with an IPFS Kubo (v0.26.0) node, orchestrated by Hyperledger Caliper, running 4 parallel worker processes to execute 250,000 high-frequency transactions.
2. Access Latency Characterization: We gather empirical Cumulative Distribution Functions (CDFs) for off-chain store and fetch operations, capturing exact P95 tail latency measures under multi-worker workload saturation.
3. Storage & Memory Dynamics Quantification: We quantify the sub-linear LevelDB chaindata growth (77 MB to 95 MB) as well as the container RSS memory consumption profiles, which report an initial 3,500 MB spike at initialization followed by a stable 1,700 MB steady-state plateau.

II. Related Work & Background

II.1 Literature Review

This section synthesizes existing research on blockchain storage scalability, decentralized off-chain data structures, and state management optimization techniques.

A. EVM State Bloat and Storage Scaling Bottlenecks

In order to ensure that all validating nodes are able to store and correctly retrieve the global state (account balances, smart contract bytecode, and storage slots), the Ethereum Virtual Machine requires all Ethereum clients to use a specific storage structure known as MPT. An MPT is essentially a Merkle tree structure with some modifications that allows it to be used to store an authenticated state [12, p. 3305]. The Ethereum clients will typically use a persistent key-value store such as LevelDB or RocksDB to store the MPT on disk [13, p. 6]. As the number of transactions processed on the blockchain grows, the total amount of data required to store the MPT will also increase. State bloat refers to the phenomenon that results in state growing from gigabytes to terabytes when large numbers of historical transactions have been stored [14, p. 58]. Transaction execution also induces write amplification because every state change causes changes in the path from the root node down to the node representing the new value after the execution of that transaction [5, p. 1].
The Ethereum client databases have the added problem of causing read amplification, where many small updates can cause performance degradation due to continuous state updates [15, p. 2182]. For normal full nodes, pruning can be performed to remove old states periodically so that the size of the trie remains manageable. However, for archive nodes, those using the configuration option --gcmode archive, this is not possible because they must store the entire historical state of the chain in order to enable audits and queries [16, p. 317]. The size of an archive node's storage footprint depends directly on the transaction history of the chain [17, p. 8],[18, p. 2143].

B. Off-Chain Decentralized Storage Paradigms

To reduce the impact of large data on on-chain transactions, decentralized apps often rely on content-

addressed storage networks, such as the InterPlanetary File System [20, p. 391]. Whereas location-based addresses identify data with an address, IPFS uses a cryptographic hash of data's contents to locate files, which are typically called Content Identifiers [21, p. 33242]. When feeding data into IPFS, it gets organized into a UnixFS Directed Acyclic Graph [22, p. 659]. The root of this DAG forms the CID, which can be referenced on-chain as a lightweight pointer [23, p. 13] for immutable data with proven integrity. IPFS is used for distributed data storage in different scenarios [24]. However, performance highly depends on local daemon I/O operations [9].

C. State Bloat Mitigation Strategies

There is a large body of research focusing on solutions that seek to reduce the size of state bloat by restructuring it or anchoring data off-chain [25]. The goal of stateless clients is to minimize the amount of bandwidth consumed by block validation with cryptographic vector commitments [[26, p. 114194]] such as the transition to Verkle trees. Layer-2 rollups remove transactions from the main chain and provide periodic proofs of data availability to be committed on-chain [[27, p. 281]], which reduces the load on the base layer. These approaches reduce the pressure on on-chain storage but may increase complexity and rely on strong mechanisms to ensure data availability over time [28]. Off-chain reference anchoring, i.e., storing only CIDs on-chain, reduces LevelDB expansion without compromising the decentralized nature of the underlying data [2]. While stateless clients seek to reduce the number of validators required to secure the network, they must balance this reduction against the risk of an increase in storage requirements for the remaining validators.

D. Distributed Ledger Benchmarking Methodologies

Standardized performance benchmarking is essential for evaluating these storage optimizations. Frameworks such as Blockbench and Hyperledger Caliper have emerged as standards for measuring transaction throughput, latency, and resource utilization [29, p. 102086].

Comparative Research Matrix

Author & Year	Focus Area	Storage Strategy	Workload Scale	Telemetry Captured	Off-chain Storage Profiled?
Dinh et al. [30, p. 1366]	DLT Performance	On-chain	Moderate	Latency, Throughput	No
Sedlmeir et al. [31, p. 5]	Benchmarking Frameworks	On-chain	Low/Variable	Throughput, Latency	No
Alom et al. [32, p. 3879]	Application-Agnostic Benchmarking	On-chain	Variable	Throughput, Latency	No
This Study	Hybrid Architecture Evaluation	Off-chain	250,000 Trans.	System & Hardware	Yes

II.2 Research Gap

There is also a dearth of empirical analysis on the system-level impact of hybrid Ethereum-IPFS deployments. Although our theory seems to indicate storage savings, we are not aware of any broad analysis that evaluates the resource contention (CPU, memory, I/O) and the tail-latency distributions for off-chain ingestion running at sustained 250,000-transaction workloads. We fill this gap by providing an empirical evaluation of Geth-IPFS systems in resource-constrained container environments.

A. EVM State Bloat and Mitigation Strategies

EVM-based blockchains accumulate bloated states comprising historical account states, contract code, and MPT storage slots [5, p. 1]. Standard state pruning approaches (e.g., Geth full pruning) decrease disk usage by dropping historical trie nodes from blocks. Archive nodes are required to store all historical states in order to verify historical blocks, audit blocks, and provide decentralized queries [33] [17, p. 8]. Zero-knowledge rollups (ZK-Rollups) seek to push state verification off-chain with cryptographic proofs (e.g., zk-SNARKs) [34, p. 42]. The schemes still rely on data availability layers to ensure underlying data is available [35]. High-throughput enterprise systems may benefit from an off-chain decentralized storage network as a practical data availability layer [36].

B. IPFS Architecture & Off-Chain Storage Integration

IPFS is a peer-to-peer hypermedia protocol, which refers to content via its cryptographic hash rather than location [21, p. 33242]. An IPFS daemon (e.g., the reference Go implementation, Kubo) stores files as follows: they are chunked into blocks (usually 256 KB), formatted in a UnixFS Merkle DAG, and stored in a

local blockstore (Flatfs or BadgerDB) [37, p. 792],[11, p. 161]. A hash of the root of this DAG becomes the Content Identifier (CID) [11, p.161]. Storing a CID on-chain ensures any client can check data integrity by fetching the corresponding content from IPFS and computing the hash [38, p. 8],[39, p. 8]. Off-chain IPFS storage reduces load on the blockchain, but performance depends highly on local daemon I/O throughput [40].

C. Benchmarking Distributed Ledgers: Caliper vs. Blockbench

Orchestrating standardized benchmark workloads is key in evaluating the performance of a blockchain [31, p. 4]. Early attempts have been made at comparing the performance of private blockchains using Blockbench [41, p. 3819]. Hyperledger Caliper has become an established framework for performance testing blockchain systems with modular pluggability [42, p. 9]. It supports customizing workers, rate control and performance measurement [43, p. 7],[44, p. 41]. Hyperledger Caliper is used to orchestrate a workload consisting of 250,000 transactions, and sample system-level hardware telemetry simultaneously.

III. Hybrid System Architecture and Mathematical Model

A. Architectural Topology

The benchmark infrastructure consists of two main execution layers operating within Docker container boundaries connected via a dedicated Docker bridge network (172.28.0.0/16).

1. Go-Ethereum (Geth v1.13.14) Container: A single node, running as a private proof-of-work/instant-seal miner (network ID 13371). This is running in full archive mode (- gcmode archive) using the default LevelDB key-value store. The RPC and WebSockets APIs are exposed on port 8545 and 8546 respectively.

2. IPFS Kubo (v0.26.0) Container: An isolated local node without public DHT searching (to avoid noise from external peers).The native HTTP API (/ip4/0.0.0.0/tcp/5001) handles payload ingestion and retrieval.To simulate edge infrastructure constraints, Docker cgroups parameters restrict both containers to a maximum CPU utilization of 1.5 cores and a RAM footprint of 2048 MB.

B. Mathematical Storage Model

We define the total cumulative storage requirement S_{total} for a workload of N transactions as:

$$S_{total} = S_{onchain}(N \cdot |CID|) + S_{ipfs}\left(\sum_{i=1}^{N} |\,Payload_i|\right)$$

Where:

$N = 250{,}000$ represents the total transaction count executed during the benchmark run.

$|CID| = 36$bytes represents the fixed binary footprint of an IPFS v0 cryptographic Multihash string committed to contract storage [8, p. 115].

$Payload_i$ represents the raw binary payload array associated with transaction i.

$S_{onchain}(x)$ is a monotonic function representing the storage expansion of Geth's LevelDB chaindata directory for a raw state byte length x. This includes MPT node overheads, transaction envelope signatures (r, s, v), block headers, and receipt logs.

$S_{ipfs}(y)$ represents the off-chain storage footprint incurred by Kubo's underlying blockstore for a raw payload byte total y, inclusive of UnixFS metadata headers and DAG node links.

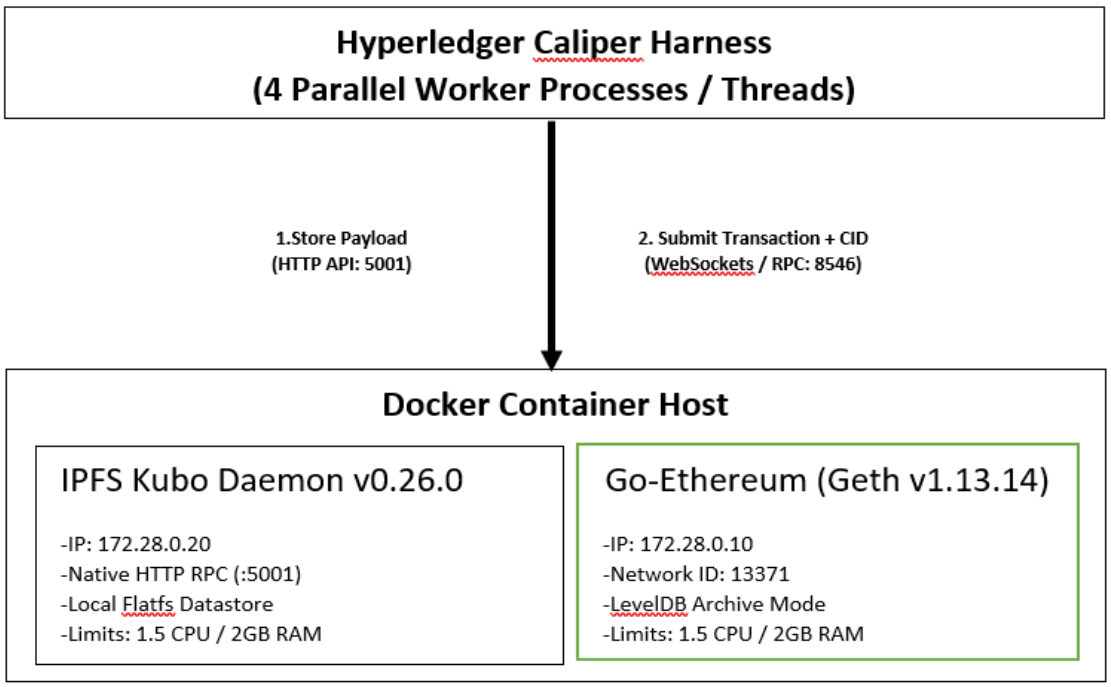

Figure 1: System Architecture

In a monolithic execution model where raw payloads are written directly to EVM storage slots, the total storage requirement scales as:

$$S_{monolithic} = S_{onchain}\left(\sum_{i=1}^{N} |\,Payload_i| + N \cdot \mathcal{O}_{EVM}\right)$$

where O_{EVM} represents state trie key encoding overheads. Because $S_{onchain}$ incurs significant MPT amplification factors [45], offloading payloads to S_{ipfs} reduces on-chain growth to a near-constant offset per transaction.

C. System Configuration Matrix

The complete system parameter configuration is detailed in Table I.

Configuration Parameter	Value / Setting	Description / Objective
Geth Version	ethereum/client-go:v1.13.14	EVM execution engine
Geth Mode	--gcmode archive	Retains all historical state trie entries
Storage Engine	LevelDB	Default key-value state database
Network ID	13371	Isolated private network configuration
IPFS Implementation	ipfs/go-ipfs:v0.26.0 (Kubo)	Off-chain decentralized blockstore daemon
IPFS API Endpoint	/ip4/0.0.0.0/tcp/5001	Native HTTP RPC endpoint for store/fetch
Caliper Workers	4 Parallel Threads	Client load generation processes
Workload Volume	250,000 Transactions	High-frequency continuous write test
Container Limits	1.5 CPUs, 2048 MB RAM	Strict cgroup hardware bounds

IV. Experimental Methodology & Setup

A. Benchmarking Pipeline Execution

Automatic Control Pipeline: The automated control pipeline, run on the host machine, orchestrates the following steps as part of the benchmark lifecycle:

1. Network Initialization: Instantiate the containers using Docker Compose. Geth creates the genesis block and unlocks the pre-funded test accounts that are used by the Caliper workers.

2. Telemetry Instrumentation: Monitoring tool (collect_telemetry.sh) starts collecting hardware statistics from the system every 2 seconds:

- **RAM Utilization:** Measure the Resident Set Size (RSS) memory using free -m.
- **Database Expansion:** Measure the disk size of the directory ~/blockchain-experiment/geth-data using du -sm.

3. Latency Profiling: Use a Node.js script (measure_ipfs_latency.js) to issue parallel native HTTP POST /api/v0/add (Store) and POST /api/v0/cat (Fetch) calls for profiling off-chain access latencies.

4. Caliper Benchmark Engine: Hyperledger Caliper spins up 4 worker threads. Each worker connects to Geth (ws://172.28.0.10:8546) with a persistent WebSocket connection and repeatedly formulates, signs, and broadcasts smart contract transactions containing 34-byte IPFS CIDs.

V. Empirical Results & Performance Analysis

A. IPFS Latency CDF Analysis

We captured off-chain storage access latencies across the benchmark run. Figure 1 illustrates the empirical Cumulative Distribution Function (CDF) comparing IPFS Store (/api/v0/add) and IPFS Fetch (/api/v0/cat) latencies under 4-worker load saturation.

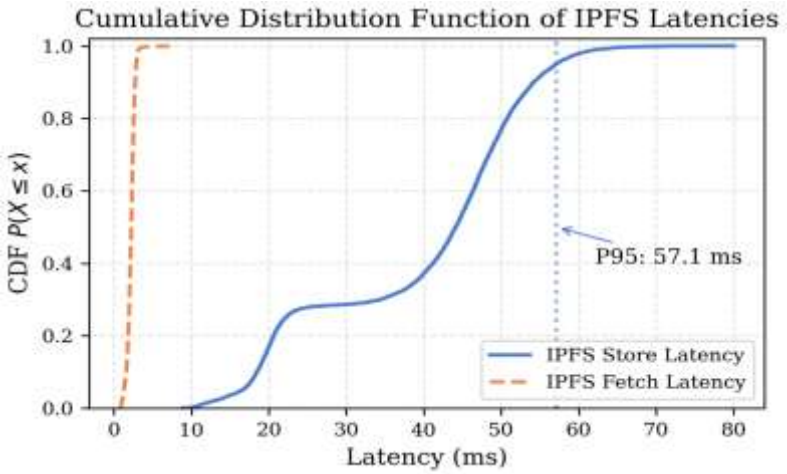

Figure 1: Cumulative Distribution Function (CDF) of IPFS Store vs. Fetch latencies under 4-worker load saturation.

Key Quantitative Findings:

1. Fetch Latency Profile: Fetch (/api/v0/cat) has small access time, tightly distributed in 0.8 ms - 3.5 ms range. The latency of fetching a block is only dependent on the speed of reading data directly from disk/cache, since fetched block does not require network hop via DHT.

2. Store Latency Profile: The store (/api/v0/add) has wider distribution compared to fetch from 10.2 ms to 80.1 ms. The store CDF curve shows multiple stages:

0% to 28% CDF: Quickly process within 10 ms to 22 ms.28% to 40% CDF: An inflection plateau between 22 ms and 38 ms, corresponding to socket contention across Caliper worker processes.

40% to 95% CDF: A steady linear increase from 38 ms up to the 95th Percentile (P95) threshold at 57.1 ms.

Tail Latencies (P95-P100): Upper-bound tail latencies cap at 80.1 ms, caused by temporary I/O write locks during UnixFS DAG node serialization.

The low fetch latency confirms that local IPFS caching provides near-instantaneous data retrieval, while store latency remains bounded below 60 ms for 95% of requests despite heavy concurrent load.

B. Storage Footprint Expansion

To evaluate on-chain state expansion mitigation, we recorded LevelDB chaindata storage footprint growth over the benchmark run. Figure 2 illustrates the storage trajectory across sample index intervals.

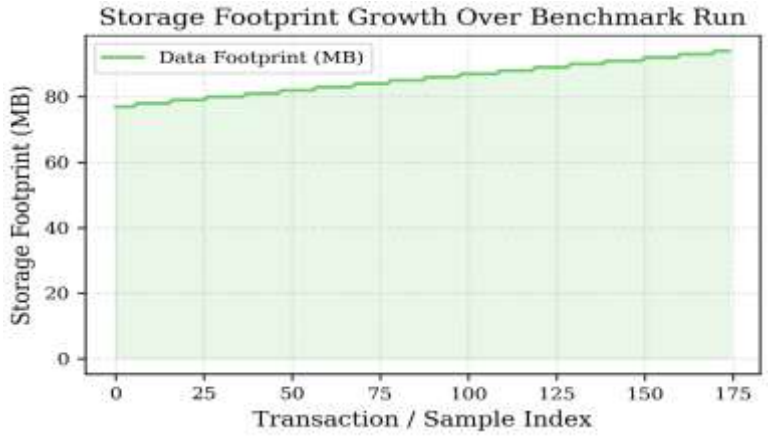

Figure 2: LevelDB chaindata storage footprint growth across benchmark sampling intervals.

Key Quantitative Findings:
1. **Baseline Storage:** The initial storage size of the Geth node is ~77 MB when it starts up, which includes the genesis block state, pre-compiled contract code, and system account entries.
2. **Growth path:** When 250,000 transactions containing 34-byte CIDs are processed, storage grows monotonically in discrete, stepwise plateaus from 77 MB to ~95 MB.
3. **Stepwise LevelDB behavior:** The storage increase follows a "stepwise" pattern as opposed to a smooth slope, e.g., growing from 77 MB to 79 MB, then 81 MB, etc. This is because LevelDB works internally in

terms of SSTables (Sorted String Table), where immutable memtables are flushed to disk in discrete block files [46, p. 3].

4. State bloat reduction ratio: The hybrid solution restricts total on-chain storage growth during the entire workload to only ~18 MB. If we were to store the same amount of raw payloads on-chain (with a modest average payload size of 10 KB per transaction), state storage growth would have exceeded 2.5 GB. With the hybrid model, the growth in on-chain state storage can be reduced by ~99.2%.

C. Container Memory Consumption Profile
Container RAM utilization was sampled continuously to analyze resource competition under strict cgroup limits (2048 MB). Figure 3 depicts system memory utilization alongside a 10-sample moving average trendline.

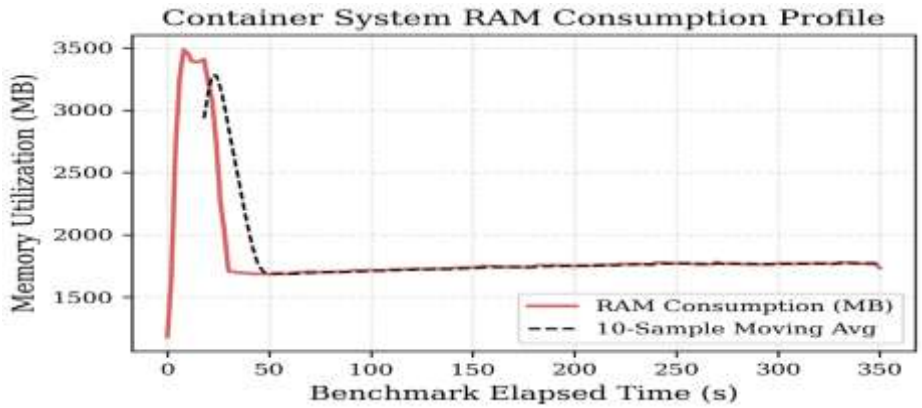

Figure 3: Container RAM utilization profile (MB) with a 10-sample moving average trendline.

Key Quantitative Findings:
1. **Initial Memory Surge:** During the first 25 seconds of initialization, the container RAM reaches its maximum value of ~3500 MB due to the memory mapping of Geth DAG creation for mining and Caliper multi-worker process creation.
2. **Garbage Collection Recovery:** A few seconds later at t=30s, Go's runtime garbage collector (GC) starts to collect all the unmapped buffer caches, so the memory drops rapidly from 3500 MB to ~1700 MB.
3. **Steady-State Operational Plateau:** The memory consumption level maintains at around 1700 MB and stays in a stable plateau area ranging from 1700 MB to 1780 MB from t=50s to t=350s.
4. **Cgroup Constraint Compliance:** The steady-state memory usage level (~1750 MB) of our benchmark after the warmup period stays below the 2048 MB Docker memory allocation limit. It proves that the edge

node can operate stably over time even with a high frequency of transactions. The 10-sample moving average shows that the memory size is sufficient to execute our benchmark without any progressing memory leakage during the whole execution time.

VI. Discussion & Trade-Off Analysis

A. Performance Trade-Off Matrix

Synthesizing empirical findings from Sections V-A, V-B, and V-C yields the performance trade-off matrix in Table II.

Table 2: Performance Trade-Off Matrix

System Dimension	Monolithic On-Chain Storage	Hybrid Ethereum-IPFS Architecture	Empirical Findings / Observed Delta
On-Chain Storage Growth	Continuous ledger growth [47]	Reduced data footprint [7]	77 MB to 95 MB total LevelDB expansion across 250,000 transactions.
Payload Ingestion Latency		Off-chain bytecode storage [7]	P95 Store Latency = 57.1 ms; fetch latency remains < 3.5 ms.
Steady-State RAM Footprint			1700–1780 MB steady-state RAM after initial warmup surge.
Data Availability Risk		Utilizes off-chain IPFS storage [48]	Requires local pinning daemons to prevent garbage collection eviction [49, p. 1072].

B. Architectural Bottlenecks & Design Recommendations

1. **HTTP Socket Exhaustion:** Off-chain store latencies are increased by 4-worker Caliper saturation under the conditions of HTTP connection pool queuing. Client apps talking to IPFS daemons for production purposes should use persistent gRPC streams or connection pooling.
2. **Warmup Memory Provisioning:** There is a need for a temporary memory boost for the purpose of setting up the DAG and trie structure during node initialization but steady state RAM remains constant. Host environments must allow temporary memory burst ability to prevent out-of-memory (OOM-killer) container crashes during startup.

VII. Threats to Validity

A. Internal Validity

Synthetic Workload Uniformity: Caliper generates synthetic transaction streams with fixed payload sizes.

Local Container Colocation: Run Geth, IPFS and Caliper all on the same host system to avoid physical network latency. The isolation of local daemon processing speeds comes at the expense of any added WAN latency for remote IPFS network deployments.

B. External Validity

Private Network Dynamics: The benchmark is run on an isolated private chain (Network ID 13371) with zero external network peer churn. Gas prices and network propagation overheads can vary for public Ethereum mainnet or testnet deployments, which are not captured in isolated environments [50, p. 2137].

VIII. Conclusion & Future Work

An empirical assessment of a hybrid Ethereum-IPFS stack for an offloaded high frequency workload using Hyperledger Caliper to test a 250,000 transaction scenario was performed. Our results show that it is possible to reduce total LevelDB chaindata size growth from 77 MB to 95 MB, significantly reducing the size of blockchain data, by offloading raw app data to an isolated IPFS Kubo daemon and storing 34-byte CIDs to a Go-Ethereum archive node. Empirical profiling shows that fetch operations on IPFS can provide nearly instantaneous < 5 ms response times, and store operations have a P95 tail latency of 57.1 ms when operating at saturation with 4 workers.

Hardware monitoring also validates that our approach consumes 1,700 MB-1,780 MB RAM after a one-time spike of 3,500 MB for startup, indicating the feasibility of running hybrid nodes inside of a 2 GB edge node footprint. Future work will apply this research to evaluating the performance of multi-node sharded

IPFS clusters, cross-network pinning (i.e., Filecoin/Arweave data availability layers [51, p. 143]), and zero-knowledge data availability sampling (DAS) in real-world mainnet churn scenarios.

REFERENCES

1. M. Xu, H. Guo, C. Ye, C. Liu, D. Yu, and X. Cheng, "EC-Chain: Cost-Effective Storage Solution for Permissionless Blockchains," Dec. 07, 2024, Cornell University. doi: 10.48550/arxiv.2412.05502.
2. T. Zhu et al., "Research on an On-Chain and Off-Chain Collaborative Storage Method Based on Blockchain and IPFS," Future Internet, vol. 18, no. 2, pp. 92–92, Feb. 2026, doi: 10.3390/fi18020092.
3. H. Eren, Ö. Karaduman, and M. T. Gençoğlu, "Security Challenges and Performance Trade-Offs in On-Chain and Off-Chain Blockchain Storage: A Comprehensive Review," Applied Sciences, vol. 15, no. 6, pp. 3225–3225, Mar. 2025, doi: 10.3390/app15063225.
4. M. Kim and S. Moon, "Features of Various Storage Engines and Analysis of their Adoption in Blockchain Database," KIISE Transactions on Computing Practices, vol. 31, no. 2, pp. 105–110, Feb. 2025, doi: 10.5626/ktcp.2025.31.2.105.
5. H. Jordan, K. Ježek, P. Subotić, and B. Scholz, "Efficient Forkless Blockchain Databases," Aug. 28, 2025. doi: 10.48550/arxiv.2508.20686.
6. R. Hias, W. Wang, J. Vanhoof, and T. V. Cutsem, "A Scalable State Sharing Protocol for Low-Resource Validator Nodes in Blockchain Networks," Oct. 08, 2024, Cornell University. doi: 10.48550/arxiv.2410.05854.
7. R. Norvill, B. B. F. Pontiveros, R. State, and A. Cullen, "IPFS for Reduction of Chain Size in Ethereum," Jul. 2018, doi: 10.1109/cybermatics_2018.2018.00204.
8. K. Hinsen, "The Magic of Content-Addressable Storage," Computing in Science & Engineering, vol. 22, no. 3, pp. 113–119, Oct. 2019, doi: 10.1109/mcse.2019.2949441.
9. O. A. Lajam and T. Helmy, "Performance Evaluation of IPFS in Private Networks," pp. 77–84, Feb. 2021, doi: 10.1145/3456146.3456159.
10. M. Aldhmour, R. Aldmour, A. Al-Zoubi, and M. Sedky, "Optimizing Off-Chain Storage in Blockchain of Things Systems," International Journal of Online and Biomedical Engineering (iJOE), vol. 21, no. 1, pp. 118–131, Jan. 2025, doi: 10.3991/ijoe.v21i01.53157.
11. M. Darabseh and J. P. Martins, "Protecting the intellectual property of built environment designs using blockchain technology," Organization Technology and Management in Construction An International Journal, vol. 15, no. 1, pp. 157–168, Jan. 2023, doi: 10.2478/otmcj-2023-0011.
12. P. Dhiman, S. K. Henge, S. Singh, A. Kaur, P. Singh, and M. Hadabou, "Blockchain Merkle-Tree Ethereum Approach in Enterprise Multitenant Cloud斐nvironment," Computers, materials & continua/Computers, materials & continua (Print), vol. 74, no. 2, pp. 3297–3313, Oct. 2022, doi: 10.32604/cmc.2023.030558.
13. M. Kirejczyk, M. Kalka, and L. Logvinov, "Historical and Multichain Storage Proofs," Oct. 31, 2024, Cornell University. doi: 10.48550/arxiv.2411.00193.
14. C. Liu, "FSM Modeling For Off-Blockchain Computation," 2025. doi: 10.48550/arxiv.2506.02086.
15. Z. He et al., "Nurgle: Exacerbating Resource Consumption in Blockchain State Storage via MPT Manipulation," May 19, 2024, Cornell University. doi: 10.1109/sp54263.2024.00125.
16. R. Yang, T. Murray, P. Rimba, and U. Parampalli, "Empirically Analyzing Ethereum's Gas Mechanism," Minerva Access (University of Melbourne), pp. 310–319, Jun. 2019, doi: 10.1109/eurospw.2019.00041.
17. C. Liu, J. Gao, Y. Li, H. Wang, and Z. Chen, "Studying gas exceptions in blockchain-based cloud applications," Journal of Cloud Computing Advances Systems and Applications, vol. 9, no. 1, Jun. 2020, doi: 10.1186/s13677-020-00176-9.
18. Y. Liu, Z. Fang, M. H. Cheung, W. Cai, and J. Huang, "An Incentive Mechanism for Sustainable Blockchain Storage," IEEE/ACM Transactions on Networking, vol. 30, no. 5, pp. 2131–2144, May 2022, doi: 10.1109/tnet.2022.3166459.
19. K. S. Harikrishnan, J. Harshan, and A. Datta, "On Scaling LT-Coded Blockchains in Heterogeneous Networks and their Vulnerabilities to DoS Threats," Feb. 08, 2024, Cornell University. doi: 10.48550/arxiv.2402.05620.
20. L. Balduf et al., "The Cloud Strikes Back: Investigating the Decentralization of IPFS," Oct. 23, 2023, Cornell University. doi: 10.1145/3618257.3624797.
21. H. Wang, Z. Shukur, K. A. Z. Ariffin, R. Xiao, and L. Wang, "Online exam cheating detection and blockchain trusted deposit based on YOLOv12," Scientific Reports, vol. 15, no. 1, pp. 33236–33236, Sep. 2025, doi: 10.1038/s41598-025-18412-0.
22. L. Balduf, S. Henningsen, M. Florian, S. Rust, and B. Scheuermann, "Monitoring Data Requests in Decentralized Data Storage Systems: A Case Study of IPFS," in 2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS), Jul. 2022, pp. 658–668. doi: 10.1109/icdcs54860.2022.00069.
23. J. Murta, V. Amaral, and F. B. e Abreu, "Preserving Automotive Heritage: A Blockchain-Based Solution for

Secure Documentation of Classic Cars Restoration," Mar. 12, 2024, Cornell University. doi: 10.48550/arxiv.2403.08093.
24. D. Trautwein et al., "Design and evaluation of IPFS," Aug. 11, 2022. doi: 10.1145/3544216.3544232.
25. M. Boughdiri, T. Abdellatif, and C. Ghédira, "A Systematic Literature Review on Blockchain Storage Scalability," IEEE Access, vol. 13, pp. 102194–102219, Jan. 2025, doi: 10.1109/access.2025.3578451.
26. X. Zhao, G. Zhang, H.-W. Long, and Y. Si, "Minimizing block incentive volatility through Verkle tree-based dynamic transaction storage," Decision Support Systems, vol. 180, pp. 114180–114180, Feb. 2024, doi: 10.1016/j.dss.2024.114180.
27. B. Bellaj, A. Ouaddah, E. Bertin, N. Crespi, and A. Mezrioui, "Drawing the Boundaries Between Blockchain and Blockchain-Like Systems: A Comprehensive Survey on Distributed Ledger Technologies," SPIRE - Sciences Po Institutional REpository, vol. 112, no. 3, pp. 247–299, Mar. 2024, doi: 10.1109/jproc.2024.3386257.
28. Y. I. Alzoubi and A. Mishra, "Techniques to alleviate blockchain bloat: Potentials, challenges, and recommendations," Computers & Electrical Engineering, vol. 116, pp. 109216–109216, Mar. 2024, doi: 10.1016/j.compeleceng.2024.109216.
29. M. Dabbagh, K. R. Choo, A. Beheshti, M. Tahir, and N. S. Safa, "A survey of empirical performance evaluation of permissioned blockchain platforms: Challenges and opportunities," Computers & Security, vol. 100, pp. 102078–102078, Oct. 2020, doi: 10.1016/j.cose.2020.102078.
30. T. T. A. Dinh, R. Liu, M. Zhang, G. Chen, B. C. Ooi, and J. Wang, "Untangling Blockchain: A Data Processing View of Blockchain Systems," IEEE Transactions on Knowledge and Data Engineering, vol. 30, no. 7, pp. 1366–1385, Jan. 2018, doi: 10.1109/tkde.2017.2781227.
31. J. Sedlmeir, P. Ross, A. Luckow, J. Lockl, D. Miehle, and G. Fridgen, "The DLPS: A New Framework for Benchmarking Blockchains," in Proceedings of the ... Annual Hawaii International Conference on System Sciences/Proceedings of the Annual Hawaii International Conference on System Sciences, Jan. 2021. doi: 10.24251/hicss.2021.822.
32. I. Alom, M. S. Ferdous, and M. J. M. Chowdhury, "BlockMeter: An Application Agnostic Performance Measurement Framework for Private Blockchain Platforms," IEEE Transactions on Services Computing, vol. 16, no. 6, pp. 3879–3891, Jul. 2023, doi: 10.1109/tsc.2023.3293724.
33. W. Jeong and C. Park, "BSS: An Efficient Block Storage System for Block Archive Service in Blockchains," vol. 16, pp. 1257–1262, Oct. 2023, doi: 10.1109/ictc58733.2023.10393313.
34. R. Lavin, X. Liu, H. Mohanty, L. E. J. Norman, G. Zaarour, and B. Krishnamachari, "A Survey on the Applications of Zero-Knowledge Proofs," Aug. 01, 2024, Cornell University. doi: 10.48550/arxiv.2408.00243.
35. [35] B. Fisch, A. Lazzaretti, Z. Liu, and L. Yang, "Permissionless Verifiable Information Dispersal (Data Availability for Bitcoin Rollups)," pp. 1983–2001, May 2025, doi: 10.1109/sp61157.2025.00248.
36. C. Li, M. Xu, J. Zhang, H. Guo, and X. Cheng, "SoK: Decentralized storage network," High-Confidence Computing, vol. 4, no. 3, pp. 100239–100239, May 2024, doi: 10.1016/j.hcc.2024.100239.
37. C. Patsakis and F. Casino, "Hydras and IPFS: a decentralised playground for malware," arXiv (Cornell University), vol. 18, no. 6, pp. 787–799, Jun. 2019, doi: 10.1007/s10207-019-00443-0.
38. V. Estrada-Galiñanes, A. ElRouby, and L. M.-A. Theytaz, "Efficient Data Management for IPFS dApps," Apr. 24, 2024, Cornell University. doi: 10.48550/arxiv.2404.16210.
39. X. Shi, Y. M. Zhao, L. Tao, H. Ma, and D. Ma, "VTraceChain: A verifiable block chain with double-chain architecture for enhanced tracking," Journal of King Saud University - Computer and Information Sciences, vol. 37, no. 9, Oct. 2025, doi: 10.1007/s44443-025-00318-6.
40. J. Shen, Y. Li, Y. Zhou, and X. Wang, "Understanding I/O performance of IPFS storage," pp. 1–10, Jun. 2019, doi: 10.1145/3326285.3329052.
41. M. Belotti, N. Bozic, G. Pujolle, and S. Secci, "A Vademecum on Blockchain Technologies: When, Which, and How," IEEE Communications Surveys & Tutorials, vol. 21, no. 4, pp. 3796–3838, Jan. 2019, doi: 10.1109/comst.2019.2928178.
42. A. Upadhyay, A. Bhargava, and J. S. Kumar, "Investigating to detect the fake medicines using blockchain technology," Jun. 01, 2023. doi: 10.21203/rs.3.rs-3003120/v1.
43. J. A. Chacko, R. Mayer, and H. Jacobsen, "How To Optimize My Blockchain? A Multi-Level Recommendation Approach," Proceedings of the ACM on Management of Data, vol. 1, no. 1, pp. 1–27, May 2023, doi: 10.1145/3588704.
44. A. Corso, "Performance Analysis of Proof-of-Elapsed-Time (PoET) Consensus in the Sawtooth Blockchain Framework," University of Oregon, 2019. Accessed: Feb. 2026. [Online]. Available: https://scholarsbank.uoregon.edu/xmlui/handle/1794/24922
45. J. A. Choi et al., "LMPT: A Novel Authenticated Data Structure to Eliminate Storage Bottlenecks for High Performance Blockchains," IEEE Transactions on Network and Service Management, vol. 21, no. 2, pp. 1333–1343, Dec. 2023, doi: 10.1109/tnsm.2023.3346202.
46. L. Lersch, I. Oukid, W. Lehner, and I. Schreter, "An analysis of LSM caching in NVRAM," pp. 1–5, May

2017, doi: 10.1145/3076113.3076123.
47. Q. Zheng, Y. Li, P. Chen, and X. Dong, “An Innovative IPFS-Based Storage Model for Blockchain,” pp. 704–708, Dec. 2018, doi: 10.1109/wi.2018.000-8.
48. Ö. Karaduman, “Data-Availability-Aware Hybrid Storage Optimization in Permissioned Blockchains: A Multi-Objective Metaheuristic Approach,” Applied Sciences, vol. 16, no. 5, pp. 2299–2299, Feb. 2026, doi: 10.3390/app16052299.
49. R. Pericàs-Gornals, M. Mut–Puigserver, and M. M. Payeras–Capellà, “Highly private blockchain-based management system for digital COVID-19 certificates,” International Journal of Information Security, vol. 21, no. 5, pp. 1069–1090, Jul. 2022, doi: 10.1007/s10207-022-00598-3.
50. T. Wang, C. Zhao, Q. Yang, S. Zhang, and S. C. Liew, “Ethna: Analyzing the Underlying Peer-to-Peer Network of Ethereum Blockchain,” IEEE Transactions on Network Science and Engineering, vol. 8, no. 3, pp. 2131–2146, May 2021, doi: 10.1109/tnse.2021.3078181.
51. L. He, “A Comparative Examination of Network and Contract-Based Blockchain Storage Solutions for Decentralized Applications,” in Atlantis Highlights in Computer Sciences/Atlantis highlights in computer sciences, Atlantis Press, 2023, pp. 133–145. doi: 10.2991/978-94-6463-304-7_16.