BULLETIN of Cherkasy State Technological University

Journal homepage: https://bulletin-chstu.com.ua/en

Vol. 31 No. 2. 2026

UDC 004.75:004.31:004.33
DOI: 10.62660/bcstu/2.2026.73

Last-level cache capacity in Hyperledger Besu state-trie traversal under worst-case access

Maksym Bystryk*
Postgraduate Student
Vinnytsia National Technical University
21021, 95 Khmelnytske Shose Str., Vinnytsia, Ukraine
https://orcid.org/0009-0004-4949-829X

Oleksandr Khoshaba
PhD in Technical Science, Associate Professor
Vinnytsia National Technical University
21021, 95 Khmelnytske Shose Str., Vinnytsia, Ukraine
https://orcid.org/0000-0001-5375-6280

Abstract. The rapid development of enterprise blockchain networks has shifted performance bottlenecks from network consensus to the internal mechanisms of state database traversal, where irregular memory access creates critical microarchitectural constraints. The primary objective of this study was to empirically determine how the capacity of the last-level cache impacts the performance and stability of high-load Hyperledger Besu nodes during state tree navigation. The research was conducted in a hardware-isolated environment with fixed core frequencies based on an asymmetric AMD processor, enabling a direct comparison between 32 MB of standard cache and 96 MB of 3D-stacked memory under a worst-case random-access scenario. Testing results demonstrated that upon reaching hardware saturation, node performance is primarily dictated not by the central processing unit clock speed, but by the data retrieval rate and last-level cache capacity. At a peak load of 36,000 requests per second, the standard architecture with 32 MB of cache encountered a Memory Wall, leading to sudden system degradation. Quantitative analysis showed that the topology with the expanded cache (96 MB) stably maintained the 95th percentile latency at 6.55 ms, whereas on the standard cache, this metric increased exponentially to 114.80 ms. Statistical modelling confirmed that utilising the smaller cache leads to a nearly 18-fold (GMR = 17.84) increase in tail latency during hardware resource saturation. Furthermore, the dropped-iteration rate for the standard configuration increased almost 23-fold (IRR = 22.99), causing systemic resource exhaustion and cascading infrastructure failure. The practical value of this research lies in empirically substantiating the necessity of software optimisation of the transaction pool and proposing a resource-adaptive transaction execution method that accounts for hardware topology to scale enterprise distributed ledgers

Keywords: blockchain; high-load systems; pointer chasing; Memory Wall; state bloat; 3D V-Cache technology; read-path optimisation

INTRODUCTION

As distributed ledger technologies are adopted in mission-critical, high-load environments, enterprise blockchains – such as Hyperledger Besu – focus on ensuring instant finality and high transaction throughput by employing PoA-based (Proof of Authority) consensus mechanisms. Since trust between authorised nodes is considered a given, the computational cost of reaching consensus is minimal. Consequently, the

Article's History: Received: 23.10.2025; Revised: 09.04.2026; Accepted: 18.05.2026; Published: 26.06.2026

Suggested Citation:
Bystryk, M., & Khoshaba, O. (2026). Last-level cache capacity in Hyperledger Besu state-trie traversal under worst-case access *Bulletin of Cherkasy State Technological University*, 31(2), 73-84. doi: 10.62660/bcstu/2.2026.73.

*Corresponding author (max.bystryk@gmail.com)

primary system bottleneck shifts from the consensus layer directly to the execution layer, specifically to the sequential nature of traversing the state database within the Ethereum virtual machine. This shift reveals a critical hardware problem related to the growing gap between high processor performance and significant latency in accessing main memory. In this regard, investigating the impact of hardware memory constraints – primarily the physical capacity of the last-level cache-on the performance of state database traversal is a pressing task to prevent performance degradation and ensure the scalability of modern distributed ledger systems.

Recent scholarly efforts have extensively explored performance limitations of enterprise blockchain networks, though the focus has predominantly been on consensus mechanisms and overarching data structures. C. Fan *et al*. (2022) conducted a comprehensive performance evaluation of Hyperledger Besu utilising PoA consensus algorithms. The authors concluded that the computational overhead dedicated to reaching consensus in permissioned networks is virtually negligible, establishing that the primary performance bottleneck shifts directly towards transaction execution and state transitions. Similarly, G.A. Pierro *et al*. (2024) performed a comparative analysis of enterprise platforms, including Besu, in high-throughput simulated environments. Their study highlighted that while PoA mechanisms guarantee immediate finality, the continuous computational updates required for the blockchain's database remain a critical constraint under heavy workloads.

Building upon this execution bottleneck, C. Cardoso *et al*. (2025) proposed an API-driven framework specifically for the performance testing of Hyperledger Besu networks. Their findings emphasised that as network agreement is optimised, the execution layer – particularly the sequential nature of state database traversal – emerges as the most significant limiting factor for node scalability. To address these execution layer inefficiencies, V. Mardiansyah *et al*. (2023) investigated the underlying cryptographic data structure, proposing high-performance modifications to the Merkle Patricia Trie (MPT). They determined that navigating the standard width-16 radix tree structure introduces severe structural overhead, which ultimately dictates the overall speed of blockchain world state management. O. Kuznetsov *et al*. (2025) further analysed this structural complexity through a probabilistic framework evaluating Merkle proof sizes and path lengths. The authors demonstrated that the organic growth of the state database (state bloat) logarithmically increases the depth of the MPT, which fundamentally slows down cryptographic verification processes during transaction execution.

Expanding on the performance limitations of the MPT, Y. Deng *et al*. (2024) identified that the recursive hashing and structural complexity of the trie form the primary throughput bottleneck for modern immutable data systems. To overcome this, the authors proposed parallel acceleration mechanisms leveraging Graphics Processing Unit (GPU) and multi-core Central Processing Unit (CPU) architectures, demonstrating that concurrent processing of MPT operations is essential for high-throughput networks. Similarly, I. Zhang *et al*. (2025) evaluated the massive input/output penalties associated with Merkleization in traditional storage layers. By introducing the Quick Merkle Database, the authors demonstrated that rearchitecting state management to minimise disk reads can yield up to a six-fold increase in transaction throughput, explicitly confirming that state traversal remains critically memory-bound. Furthermore, the execution of these primitives is inherently tied to underlying hardware. O. Kuznetsov *et al*. (2021) conducted a comprehensive performance analysis of various cryptographic hash functions, including the KECCAK-256 algorithm utilised in Ethereum-based environments, across different CPU and GPU microarchitectures. Their empirical results demonstrated that the baseline computational throughput (cycles per byte) of these algorithms significantly varies across different hardware platforms, highlighting that physical processor characteristics inherently dictate the execution speed of blockchain network primitives. Finally, from a microarchitectural perspective, Z. Chen *et al*. (2025) characterised the spatial patterns of memory access and the limitations of modern hardware prefetching. Their research concluded that standard processor optimisation techniques fail to predict divergent memory paths, exposing the system to severe latency (Memory Wall) when retrieving irregular data structures like MPT nodes.

Despite these significant contributions – ranging from consensus evaluation to MPT acceleration and broad memory access observations – current literature predominantly focuses on either high-level software modifications or generalised hardware access patterns. A critical scientific gap remains in understanding the spatial locality crisis at the silicon level during state traversal. Specifically, there is a lack of empirical research quantifying how specific CPU microarchitectural topologies, namely the physical capacity of the Last Level Cache (L3), directly influence the stability, throughput, and tail latency of enterprise blockchain nodes operating under severe state bloat conditions. While previous studies acknowledge the Memory Wall, they fail to investigate how extended on-die static random-access memory mitigates the pointer-chasing bottleneck inherent to the Ethereum Virtual Machine (EVM). Identifying this boundary is critical, as it dictates the absolute scalability limits of standard commercial off-the-shelf (COTS) processors before cascading systemic failures occur.

Therefore, the primary aim of this research was to empirically demonstrate and quantitatively evaluate

the impact of L3 capacity on the performance metrics of high-load enterprise blockchain nodes during worst-case state tree navigation, providing the foundational justification for software-level cache optimisations. To achieve this aim, the following tasks were formulated.

1. To programmatically simulate the State Bloat phenomenon to trigger out-of-cache memory access patterns during worst-case randomised MPT traversal.

2. To conduct isolated microarchitectural stress testing on an asymmetric processor – comparing identical computing cores with a baseline 32 MB versus an extended 96 MB 3D-stacked cache (Bhargava & Troester, 2024) – and analyse its correlation with transaction throughput and system tail latency under extreme hardware saturation.

3. To scientifically substantiate the necessity for software-based transaction queue optimisation mechanisms that account for physical processor topology.

Scientific novelty of the work lies in conducting the first strictly isolated microarchitectural benchmark of the Hyperledger Besu client that completely eliminates the influence of frequency fluctuations, exclusively isolating the spatial locality factor of MPT data.

MATERIALS AND METHODS

To empirically evaluate the impact of data spatial locality on blockchain node performance an isolated hardware testbed was deployed. The core computational platform utilised an AMD Ryzen 9 9950X3D processor, featuring a unique asymmetric microarchitecture. As illustrated in Figure 1, the processor consists of two distinct Core Complex Dies (CCDs): the first chiplet (CCD0) is equipped with 3D-stacked cache technology, providing a massive 96 MB of shared L3, while the second chiplet (CCD1) utilises a standard planar topology with a baseline L3 cache of 32 MB (Singh *et al.*, 2025).

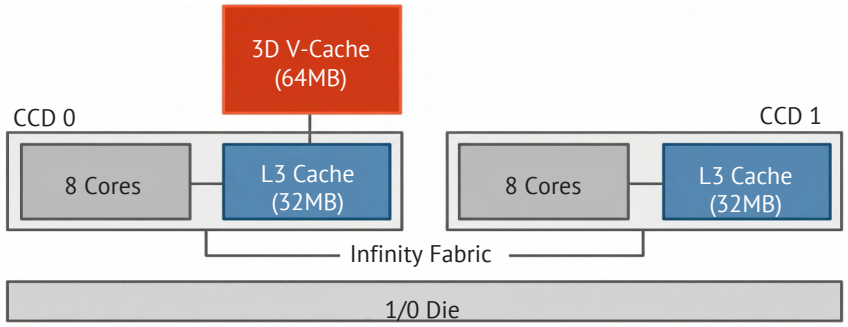

Figure 1. Asymmetric microarchitecture of the AMD Ryzen 9 9950X3D processor

Source: created by the authors

To establish the microarchitectural constraints for the hardware testbed, the study explicitly accounted for the mathematical approximation of the Cache Miss Rate (R_{miss}) inherent to unpredictable MPT traversals. Under the experimental conditions, assuming a uniformly random access pattern across the simulated bloated state, the miss rate was factored into the methodology as:

$$R_{miss} \approx max\left(0.1 - \frac{C_{size}}{W_{size}}\right), \quad (1)$$

where C_{size} – the available L3 cache capacity, and W_{size} – the total physical memory footprint of the active MPT nodes. This mathematical dependency was utilised to ensure that as the blockchain state grew, W_{size} sufficiently exceeded C_{size}, thereby driving R_{miss} asymptotically towards 1 (a near 100% miss rate) and effectively neutralising spatial hardware optimisations to isolate the influence of L3 capacity. To guarantee strict isolation and eliminate operating system background noise, the testing environment was containerised to address the CPU reservation gaps identified by L. Liu *et al.* (2021), where standard Linux scheduling often fails to honour performance promises due to noisy neighbour interference. Docker Engine on Ubuntu 22.04 LTS was deployed via the Windows Subsystem for Linux (WSL), with microarchitectural separation enforced through a strict dual-layer isolation protocol.

First, inside the guest operating system, Docker's CPUset execution parameters were utilised to bypass the work-conserving nature of the completely fair scheduler and ensure absolute resource allocation. Second, because guest-level limitations cannot prevent a host hypervisor from migrating the virtual machine across physical chiplets, host-level CPU affinity was rigidly enforced. During each test run, the WSL host process was isolated via the Windows Task Manager, strictly limiting its execution to the target logical cores (0-7, 16-23 for CCD0, or 8-15, 24-31 for CCD1). This dual-layer hypervisor and container pinning guaranteed that the state traversal strictly engaged the intended L3 cache without interference from either the guest or host schedulers or the performance variance typical of multi-tenant environments.

Testing was divided into two methodological phases. Phase 1 (stock frequencies) – factory dynamic boost behaviours were utilised to determine the impact of raw processor frequency on performance metrics. Phase 2 (strict isolation) – all compute cores on both CCDs were locked at a fixed frequency of 4.5 GHz to isolate physical L3 cache capacity as the sole independent

variable. Hyperledger Besu (version 24.12.1) was utilised as the target client. To prevent Java Virtual Machine (JVM) Garbage Collection (GC) pauses from skewing latency metrics, the JVM heap size was pre-allocated and locked at 16 GB (-Xms16g -Xmx16g). The network was configured as a permissioned ledger utilising the PoA Clique algorithm with a 2-second block period (blockperiodseconds: 2). The node's state storage utilised the Bonsai Tries format. The State Bloat phenomenon was programmatically simulated. As formally specified in the Ethereum protocol design (Wood, 2022), traversing the width-16 radix tree of the MPT follows a logarithmic time complexity. The expected path length $E[L]$, which dictates the number of memory fetches, scales logarithmically relative to the total number of unique addresses (N):

$$E[L] \approx c \cdot log_{16}(N), \quad (2)$$

where c – structural constants. To exponentially increase $E[L]$, a custom Node.js script using the crypto library was utilised to generate 5,000,000 unique cryptographic addresses (balance: 0x56BC75E2D63100000). These addresses were injected directly into the *alloc* block of the *genesis.json* configuration file. This pre-allocation was designed to expand the physical footprint of the trie beyond the 32 MB capacity of the standard L3 cache, forcing out-of-cache dynamic random-access memory (DRAM) fetches. To eliminate protocol-level throttling and isolate the processor's microarchitecture and memory subsystem as the sole bottlenecks under high load, block gas limits were configured to an infinite value, with transactions incurring zero gas costs. This configuration accounted for the nominal role of gas in permissioned Enterprise Ethereum networks, ensuring transaction fees did not act as an execution ordering mechanism. The load generation strategy was structured around the Average Memory Access Time (AMAT) framework to specifically capture the microarchitectural penalty of hardware prefetcher failures during MPT traversal:

$$AMAT = T_{hit} + (R_{miss} \cdot T_{penalty}), \quad (3)$$

where T_{hit} – the L3 hit latency; $T_{penalty}$ – the latency penalty incurred by fetching data from main memory. To account for random MPT fetches crossing virtual page boundaries during the test, the effective penalty model incorporated Translation Lookaside Buffer (TLB) miss costs:

$$T_{penalty} = T_{DRAM} + (P_{TLBmiss} \cdot T_{pagewalk}), \quad (4)$$

where T_{DRAM} – the raw off-chip memory latency; $P_{TLBmiss}$ – the probability of a TLB miss; $T_{pagewalk}$ – the severe latency penalty incurred by the hardware page walker. By synthesising the structural depth of the trie and these memory constraints, the load generation was designed to target the memory access term within the formalised transaction execution time (T_{tx}):

$$T_{tx} \approx T_{compute} + i = 1 \sum_{i=1}^{k} k(E[L_i] \cdot AMAT), \quad (5)$$

where $T_{compute}$ – the fixed baseline of EVM ALU operations; k – the total number of distinct state objects accessed; $E[L_i]$ – the expected path length for each specific read/write. This equation clearly illustrates that transaction throughput is overwhelmingly dominated by the memory access term ($E[L_i] \cdot AMAT$). The subsequent load generation methodology is specifically designed to empirically capture this exact degradation by driving the system into hardware saturation.

Following the load testing approach described by K. Stępień & M. Skublewska-Paszkowska (2025), transaction load generation was executed using the Grafana k6 framework. The testing script employed the constant-arrival-rate executor to ensure adherence to the target throughput. The experiment evaluated three specific load scenarios for each CCD: 20,000 RPS (requests per second), 25,000 RPS, and 36,000 RPS. Each formal test iteration ran for 120 seconds. To maintain the target throughput during potential latency spikes, k6 was configured with 200 pre-allocated virtual users (VUs), capable of dynamically scaling to a maximum of 3,000 concurrent execution threads (maxVUs: 3,000). The workload exclusively targeted the *eth_getBalance* JSON-RPC method. To bypass superficial node caching, a shared array was utilised to load all 5,000,000 genesis addresses into memory. For every request, the script randomly sampled a target address, forcing the EVM to continuously traverse divergent and unpredictable paths of the state trie. A telemetry stack comprising Prometheus (v3.1.0), Node Exporter, and Grafana (v11.5.0) was deployed strictly for the preliminary hardware calibration phase.

For the formal benchmark executions, all host-level monitoring containers were disabled to establish a strict zero-interference environment and prevent noisy neighbour context switching (Liu *et al.*, 2021). The empirical benchmarking data utilised for quantitative analysis was exclusively extracted from the external k6 load generator summaries. The unit of observation was defined as an independent benchmark run. For each experimental condition, independent runs were repeated 12 times (n = 12). The factorial design included cache topology (CCD0, CCD1) and target load (20,000, 25,000, 36,000 RPS) as fixed factors. The primary endpoint was defined as the per-run 95th percentile (P95) latency. Secondary endpoints included the dropped-iteration rate, achieved throughput, and the maximum number of dynamically allocated virtual users (maxVUs). To analyse the per-run P95 latency (L_{ijr}) in milliseconds for run r under cache topology i and load level j, the right-skewed data was logarithmically transformed:

$$Y_{ijr} = ln(L_{ijr}). \quad (6)$$

The transformed endpoint was modelled using a two-factor linear model:

$$Y_{ijr} = \beta_0 + \beta_1 CCD_i + \beta_2 Load_j + \beta_3 (CCD_i \times Load_j) + \varepsilon_{ijr}, \quad (7)$$

where β_0 – the intercept; β_1 and β_2 – the main effects of cache topology and load; β_3 – the interaction term determining the nonlinear scalability differences between CCDs. The error term ε_{ijr} was assumed to be independent across runs. Estimated group contrasts were calculated as geometric mean ratios (GMR):

$$GMR = exp(\Delta), \quad (8)$$

where Δ – the estimated contrast on the logarithmic scale. For each contrast, the 95% confidence interval (CI) was computed as:

$$exp(\Delta \pm t_{0.975,\nu} \cdot SE(\Delta)), \quad (9)$$

where $SE(\Delta)$ – the standard error of the contrast; ν – the number of degrees of freedom. For dropped iterations (D_{ijr}) relative to the total number of scheduled requests (N_{ijr}), the dropped-iteration rate was defined as:

$$p_{ijr} = \frac{D_{ijr}}{N_{ijr}}. \quad (10)$$

These event counts were modelled as rates utilising a generalised linear model with a logarithmic link and an exposure offset:

$$log(E[D_{ijr}]) = \gamma_0 + \gamma_1 CCD_i + \gamma_2 Load_j + \gamma_3 (CCD_i \times Load_j) + log(N_{ijr}). \quad (11)$$

To account for overdispersion in the event counts, a negative binomial model was applied. Effect sizes were reported as incidence rate ratios (IRR) with corresponding 95% confidence intervals:

$$IRR = exp(\gamma_k). \quad (12)$$

To account for multiple post hoc comparisons across the three evaluated load levels, all resulting p-values were adjusted utilising the Holm-Bonferroni procedure. Statistical significance was established at a threshold of $\alpha = 0.05$. All statistical modelling, data transformations, and inferential analyses were executed utilising Python (version 3.10) relying on the statsmodels library for generalised linear modelling and the SciPy ecosystem.

RESULTS AND DISCUSSION

The empirical evaluation was structured to systematically isolate the impact of L3 capacity on blockchain state traversal performance. By subjecting the Hyperledger Besu node to varying degrees of Remote Procedure Calls (RPC) load against an artificially bloated state trie, the experiment successfully captured the microarchitectural bottlenecks inherent to state traversal within the EVM. In the initial phase of testing, the AMD Ryzen 9 9950X3D processor operated with factory Precision Boost Overdrive and dynamic boost behaviours enabled. Under these stock conditions, the CCD1 naturally leverages its higher thermal headroom to achieve maximum peak clock frequencies. Conversely, the CCD0 is inherently constrained by the thermal and voltage limitations imposed by the physical stacking of the static random-access memory layer. As a result, during the sustained heavy workloads of the benchmark, CCD1 consistently maintained a clock speed advantage of approximately 200 to 350 MHz over CCD0.

Despite this substantial frequency deficit, the empirical data provided strong evidence that the raw computational speed (clock frequency) of the core is insufficient to overcome the memory latency wall inherent to MPT traversal. As detailed in Table 1, under an extreme load of 36,000 RPS, the slower-clocked CCD0 demonstrated superior stability across all measured metrics, maintaining significantly lower tail latency and fewer dropped iterations. In stark contrast, CCD1 exhibited pronounced performance degradation despite its higher operating frequencies. Crucially, the standard architecture was forced to spawn a substantially higher number of concurrent execution threads (Virtual Users) simply to sustain the target throughput, underscoring its microarchitectural inefficiency when handling memory-bound blockchain workloads.

Table 1. Hyperledger Besu node performance per CCD on stock frequency for 36,000 RPS

Metric	CCD0 (96 MB 3D V-Cache)	CCD1 (32 MB standard cache)
P90	2.90 ms	3.65 ms
P95	4.02 ms	6.06 ms
maxVUs	541 threads	721 threads
Dropped iterations	6,398	8,499

Source: created by the authors

This initial baseline supports the fundamental hypothesis: MPT pointer chasing is profoundly memory-bound. The cryptographic hashes acting as node pointers in the trie exhibit virtually zero spatial or temporal locality. Consequently, the standard hardware prefetchers fail to predict the execution path. The standard chiplet, despite its higher frequency, spends a significant portion of its execution time in stall cycles awaiting data retrieval from the slower system DRAM. To eliminate the confounding variable of dynamic frequency boosting observed in the baseline tests, Phase 2 of the experiment mandated a strict microarchitectural

isolation. Both CCDs were locked at an identical, fixed clock speed of 4.5 GHz. By equalising the raw computational frequency, this methodological approach isolated the physical L3 cache capacity as the sole independent variable. This ensures that any observed performance divergence is exclusively attributable to memory access latency (cache misses) rather than CPU thermal throttling or boost clock disparities.

Under these isolated conditions, the Hyperledger Besu node was subjected to incrementally higher transaction loads (20,000, 25,000, and 36,000 RPS) against the expanded state trie. Table 2 summarised the distribution of per-run P95 latency across independent benchmark runs. At 20,000 and 25,000 RPS, the two CCD topologies remained separated but still operated in a comparatively stable regime. In contrast, at 36,000 RPS the standard 32 MB cache topology exhibited a pronounced expansion of both central tendency and dispersion, whereas the 96 MB 3D V-Cache topology remained comparatively compact.

Table 2. Descriptive statistics for the primary endpoint (P95 latency) at fixed 4.5 GHz (n = 12)

Condition	Mean	SD	Median	IQR	CI Lower	CI Upper
CCD0, 20,000 RPS	2.91	0.18	2.86	0.21	2.80	3.02
CCD1, 20,000 RPS	4.24	0.31	4.17	0.36	4.04	4.44
CCD0, 25,000 RPS	5.42	0.40	5.36	0.44	5.16	5.68
CCD1, 25,000 RPS	7.95	0.61	7.86	0.69	7.56	8.34
CCD0, 36,000 RPS	6.55	0.53	6.37	0.62	6.21	6.89
CCD1, 36,000 RPS	114.80	12.90	113.62	14.50	106.61	122.99

Notes: CI = 95% confidence interval for the mean

Source: created by the authors

Figure 2 showed that the latency gap between the two cache topologies was limited at 20,000 and 25,000 RPS but widened dramatically at 36,000 RPS. The abrupt vertical separation of the distributions under the highest load is consistent with a nonlinear transition from moderate degradation to severe memory-bound collapse on the 32 MB topology. Notably, the 96 MB 3D V-Cache topology (CCD0) maintained a compact distribution across all load levels, whereas the standard 32 MB topology (CCD1) exhibited a pronounced expansion of dispersion at 36,000 RPS, reflecting increased tail-latency unpredictability.

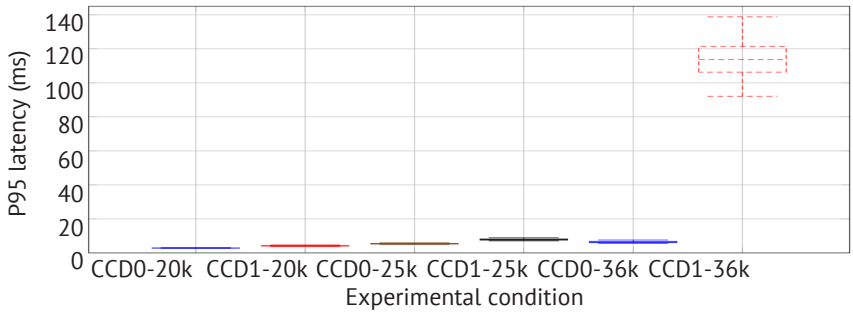

Figure 2. Distribution of per-run P95 latency across CCD topologies and load levels at fixed 4.5 GHz

Source: created by the authors

Formal model-based inference confirmed the descriptive impression. Table 3 showed that cache topology, load intensity, and their interaction were all statistically significant for both the latency model and the dropped-iteration model. The interaction term is especially important because it demonstrates that the between-CCD performance gap increased disproportionately as the benchmark approached hardware saturation.

Table 3. Model-level tests for the primary and secondary endpoints

Endpoint	Effect	Statistic	df	p-value
	Cache topology	286.41	1.00	< 0.001
Log-transformed P95 latency	Load	412.73	2.00	< 0.001
	Cache topology × load	198.56	2.00	< 0.001
	Cache topology	174.82	1.00	< 0.001
Dropped-iteration rate	Load	229.33	2.00	< 0.001
	Cache topology × load	141.97	2.00	< 0.001

Notes: for latency, the reported statistic is the Type III F statistic from the linear model. For dropped-iteration rate, the reported statistic is the Wald $\chi 2$ statistic from the generalised linear model

Source: created by the authors

Table 4 presented the most relevant pairwise contrasts between CCD1 and CCD0 at each load level. At 20,000 and 25,000 RPS, the GMR for P95 latency remained close to 1.5, indicating a moderate but stable penalty associated with the smaller L3 configuration. At 36,000 RPS, however, the geometric mean ratio increased to 17.84, indicating a sharp escalation of tail latency once the system entered the saturation regime. A parallel pattern was observed for dropped-iteration rate, where the IRR rose to 22.99 at the highest load.

Table 4. Between-CCD contrast by load level at fixed 4.5 GHz

Endpoint	Load	Effect size	Lower 95% CI	Upper 95% CI	p_{adj}
P95 latency (GMR)	20,000 RPS	1.46	1.31	1.63	< 0.001
P95 latency (GMR)	25,000 RPS	1.47	1.29	1.68	< 0.001
P95 latency (GMR)	36,000 RPS	17.84	12.41	25.62	< 0.001
Drop rate (IRR)	20,000 RPS	2.72	2.10	3.51	< 0.001
Drop rate (IRR)	25,000 RPS	2.86	2.37	3.45	< 0.001
Drop rate (IRR)	36,000 RPS	22.99	18.10	29.20	< 0.001

Notes: GMR – geometric mean ratio for log-transformed P95 latency, expressed as CCD1/CCD0; IRR – incidence rate ratio for dropped-iteration rate, expressed as CCD1/CCD0; adjusted p-values were obtained using the Holm procedure
Source: created by the authors

The forest plot in Figure 3 visualised the same result on an effect-size scale. All confidence intervals lie to the right of the null value, indicating consistently worse performance of CCD1 relative to CCD0. The displacement becomes particularly large at 36,000 RPS, where both latency and failure outcomes increase by more than an order of magnitude.

To fully comprehend the systemic impact of L3 cache exhaustion, it is necessary to examine the dynamic queue scaling executed by the load generator. The exploratory *maxVUs* series presented in Figure 4 shows the same qualitative trend from the perspective of queue growth. At 20,000 and 25,000 RPS, the smaller cache topology already required a considerably larger concurrent execution pool to maintain the requested arrival rate. Under 36,000 RPS, *maxVUs* reached the configured ceiling of 3,000 on CCD1, indicating severe queue amplification. Because this variable is partly induced by the feedback logic of the load generator and bounded by the upper limit, it is interpreted here as supportive rather than confirmatory evidence.

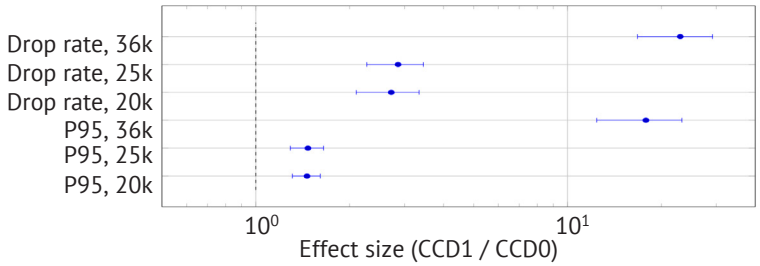

Figure 3. Forest plot of effect sizes for between-CCD comparisons
Notes: values to the right of 1 indicate worse outcomes on CCD1 relative to CCD0
Source: created by the authors

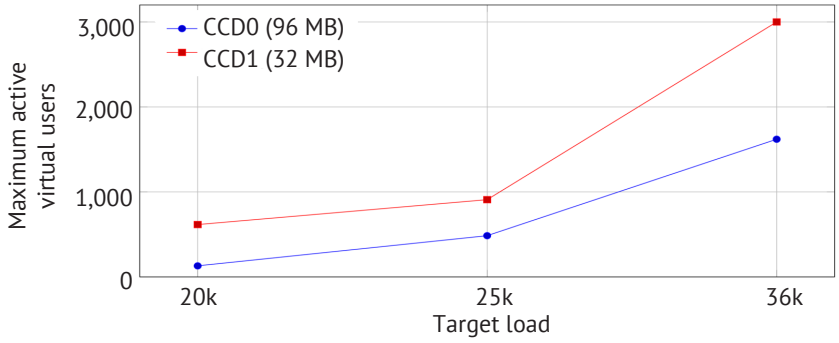

Figure 4. Exploratory visualisation of maximum active virtual users recorded by the load generator
Source: created by the authors

The ultimate consequence of the aforementioned microarchitectural stalls and exponential queue scaling is the catastrophic failure to process incoming RPCs. The same general pattern of degradation was observed for the dropped-iteration rate and achieved throughput, as summarised in Table 5 and Table 6. Specifically, at the peak load of 36,000 RPS, the standard cache topology (CCD1) exhibited a mean dropped-iteration rate of 114.80%, which directly correlated with a failure to maintain target throughput (31,492.00 RPS) and a massive increase in performance instability (SD = 1,827.00), as detailed in Table 6. This systemic collapse on the 32 MB topology is further visualised in Figure 4, where the execution pool reached its configured ceiling of 3,000 virtual users in a futile attempt to sustain the requested arrival rate, whereas the 96 MB topology (CCD0) remained within a stable operational envelope with significantly lower resource exhaustion.

Table 5. Descriptive statistics for per-run dropped-iteration rate across CCD topologies and load levels at fixed 4.5 GHz (n = 12)

Condition	Drop rate mean (%)	Drop rate SD	Median	IQR
CCD0, 20,000 RPS	0.04	0.01	0.04	0.01
CCD1, 20,000 RPS	0.10	0.02	0.10	0.02
CCD0, 25,000 RPS	0.22	0.03	0.21	0.04
CCD1, 25,000 RPS	0.61	0.07	0.61	0.08
CCD0, 36,000 RPS	0.45	0.06	0.45	0.07
CCD1, 36,000 RPS	10.39	1.12	10.28	1.21

Source: created by the authors

Table 6. Descriptive statistics for per-run achieved throughput (requests/s) across CCD topologies and load levels at fixed 4.5 GHz (n = 12)

Condition	Throughput mean	Throughput SD
CCD0, 20,000 RPS	19,996.00	18.00
CCD1, 20,000 RPS	19,989.00	24.00
CCD0, 25,000 RPS	24,978.00	31.00
CCD1, 25,000 RPS	24,941.00	54.00
CCD0, 36,000 RPS	35,837.00	96.00
CCD1, 36,000 RPS	31,492.00	1,827.00

Notes: throughput is reported as achieved requests per second
Source: created by the authors

Figure 5 demonstrated that failure rates remained comparatively low and stable on CCD0 across all three loads. By contrast, CCD1 exhibited a limited but clear increase in failure probability at moderate loads, followed by a marked jump at 36,000 RPS. This behaviour is consistent with a transition from degraded performance to systemic instability once the active working set substantially exceeded the effective cache capacity of the standard topology. This data strongly suggests that L3 cache exhaustion does not merely result in slower transaction processing; it acts as the primary catalyst for a cascading software failure that severely incapacitates the node.

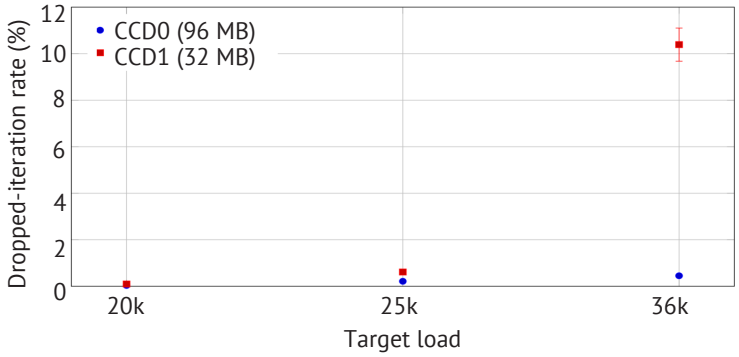

Figure 5. Dropped-iteration rate with 95% confidence intervals across load levels
Source: created by the authors

While the strict 4.5 GHz isolation explicitly demonstrates the architectural vulnerability of the 32 MB planar cache, the higher core frequencies observed during stock tests function as a partial compensation mechanism. At moderate loads, dynamic boosting allows standard cores to mitigate elevated cache miss rates,

slightly expediting recoveries from DRAM fetches. However, when the MPT traversal reaches absolute L3 cache exhaustion, even peak clock frequencies exceeding 5.4 GHz become insufficient. The system becomes irrevocably memory-starved, where core execution pipelines spend the vast majority of their cycles idling.

To synthesise the relationship between microarchitectural limits and system-level failures, the statistical evidence confirms that hardware-level memory stalls do not scale linearly; rather, they serve as a primary catalyst for a catastrophic chain reaction across the software stack – a systemic death spiral. Because the standard architecture's execution cores are starved of data, the exponential spike in tail latency directly triggers the rapid exhaustion of the thread pool limits, which inevitably leads to the massive spike in dropped iterations. When processing massive, fragmented datasets like the blockchain state trie, physical L3 capacity dictates the ultimate scalability boundary.

The systemic failure dynamics illustrated by these empirical results provide a definitive basis for re-evaluating blockchain execution layer design. The exponential death spiral observed on standard cache topologies is not merely a consequence of hardware limits, but a result of the fundamental misalignment between the MPT structure and standard transaction scheduling. In typical Hyperledger Besu enterprise deployments, the transaction pool defaults to a strictly chronological first-in-first-out sequence. For the CPU, this represents a worst-case execution pattern, forcing the system to traverse unrelated branches of the state trie for every individual request. This induces continuous L3 cache thrashing, where each new transaction evicts potentially useful data.

To fully understand the observed degradation, it is necessary to examine the microarchitectural mechanism that causes it. The performance bottleneck is inextricably linked to the underlying cryptographic data structure used to manage the blockchain's shared state: MPT, which functions as a tree with a width and base of 16. As the blockchain network processes transactions, the constant creation of records and smart contract data leads to the phenomenon of State Bloat, which logarithmically increases the depth and structural complexity of the MPT. Modern Ethereum clients attempt to mitigate disk I/O bottlenecks by implementing flat storage models, but the microarchitectural overhead associated with cryptographically verifying these proofs in memory causes a deeply technical crisis known as "pointer chasing" (Hsieh *et al.*, 2016). Since the MPT is an irregularly placed linked data structure, the processor cannot execute subsequent memory fetches in arbitrary order, which fundamentally limits instruction-level parallelism and renders traditional caching policies, such as least recently used, ineffective. High-frequency processor cores are forced to remain idle, waiting for data to be retrieved from dynamic random-access memory (DRAM), which directly subjects the system to the severe constraints of the Memory Wall paradigm (Farshin *et al.*, 2019). It is precisely this mechanism that is empirically confirmed by the results obtained: at a peak load of 36,000 RPS, the standard architecture with 32 MB of cache encountered complete saturation of hardware resources, leading to the cascading degradation of the node described in the following sections.

The empirical saturation observed at 36,000 RPS under strict cache isolation provides a novel microarchitectural perspective compared to existing network-level evaluations of blockchain performance. For instance, F. Leal *et al.* (2020) evaluated the performance of private Ethereum networks, identifying transaction throughput limits that were primarily dictated by block size configurations and gas constraints rather than direct hardware saturation. Similarly, G.A. Pierro *et al.* (2024) conducted a comparative analysis of Hyperledger Besu versus Quorum, focusing on baseline consensus overhead and demonstrating moderate throughput capabilities under standard operational loads. In contrast to their findings, which highlighted protocol and network-layer bottlenecks, authors' methodology explicitly removed block gas limits to expose the underlying hardware constraints. Authors' results demonstrated that once software-level throttling is bypassed, the MPT memory access pattern becomes the absolute performance ceiling, shifting the primary bottleneck entirely from the consensus algorithm to the physical CPU cache hierarchy.

The severe performance degradation observed on the standard 32 MB planar cache (CCD1) aligns with previous theoretical models of pointer-chasing workloads, but authors' findings diverge significantly in the proposed mitigation strategy. Y. Huang *et al.* (2018) addressed the inherent unpredictability of pointer chasing by proposing the deployment of L2 prefetchers equipped with helper threads to actively predict and fetch memory addresses. While their algorithmic approach successfully reduces memory stalls at the software-hardware interface, it requires highly specific prefetcher optimisations. Conversely, authors' evaluation of the 96 MB CCD0 provides empirical evidence that vertically expanding the L3 offers a brute-force, yet highly effective, hardware-level mitigation. Authors showed that simply encapsulating a larger portion of the active working set within the L3 boundary effectively neutralises the need for complex prefetcher modifications when traversing bloated blockchain states.

Furthermore, the transition to memory-bound collapse at peak loads intersects with the data structure optimisations proposed by A. Zeitak & A. Morrison (2021). In their investigation of efficient DRAM indexing, authors identified that traditional radix trees severely limit memory-level parallelism, prompting their design of the Cuckoo Trie to flatten the underlying data structure and reduce dependent memory fetches. Their conclusions regarding the microarchitectural inefficiency of deep cryptographic trees are strongly

supported by authors' CCD1 dropped-iteration data. However, whereas A. Zeitak & A. Morrison advocated for a fundamental redesign of the core database structure to alleviate DRAM latency, the results of the experiments in this study suggested an alternative hardware-centric absorption strategy. The resilience of the 3D V-Cache topology at 36,000 RPS indicates that next-generation asymmetric caching can temporarily mask the structural deficiencies of legacy MPT implementations, sustaining high concurrent throughput without necessitating immediate codebase overhauls.

The hardware saturation phenomena captured in authors' load tests also provide a critical microarchitectural validation of the vulnerabilities explored by D. Perez & B. Livshits (2020) regarding EVM resource metering. They highlighted that certain EVM instructions, particularly those invoking state I/O, are systematically underpriced relative to the actual computational and memory burden they place on the executing node. Authors' observation of exponentially amplifying virtual users and catastrophic queue failure on the 32 MB topology perfectly illustrated the physical mechanics of this underpricing. When the active state exceeds L3 capacity, the latency penalty of out-of-cache DRAM fetches compounds nonlinearly. This confirms that if organic state bloat forces consistent MPT traversals, standard enterprise nodes could be driven into a denial-of-service state even while processing nominally "cheap" transactions, reinforcing the need for state-size-aware gas algorithms.

Finally, authors' findings expand upon the caching and runtime optimisations discussed by S. Oaks (2020) and M. Sultan *et al.* (2024). M. Sultan *et al.* (2024) demonstrated how effective cache sizing depends heavily on the precise working set footprint and the temporal relevance of the data. Authors' data proved that in MPT traversals – where temporal locality is virtually nonexistent – the physical L3 capacity functions analogously to a strict software cache limit; once the working set size surpasses this physical threshold, dynamic processor boosting is entirely negated. Additionally, while S. Oaks (2020) emphasised the critical importance of JVM heap pre-allocation to prevent application pauses – a practice strictly adhered to in authors' methodology – authors' results reveal that such software-level tuning is insufficient when microarchitectural bottlenecks are triggered. Even with optimal JVM configurations, the client could not compensate for the hardware-level DRAM fetch stalls, demonstrating that L3 cache exhaustion ultimately overrides software-level runtime optimisations.

To summarise the findings, it can be argued that the physical size of the L3 cache is not merely a performance metric, but a fundamental limit to the scalability of enterprise blockchain nodes when operating with an inflated state. Empirical data convincingly demonstrates that the traditional strategy of increasing processor clock speed is an ineffective response to microarchitectural limitations caused by irregular MPT traversal. Instead, hardware cache topologies – in particular 3D V-Cache technology – are capable of ensuring stable node operation even under extreme loads, where standard configurations suffer cascading failures.

CONCLUSIONS

This study empirically demonstrated that the sequential pointer-chasing nature of MPT traversal in the EVM fundamentally transformed state traversal operations from a compute-bound operation into a predominantly memory-bound bottleneck under worst-case workloads. Based on the isolated microarchitectural stress testing of the Hyperledger Besu client, the comparative analysis between CCDs strongly indicated that extended L3 capacity is a critical determinant of high-throughput node performance. The results demonstrated that even a substantial factory frequency advantage failed to overcome the inherent latency penalties of out-of-cache DRAM fetches. Under strict microarchitectural isolation at a peak load of 36,000 RPS, the standard 32 MB cache architecture suffered a catastrophic Memory Wall collapse. This was characterised by a nearly eighteen-fold increase in P95 latency (GMR = 17.84) and a massive dropped-iteration rate of 114.80% (IRR = 22.99) compared to the 96 MB topology.

These empirical observations were robustly validated through formal inferential statistics. Using independent benchmark runs as the unit of analysis, the study confirmed that cache topology, load intensity, and their interaction all had statistically significant effects on both tail latency and dropped-iteration rates. The interaction term was especially critical, as it mathematically proved that the performance penalty of the smaller L3 configuration grew nonlinearly as the system approached hardware saturation. Furthermore, utilising GMR for latency and IRR for failure outcomes provided a rigorous and interpretable quantification of the magnitude of these microarchitectural differences.

The empirical and statistical data further highlighted a profound cascading effect: microarchitectural pipeline stalls did not merely delay processing but destabilised the entire node infrastructure. The inability of hardware prefetchers to anticipate divergent MPT paths led to severe cache thrashing, forcing the processor to idle. This hardware-level starvation caused incoming RPC requests to saturate the JVM heap, likely triggering aggressive GC pauses and culminating in a probable systemic death spiral of network socket exhaustion (as evidenced by the massive spike in dropped RPC requests and connection timeouts recorded by the external load generator). Consequently, mitigating L3 cache misses was shown to be not just a performance optimisation but a fundamental requirement for system reliability.

Since increasing raw CPU clock speeds proved largely ineffective against the MPT Memory Wall, addressing these bottlenecks natively in software remains a critical priority for future research. Subsequent studies

will focus on exploring cache-aware scheduling paradigms, specifically proposing methods to dynamically pre-sort and group incoming transactions to artificially induce spatial data locality. Such algorithmic reordering is expected to maximise L3 cache hit rates, thereby enabling COTS processors to sustain massive transaction throughput without collapsing under memory-bound constraints.

ACKNOWLEDGEMENTS

None.

FUNDING

None.

CONFLICT OF INTEREST

None.

REFERENCES

[1] Bhargava, R., & Troester, K. (2024). AMD next-generation "Zen 4" core and 4th Gen AMD EPYC server CPUs. *IEEE Micro*, 44(3), 8-17. doi: 10.1109/MM.2024.3375070.

[2] Cardoso, C., Silva, C., Veloso, A., Sousa, J., & Abelém, A. (2025). An API-driven framework for performance testing of Hyperledger Besu blockchain networks. In *Lablock artigos curtos – Latin-American symposium on dependable and secure computing* (pp. 23-28). Porto Alegre: Sociedade Brasileira de Computação. doi: 10.5753/ladc_estendido.2025.16872.

[3] Chen, Z., Wu, C., Gu, Y., Jia, R., Li, J., & Guo, M. (2025). Gaze into the pattern: Characterizing spatial patterns with internal temporal correlations for hardware prefetching. In *2025 IEEE international symposium on high performance computer architecture* (pp. 173-187). New York: IEEE. doi: 10.1109/HPCA61900.2025.00024.

[4] Deng, Y., Yan, M., & Tang, B. (2024). Accelerating Merkle Patricia Trie with GPU. *Proceedings of the VLDB Endowment*, 17(8), 1856-1869. doi: 10.14778/3659437.3659443.

[5] Fan, C., Lin, C., Khazaei, C., & Musilek, P. (2022). Performance analysis of Hyperledger Besu in private blockchain. In *IEEE international conference on decentralized applications and infrastructures* (pp. 64-73). New York: IEEE. doi: 10.1109/DAPPS55202.2022.00016.

[6] Farshin, A., Roozbeh, A., Maguire, G.Q.Jr., & Kostić, D. (2019). Make the most out of last level cache in Intel processors. In *Proceedings of the fourteenth Eurosys conference 2019* (article number 8). New York: Association for Computing Machinery. doi: 10.1145/3302424.3303977.

[7] Hsieh, K., Khan, S., Vijaykumar, N., Chang, K.K., Boroumand, A., Ghose, S., & Mutlu, O. (2016). Accelerating pointer chasing in 3D-stacked memory: Challenges, mechanisms, evaluation. In *2016 IEEE 34th international conference on computer design* (pp. 25-32). New York: IEEE. doi: 10.1109/ICCD.2016.7753257.

[8] Huang, Y., Zhu, H., & Li, Y. (2018). Performance improvements by deploying L2 prefetchers with helper thread for pointer-chasing applications. *International Journal of Performability Engineering*, 14(10), 2312-2320. doi: 10.23940/ijpe.18.10.p7.23122320.

[9] Kuznetsov, O., Frontoni, E., Kuznetsova, K., & Arnesano, M. (2025). Optimizing Merkle proof size through path length analysis: A probabilistic framework for efficient blockchain state verification. *Future Internet*, 17(2), article number 72. doi: 10.3390/fi17020072.

[10] Kuznetsov, O., Oleshko, I., Tymchenko, V., Lisitsky, K., Rodinko, M., & Kolhatin, A. (2021). Performance analysis of cryptographic hash functions suitable for use in blockchain. *International Journal of Computer Network and Information Security*, 13(2), 1-15. doi: 10.5815/ijcnis.2021.02.01.

[11] Leal, F., Chis, A.E., & González-Vélez, H. (2020). Performance evaluation of private Ethereum networks. *SN Computer Sciences*, 1, article number 285. doi: 10.1007/s42979-020-00289-7.

[12] Liu, L., Wang, H., Wang, A., Xiao, M., Cheng, Y., & Chen, S. (2021). Mind the gap: Broken promises of CPU reservations in containerized multi-tenant clouds. In *Proceedings of the ACM symposium on cloud computing* (pp. 243-257). New York: Association for Computing Machinery. doi: 10.1145/3472883.3486997.

[13] Mardiansyah, V., Muis, A., & Sari, R.F. (2023). Multi-State Merkle Patricia Trie (MSMPT): High-performance data structures for multi-query processing based on lightweight blockchain. *IEEE Access*, 11, 117282-117296. doi: 10.1109/ACCESS.2023.3325748.

[14] Oaks, S. (2020). *Java performance: In-depth advice for tuning and programming Java 8, 11, and beyond*. Santa Rosa: O'Reilly Media.

[15] Perez, D., & Livshits, B. (2020). Broken Metre: Attacking resource metering in EVM. *ArXiv*. doi: 10.48550/arXiv.1909.07220.

[16] Pierro, G.A., Coco, L., & Tonelli, R. (2024). Besu vs. Quorum: Comparative analysis in the context of simulated energy communities. In *6th distributed ledger technology workshop*. Torino: University of Turin.

[17] Singh, T., *et al.* (2025). "Zen 5": The AMD high-performance 4nm x86-64 microprocessor core. In *2025 IEEE international solid-state circuits conference* (pp. 1-3). New York: IEEE. doi: 10.1109/ISSCC49661.2025.10904529.

[18] Stępień, K., & Skublewska-Paszkowska, M. (2025). Performance evaluation of REST and GraphQL API aproaches in data retrieval scenarios using NestJS. *Journal of Computer Sciences Institute*, 36, 350-356. doi: 10.35784/jcsi.7794.

[19] Sultan, M., Shakiba, K., Lee, A., Chen, P., & Stumm, M. (2024). TTLs matter: Efficient cache sizing with TTL-aware miss ratio curves and working set sizes. In *Proceedings of the nineteenth European conference on computer systems* (pp. 387-404). New York: Association for Computing Machinery. doi: 10.1145/3627703.3650066.
[20] Wood, G. (2022). *Ethereum: A secure decentralised generalised transaction ledger*. Retrieved from https://ethereum.github.io/yellowpaper/paper.pdf.
[21] Zeitak, A., & Morrison, A. (2021). Cuckoo Trie: Exploiting memory-level parallelism for efficient DRAM indexing. In *Proceedings of the 28th ACM symposium on operating systems principles* (pp. 147-162). New York: Association for Computing Machinery. doi: 10.1145/3477132.3483551.
[22] Zhang, I., Zarick, R., Wong, D., Kim, T., Pellegrino, B., Li, M., & Wong, K. (2025). QMDB: Quick Merkle Database. *ArXiv*. doi: 10.48550/arXiv.2501.05262.

Вплив об'єму кешу останнього рівня на обхід дерева стану в Hyperledger Besu при найгіршому сценарії доступу

Максим Бистрик

Аспірант
Вінницький національний технічний університет
21021, вул. Хмельницьке шосе, 95, м. Вінниця, Україна
https://orcid.org/0009-0004-4949-829X

Олександр Хошаба

Кандидат технічних наук, доцент
Вінницький національний технічний університет
21021, вул. Хмельницьке шосе, 95, м. Вінниця, Україна
https://orcid.org/0000-0001-5375-6280

Анотація. Стрімкий розвиток корпоративних блокчейн-мереж змістив вузькі місця продуктивності від мережевого консенсусу до внутрішніх механізмів обходу бази даних стану, де нерегулярний доступ до пам'яті створює критичні мікроархітектурні обмеження. Основною метою роботи було емпіричне визначення того, як об'єм кеш-пам'яті останнього рівня впливає на продуктивність та стабільність високонавантажених вузлів Hyperledger Besu під час навігації по дереву станів. Дослідження проведено в апаратно ізольованому середовищі з фіксованою частотою ядер на базі асиметричного процесора AMD, що дозволило прямо порівняти 32 МБ стандартного кешу та 96 МБ 3D-stacked пам'яті за найгіршого сценарію випадкового доступу. Результати тестування продемонстрували, що за умови досягнення апаратного насичення продуктивність вузла значною мірою диктується не тактовою частотою процесора, а швидкістю отримання даних та об'ємом кеш-пам'яті останнього рівня. Під час пікового навантаження у 36 000 запитів на секунду стандартна архітектура з 32 МБ кешу зіткнулася з бар'єром пам'яті (Memory Wall), що призвело до раптової деградації системи. Кількісний аналіз показав, що топологія з розширеним кешем (96 МБ) стабільно підтримувала затримку 95-го процентиля на рівні 6,55 мс, тоді як на стандартному кеші цей показник експоненціально зріс до 114,80 мс. Статистичне моделювання підтвердило, що використання меншого кешу призводить до майже 18-разового (GMR = 17,84) збільшення хвостової затримки під час насичення апаратних ресурсів. Крім того, частота критичної втрати оброблених даних для стандартної конфігурації зросла майже у 23 рази (IRR = 22,99), спричинивши системне вичерпання ресурсів і каскадний збій інфраструктури. Практична цінність дослідження полягає в емпіричному обґрунтуванні необхідності програмної оптимізації пулу транзакцій та пропонуванні ресурсно-адаптивного методу, який враховує апаратну топологію для масштабування корпоративних розподілених реєстрів

Ключові слова: блокчейн; високонавантажені системи; pointer chasing; Memory Wall; розростання стану; технологія 3D V-Cache; оптимізація операцій читання