Teku Version Comparison Report

Cross-Version Resource & Performance Analysis

v25.12.0 **v26.2.0** **v26.3.0**

StereumLabs Fleet • NDC2 Bare-Metal • 6 Execution Client Pairings
Measurement Windows: Jan 15–29 2026 • Feb 15 – Mar 1 2026 • Mar 15–29 2026
Report generated: April 2, 2026

StereumLabs — RockLogic GmbH

Table of Contents

1 Executive Summary

2 Methodology

3 Version Release Context

4 CPU Utilization

5 Memory & JVM Analysis

6 JVM Garbage Collection

7 Storage Engine & Disk I/O

8 Block Import Performance

9 P2P Networking & Peer Connectivity

10 Beacon Chain Protocol Metrics

11 Data Availability Sampling (DAS) & PeerDAS

12 Process-Level Resource Handles

13 Summary & Recommendations

1. Executive Summary

This report compares three Teku consensus client releases—25.12.0 (Dec 2025), 26.2.0 (Feb 2026), and 26.3.0 (Mar 2026)—across the StereumLabs NDC2 bare-metal fleet. Each Teku instance is paired with one of six execution clients (Besu, Erigon, Ethrex, Geth, Nethermind, Reth). All measurements use 14-day rolling averages from the Prometheus-cold datasource, aggregated per EC pairing then averaged to fleet-wide figures.

CPU USAGE (FLEET AVG)
−44%
25.12.0 → 26.3.0

RSS MEMORY
+2.8%
Peak at 26.2.0, recovered in 26.3.0

GC OVERHEAD
−35%
jemalloc impact in 26.3.0

BLOCK IMPORT DELAY
−24%
26.3.0 fastest across all EC pairings

DISK WRITE RATE (HOST)
−61%
RocksDB migration effect

DISK READ RATE (HOST)
−79%
RocksDB caching superiority

OPEN FILE DESCRIPTORS
+100%
Expected: RocksDB SST files

PEER COUNT (LIBP2P)
+9%
Steady improvement

Key findings: The migration from LevelDB to RocksDB in 26.2.0 was the single largest driver of improvement, slashing disk I/O dramatically while increasing CPU efficiency. Version 26.3.0 compounded the gains by introducing jemalloc for memory allocation in Docker images, which reduced JVM garbage collection overhead by ~35% and brought block import latencies to their lowest levels. The trade-off is a substantial increase in open file descriptors—a known RocksDB characteristic that requires appropriate `ulimit` configuration.

2. Methodology

All data is sourced from the `prometheus-cold` datasource (UID: `aez9ck4wz05q8e`, Org 6) in the StereumLabs Grafana instance. Teku nodes are filtered with `cc_client="teku"` and `role="cc"`.

Time windows: For each version, a 14-day measurement window was selected during the version's active deployment period, avoiding the first few days after upgrade to exclude transition effects.

Version	Release Date	Measurement Window	Key Changes
25.12.0	Dec 16, 2025	Jan 15 – Jan 29, 2026	Late block reorg, block building prep, sidecar recovery
26.2.0	Feb 11, 2026	Feb 15 – Mar 1, 2026	RocksDB default, DAS backfiller, getBlobs API
26.3.0	Mar 5, 2026	Mar 15 – Mar 29, 2026	jemalloc, SSZ serialization fix, partial sidecar import

Query pattern: `avg by (ec_client) (avg_over_time(metric{...}[14d:1h]))` evaluated as an instant query at the end of each window. Rate metrics use `rate(...[5m])` inside the subquery. All 6 EC pairings are averaged to produce fleet-wide figures unless noted otherwise.

Environment: All nodes run on NDC2 bare-metal servers (Vienna, Austria), managed by RockLogic GmbH. This eliminates cloud-based noise from noisy neighbors and provides consistent hardware baselines. Prometheus scrape interval is 15 seconds.

3. Version Release Context

25.12.0 — Fulu-Ready Stabilization (Dec 16, 2025)

Released as a recommended update following the Fulu fork activation (epoch 411392, Dec 3, 2025). Key additions include the `beacon_earliest_available_slot` and `data_column_sidecar_processing_validated_total` metrics, late block reorg enabled by default, and a new sidecar recovery mechanism. This version still uses **LevelDB** as its database backend. Bug fixes addressed a storage crash during shutdown and a peer_count metric issue.

26.2.0 — The RocksDB Migration (Feb 11, 2026)

The most architecturally significant release in this comparison. RocksDB became the default storage engine for new databases (existing LevelDB databases are not auto-migrated). A new data column sidecar backfiller improves custody data downloads. New CLI flags include `--rest-api-getblobs-sidecars-download-enabled` and `--force-clear-db`. The release notes caution operators to monitor CPU usage from the backfiller, with tuning options available.

26.3.0 — jemalloc & SSZ Fix (Mar 5, 2026)

A mandatory update addressing a critical SSZ serialization bug causing integer overflow errors on mainnet state serialization. Beyond the fix, this release introduces **jemalloc** as the memory allocator in Docker images—replacing the default glibc malloc—which improves memory allocation patterns and reduces fragmentation. Nodes with >50% custody requirements now benefit from partial sidecar import (downloading 50% first, backgrounding the rest).

4. CPU Utilization

Metric: `rate(process_cpu_seconds_total{job="teku"}[5m])` — measures the Teku JVM process CPU consumption in CPU-seconds per second (equivalent to fraction of one CPU core).

EC Pairing	25.12.0	26.2.0	26.3.0	Δ 25.12→26.2	Δ 26.2→26.3
Besu	0.761	0.441	0.431	−42%	−2%
Erigon	0.704	0.414	0.409	−41%	−1%
Ethrex	0.594	0.552	0.419	−7%	−24%
Geth	0.771	0.553	0.429	−28%	−22%
Nethermind	0.720	0.330	0.305	−54%	−8%
Reth	0.722	0.364	0.377	−50%	+4%
Fleet Average	**0.712**	**0.442**	**0.395**	−38%	−11%

CPU usage dropped sharply (−38%) from 25.12.0 to 26.2.0, with the transition to RocksDB as the primary driver. RocksDB's block cache and bloom filters reduce the amount of processing needed per state lookup

compared to LevelDB. Version 26.3.0 added a further 11% reduction, attributable to jemalloc's more efficient memory allocation reducing GC-driven CPU spikes. The Nethermind pairing consistently shows the lowest Teku CPU usage, while Besu/Geth pairings are slightly higher.

5. Memory & JVM Analysis

5.1 Process RSS Memory

Metric: `process_resident_memory_bytes{job="teku"}` — total physical memory consumed by the Teku JVM process.

EC Pairing	25.12.0 (GB)	26.2.0 (GB)	26.3.0 (GB)	Δ 25.12→26.3
Besu	8.67	9.43	9.20	+6.2%
Erigon	9.10	9.31	9.16	+0.6%
Ethrex	8.74	9.48	9.20	+5.3%
Geth	9.03	9.50	9.10	+0.7%
Nethermind	9.07	9.34	9.13	+0.6%
Reth	8.90	9.39	9.20	+3.4%
Fleet Average	**8.92**	**9.41**	**9.16**	+2.8%

RSS memory peaked in 26.2.0 (+5.5% vs 25.12.0), consistent with RocksDB's larger in-memory structures—including its block cache, memtables, and bloom filter data. Version 26.3.0 reclaimed approximately half of that increase through jemalloc's reduced memory fragmentation, bringing the net increase down to a modest +2.8% end-to-end.

5.2 JVM Heap Memory

Metric: `jvm_memory_used_bytes{area="heap"}` — Java heap utilization within the JVM.

EC Pairing	25.12.0 (GB)	26.2.0 (GB)	26.3.0 (GB)	Δ 25.12→26.3
Besu	5.33	5.73	5.99	+12.3%
Erigon	5.76	5.70	5.82	+1.0%
Ethrex	5.89	5.77	6.05	+2.7%
Geth	5.38	6.14	6.05	+12.3%
Nethermind	5.36	6.01	6.16	+15.0%
Reth	5.32	5.78	6.15	+15.5%
Fleet Average	**5.51**	**5.85**	**6.04**	+9.6%

JVM heap usage grew steadily across all three versions. This reflects increased in-heap state caching (the RocksDB JNI bridge keeps Java-side references) and growing chain state size as Ethereum's validator set and state tree expand. The increase is gradual (~5% per version step) and does not indicate a regression—heap is staying well within normal bounds for Teku's default 8 GB max heap configuration.

5.3 JVM Native Memory

Version	Fleet Avg (MB)	Delta
25.12.0	705	—
26.2.0	739	+4.8%
26.3.0	736	−0.4%

Native (off-heap) memory rose modestly with RocksDB (which uses native memory for its block cache) and stabilized with jemalloc.

6. JVM Garbage Collection

Metric: `rate(jvm_gc_collection_seconds_sum[5m])` — fraction of time spent in GC per second. Lower is better. This is a critical Teku-specific metric because GC pauses directly impact block processing latency.

EC Pairing	25.12.0 (ms/s)	26.2.0 (ms/s)	26.3.0 (ms/s)	Δ 25.12→26.3
Besu	2.77	2.82	1.90	−31%
Erigon	1.59	3.01	2.29	+44%
Ethrex	2.85	3.87	1.58	−45%
Geth	3.29	3.01	1.60	−51%
Nethermind	3.15	2.14	1.73	−45%
Reth	2.83	2.34	1.47	−48%
Fleet Average	**2.75**	**2.87**	**1.76**	−36%

GC overhead was essentially flat from 25.12.0 to 26.2.0 (even rising slightly for some pairings due to RocksDB's JNI allocation patterns). The introduction of **jemalloc** in 26.3.0 is the standout change: fleet-wide GC time dropped 36% end-to-end. jemalloc reduces heap fragmentation, which means fewer large-object promotions to old gen and thus fewer full GC cycles. For a Java-based client like Teku, this is a particularly impactful improvement—GC pauses are a primary contributor to block import latency.

> **Note:** The Erigon pairing shows an anomalous increase in GC time across all versions. This may reflect interaction effects with Erigon's execution response patterns rather than a Teku-internal issue.

7. Storage Engine & Disk I/O

7.1 Host-Level Disk I/O

Metrics: `rate(node_disk_read_bytes_total[5m])` and `rate(node_disk_written_bytes_total[5m])` via node_exporter. These are host-level metrics that include both the Teku CC and its paired EC, but since EC versions didn't change, the delta is attributable to the Teku version change.

EC Pairing	Disk Read Rate (KB/s)			Disk Write Rate (KB/s)		
	25.12.0	26.2.0	26.3.0	25.12.0	26.2.0	26.3.0
Besu	2,625	483	390	6,040	1,305	1,272
Erigon	424	470	350	1,141	1,282	1,294
Ethrex	1,127	603	392	2,143	1,888	1,303
Geth	1,528	710	259	3,172	2,003	1,227
Nethermind	1,489	220	186	2,583	1,145	1,304
Reth	1,916	326	283	4,468	1,243	1,220
Fleet Average	**1,518**	**469**	**310**	**3,258**	**1,478**	**1,270**

Disk I/O improvements are dramatic. Read throughput dropped 79% end-to-end (1,518 → 310 KB/s), and write throughput fell 61% (3,258 → 1,270 KB/s). RocksDB's block cache and SST-based storage design is far more read-efficient than LevelDB's approach, requiring fewer disk reads per state access. The Besu pairing saw the most dramatic read improvement (−85%), suggesting its execution workload pattern particularly benefited from RocksDB's caching.

7.2 RocksDB Internal Metrics (26.2.0+ only)

> These metrics are only available for versions 26.2.0 and 26.3.0. Version 25.12.0 used LevelDB which does not expose equivalent counters.

Metric	26.2.0 Fleet Avg	26.3.0 Fleet Avg	Delta
storage_bytes_read rate (KB/s)	84.3	103.6	+23%
storage_bytes_written rate (MB/s)	1.53	1.35	−12%
storage_compact_write_bytes rate (KB/s)	543.5	484.0	−11%

RocksDB's internal write amplification improved between versions: both raw write rate and compaction write rate decreased by ~11%. The slight increase in read rate likely reflects the partial sidecar import feature in 26.3.0, which reads more data during background sidecar reconciliation.

8. Block Import Performance

Metric: `beacon_block_import_delay_latest` — the most recent block import delay in milliseconds. This captures the end-to-end latency from receiving a block to completing its import into the beacon state.

EC Pairing	25.12.0 (ms)	26.2.0 (ms)	26.3.0 (ms)	Δ 25.12→26.3
Besu	268	482	221	−18%
Erigon	519	490	347	−33%
Ethrex	772	303	209	−73%
Geth	251	319	209	−17%
Nethermind	239	513	444	+86%
Reth	284	426	352	+24%
Fleet Average	**389**	**422**	**297**	−24%

Block import latency shows an interesting pattern: 26.2.0 was slightly slower fleet-wide than 25.12.0, likely due to early-stage RocksDB tuning and the concurrent DAS backfiller adding load. Version 26.3.0 brought a strong recovery, achieving the lowest fleet-wide latency at 297 ms. The Ethrex pairing benefited most dramatically (−73%), while the Nethermind pairing shows elevated import times that persisted—this warrants EC-side investigation.

> **Operator impact:** Lower block import delay means attestations can be created sooner after a block arrives, directly improving validator effectiveness and reducing inclusion delay.

9. P2P Networking & Peer Connectivity

9.1 Peer Counts

Metric	25.12.0	26.2.0	26.3.0	Trend
beacon_peer_count (fleet avg)	44.9	45.9	49.5	+10%
libp2p_peers (fleet avg)	90.8	91.9	98.9	+9%
discovery_live_nodes (fleet avg)	160.7	168.4	176.4	+10%

All peer connectivity metrics show steady improvement across versions. The beacon_peer_count (which Teku 26.2.0 fixed for the metrics-publish-endpoint) rose from ~45 to ~50. The libp2p_peers metric—which includes both mesh and non-mesh peers—grew from ~91 to ~99. Discovery live nodes increased from ~161 to ~176, suggesting the Discv5 layer is finding and maintaining more active ENR records. These improvements are partly version-related and partly reflect growing network maturity post-Fulu.

9.2 Gossip Message Rate

Version	Fleet Avg (msg/s)	Notes
25.12.0	6.23	High variance: some EC pairings near 0 (sync issues)
26.2.0	5.99	More consistent across pairings
26.3.0	7.85	Highest, most uniform distribution

Gossip throughput increased in 26.3.0, consistent with improved block processing speed allowing faster re-gossip of received messages. The 25.12.0 values are skewed by the Erigon and Ethrex pairings showing near-zero gossip rates—likely indicating sync or connectivity issues during the measurement window that were resolved in subsequent versions.

10. Beacon Chain Protocol Metrics

10.1 Chain Reorgs

Version	Fleet Avg Reorg Rate (/hour)	Assessment
25.12.0	0.47	Baseline (late block reorg newly enabled)
26.2.0	0.26	−45% — lowest reorg rate
26.3.0	0.74	+185% vs 26.2.0

The reorg rate dipped in 26.2.0, possibly due to the dependent root calculation changes and improved block import pipeline. The rise in 26.3.0 coincides with the SSZ serialization bug fix—the serialization errors in the days before the fix may have caused temporary state inconsistencies that elevated the reorg count in the early portion of the measurement window. The absolute rates remain low (less than 1 reorg/hour is well within normal Ethereum mainnet behavior).

10.2 JVM Thread Pool Activity

Version	Fleet Avg Thread Count	Notable Executor Queues
25.12.0	180	p2p queue: ~0.6, storage_query: ~0.3
26.2.0	175	All queues near zero (improved RocksDB throughput)
26.3.0	180	All queues near zero (stable)

Thread counts are stable across versions (~175–180). The p2p and storage_query executor queues, which showed non-trivial depths in 25.12.0 (especially on Besu and Reth pairings), were driven to near-zero in 26.2.0+, confirming that RocksDB's faster read path eliminates the storage bottleneck that was causing queue buildup under LevelDB.

11. Data Availability Sampling (DAS) & PeerDAS

Metric: `rate(beacon_data_column_sidecar_processing_requests_total[5m])` — rate of data column sidecar processing requests, a key Fulu/PeerDAS metric measuring the DAS workload.

EC Pairing	25.12.0 (/s)	26.2.0 (/s)	26.3.0 (/s)
Besu	0.441	0.170	0.306
Erigon	0.000	0.164	0.358
Ethrex	0.000	0.288	0.284
Geth	0.424	0.350	0.434
Nethermind	0.420	0.000	0.048
Reth	0.240	0.059	0.177

DAS processing rates show significant per-pairing variance, reflecting differences in how quickly each EC responds to execution payload requests (which gates block processing, which gates sidecar validation). The Erigon and Ethrex pairings produced zero DAS requests in 25.12.0 (likely due to sync issues), but normalized by 26.3.0. The new DAS backfiller in 26.2.0 initially appeared to reduce request rates as it optimized batch downloads, while 26.3.0's partial sidecar import feature brought rates back up with a more efficient processing pattern.

> **Context:** Version 26.3.0 allows nodes with >50% custody requirements to begin importing blocks after downloading only 50% of sidecars, backgrounding the remainder. This architectural change improves block processing throughput without compromising data availability guarantees.

12. Process-Level Resource Handles

12.1 Open File Descriptors

EC Pairing	25.12.0	26.2.0	26.3.0	Δ 25.12→26.3
Besu	480	957	1,329	+177%
Erigon	780	931	1,375	+76%
Ethrex	801	1,031	1,299	+62%
Geth	643	939	1,319	+105%
Nethermind	682	751	1,245	+83%
Reth	527	787	1,279	+143%
Fleet Average	**652**	**899**	**1,308**	+101%

File descriptor usage doubled from 25.12.0 to 26.3.0—the most significant trade-off of the RocksDB migration. RocksDB's LSM-tree architecture maintains many open SST (Sorted String Table) files simultaneously, each held open for efficient random reads. This growth continued from 26.2.0 to 26.3.0 as the database matured and accumulated more SST files through compaction cycles.

> **Operator action required:** Ensure `ulimit -n` is set to at least 65536 on hosts running Teku 26.x. The default 1024 limit on many Linux distributions would cause immediate failures. Docker containers should pass `--ulimit nofile=65536:65536`.

13. Summary & Recommendations

Version-by-Version Assessment

Dimension	25.12.0	26.2.0	26.3.0
CPU Efficiency	Baseline	Major improvement	Best
Memory Footprint	Lowest	+5.5%	+2.8% (recovering)
GC Overhead	Baseline	Similar	Best (–36%)
Disk I/O	Highest	Major improvement	Best
Block Import Speed	Moderate	Slight regression	Best (297ms)
Peer Connectivity	Good	Good	Best
File Descriptors	Lowest	+38%	+101% (monitor)
Storage Backend	LevelDB	RocksDB (new default)	RocksDB + jemalloc
Stability	Stable	Stable	Mandatory (SSZ fix)

Key Recommendations

1. Upgrade to 26.3.0 is mandatory. Beyond the SSZ serialization bug fix, 26.3.0 delivers the best performance profile across nearly every dimension. The combination of RocksDB and jemalloc produces the lowest CPU usage, lowest GC overhead, fastest block imports, and best peer connectivity of any Teku release in this comparison.

2. Verify file descriptor limits. With 26.3.0 averaging 1,308 open FDs (and likely higher under peak load), operators must ensure their system and container configurations allow at least 65,536 file descriptors. Failure to do so risks `java.io.IOException: Too many open files` errors.

3. Monitor RocksDB compaction. The new RocksDB storage metrics (`storage_compact_write_bytes`, `storage_bytes_written`) should be added to operator dashboards. Abnormal spikes in compaction write rate can indicate database health issues.

4. Consider DAS backfiller tuning. If 26.2.0/26.3.0 nodes show elevated CPU during backfill, `--Xp2p-reworked-sidecar-custody-sync-batch-size=1` can throttle the backfiller at the cost of slower historical data sync.

5. Watch Nethermind pairing block import latency. The Nethermind+Teku pairing shows persistently higher block import delays across 26.x versions. This appears to be an interaction effect rather than a Teku regression, but warrants investigation on the EC side.