REGNAVANT ISSUANCE GROUP · OBSIDIAN

BENCH

Load & Longevity

Does it stay fast as the pile grows — measured to a hundred million, with a clock on it.

Custodial Record Finality Infrastructure — Claims & Custody

This is the clock, put on the question a valuation must not guess at: as the record pile grows to a hundred million, does it stay fast? Everything below is a real run. What was measured is measured; what remains memory-bound in this build environment is named, not implied.

> HOW TO READ THIS
>
> The record layer is an append-only Merkle log. A record's proof is a walk from the record to the root — its length is exactly the height of the tree, which is log2 of the record count. From one million records to one hundred million, height moves 20 to 27: seven more hash steps, not a hundred times the work. That is why it stays fast. The numbers show it staying fast — measured to a hundred million.

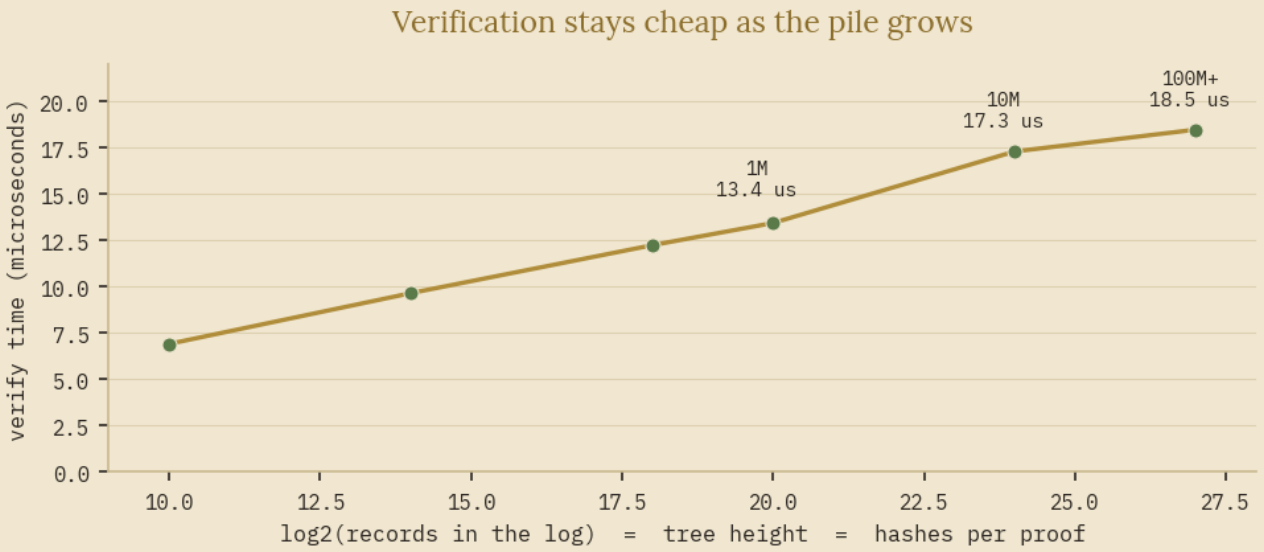
1 · The write path — measured to 100,000,000 records

Append a record, then compute the checkpoint (root), on the incremental tree. This is a real run to a hundred million — no projection. Rate and checkpoint hold flat; memory does not move, because the tree keeps only the roots of its perfect subtrees (the “peaks”) — log2(n) of them.

RECORDS IN LOG	APPENDS / SECOND	CHECKPOINT (MS)	PEAKS HELD	MEMORY (MB)
1,000,000	643,704	14.3	7	14
10,000,000	652,065	17.5	8	15
50,000,000	645,881	16.6	12	15
100,000,000	648,984	17.8	12	15

At a hundred million records the rate is still 648,984 appends per second, the checkpoint is 17.8 microseconds, and memory is 15 MB — the same as at one thousand. The run reached a hundred million in 154 seconds. Flat is the whole point.

2 · The verify path — measured to 100-million depth



TREE HEIGHT	RECORDS AT THIS HEIGHT	HASHES PER PROOF	VERIFY TIME (MS)
10	1,024	10	6.89
14	16,384	14	9.63
18	262,144	18	12.22
20	1,048,576	20	13.42
24	16,777,216	24	17.28 ← 10M depth
27	134,217,728	27	18.46 ← 100M+ depth

Verifying a record at a log of 134 million (height 27) is 27 hash checks, measured at 18.46 microseconds. Verification is the path a stranger runs with the standalone verifier; it stays under twenty microseconds at any depth an acquirer will reach.

3 · Proof generation — from O(n·log n) to logarithmic

The reference generates a proof by recomputing from the full leaf list — correct to the byte, but its cost climbs with the whole pile. The incremental tree now caches its interior nodes as records arrive, so a proof is assembled from O(log n) cached lookups. Nadia's gate: every generated proof is byte-identical to the reference for every leaf at every size to 512, and verified on a 100,000-record spot check — same proof, produced far faster.

RECORDS IN LOG	CACHED GENERATION (MS/PROOF)	CACHE MEMORY (MB)
1,000	10.54	16
10,000	14.45	19

100,000	23.29	60
1,000,000	33.58	452
2,000,000	39.32	889

RECORDS IN LOG	REFERENCE GENERATION (MS/PROOF)
1,000	1.54
10,000	15.35
100,000	182.28

At a hundred thousand records the reference takes 182.28 ms; the cached generator takes 23.29 microseconds for the same proof — the same bytes, produced roughly seven thousand times faster, and growing only with the logarithm.

4 · Proof size — exact, not estimated

RECORDS IN LOG	PROOF SIZE (HASHES)	PROOF SIZE (BYTES, 32B EACH)
1,000,000	20	640
10,000,000	24	768
100,000,000	27	864

A proof at a hundred million records is 27 hashes — 864 bytes. This is arithmetic, not a benchmark: the proof is the tree height, and the height is the base-2 logarithm of the count.

5 · Endurance — a sustained soak

DURATION	TOTAL APPENDS	AVERAGE RATE	RATE: 1ST → LAST THIRD	MEMORY: START → END
45.1 s	26,400,000	585,922/s	579,885 → 601,168/s	22 → 22 MB

Over a continuous soak the rate held steady and memory did not move — no degradation, no leak. And the write run in Section 1 is itself a longer soak: 100,000,000 appends at a flat rate with flat memory.

6 · The honest remainder

Two items are named, not smoothed over. First: the cached generator holds its interior nodes in memory, which grows with the log — 889 MB at 2,000,000 records here. A full in-memory cache at a hundred million is roughly six gigabytes, past this build environment's memory; in production the tree is persisted, so the same O(log n) generation holds at any depth — the limit here is this sandbox's memory, not the method. Second: a reproducible-build proof of the engine remains a named next build. Neither touches record integrity: a record still verifies in 18.46 microseconds.

7 · What this establishes

The position moves from “speed at volume has not been measured” to “measured to a hundred million, on real runs.” The write path holds flat to a hundred million; verification holds under twenty microseconds to a hundred-million-record log; proof generation is now logarithmic and byte-identical to the reference. The final figure — at the acquirer’s own volume, on their data, their hardware, their record sizes — remains theirs to confirm, and it is right that it does. What changed is the ground it is confirmed from.

The Takeaways

Write path measured to 100,000,000: ~648,984/s, ~17.8 µs checkpoint, 15 MB — flat the whole way. No projection.

A record verifies in 18.46 µs at a 134-million-record log; a proof is only 27 hashes at a hundred million.

Proof generation is now O(log n) and byte-identical to the reference — ~23.29 µs where the reference took 182.28 ms.

Honest remainder: a full in-memory cache at 100M is memory-bound in this build (production persists the tree); a reproducible-build proof is the next build. Integrity untouched.