PURETENSOR
INTELLIGENCE DISTILLED

Why a Cold Vector Search Takes Tens of Minutes

47,000 serial disk reads at queue depth one, and which levers move it

Paper: PT-R-2026-060 v1.0

Author: PureTensor Ltd

Date: 29 October 2026

Status: Published

Abstract

A semantic query over our library-scale passage index (84.3 million passages, 768-dimension embeddings, 64 partitions, each with its own HNSW index, a graph-based approximate nearest-neighbour structure) takes tens of minutes when nothing is in memory, and 0.05 seconds when it is. Production telemetry for the first incident shows 47,064 device reads at a mean of 21.9 ms, which sums to about 17 minutes of device read time; a second incident sums to 24.1 minutes.

We ran a pre-registered study on copy-on-write clones of the production volumes. The cold query is 47,000 to 57,000 page reads of 8 KB, issued one at a time (queue depth one) against a volume that now sits on 12 hard disks. Reads multiplied by per-read latency account for 99.87% and 99.96% of wall time on PostgreSQL's own counters (98.7% and 99.1% on block-device counters) for the two bench runs, which took 31.9 and 22.8 minutes. Buffer settings, a sequential-scan fallback and the client node were not causes. Halving the HNSW search width trimmed reads only 1.5x. Priming the operating-system cache with the query's own blocks cut that run to 151 s, but only for that vector.

What moves the time is concurrency and media. Running one scan per partition concurrently took a cold query from 1,912 s to 95 s (20x, with caveats on vector and hour; 22x on reads per second). A SATA SSD tier read a page in 0.7 to 1.3 ms instead of about 27 to 44 ms. Extrapolating measured partition times gives 1 to 3 minutes for a serial SSD query, which we did not run. A streaming merge of the per-partition results reproduced today's top three on the one query we tested, including a near-tie miss.

Most results are single runs, and per-read latency on the hard-disk pool drifted between 13.5 and 45.8 ms within one day at low write rates. Section 8 states what survives.

1. The Query That Takes Tens of Minutes

Production runs PostgreSQL 15.18 with pgvector 0.5.1 over the passage index: 84.3 million passages in 64 hash partitions, each with an HNSW index (m=16, ef_construction=200). The 64 indexes total 345.3 GB against an 82.4 GB heap and sit on their own tablespace on a Ceph block volume. The volume's pool still carries "nvme" in its name, but since 25 September 2026 it has been placed on 12 hard-disk OSDs (Ceph's per-disk daemons), 3-way replicated across

four storage hosts.

The query has no partition filter, so the plan is a Limit over joins over a Merge Append of 64 HNSW index scans. Settings: shared_buffers 8 GB, effective_cache_size 24 GB, random_page_cost 1.1, hnsw.ef_search 40.

Two production incidents prompted the study. Both come from block-device counters, so the read counts cover a window, not exactly one query.

Incident	Window	Device reads	Mean read latency	Summed read time	Pool scrubbing
Index volume on a small storage host, 28 October	20:02 to 20:22 (1,200 s)	47,064	21.89 ms	1,030.0 s = 17.2 min	0 PGs
Index volume on a large compute host, 29 October	04:15 to 05:10 (3,300 s)	47,295	30.59 ms	1,446.7 s = 24.1 min	up to 8 PGs, 4 deep

Table 1. The two production incidents, from device telemetry (UTC). PGs are Ceph placement groups.

Neither row is a wall time. The device was busy 85.9% of the first window and 43.7% of the second, so "24.1 minutes" is read time summed inside a 55-minute window. The bench wall times (Section 3) were 22.8 and 31.9 minutes.

We asked what a cold query waits on, and which changes move it, by how much and at what cost.

2. How We Set Up the Study

We wrote hypotheses and decision rules down first, tested them on a clone, and gave every result to a sceptic whose job was to explain it away. The bench was a throwaway pod on a four-GPU compute node, mounting copy-on-write clones of the production snapshots on the same 12 hard-disk OSDs. A guard in the harness refused to target the production database, which was never queried or changed.

The harness recorded, per run, PostgreSQL buffer reads and hits, per-read I/O time, block-device reads and latency, and Ceph load. Raw probes used fio on the same files. "Cold" means the page cache for the index, catalog and heap files was evicted first, checked with fincore (resident fraction 0.0). We used 48 short query vectors from an embedding model. Ten validation checks passed before any study run, including that PostgreSQL read counters agree with block-device counters (ratio 0.94).

Hypotheses H1 to H9 were written at 05:24:30 with the file hash recorded, each with a predicted value, a 95% band, and accept and reject rules; H10 was added at 10:55:52, before any study run. A hypothesis is accepted only if the harness shows the effect and the sceptic does not refute it, rejected if its registered reject rule is met. The log holds 262 records. "The gate" below means the full production query shape, run over all 64 partitions.

The holdout used 24 predictions (value and 95% band) from a model fitted on non-holdout runs, written at 19:26:50, before the first holdout run at 19:27:36, on partitions and queries no design run had used (one disclosed exception, Section 8). The pass threshold was a median relative error of 0.35. Statistics are descriptive.

3. Where the Time Goes

Run	Wall	PG reads	Device reads	Device per read	PG per read	I/O / wall	Busy
Baseline gate, query A	1,912.1 s = 31.9 min	57,419	54,586	34.57 ms	33.26 ms	99.87%	0.987
Holdout gate, query B	1,369.9 s = 22.8 min	53,868	49,351	27.52 ms	25.42 ms	99.96%	0.991

Table 2. The two bench cold gates, each with 12 PGs scrubbing and no bulk writes (averaging 9.8 and 0.46 MiB/s). I/O / wall is PostgreSQL-counted reads times PostgreSQL per-read time over wall.

On block-device counters, read time alone accounts for 98.7% (1,887 of 1,912 s) and 99.1% (1,358 of 1,370 s) of wall. With the pages already in shared_buffers the same query took 0.052, 0.050 and 0.049 s with zero reads: cold over warm is 36,800x to 38,700x.

The range of 47,000 to 57,000 reads mixes device-counted and PostgreSQL-counted reads across three query vectors. It is an order of magnitude, not a constant.

Heap blocks were 0.34 to 0.37% of index blocks (1.6 to 1.7% with TOAST, out-of-line storage, included). The query is an index walk.

4. Why the Reads Are Serial, and What a Read Costs

A cold partition query reads 431 to 1,362 pages (median 941, n = 56). The Merge Append walks the 64 partitions one after another, and each HNSW hop needs the previous page before it knows the next, so the device sees queue depth one. The evidence is the device busy fraction (median 0.975 at concurrency 1; 0.987 and 0.991 for the gates) and the ratio of PostgreSQL per-read latency to raw fio latency on the same files: 0.86 to 0.95.

The same disks serve far more when asked in parallel. Raw fio on the hard-disk volume gave 25.4 IOPS at queue depth 1 (39.3 ms mean) and 541 IOPS at queue depth 16 (29.4 ms mean): 21x the throughput at no higher mean latency (single probes of 300 and 600 reads). The SATA SSD volume gave 1,554 and 14,085 IOPS, local NVMe 2,460 and 34,483.

The distribution matters. Three long queue-depth-1 probes of 2,500 reads on a quiet pool gave means of 27.6, 29.4 and 28.9 ms, but medians of 11.3 to 12.4 ms and a 99th percentile of 304 to 317 ms: the mean is set by the tail.

Nor is the mean constant through the day. Cold per-read latency on single-partition hard-disk queries was 87.8 to 116.6 ms (median 102.0, 8 runs) from 11:53 to 12:07 under 148.5 to 168.8 MiB/s of pool writes; 13.5 to 26.4 ms (median 16.2, 24 runs) from 12:53 to 13:05; and 24.1 to 45.8 ms (median 40.4, 13 runs) from 13:06 to 13:55 with writes at 0.8 to 8.5 MiB/s. The day's extremes, 13.5 and 45.8 ms, both came at 4 to 8 MiB/s of writes.

Raw fio sentinel probes (300 reads, run all day) drift the same way: their median stayed at 11 to 16 ms while their mean ranged from 15.0 ms (13:01) to 52.3 ms (13:25). The session coefficient of variation is 0.29 (n = 46), below the registered 0.30 but 0.31 to 0.36 by other cuts. A cold query's wall time scales with whatever the pool's tail is doing that hour.

5. What Does Not Matter, and What Barely Helps

Buffer settings (H4). Setting shared_buffers to 128 MB instead of 8 GB changed nothing: 12 of 12 same-partition cold-then-replay pairs read an identical number of pages, and a 4 MB control raised reads only 1.017x. A cold query touches each page once, so buffer size changes retention, not the cold cost.

Sequential-scan fallback (H6). Every gate plan had 64 HNSW scans under one Merge Append and no sequential scan, even with random_page_cost 4 and parallelism off.

HNSW search width (H3). Lowering hnsw.ef_search from 40 to 20 on 53 partitions in a concurrent run cut reads 1.57x and 1.48x and wall time 1.39x and 1.21x. Applied to the 17-minute query this would give about 11 to 12 minutes (projected, not run), and it costs recall.

Client node (H9, one node measured). The bench client, a different node from both incidents, read at 22.8 to 29.5 ms in its final block: 1.04x to 1.35x the first incident and 0.75x to 0.96x the second. An earlier block read 1.9x the first incident, which is the hour-to-hour drift of Section 4. The 1.40x gap between the incidents is confounded with scrubbing, so we do not attribute it to the node.

Prewarming (H2). Block-list prewarm of the 830 to 905 blocks a partition query touches took 0.73x to 1.23x the cold query's wall time (median 0.88x): the same random reads, issued earlier. Priming the operating-system cache with the gate's own blocks and restarting PostgreSQL did help that query: wall fell from 1,912 s to 150.7 s (12.7x), with 4,797 of 54,586 device reads left. It is still not a fix: the benefit is specific to the vector that supplied the blocks, warming with one vector left about 77% of the cold reads for six others (exploratory, pair ratios 0.48 to 1.48), and the warm-up costs the cold reads it saves.

Competing load (H7). Four extra fio readers raised the queue-depth-1 mean about 1.3x (pooled interval 1.04 to 1.61, straddling the registered line), mostly in the tail. H7 is unresolved.

Bulk writes (H10). The registered test (at least 3x latency under bulk writes) was rejected: fio probes under 87 to 150 MiB/s of pool writes read at most 1.5x slower. Against that, the 8 cold partition queries at the start of the session read at a median of 102.0 ms against 25.4 ms for 41 runs under 20 MiB/s, yet across all 56 rho is -0.03. The loaded runs all fall in the first 14 minutes, so time and load are confounded; we do not claim bulk writes cause a 4x slowdown.

6. What the Fixes Do

Per-partition fan-out. The cheapest change is in code: run one scan per partition concurrently and merge the results. Per-read latency rises with concurrency, consistent with the pool queueing.

Run	Concurrency	Partitions	Device reads	Wall	Reads/s	Mean read
Serial	1	3	2,104	95.3 s	22.1	44.3 ms
Serial (later)	1	6	4,899	118.9 s	41.2	23.9 ms
Fan-out	8	8	6,475	51.7 s	125	57.2 ms
Fan-out	16	16	14,983	70.9 s	211	63.7 ms
Fan-out (117 MiB/s of writes)	16	16	14,088	155.7 s	90.5	145 ms
Fan-out, fresh vector	16	16	13,933	44.8 s	311	33.8 ms
Fan-out (57 MiB/s of writes)	32	51	50,431	71.4 s	706	33.0 ms
Fan-out, all partitions	64	64	46,406	95.0 s	488	89.5 ms

Table 3. Serial against concurrent per-partition scans on the hard-disk bench. Two further 53-partition runs, not plotted, read 477 to 655 reads/s.

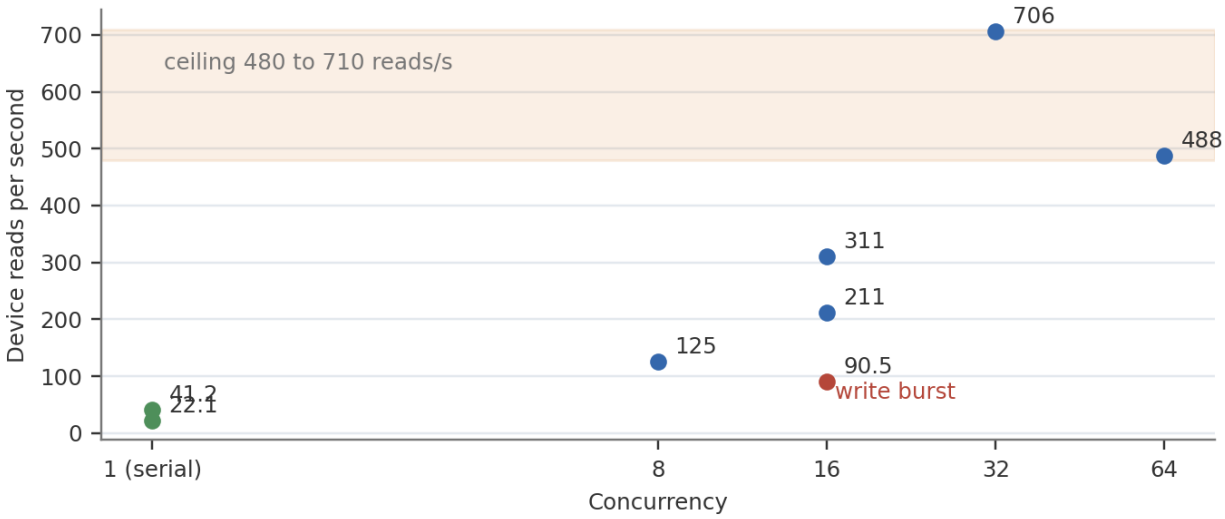


Figure 1. Device reads per second against concurrency, from Table 3. Serial runs sit at 22 to 41 reads/s; fan-out runs of 32 and 64 sit at 488 to 706.

Against the serial baseline gate (1,912 s), the all-partition run at concurrency 64 took 95.0 s: 20.1x. That compares different query vectors at different hours, so we prefer reads per second: against the 22 reads/s serial rate, concurrency 64 is 22x. At concurrency 16 the ratio is 9.6x to 14x, or 7.5x against the later 41 reads/s serial run; the run during a write burst got only 4.1x. Concurrency 32 (706 reads/s) beat 64 (488), so we do not take either figure for a measured spindle limit; the 706 run exceeds the 541 IOPS raw probe. A single later all-partition concurrency-16 run took 151.1 s cold; it ran with a background archive verifier resuming and has no per-run record beyond its wall time.

Streaming merge. H8 required fan-out to return an identical top-k, and failed as registered: on one query the harness's merge (a full sort of the per-partition candidates by recomputed distance) returned three ids that differed from the production gate's. An exploratory follow-up points to a cause. On the bench, 17 of the 64 partitions return their top candidates slightly out of distance order (near-ties within about 2e-7), and Merge Append appears to trust each child's order. A streaming k-way merge of the per-partition lists in returned order reproduced the gate's top three exactly; the full sort reproduced the fan-out's list.

The gate is not the true ranking. The three smallest distances among the five ids seen are 4.47e-7, 5.04e-7 and 5.07e-7, and the full sort returned exactly those. The gate returned 5.04e-7, 5.07e-7 and 5.15e-7. In the per-partition SQL, one partition returned a slightly farther candidate first (6.48e-7 before 4.47e-7), at every ef_search we tried. That is consistent with Merge Append not looking past it, but we did not re-run the full gate on this vector with a plan to confirm, the sceptic could not explain the miss, and the order test covered 6 of 64 partitions. The streaming merge therefore matches today's behaviour, near-tie ordering included, and does not improve recall. This is one query vector on pgvector 0.5.1; we did not test later releases, and near-tie ordering is what an approximate index may return.

SSD tablespace. We placed cloned partitions, one at a time, on a replicated SATA SSD pool (four SSD OSDs, one per storage host, no power-loss protection) and on local NVMe.

Tier	Per-read device latency	Cold single-partition query
Hard disk (contemporary)	27 to 44 ms	28.99 to 37.4 s; 42.3 s control; 69 to 131 s in the loaded hour
SATA SSD	0.704 to 1.292 ms (7 runs; four lowest 0.704 to 0.777)	1.02 to 2.49 s
Local NVMe	0.034 to 0.443 ms (7 runs)	0.063 to 0.957 s

Table 4. Per-read latency and cold single-partition query time by storage tier.

Per read, SATA SSD was about 21x to 62x faster than the contemporary hard-disk readings, taking the extremes of both ranges; cold partition wall-time ratios run from 14x to 129x depending on the normalisation. Heap and catalog reads stay on the hard-disk data volume and set a floor.

We did not run the full gate on SSD. Per-read arithmetic (47,000 to 57,000 reads at 0.70 to 1.29 ms, plus up to 23 s of heap and catalog reads on hard disk) gives 33 to 97 s, but it undershoots our own measured partition rows: 1.02 to 2.49 s times 64 partitions is 65 to 159 s, and the median partition (1.67 s) extrapolates to 107 s. We take the extrapolation, roughly 1 to 3 minutes, as the better guide, and make no projection for SSD combined with fan-out.

Case	Cold full query (s)	Basis
Serial, hard disk, baseline gate	1,912	measured
Serial, hard disk, holdout gate	1,370	measured
Fan-out concurrency 16, all partitions	151	measured, single run, background load
Fan-out concurrency 64, all partitions	95	measured
Serial, SSD tablespace	65 to 159 (median 107)	extrapolated from measured partition times

Table 5. Cold full-query time by lever. The first four rows are measured; the last is an extrapolation from single-partition SSD queries.

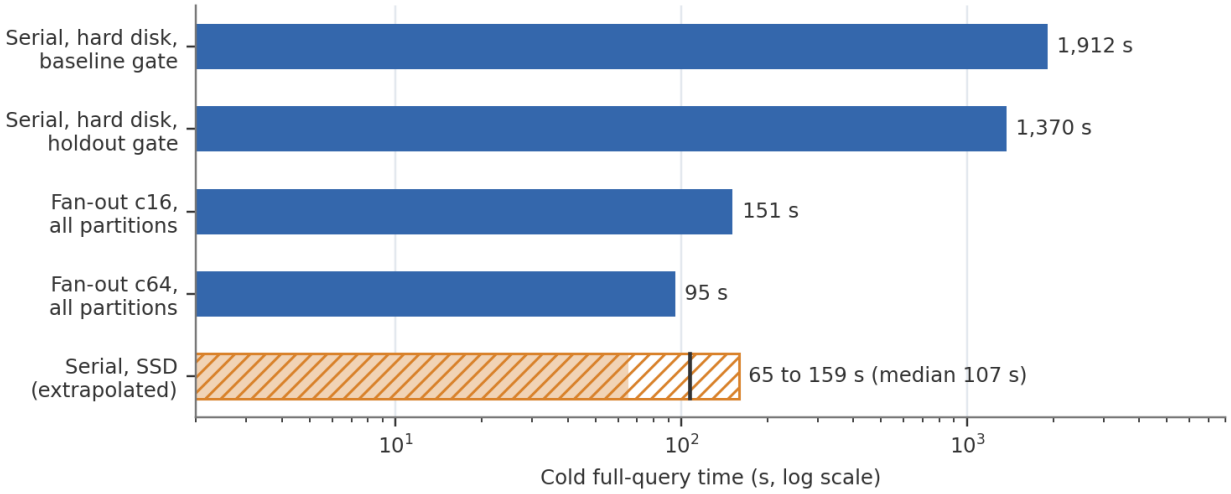


Figure 2. Cold full-query time by lever on a logarithmic axis. Solid bars are measured; the hatched bar is an extrapolation, with its median marked.

The move has a price. Copying a partition to SSD ran at 52 MB/s, so 345 GB would take about 1.8 hours, one partition at a time, each locking its index for roughly 100 to 150 s. The SATA SSDs lack power-loss protection, and our earlier published work found one of them already degraded; a 345 GB write onto them is a risk we did not test.

7. Verdicts and the Holdout

Registered verdicts: hard-disk latency as the multiplier (H1), the residency miss (H2), the SSD and NVMe speed-ups (H5) and the client node result (H9) were accepted; ef_search (H3), buffer settings (H4) and the sequential-scan fallback (H6) were confirmed non-causes; competing I/O (H7) was unresolved; fan-out (H8) failed on its identity clause while the speed-up was supported; bulk writes (H10) were rejected.

The holdout scored a median relative error of 0.115, with 22 of 24 predictions inside their 95% band (threshold 0.35). Bands for the noisy latencies are wide, which makes them easy to hit. Headline hits: the full cold gate at 1,370 s against 1,424 s predicted, and gate page reads at 53,868 against 57,420. The two misses were hard-disk queue-depth-16 IOPS (541 against 250 predicted) and the copy to local NVMe (144.3 s against 25.2 s).

8. What These Numbers Will Not Carry

Single runs dominate: each gate is one run and each incident row one query. Per-read latency drifted from 13.5 to 45.8 ms at low write rates, so any wall-time claim should be a range. The 17 minutes is summed read time from telemetry; the bench gave 22.8 and 31.9 minutes on different vectors and hours.

The 20x headline compares different vectors at different hours; reads per second is cleaner, and it levels off. The SSD time is an extrapolation. Not measured: an end-to-end gate on SSD, a run controlled for write load, a second client node, and fan-out with the joins inside the search service.

Deviations. The first cold gate attempt, during a 235 to 245 MiB/s write burst, was lost without a record and rerun. A fio bug returned zero-I/O records; they are excluded and the probes were rerun. The concurrency-64 run covered the 8 held-out partitions, which we disclosed before the holdout ran. The designers, sceptics and harness were all language-model agents from one family, and the sceptics ran as sub-tasks, not a separate review.

What survives: the cold query is dominated by serial hard-disk reads (reads times latency is 99.9% of wall on PostgreSQL's counters, 98.7% to 99.1% on the block device); settings, plan and client node do not change it, and prewarming helps only the vector that supplied the blocks; concurrency and faster media change it by large factors; and hard-disk read time varies by hour in a way we cannot yet explain.

9. What We Changed, and What Is Open

As of 29 October 2026 only one thing has changed in production: the database pod was restarted back onto the large compute host, which keeps pages resident longer but does not shorten the first cold query. The open decisions are to ship a per-partition fan-out in the search service (a pool of 16 to 32 with a streaming merge), to move the index tablespace to the SSD pool in an operator window with the power-loss risk in view, and to keep bulk writes to the hard-disk pools away from latency-sensitive checks such as post-deploy health probes. We will leave shared_buffers, effective_cache_size and ef_search alone and skip block-list prewarm.

10. Methods and Artefacts

Software: PostgreSQL 15.18 with pgvector 0.5.1 (HNSW); Ceph block volumes on 12 hard-disk OSDs and 4 SATA SSD OSDs; Kubernetes with copy-on-write clones; fio; fincore; Prometheus (read-only); kernel block-device counters; nomic-embed-text. Statistics: medians, ranges, ratios and Spearman rho. We retain the pre-registration, the 262-record experiment log, the holdout predictions and scoring, the sceptic reports and the runner logs.

PureTensor operates a sovereign AI infrastructure fleet (on-premises GPU compute, storage, and serving) run day-to-day by autonomous agents under human direction. This paper is PT-R-2026-060. The measurements were taken on 28 and 29 October 2026; the raw artefacts are retained.