Chroma vs pgvector – ROUGH DRAFT / NOTES

Leo – el20166 / Team 25

Abstract—TODO: write proper abstract later
Quick version:

- comparing chroma & pgvector for vector search
- why: need actual numbers for picking a vector DB (RAG, embeddings, etc)
- what we measured: ingest speed, query latency, recall, storage, cpu/mem
- setup: 10k–2M vectors, dims 32–1536, w/ and w/o filters
- both in docker, same limits (6cpu 4gb)
- results:
  - ingest: chroma faster early (~7k v/s vs 3.8k), but eats RAM
  - queries: pgvector wins latency (~1ms), recall is mixed
  - filters: pgvector handles them way better
- code is public for repro

Index Terms—vector db, similarity search, chroma, pgvector, benchmarks, ANN, HNSW, IVFFlat

I. Intro

what are vector dbs?

- store embeddings, do NN queries fast
- used everywhere now: search, RAG, recs, vision stuff

problem:

- regular SQL DBs bad at high-dim distance calcs
- need ANN for scale

what we're comparing:

- Chroma: standalone embedding db, python-native, uses HNSW
- pgvector: postgres extension, SQL, IVFFlat+HNSW, gets ACID for free

goals:

- run both under same constraints
- vary dataset size + dims
- measure ingest + queries + recall + resource usage
- test filters
- compare index configs

II. Background

A. Vector Search

- find closest vectors to query – basic def, skip formulas for now

B. Indexes

1) HNSW:

- graph-based ANN, multi-layer small world thing
- params: M (connections), ef_construction (build quality)

2) IVFFlat:

- k-means clustering, search subset of clusters
- param: lists (num clusters)

C. Chroma

- embedding db, client-server
- HNSW built on insert (important!)
- python API, collections
- has metadata filtering

D. pgvector

- postgres ext
- HNSW + IVFFlat
- full SQL, joins, WHERE clauses work
- ACID from postgres

III. Method

A. Hardware

[TODO: fill in machine specs]

B. Software

[TODO: versions table]

C. Docker setup

- both in containers, 6cpu 4gb limit
- same network, same constraints

D. Fairness stuff

- reset DBs between runs (docker down -v, up)
- warmup queries before measuring
- same vectors + metadata for both (fixed seed=42)

E. Data gen

- gaussian random vectors
- metadata: cls in A,B,C, dist 10/30/60%

F. Datasets

- size scaling (dim=128): 10k, 50k, 100k, 500k, 1m, 2m
- dim tests: 32d, 768d, 1536d variants

G. What we measure

- ingest: time, vecs/sec, mem, cpu, disk
- queries: latency (mean/p50/p95/p99), recall vs brute force

H. Recall calc

- ground truth = exact L2 scan (np)
- recall@k = overlap / k

IV. Ingestion

A. How we ran it

- for each dataset: reset, ingest all batches (500/1000/5000), create index, record stats
- KEY: chroma builds HNSW during insert!! pgvector inserts first then indexes

B. Phases

- phase1: size scaling w/ IVFFlat baseline
- phase2: dimensionality tests
- phase3: HNSW param sweep
- phase4: IVFFlat param sweep
- phase5: HNSW at scale

C. Main numbers (batch=1000, IVFFlat)

- 50k: chroma 6405 v/s vs pg 3840 v/s
- 100k: chroma 5874 vs pg 3665
- 500k: chroma 4270 vs pg 3814
- 1m: chroma 3350 vs pg 3817 – they converge!
- → chroma wins early (1.5-1.7x), gap shrinks, pg catches up at 1m

D. Batch size effect (500k)

chroma: 500→4238, 1000→4270, 5000→5569 (loves big batches)
pgvector: basically flat ~3.8k regardless
why: chroma has HTTP overhead, bigger batches amortize it

E. Storage (MB)

- 50k: chroma 33 vs pg 55
- 100k: 66 vs 110
- 500k: 332 vs 551
- 1m: 664 vs 1099
- chroma uses ~40% less disk! pg IVFFlat index is huge

F. Memory (peak)

- 50k: chroma 132MB vs pg 88MB
- 100k: 240 vs 149
- 500k: 1.07GB vs 275MB
- 1m: 2.06GB vs 401MB
- chroma RAM explodes – HNSW building in memory

G. CPU

- chroma: 150-180% (building graph while inserting)
- pgvector: 32-35% (just inserting rows)
- 4-5x diff but chroma still wins throughput early

H. pgvector time breakdown

- 50k: insert 13s, index 0.3s, total 13.3s
- 1m: insert 261s, index 4.5s, total 265s
- IVFFlat index build is FAST (1.5-1.7% of total)

I. Dimensionality

high dims hurt both, but pg gets destroyed:

- 50k_32d: chroma 7747 vs pg 6175
- 50k_1536d: chroma 1583 vs pg 259 (!!!!)
- if you have 768d/1536d embeddings, pg ingest is rough

J. ingest summary

chroma wins: faster early, less disk, simple API
chroma loses: RAM grows crazy fast, slows at 1m+
pgvector wins: stable scaling, predictable mem, fast IVFFlat build
pgvector loses: slower early, more disk, HNSW build is slow

V. Queries

A. How we ran it

ingest, restart containers (clear cache), ground truth, 10 warmups, 100 queries
k = 10/50/100, filter vs no filter

B. Latency (HNSW, k=10, no filter)

10k: chroma 1.76ms vs pg 0.83ms
50k: 1.65 vs 0.89
100k: 1.89 vs 0.95
500k: 2.34 vs 1.21
pg consistently ~2x faster

C. Recall

10k: chroma 0.916 vs pg 0.868 – chroma
50k: 0.732 vs 0.795 – pg
100k: 0.689 vs 0.742 – pg
no clear winner

D. Filters

chroma: 1.65ms → 3.81ms w/ filter (2.3x slower!!)
pg: 0.89ms → 0.84ms (same)
pg handles filters way better

E. Top-K

chroma slows w/ bigger k
pg stays flat

F. At 2m scale

both slow down, recall tanks (0.13–0.21)
pg still faster

G. CPU

pg so fast docker barely sees it
chroma: spikes at 500k+

H. results

latency: pg wins
recall: mixed
filters: pg wins big

- 10k: Chroma 1.76 (P99 1.99) vs pgvector 0.83 (P99 1.50)
- 50k: Chroma 1.65 (P99 2.11) vs pgvector 0.89 (P99 1.62)
- 100k: Chroma 1.89 (P99 2.45) vs pgvector 0.95 (P99 1.78)
- 500k: Chroma 2.34 (P99 3.12) vs pgvector 1.21 (P99 2.15)

takeaway:

- pgvector is consistently ~2x lower latency in these runs

I. Recall (HNSW, no filter)

- measured recall:
  - 10k: Chroma mean 0.916 (min 0.60) vs pgvector mean 0.868 (min 0.40)
  - 50k: Chroma mean 0.732 (min 0.30) vs pgvector mean 0.795 (min 0.48)
  - 100k: Chroma mean 0.689 (min 0.28) vs pgvector mean 0.742 (min 0.43)
- takeaway:
  - Chroma looks better at 10k recall
  - pgvector looks better at 50k/100k in these configs
  - so: no universal winner; depends on config + scale

J. Filters (50k, HNSW, k=10)

- no filter vs with filter:
  - Chroma: 1.65ms / 0.732 recall → 3.81ms / 0.624 recall
  - pgvector: 0.89ms / 0.795 recall → 0.84ms / 0.585 recall
- takeaway:
  - Chroma slows down ~2.3x with filter
  - pgvector latency stays basically the same (SQL WHERE is strong here)
  - recall drops for both with filters

K. Top-K impact (50k, HNSW, no filter)

- Chroma:
  - k=10: 1.65ms, recall 0.732
  - k=50: 2.52ms, recall 0.615
  - k=100: 2.60ms, recall 0.557
- pgvector:
  - k=10: 0.89ms, recall 0.795
  - k=50: 0.89ms, recall 0.796
  - k=100: 0.70ms, recall 0.796
- takeaway:
  - Chroma latency rises with k, pgvector is stable in these measurements
  - recall behavior differs; pgvector looks stable at 50k here

L. Large summary table trend (HNSW, no filter)

- as dataset size increases (500k, 1m, 2m):
  - latency increases for both systems
  - recall drops for both systems (expected for ANN at scale if params not scaled too)
- standout observation from our aggregated table:
  - at 2m: Chroma latency can spike (e.g., 4.36ms at k=10, 9.99ms at k=50) while pgvector stays lower (3.54ms, 4.22ms)
  - recall at 2m is low for both (0.21 down to 0.13 range depending on k/system)

M. CPU during queries (HNSW, k=10, no filter)

- measured CPU %:
  - 10k: Chroma 14.0 vs pgvector 0.04
  - 50k: Chroma 13.3 vs pgvector 0.06
  - 100k: Chroma 12.3 vs pgvector 0.06
  - 500k: Chroma 108.7 vs pgvector 0.04
- takeaway:
  - pgvector queries are so fast that Docker CPU sampling barely catches them
  - Chroma CPU spikes at bigger sizes (graph traversal work becomes heavier / parallel)

N. Query summary

- latency:
  - pgvector is consistently faster in our runs (often ~2x)
- recall:
  - mixed; depends on dataset size + parameters
  - you can't pick a winner without stating recall target + latency budget
- filters:
  - pgvector is clearly better for filtered vector search (latency basically unchanged)
  - Chroma filter adds significant overhead in our setup

VI. Limitations / TODO

what we didn't do:

- no distributed/cluster mode
- only synthetic gaussian data (real embeddings might behave diff)
- single-client only

future stuff:

- test w/ real embeddings (openai, sbert, etc)
- concurrent clients
- update/delete perf
- tighter RAM limits

VII. Conclusion

bottom line:

chroma = fast ingest for small/med, less disk, simple, but RAM hungry at scale

pgvector = lower latency, great filters, stable scaling, but more disk + slow HNSW builds

when to pick what:

chroma if:

- prototyping / RAG experiments
- < 500k vectors
- want minimal ops work

pgvector if:

- already have postgres
- need low latency
- do filtered searches
- want SQL + ACID + vectors together

repo: [TODO: add link]

refs & AI ack: [TODO]