A ⚡Year-in-Review of vectors in PostgreSQL

Jonathan Katz

Principal Product Manager – Technical
Amazon RDS

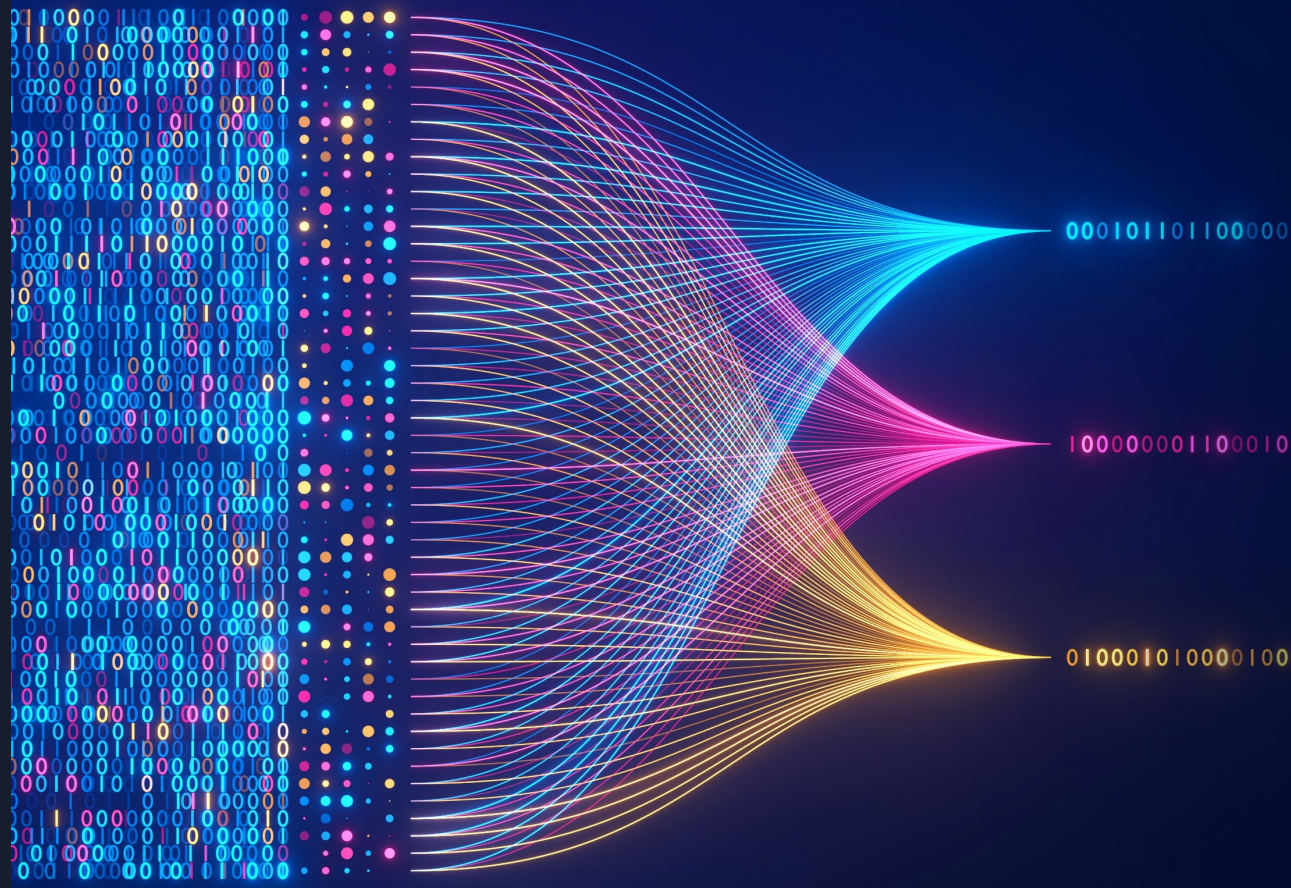
Generative AI is powered by foundation models

Pretrained on vast amounts of unstructured data

Contain a large number of parameters that make them capable of learning complex concepts

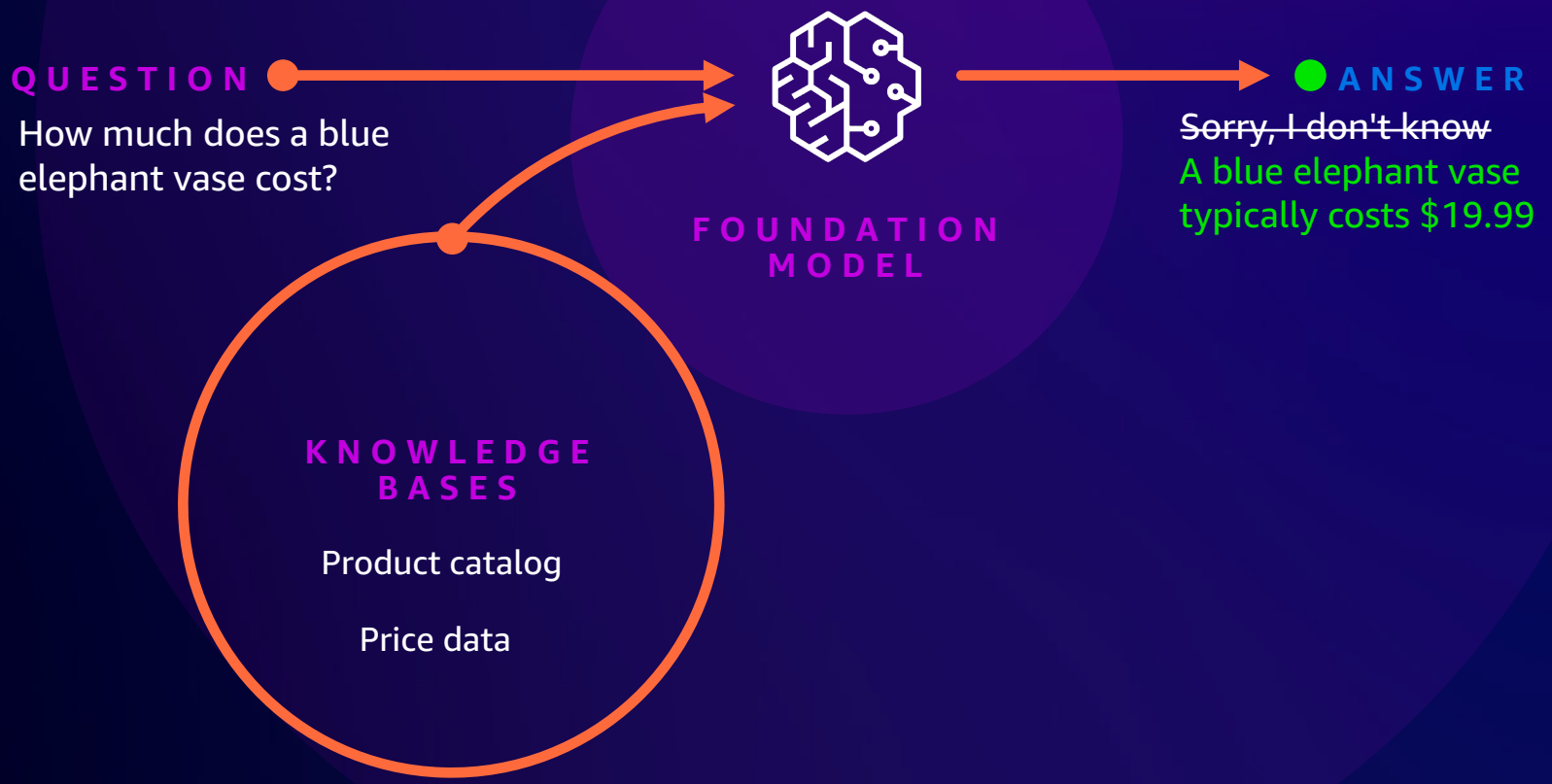
Can be applied in a wide range of contexts

Customize FMs using your data for domain-specific tasks

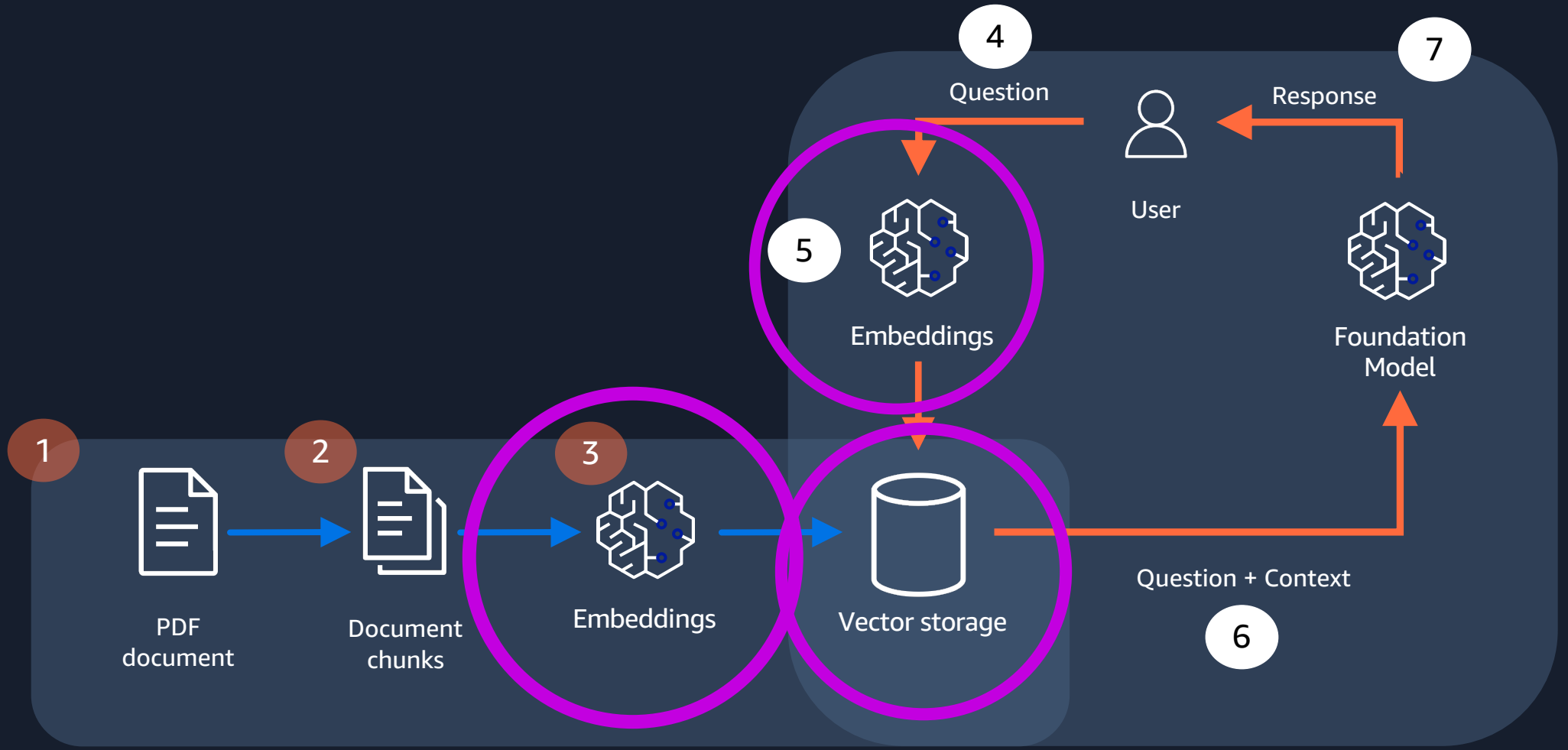


Retrieval-augmented generation (RAG)

Configure FM to interact with your company data

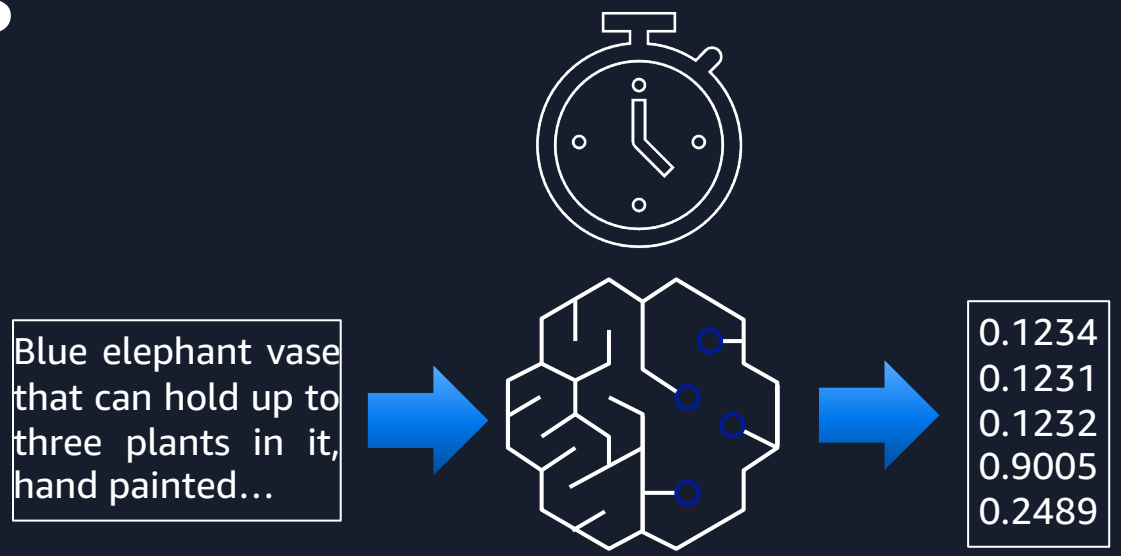


How vector embeddings are used



Challenges with vectors

- Time to generate embeddings
- Embedding size
- ~~Compression~~
- Query time



1536-dim 4-byte floats
6152B => 6KiB
1,000,000 => 5.7GB

Approximate nearest neighbor (ANN)

- Find similar vectors without searching all of them
- Methods include hash, tree, clustering, graph
- Faster than exact nearest neighbor
- “Recall” – % of expected results

Recall: 80%

Why use PostgreSQL for vector searches?

- Existing client libraries work without modification
- Convenient to co-locate app + AI/ML data in same database
- Interfacing with PostgreSQL storage gives ACID transactional storage

PostgreSQL support for vectors

Native

- ARRAY
- cube

Extensions

- pgvector
- ~~pg_embedding~~
- pgvecto.rs
- Lantern
- Timescale Vector
- pgvector-remote

What is pgvector?

An open-source extension that:

Adds support for storage, indexing, searching, metadata with choice of distance

vector data types

Supports IVFFlat/HNSW indexing

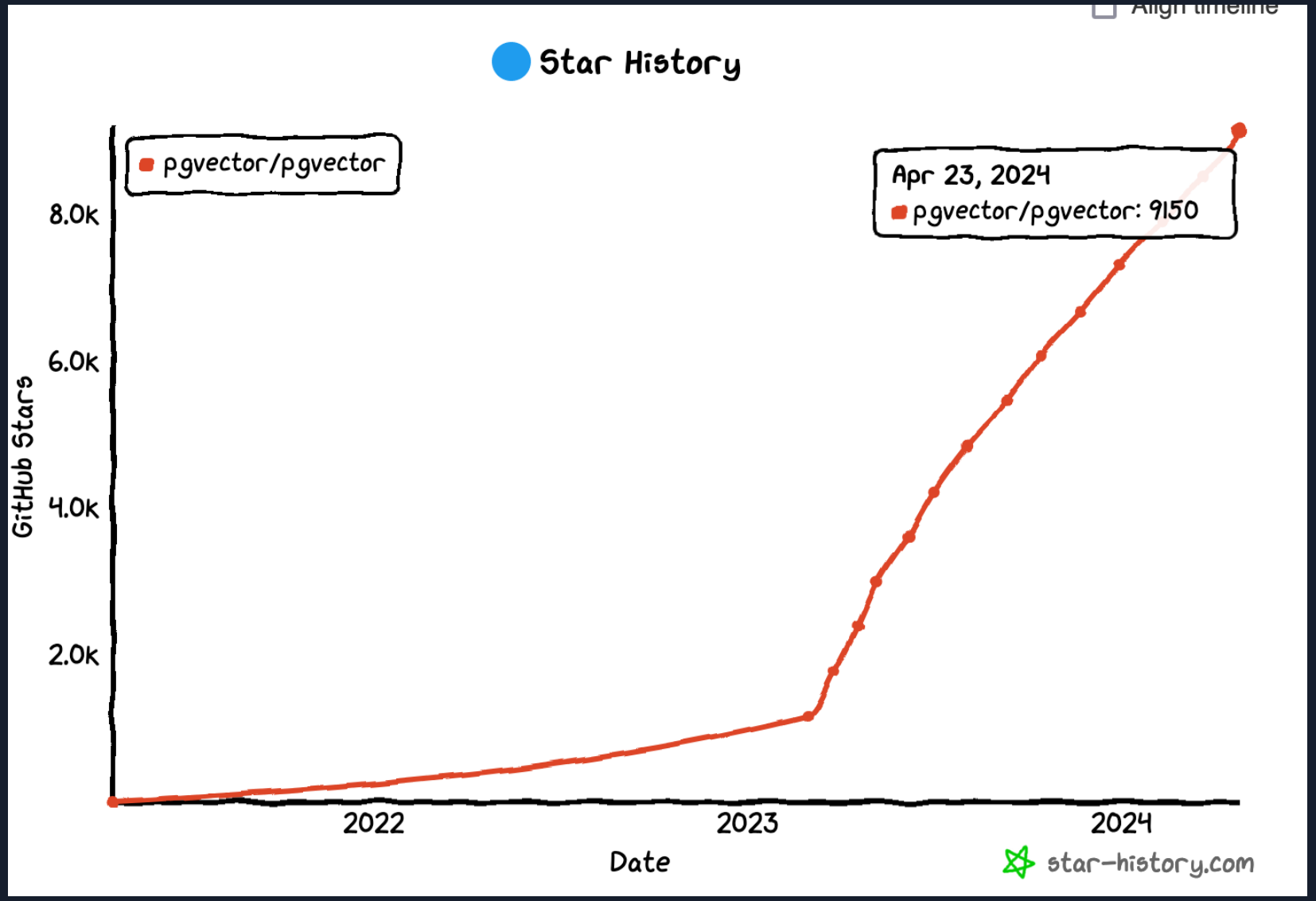
Exact nearest neighbor (K-NN)
Approximate nearest neighbor (ANN)

Co-locate with embeddings

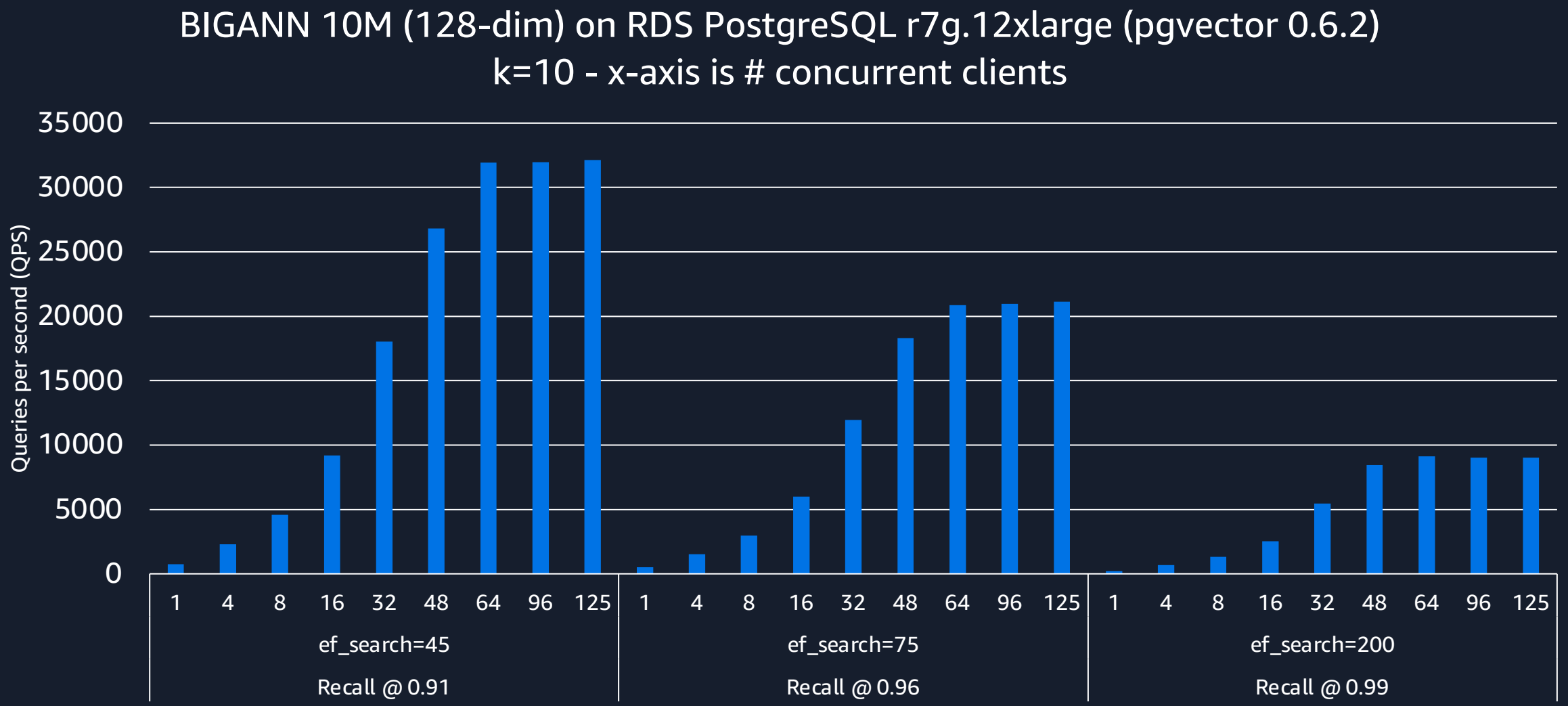
Distance operators (<->, <=>, <#>)

github.com/pgvector/pgvector

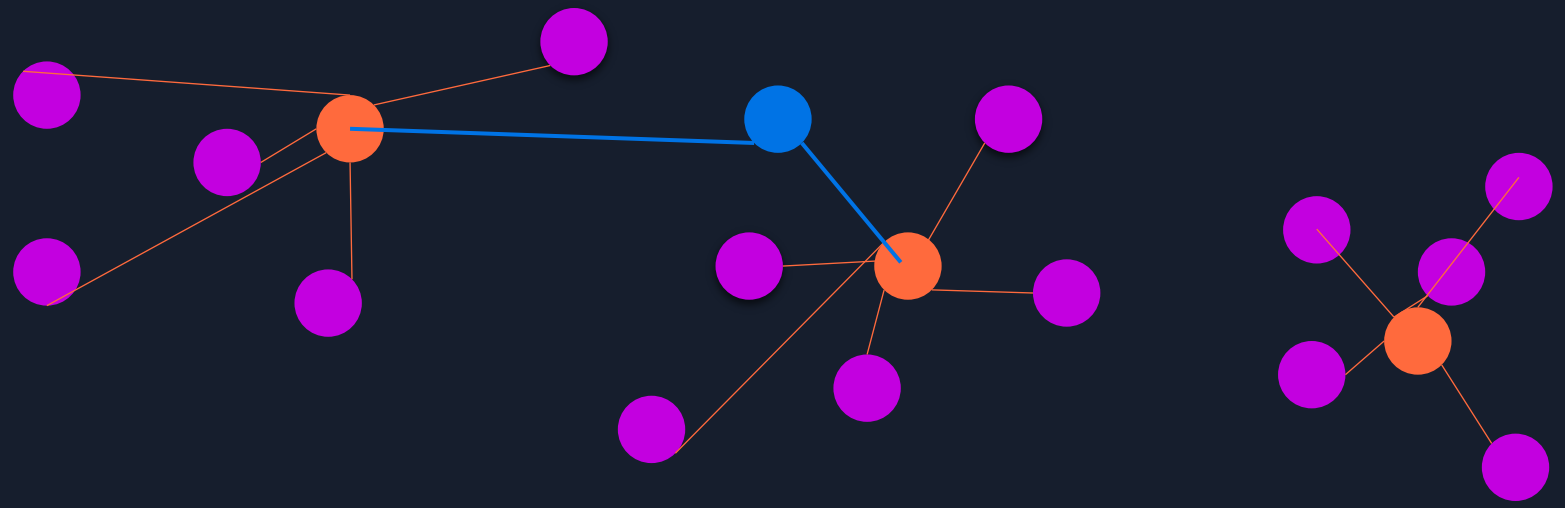
pgvector popularity



Does pgvector scale?



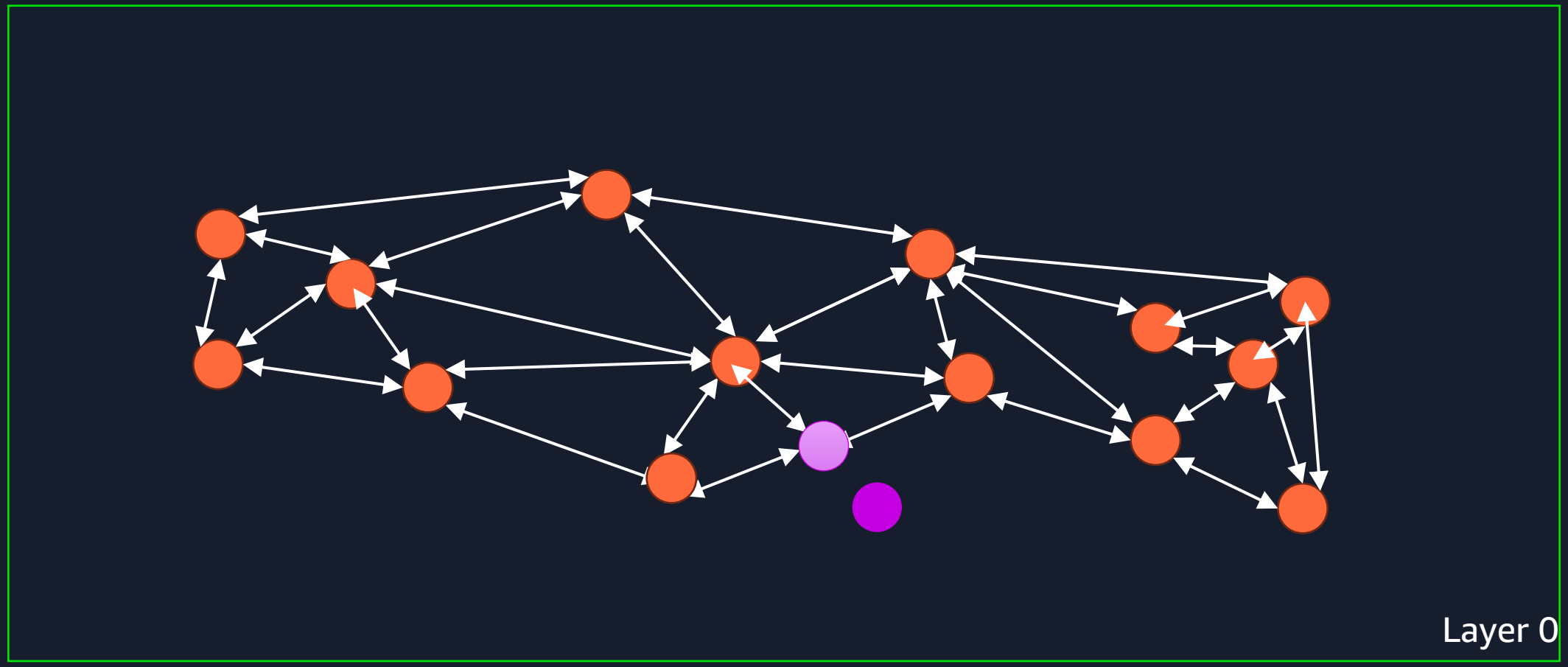
Querying an IVFFlat index



```
SET ivfflat.probes TO 2

SELECT id FROM products ORDER BY $1 <-> embedding LIMIT 3
```

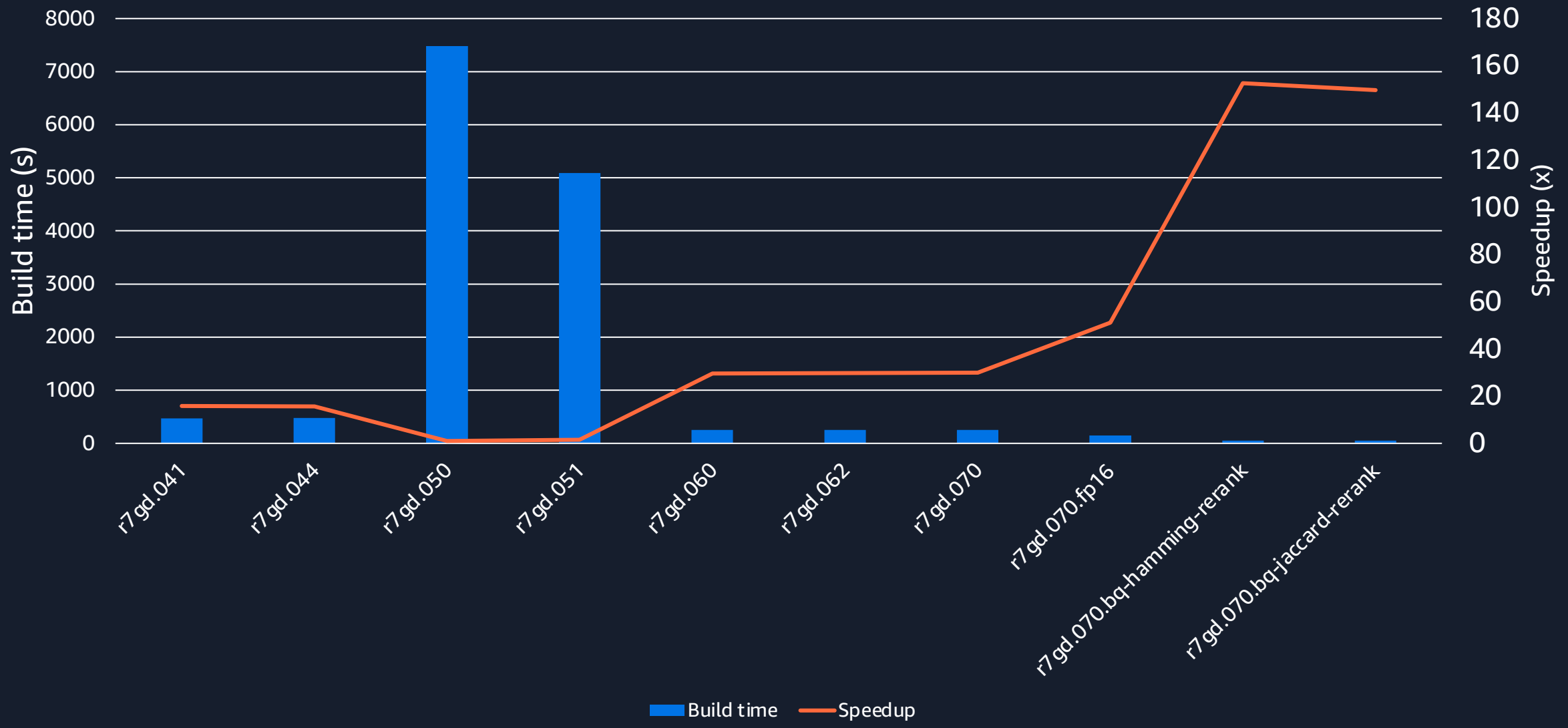
Querying an HNSW index

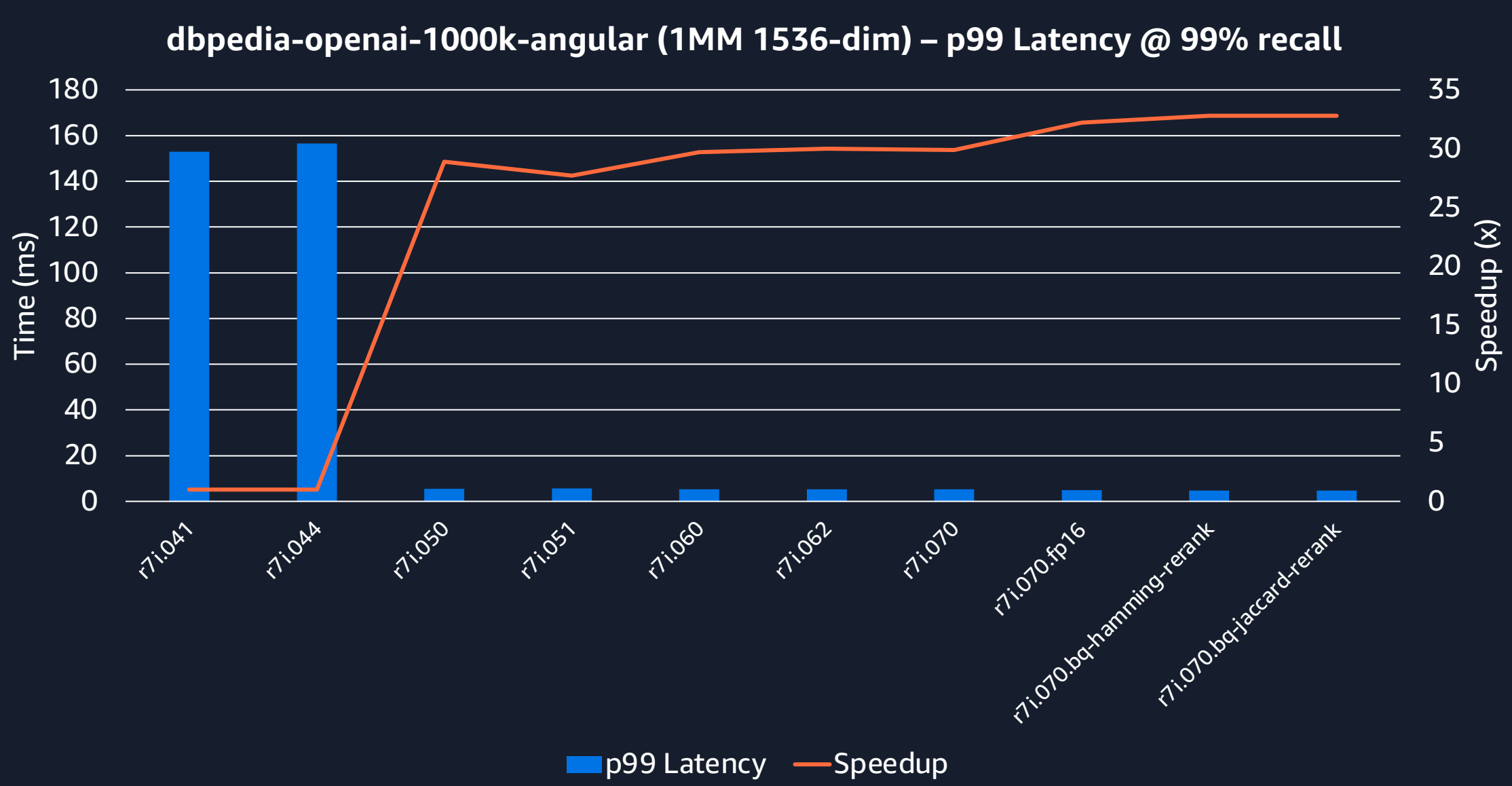


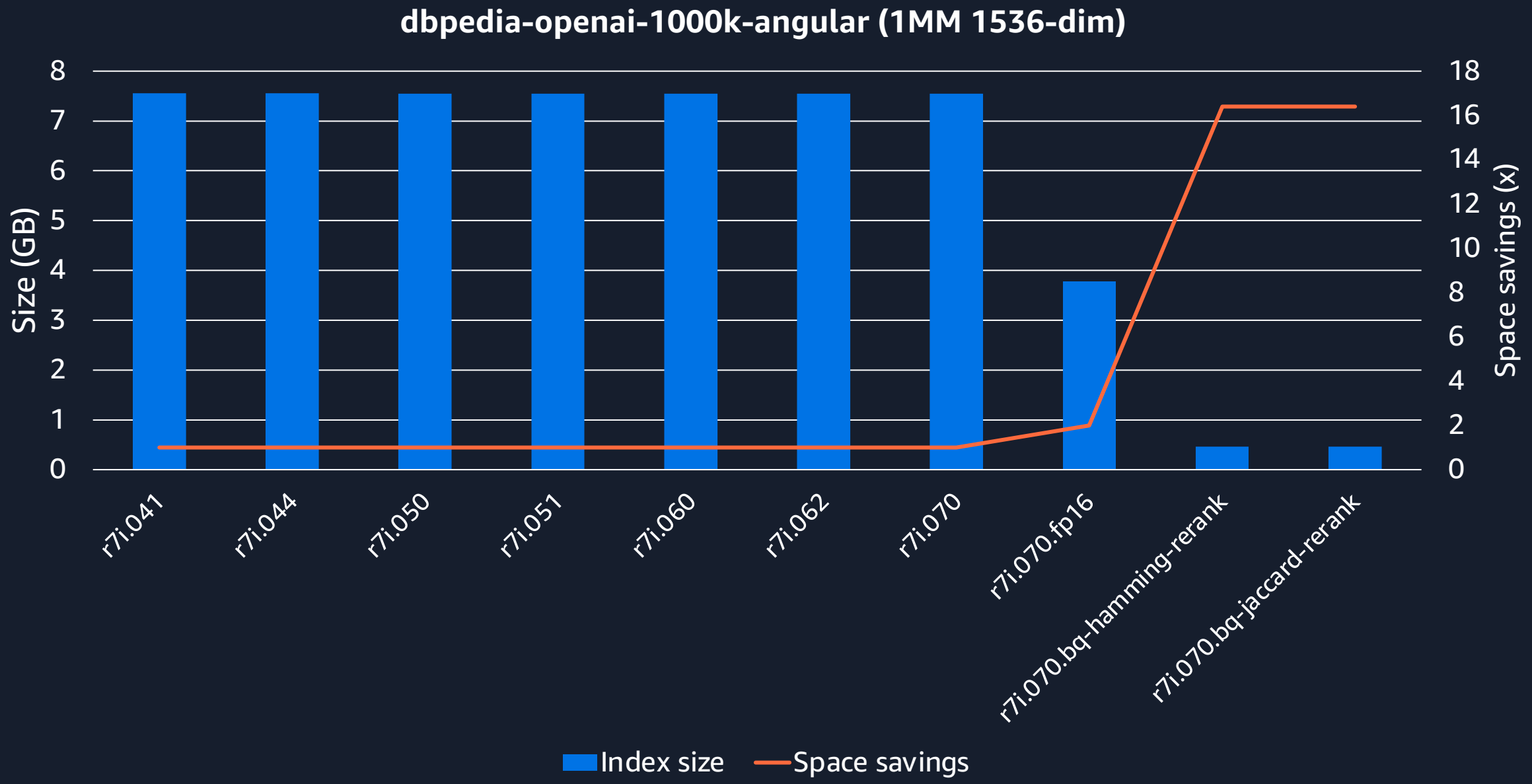
pgvector: Year-in-review timeline

- **v0.4.x** (1/2023 – 6/2023)
  - Improved IVFFlat plan costs
  - Increasing dimension of vectors stored in table + index
- **v0.5.x** (8/2023 – 10/2023)
  - Add HNSW index + distance function performance improvements
  - Parallel IVFFlat builds
- **v0.6.x** (1/2024 – 3/2024)
  - Parallel HNSW index builds + in-memory build optimizations
- **v0.7.x** (Unreleased, planner 4/2024)
  - halfvec (2-byte float), bit(n) index support, sparsevec (up to 1B dim)
  - Quantization (scalar/binary), Jaccard/hamming distance, explicit SIMD

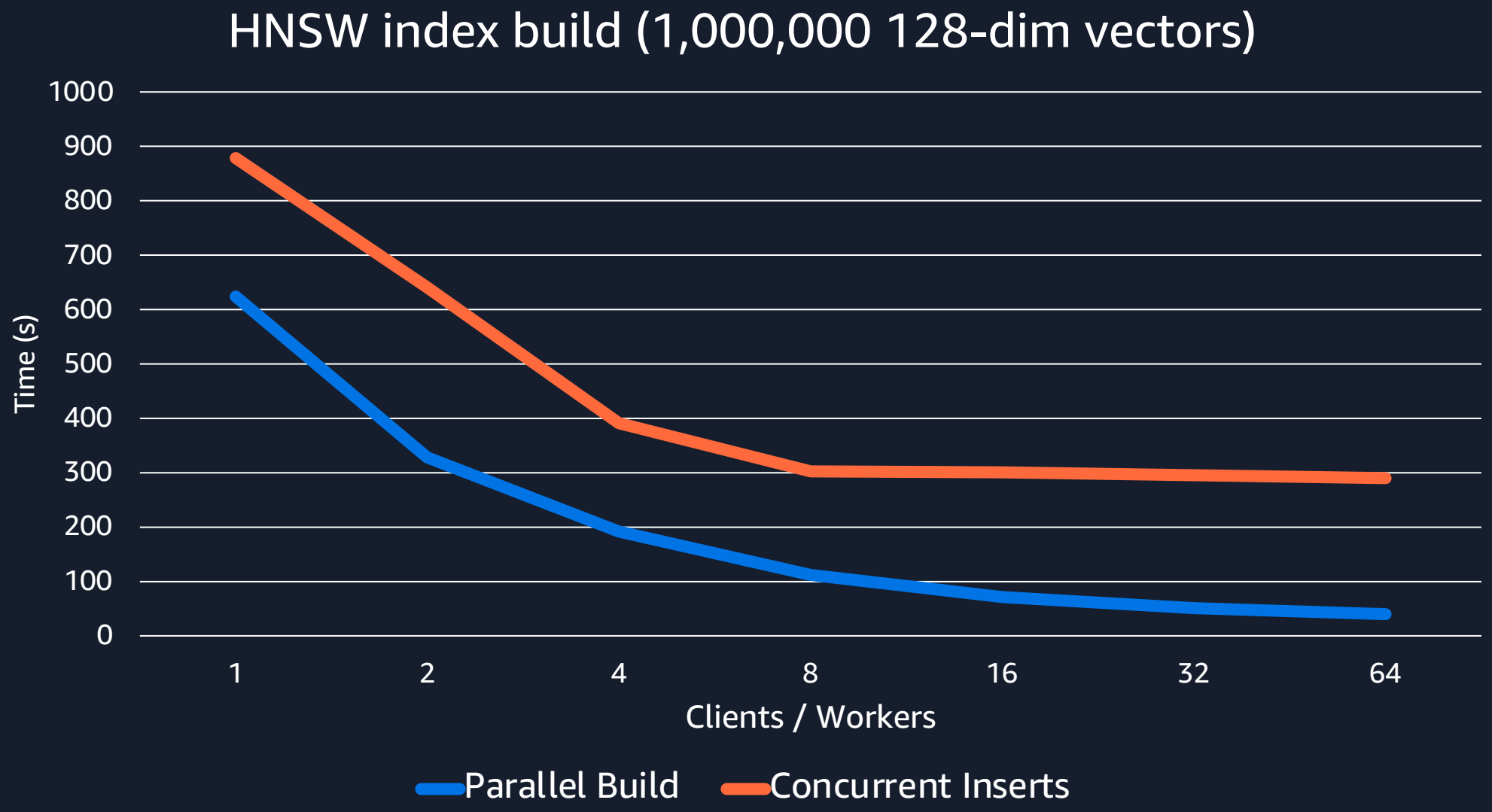
dbpedia-openai-1000k-angular (1MM 1536-dim) - Index Build Time



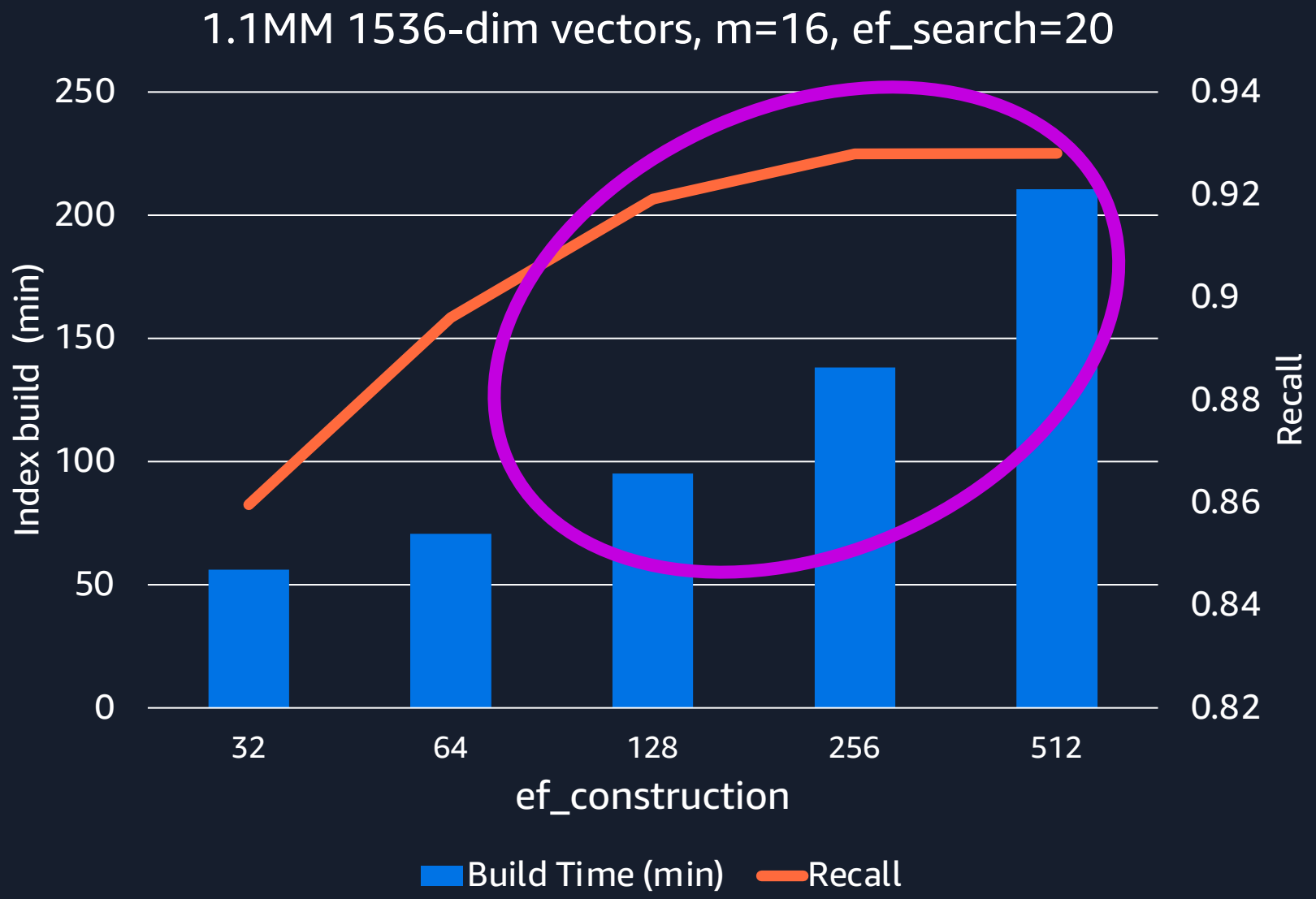




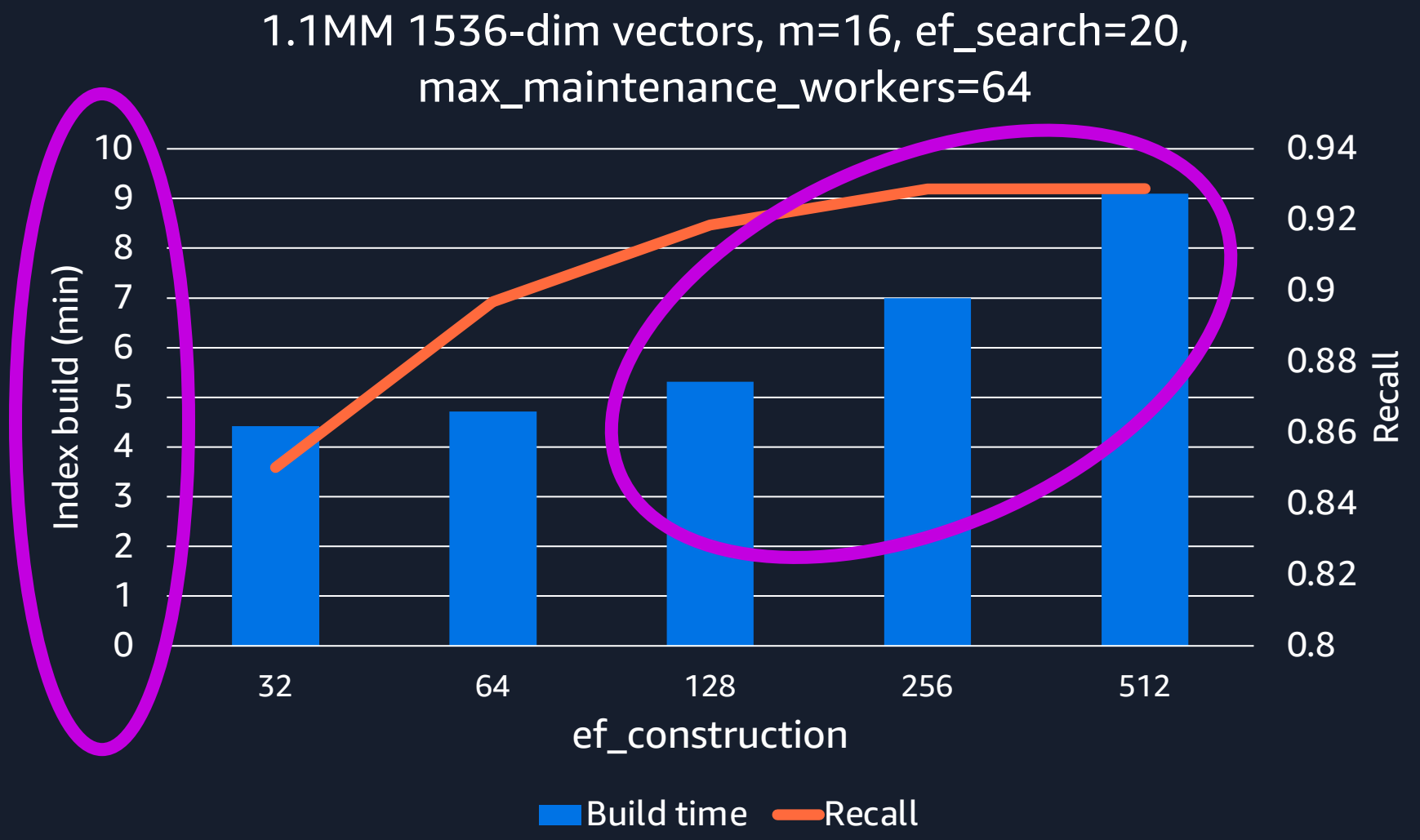
Impact of parallelism on HNSW build time



Choosing m and ef_construction (serial)



Choosing m and ef_construction (parallel)



pgvector roadmap

- Enhanced index-based filtering ("pre-filtering") (in progress)
- Additional quantization techniques (integer, product)
- Parallel query
- Improvements to PostgreSQL async query pushdown for federated sorts

Thank you!

Jonathan Katz

jkatz@amazon.com

@jkatz05