Are There Fundamental Limitations in Supporting Vector Data Management in Relational Databases?

A Case Study of PostgreSQL

Yunan Zhang · Shige Liu · Jianguo Wang

ICDE 2024

Presenter: Lim Yuan Jee

National Cheng Kung University

國立成功大學
National Cheng Kung University

Discussion topics

- Introduction
- Methodology
  - Dataset Used
- Key Findings
  - Index Construction
  - Index Size
  - QPS Performance
- Insights
  - Root Causes
  - Impact of SGEMM
  - Impact of parallelism
- Conclusion

Discussion topics

- Introduction
- Methodology
  - Dataset Used
- Key Findings
  - Index Construction
  - Index Size
  - QPS Performance
- Insights
  - Root Causes
  - Impact of SGEMM
  - Impact of parallelism
- Conclusion

Introduction

- Vector data is widely used in ML, IR, and data science.
- Comparison between:
  - Specialized Vector Databases (e.g., Faiss, Milvus).
  - Generalized Vector Databases (e.g., PASE, pgvector).
- Motivation: Are there fundamental limitations in relational database to support vector data management.
- Research Question: Can relational databases effectively support vector data management?

Discussion topics

- Introduction
- Methodology
  - Dataset Used
- Key Findings
  - Index Construction
  - Index Size
  - QPS Performance
- Insights
  - Root Causes
  - Impact of SGEMM
  - Impact of parallelism
- Conclusion

Methodology

- Approach: Compare relational database against specialized vector database
  - PASE (PostgreSQL)
  - FAISS
- Metrics:
  - Index Construction Time
  - Query Time
  - Index Size
  - Recall
- Dataset: Include benchmarks like SIFT1M, GIST1M, and Deep10M

Datasets used

Dataset	# Dimensions	# Vectors	# Queries
SIFT1M [51]	128	1,000,000	10,000
GIST1M [51]	960	1,000,000	1,000
Deep1M [8]	256	1,000,000	1,000
SIFT10M [51]	128	10,000,000	10,000
Deep10M [8]	96	10,000,000	10,000
TURING10M [52]	100	10,000,000	10,000

Discussion topics

- Introduction
- Methodology
  - Dataset Used
- Key Findings
  - Index Construction
  - Index Size
  - QPS Performance
- Insights
  - Root Causes
  - Impact of SGEMM
  - Impact of parallelism
- Conclusion

Key Findings: Index Construction

- Observations:
  - PASE is 6x-80x slower than Faiss for index construction.
  - Reason: Lack of SGEMM optimization in PASE.

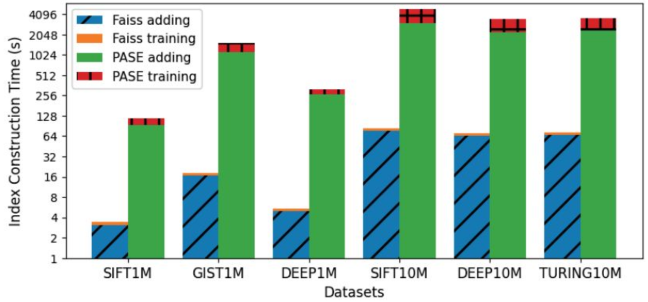

Fig. 3: Index Construction Time for IVF_FLAT

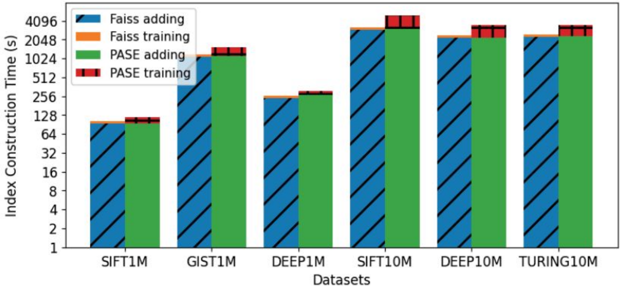

Fig. 4: Index Construction Time for IVF_FLAT Without SGEMM

Discussion topics

- Introduction
- Methodology
  - Dataset Used
- Key Findings
  - Index Construction
  - Index Size
  - QPS Performance
- Insights
  - Root Causes
  - Impact of SGEMM
  - Impact of parallelism
- Conclusion

Key Findings: Index Size

- Observations:
  - PASE consumes 3x-13x more space for HNSW due to page-based storage.
  - Minimal difference in index size for IVF FLAT and IVF PQ.

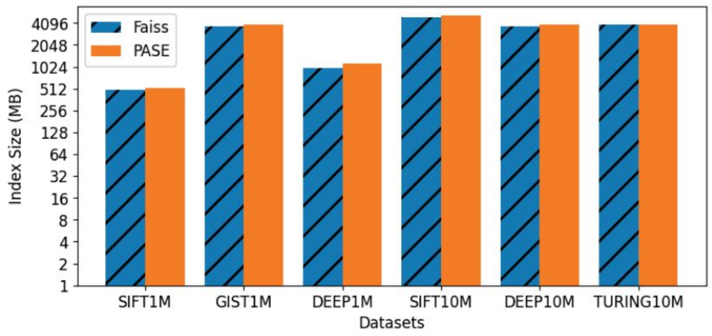

Fig. 11: Index Size for IVF_FLAT

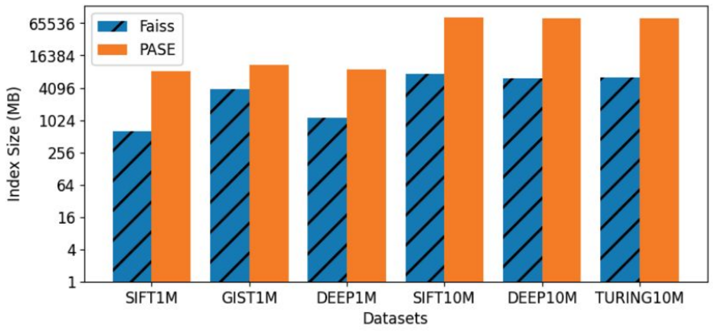

Fig. 13: Index Size for HNSW

Discussion topics

- Introduction
- Methodology
    - Dataset Used
- Key Findings
    - Index Construction
    - Index Size
    - QPS Performance
- Insights
    - Root Causes
    - Impact of SGEMM
    - Impact of parallelism
- Conclusion

Key Findings: QPS Performance

- Observations:
  - PASE query time is 2x-7x slower than Faiss.
  - Performance bottlenecks: Memory management, top-k heap size.

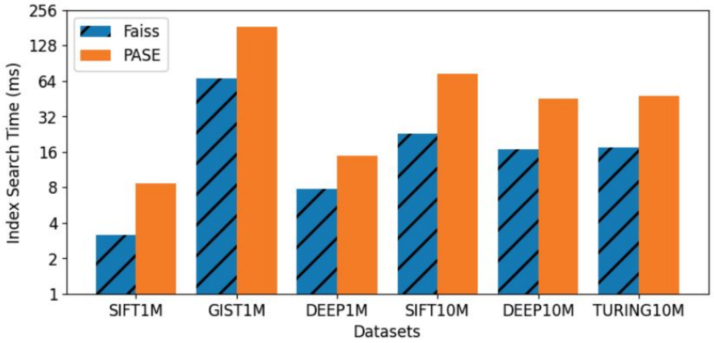

Fig. 14: Search Time for IVF_FLAT

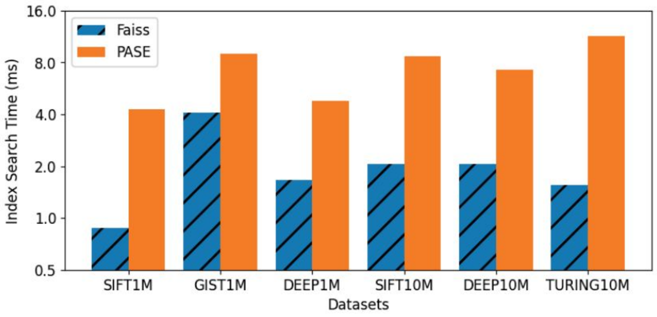

Fig. 17: Search time for HNSW

Discussion topics

- Introduction
- Methodology
  - Dataset Used
- Key Findings
  - Index Construction
  - Index Size
  - QPS Performance
- Insights
  - Root Causes
  - Impact of SGEMM
  - Impact of parallelism
- Conclusion

Insights: Root Causes

- Root causes of performance gap
  - SGEMM Optimization
  - Memory Management
  - Parallel Execution
  - Memory centric Page Structure
  - K-means implementation
  - Heap Size in Top-k Computation.
  - Precomputed Table Implementation

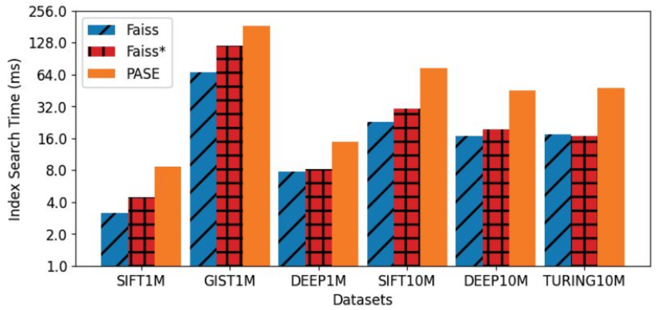

Fig. 15: Search Time for IVF_FLAT With Replaced Centroids (Faiss$*$)

Discussion topics

- Introduction
- Methodology
  - Dataset Used
- Key Findings
  - Index Construction
  - Index Size
  - QPS Performance
- Insights
  - Root Causes
  - Impact of SGEMM
  - Impact of parallelism
- Conclusion

Insights: Impact of SGEMM (Single Precision General Matrix Multiplication)

- SGEMM plays a crucial role in optimizing computations within vector databases, particularly for distance calculations and K-means clustering.
- Why is it important?
    - Matrix Operations in Vector Databases
- Challenges Without SGEMM:
    - Computations are performed using basic loops or less efficient matrix operations.
    - ex: Faiss uses SGEMM to transform the distance calculation problem into a matrix-matrix multiplication

Discussion topics

- Introduction
- Methodology
  - Dataset Used
- Key Findings
  - Index Construction
  - Index Size
  - QPS Performance
- Insights
  - Root Causes
  - Impact of SGEMM
  - Impact of parallelism
- Conclusion

Insights: Impact of Parallelism

- Index Construction
  - PASE lacks efficient multi-threading during the index construction phase
- Query Execution
  - PASE Query Bottleneck
    - Searching multiple buckets simultaneously or updating shared data structures (e.g., heaps) become bottlenecks.
    - Faiss utilizes local heaps for parallel searching within buckets.
- Results
  - Faiss demonstrates near-linear scalability with thread count during query execution

Discussion topics

- Introduction
- Methodology
  - Dataset Used
- Key Findings
  - Index Construction
  - Index Size
  - QPS Performance
- Insights
  - Root Causes
  - Impact of SGEMM
  - Impact of parallelism
- Conclusion

Conclusion

- Conclusion: No fundamental limitations; performance gaps are due to implementation.
- Future Work:
  - Build a memory-centric generalized vector database.
  - Improve algorithmic and parallelism support.
    - High priority for SGEMM
    - Parallelism
    - Optimized top k computation