Discussion and Analysis of New Vector Database Benchmark in MLPerf Storage

Sayali Shirode
Senior Systems Performance Engineer

micron Intelligence Accelerated™

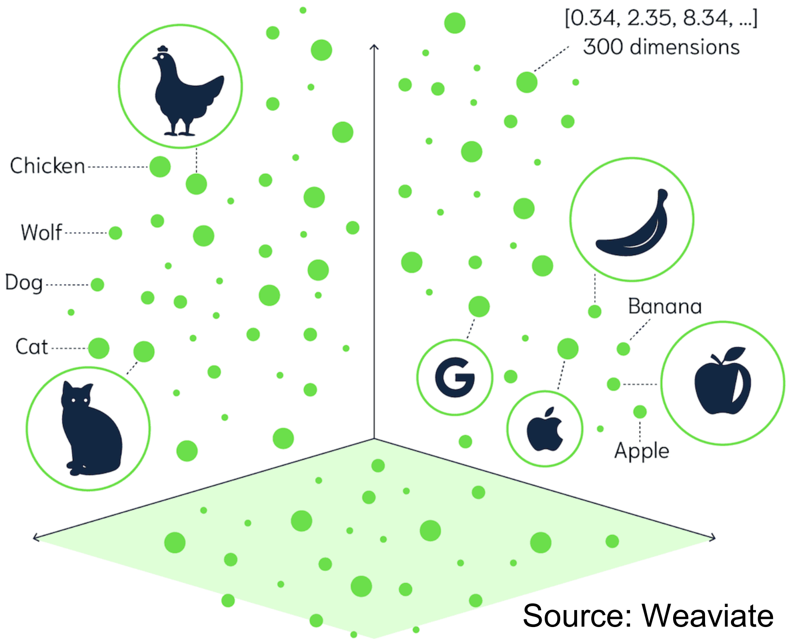
FMS

Today's Agenda

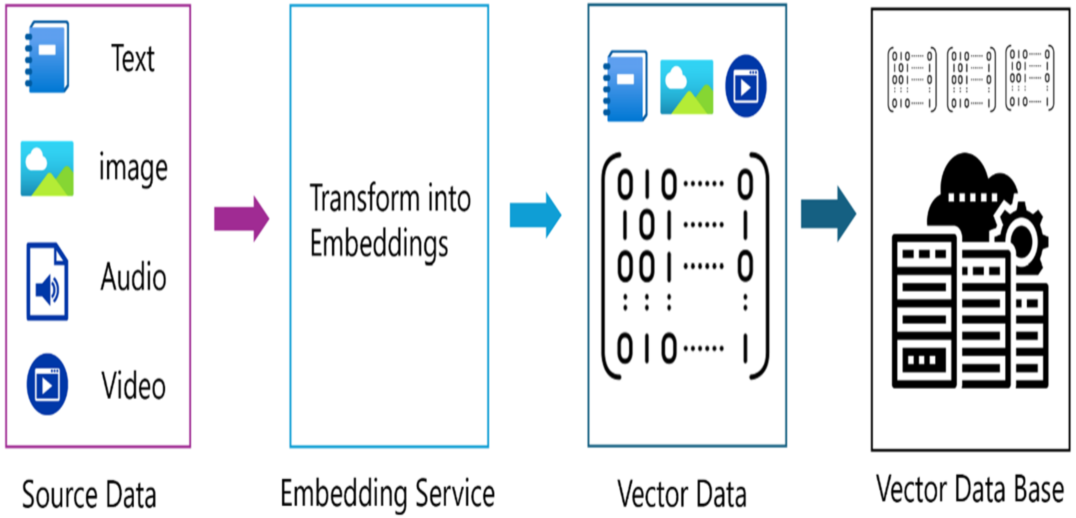
- ❑What is a Vector Database?
- ❑Why Vector DBs in MLPerf?
- ❑Results
- ❑Q&A
- ❑Takeaways

What is a Vector Database?

- Vector databases do very fast “approximate” nearest neighbor search
  - Useful when perfect accuracy is not a requirement
  - Example: Recommender systems and Retrieval Augmented Generation for LLMs
- Widely used in the industry
- Datasets are getting larger YoY which is driving innovations in using storage for large indexes instead of memory



Source: Weaviate



Source: vector-database

Why Vector DBs in MLPerf?

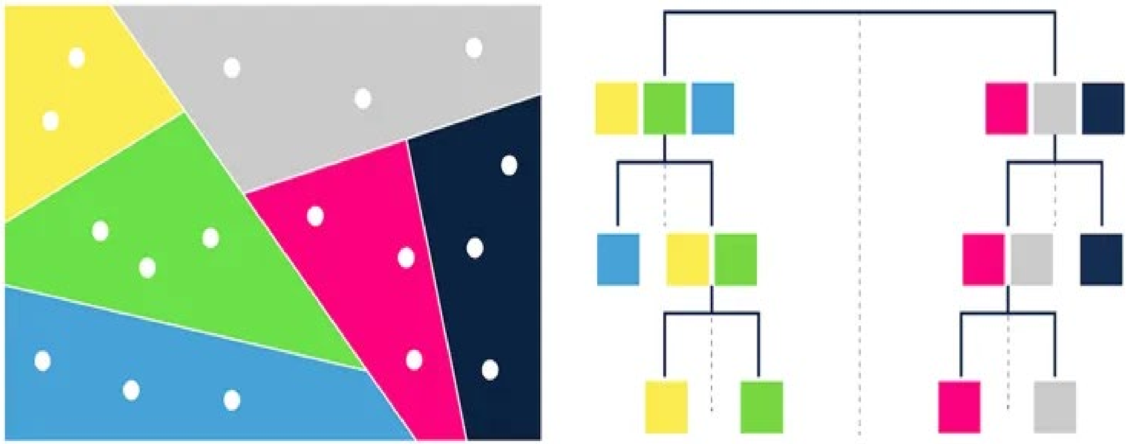
- MLPerf Benchmarks aim to represent modern use cases and drive development of the entire stack (including the AI frameworks)
- Wide adoption of VectorDBs and increasing size of datasets is shifting VectorDBs to be storage-sensitive workloads
- Existing VectorDB benchmarks are focused on measuring performance and ***accuracy*** of a given database without a capability to test at arbitrary scales
- MLPerf Storage(mlcommons) will include a VectorDB workload in the Next version
  - Sign up for the MLPerf mailing list to find out more about the PoC tools available and contribute to the rules and process

DiskANN

- **Disk-Based Storage**: Stores the bulk of vector data on disk, using memory for indexing a subset of the most relevant data.
- **I/O Optimization**: Employs techniques like multi-threading and asynchronous I/O to minimize latency due to disk access.
- **Hybrid Approach**: Combines in-memory indexing for a subset of vectors with disk-based storage for the majority.

Advantages

- **Cost-Effective**: Reduces the need for expensive RAM by leveraging disk storage.
- **Scalability**: Supports extremely large datasets that exceed memory limitations.



Disk-ANN: Bridging Scalability and Storage Efficiency

References:
vector-search-algorithms-knn-ann-and-disk-ann

Results

- Milvus Vector Database
  - 100 Million vectors
    - Randomly generated with a uniform distribution
  - Disk ANN index
    - 10 shards
    - Vector Dimension = 512
    - Data type = Float16

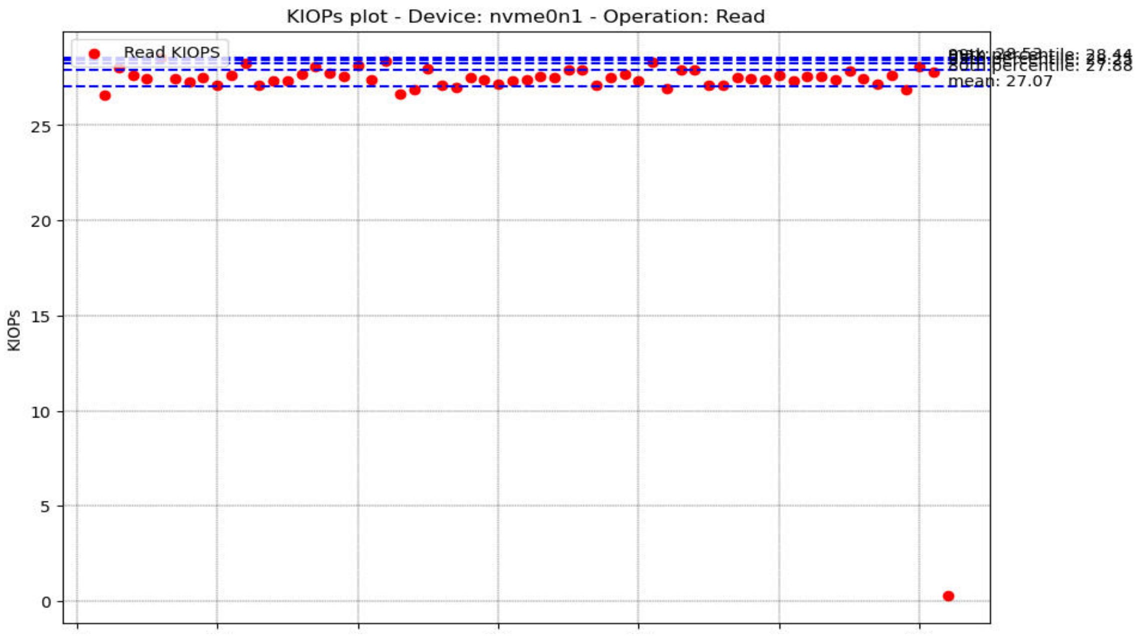
- Vector DB Benchmark
  - Randomly generate query vectors
  - Execute batches of queries in multiple processes
  - Measure throughput, latency, and QoS latencies

	1 process	64 processes	Difference
Throughput(queries/sec)	189	1374	7.3x
Read bandwidth(GB/s)	6.3	46.8	7.4x
Average latency(ms)	5	46	9.2x

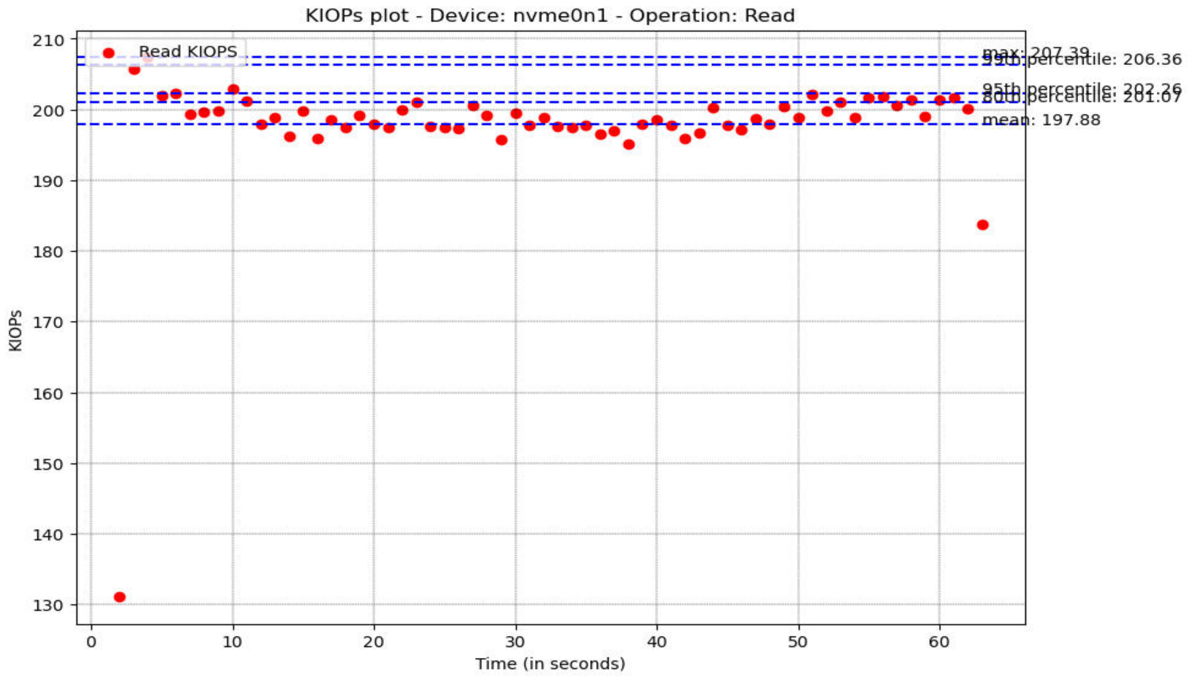
IOPS

DiskANN Index – 100 million vector, 10 shards, 512 dimensions

Batch size 1, process 1



Batch size 64, processes 64

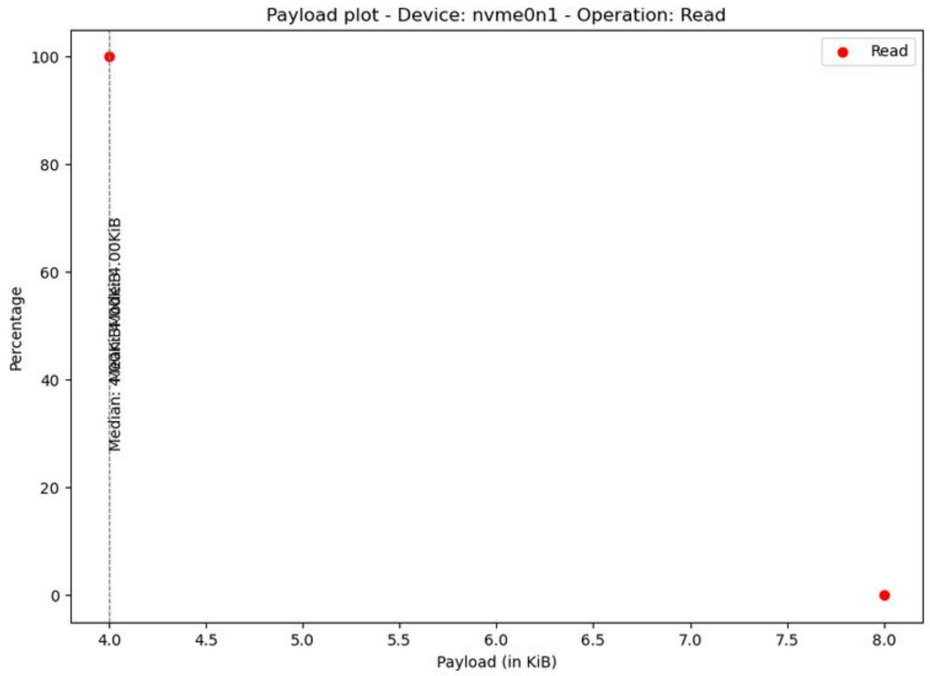


- KIOPS increases from 27k to 200k under load (~7.4x)

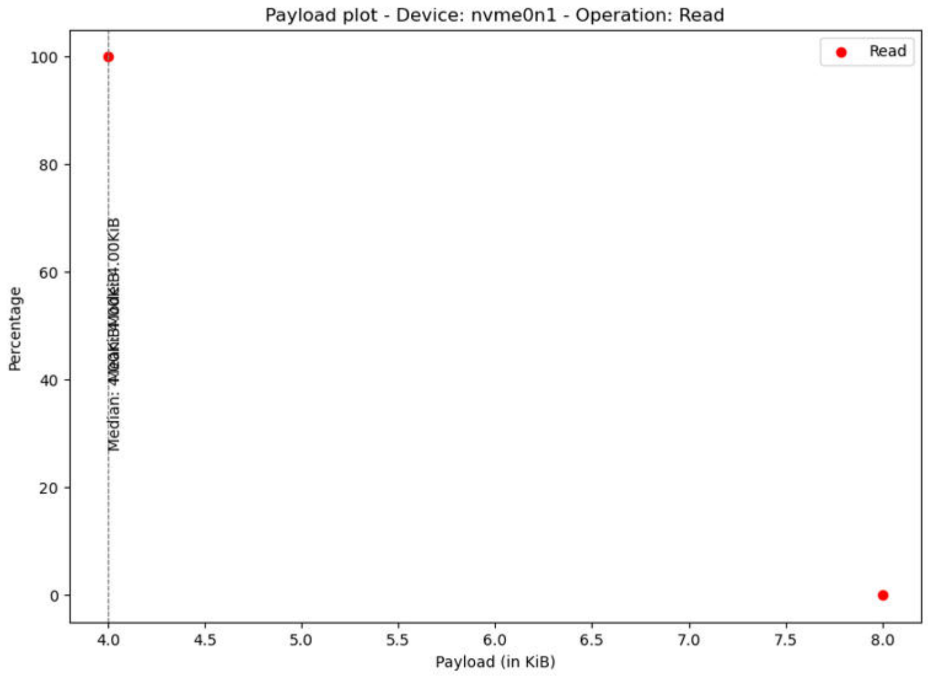
IO Size Distribution

DiskANN Index – 100 million vector, 10 shards, 512 dimensions

Batch size 1, process 1



Batch size 64, processes 64

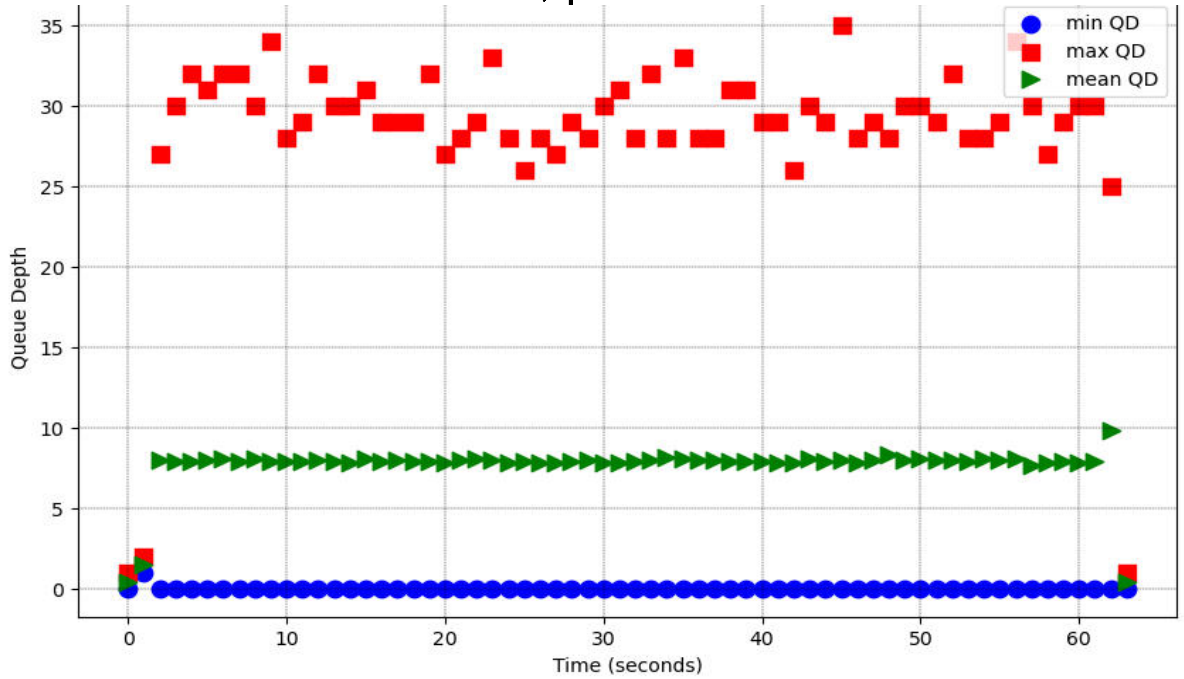


- Despite high IOPS, IO size remains constant at 4k which indicates no merged IOs which indicates a random-access pattern

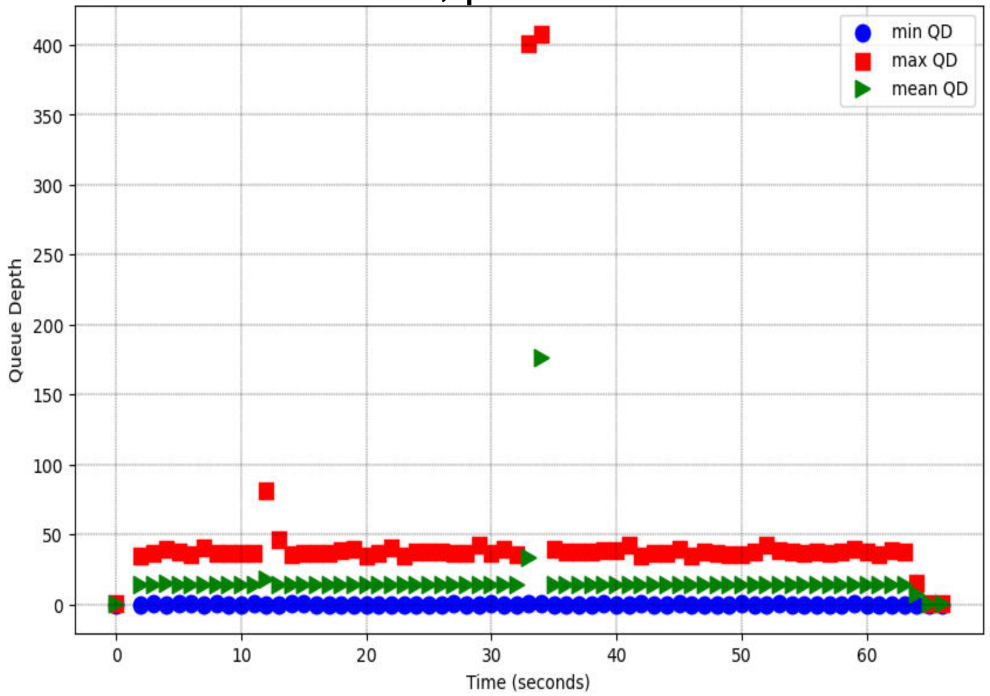
Queue Depth Over Time

DiskANN Index – 100 million vector, 10 shards, 512 dimensions

Batch size 1, process 1



Batch size 64, processes 64

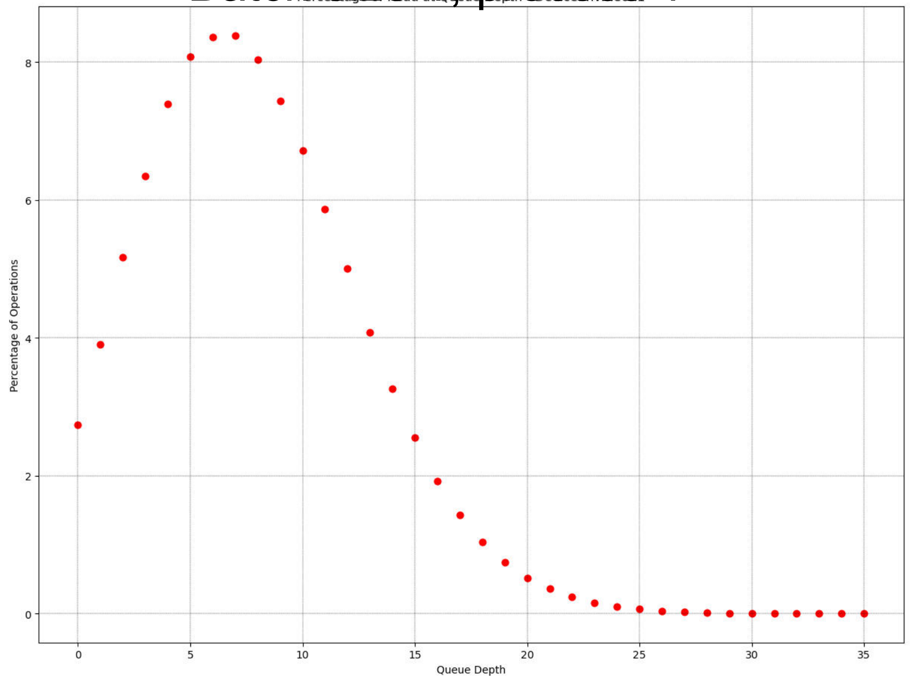


- Queue depth remains consistent over time
- “High” load to the database results in moderate load to storage as measured by Queue Depth
- Workload is likely QoS sensitive vs maximum IOPS dependent
- Opportunities may exist on the database stack or system design to optimize compute.

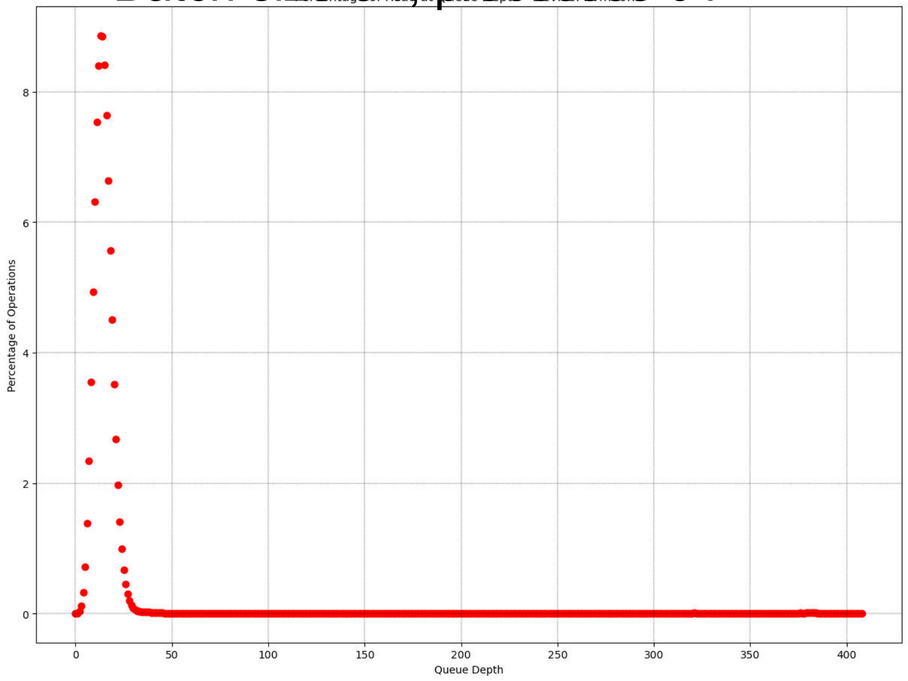
Queue Depth Over Time

DiskANN workload – 100 million 512 dimensions uniform

Batch size 1, process 1



Batch size 64, processes 64



- Queue depth remains consistent from low to high load.
- Even at a moderately loaded system, the database side experiences high load while the disk side remains moderate.
- There is still performance left on the disk.
- Opportunities may exist on the database stack or system design to optimize compute.

Takeaways

Integrate with MLPerf Storage vNEXT

Vector search is small I/Os

Vector search is random I/Os

QoS sensitive but not Throughput sensitive micron Intelligence Accelerated™

FMS