Efficient Vector Search on Disaggregated Memory with d-HNSW

Yi Liu*, **Fei Fang***, Chen Qian

University of California Santa Cruz

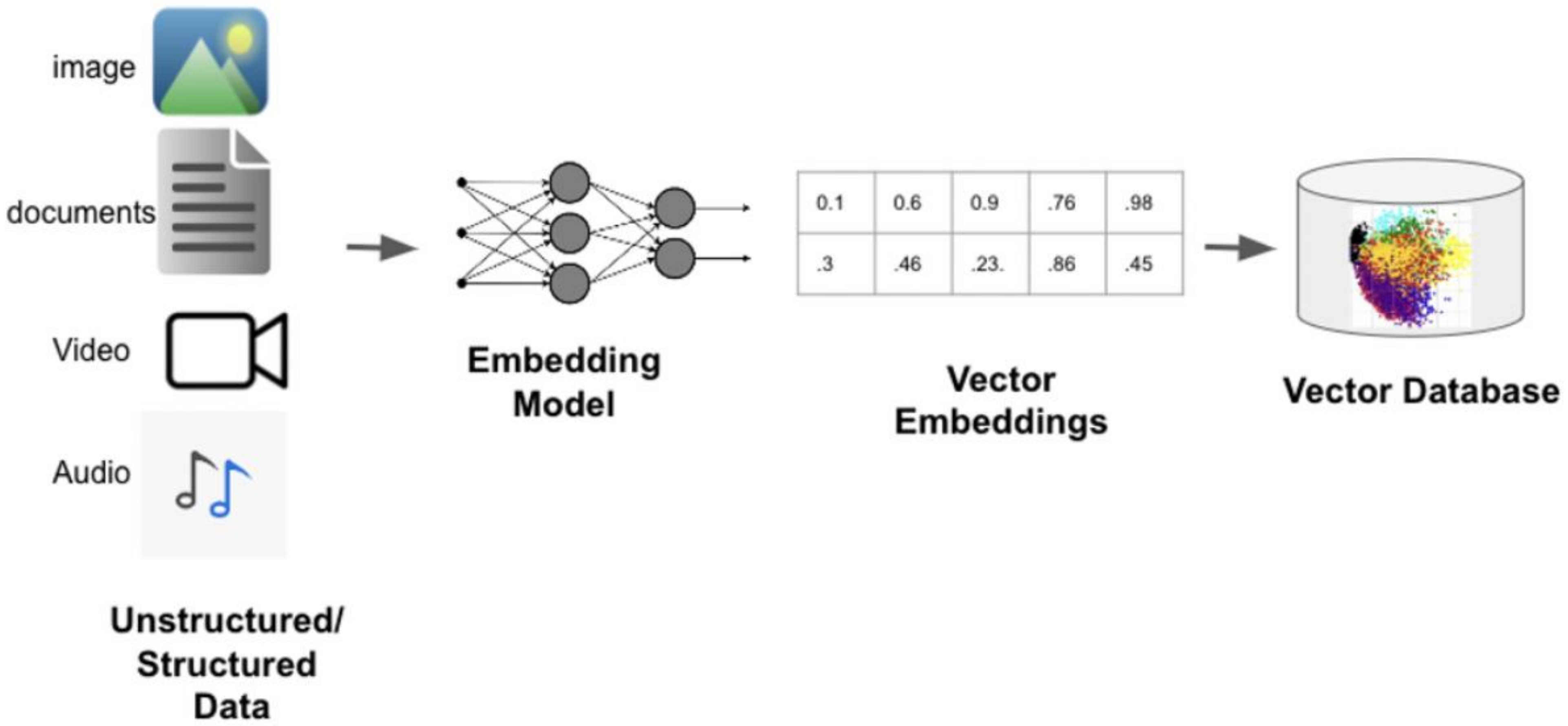
HotStorage 2025

The 17th ACM Workshop on
Hot Topics in Storage and File Systems

July 10-11
Boston, MA

Baskin Engineering
UC SANTA CRUZ

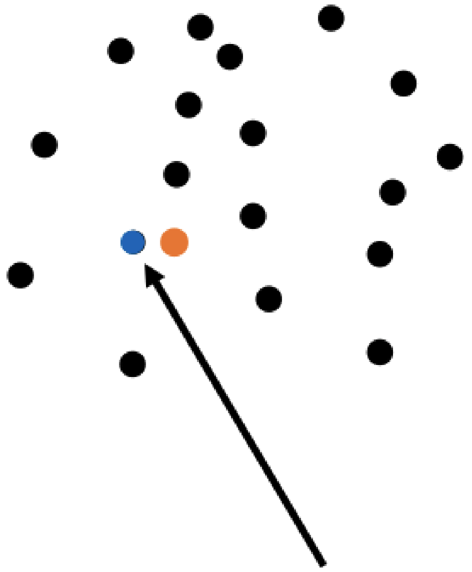
Vector DataBases



Vector Query

Given a query vector

Return Top-k Nearest vectors

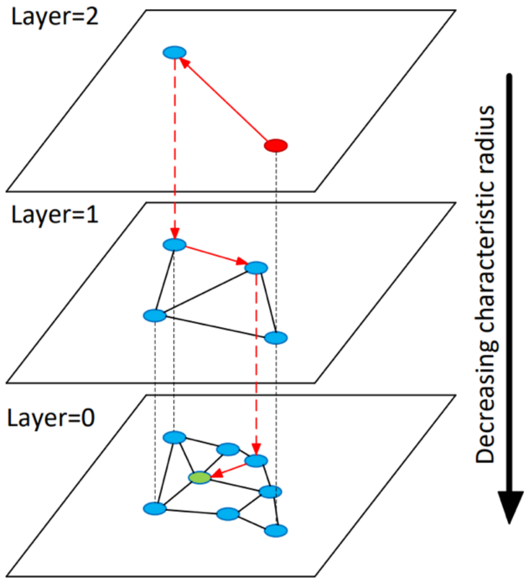


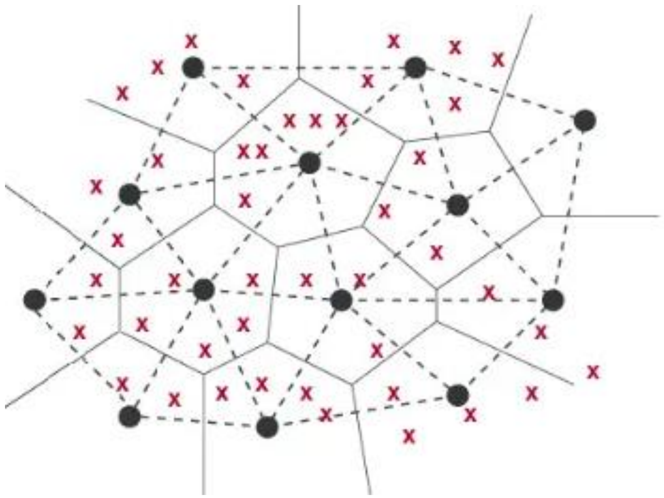
Top-1(k=1) Nearest Vector

Approximate Vector Query

Graph Index

Inverted Index



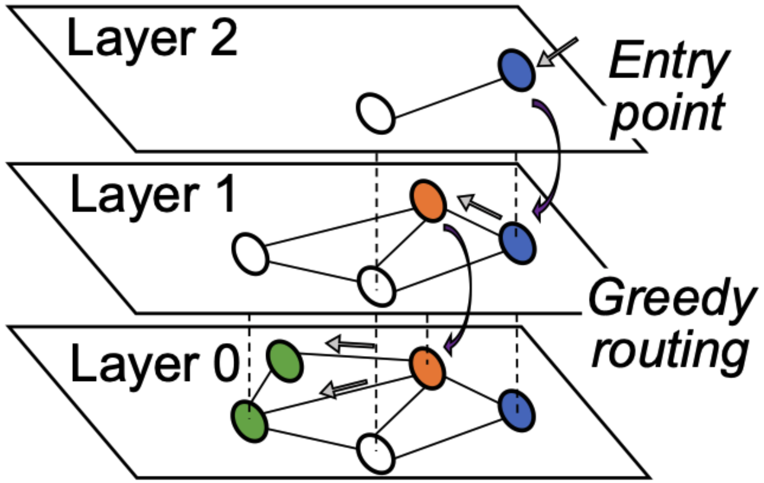


Hierarchical Navigable Small World (HNSW)

similarity search

trade off: latency vs accuracy

Robustness and high performance

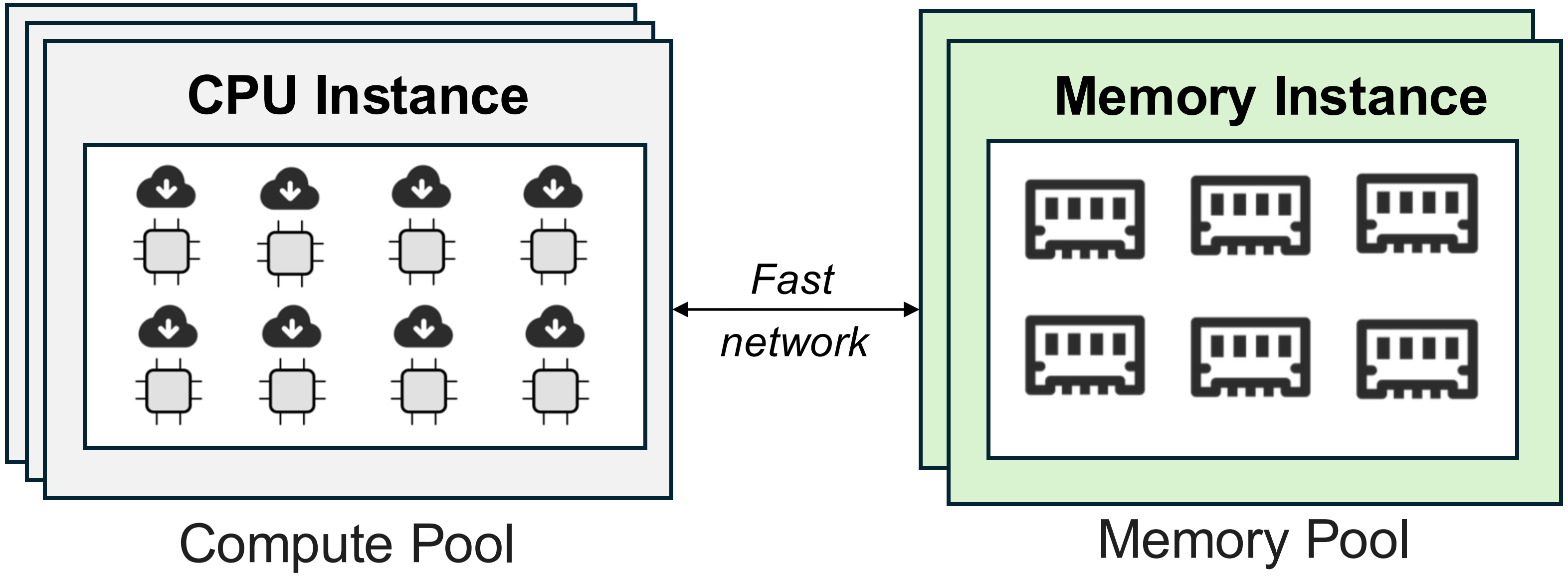


Disaggregated Memory Systems (DMS)

High resource utilization

Flexible hardware scalability

Efficient data sharing



RDMA-based Disaggregated Memory System

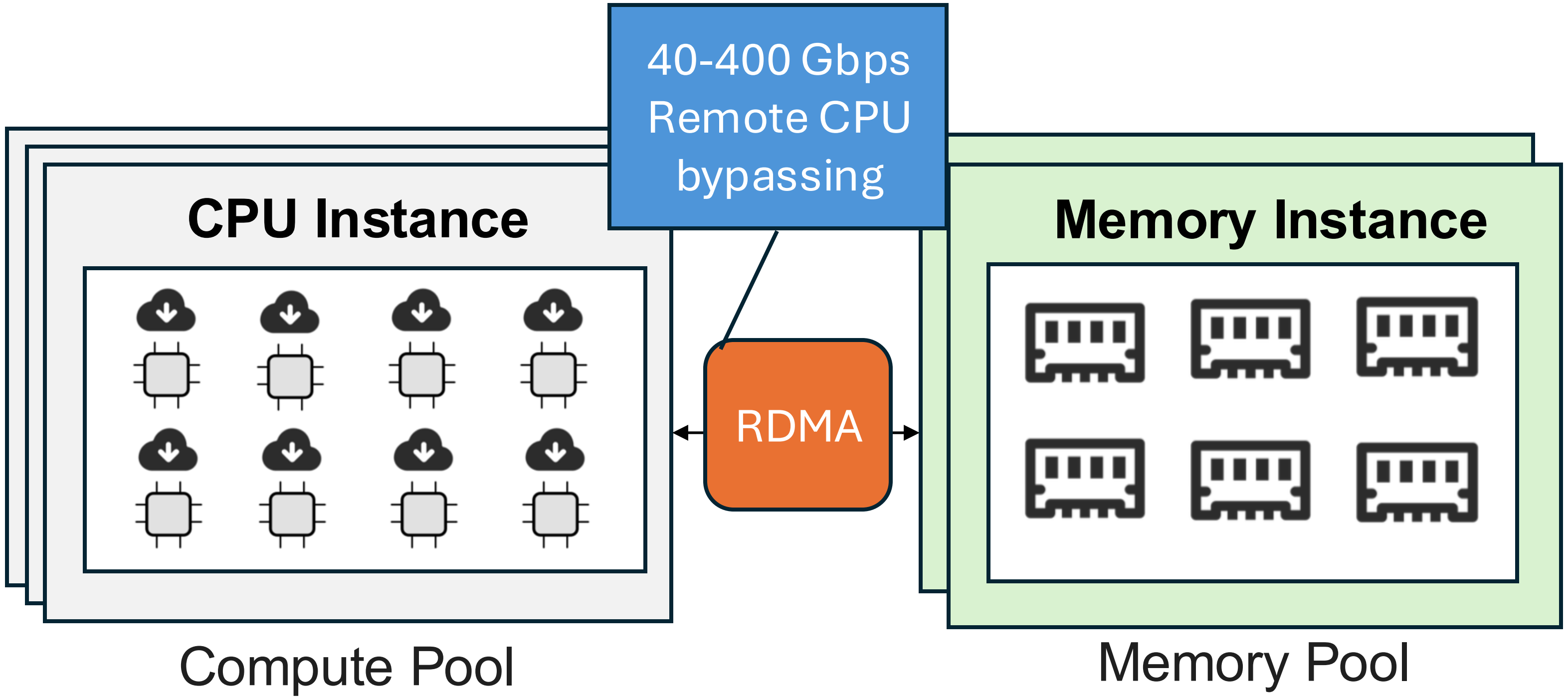
file system

key-value stores

transactional databases

...

vector databases



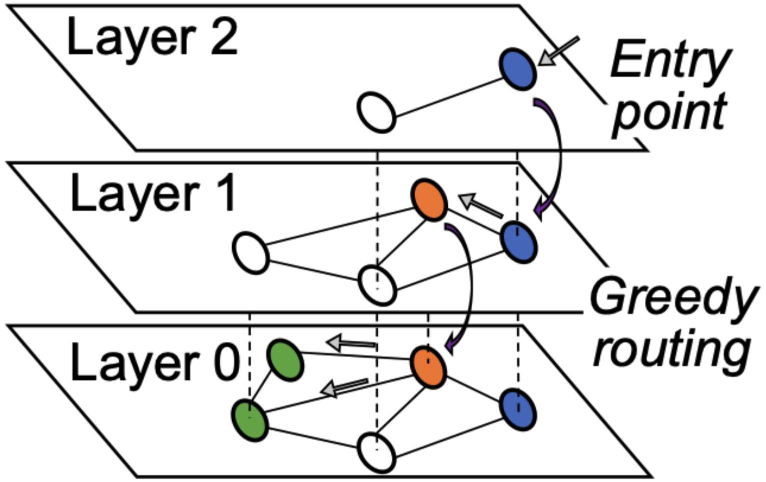
Vector Database on Disaggregated Memory System

- Challenge 1: How to reduce network round trips?
- Challenge 2: How to enable one-side insertions?
- Challenge 3: How to support efficient batched operations?

Reduce Network Round Trips Challenge

- greedy algorithm
- node represents vector
- unpredictable traversal path
- fetch each step

excessive round-trips

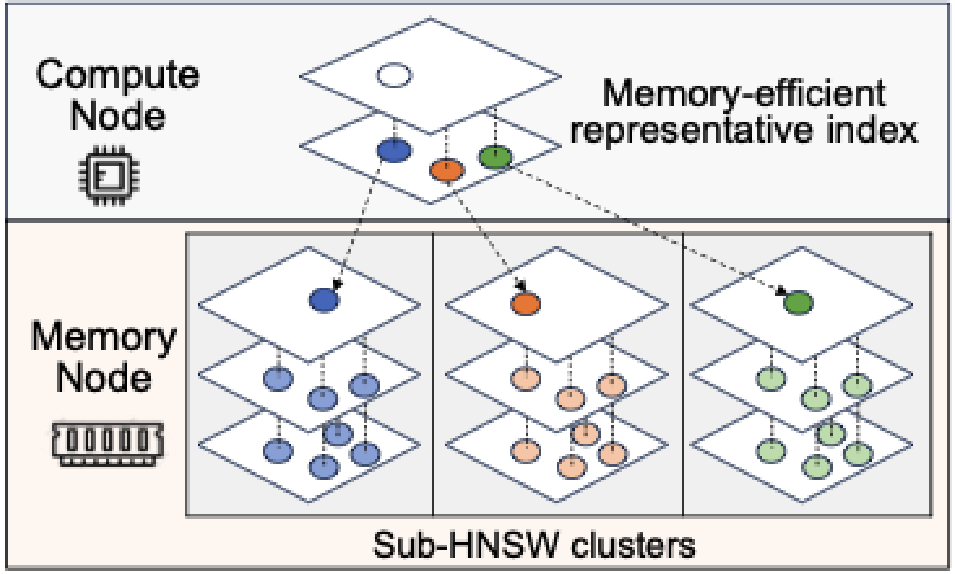


Reduce Network Round Trips Challenge

- Our approach: Representative index caching

a lightweight meta-HNSW

- **Minimizes network transfer** of irrelevant vectors
- **Reduces latency** and bandwidth usage
- **Preserves high recall**



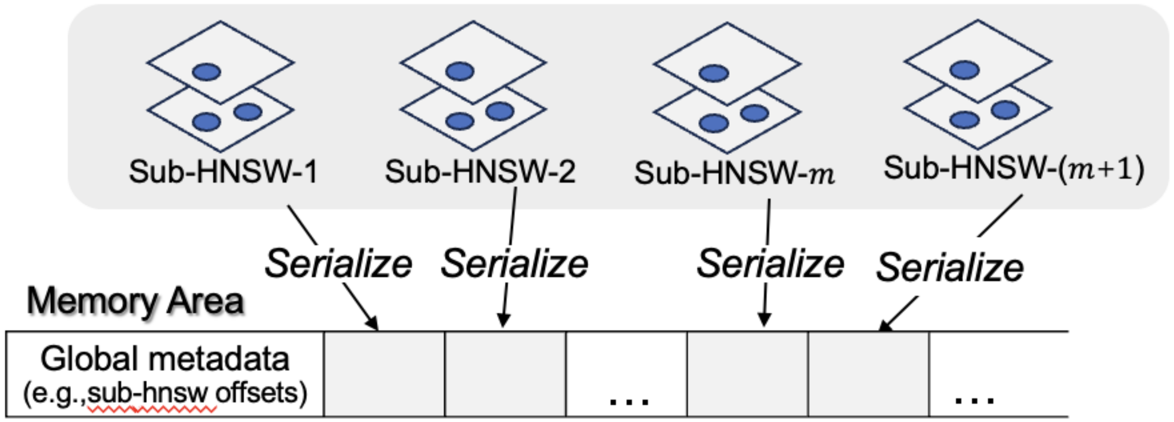
One-side Insertions Challenge

compactly serialized

new vector insertion

scatters vectors

High latency from fragmented memory access and multiple RDMA round-trips.

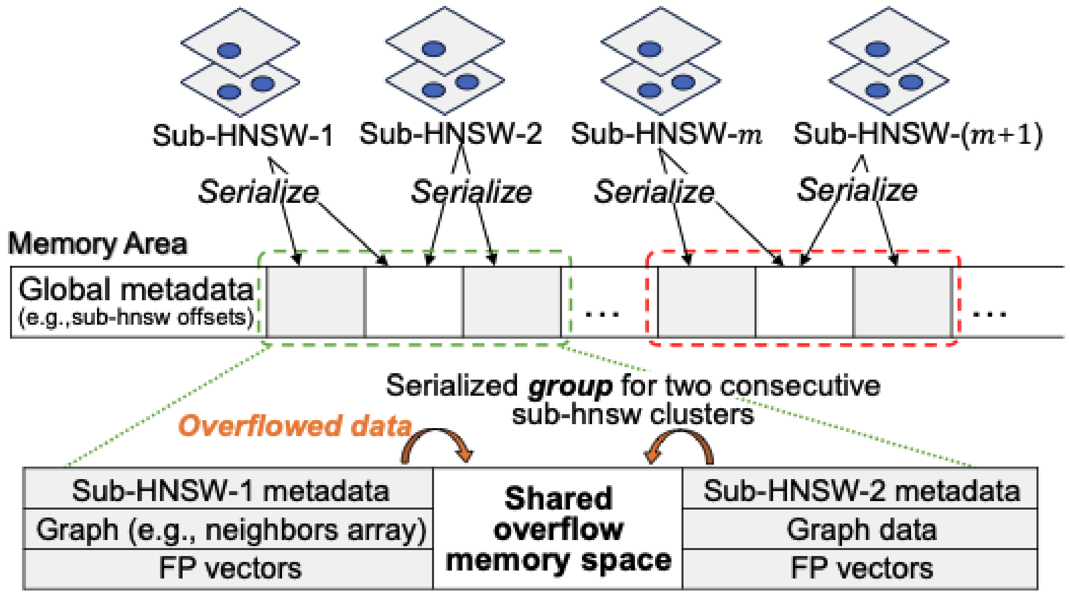


One-side Insertions Challenge

- Our approach: RDMA-friendly graph index storage layout in remote memory.

Internal gap

Shared overflow memory space



- Enables **single-round RDMA reads** for query
- **Avoid fragmented memory access**
- **Preserves high throughput** under insertions
- **Balances insertions between two sub-HNSW**

Efficient Batched Operations Challenge

search top-k candidates across b closest sub-HNSWs

can only cache limited sub-HNSWs

redundant transfers and **high bandwidth usage.**

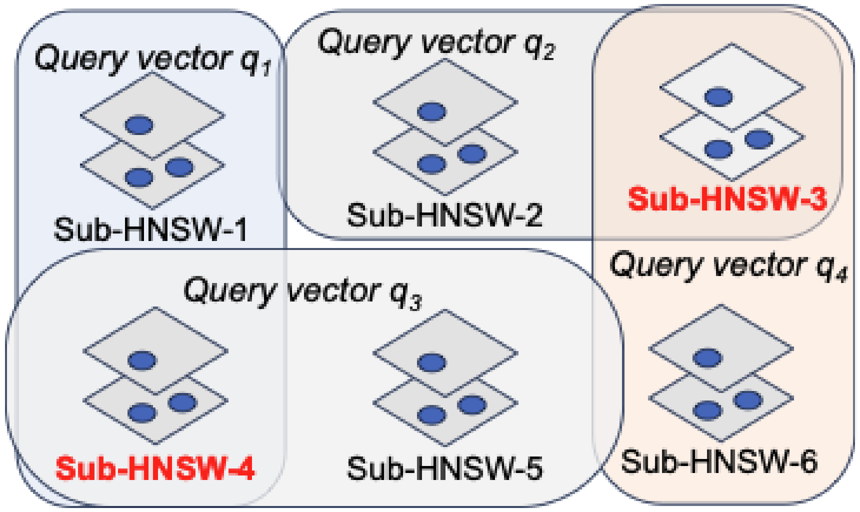
Efficient Batched Operations Challenge

- Our approach: Query-aware batched data loading

required sub-HNSW **only once per batch**

doorbell batching

- **Minimizes redundant data transfers**
- **Reduces network round-trips** with batching
- **Preserves cache efficiency**

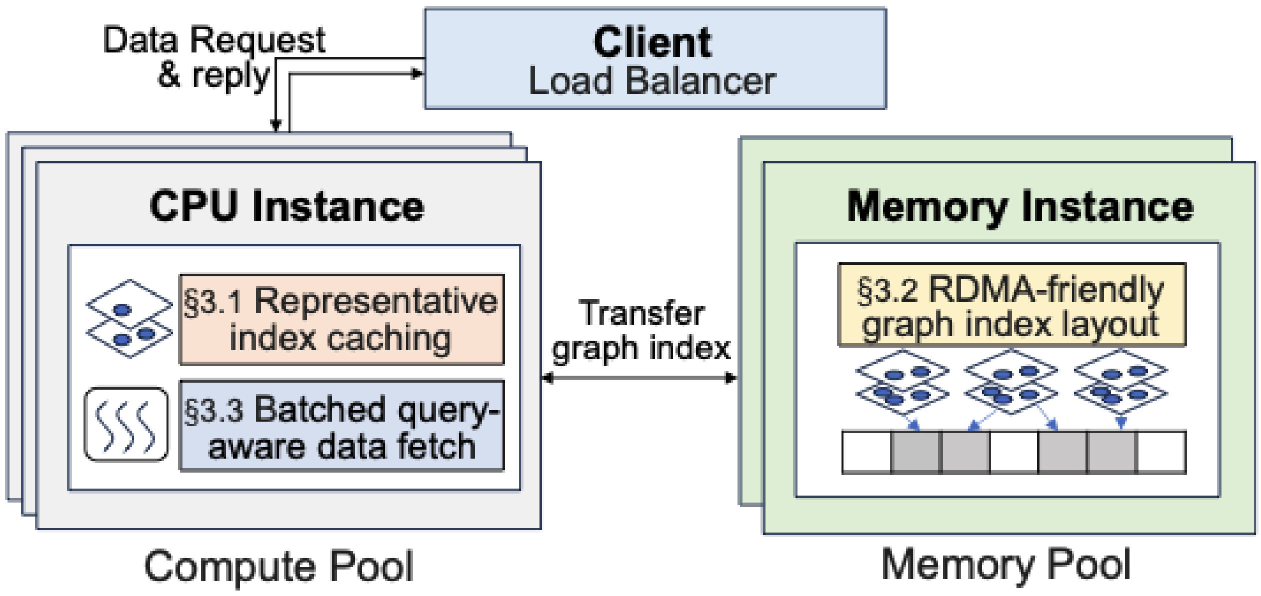


Overall design

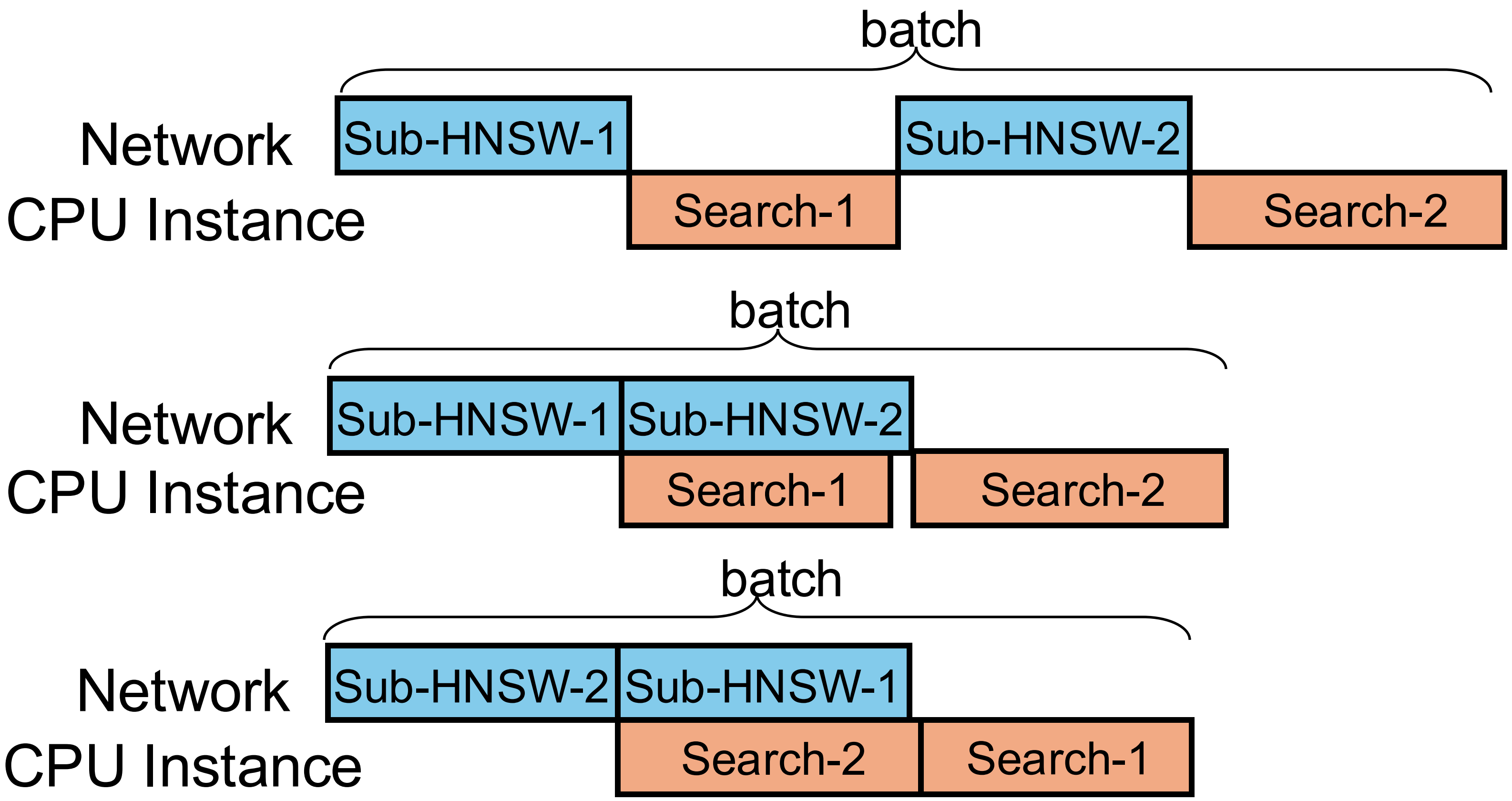
Representative index caching

RDMA-friendly graph index storage layout in remote memory.

Query-aware batched data loading



Pipeline Parallelism

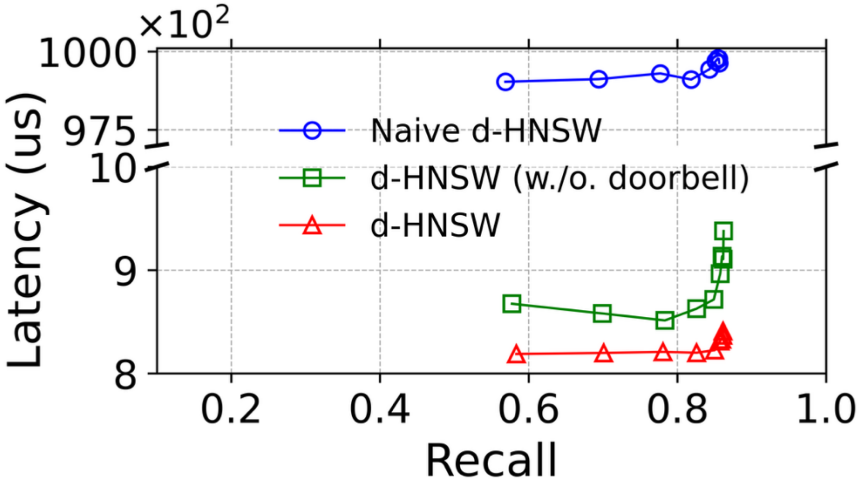


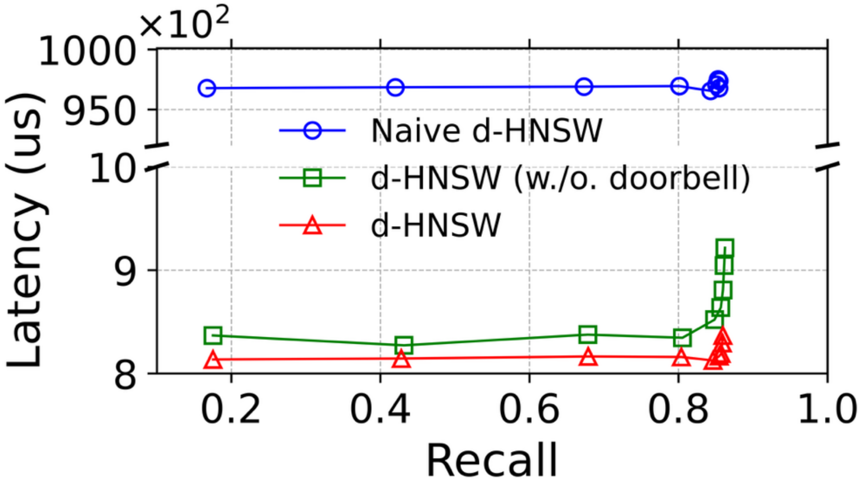
Implementation and Evaluation Step

- Implementation
  - ➢~12K LoC C++
- Testbed
  - ➢ Dell PowerEdge R650: 2×36-core Intel Xeon Platinum CPUs, 250GB RAM, 1.6TB NVMe SSD, **Mellanox ConnectX-6 100Gb NIC**
  - ➢3 as computing
  - ➢1 as memory node
- Datasets
  - ➢SIFT1M
  - ➢GIST1M

Evaluation

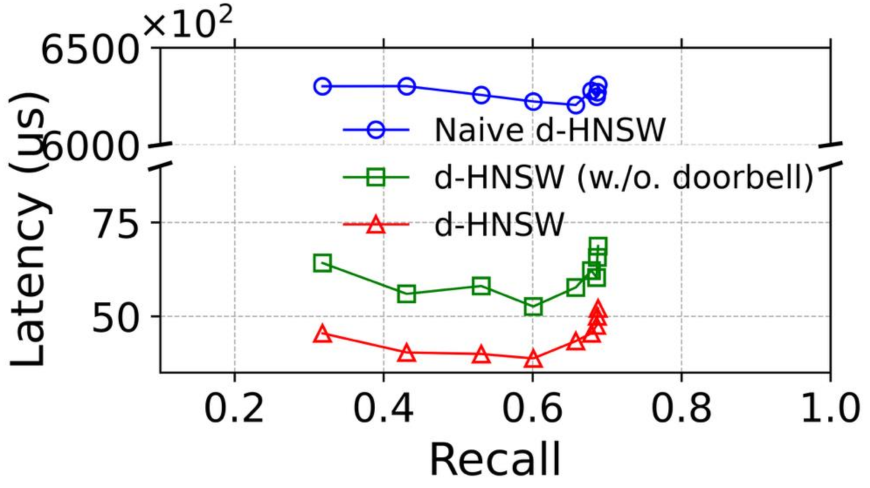
- d-HNSW reduces latency by up to 117× and 1.12× compared to naive d-HNSW and d-HNSW without doorbell

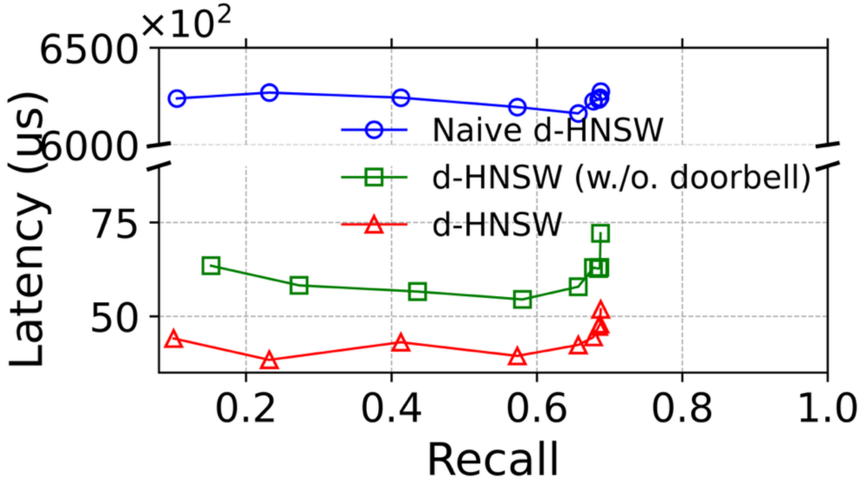




Evaluation

- d-HNSW achieves up to 121× and 1.30× lower latency compared to naive d-HNSW and d-HNSW without doorbell.





Evaluation

- d-HNSW outperforms Naïve d-HNSW and d-HNSW without doorbell in both network and sub-HNSW search latency.

Scheme	Network	Sub-HNSW	Meta-HNSW
Naive d-HNSW	90271.2μs	6564.5μs	13.52μs
d-HNSW (w./o. doorbell)	607.5μs	287.0μs	9.97μs
d-HNSW	**527.6μs**	269.2μs	9.75μs

Table 1: Latency breakdown for SIFT1M@1 with ef-Search as 48.

Scheme	Network	Sub-HNSW	Meta-HNSW
Naive d-HNSW	422.9ms	35.3ms	61.1μs
d-HNSW (w./o. doorbell)	2.9ms	1.27ms	52.6μs
d-HNSW	**1.3ms**	1.48ms	46.9μs

Table 2: Latency breakdown for GIST1M@1 with $efSearch$ **as 48.**

Conclusion

- We present d-HNSW: the first RDMA-based vector similarity search engine for disaggregated memory system.
- d-HNSW enhances vector request throughput and minimizes data transfer overhead by implementing an RDMA-friendly data layout for memory nodes.
- d-HNSW optimizes batched vector queries by eliminating redundant vector transfers for batched vector queries.

Thank you!