VecFlow-Chamfer: A GPU-based Data Management System for High-Performance Multi-Vector Search on Superchips

Chenghao Mo[1], Ben Karsin[2], Philip Adams[3], Minjia Zhang[1]

[1]: SSAIL Lab, UIUC [2]: Nvidia [3]: Microsoft

What is Multi-Vector Search?

Doc 1:	The	**cat**	sits	on	the	mat
	d0	**d1**	d2	d3	d4	d5
Query:	Where	does	the	**cat**	sit	
	q0	q1	q2	**q3**	q4	

Chamfer Score(Query, Doc *i*) = Σ_i $\max_j \langle q_i, d_j \rangle$

Query token	Best match	Sim score
Where	mat	0.61
does	The	0.52
the	the	0.95
cat	cat	**0.98**
sit	sits	0.89

Motivation

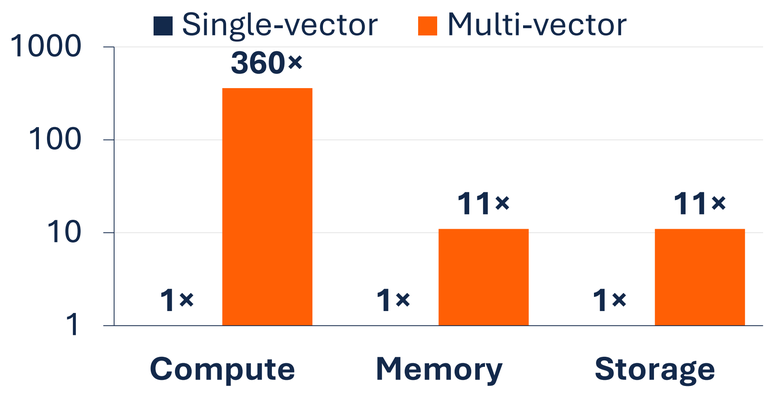
Multi-vector gives fine-grained representation

Heavy Compute and memory loads suit GPUs

Storage needs a GPU + CPU hierarchy

Challenges

Multi-vector search costs far more



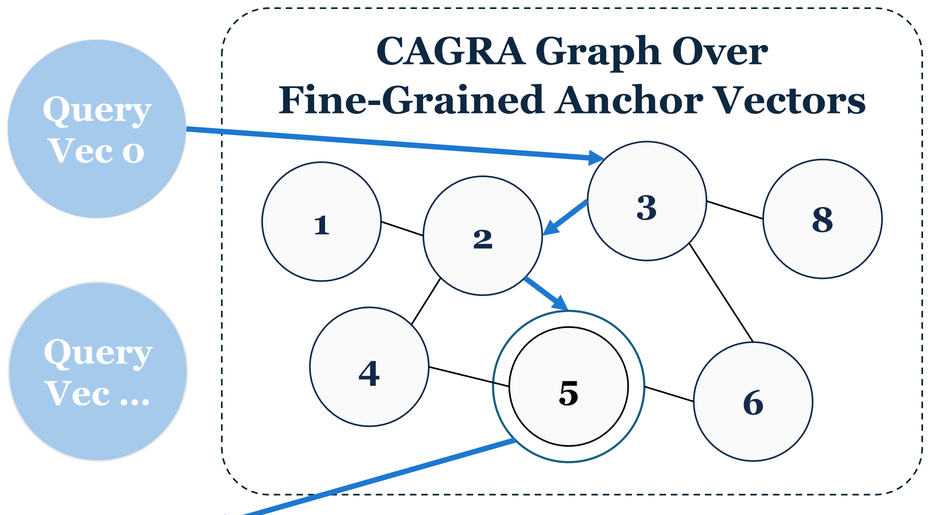
Single-vector dim 768, multi-vector dim 128, ~68 token/doc (MS MARCO).

VecFlow-Chamfer Architecture Overview

1 Retrieval Pipeline (Superchips)

Query tokens → **Candidate Generation** MaxIVF-CAGRA routing → **Proxy-based Filtering** **set-to-set** (anchors) → **Final Reranking** **set-to-set** (full-precision via C2C) → **Top-K results**

2 MaxIVF-CAGRA Index

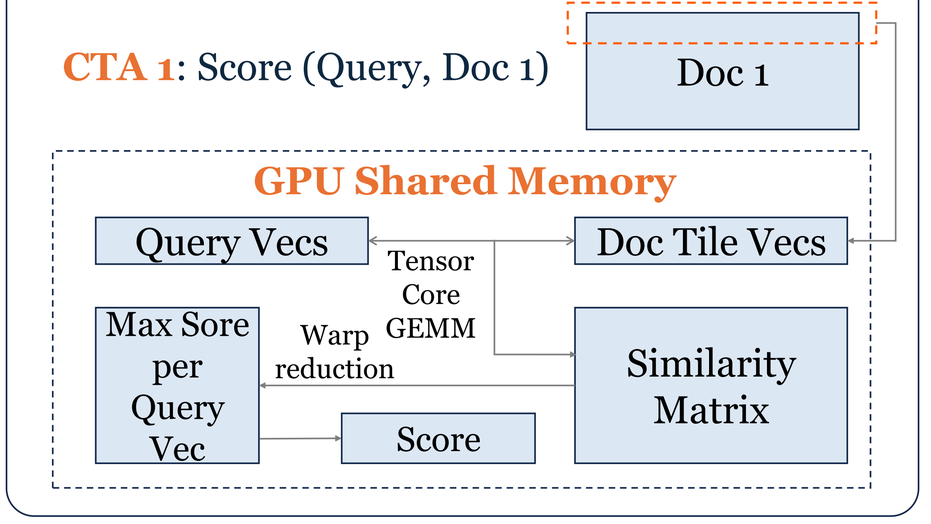


Doc 0
Vec 0 → A5
Vec 1 → A2
...
Doc 1
Vec 0 → A3
...

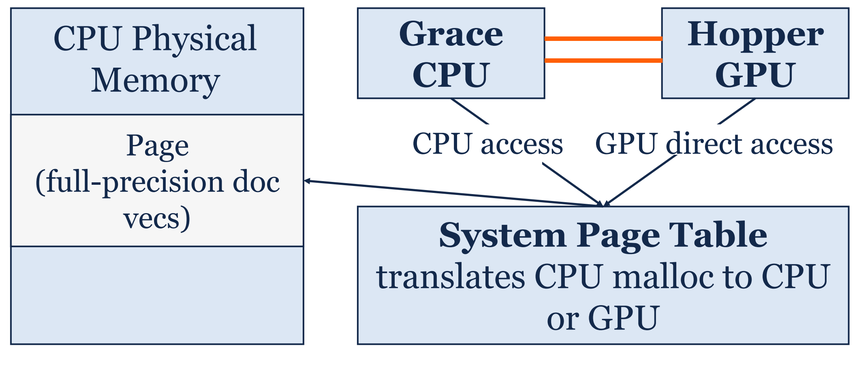
Each doc token → one anchor (CAGRA)

Doc IDs for this anchor each anchor → list of doc IDs → **Aggregate doc IDs** from all query vecs → **Hash-table dedup** remove duplicates

3 ChamferCore Kernel (set-to-set)



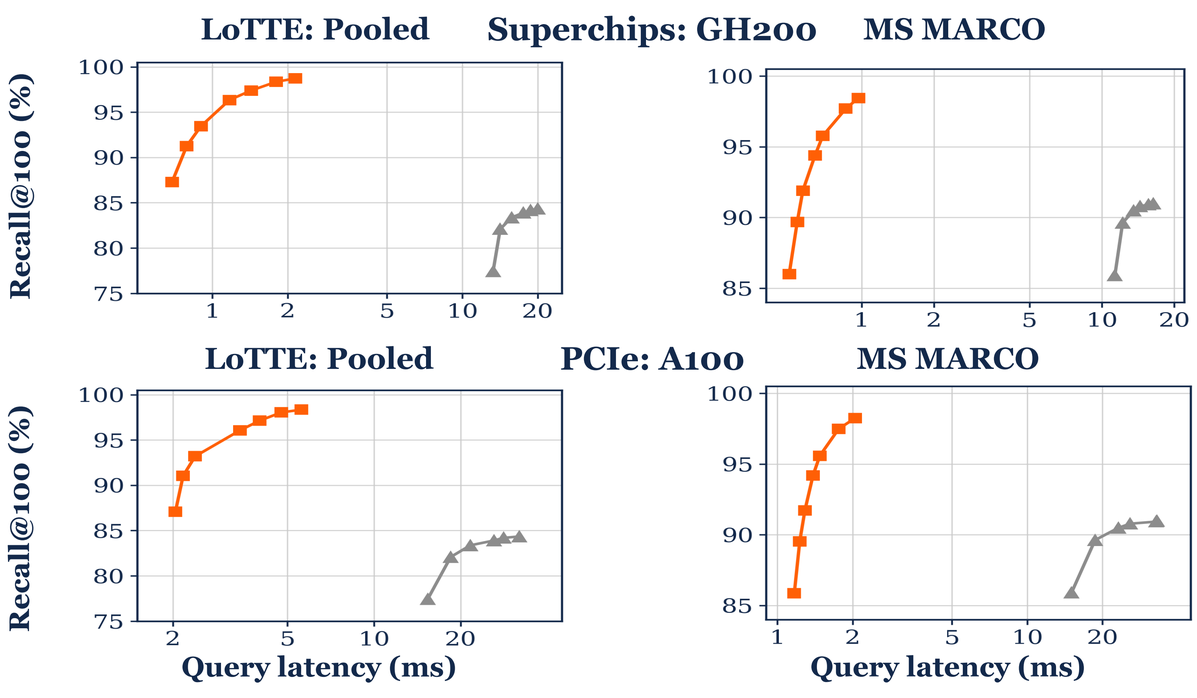
4 GraceStore



5 Retrieval Pipeline (PCIe)

Proxy-based Filtering (anchors) → **Second Filtering** (PQ Vectors) → **Final Reranking** (full-precision via PCIe)

Evaluation Highlights



- Up to **16.9×** faster, and **+7.5 pts** recall with 98.45% R@100 in 0.97 ms vs PLAID's 91% in 16.35 ms
- MaxIVF-CAGRA: **~25%** fewer candidates, higher coverage
- Sub-ms Chamfer scoring **31K docs** in **0.12 ms** ((MS MARCO)

Scan to Share

Code: github.com/Supercomputing-System-AI-Lab/VecFlow

Paper: dl.acm.org/doi/abs/10.1145/3786706