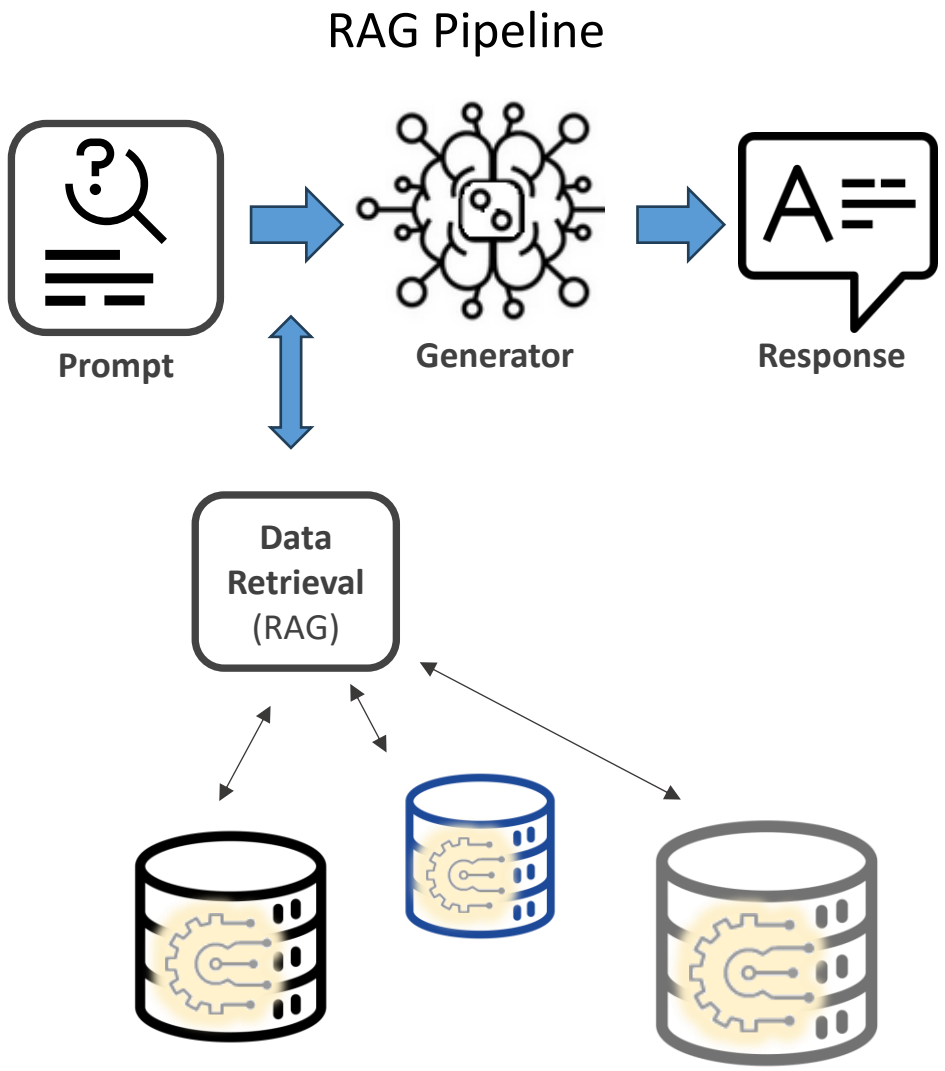
SSD controller architecture for similarity search in Vector DBs

Roman Pletka – IBM Research, Zurich

Motivation

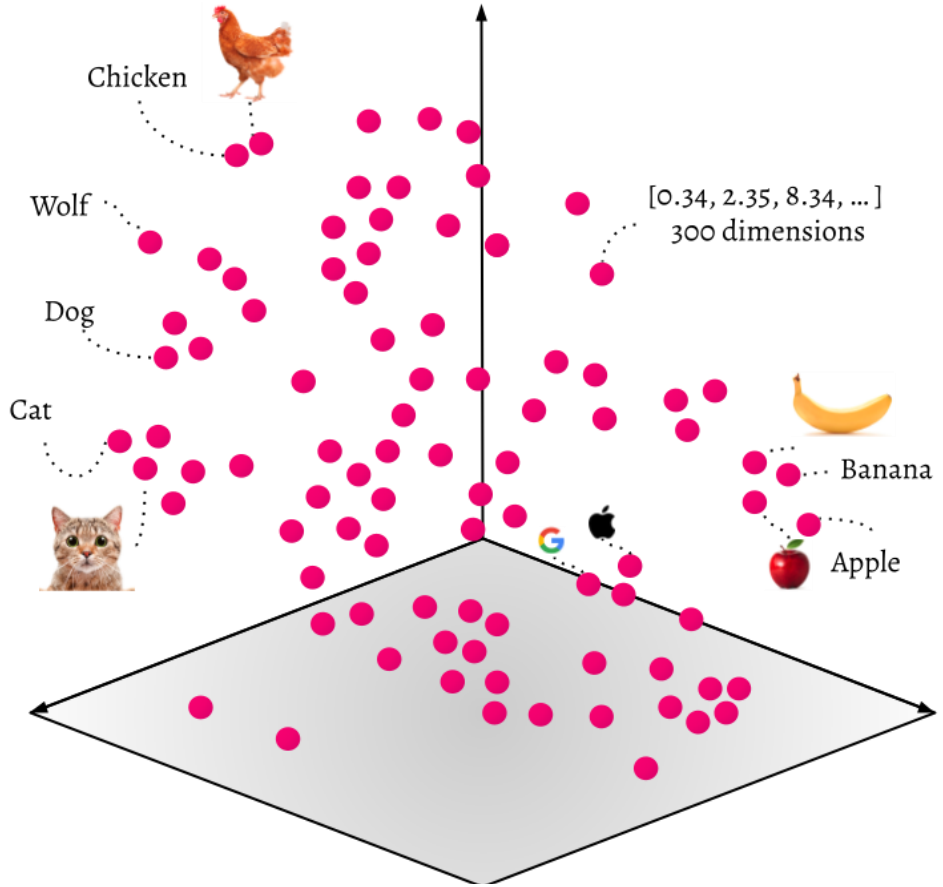
Unstructured data growth drives new possibilities for business value; **But are businesses able to interpret and extract content from this data?**

- Semantic search technologies that provide contextually relevant input tokens to a model can significantly improve the quality of search results.
- Contextually relevant data can be extracted using a process called Retrieval Augmented Generation (RAG) that retrieves semantically similar related facts from a large data store (e.g., vector database).
- However, the number of vectors to be searched for similarity is growing towards several billions and can no longer be kept in DRAM.
- Performing searches in storage directly can be highly parallelized to achieve high scalability and fast retrieval.



Vector similarity search background

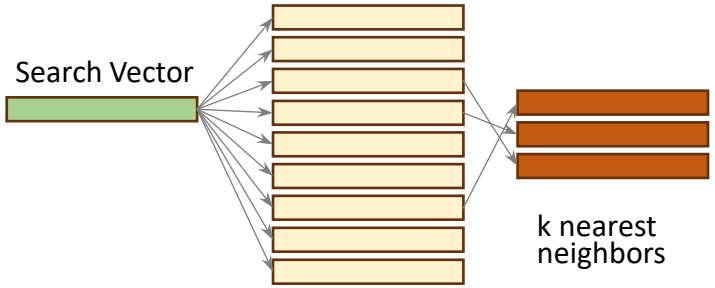
- Goal: Find similar data/documents based on their vector embedding representations.
- Enables efficient exploration, analysis, and information retrieval on very large data sets.
- Fundamental building block in a wide range of data-driven applications, Wide range of applications:
  - Image, video, audio similarity search
  - AI drug discovery
  - Semantic search engine
  - DNA sequence classification
  - Question answering system
  - Recommender system
  - Anomaly detection
  - Retrieval Augmented Generation (RAG)



Source: https://opendatascience.com/wp-content/uploads/2022/03/vectors-3d-multi.png

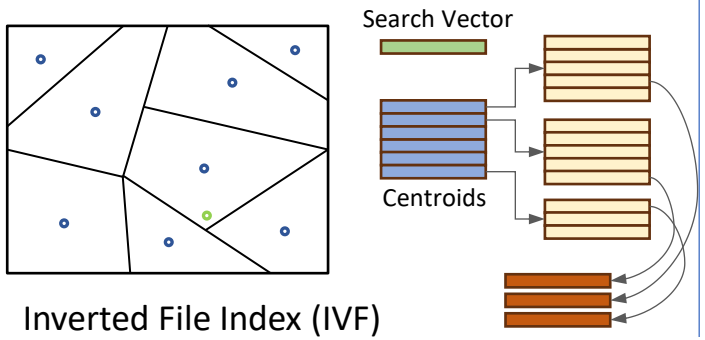
Vector search overview

Nearest Neighbor Search



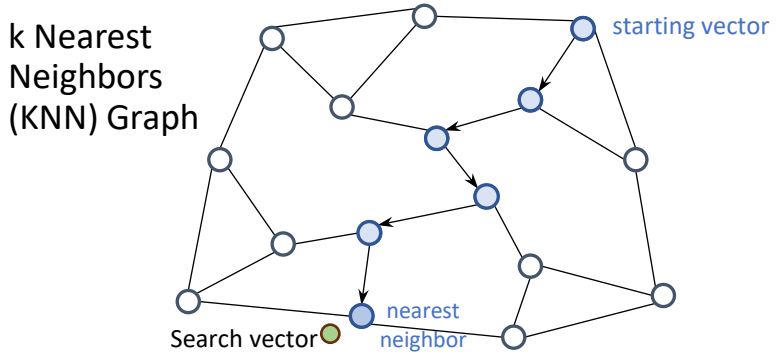
Exhaustive Search

Partitioning: Lists

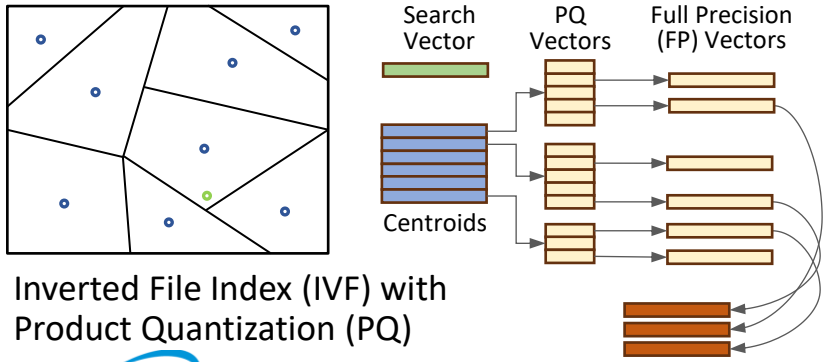


Inverted File Index (IVF)

Partitioning: Graphs

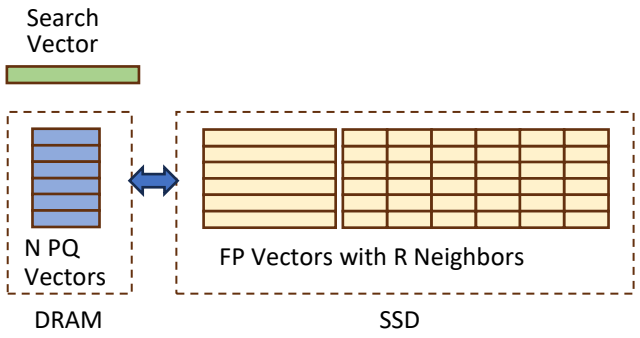


Quantization-based: IVF-PQ

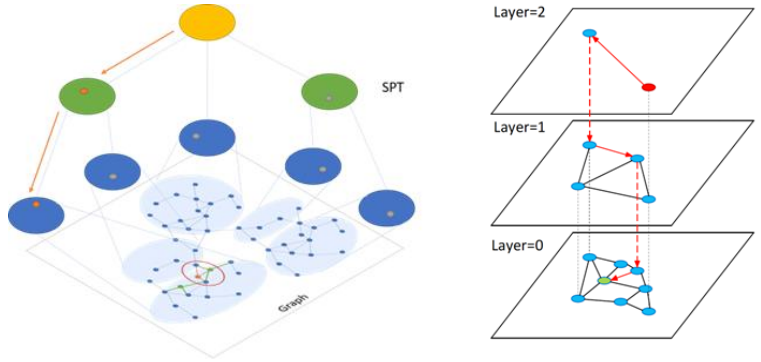


Inverted File Index (IVF) with Product Quantization (PQ)

Quantization-based: DiskANN



Hierarchical Searches



SPTAG, SPANN, HNSW, …

Vector Search Module (VSM)

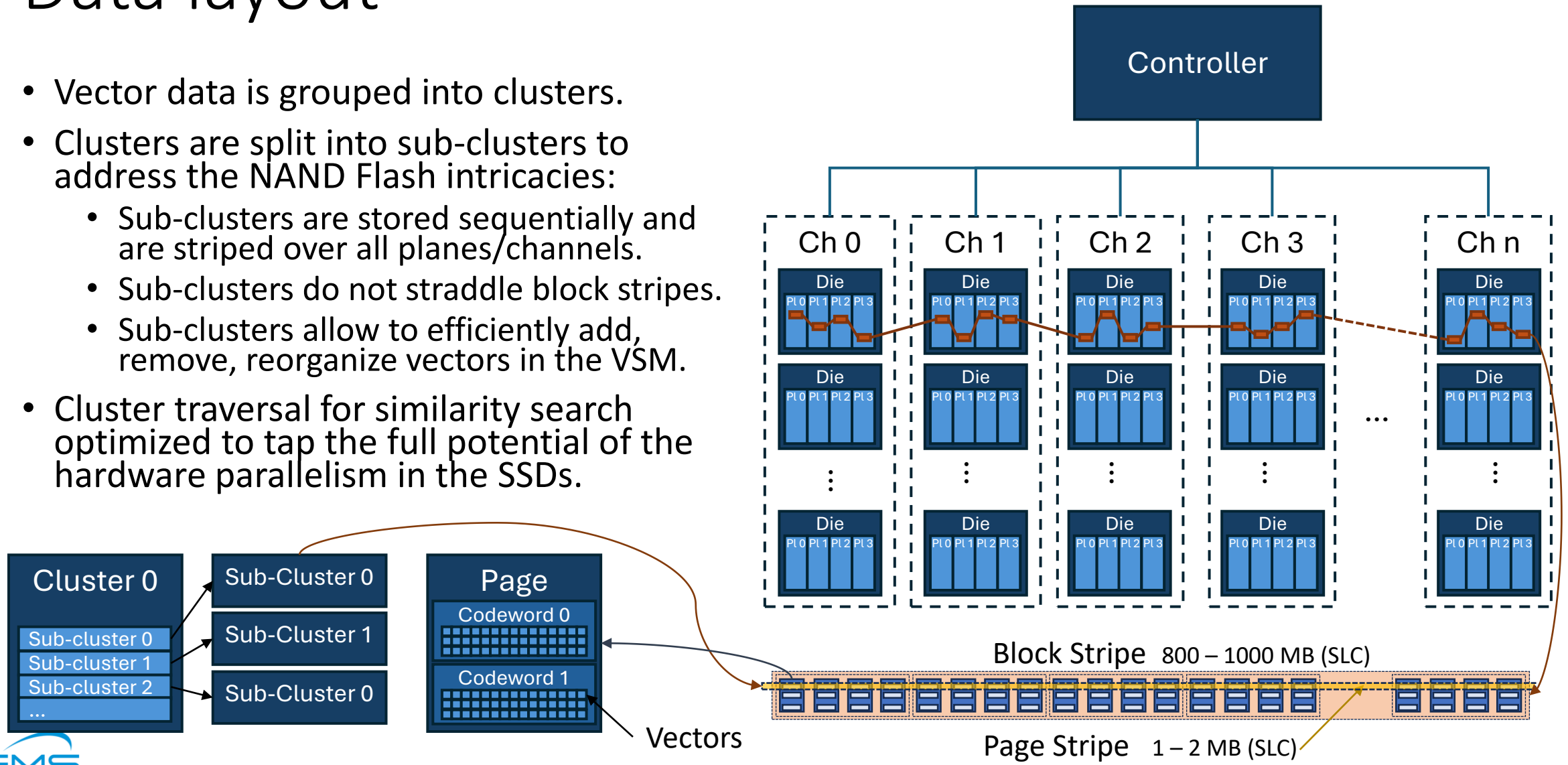
An FPGA-based hardware architecture for similarity search

	Flash Core Module FCM	Vector Search Module VSM
Block mode	QLC (dynamic SLC/QLC)	SLC only
Capacity	- 14.4 - 115.2 TB effective - 4.8 - 38.4 TB physical	1.2 – 9.6 TB; up to: - 37B vectors (d=128, FP16) - 2.3B vectors (d=1024, FP32)
LPT	- 4kB granularity - Pageable LPT	- Variable sized data chunks > 3500 x smaller LPT - LPT fully in DRAM
Data format	- Data straddles across codewords (compression) - 3:1 hardware-based compression	- Clusters and sub-clusters, - Vector information aligned to codewords - No compression

Built on same hardware !

Data layout

- Vector data is grouped into clusters.
- Clusters are split into sub-clusters to address the NAND Flash intricacies:
  - Sub-clusters are stored sequentially and are striped over all planes/channels.
  - Sub-clusters do not straddle block stripes.
  - Sub-clusters allow to efficiently add, remove, reorganize vectors in the VSM.
- Cluster traversal for similarity search optimized to tap the full potential of the hardware parallelism in the SSDs.



LPT and GC in a VSM

The traditional LPT is replaced by:

- Cluster-to-subcluster mapping table.
- Sub-cluster to physical NAND location table.

Challenges and benefits:

- Granularity is no longer fixed-size.
- VSM LPT is 3500x smaller and fits entirely into DRAM => No LPT paging required.

What is kept:

- Stripe-to-blocks allocation table.

Garbage collection:

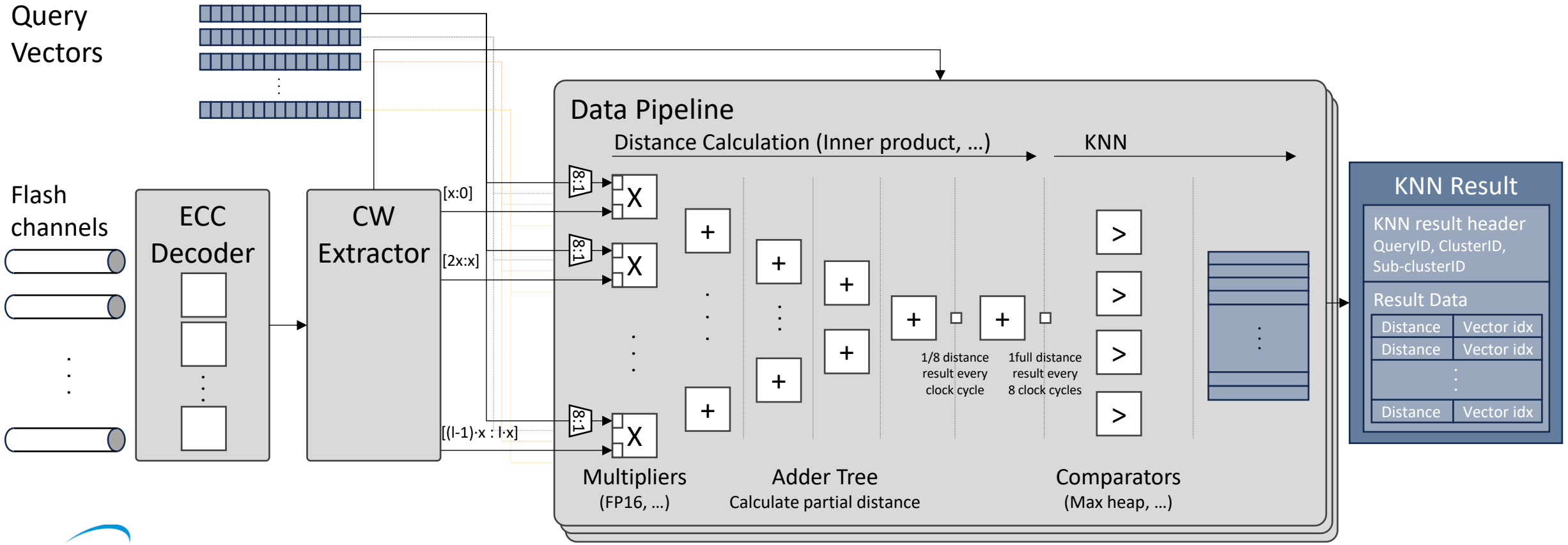
- Entire sub-clusters are invalidated, and still valid sub-clusters must be relocated.
- Track validity of sub-clusters on the block stripe level.
- Same GC algorithm can be used as in FCM.

Further improvements

- A limited number of individual vectors in a sub-cluster can be invalidated using skip lists.
- Any new vector added to a cluster can be appended to the last or added into a new sub-cluster.
- GC cleanup of sub-cluster with large skip list.

VSM Hardware Architecture

- Sub-cluster data is read in parallel from Flash. After ECC decoding, vector data is extracted from codewords.
- The VSM currently supports 32 parallel data pipelines for processing query vectors.



Data set and evaluation environments

Data sets:

- ArXiv vector data set:
  - 3.5 millions PDF papers collected from arxiv.org (2007 to 2024) by a research team internally.
  - Embedding:
    - Using ColPali vision LLM: PDF pages divided into patches; each patch is embedded into a vector with 128 dimensions using float16.
  - Data set contains up to 6 billion vectors.
  - Data set includes ~10k query vectors and ground truth for top-100 for each query vector using inner product.
- Common Crawl data set
  - Documents from commoncrawl.org
  - Embedding: 384 dim using float16
  - Total up to 1.2 billion vectors

Evaluation environments:

- Exhaustive search:
  Search all vectors stored in VSM and return top-k vectors. The search uses a single cluster.
  - Data sets: 1M vectors from ArXiv data set, 1.2B from Common Crawl data set
  - Result: up to top-100 vectors for each query vector
- ANN search using IVF:
  - Using 100M (Common Crawl) and 1B vectors (ArXiv)
  - Used balanced clustering to create 10k clusters
  - Number of clusters queried = 100
  - Centroids search performed outside VSM
  - All queried clusters are search in VSM

Exhaustive search performance

Performance comparison of various configurations with a single VSM using the ArXiv data set:

Vectors	NAND Channels	ECC Decoders	Queries	Top-k	Time (s)	QPS	QPS/W
1 million	8	1	32	1	0.071	3601.50	144.06
	8	1	32	10	0.071	3601.50	144.06
	8	1	32	100	0.071	3601.50	144.06
	8	max	32	10	0.015	17031.10	681.24
	8	max	32	100	0.015	17031.10	681.24
	16	max	8	10	0.008	8515.55	340.62
	16	max	8	100	0.008	8515.55	340.62

- A single VSM with 16 channels searches 1 million vectors in less than a hundredth of a second.
- The top-k nearest neighbor search is not in the critical path (k<100). No measurable impact on search time.
- Number of concurrent ECC decoders limits processing throughput requiring more decoders. Alternatively, simpler decoders can be used as NAND Flash is used in SLC mode to reach the maximum Flash channel bandwidth.
- Currently investigating how to further increase number of parallel query vectors searched while using all NAND Flash channels.

Exhaustive search – system-level performance

- Estimated system-level efficiency gains from using VSMs compared to CPU-based, GPU and VSM approaches performing exhaustive search with 32 query vectors on the Common Crawl 1.2B data set:

System Configurations with VSMs	Efficiency Gains w.r.t : CPU only: Intel 8568Y+, 96c, 2TB DRAM DDR5 @5600MT/s		GPU-based: 1x NVIDIA L4 GPU, 2-socket server (Intel 5520+)		8x NVIDIA L4 GPUs, 2-socket server (Intel 5520+)		1x NVIDIA H100 GPUs, 2-socket server (Intel 8568Y+)		4x NVIDIA H100 GPUs, 2-socket server (Intel 8568Y+)	
	QPS gain	QPS/W gain	QPS gain	QPS/W gain	QPS gain	QPS/W gain	QPS gain	QPS/W gain	QPS gain	QPS/W gain
Server (Intel 8462Y+) - 1x VSM, single ECC dec	2.295	2.688	0.097	0.089	0.084	0.144	0.040	0.090	0.036	0.203
Server (Intel 8462Y+) - 1x VSM, multiple ECC dec	20.657	24.193	0.877	0.802	0.755	1.300	0.361	0.809	0.327	1.831
1U enclosure with processor - 12x VSMs	247.883	201.612	10.519	6.686	9.060	10.831	4.336	6.745	3.923	15.257
4U enclosure with processor - 48x VSMs	991.533	403.224	42.077	13.371	36.239	21.663	17.344	13.490	15.693	30.515

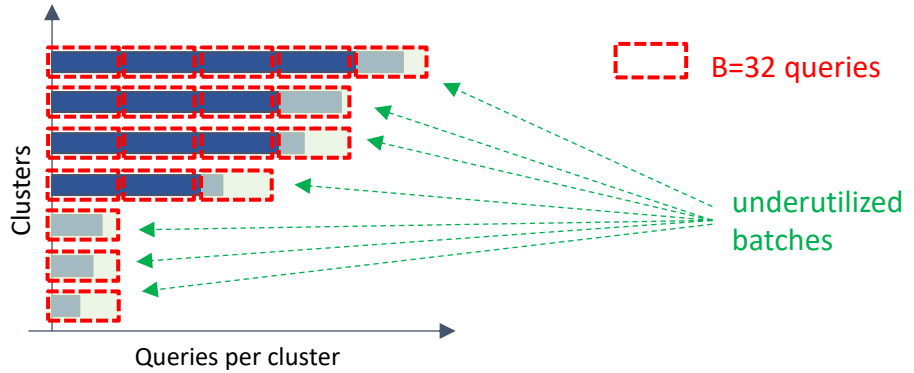
Data storage and processing using IVF

IVF search approach with VSMs:

- Keep cluster centroids in DRAM.
- Store entire clustered vector data set in VSMs.
- Send query vectors of selected clusters to VSMs.
  - Maintain several replicas of a cluster across VSMs.
- Each VSM performs top-k on all clusters queried.
- Perform final top-k on all results returned from VSMs.

Challenges in batch processing:

- For a set of *b* query vectors, determine which clusters must be queried for each query based on centroids, then schedule the computation in batches on the VSMs using this mapping:
  - Best case: All queries target the same clusters. => the most acceleration across all options.
  - Worst case: All queries target different clusters. => sequential processing of multiple clusters.



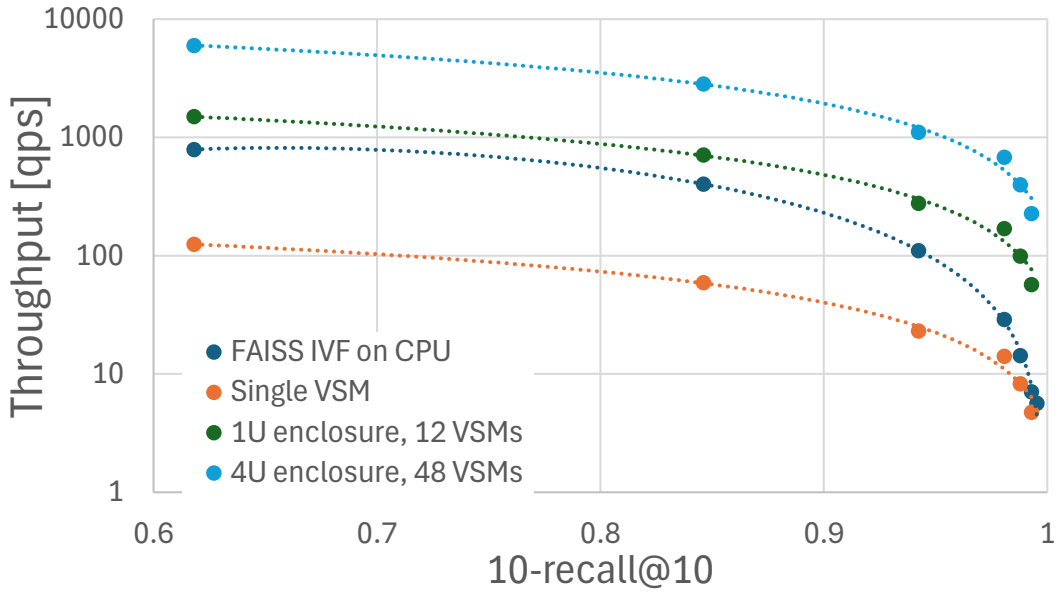
Performance results for ANN with IVF

- Dataset: Common Crawl 100M, 10k clusters.
- Workload: up to 10k queries, batch size = 32 queries.
- Vary the query set size (nqueries) and the number of clusters queried (nprobes).

Percentage of fully-utilized query batches

nqueries	nprobe						
	4	8	32	64	128	256	512
1	0%	0%	0%	0%	0%	0%	0%
32	0%	0%	0%	0%	0%	0%	0%
64	0%	0%	0%	0%	0%	48.0%	50.0%
128	0%	0%	0%	0%	48.7%	66.5%	73.3%
512	0%	0%	25.5%	61.3%	77.7%	86.3%	90.4%
1024	0%	0%	53.3%	72.6%	85.4%	91.5%	94.2%
10000	32.0%	57.6%	88.6%	94.1%	96.6%	97.9%	98.6%
1-recall@10	0.734	0.827	0.947	0.972	0.978	0.994	1
10-recall@10	0.611	0.716	0.861	0.912	0.944	0.968	0.997

- Dataset: ArXiv 1B, 10k clusters, 1000 queries.
- Workload: 1000 queries, batch size = 32 query vectors.
- Vary the number of clusters queried (nprobes: 1, …, 256).
- VSM-based architectures show extremely good performance when number of clusters queried is high: On par with CPU-only for single VSM; > 10x for multiple VSMs.



Existing interfaces

NVMe Key Value Command Set Specification 1.2

https://nvmexpress.org/specification/key-value-command-set-specification/

- Provides interface to store unstructured data by a key.
- There is no meta-data associated with values.
- Key-value operations:
  - Store, retrieve, delete, list
  - Values are not updatable in place
- The interface could be used for vector search where individual vectors or clusters are stored ad key-value pairs.

Amazon S3 Vectors

https://nvmexpress.org/specification/key-value-command-set-specification/

- Cloud object storage interfaces for storing and querying vectors using a REST API.
- Vectors are stored in buckets each having up to 50M vectors. Indexes are then created on buckets => Clusters can be organized into buckets.
- Operations:
  - PutVectors, QueryVectors, DeleteVectors, ListVectors
  - CreateVectorBucket, CreateIndex, DeleteIndex, …

Existing interfaces lack Flash-friendly capabilities to organize clusters into sub-clusters for vector search

Conclusion and outlook

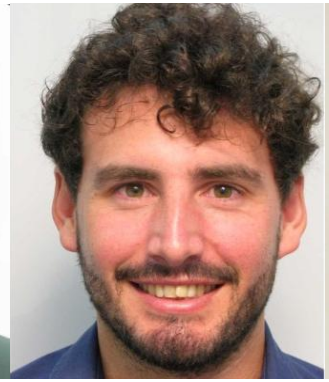
- VSMs are a cost and power-efficient solution to build highly scalable vector similarity search engines.
- CSD SSDs can be either used as an FCM storage device or VSM in a storage array. The storage array can dynamically convert devices to be used as FCM or VSM.
- Next steps:
  - Full integration of VSMs into an all-flash array.
  - Optimize VSM hardware to support multiple vector formats.
  - Reliability: Array-level RAID5/6 replaced by storing cluster replicas across VSMs. Determine optimal placement strategies of cluster replicas.

Team

Roman Pletka
Senior Research Scientist
Master Inventor

Haris Pozidis
Principal Research Scientist
Master Inventor

Jovan Blanuša
Research Scientist



Dionysios Diamantopoulos
Staff Research Scientist

Dan Lazar
Senior Hardware Developer

Justin Haggard
Senior Firmware Developer

Tim Fisher
FlashCore Module Chief Architect
STSM
Master Inventor

IBM Research Europe – Zurich

the Future of Memory and Storage

Thank you!

Roman Pletka
Senior Research Scientist
Master Inventor
rap@zurich.ibm.com

Contributors:
- Haris Pozidis
- Jovan Blanuša
- Dionysios Diamantopoulos
- Dan Lazar
- Justin Haggard
- Tim Fisher