

# COSMOS: A CXL-Based Full In-Memory System for Approximate Nearest Neighbor Search

Seoyoung Ko, Hyunjeong Shim, Wanju Doh, Sungmin Yun, Jinin So, Yongsuk Kwon, Sang-Soo Park, Si-Dong Roh, Minyong Yoon, Taeksang Song, and Jung Ho Ahn, *Senior Member, IEEE*

**Abstract**—Retrieval-Augmented Generation (RAG) is crucial for improving the quality of large language models by injecting proper contexts extracted from external sources. RAG requires high-throughput, low-latency Approximate Nearest Neighbor Search (ANNS) over billion-scale vector databases. Conventional DRAM/SSD solutions face capacity/latency limits, whereas specialized hardware or RDMA clusters lack flexibility or incur network overhead. We present COSMOS, integrating general-purpose cores within CXL memory devices for full ANNS offload and introducing rank-level parallel distance computation to maximize memory bandwidth. We also propose an adjacency-aware data placement that balances search loads across CXL devices based on inter-cluster proximity. Evaluations on SIFT1B and DEEP1B traces show that COSMOS achieves up to  $6.72\times$  higher throughput than the baseline CXL system and  $2.35\times$  over a state-of-the-art CXL-based solution, demonstrating scalability for RAG pipelines.

**Index Terms**—CXL, Approximate Nearest Neighbor Search, Processing Near Memory, Retrieval-Augmented Generation

## I. INTRODUCTION

**R**ETRIEVAL-AUGMENTED Generation (RAG) enhances Large Language Models (LLMs) by dynamically retrieving relevant information from external databases, enabling more accurate and contextually appropriate responses [1]. Techniques such as Agentic RAG [2] further improve quality through iterative retrieval. Central to RAG is Approximate Nearest Neighbor Search (ANNS), which enables fast retrieval in high-dimensional vector spaces by efficiently identifying the top- $k$  most relevant vectors—those closest to a given query—based on a similarity metric. As datasets grow, efficient and scalable ANNS is critical for real-time inference.

Handling billion-scale ANNS workloads challenges traditional systems. DRAM lacks capacity, whereas SSDs suffer from high latency incompatible with fine-grained ANNS access patterns [3], [4]. Alternative solutions such as conventional Processing Near Memory (PNM) [3], [5] lack flexibility, and RDMA clusters [4] incur network latency and complexity.

Compute Express Link (CXL) offers a promising solution with high-bandwidth, low-latency memory expansion [6], beneficial for latency-sensitive RAG [1]. Still, ANNS remains

This work was partly supported by Samsung Electronics Co., Ltd (IO250301-12185-01) and IITP (RS-2021-II211343 and RS-2023-00256081).

Seoyoung Ko, Hyunjeong Shim, Wanju Doh, Sungmin Yun, and Jung Ho Ahn are with Seoul National University, Seoul 08826, South Korea. E-mail: {seoyoungko, simhj1212, wj.doh, sungmin.yun, gajh}@snu.ac.kr.

Jinin So, Yongsuk Kwon, Sang-Soo Park, Si-Dong Roh, Minyong Yoon, and Taeksang Song are with Samsung Electronics Corporation, Hwaseong-si, Gyeonggi-do 18448, South Korea. E-mail: {jinin.so, yssh.kwon, ss23.park, sidong.roh, casper.yoon, taeksang.song}@samsung.com.

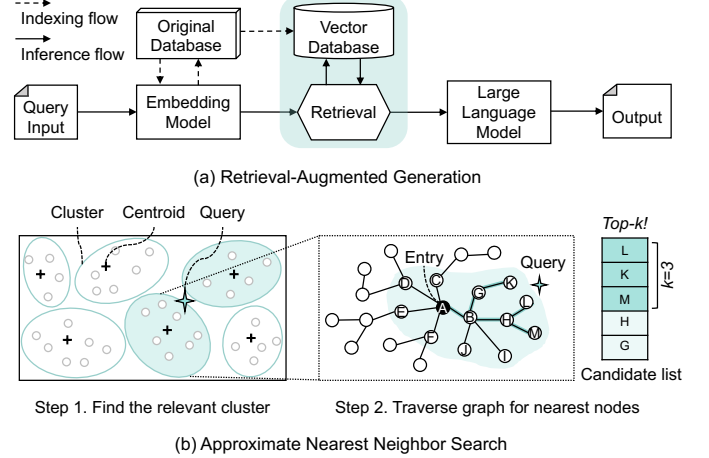


Fig. 1. Overview of retrieval-augmented generation (RAG) and approximate nearest neighbor search (ANNS).

memory-bandwidth bound, dominated by distance calculations [7]. Distributing indices across multiple CXL devices requires intelligent data placement to avoid load imbalance.

In this paper, we propose COSMOS, a full in-memory ANNS system using compute-capable CXL devices. COSMOS makes the following three key contributions:

- **Full ANNS Offload via CXL GPCs:** COSMOS integrates programmable general-purpose cores (GPCs) in CXL memory controllers for local ANNS execution, eliminating host intervention and PCIe traffic during search.
- **Rank-level Parallel Distance Computation:** COSMOS exploits DRAM rank-level parallelism for concurrent distance computation, reducing data movement and maximizing memory bandwidth.
- **Adjacency-aware Data Placement:** A lightweight algorithm uses cluster metadata to distribute vectors across CXL devices, balancing load and enabling parallelism without runtime profiling.

## II. BACKGROUND

RAG enhances LLMs by retrieving relevant information from external vector databases during inference (Fig. 1(a)). This typically involves indexing documents into a vector database and, during inference, embedding user queries to retrieve similar documents. Recently, Agentic RAG [2] improves quality via iterative search strategies. The core retrieval mechanism relies on finding vectors (documents) most similar to the query vector.

Exact Nearest Neighbor Search (ENNS) guarantees the highest accuracy; however, its linear scaling with dataset

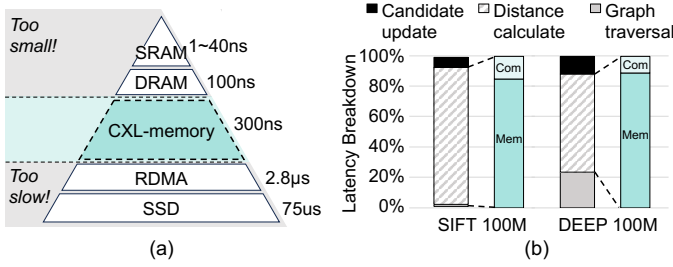


Fig. 2. (a) Memory latency hierarchy highlighting the potential of CXL-attached memory as a new tier between DRAM and RDMA/SSD in terms of latency and capacity. (b) Latency breakdown of graph-based ANN search on large-scale datasets (SIFT and DEEP with 100M vectors).

size in computational cost makes it impractical for billion-scale data [3]. By contrast, ANNS offers a trade-off between accuracy and efficiency, enabling real-time search.

ANNS methods include graph-based and cluster-based approaches [4], [8]. The former links similar vectors and searches greedily along edges, but can suffer from irregular memory access. The latter partitions data, finds the closest cluster(s) to the query, and searches within them, which offers better memory efficiency but suffers from potential read amplification. Hybrid approaches combine these, restricting graph traversal to relevant clusters, improving efficiency for large datasets while maintaining quality in billion-scale datasets. Fig. 1(b) shows an exemplar hybrid-based ANNS process when  $k$  is 3.

### III. MOTIVATION

#### A. Scalability Challenges in Billion-Scale ANNS

Billion-scale ANNS demands vast memory (terabytes) [9], [3], exceeding single-node DRAM limits. Using SSDs introduces high latency (tens of microseconds), and their coarse-grained access (kilobyte-scale pages) is unsuitable for fine-grained ANNS, potentially dominating search time [3], [4].

SSD-based PNM accelerators reduce data movement [3], [5], but they lack flexibility against evolving algorithms or parameters. RDMA-based multi-node clusters offer lower latency than SSDs but still suffer network overhead (few to several microseconds [4], Fig. 2(a)) and add complexity.

Compute Express Link (CXL), a PCIe-based interconnect standard, has emerged as a promising alternative. CXL provides direct load/store access to expanded memory with near-native-DRAM latency (few hundred nanoseconds, Fig. 2(a)) and high bandwidth, eliminating network overhead [6]. This low latency is critical for RAG, where retrieval time significantly impacts overall performance, especially with iterative techniques like Agentic RAG. Studies show that retrieval accounts for 36% of the time-to-first-token in a vanilla RAG and up to 97% in scenarios involving frequent re-retrieval [1].

#### B. Leveraging Compute-capable CXL Devices

Recent CXL memory devices optionally incorporate near-memory compute capabilities [10], [11]. Performing computation near memory reduces data transfer and alleviates bottlenecks for memory-intensive workloads like ANNS. We propose *offloading the entire ANNS pipeline to compute-capable*

TABLE I  
ANN DATASETS AND SEARCH PARAMETERS

Billion-scale ANN Datasets				
Data Type	SIFT	DEEP	Text2Image	MSSPACEV
Dimension	uint8 128	fp32 96	fp32 200	int8 100
Search Parameters				
max_degree	Maximum number of neighbors per node			
cand_list_len	Candidate list size			
num_clusters	Total number of clusters			
num_probes	Number of clusters searched per query			

*CXL devices* featuring programmable General-Purpose Cores (GPCs). GPCs offer flexibility over fixed accelerators, adapting to diverse datasets and parameters (see Table I for the BigANN benchmark [12]) without redesign, which mitigates over/under-provisioning issues on the fixed accelerators.

While offloading helps, ANNS remains bottlenecked by memory bandwidth, primarily due to distance calculations loading large vectors (Fig. 2(b)). Our architecture addresses this by integrating a GPC with a DRAM *rank-level processing unit (PU)*. This exploits DRAM's internal parallelism by partitioning vector dimensions across ranks, computing partial distances concurrently within each rank, which reduces data movement and improves bandwidth efficiency.

However, billion-scale datasets require distributing the index across *multiple* CXL devices. Naïve distribution can lead to load imbalance when co-accessed data resides on the same device. We address this with an *adjacency-aware cluster placement* algorithm that assigns clusters based on proximity, balancing load and enabling parallel search.

### IV. COSMOS: ARCHITECTURE AND MECHANISMS

#### A. System Architecture and Workflow

COSMOS utilizes a CXL-based architecture with a host CPU, CXL switch, and multiple CXL memory devices (Fig. 3(a)). Each CXL device consists of a CXL controller with a CXL-PNM module and DRAM devices supporting rank-level PUs. The host dispatches queries via the switch to relevant CXL devices. Each device performs local ANNS using its GPC (performing graph traversal and candidate list management) and returns local top- $k$  results. The host aggregates these for the global top- $k$ . Interface registers mapped in host memory facilitate host-PNM communication. Intermediate results generated during computation are stored in the temporary buffers, minimizing unnecessary memory access.

COSMOS exploits DRAM rank-level parallelism for performance. Data is column-wise partitioned across ranks, allowing independent processing within each rank's PU, alleviating channel contention. Rank-level PUs compute partial distances (e.g., for L2 distance and inner product) on 64B sub-vector segments in parallel (Fig. 3(c)). Unlike prior work (CXL-ANNS [9]), which offloaded only distance calculation to a domain-specific accelerator and required host-managed traversal, COSMOS fully offloads traversal to the CXL GPC and uses rank-level PUs. This significantly reduces PCIe traffic (only local top- $k$  results are transferred) and memory bandwidth bottlenecks, enabling scalable ANNS for billion-scale data.

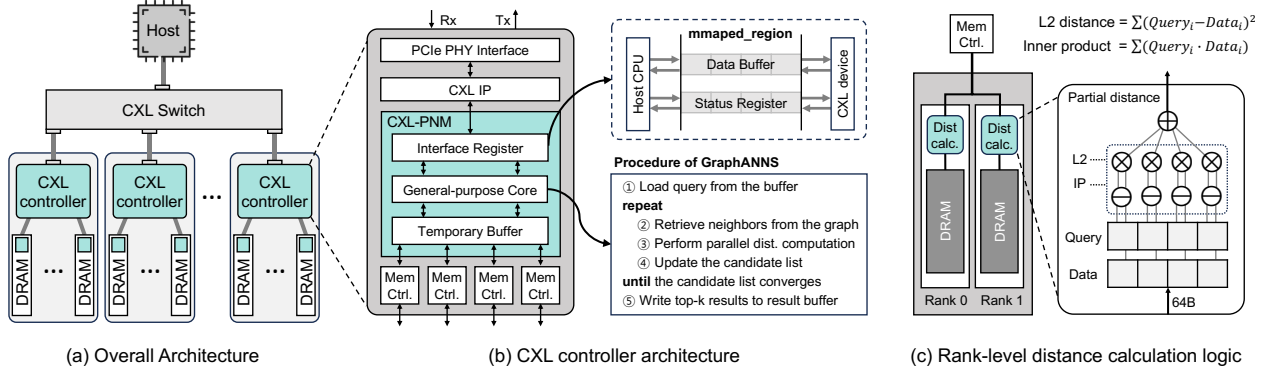


Fig. 3. (a) Overview of the system architecture. (b) CXL controller architecture featuring a general-purpose core for executing graph-based ANN search via a memory-mapped host interface. (c) Rank-level distance calculation logic that enables parallel L2 and inner product calculations across memory ranks.

#### Algorithm 1 Adjacency-aware Cluster Placement

**Input:** *cluster*: A cluster to be placed, including *.size* and a proximity-ordered list *.adj* of the nearby clusters.  
*devices*: A list of available CXL devices in the system.  
**Output:** *best\_d*: The best CXL device for placing the *cluster*.

```

1: best_d, max_cap, min_loss  $\leftarrow -1, 0, \infty$ 
2: for d in devices do
3:   if d.remain  $\geq$  cluster.size then
4:     loss, proximity  $\leftarrow 0$ , num_devices
5:     for adj in cluster.adj do
6:       if adj  $\in$  d.clusters then
7:         loss  $\leftarrow$  loss + proximity
8:       proximity  $\leftarrow$  proximity - 1
9:   if (best_d = -1) or (loss < min_loss) or
      (loss = min_loss and d.remain > max_cap) then
10:    best_d, min_loss, max_cap  $\leftarrow$  d, loss, d.remain
11: best_d.remain  $\leftarrow$  best_d.remain - cluster.size
12: return best_d

```

#### B. Memory Space Management

COSMOS integrates CXL Host-managed Device Memory (HDM) into the host physical address (HPA) space using static mapping, eliminating runtime translation overhead. Following [6], the kernel driver maps HDM regions into HPA during enumeration and informs the devices. User applications allocate HDM via a namespace interface and `mmap()` system call. A segment table ensures contiguous physical/virtual mappings.

Given ANNS's read-only nature after indexing, graphs and embedding data can be allocated using a static memory layout. This eliminates the need for dynamic virtual-to-physical address translation. During preprocessing, both the graph and embedding data are placed in HDM, and their metadata (e.g., base addresses and sizes) is registered with the controller. Address calculation becomes simple arithmetic:

$$\text{addr}_{\text{node}} = \text{addr}_{\text{graph\_base}} + (\text{node\_index} \times \text{node\_stride})$$

$$\text{addr}_{\text{vector}} = \text{addr}_{\text{embedding\_base}} + (\text{vector\_index} \times \text{vector\_stride})$$

To ensure mapping validity within the CXL device, the `mlock()` system call pins HDM regions in physical memory. This prevents any swapping or migration of the allocated memory, thereby maintaining a consistent address mapping.

#### C. Adjacency-aware Cluster Placement

Partitioning datasets into clusters for parallel search is common; however, it risks load imbalance if nearby clusters

reside on the same CXL device. This issue is exacerbated when multiple queries target similar regions. We propose *adjacency-aware cluster placement* (Algorithm 1) to distribute adjacent clusters across different devices, enhancing parallelism.

All clusters are initially sorted by size in descending order, prioritizing the placement of larger clusters first. For each cluster, it calculates adjacency penalties (referred to as *loss*) for devices with sufficient capacity (line 3). Penalties increase based on the proximity of neighboring clusters already on a device (lines 5~8). The cluster is assigned to the device with the lowest penalty, the one with greater remaining capacity in case of ties (lines 9~10). As opposed to the CXL-ANNS's hop-count-based round-robin placement that ignores topology, our algorithm considers cluster adjacency.

The host identifies *k*-nearest clusters via centroids and dispatches searches to the corresponding devices. With adjacent clusters distributed, traversals proceed in parallel, maximizing utilization. In Section V-C, we analyze the effect of our cluster placing algorithm.

### V. EVALUATION

#### A. Experimental Setup

To evaluate the performance of COSMOS quantitatively, we developed a simulator integrated with Ramulator [13]. Our setup models a 1TB CXL memory comprising four CXL devices, each with four DDR5-4800 channels and two ranks of 16Gb  $\times$  4 DRAM chips per channel (256GB per device).

We used two representative billion-scale datasets, SIFT1B and DEEP1B, from the BigANN benchmark [12]. We incorporated a clustering mechanism into DiskANN (in-memory mode) [7] and extracted node visit traces from 10,000 queries per dataset to emulate realistic access patterns. These traces were used as input to our simulator to model the memory access patterns of the three main query processing operations: graph traversal, distance calculation, and candidate updates.

The generated memory requests were injected into Ramulator to measure query latency and analyze memory access behavior under various system configurations. We modeled a streaming scenario where queries are dispatched to the first available CXL device, enabling query-level parallelism. To evaluate data placement, we compared our adjacency-aware algorithm with round-robin (RR) placement, which ignores inter-cluster proximity.



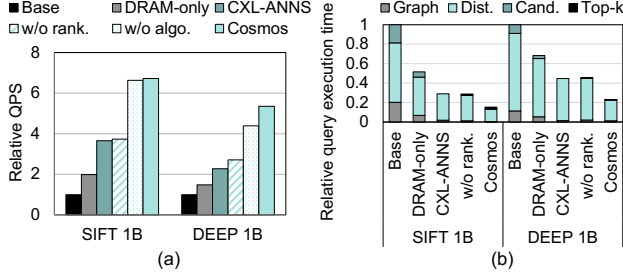


Fig. 4. (a) Relative query throughput (Query Per Second, QPS). (b) Breakdown of query execution time.

### B. Overall Performance

Fig. 4(a) illustrates the relative throughput (Queries Per Second, QPS) of various methods normalized to the **Base**, where all data resides in CXL-memory and computations are performed on the host side. The **DRAM-only** scenario assumes unlimited DRAM capacity, placing all data within DRAM. **CXL-ANNS** [9] improves performance by offloading distance computation, applying fine-grained query scheduling, and hop-count-based graph caching; we reproduced the first two but excluded caching as it is beyond the scope of this work, which only affects graph traversal and has a negligible impact on total latency (Fig. 4(b)). To evaluate each component of **COSMOS**, we evaluated three configurations: (1) without rank-level PU (**w/o rank.**), (2) without the data placement algorithm (**w/o algo.**), and (3) the full system (**COSMOS**).

**COSMOS** achieves the highest performance, improving QPS by  $6.72\times$  (SIFT1B) and  $5.35\times$  (DEEP1B) over **Base**. While **DRAM-only** eliminates host-device data transfers, it is still bandwidth-limited. **CXL-ANNS** performs better by leveraging offloading and scheduling, but frequent transfers and bandwidth bottlenecks remain. **COSMOS** addresses both issues by fully offloading graph traversal to CXL-side GPCs and accelerating distance computation using rank-level PUs.

Fig. 4(b) shows the single query latency breakdown within a single CXL device, excluding the impact of the data placement. **COSMOS** significantly reduces graph traversal and distance calculation latency by combining in-memory execution with rank-level parallelism. **DRAM-only** benefits from reduced data movement, and **CXL-ANNS** reduces latency through scheduling, but neither entirely eliminates bandwidth-related overhead as **COSMOS** does.

### C. Effectiveness of cluster placing

Fig. 5 highlights the effectiveness of our adjacency-aware data placement algorithm (Algorithm 1). To isolate its impact, we fixed all other system configurations and compared against a baseline that distributes clusters across CXL devices in a round-robin (**RR**) manner. Fig. 5(a) shows the load imbalance ratio (LIR) across devices, defined as the maximum device load divided by the ideal uniform load under perfect distribution. Lower values indicate better load balancing.

Across varying `num_probes` (4, 8, and 16), **COSMOS** effectively balances query load across devices, showing consistently lower LIR than **RR**. Fig. 5(b) presents a heatmap of cluster assignments handled per device over 10k queries. Unlike **RR**,

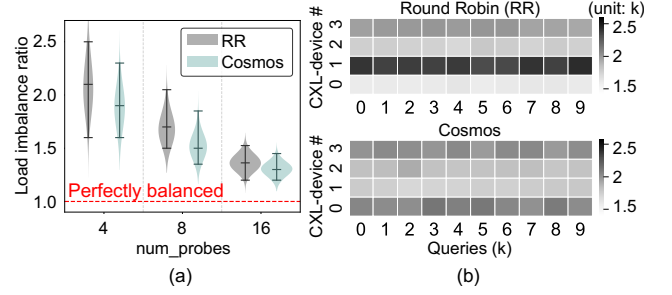


Fig. 5. (a) Load imbalance ratio according to the increasing number of probes. (b) Heatmap showing the number of clusters handled per device.

which leads to uneven device utilization, **COSMOS** ensures a uniform distribution. By relying solely on centroid distances and cluster sizes, without additional profiling, **COSMOS** effectively balances query load across CXL devices, thereby enhancing system scalability and maximizing parallelism.

## VI. CONCLUSION

We have introduced **COSMOS**, a scalable, full in-memory ANNS system designed to overcome the memory bandwidth and data movement bottlenecks inherent in billion-scale vector search. By integrating programmable cores and rank-level processing units within CXL devices, **COSMOS** eliminates host intervention during search and maximizes memory bandwidth utilization through parallel distance computation. Further, we proposed an adjacency-aware data placement algorithm that effectively balances search load across CXL devices by strategically distributing neighboring clusters, enhancing parallelism and scalability. Our evaluations demonstrated that **COSMOS** significantly outperforms existing DRAM-based and prior CXL-based approaches in query throughput and latency.

## REFERENCES

- [1] M. Shen *et al.*, “Towards Understanding Systems Trade-offs in Retrieval-Augmented Generation Model Inference,” 2024, arXiv:2412.11854.
- [2] A. Singh *et al.*, “Agentic Retrieval-Augmented Generation: A Survey on Agentic RAG,” 2025, arXiv:2501.09136.
- [3] B. Tian *et al.*, “Scalable Billion-point Approximate Nearest Neighbor Search Using SmartSSDs,” in *USENIX ATC*, 2024.
- [4] R. Cheng *et al.*, “Characterizing the Dilemma of Performance and Index Size in Billion-Scale Vector Search and Breaking It with Second-Tier Memory,” 2024, arXiv:2405.03267.
- [5] Y. Wang *et al.*, “NDSEARCH: Accelerating Graph-Traversal-Based Approximate Nearest Neighbor Search through Near Data Processing,” in *ISCA*, 2024.
- [6] D. Gouk *et al.*, “Direct Access, High-Performance Memory Disaggregation with DirectCXL,” in *USENIX ATC*, 2022.
- [7] S. J. Subramanya *et al.*, “DiskANN: fast accurate billion-point nearest neighbor search on a single node,” in *NeurIPS*, 2019.
- [8] W. Jiang *et al.*, “Chameleon: A Heterogeneous and Disaggregated Accelerator System for Retrieval-Augmented Language Models,” *Proc. VLDB Endowment*, 2024.
- [9] J. Jang *et al.*, “CXL-ANNS: Software-Hardware Collaborative Memory Disaggregation and Computation for Billion-Scale Approximate Nearest Neighbor Search,” in *USENIX ATC*, 2023.
- [10] Y. Gu *et al.*, “PIM Is All You Need: A CXL-Enabled GPU-Free System for Large Language Model Inference,” in *ASPLOS*, 2025.
- [11] S.-S. Park *et al.*, “An LPDDR-based CXL-PNM Platform for TCO-efficient Inference of Transformer-based Large Language Models,” in *HPCA*, 2024.
- [12] “Big ANN Benchmarks,” 2024. [Online]. Available: <https://big-ann-benchmarks.com>
- [13] Y. Kim, W. Yang, and O. Mutlu, “Ramulator: A Fast and Extensible DRAM Simulator,” p. 45–49, 2016.