Driving Interconnects

Memory and storage fabrics for new AI/ML workloads

A Panel Discussion on

Architecture, Bottlenecks, and System Requirements

Aug 7th, 2025
AI/ML Track: AIML-304-1

the Future of Memory and Storage

Moderator:

- *Siamak Tavallaei,* ***Samsung Semiconductor Inc.***

Panelists:

- *Ardavan Sherafat, AI/ML researcher,* ***Cal Poly, Pomona University***
- *Kurt Keville, Chief Architect,* ***SemiconDx***
- *Manoj Wadekar, AI Systems Technologist,* ***Meta***
- *Elizabeth Leake, Advanced Cyberinfrastructure Project Manager,* ***Texas A&M University***
- *Samir Rajadnya, Principle Architect,* ***Microsoft, Azure***

Panelists:

Ardavan Sherafat, AI/ML researcher, ***Cal Poly, Pomona University***

Ardavan is an AI/ML Researcher at Cal Poly Pomona and a recent M.S. Computer Science graduate, specializing in Data Science, Machine Learning, and AI. With a solid foundation in both academic research and industry experience, he has contributed to innovative projects in computer vision and natural language processing. He brings broad technical expertise across programming, machine learning frameworks, cloud computing, and MLOps. He has designed and implemented a range of applied projects spanning AI systems, cloud architectures, and intelligent automation.

Kurt Keville, Chief Architect, ***SemiconDx***

Kurt works in Research Computing and Systems Design. His MIT thesis work was on energy-efficient supercomputing. He has investigated research enabling and accelerating technologies that can unlock new programming paradigms for grand challenge problems. He currently works on a cluster model as a notional Tactical Datacenter design with strong focus on energy-efficiency, composability, and memory management.

Manoj Wadekar, AI Systems Technologist, ***Meta***

Manoj is a AI Systems Technologist at Meta. He is leading OCP/CMS subproject and has been a great inspiration for memory and interconnect technologies for AI/ML systems.

Panelists:

Elizabeth Leake, Advanced Cyberinfrastructure Project Manager, ***Texas A&M University***

Elizabeth is a project manager of advanced cyberinfrastructure at Texas A&M University.
She founded STEM-Trek, a global, grassroots nonprofit organization that supports travel, mentoring, and professional development opportunities for science, technology, engineering, and mathematics scholars from underrepresented groups and regions. In 2019, Elizabeth formed the Isango project - with the goal of developing a composable, portable and affordable supercomputer in a suitcase (FPGAs, GPUs, and CPUs). She hopes to make the platform available to K-20 schools and development teams who build custom-computing solutions.

Samir Rajadnya, Principle Architect, ***Microsoft, Azure***

Samir is currently a Principal Architect in Microsoft's Azure Strategic Planning and Hardware Architecture (Sparc) team, which is responsible for long-range technology pathfinding for future Azure Cloud systems. Within Sparc, Samir is part of a team responsible for future memory systems. This team investigates future architecture directions for Azure and serves as the primary architecture contact point in technical

Retrieval-Augmented Generation (RAG)

Architecture, Bottlenecks, and System Requirements

2025

AI/ML Track: AIML-304-1

Ardavan Sherafat (ardavan.sherafat@gmail.com)
AI/ML researcher, **Cal Poly, Pomona University**

RAG Definition

A machine-learning architecture that integrates information-retrieval with generative models to provide accurate, grounded, and context-rich responses.

Traditional LLM Limitations:

- Constrained by training data cutoff dates
- Limited context window size
- Cannot dynamically access external data during inference
- Lack domain-specific or real-time information

RAG Advantages:

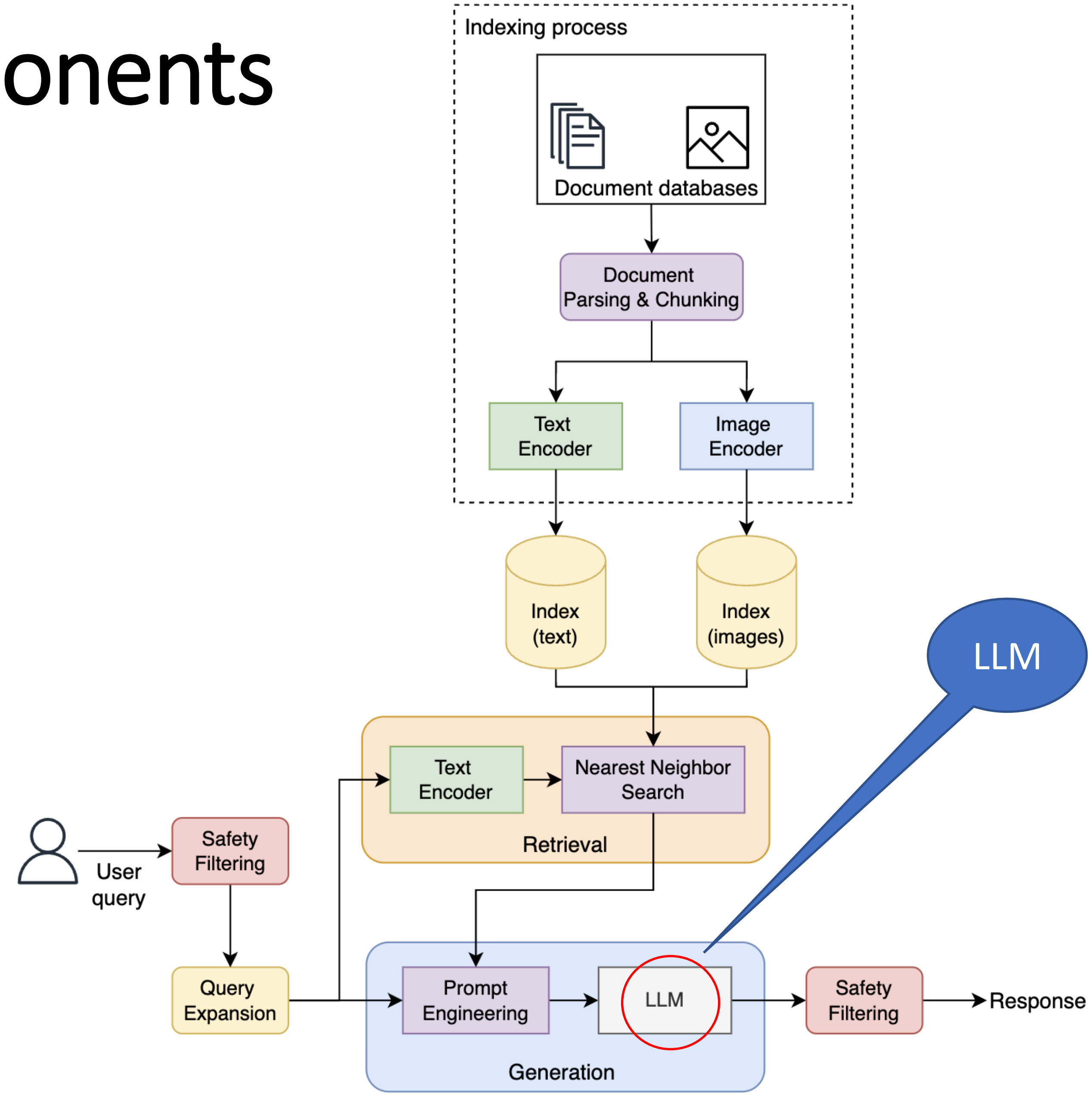
- No retraining required (unlike fine-tuning)
- Not limited by static knowledge (unlike prompt engineering)
- Provides verifiable, traceable sources for the response
- Handles specialized and frequently changing content

RAG Architectural Components

1. Indexing Subsystem
2. Retrieval Engine
3. Generation Component

Three interconnected components working together to deliver accurate, contextual responses.

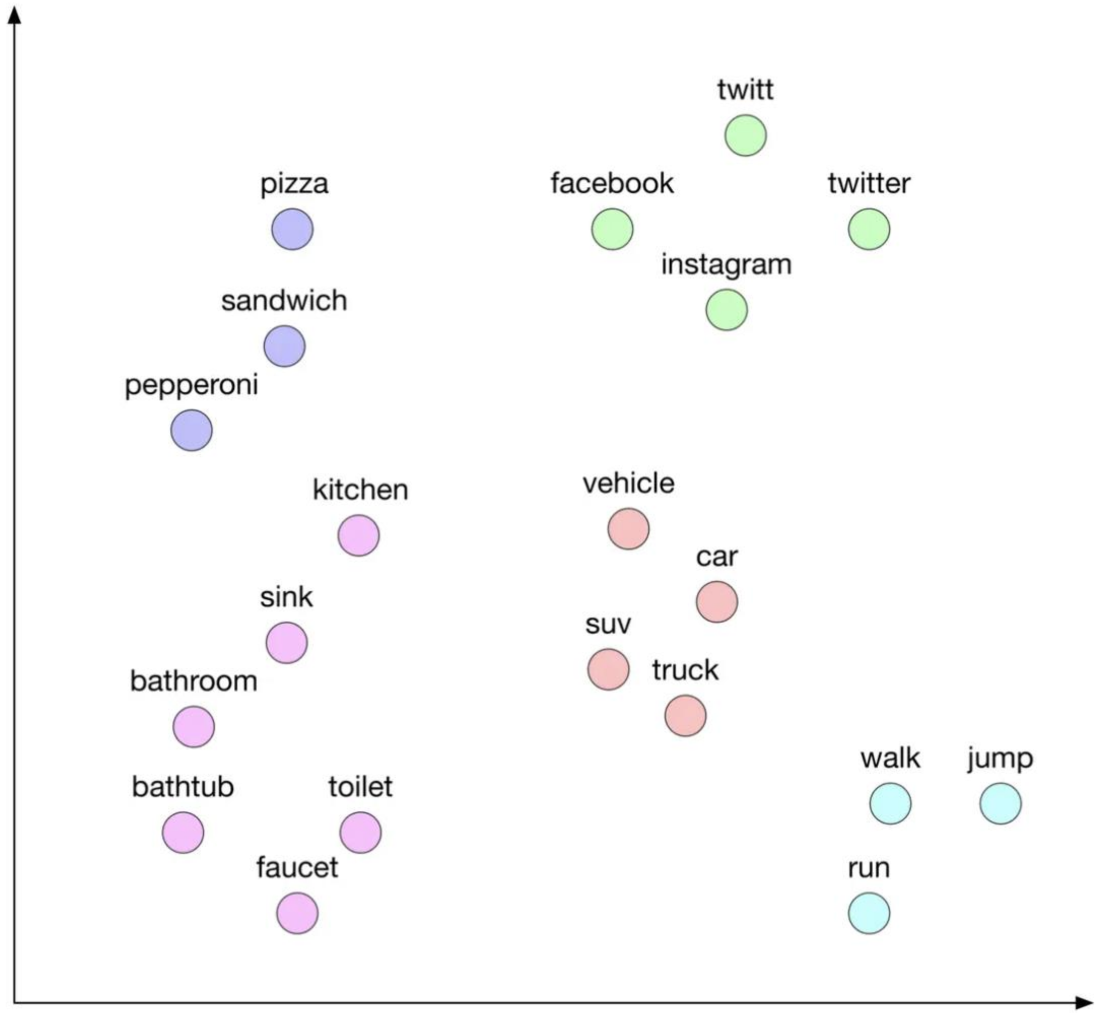
RAG System Architecture [1]



[1] A. Aminian and H. Sheng, Generative AI System Design. ByteByteGo, Nov. 16, 2024. ISBN 978-1736049143.

Indexing Subsystem *(Embedding)*

- Document Processing: Parse PDFs, web pages, images, and scanned files
- Chunking: Divide documents into semantically coherent segments
- Embedding Generation (An embedding is a way to represent data (like words or images) as numbers (vectors) so that a computer can understand and compare them):
  - Text encoders (BERT-based Transformers)
  - Image encoders (CNNs, Vision Transformers)
  - Cross-modal models (CLIP for text-image alignment)
- Vector Storage: Store embeddings in optimized indexes (FAISS, ScaNN, Elasticsearch)



Words in the 2D embedding space [1]

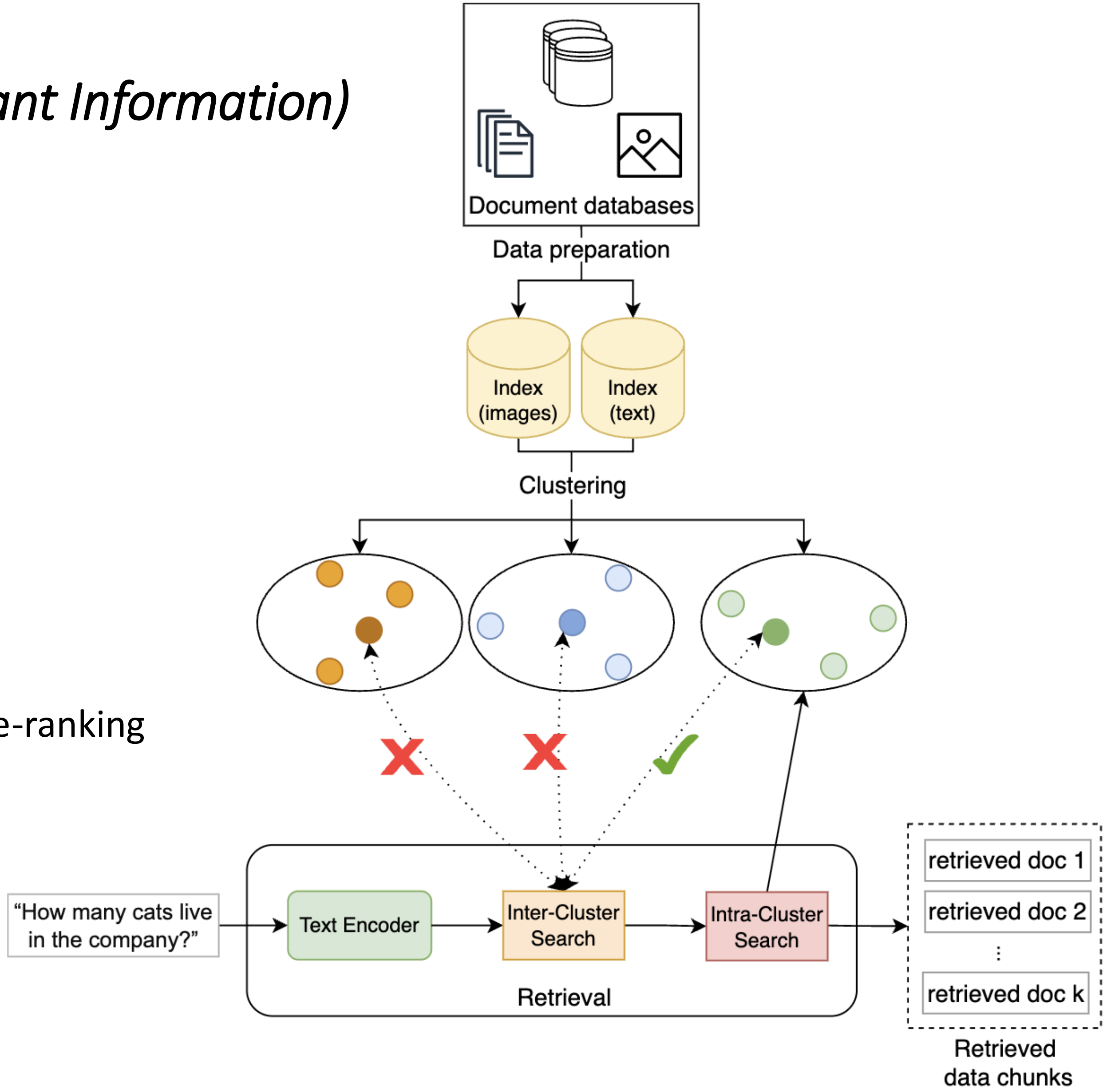
[1] A. Aminian and H. Sheng, Generative AI System Design. ByteByteGo, Nov. 16, 2024. ISBN 978-1736049143.

Retrieval Engine *(Finding Relevant Information)*

- Query Encoding: Convert user query to embedding using same encoder
- Similarity Search: Nearest neighbor search over vector index
- Scalability Solutions using Approximate Nearest Neighbor (ANN) algorithms
  - Tree-Based
  - Hierarchical Navigable Small World (HNSW)
  - k-means clustering
  - Locality-Sensitive Hashing (LSH)
- Quality Control: Relevance thresholds, metadata filtering, re-ranking

Note – Vector DB Size Estimate

- 1536-dim embeddings (e.g., OpenAI text-embedding-ada-002)
- Each vector ≈ 6 KB (float32 × 1536)
- 1B vectors ≈ 6 TB raw
- With metadata & ANN index overhead: 12–24 TB typical
- Large-scale systems (e.g., Google, Meta) may exceed 50–500+ TB



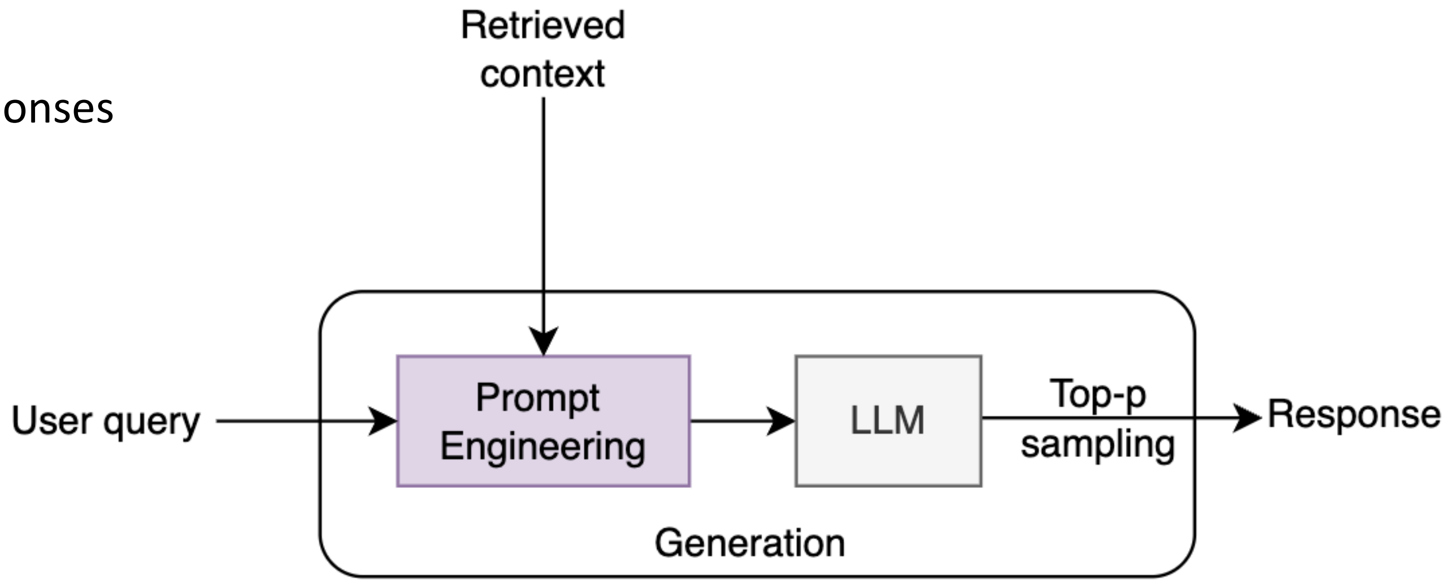
Retrieval component [1]

the Future of Memory and Storage

[1] A. Aminian and H. Sheng, Generative AI System Design. ByteByteGo, Nov. 16, 2024. ISBN 978-1736049143.

Generation Component *(LLM)*

- Automatic Context Insertion: Retrieved passages automatically added to prompts
- LLM Processing: Model processes query + contextual passages
- Advanced Prompting Techniques:
  - Few-shot prompting
  - Chain-of-thought (CoT) reasoning
  - Role-specific prompting
- Produces grounded, verifiable responses



Generation component [1]

[1] A. Aminian and H. Sheng, Generative AI System Design. ByteByteGo, Nov. 16, 2024. ISBN 978-1736049143.

Bottlenecks in RAG Systems

1. **Memory Footprint**
   - Billions of high-dimensional vectors require terabytes of storage
   - Memory/disk trade-offs lead to latency from cache misses
   - Full in-memory storage improves speed but hits capacity limits

2. **Memory Bandwidth**
   - DRAM capacity isn't enough — bandwidth saturation causes bottlenecks
   - CPUs become memory-bound, not compute-bound
   - Near-memory processing highlights the dominance of data access time

3. **Data Transfer Overheads**
   - Vectors must cross PCIe or NVLink to reach GPUs
   - Multi-GPU setups face traffic congestion and NUMA issues
   - Affects responsiveness in interactive applications

Bottlenecks in RAG Systems

4. **Resource Fragmentation**
   - Redundant dataset copies across distributed nodes
   - Wastes memory and increases consistency overhead
   - Lack of shared dynamic memory limits optimization

5. **Generation Phase Limits**
   - Long-context inference pushes state to CPU memory
   - SSD swaps are too slow, large memory is too costly
   - Result: truncated contexts, smaller batches, underused model potential

Observations

Architecture, Bottlenecks,
and System Requirements

2025
AI/ML Track: AIML-304-1

Siamak Tavallaei
Senior Principal Engineer, **Samsung Semiconductor Inc.**

Observations

- We need more memory (*compute capability grows with increased memory*)

- Workloads use different amounts of the available memory footprint during various phases in the pipeline

- If we don't provision enough memory, the size of problems we can solve is limited
- If we maximally provision memory, we end up with under-utilized resources

- Resource-**pooling** based on disaggregated computing helps **inflate** and **deflate** available memory for each processing element at the appropriate phase during the pipeline

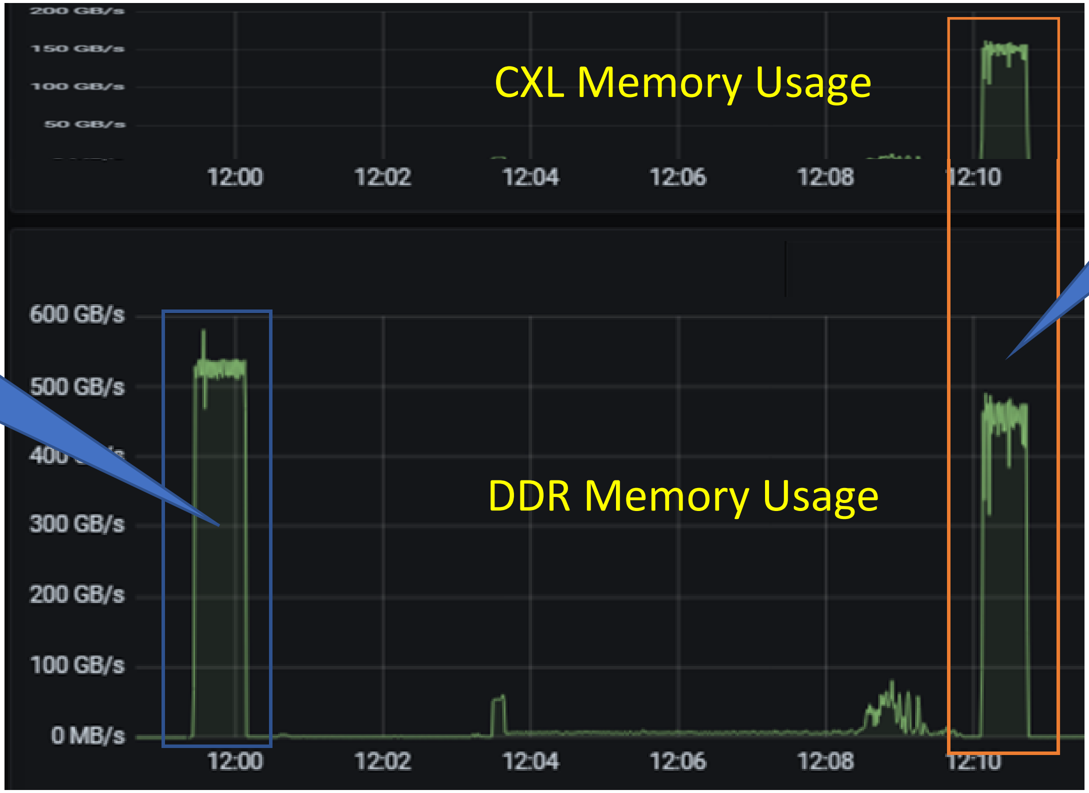
- The results show:
  - Memory capacity and bandwidth utilization throughout the pipeline stages
  - Performance **gain** when enough memory is available during the critical phase
  - With reasonable size of deployed resources (**TCO**)

- Driving solutions based on simulation and lab analysis through the **open-source** efforts

Memory Bandwidth Utilization

Memory System DDR5 Bandwidth
(DDR5-only Memory)

Read B/W : 539GB/s



Memory System Bandwidth
(DDR5 + CXL Memory)

Aggregated Read B/W of
Weighted Interleaving (4:1)
636GB/s

Observations

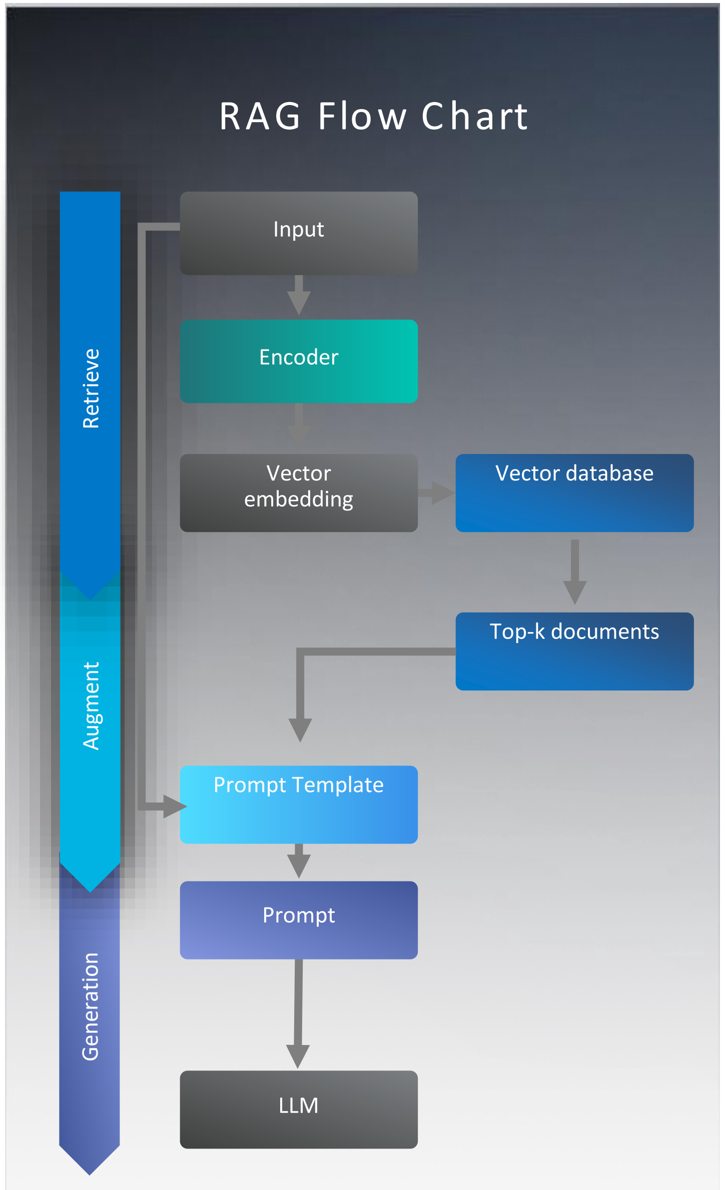
Data-**retrieval** & **Search** Techniques (Databases)

- MemCacheD
- Vector DB
- Key-Value Store and KV-Cache

- Mem **expansion** (large memory footprint to keep processed data and to process more!)
- Memory **pooling** (expand and deflate memory footprint for different phases of the flow)
- Memory **sharing** (for reducing communication time and energy overhead)
- Near-memory & In-memory **Processing**

Pre processing

- Encoding, Embedding
- Document processing
  - organizing and structuring unstructured data from different input types)
  - Reducing data-transfer through summarization and compression
  - All-Reduce, All-Gather, …
- Check-pointing & Restart
- Agentic Role-back



Source: Samsung FMS presentation, Aug 2025

AI System Trends

FMS
2025
AI/ML Track: AIML-304-1

Manoj Wadekar
*AI Systems Technologist, **Meta***

AI Systems Memory Trends:

Memory capacity needs continue to grow

- Driven by model sizes (in addition to activations, KV Cache etc)

Capacity needs drive Tiered memory solutions

- Tier 1 Memory bandwidth needs getting satisfied by HBM community with roadmap
    - HBM4, HBM4e..
- Tier 2 Memory
    - Host attached or GPU attached
    - DDRx: Form Factors for capacity and bandwidth needs (RDIMM, MRDIMM)
    - LPDDRx: RAS, Form Factors

AI Memory Fabrics:
Local and Scale-Up networks

Local Memory Interconnects:

- Provides Tier 1 (HBM) and Tier2 Memory (DDRx, LPDDRx)

Accessing memory on other accelerators (e.g. Collectives)

- Connecting to other devices with memory semantic fabrics
  - (E.g. CXL, NVL, UAL, RDMA (IB/RoCE)..)
- High-bandwidth, Low-latency, SW-coherent,
- Memory Semantic (Load/Store), Channel Semantic (DMA, Send/Receive)
- Scale-up network sizes:
  - 8 -> 72 -> 144 ->..

How Academia sees Bottlenecks in RAG Systems

2025

AI/ML Track: AIML-304-1

Kurt Keville, (kurt@semicondx.com)

Chief Architect, **SemiconDx**

How Academia sees Bottlenecks in RAG Systems

1. **Memory Footprint**
   - Historically, Research Computing (RC) Datacenter sysadmins recommend traditional workarounds for memory bound HPC applications. University support teams are already using standard practices for IT management and support. Users generally get a filesystem space allocation and if they need more space they can attach and use a temporary partition for a limited period of time. If they need more space for a memory-centric job, they can create a swap partition on that filesystem. Likewise, most Universities will offer to advise on RDMA, NUMA and remote memory coding techniques.
   - These are generally services provided by the University support staff and are not fixes that are expected of the application programmers. To that end, many schools offer classes and consulting services for these advanced processes.
2. **Memory Bandwidth**
   - University RC services preach the gospel of "Reduce, reuse, recycle." They are highly motivated to train students up on efficient coding practices to lower the electricity bill at their shared services. The incentive for the students to write (energy) efficient code is that their jobs will run faster. Programmers learn and utilize the practice of determining which values are candidates for cache and hot memory location management, and in turn they rewrite their code accordingly to take advantage of that.
3. **Data Transfer Overheads**
   - Reassess and profile application code to keep as much work on the initiating server. Data movements that cross boundaries (system memory bus to PCI bus or PCI to network card, etc) are considerably more resource intensive than otherwise. Here again, there are tutorials on how to make production codes data parallel.

How Academia sees Bottlenecks in RAG Systems

4. **Resource Fragmentation**
   - Dedupe is already quite popular at RC centers in academia. If you want to use a framework that is common and which a lot of students are already using, you can perform a simple conda install, often at login if you use it a lot. Many students still perform a full pip install if they are not sure of the library versions, but they use the rewritable partition for that. Ideally, the sysadmins can write lock memory as well as filesystem locations. This has been very difficult to manage.

5. **Generation Phase Limits**
   - Many of the features that we considered assets of SSD are no longer priorities in this community. We do not need to save much of the interim data generated by student programmers. They sometimes save object files if they believe there is an archive requirement but students traditionally perform a ”make clean” operation whether they need to or not since it requires the least amount of debugging of a build. To that end, an operation of this sort is best performed on a ramdisk or equivalent. The potential for creating these data structures are facilitated by CXL. Now, not only is the sysadmin resigned to the end user exercising legacy compilation techniques but they may recommend it now if it is the best operation.

Why CXL?

ACES

Disaggregated, composable platform at Texas A&M University

Elizabeth Leake

ACES
ACCELERATING COMPUTING FOR EMERGING SCIENCES

TACC
TEXAS ADVANCED COMPUTING CENTER

High Performance Research Computing
DIVISION OF RESEARCH

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

Why CXL? Lessons Learned in Academia

40-year trajectory unsustainable (old standard inefficient/energy/storage)

- *Next-gen arithmetic promising*

HPC for general use can't easily support fully-composable elements

- *Training and development best accommodated with small platform, like Isango*

Large models require agile memory (I/O, cache); CXL® facilitates data management

- *Plethora of memory solutions emerged to solve this issue - again, it isn't sustainable*

Agency-funding slow. End-to-end can be a year (one or two new iterations released in that timeframe).

- *Spec open-ended agreement where vendor provides new hardware as it is released*

Reasons for Additional memory attach

2025

AI/ML Track: AIML-304-1

Samir Rajadnya

*Principle Architect, **Microsoft, Azure***

Reasons for additional memory attach

CPU Platforms:

Memory stranding

As CPU cores increase, more memory is required (maintain core:memory ratios)

Memory is not scaling at the same rate as CPU cores

Higher capacity DIMMs (3DS) cost ~2x of lower capacity DIMMs

Sometimes we need more capacity than possible through locally attached

Long term, CPUs may move away from multi-socket architectures

GPU Platforms:

KV-Cache

HBM capacity limitation

The State of Cloud Memory

Memory is becoming the largest portion of server costs for Cloud Datacenters

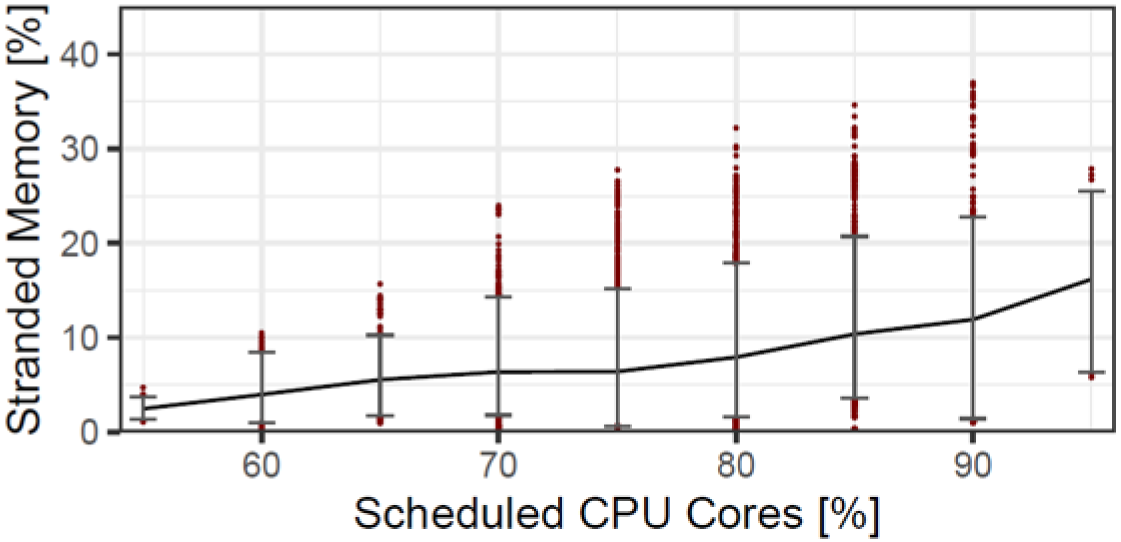
DRAM can account for up to 50% of server costs

Often that memory is poorly utilized

Up to 25% is *stranded*
(no free CPU cores but unsold memory)

Customers also overprovision memory

Median VM: **45%** untouched memory



Memory Pooling Can Help

Simple idea:

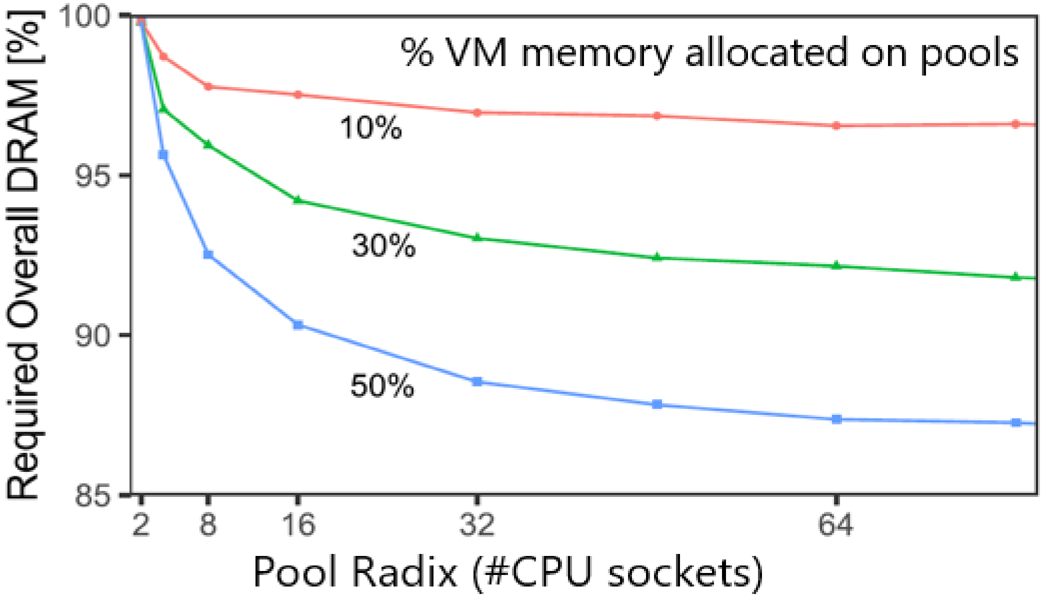
- Share stranded memory with nearby servers that have free cores

Analysis results:

- Even small pool sizes are useful
- This does **not** fit the "triangle" model!

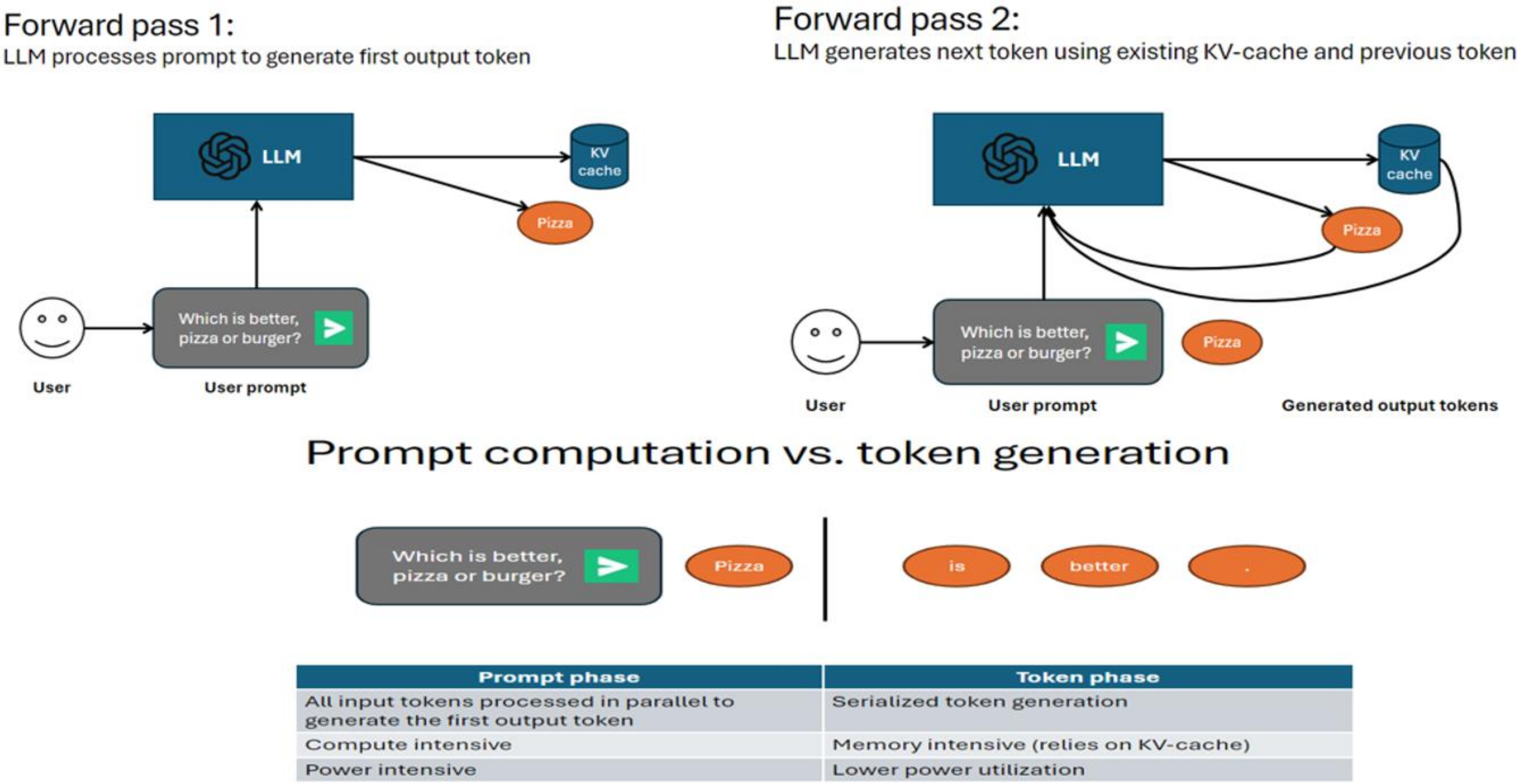
Challenge

- Managing performance of workloads in hybrid memory



Exploration of some solutions:

Pond: CXL-Based Memory Pooling Systems for Cloud Platforms – ASPLOS 2023



Forward pass 1:

LLM processes prompt to generate first output token

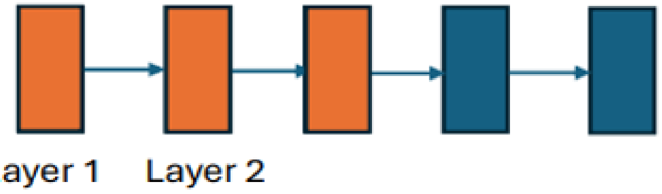
Forward pass 2:

LLM generates next token using existing KV-cache and previous token

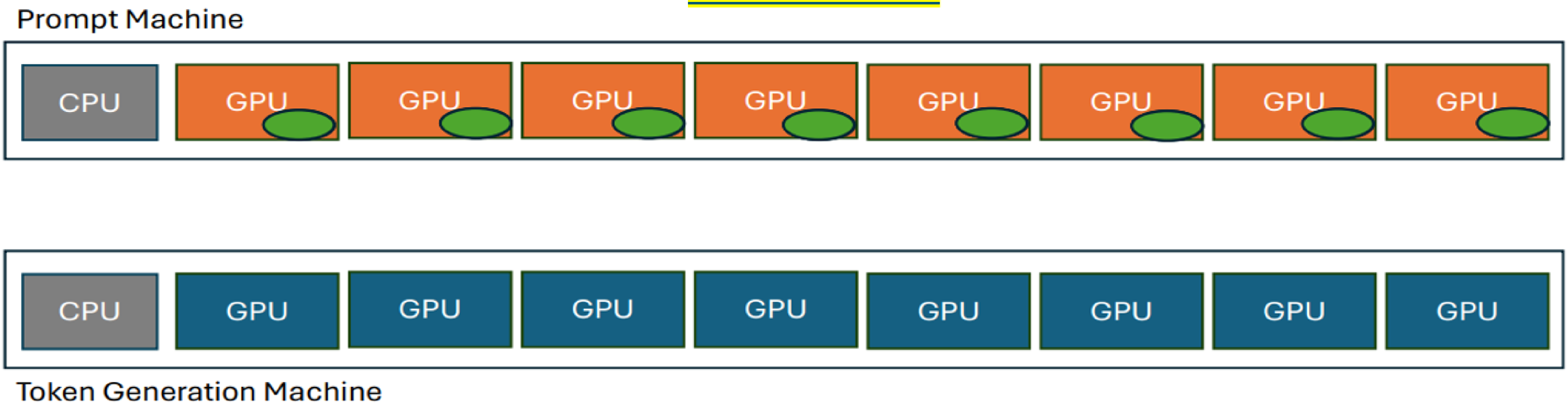
Prompt computation vs. token generation

Prompt phase	Token phase
All input tokens processed in parallel to generate the first output token	Serialized token generation
Compute intensive	Memory intensive (relies on KV-cache)
Power intensive	Lower power utilization

Splitwise: Layerwise KV-cache transfer



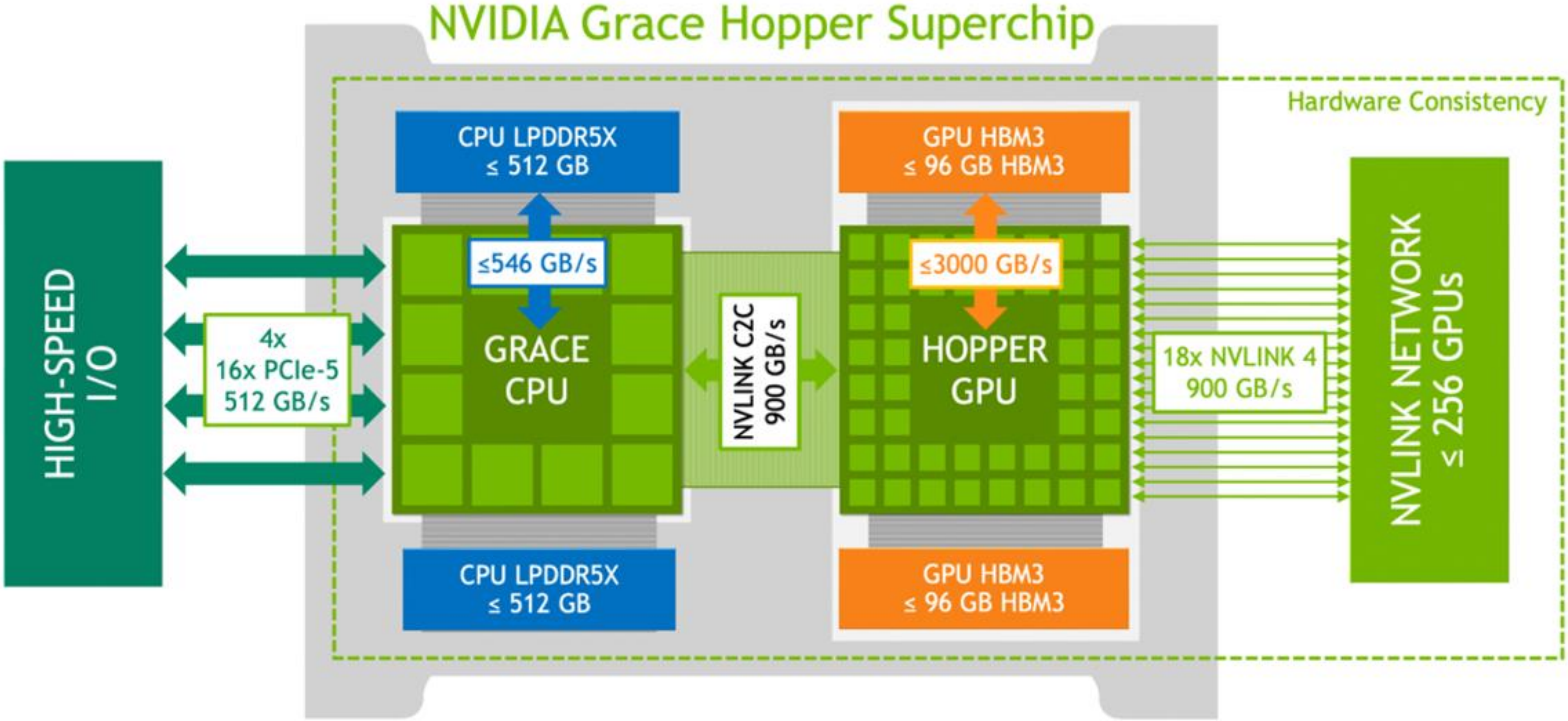
Configurable secondary memory makes it possible to reconfigure GPUs as higher memory token machine



● : 1/8th of the KV-cache for an 8-GPU model

Splitwise has less than 0.8% impact on the end-to-end inference time

GPU shoreline => HBM capacity limitation => Additional memory attach points



the Future of Memory and Storage

Interconnects

PCIe is everywhere

CXL® has reached maturity

New emerging interconnects solve limitations of PCIe/CXL

Time-to-market is important.

Think what we can do right **now** with what is available!

Extra

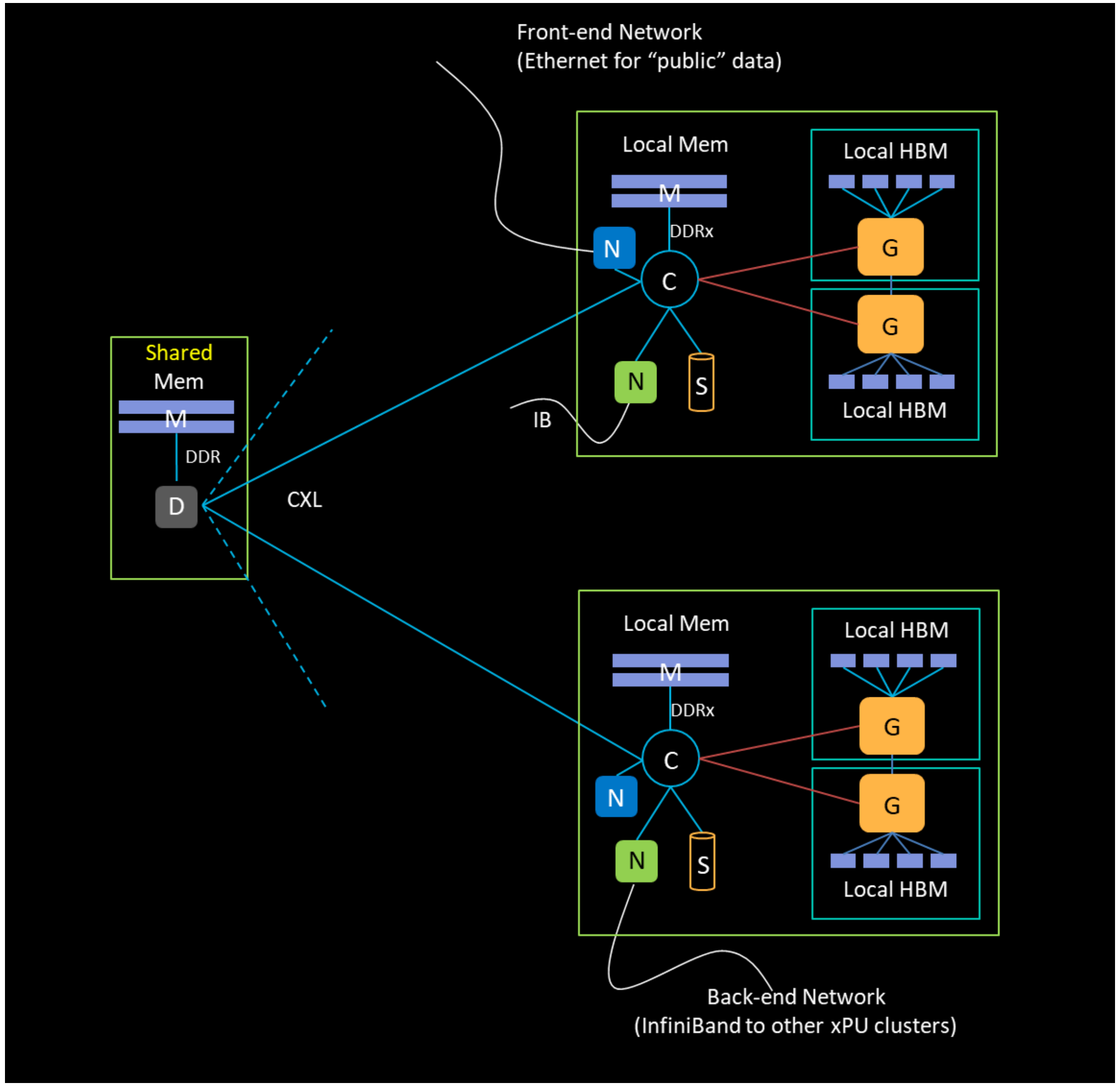
Potential HW Solutions

2025

AI/ML Track: AIML-304-1

Siamak Tavallaei

Senior Principal Engineer, **Samsung Semiconductor Inc.**


Front-end Network
(Ethernet for “public” data)
Local Mem
M
DDRx
N
C
Local HBM
G
G
Local HBM
N
S
IB
Shared
Mem
M
DDR
D
CXL
Local Mem
M
DDRx
C
N
N
S
Local HBM
G
G
Local HBM
Back-end Network
(InfiniBand to other xPU clusters)

Memory Bandwidth and Capacity **Expansion**

Baseline:

- **RDIMMs** are the baseline.

More Bandwidth:

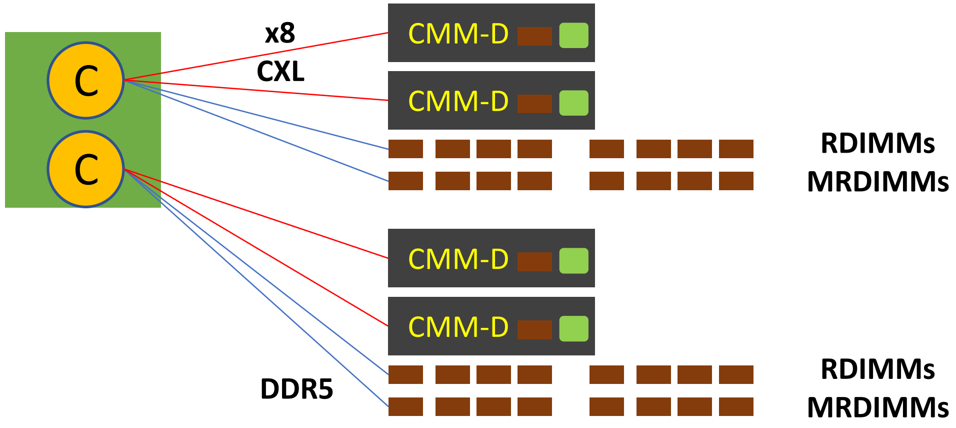
- **MRDIMMs** offer more bandwidth. They can offer more capacity in a Tall DIMM form factor.

More Bandwidth, Capacity, and Pooling/Sharing:

- **CXL memory modules** offer more bandwidth, even more memory capacity, and provide a means for memory-pooling/sharing to inflate and deflate memory footprint during different phases of the AI/ML pipeline.
- **Reduce** overall **TCO**

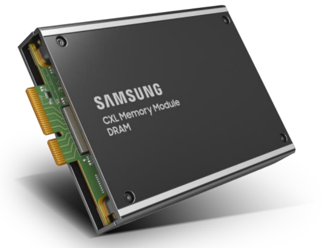
2S Servers
4 x CXL Memory Controllers
x8 CXL Links

CXL Mem (E3.s): **256 GiB** per CMM-D
4 x 256 GiB = 1 TiB per server



8 x DDR5 Channels per CPU (2DPC)
2 x 8 x 2 x 64 GiB = 2 TiB per server

Local Mem (32 x RDIMMs)

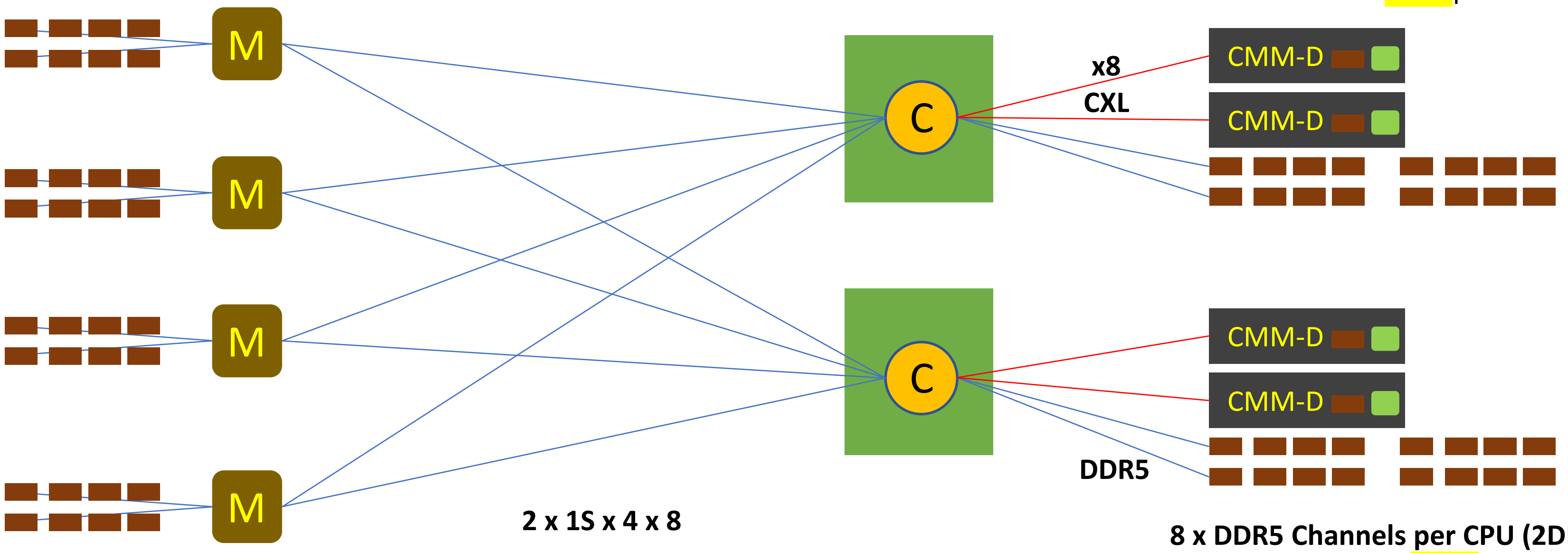


Memory **Expansion** + Memory **Pooling** (CMM-D & RDIMMs)

Pooled/Shared Memory
256 GiB RDIMMs

Two 1S Servers
4 x CXL Memory Controllers
x8 CXL Links

CXL Mem (E3.s): **256 GiB** per CMM-D
4 x 256 GiB = 1 TiB per server



4 x DDR5 Channels, 2DPC
4 x 2 x 256 GiB = **2 TiB**

2 x 1S x 4 x 8

Two 1S servers and **4** MCs
using **x8** CXL Links

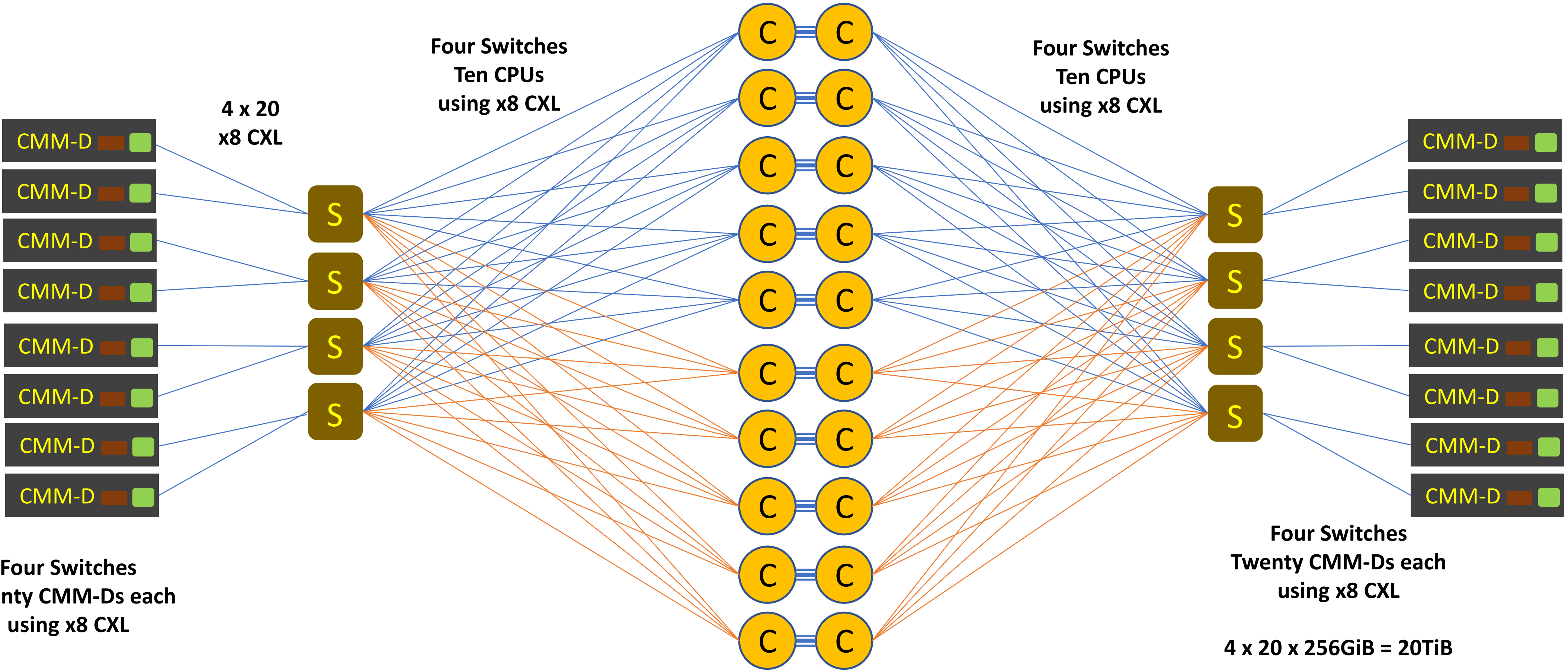
8 x DDR5 Channels per CPU (2DPC)
2 x 8 x 2 x 64 GiB = 2 TiB per server

Local Mem (32 x RDIMMs)

4 x 8 = **32 DIMMs**
1280 DRAM devices
32 x 256 GiB = **8 TiB** per server

Memory Pooling/Sharing with ten 2S Servers, 8 Switches, and 160 CMM-Ds

10 x 2S x 8 x 8



4 x 20
x8 CXL

Four Switches
Ten CPUs
using x8 CXL

Four Switches
Ten CPUs
using x8 CXL

Four Switches
Twenty CMM-Ds each
using x8 CXL

4 x 20 x 256GiB = 20TiB

Four Switches
Twenty CMM-Ds each
using x8 CXL

4 x 20 x 256GiB = 20TiB

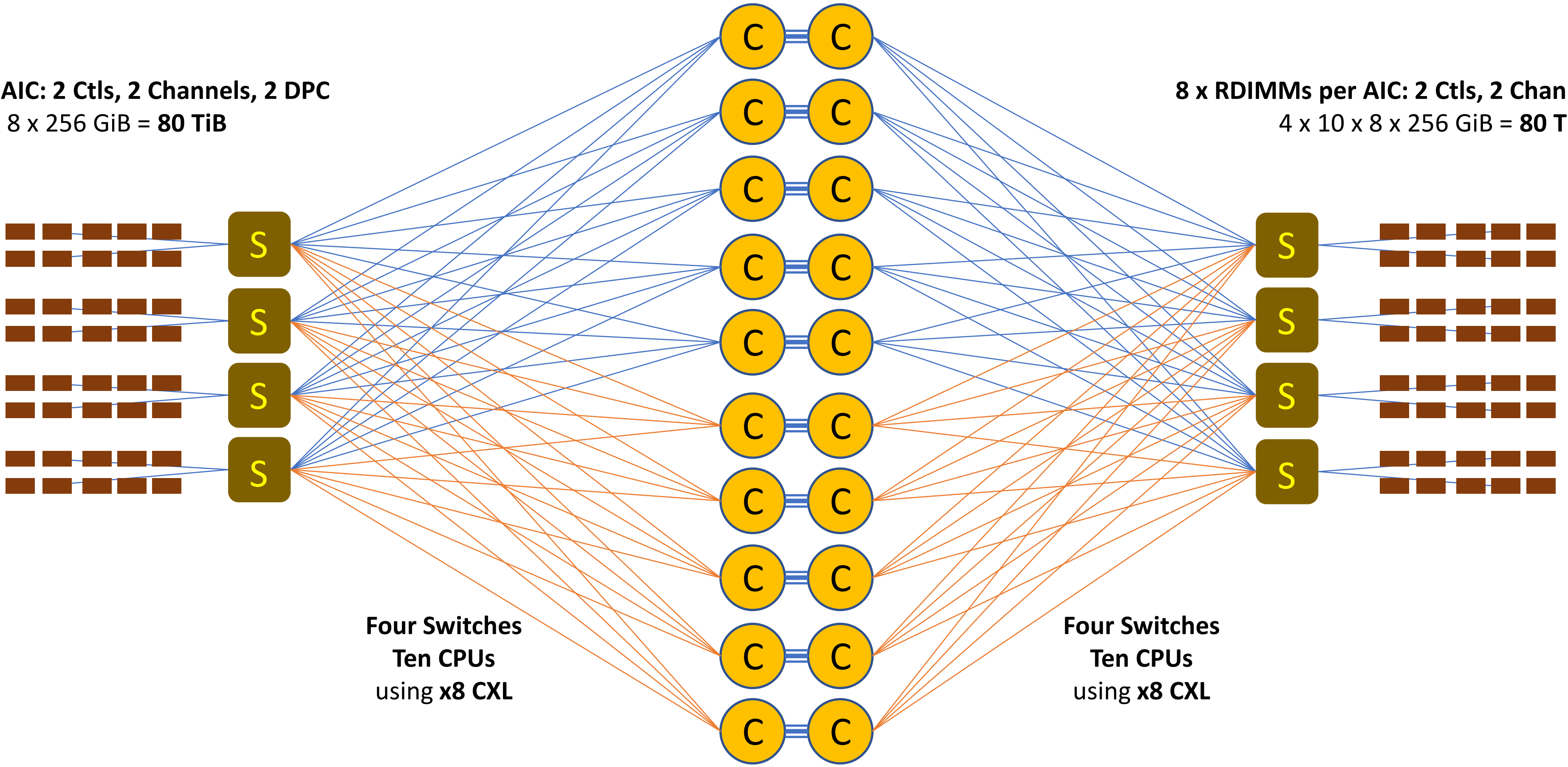
10 x 2S Servers
8 CXL Switches
x8 CXL Links
160 CMM-Ds
40 TiB of Shared DRAM

Memory Pooling/Sharing with ten 2S Servers, 8 Switches, and 640 RDIMMs

10 x 2S x 8 x 8

8 x RDIMMs per AIC: 2 Ctls, 2 Channels, 2 DPC
4 x 10 x 8 x 256 GiB = **80 TiB**

8 x RDIMMs per AIC: 2 Ctls, 2 Channels, 2 DPC
4 x 10 x 8 x 256 GiB = **80 TiB**



Four Switches
Ten CPUs
using **x8 CXL**

Four Switches
Ten CPUs
using **x8 CXL**

10 x 2S Servers
8 CXL Switches
x8 CXL Links
8 CXL Switches
8 x 10 x 8 = 640 RDIMMs
160 TiB of DRAM

FMS
the Future of Memory and Storage

CXL® Demos at Samsung Booth

① CXL DSA Demo	- CMM-H memory pooling demo using Liqid system (Research Project)
② CXL DSK Demo 1	- CXL benefits in RAG application
③ CXL DSK Demo 2	- Collaboration for CXL Memory Pooling Technology - Montage, H3 Platform, and Samsung

Product Display & Demo		
④ Liqid system + CMM-H 	⑤ CMM-D 256GB 80-DRAM + Laptop 	⑥ H3P system + CMM-D

Backup

Among Many AI Applications, why RAG?

Retrieval-Augmented Generation (RAG)

- Can be performed on CPU

  - Almost the only AI application that is more advantageous in CPU than GPU, giving optimization opportunities to memory products (DIMM, CXL, SSD) other than HBM.

- Selected in real world

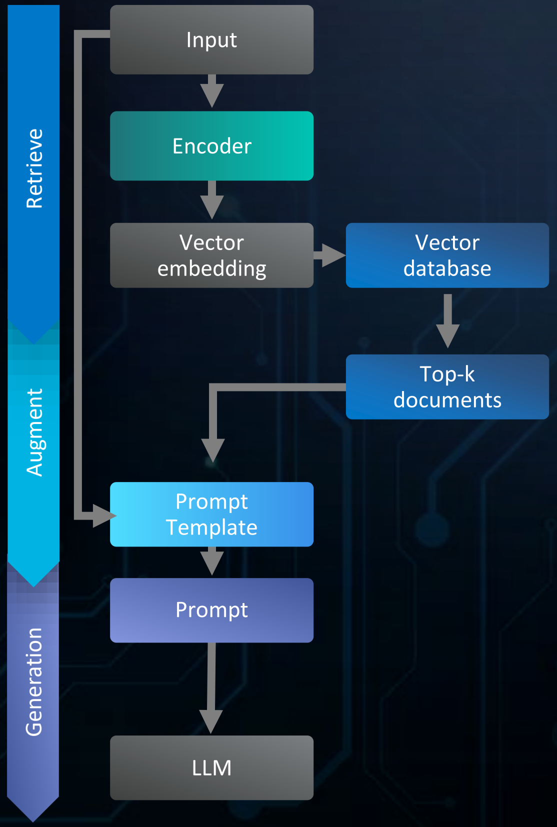
  - No one can have a huge foundation model and the way to increase accuracy by using RAG in a lightweight small LLM model is becoming a trend.

- Requires a lot of memory and has bandwidth-intensive characteristics

  - The size and size of the dataset continue to increase and index building for embedding and search requires a larger memory than the original data.

  - The Approximate Nearest Neighbor Search (ANNS) method for vector search is bandwidth sensitive, so when bandwidth increases, performance also increases.

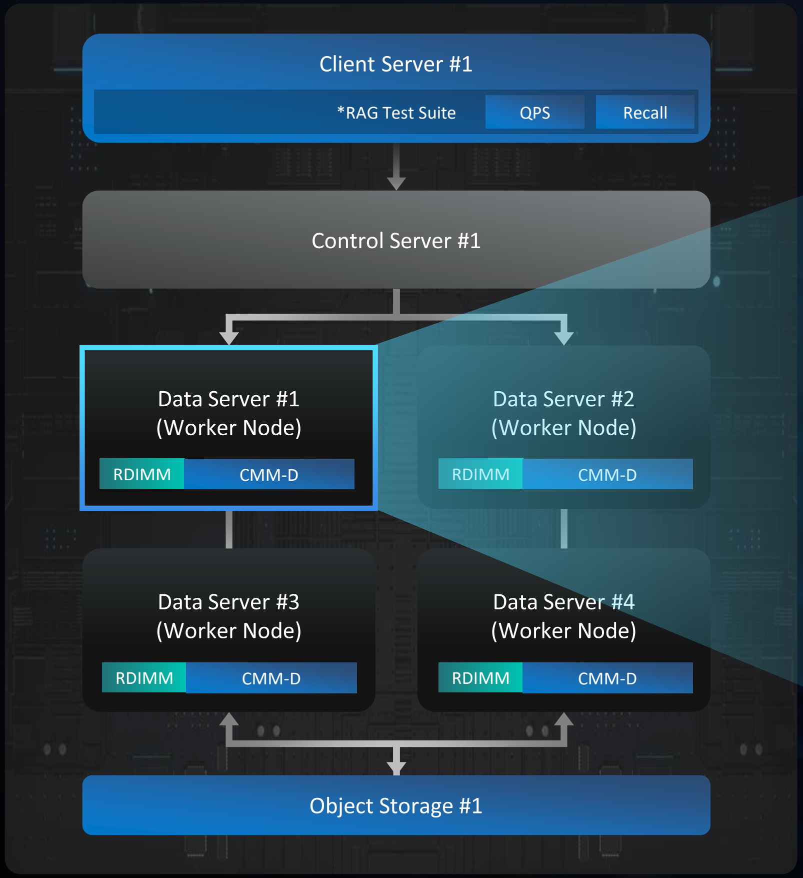
RAG Flow Chart



Source: Samsung FMS presentation, Aug 2025

Samsung Platform for Evaluating CMM-D in RAG Cluster

Architecture of RAG Cluster and RAG Test Suite



Query Node	Executes queries against data.
Index Node	Creates manages indexes for data retrieval.
Data Node	Stores and manages data.

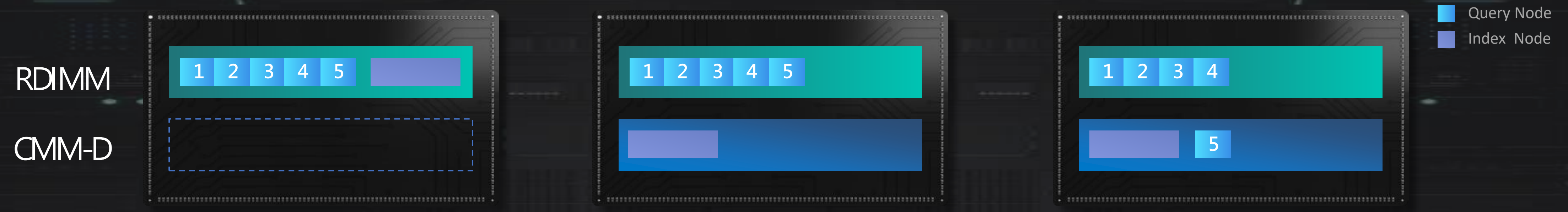
*RAG Test Suite
: Provides query search evaluation and management on the RAG platform.

Source: Samsung FMS presentation, Aug 2025

Advantages : Workload Distribution

Provides sustainable performance through workload separation to CMM-D.

- **The data server's workload consists of query node, index node and data node can operate simultaneously.**
- **Separating the query node and index node workload into RDIMM and CMM-D maintain QPS of data search.**

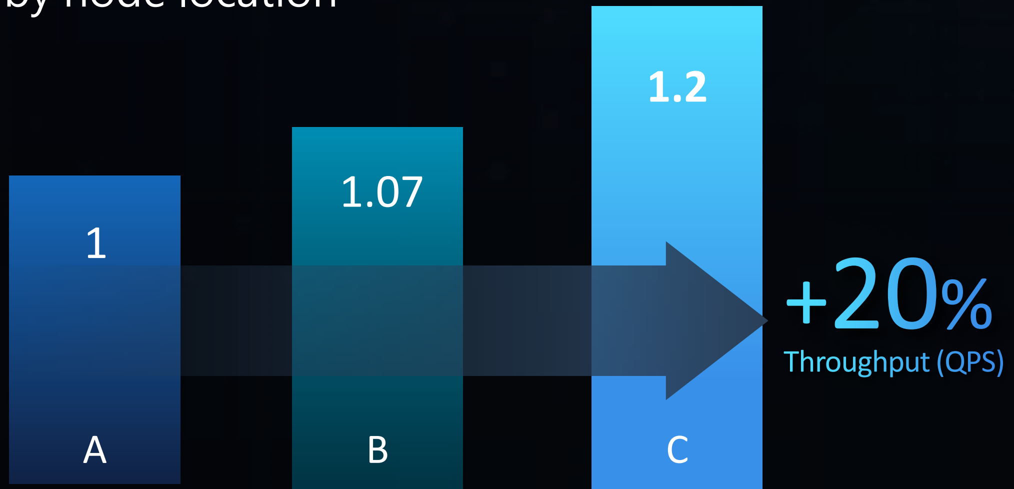


(a) Query Node & Index Node in RDIMM

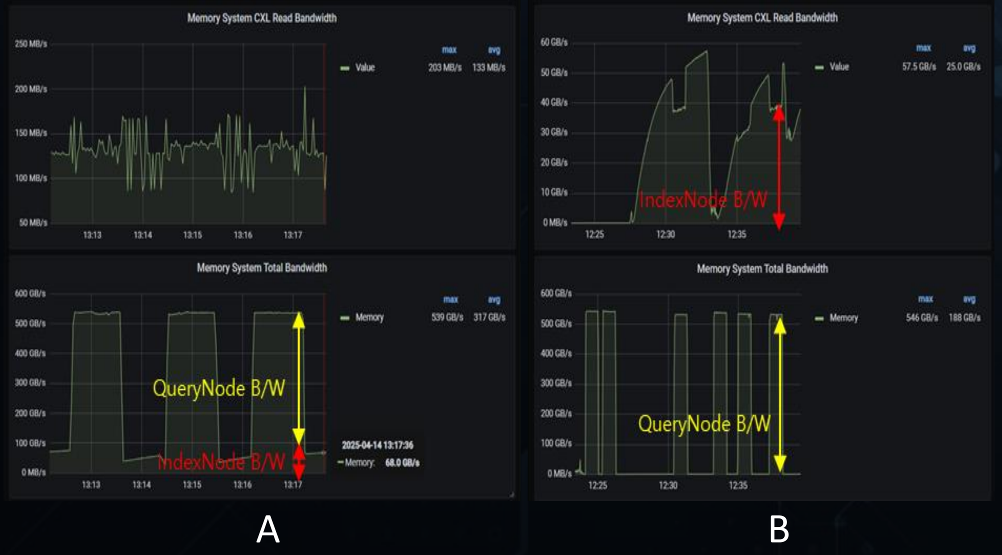
(b) Query Node in RDIMM Index Node in CMM-D

(c) Query Node (w.i. 4:1) Index Node in CMM-D

Normalized performance result by node location



RDIMM BW (Max) : 550 GB/s



Source: Samsung FMS presentation, Aug 2025

What is a RAG Pipeline

Core components

- Data
- Model
- Embeddings
- Query

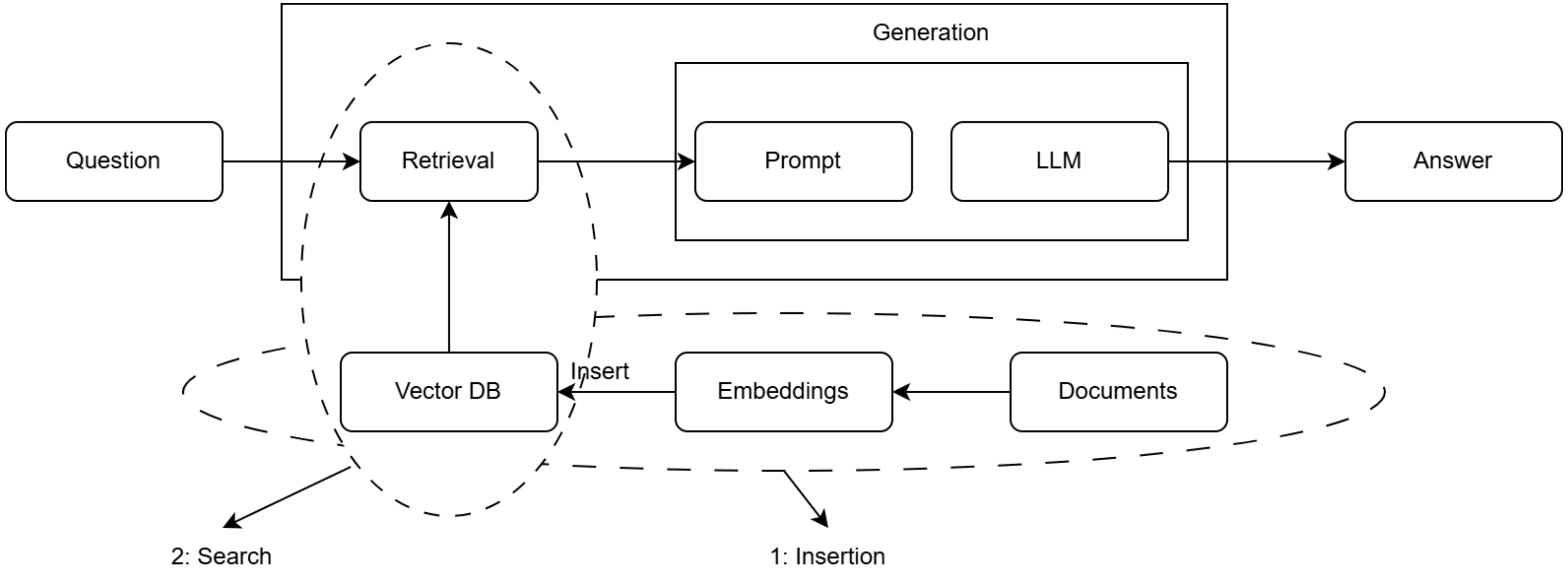
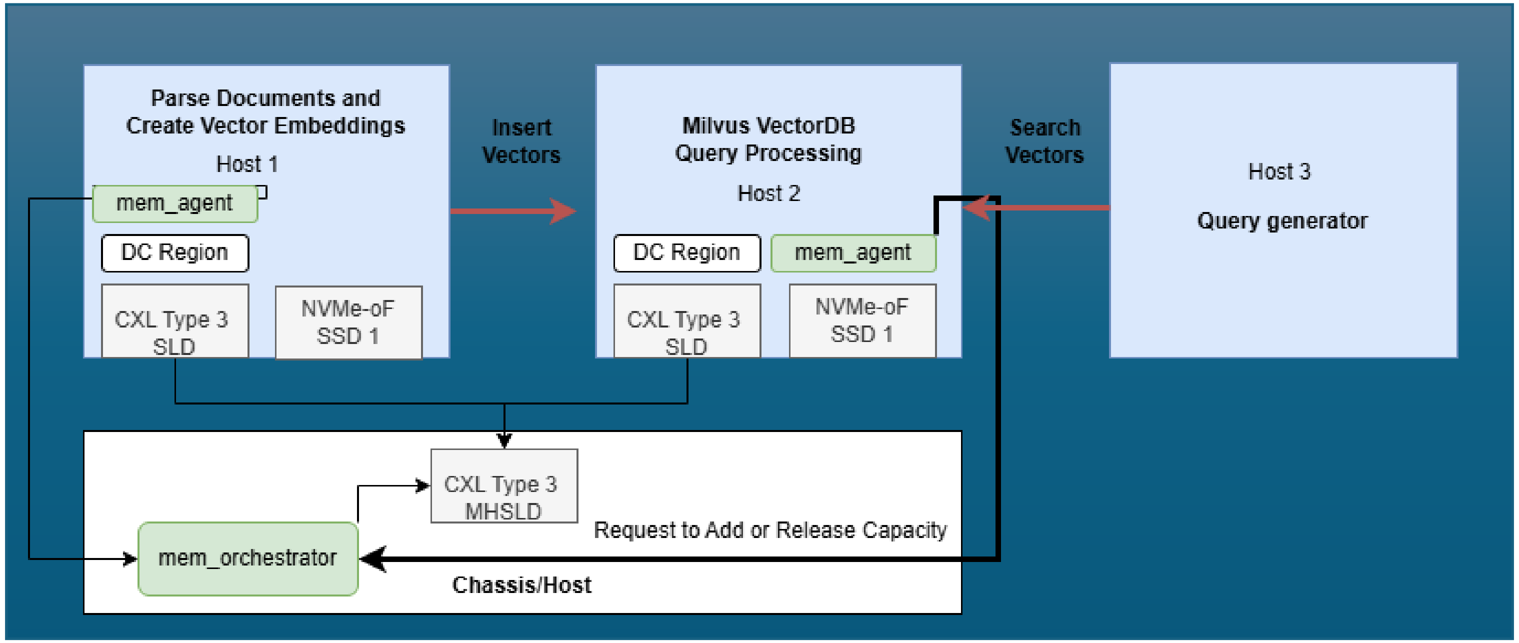


Figure based on[1]

Memory Demand

Phased Approach

- Generate Embeddings
  - Memory demand spikes
- Running the pipeline
  - Based upon the app



[1] Accelerating Data Retrieval in Retrieval Augmentation Generation (RAG) Pipelines using CXL - MemVerge

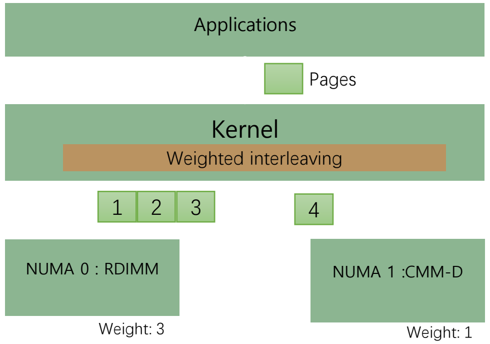
Advantages for RAG Cluster with CXL Memory

Up to 19% higher performance with CMM-D in VectorDB search compared to DRAM case in Milvus RAG cluster

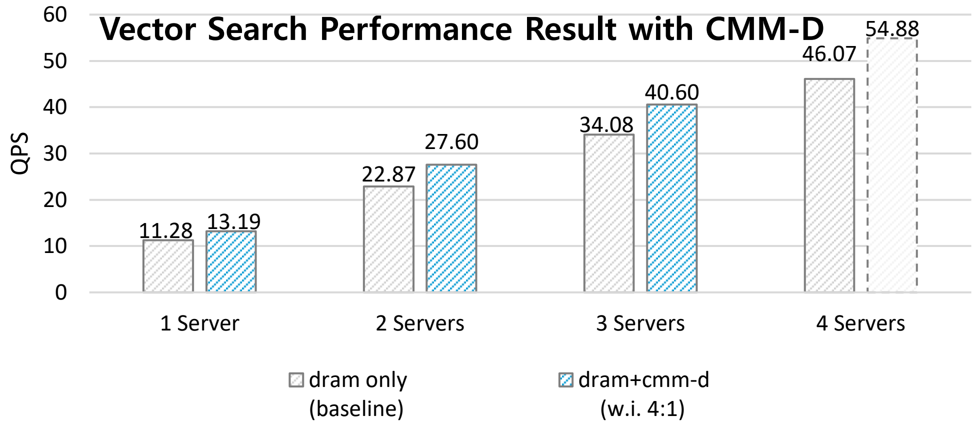
- Performance gain with bandwidth expansion through the CMM-D in Milvus RAG Cluster
- Using SW interleaving (between DRAM and CMM-D) to achieve optimal CXL bandwidth performance

****Weighted Interleaving**

- Linux kernel SW weighted interleaving provides opportunity to define an interleave ratio to best utilize DRAM and CXL memory for optimal performance in a workload
- Included in Kernel Mainline (v6.9)



Comarison of QPS by Number of Servers

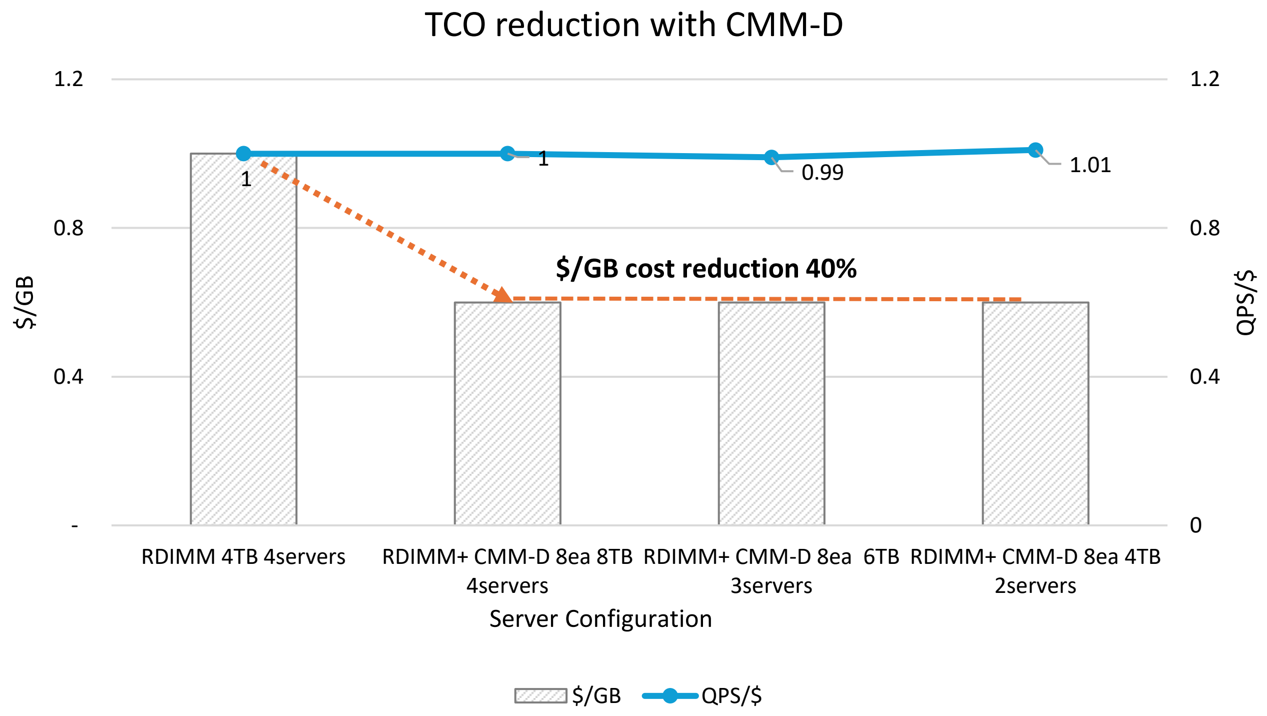


Compute Express Link® and CXL® are trademarks of the Compute Express Link Consortium.

Advantages with CMM-D in RAG Cluster

TCO reduction effect and memory expansion effect can be secured

- Equivalent QPS/$ and 40% reduction in $/GB cost
- Operating Power reduction through application can reduce operating cost.

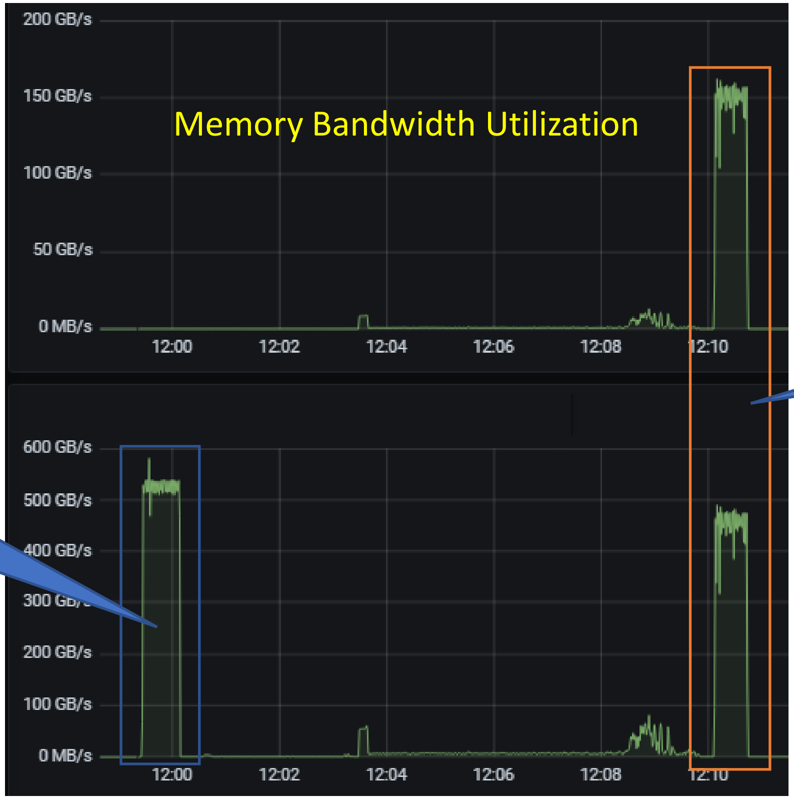


Dataset : MSMARCO-V2

Raw Size	Indexing Size(HNSW)	Entity Count	Dimension	Precision	Vector Size
290GB	673GB	138 Million	1024 (cohere)	FP32	4096B

Reference TCO Calculator :https://v0-cxl-tco-2-nvdatd.vercel.app/

Memory Bandwidth Utilization



Memory System DDR5 Bandwidth
(DDR5-only Memory)

Read B/W : 539GB/s

Memory System Bandwidth
(DDR5 + CXL Memory)

Aggregated Read B/W of
Weighted Interleaving (4:1)
636GB/s

the Future of Memory and Storage

Observations

- We need more memory (compute capability grows with increased memory)

- Workloads use different amounts of the available memory footprint during various phases in the pipeline

- If we don't provision enough memory, the size of problems we can solve is limited
- If we maximally provision memory, we end up with under-utilized resources

- Resource-**pooling** based on disaggregated computing helps **inflate** and **deflate** available memory for each processing element at the appropriate phase during the pipeline

- The results show:
  - Memory capacity and bandwidth utilization throughout the pipeline stages
  - Performance **gain** when enough memory is available during the critical phase
  - With reasonable size of deployed resources (**TCO**)

- Driving solutions based on simulation and lab analysis through the **open-source** efforts

Session Outline

Track:	**AI/ML**
Session:	**AIML-304-1 (Thurs 1:25pm-2:30pm) (Panel)**
Title:	**Driving Interconnects: Memory and storage fabrics for new AI/ML workloads**

Abstract:

AI/ML applications demand on memory sub-system is driving higher memory performance, lower latency, and increased capacity requirements. Memory-tiering supports the first two metrics by offering different memory technologies; however, the traditional method of addressing a larger virtual memory footprint has relied on storage-class solutions such as NVMe SSDs. Innovations in interconnect standards such as CXL and UALink as well as advancements in PCIe and Ethernet physical layers help support higher date-movement requirements, while innovations in memory buffer-caching with SSDs enable increased virtual-memory capacity without significantly impacting latency. This panel provides different perspectives on the role of storage and related memory architectures that support the growth in AI/ML application requirements.

Panel Format: (60 min)

- 35 min: (four panelists)
  - 11 min: RAG Workflow and bottlenecks as a representative example (several slides)
  - Three 8-min presentations with 2-3 slides to set the background for each panelist along with a commentary
- 15 min panel Q&A for all panelists (suggest questions)
- 10 min audience Q&A

Content:

- Overview of applications, algorithms, software flow diagrams, block diagrams, etc
- Decomposition of the requirements into Compute, Memory, Storage, & Interconnect
  - Compute: GPU, CPU, and xPU
  - Memory: Latency, BW, and Capacity
  - Storage: IOPS (and Capacity)
  - Interconnect: Scale-Up, Scale-Out, Networking, etc

- Use-cases:
  - Training & Inference
  - Gen AI & Ranking/Recommendation
  - KV-Cache & RAG

- Gaps and bottlenecks

- Solutions:
  - Current HW Block Diagrams
  - Future roadmap

Q&A

1. How are AI/ML workloads affecting memory and storage requirements?
2. What are the new aspects of AI/ML that are different from the needs of traditional database and HPC applications?
3. What are the new bottlenecks that AI/ML workloads expose? (memory, storage, and interconnect)
4. How are we addressing these requirements using the current hardware and software techniques?
5. What new techniques will we need?