Semantic Data Placement for CXL Memory Systems

Allen Aboytes, Pankaj Mehra
Center for Research in Systems and Storage
University of California, Santa Cruz

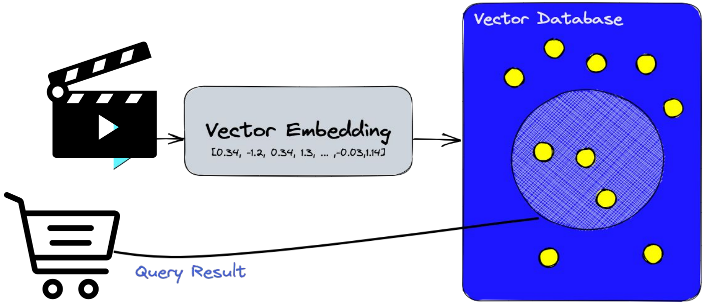
UC SANTA CRUZ
BaskinEngineering

Center for Research
in Systems and Storage

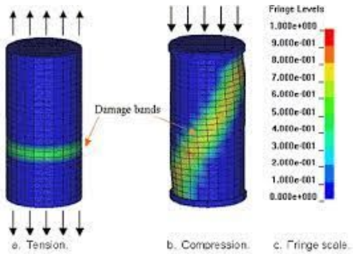
DARPA

NSF

Vector Processing Applications



Vector Databases

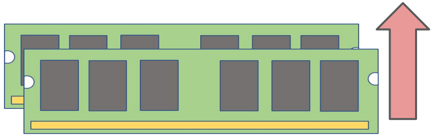


Simulations

CHATGPT

Large Language Models

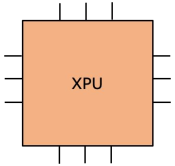
VPA Characteristics:



Memory-intensive

$$\vec{u} \cdot \vec{v}$$

Perform many vector operations

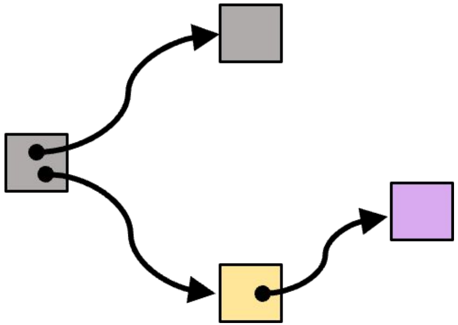


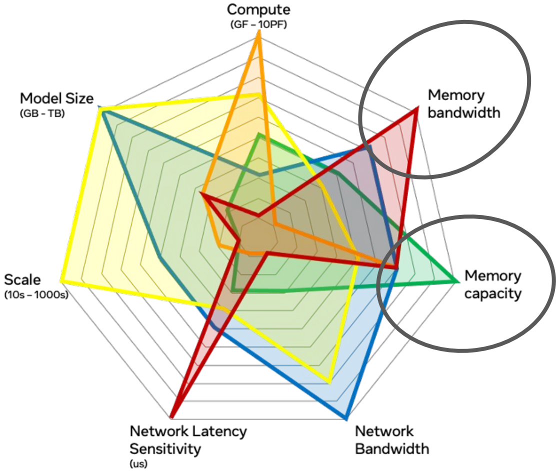
Use Accelerators

Image Credits: ONERA, U.S. DOT, Pinecone, OpenAI, Flaticon.com

Problem: Pressure on the Memory Hierarchy

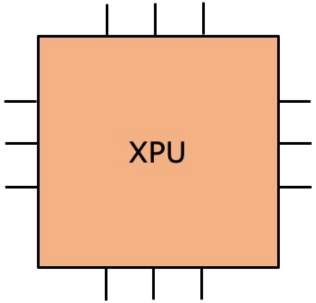
Application





Manoj Wadekar, Meta [FMS'24]

Accelerator

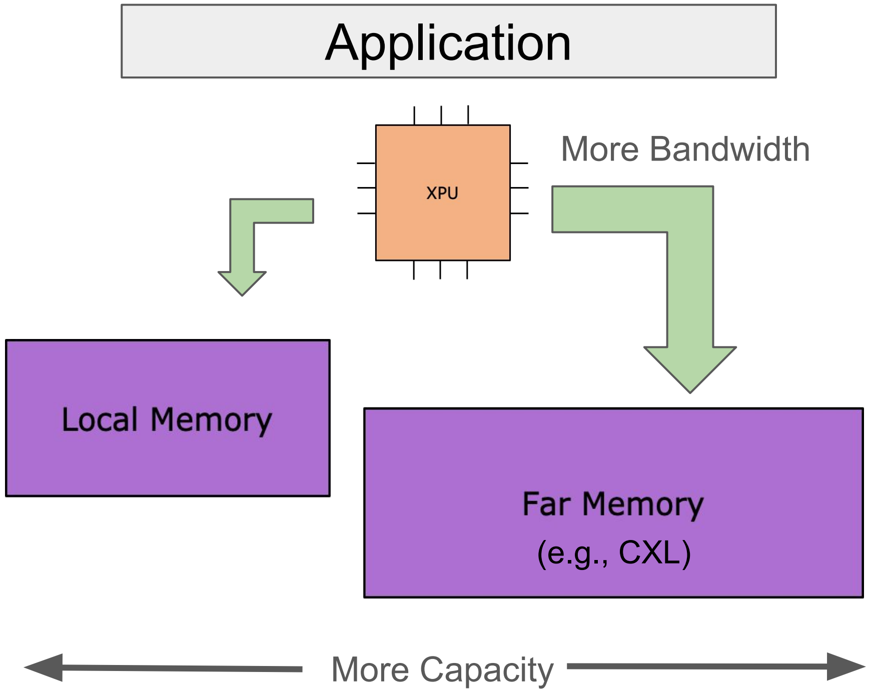


Needs large *memory capacity*

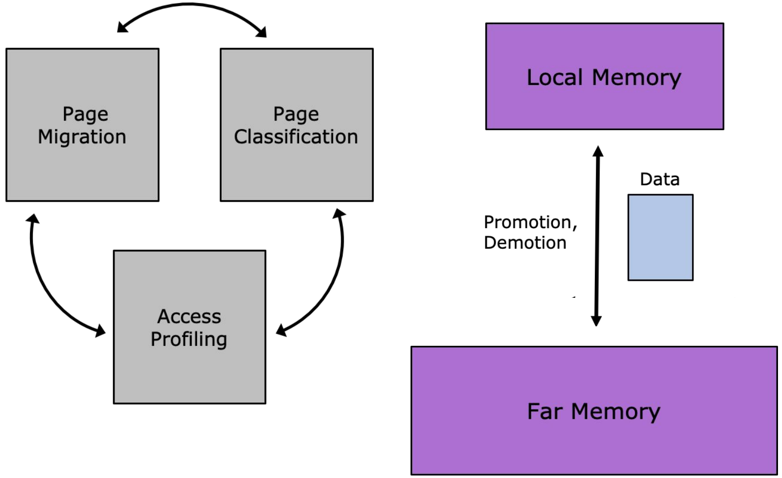
Wants high *memory bandwidth*

Issue: application working set >> accelerator memory capacity

Potential Solution: Memory Expansion

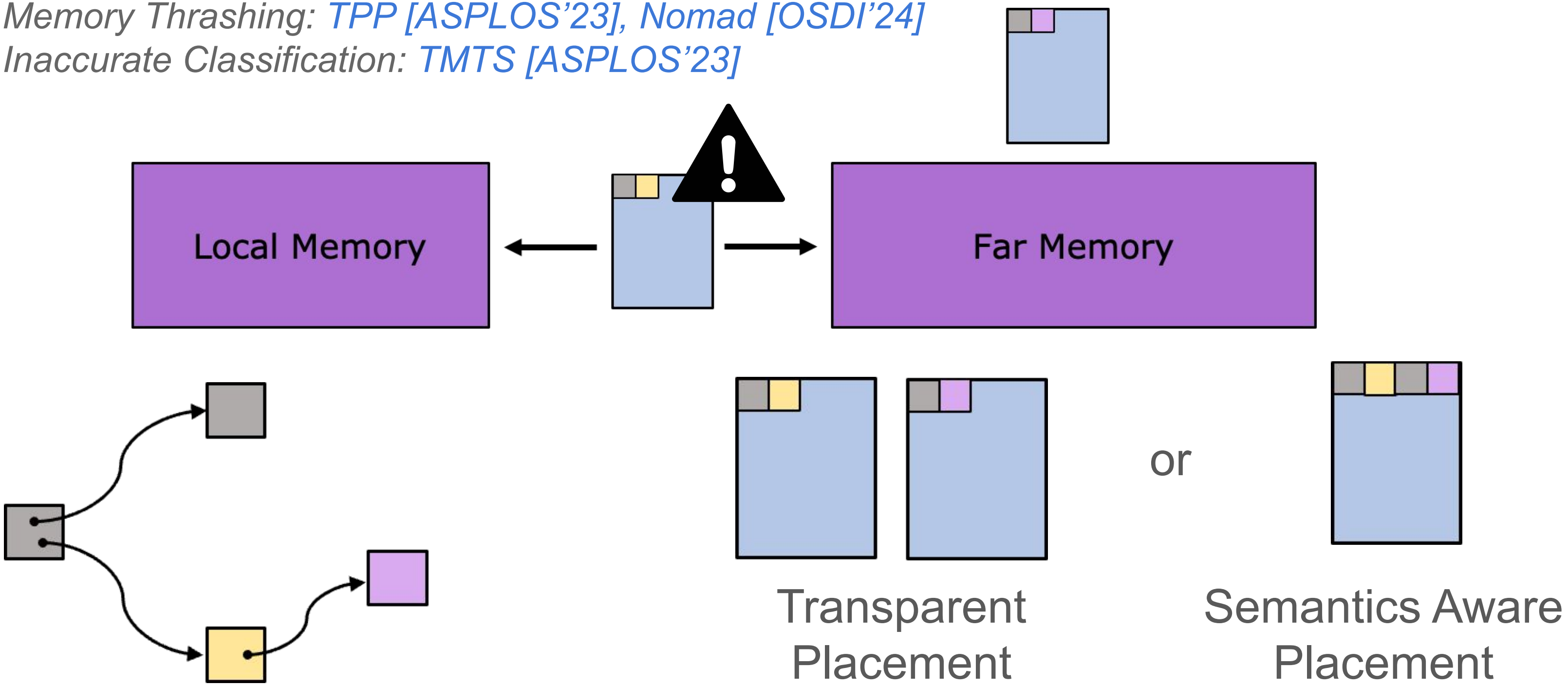


- **Manage additional memory with software-based memory tiering**
  - TPP [ASPLOS'23], Nimble [ASPLOS'19], HeMem [SOSP'21], TMTS [ASPLOS'23]



Limitations of Memory Tiering

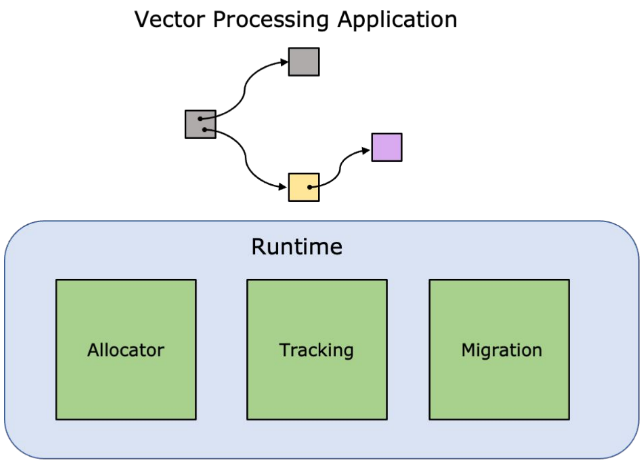
- *Data Amplification: AIFM [OSDI’20], DiLoS [Eurosys’23]*
- *Memory Thrashing: TPP [ASPLOS’23], Nomad [OSDI’24]*
- *Inaccurate Classification: TMTS [ASPLOS’23]*

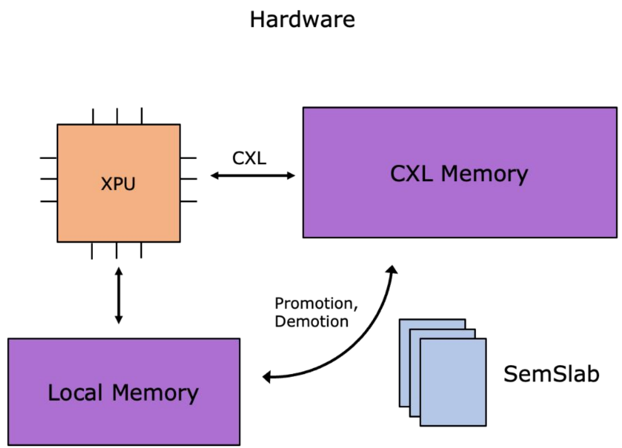


Research Question

What is the impact of a semantics-aware tiered memory system on resource utilization and application performance in heterogeneous systems?

Proposal: Semantic Data Placement





- Last time
  - Idea: Tiering for VPAs
- New Research
  - Refined ideas and system design
  - Work towards prototype
  - Identifying research challenges

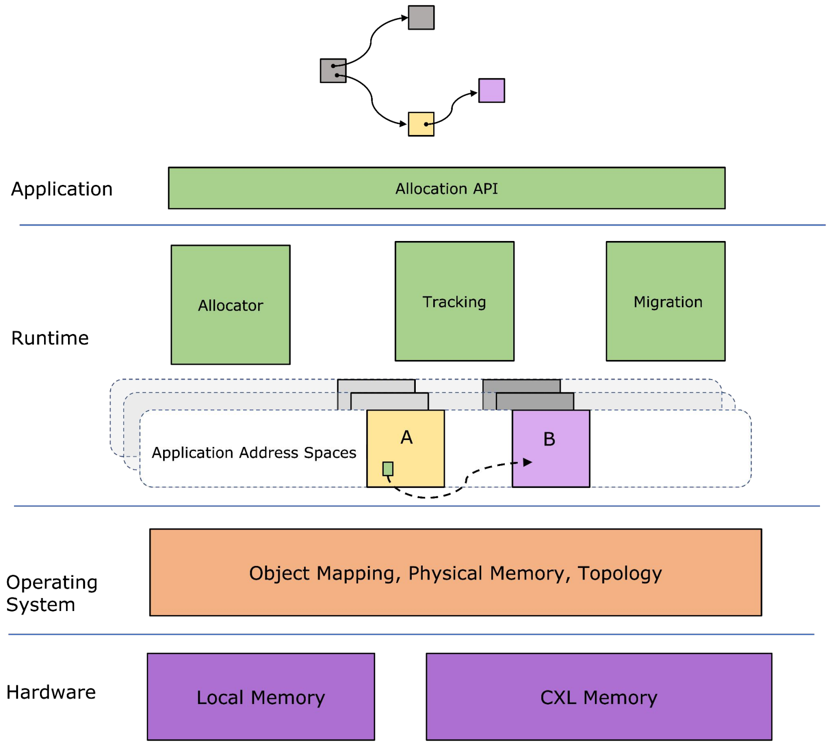
Research Challenges

- How can application semantics be used to reduce **memory bandwidth consumption** and improve memory **goodput**?
- What effect does semantic data placement have on application **performance** and **memory utilization**?
- How can we organize metadata for hotness tracking and other management functions to improve **scalability**?

System Architecture Overview

Our Proposed System:

Mnemonic Memory Tiers (M2T)



M2T’s Memory Allocation API: mnalloc

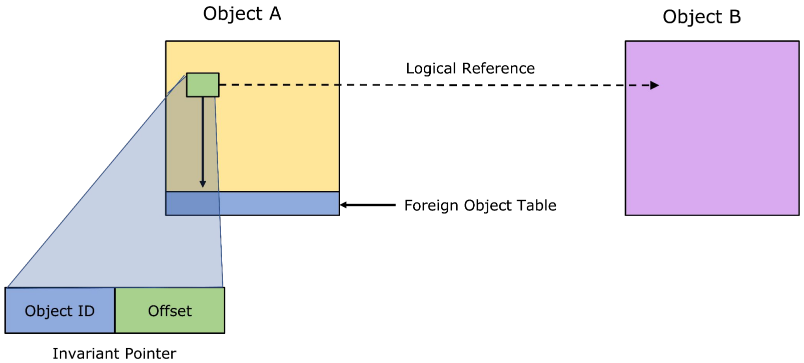
“Mnemonic Allocator”

```
mnalloc(size, PlacementDirective) → MemRef<T>
```

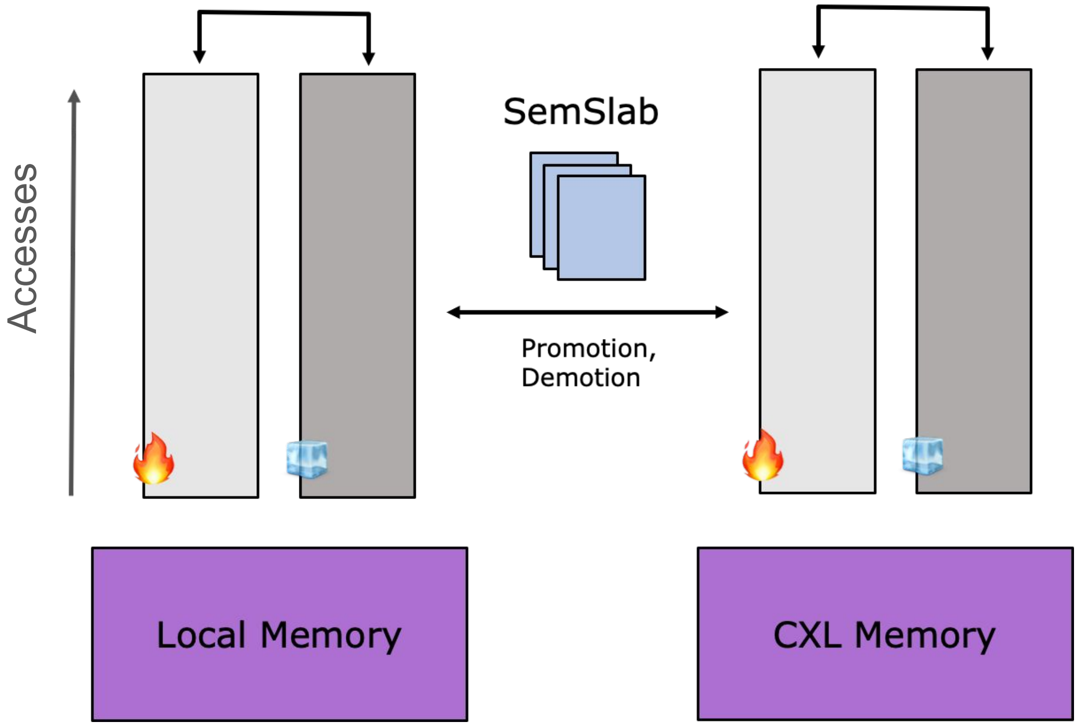
- **Encodes application semantics to the system**
  - Captures characteristics of a program from the perspective of its data
- **Similar approaches common in industry**
  - Google: TMTS [ASPLOS’23], Meta: TPP [ASPLOS’23]

M2T’s Memory Organization

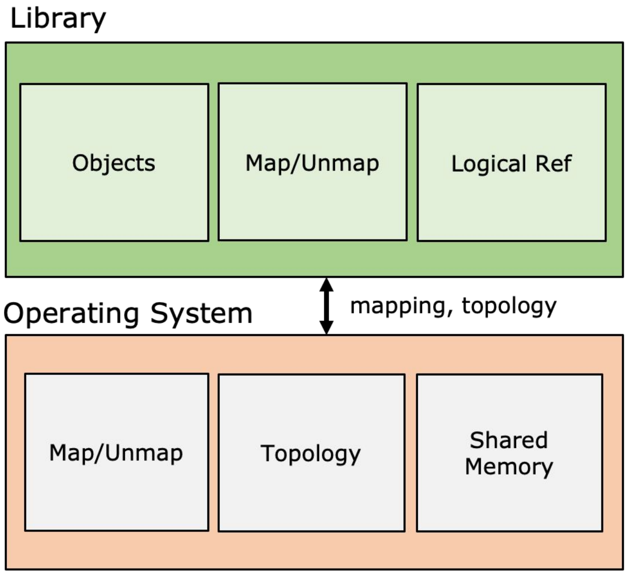
- **Adopt Twizzler’s Memory Model [ACT’20]**
  - Twizzler uses invariant pointers to memory objects as globally valid logical references
  - Software intercepts memory allocations and initial dereference only
- **Semantic Slabs (SemSlabs)**
  - Twizzler Memory Objects containing data allocated using **mnalloc**

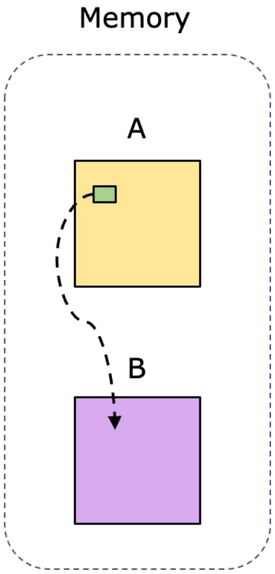


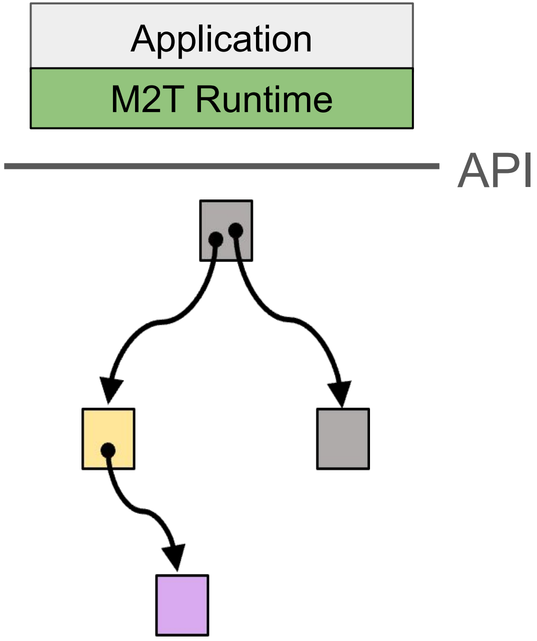
Tracking and Migration at SemSlab Granularity



Application Integration



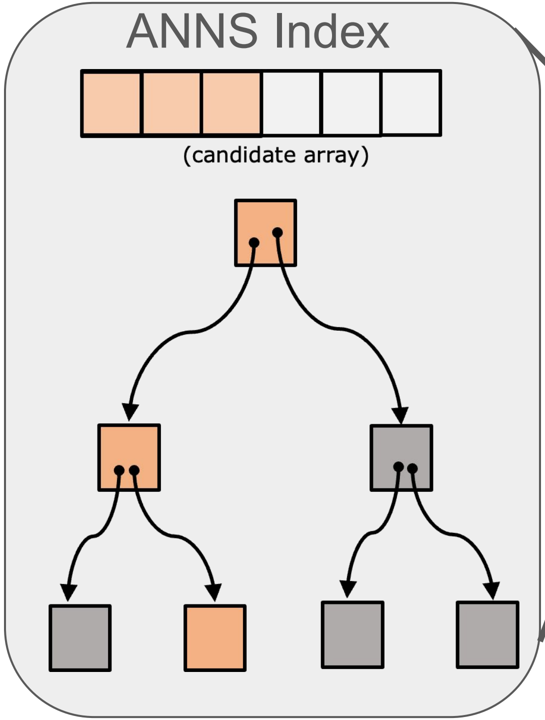




Applications link to the M2T runtime

Data Structure interface remains the same

Example: Scaling RAG Pipelines



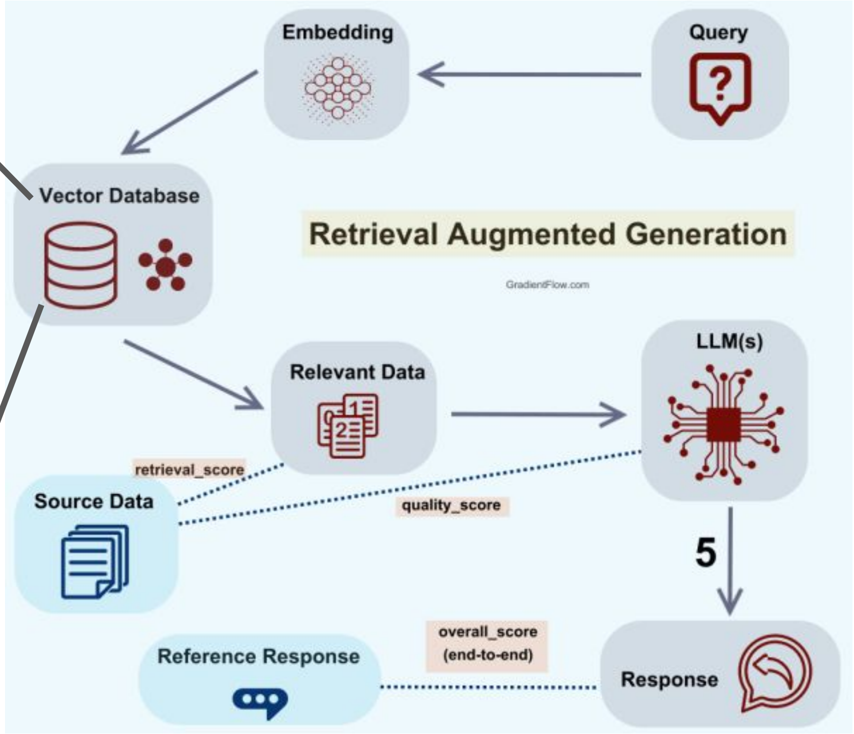
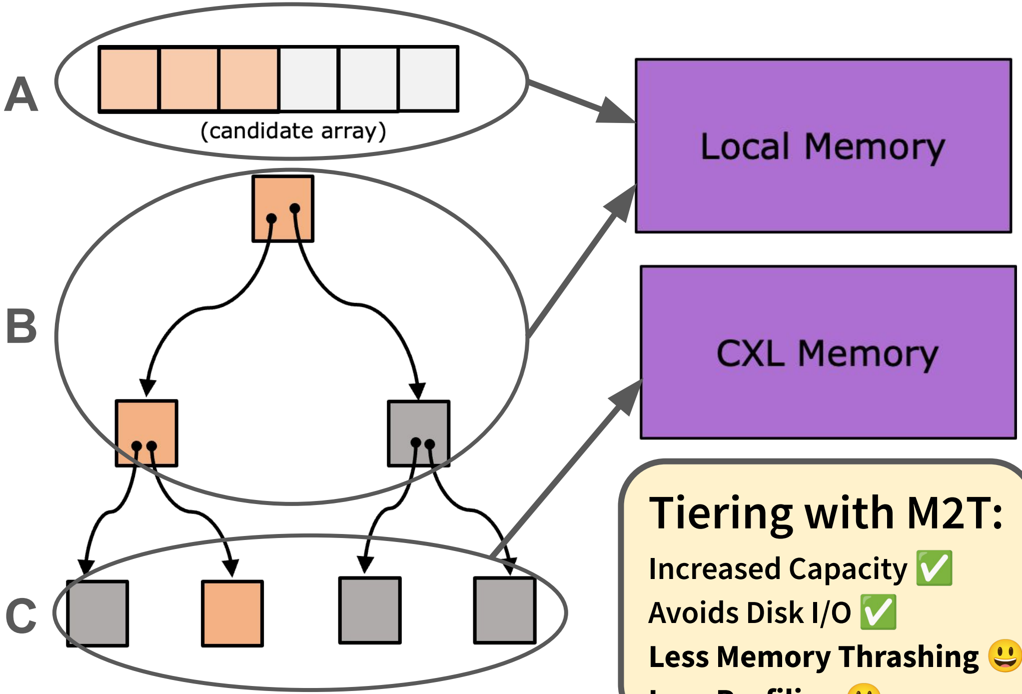


Image Credits: https://gradientflow.com/techniques-challenges-and-future-of-augmented-language-models/

Transparent Memory Tiering:

Increased Capacity ✅
Avoids Disk I/O ✅
Memory Thrashing 🙁
Profiling Overhead 🙁

Example: Scaling RAG Pipelines



Tiering with M2T:

Increased Capacity ✅
Avoids Disk I/O ✅
Less Memory Thrashing 😃
Less Profiling 😃

Application Code:

```
A  let carr = mnalloc(Placement::None);
   ...
   let root = mnalloc(Placement::Hot);
B  let root_data = mnalloc(Placement::Hot);
   let root_neighbor = mnalloc(Placement::Hot);
   let root_neighbor = mnalloc(Placement::Hot);
   ...
   let leaf =
        mnalloc(Placement::LatencyInsensitive);
C  let leaf_data =
        mnalloc(Placement::LatencyInsensitive);
```

What semantics can we express?

- **Developers use `mnalloc` to steer how M2T places data**
- **Memory objects placed based on associated semantics**
  - Temperature → Hot, Cold
  - Objects are “related”: NextTo(r)
  - Performance Insensitive → LatencyInsensitive, BwInsenstive
- `PlacementDirective` **could be determined automatically**
  - compiler techniques: Mira [SOSP’23], TrackFM [ASPLOS’24]
  - analyzing the call stack: TMC [SoCC’23], 2PP [PACT’15]

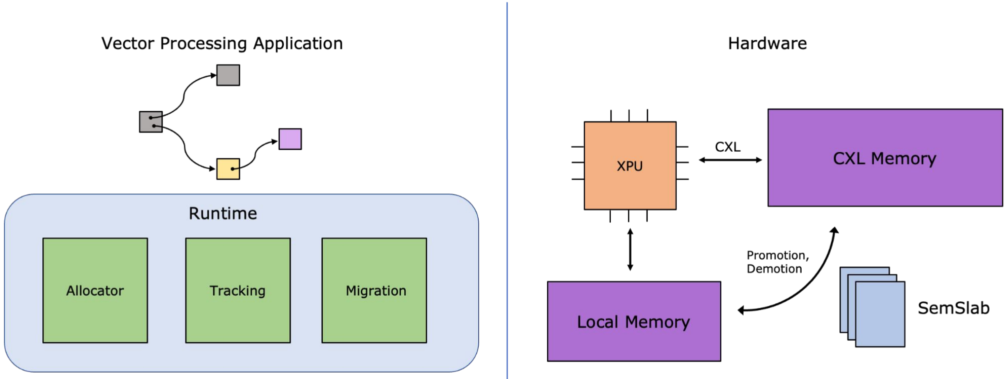
Goals for 2024-2025

CRSS

- **Implement a proof-of-concept M2T runtime**
- **Modify applications to use** `mnalloc`
  - Initial focus on vector indexes and Vector DB's
  - Applicable to HPC, simulation, graph processing, DBMS, and kv-stores
- **Evaluate the impact of semantic data placement**
  - On application performance?
  - On memory utilization?

Conclusion

- **Vector Processing Applications need robust system architectures**
    - to manage their growing memory footprint efficiently
    - CXL memory expansion provides a path forward
- **Semantic data placement potentially impacts**
    - application performance
    - system resource utilization



Thank You

Allen Aboytes

aaboytes@ucsc.edu

Questions?

Thank you to our sponsors!

CRSS

arm

cerabyte

intel

MARVELL

NUTANIX

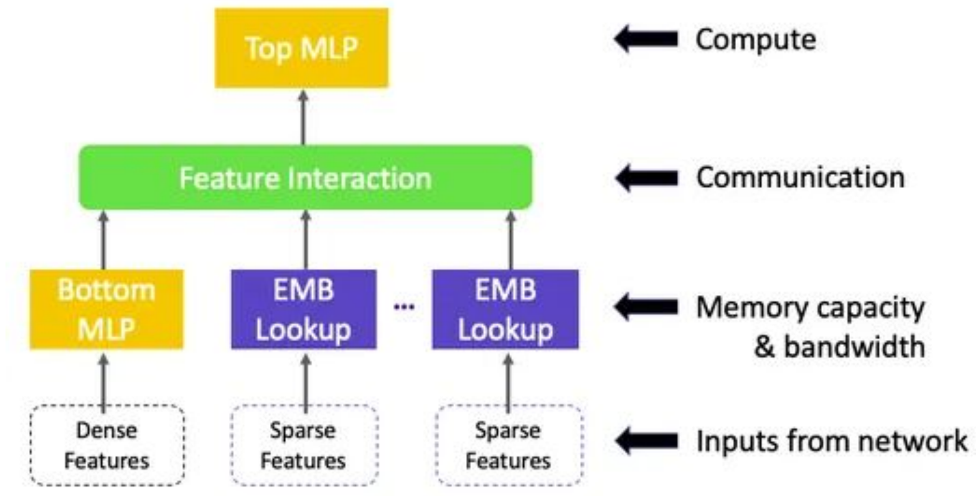
NSF

Backup Slides

Simple Example: Using M2T to build a Linked List

```
struct Node {
  next: InvPtr<Node>,
  data: u64
}

let mut a = mnalloc::<Node>(Placement::None);

*a = Node::new(42);

let mut b = mnalloc::<Node>(Placement::NextTo(a));

*b = Node::new(101);

a.next.assign(b);
```

Use Case: DLRM Inference Embedding Offload




```
let itemEmbedding =
mnalloc(size,LowLatency)
```

Local Memory

```
let userEmbedding =
mnalloc(size,BwInsenstive)
```

CXL Memory

Image Credits: Nishant Kumar, "Deep Learning Recommendation Models (DLRM): A Deep Dive". Medium.

Towards Low Overhead Tracking

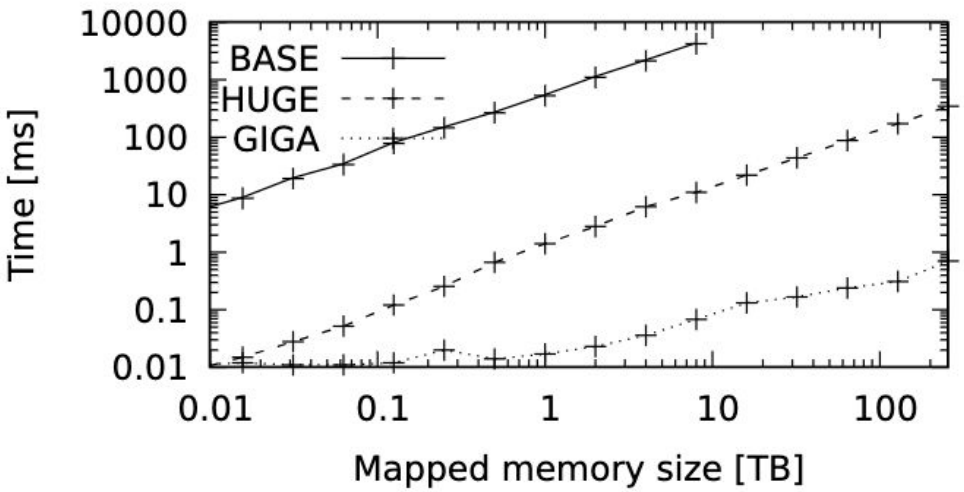


Figure 3: Page table scan time.

Amanda Raybuck, et al HeMem: Scalable Tiered Memory Management for Big Data Applications and Real NVM. SOSP 2021.