



Tier-Aware Dynamic Data Types: Fine-Grained Data Structure Partitioning Across Heterogeneous Memory Tiers

The rapid growth of data-intensive applications such as in-memory databases, key-value stores and graph analytics has made memory capacity a first-order constraint in modern servers. Compute Express Link (CXL) memory expanders offer a cost-effective way to extend this capacity. Built on top of PCIe, CXL allows a host processor to access external memory coherently, through ordinary load and store instructions, exposing it to the operating system as an additional NUMA node. The resulting system is a tiered memory hierarchy, where fast but limited local DRAM is complemented by larger CXL memory with significantly higher access latency. The performance of such a system depends critically on which data resides in each tier.

The majority of these applications organize their data in dynamic data types (DDTs), such as linked lists, trees, hash tables and graphs, whose elements are allocated and freed at runtime. Accesses to these structures are rarely uniform across their elements. In a linked list used as a queue or a most-recently-used cache, the first nodes are accessed far more often than the middle and tail nodes. In a B-tree, the root and upper levels are traversed by every lookup, while only a subset of paths towards the leaves is frequently used, depending on the key distribution of the workload. Hash tables exhibit similar skew across buckets. Consequently, the access frequency of a data structure varies at the granularity of individual nodes, not of the structure as a whole.

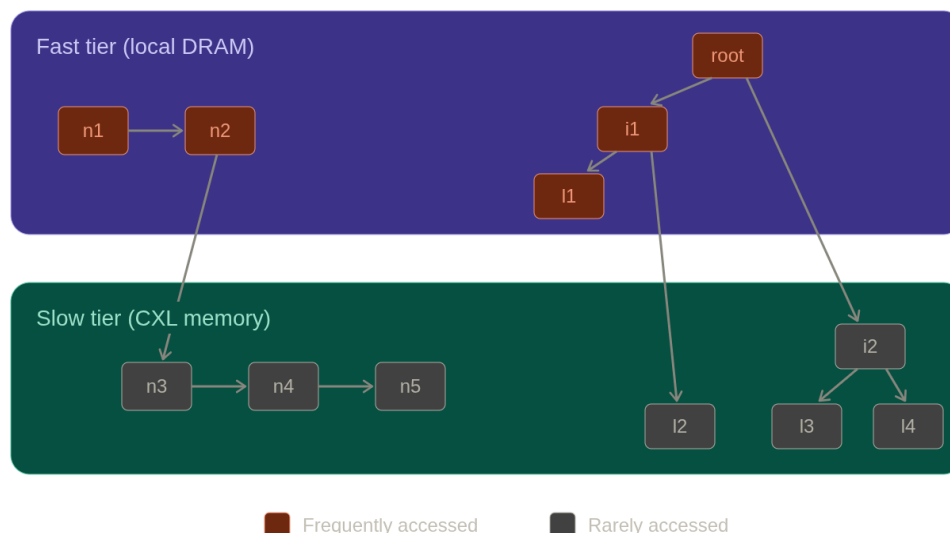


Fig. 1: A linked list and a B-tree partitioned across memory tiers. Frequently accessed nodes, such as the head of the list and the hot path of the tree, reside in local DRAM, while rarely accessed nodes are placed in CXL memory.

Existing tiered-memory policies do not operate at this granularity. Linux NUMA balancing in tiering mode and TPP classify memory as hot or cold at page granularity, using page faults or hardware sampling, and migrate entire pages between tiers. However, general-purpose allocators place nodes on pages according to allocation order, not access frequency. A single 4 KB page therefore typically contains a mix of hot and cold nodes from the same structure, or even from different structures. A page containing a single frequently accessed node is classified as hot and occupies scarce DRAM capacity together with all the cold nodes that share it, while hot nodes scattered across otherwise cold pages remain in the slow tier. This mismatch between object-level access behavior and page-level placement limits the effectiveness of both the fast tier's capacity and the migration mechanism itself.

Addressing this problem involves three main challenges:

1. **Object-level access profiling.** Placement decisions require knowledge of the access frequency of individual nodes, or of groups of nodes with similar behavior, such as tree levels or list segments. This information must be obtained with low overhead, either through lightweight instrumentation of the data structure operations or through hardware-assisted sampling mapped back to data structure elements. This thesis will investigate which profiling approach provides sufficient accuracy at acceptable runtime cost.
2. **Tier-aware allocation.** Nodes with different access frequencies must be physically separated, so that hot and cold nodes do not share the same pages. This requires allocators that maintain separate memory pools per tier, backed by local DRAM and CXL memory respectively, and that select the appropriate pool at allocation time based on the expected behavior of each node, such as its position in the structure.
3. **Runtime node migration.** Access patterns change over time, for instance as the key distribution of a workload shifts or as elements move within a list. Nodes must therefore be relocated between tiers during execution. Unlike page migration, relocating a node changes its address, requiring all references to it to be updated consistently. The migration mechanism must preserve correctness while keeping its overhead lower than the performance it recovers.

Since access to physical CXL hardware remains expensive and limited, the gem5 simulator will be used to model the system. gem5 is a modular, open-source full-system simulator widely used in computer architecture research, and prior work has already extended it with CXL memory models. Its modularity allows the configuration of a single-host tiered memory system with arbitrary latency and bandwidth parameters for each tier, making it possible to evaluate node-level placement across a range of CXL configurations, including configurations not yet available in commercial hardware.

This thesis addresses the problem of fine-grained placement of dynamic data structures in a single-host CXL-based tiered memory system. The goal is to design a library of tier-aware dynamic data types that splits each data structure across memory tiers according to the access frequency of its elements. Frequently accessed nodes, such as the head of a list or the hot paths of a tree, are kept in local DRAM, while rarely accessed nodes are placed in CXL memory. The library will combine per-node or per-region profiling, per-tier memory pools bound to the corresponding NUMA nodes through `mbind`, and runtime node migration, entirely in user space and without kernel modifications. The approach will be evaluated on a simulated CXL memory hierarchy in gem5 and compared against page-level baselines, including first-touch, interleaving and TPP.

The scope of this thesis includes:

- A characterization of the access behavior of common dynamic data types, such as linked lists, B-trees, hash tables and skip lists, under representative workloads, quantifying the skew in node access frequency and the degree to which hot and cold nodes are co-located on the same pages.
- The design and implementation of a tier-aware DDT library that provides per-tier allocation pools, low-overhead node-level profiling and a placement engine that assigns and migrates

nodes between local DRAM and CXL memory, preserving structural correctness during relocation.

- A set of structure-specific placement strategies, such as keeping list heads, upper tree levels or hot tree paths in the fast tier, alongside a generic frequency-based strategy applicable to arbitrary structures.
- An empirical evaluation on database and key-value store benchmarks, comparing the proposed approach against page-level placement policies in terms of application performance, fast-tier capacity utilization, profiling overhead and migration overhead.

PREREQUISITES:

Familiarity with:

- Computer Architecture
- Operating Systems
- Data structures and algorithms
- Coding languages (C/C++, Python)

RELATED MATERIAL:

- Performance, Energy and NVM Lifetime-Aware Data Structure Refinement and Placement for Heterogeneous Memory Systems (TACO '25)
<https://dl.acm.org/doi/full/10.1145/3736174>
- TPP: Transparent Page Placement for CXL-Enabled Tiered Memory (ASPLOS '23)
<https://dl.acm.org/doi/10.1145/3582016.3582063>
- Pond: CXL-Based Memory Pooling Systems for Cloud Platforms (ASPLOS '23)
<https://dl.acm.org/doi/10.1145/3575693.3578835>
- Demystifying CXL Memory with Genuine CXL-Ready Systems and Devices (MICRO '23)
<https://dl.acm.org/doi/10.1145/3613424.3614256>

CONTACT INFORMATION:

Spyros Chalkias: schalkias@microlab.ntua.gr

Manolis Xenos: mxenos@microlab.ntua.gr

Dr. Manolis Katsaragakis: mkatsaragakis@microlab.ntua.gr

Prof. Dimitrios Soudris: dsoudris@microlab.ntua.gr

Asst. Prof. Sotirios Xydis: sxydis@microlab.ntua.gr