PIPM: Partial and Incremental Page Migration for Multi-host CXL Disaggregated Shared Memory

Gangqi Huang† Heiner Litz† Yuanchao Xu†
University of California, Santa Cruz†

I. Introduction and Motivation

Emerging datacenter applications increasingly demand higher memory capacity, greater bandwidth, and lower costs. With the slowdown of DRAM technology scaling, architects have turned to Compute eXpress Link (CXL) for flexible, disaggregated shared memory (CXL-DSM) within compute racks. The latest CXL standards (CXL 3.x) further support coherent multi-host shared memory to enable dynamic compute and memory resource allocation and scaling, enhancing throughput and cost efficiency. Compared to traditional RDMA-based DSM systems [2], CXL-DSM provides much lower latency and simpler operating semantics, making it a promising architecture.

Despite its potential, CXL-based system performance is still limited by the high latency of remote CXL memory accesses. A common solution is page migration: page identified as frequently accessed are migrated from CXL memory to host-side local memory to reduce remote accesses.

However, existing page migration schemes designed for single-host CXL disaggregated memory are ineffective in multi-host CXL-DSM for two reasons: **(1) Local gain, global pain.** In single-host systems, migrating a hot page to local DRAM is strictly beneficial, assuming sufficient local memory capacity is available. In a multi-host CXL-DSM, however, moving a hot page from shared CXL memory to one host's local DRAM may harm overall performance, outweighing the local benefit. To preserve coherence and consistency, the migrated page needs to become non-cacheable for all other hosts. As a result, remote accesses incur extra hops, round-trips, and address remapping overheads, significantly increasing latency for other hosts accessing the page. **(2) Poor Migration Scalability.** The side-effects of page migration in multi-host CXL-DSM systems pose significant challenges for supporting efficient and timely migration. However, migration overheads grow significantly as page migration is no longer entirely local but instead requires coordination across hosts, including RPCs, per-host page-table updates and TLB shootdowns.

To address these challenges, we propose **Partial and Incremental Page Migration (PIPM)**[1] for multi-host CXL-DSM. **Partial Migration:** Instead of migrating entire pages into local memory or retaining them fully in CXL memory, PIPM selectively migrates only those cache blocks frequently accessed by a host into its local memory, while keeping less-frequently or remotely accessed blocks in CXL memory. This selective strategy differentiates local from inter-host access patterns at a fine granularity, effectively resolving the ”local-gain, global-pain” issue. Moreover, by maintaining two possible destinations for cache blocks, PIPM significantly reduces migration management overheads, such as page-table updates and TLB invalidations. **Incremental Migration:** Rather than explicitly migrating entire pages, which incurs substantial data-transfer overhead, PIPM leverages intrinsic memory accesses of programs to migrate cache blocks incrementally and selectively. Specifically, PIPM determines whether to incrementally migrate cache blocks from CXL memory into the requester host's local memory or back to CXL memory during cache coherence request handling. Consequently, incremental migration involves no additional data transfers beyond regular cache-fill and eviction operations. The partial migration policy identifies cache blocks and target hosts without initiating immediate data transfers, relying entirely on incremental migration for data movement. Together, these techniques enable PIPM to systematically address the previously identified challenges.

We develop architectural support for PIPM, including a majority-vote migration policy, a two-level hardware remapping table, and PIPM-coherency to effectively enable partial and incremental page migration. Our simulation shows that PIPM achieves an average speedup of 1.86× on multi-host CXL-DSM systems and surpasses six state-of-the-art methods.

[1]This abstract summarizes our accepted paper in ASPLOS 2026, a preview version can be found here.

II. PIPM Design

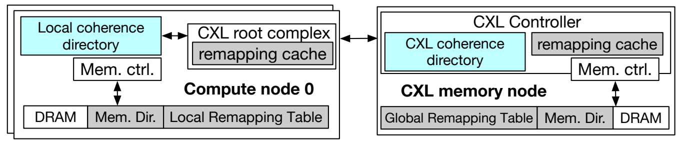

Fig. 1: PIPM design overview.

Overview. The architectural overview for Partial and Incremental Migration is shown in Figure 1. Our design introduces a per-host **Local Remapping Table** and a **Global Remapping Table** located on CXL memory to track pages undergoing partial migration. Specifically, the global remapping table records the migration destination host ID for each CXL-DSM page, while the local remapping table on each host stores the physical address mappings of CXL-DSM pages that migrate to the local memory of that host. We propose a **PIPM Majority-vote Migration Policy** that aggregates page-access information across multiple hosts, enabling globally optimized decisions regarding the necessity and placement of partially migrated pages. To ensure coherent access to partially migrated pages, we design the **PIPM Coherence** to incorporate partially migrated pages into the coherence domain, enabling incremental migration and cacheable access by other hosts.

PIPM Migration Policy. The intuition behind PIPM majority-vote migration policy is that partial migration is initiated only when the number of page accesses from a single host exceeds the combined accesses from all other hosts by a predefined threshold. Initiating partial migration solely involves updating the local and global remapping tables; thus, no page-table updates or TLB invalidations are required. The partial migration step only identifies the host to which cache blocks should be migrated, without triggering immediate data transfers.

The workflow of the PIPM migration policy is as follows: The global counter in a Global Remapping Table Entry is incremented by one when the access originates from the candidate host (also maintained in the entry) and decremented by one when from other hosts. When the global counter reaches zero, the next host to access the page updates the candidate host ID ❶. If the global counter reaches a predefined partial migration threshold ❷, partial migration of this page is initiated by creating an entry in the candidate host's local remapping table. The local PFN for this entry, allocated by the host's OS/hypervisor, identifies the location where partially migrated data from CXL memory is stored, and the entry's local counter is initialized to the migration threshold ❸.

The local counter, stored in each host's local remapping table, records local accesses to partially migrated pages since local accesses bypass the global counter maintained at the CXL memory node ❹. Also, inter-host accesses decrement the local counter for that page ❺. If the local counter of a partially migrated page reaches zero, partial migration for the page is revoked by migrating all cache blocks from local memory back to their original CXL memory location, removing the corresponding entry from the local remapping table, and resetting the current host ID in the corresponding global remapping table entry ❻.

Local Coherence States

Host	Dir. State	Mem. State	PIPM State
Default	M/S/I	0	M/S/I
Extra	ME/I	1	ME/I'

Extra State I' Description: Migrated to local memory; the latest version is not in local cache.
Extra State ME Description: Migrated to local memory, exclusively cached in local cache

Device Coherence State

Device	Dir. State	Mem. State	PIPM State
Default	M/S/I	0	M/S/I
Extra	I	1	I'

Extra State I' Description: Migrated to local memory

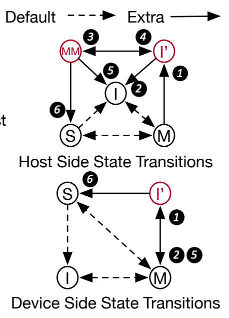

Fig. 2: PIPM coherence design.

PIPM Coherence and Incremental Migration. The objective of PIPM coherence design is to ensure that all local accesses to a partially migrated page first query the local memory for the latest data before forwarding requests to the CXL memory node, and to enable incremental migration based on the most recent accessor (i.e., migrate to local DRAM if the most recent accessor is the local host, migrate back to CXL-DSM upon an inter-host access). To accomplish this, we redesign the coherence protocol and utilize 1-bit per-line in-memory states in both local and CXL memory for partially migrated pages. As shown in Figure 2, the in-memory bits in CXL memory (or local DRAM) along with the coherence states in the CXL device-side directory entries (or local directory entries) define new coherence states, enabling cache line piggybacks during coherence request (e.g., Loads/Stores, WriteBacks) handling.

III. Evaluation

Methodology. We model the multi-host CXL-DSM architecture using a trace-based cycle-level simulator [1], [3]. We compare PIPM against: (1) **Native CXL-DSM**, that does not support data migration to hosts' local memory, and several state-of-the-art page migration schemes, including (2) **Nomad** [6], (3) **Memtis** [4] and (4) **Hemem** [5] on memory-intensive workloads selected from GAPBS, XSBench, SiloDB and PARSEC 3.0.

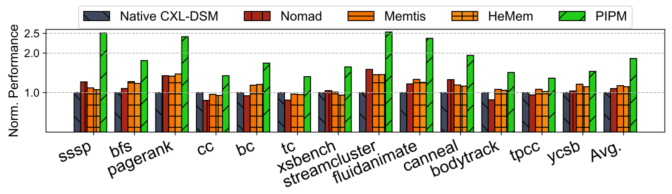

Fig. 3: End-to-end performance norm. to Native CXL-DSM.

End-to-end Performance. Figure 3 presents the overall performance of all evaluated schemes normalized to the *Native CXL-DSM* baseline. PIPM outperforms the other schemes across all workloads, achieving an average speedup of 1.86× and up to 2.54× relative to *Native CXL-DSM*, underscoring its substantial performance benefits.

IV. Conclusion

We propose **Partial and Incremental Page Migration (PIPM)** for multi-host CXL-DSM, which selectively migrates frequently accessed cache blocks into local memory and incrementally transfers data using intrinsic memory accesses. Evaluations show PIPM achieves up to 2.54× (1.86× average) speedup over existing methods, systematically overcoming key limitations of multi-host CXL-DSM.

References

[1] "Champsim," https://github.com/ChampSim/ChampSim, 2025.

[2] Q. Cai, W. Guo, H. Zhang, D. Agrawal, G. Chen, B. C. Ooi, K.-L. Tan, Y. M. Teo, and S. Wang, "Efficient distributed memory management with rdma and caching," Proceedings of the VLDB Endowment, vol. 11, no. 11, pp. 1604–1617, 2018.

[3] N. Gober, G. Chacon, L. Wang, P. V. Gratz, D. A. Jimenez, E. Teran, S. Pugsley, and J. Kim, "The championship simulator: Architectural simulation for education and competition," arXiv preprint arXiv:2210.14324, 2022.

[4] T. Lee, S. K. Monga, C. Min, and Y. I. Eom, "Memtis: Efficient memory tiering with dynamic page classification and page size determination," in Proceedings of the 29th Symposium on Operating Systems Principles, 2023, pp. 17–34.

[5] A. Raybuck, T. Stamler, W. Zhang, M. Erez, and S. Peter, "Hemem: Scalable tiered memory management for big data applications and real nvm," in Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles, 2021, pp. 392–407.

[6] L. Xiang, Z. Lin, W. Deng, H. Lu, J. Rao, Y. Yuan, and R. Wang, "Nomad:{Non-Exclusive} memory tiering via transactional page migration," in 18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24), 2024, pp. 19–35.