A Comparison of Modern Memory Management Schemes in HPC

Sina Karimi[1] , Kurt Keville[2] , Ajay Joshi[1] , Ayse Coskun[1]
[1] Boston University, [2] MIT

Motivation

As data-intensive applications gain more attention, the requirements for main memory increase at least linearly. Unfortunately, DRAM's scaling is at its limit. To solve this problem, researchers have suggested new types of memory, such as high-bandwidth memories (HBM), non-volatile memories (NVM), compute express link-based memory (CXL), and other emerging technologies.

CXL memories can help expand the capacity of the system in exchange for longer latency. Due to this drawback, data centers are employing CXL memory alongside DDR DRAM to take advantage of DDR performance while increasing their memory capacity.

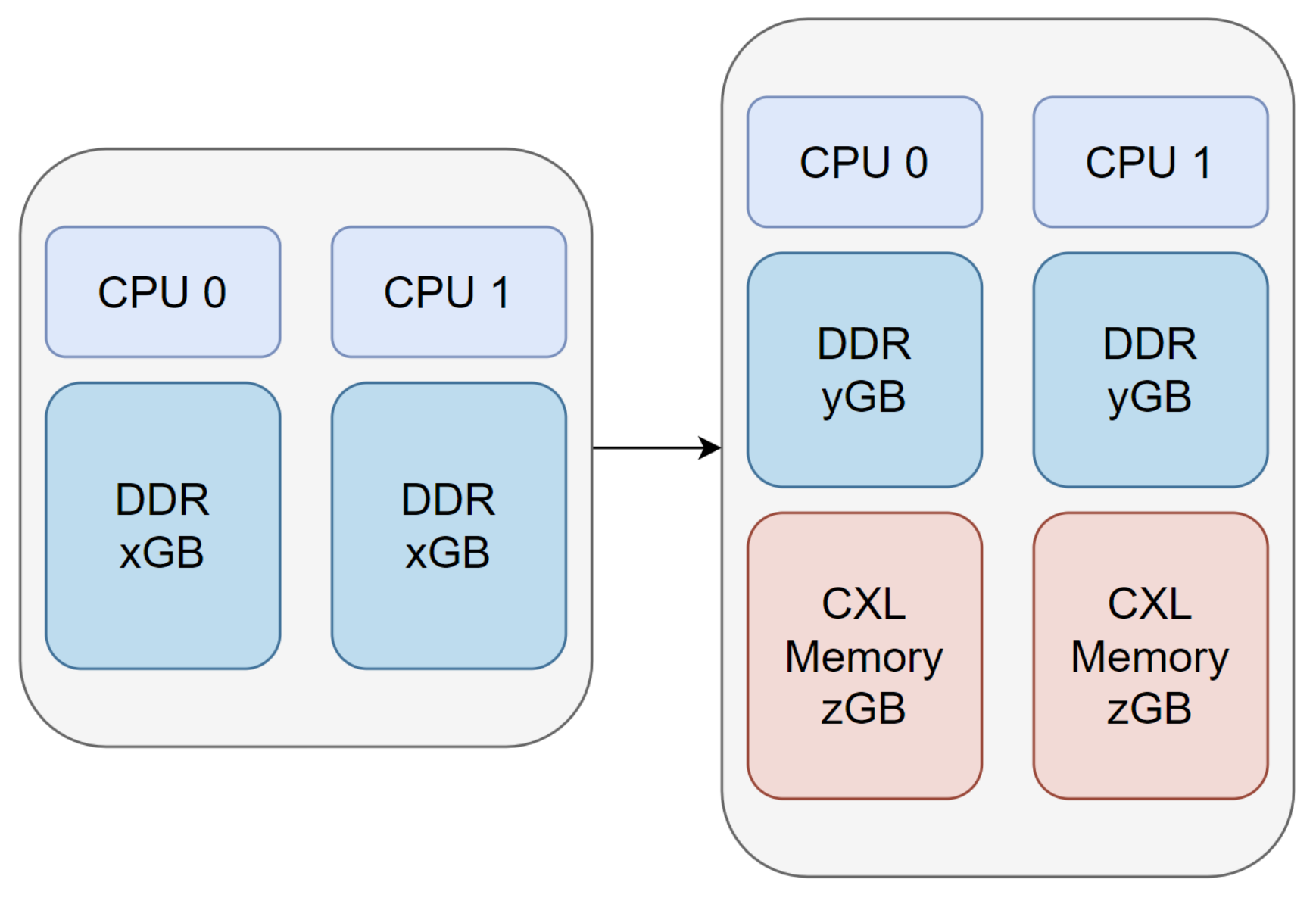

Figure 1:Memory Expansion using CXL memories

Current Solutions

- Recent works have proposed moving pages based on their access frequency.
- Hot pages (pages with frequent access) are promoted from CXL memory and placed in DRAM for better performance.
- Cold pages (pages with rare access) are demoted to CXL memory to save space for hot pages on DRAM.
- To implement this solution, the system requires a monitoring framework to differentiate between hot and cold pages.

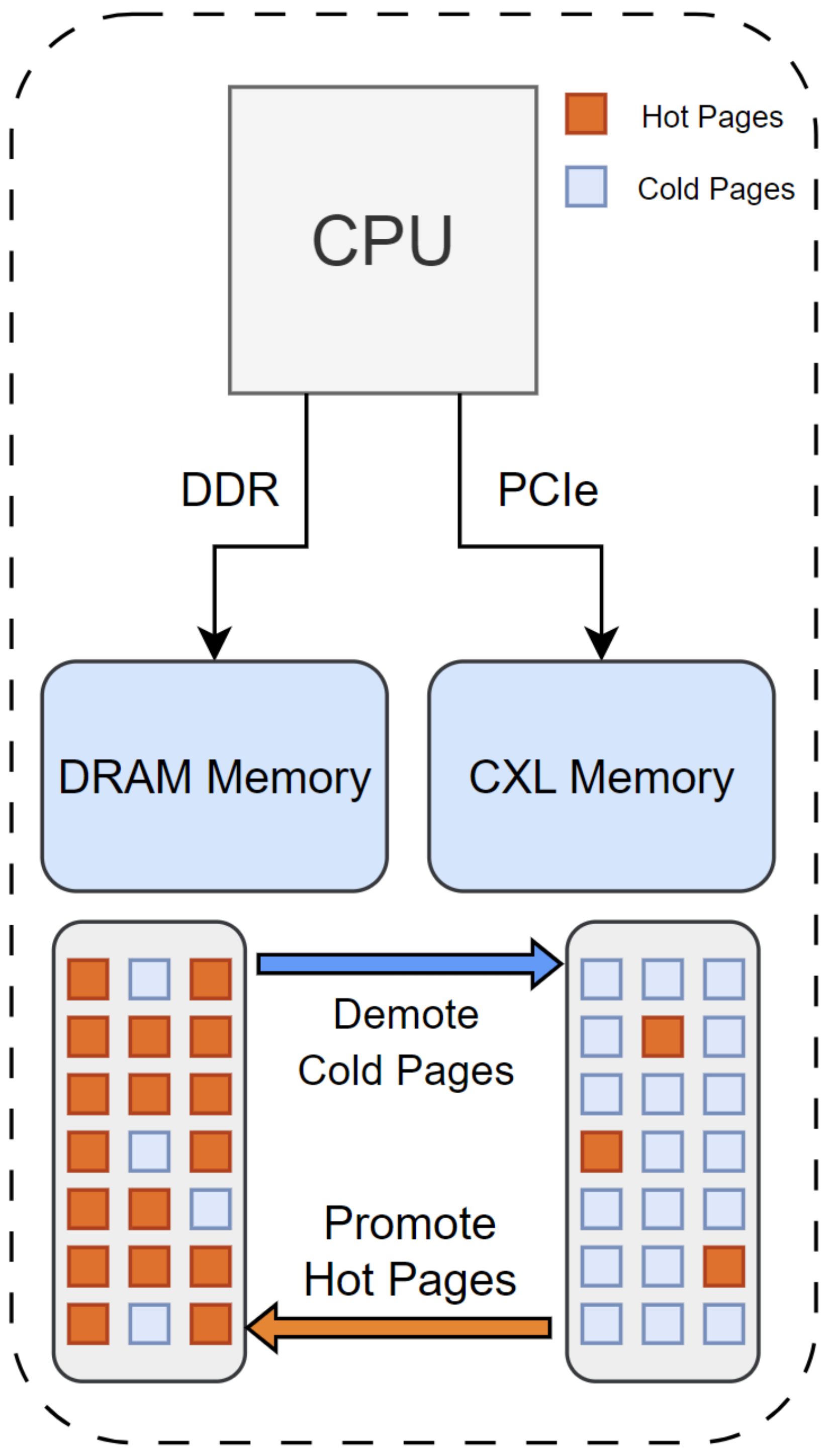

Figure 2: Migration in tiered memory systems

Method	Advantage	Drawback
PTE Scanning	Implemented in kernel	As the size of memory increases, the delay also increases High accuracy requires high overhead
Page Fault	Very precise Captures recency	Significant increase in page access latency
Hardware Sampling	Does not affect critical path for memory access	Miss very hot pages, High accuracy requires high overhead

Table 1: Comparison between different software monitoring methods

Typically, page access is hidden from software. However, different methods have been used in previous works to capture page access in the kernel. Each method has its advantages and drawbacks

Challenges

- Although many applications can benefit from migration policies, there are cases that migration can cause extra overhead for the system.
- In scenarios that size of hot data exceeds the capacity of the fast memory tier, migration policies cause memory thrashing and degrade the performance.
- Memory thrashing occurs when migration policy replaces hot pages in DRAM with other hot pages that exist in CXL memory, causing overhead without improving the performance.
- Memory thrashing can also occur in workloads with mostly uniform memory access.
- Because of memory thrashing caused by migration policies, no migration policy outperforms other proposed solutions in scenarios mentioned above.

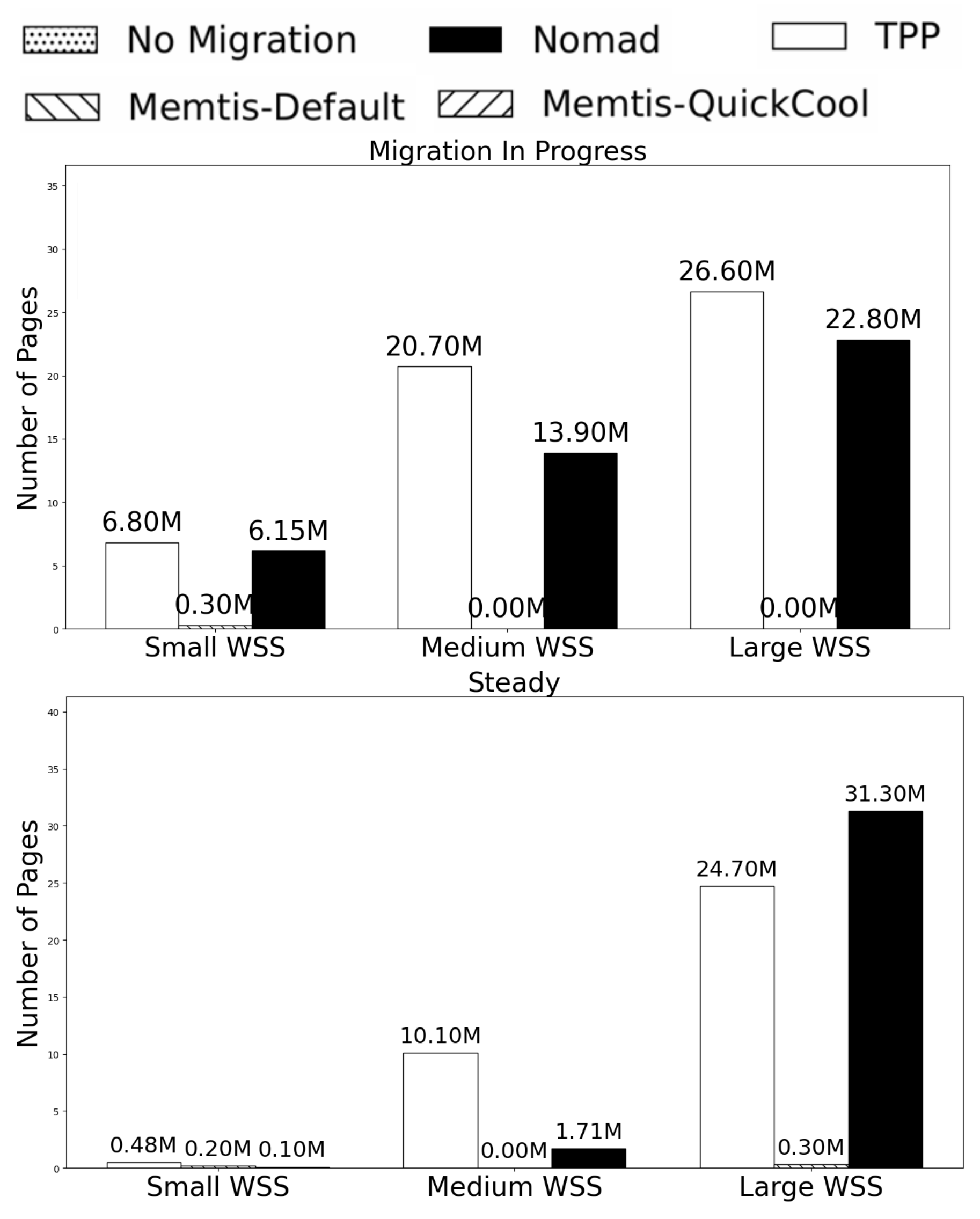

Figure 3: Number of pages being migrated in different phases of migration according to data from [1] for a benchmark with different working set sizes (WSS)

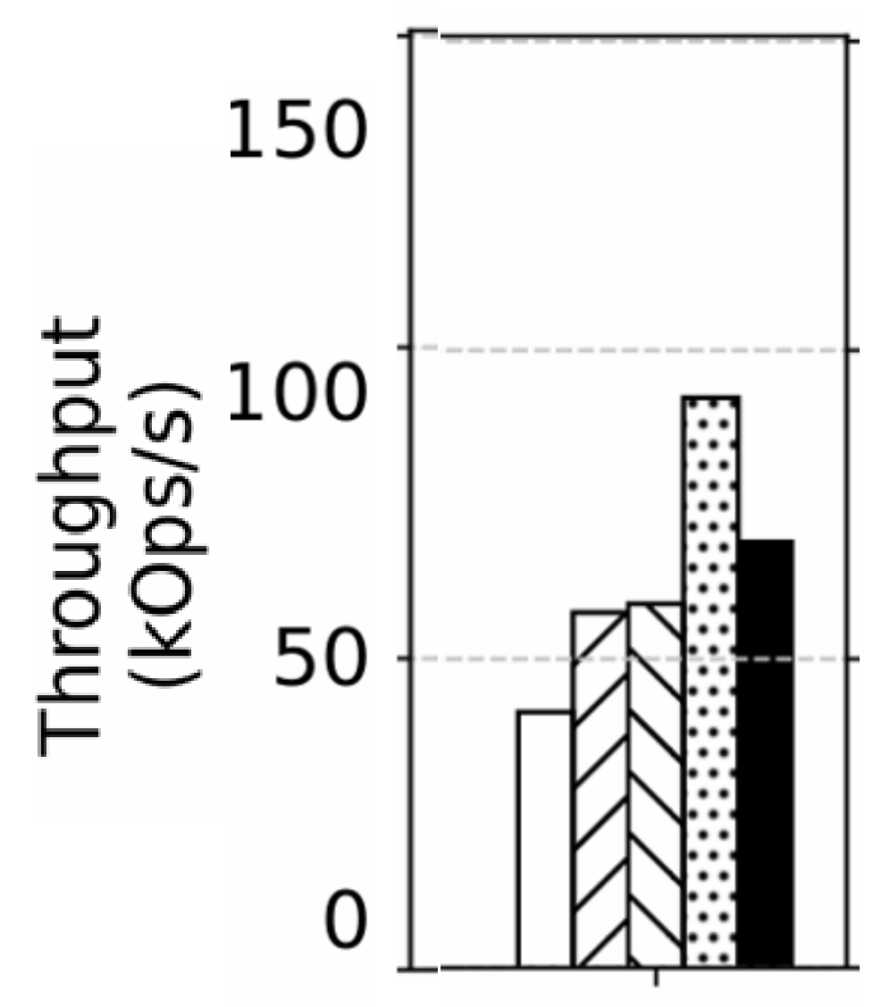

Figure 4: Performance of Redis with YCSB workload on tiered memory system with CXL memory reported in [1]

Future Directions

The difficulties in tiered memory systems can become even more complicated for systems employing non-volatile memories, as the cost of data migration can increase significantly compared to CXL memories.
One crucial question in tiered memory systems is whether migration policies provide improvement for the system.
For future research, we plan to estimate program memory access patterns to assess if migration could improve system performance.

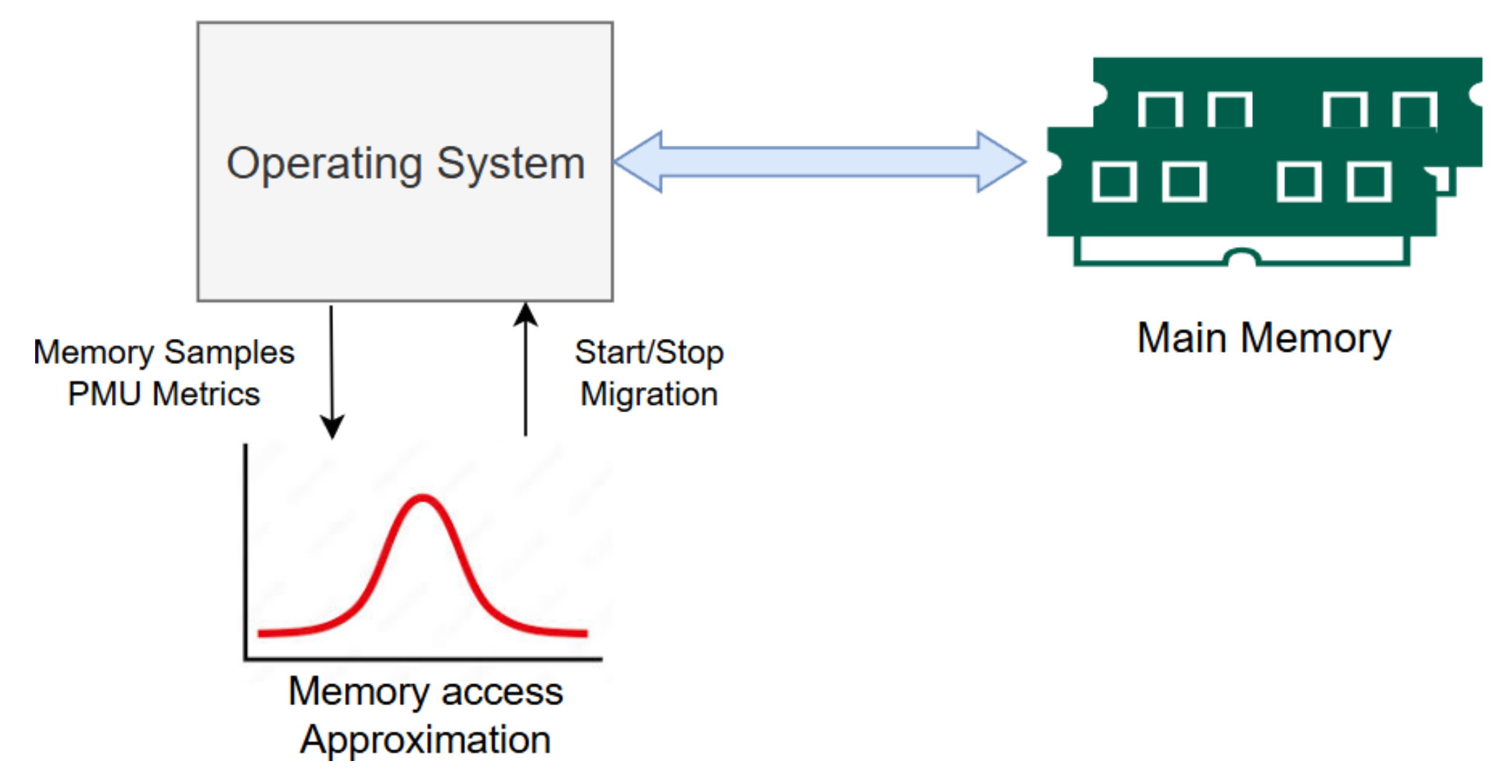

Figure 5: Our proposed solution to prevent overhead caused by migration policies

References

[1] Xiang, Lingfeng, et al. "Nomad:{Non-Exclusive} Memory Tiering via Transactional Page Migration." 18th USENIX Symposium on Operating Systems Design and Implementation 2024.
[2] Maruf, Hasan Al, et al. "Tpp: Transparent page placement for cxl-enabled tiered-memory." Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3. 2023.
[3] Lee, Taehyung, et al. "Memtis: Efficient memory tiering with dynamic page classification and page size determination." Proceedings of the 29th Symposium on Operating Systems Principles. 2023.
*This research was partially funded by the NSF CCF 2131127 grant.