NOMAD: Non-Exclusive Memory Tiering via Transactional Page Migration

Lingfeng Xiang, Zhen Lin, Weishu Deng, Hui Lu, Jia Rao
Yifan Yuan†, Ren Wang†

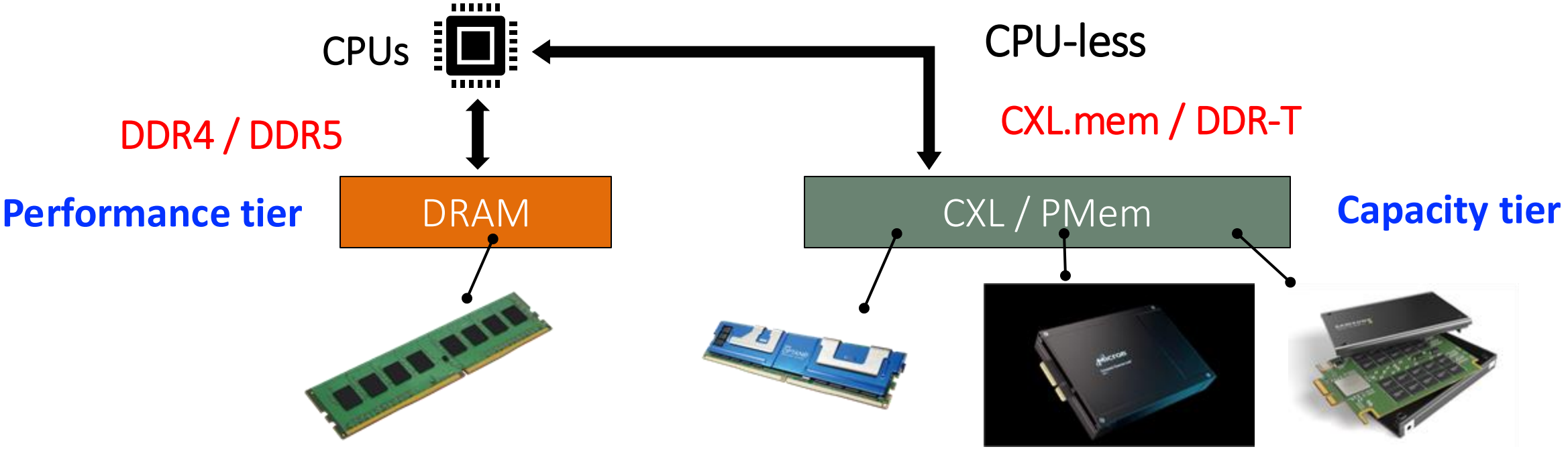
The University of Texas at Arlington, Intel Labs†

The new memory hierarchy

- Diverse memory devices with distinct characteristics
  - High bandwidth memory (HBM), Compute Express Link (CXL)-based memory, persistent memory, and storage-class memory
  - They differ in speed, size, cost, scalability, persistence, etc.
  - but all are byte-addressable via ordinary `load` and `store` inst.
- Memory hierarchy with tiered memory
  - Performance remains hierarchical but the gaps narrow
  - Memory management becomes non-hierarchical

The key is page management in the OS

Tiered memory management in the OS



- High $/GB
- High performance
- Low capacity

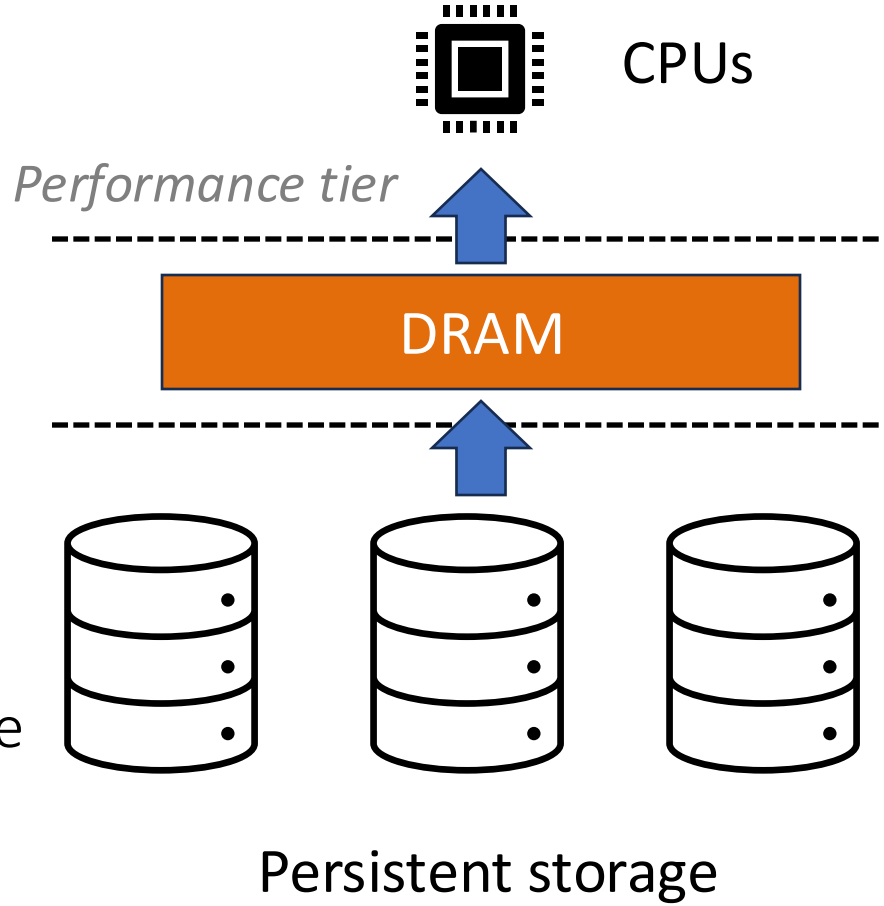
- Low $/GB
- Low performance
- High capacity

UNIVERSITY OF TEXAS ARLINGTON

Memory caching

Traditionally, OS employs *inclusive caching* to manage data between DRAM and disks

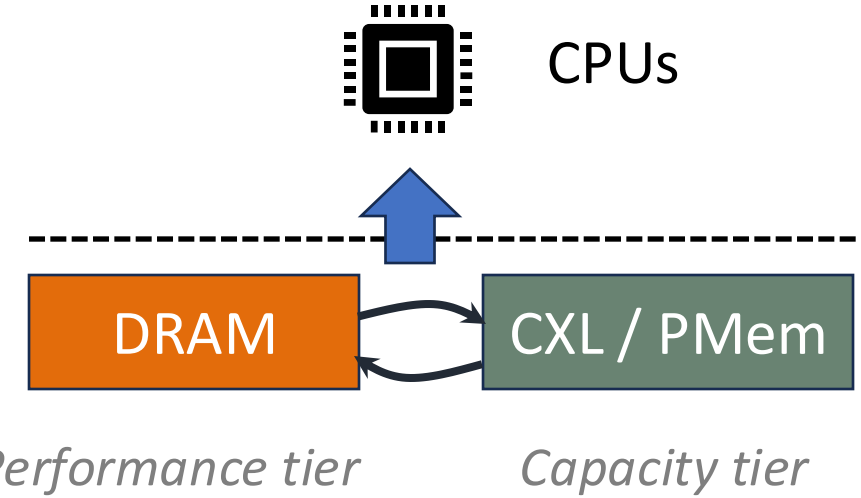
- 1-3x orders of magnitude performance and capacity gaps
- Data is replicated in and must be served from the performance tier
- data is not directly accessible by the CPU from the capacity tier



Performance tier hit rate is paramount

Memory tiering

Traditionally, OS employs ***inclusive caching*** to manage DRAM + disks



Now OS employs ***exclusive tiering***:

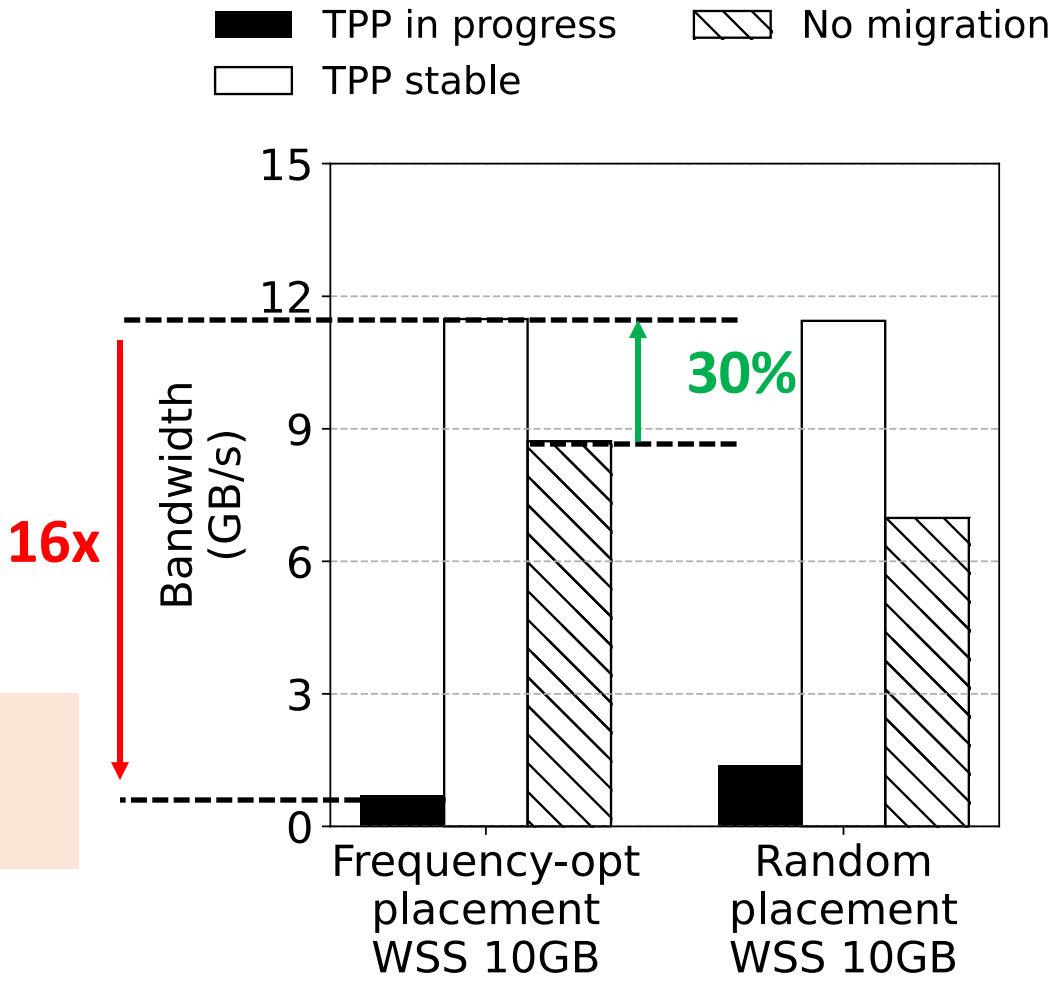
- 2-3x performance gap and within one order of magnitude capacity gap
- Data only resides in one of the tiers
- Data migration to keep hot data in the performance tier

Is exclusive tiering the optimal solution?

Synthetic workload

- WSS fits in the performance tier
- Initial page placement in the capacity tier
- Zipfian access pattern
- Evaluate TPP [ASPLOS'23], the default tiering approach in Linux

TPP **improves** fast-tier hit rate, but page migration can be prohibitively **expensive**



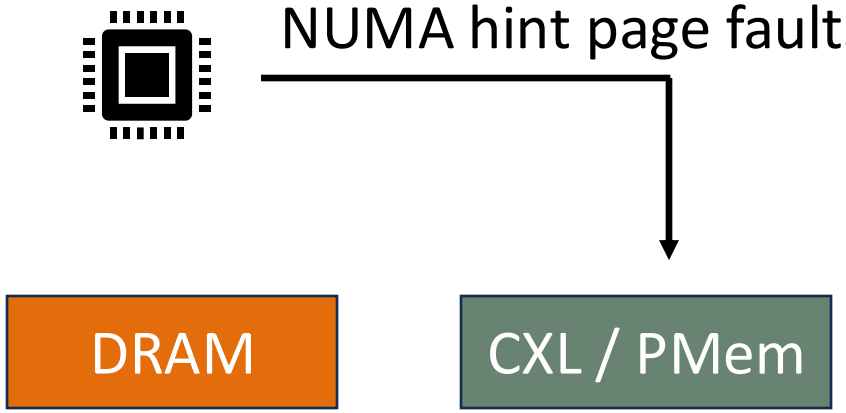
Key issues in tiered memory management

- How to **accurately** determine page temperature (hotness)?
- How to **efficiently** migrate pages between tiers?

Memory access tracking

Page fault-based tracking

- AutoNUMA, TPP [ASPLOS'23]
- Mark all pages in slow memory as `protected`
- Any access triggers a hint (minor) page fault, upon which a migration decision is made
- **Accurate**
- **Expensive**



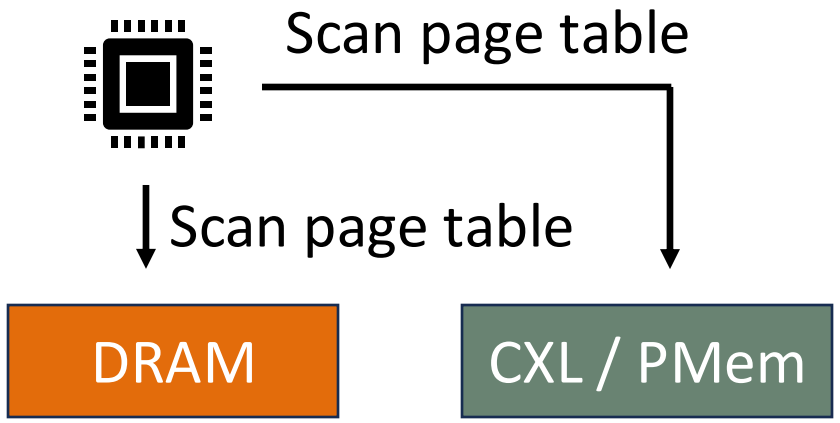
Memory access tracking

Page fault-based tracking

- AutoNUMA, TPP [ASPLOS'23]

Page table scanning

- Nimble [ASPLOS'19], Multiclock [HPCA'22], TMTS [ASPLOS'23]
- **Difficult overhead-accuracy tradeoff**



Memory access tracking

Page fault-based tracking

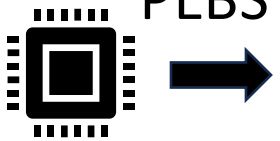
- AutoNUMA, TPP [ASPLOS'23]

Page table scanning

- Nimble [ASPLOS'19], Multiclock [HPCA'22], TMTS [ASPLOS'23]

Hardware sampling, e.g., Intel PEBS

- TMTS [ASPLOS'23], Memtis [SOSP'23]
- **Lightweight**
- **Coarse-grained, inaccurate**



Memory access address	Event
0xffff1234	load
0xffffabcd	TLB miss
…	…

DRAM

CXL / PMem

Page migration

Page fault-based tracking

- AutoNUMA, TPP [ASPLOS'23]

Synchronous migration, on the critical path of data access, expensive

Page table scanning

- Nimble [ASPLOS'19],

  Multiclock [HPCA'22], TMTS [ASPLOS'23]

Asynchronous migration handled by a separate kernel thread, off the critical path of data access

Hardware sampling, e.g., Intel PEBS

- TMTS [ASPLOS'23], Memtis [SOSP'23]

Can tiered memory management be both **accurate** and **lightweight**?

Our solution: NOMAD

Goals:

- Enable the CPU to **freely access** both fast and slow memory
- Move page migration **off the critical path** of users' data access

Approaches:

- Transactional page migration
- Non-exclusive tiering via page shadowing

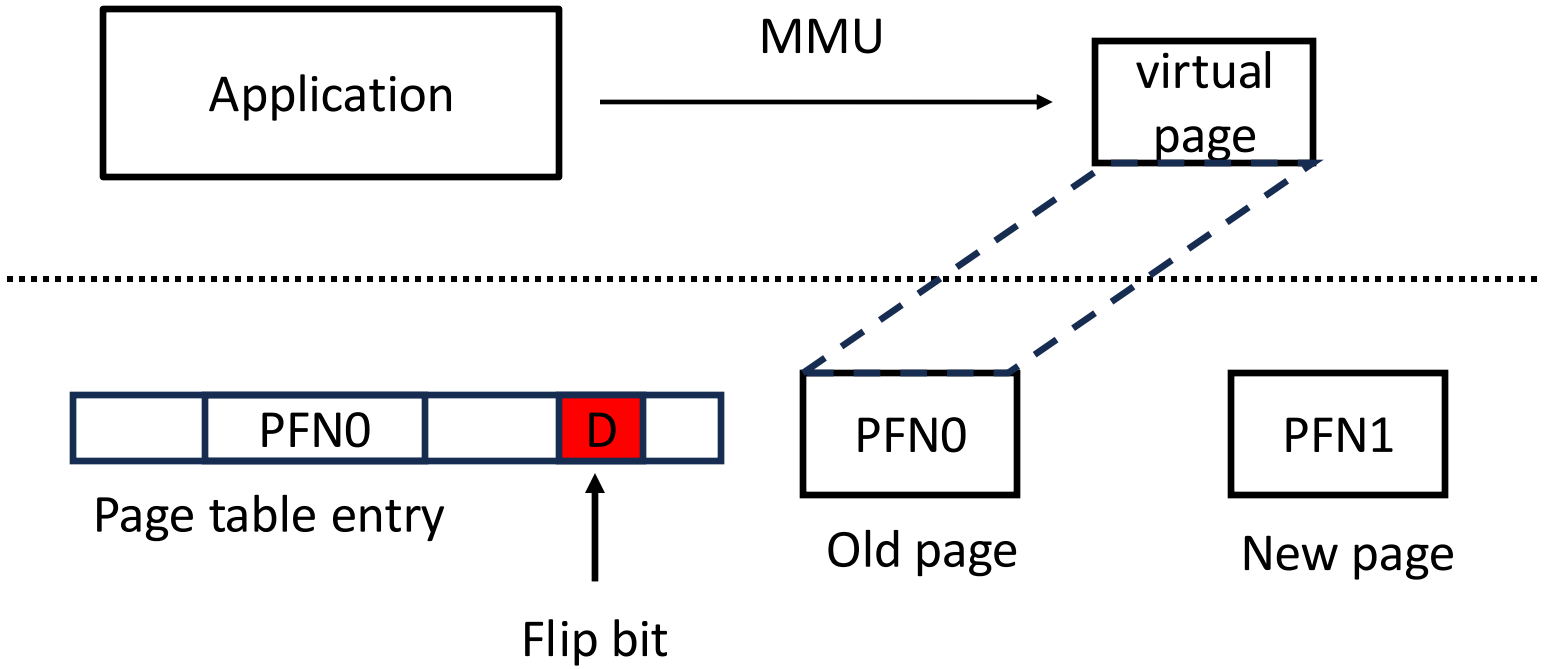
NOMAD is a page fault-based page management approach and is orthogonal to the existing work on memory access tracking

UNIVERSITY OF TEXAS ARLINGTON

Transactional page migration

Key idea:

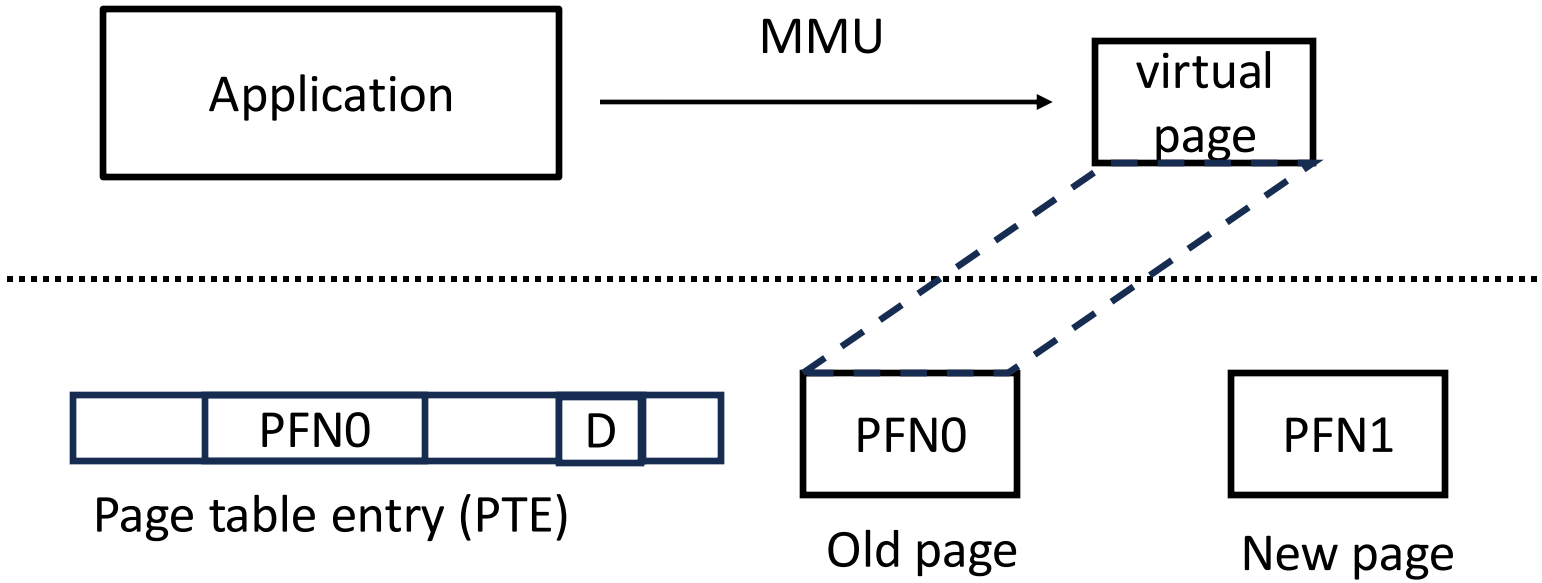
- Keep pages accessible during migration
- Invalidate the migration if pages are dirtied



Transactional page migration (TPM)

Major steps:

1. Clear the dirty bit in PTE

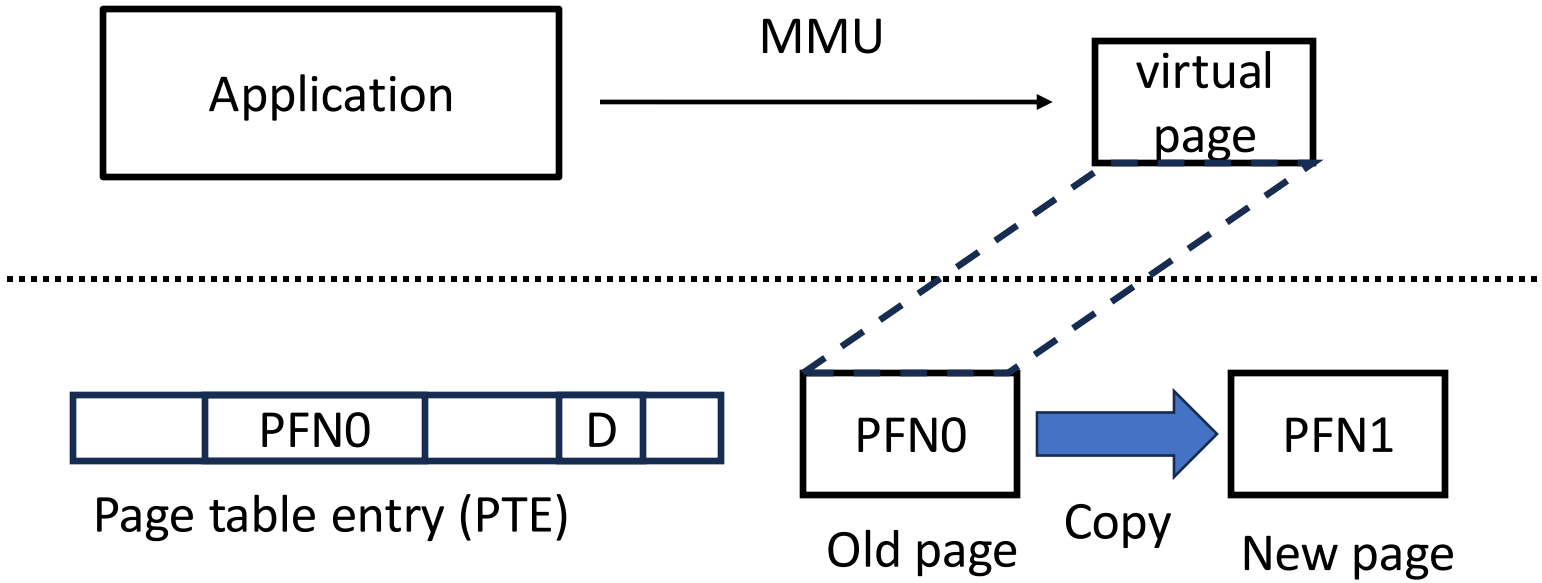


Issue a TLB shootdown. The page remains accessible

Transactional page migration

Major steps:

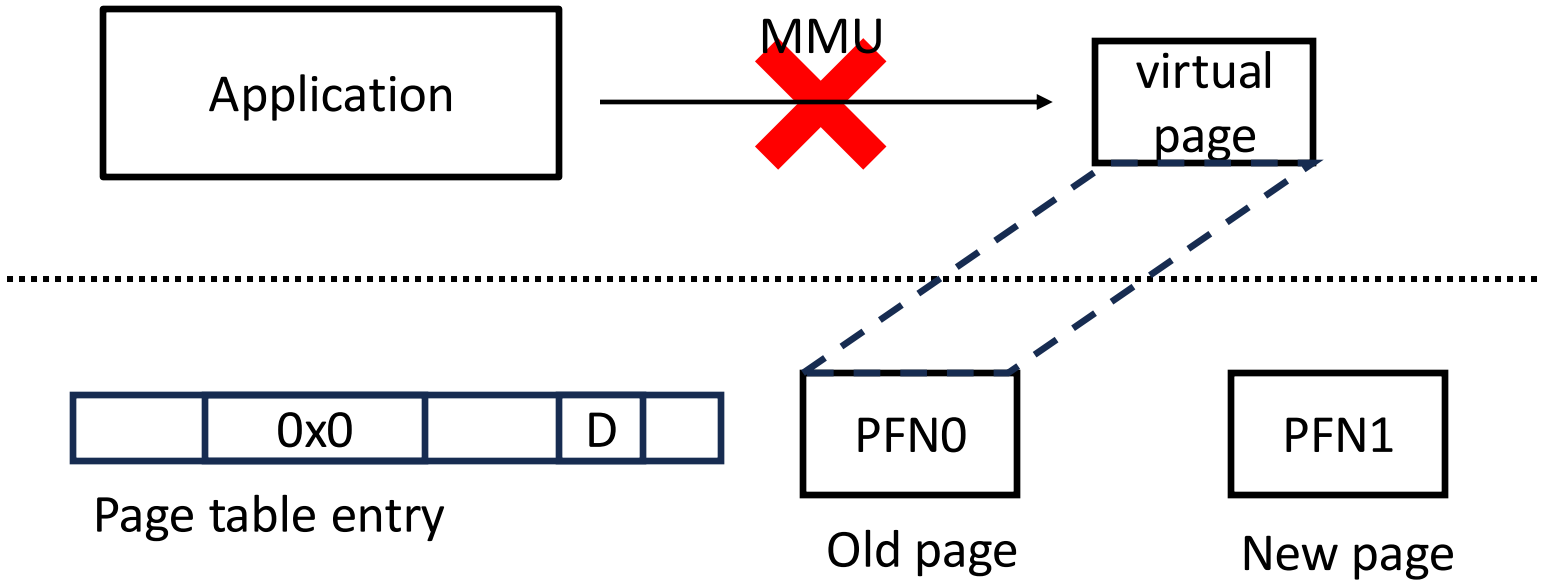
1. Clear the dirty bit in PTE
2. Copy page



Transactional page migration

Major steps:

1. Clear the dirty bit in PTE
2. Copy page
3. Unmap page



Issue a second TLB shootdown, after which the page becomes inaccessible until it is remapped

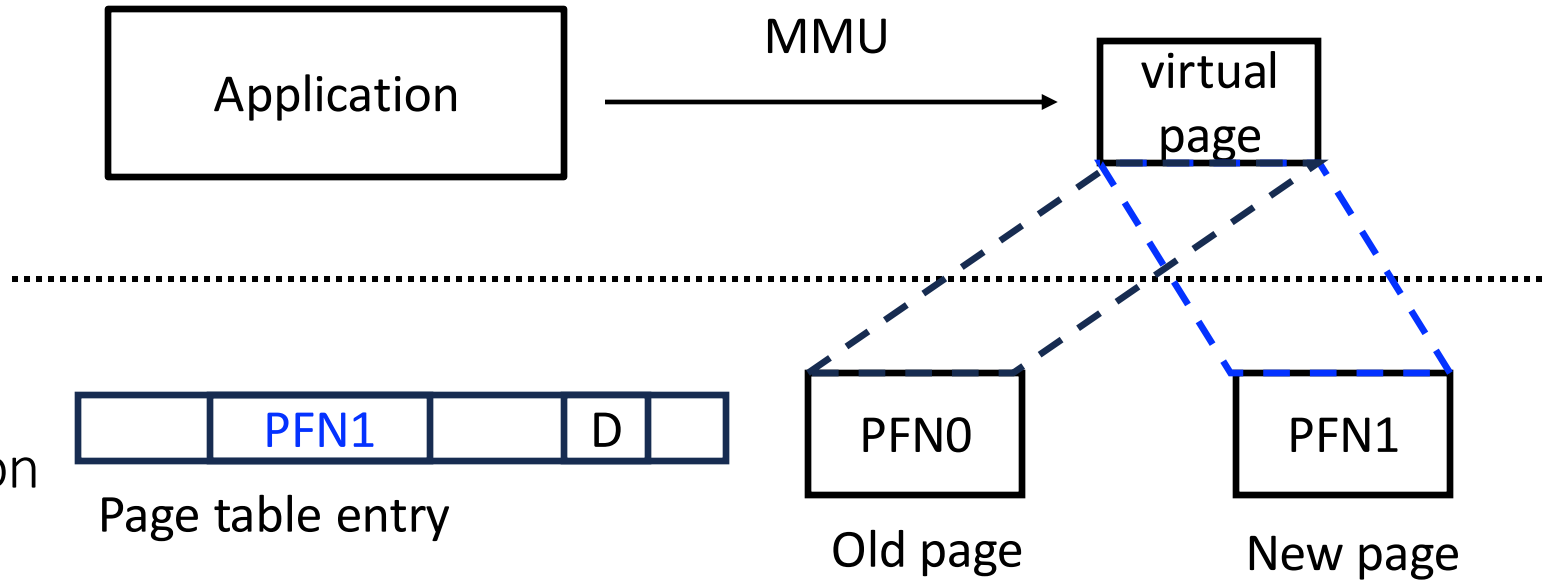
Transactional page migration

Major steps:

1. Clear the dirty bit in PTE
2. Copy page
3. Unmap page

*If **D** bit is clean*:

4. Map the page to destination



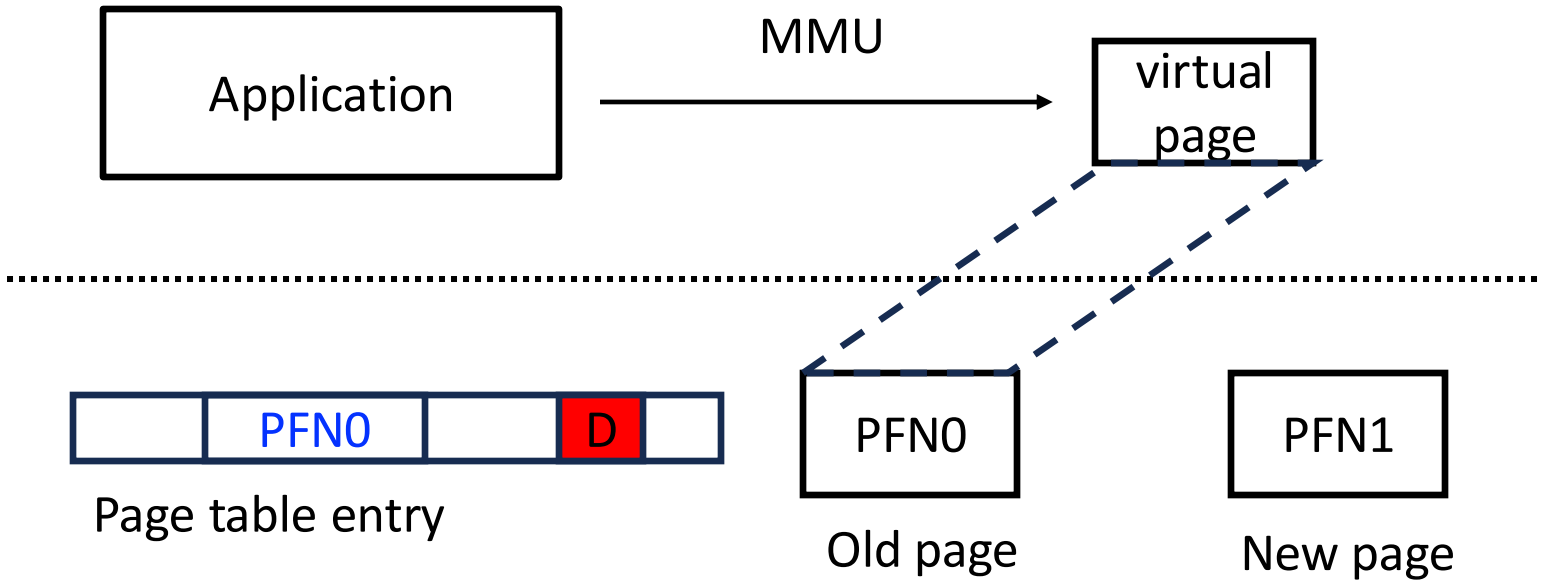
Transactional page migration

Major steps:

1. Clear the dirty bit in PTE
2. Copy page
3. Unmap page

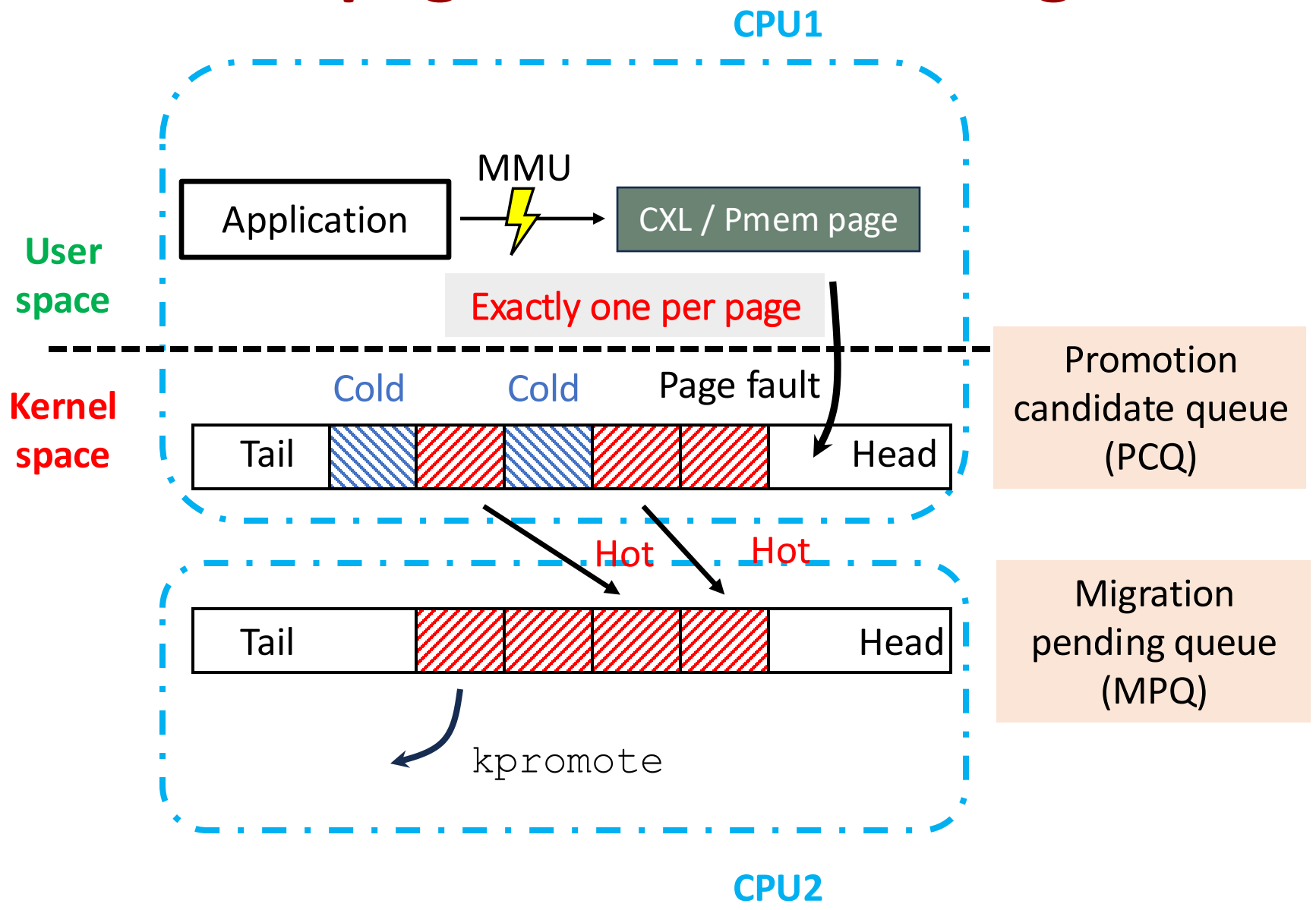
*If **D** bit is dirty*:

4. Remap the page to source and migration is aborted



Minimizing the number of page faults – 2Q design

- Queue both hot/cold pages in PCQ via exactly one fault/page
- Decide whether to promote a page to MPQ on every subsequent fault by scanning the PTEs of pages in PCQ
- Kernel thread `kpromote` asynchronously performs TPM on pages in MPQ

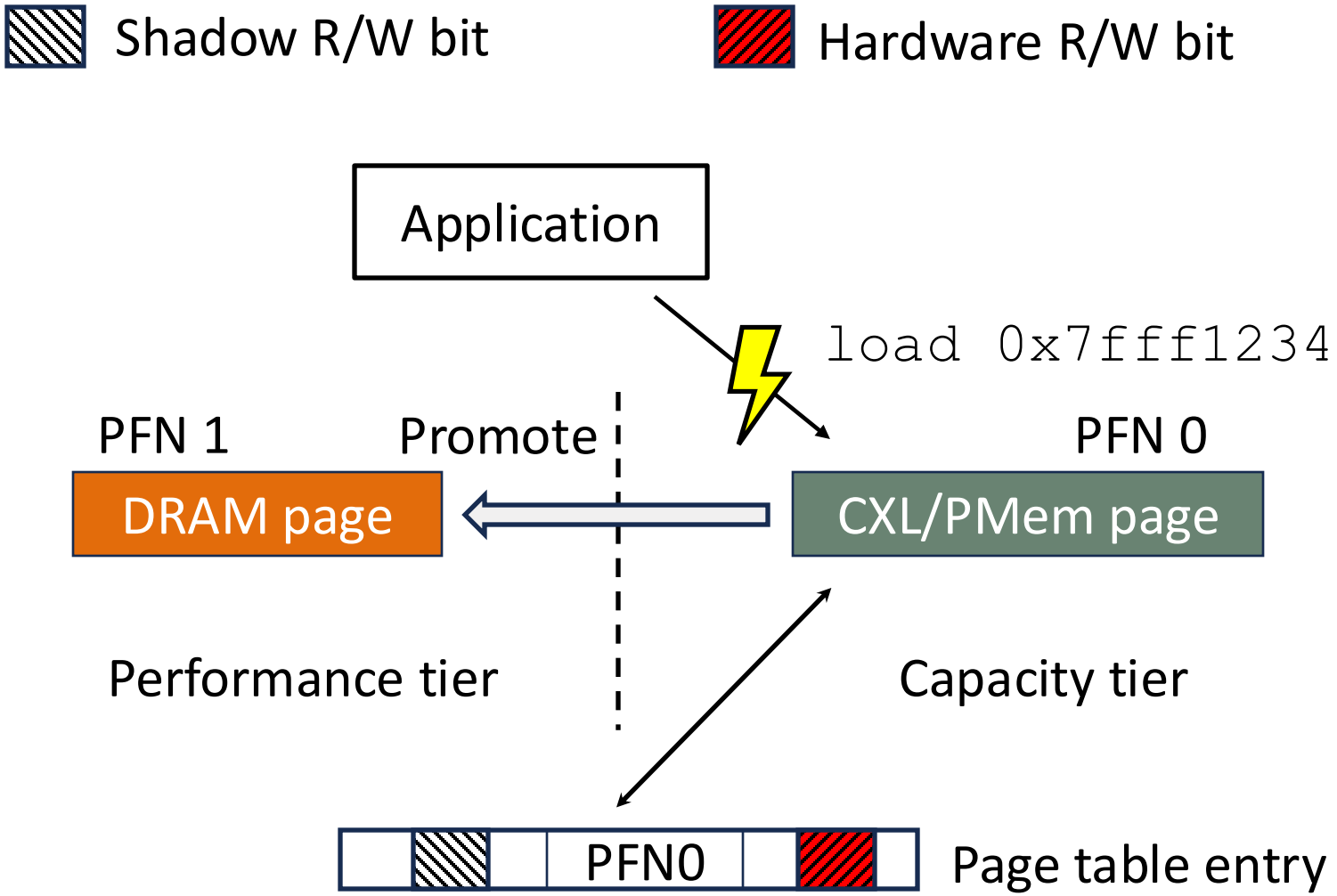


UNIVERSITY OF TEXAS ARLINGTON

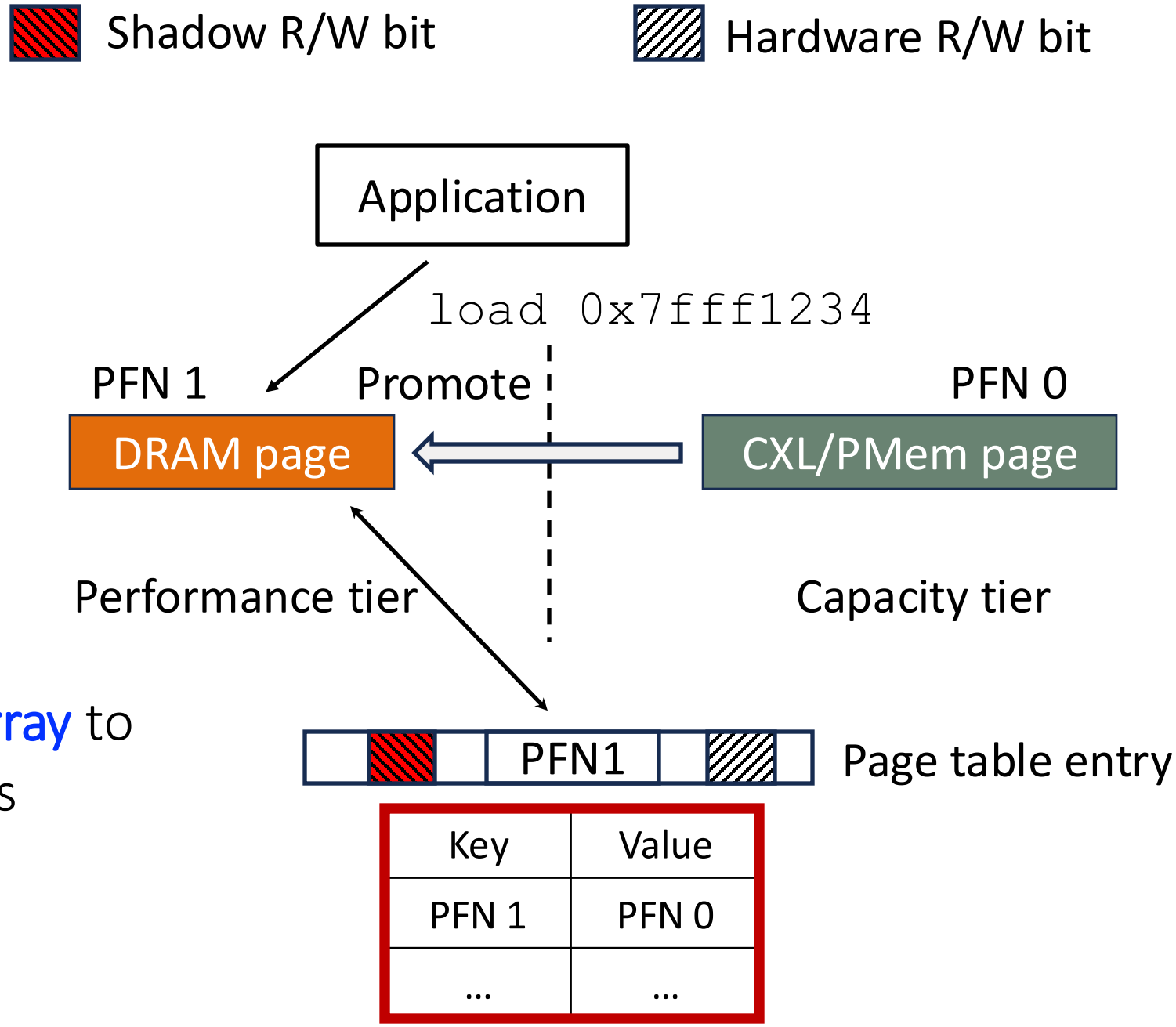
Page shadowing

Key idea:

Temporarily keep a shadow copy of a page promoted from slow to fast memory



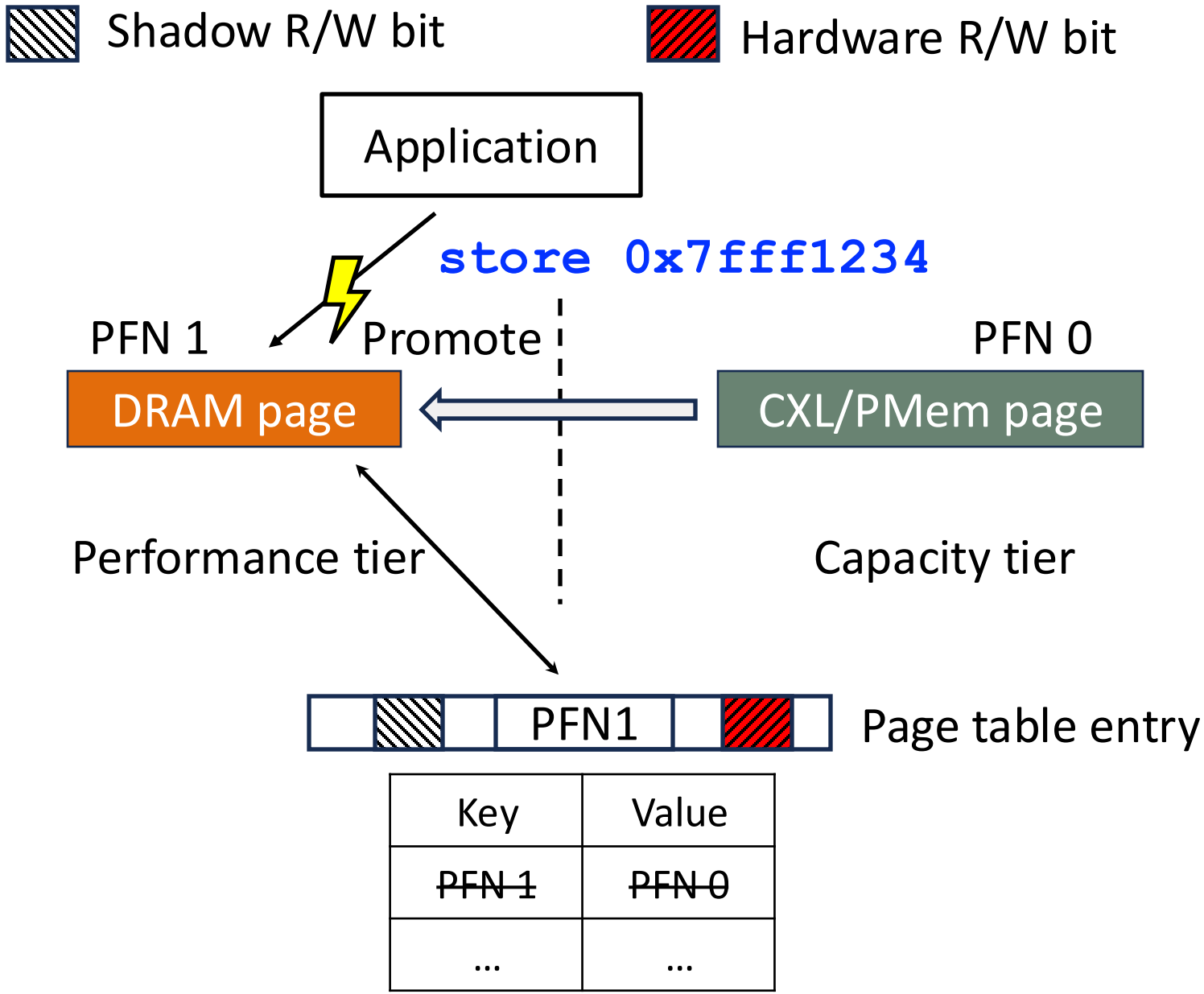
Page shadowing



NOMAD uses an XArray to index shadow pages

Page shadowing

NOMAD marks all master pages as read-only



Evaluation

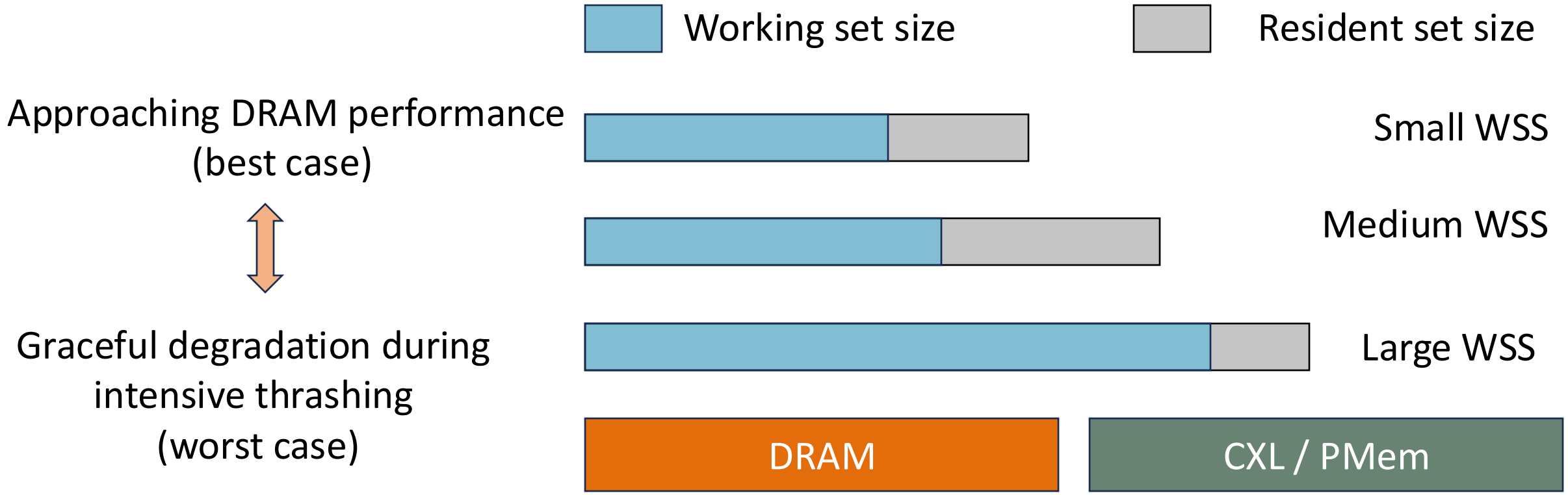
Testbeds:

- Intel Xeon Cascade Lake + Intel Optane persistent memory (PMem)
- Intel Sapphire Rapids CPU + Intel Agilex FPGA-based CXL memory (CXL-FPGA)
- AMD Geona CPU + Micron ASIC-based CXL memory (CXL-product)

Baselines for comparison:

- Transparent page placement (TPP), ASPLOS'23, a page fault-based and the default tiered memory management scheme in Linux
- Memtis, SOSP'23, a PEBS-based hardware sampling approach

Micro-benchmarks



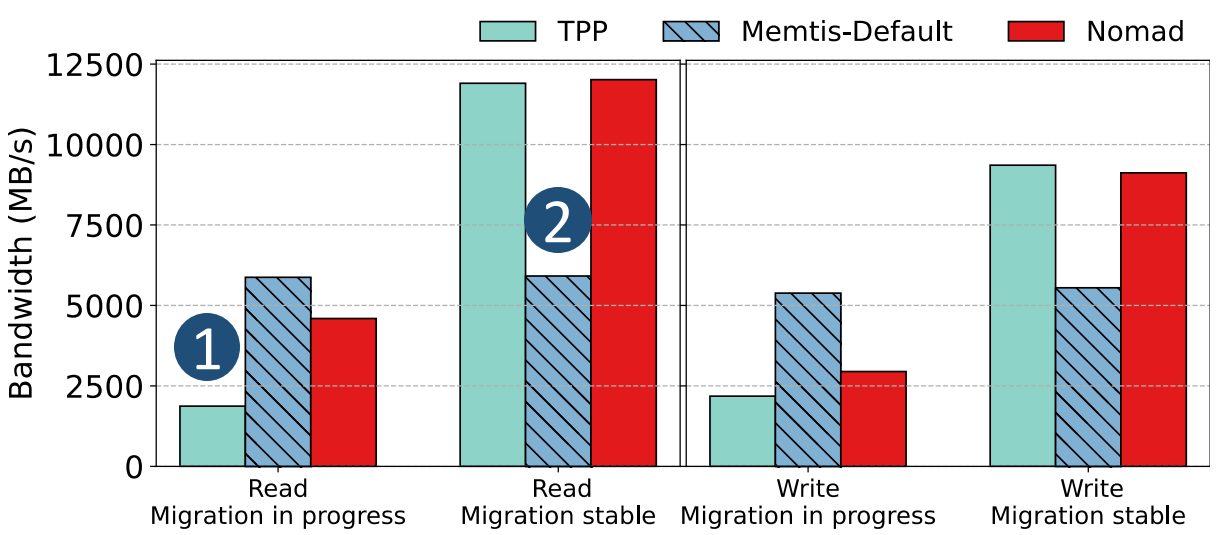
Migration in-progress

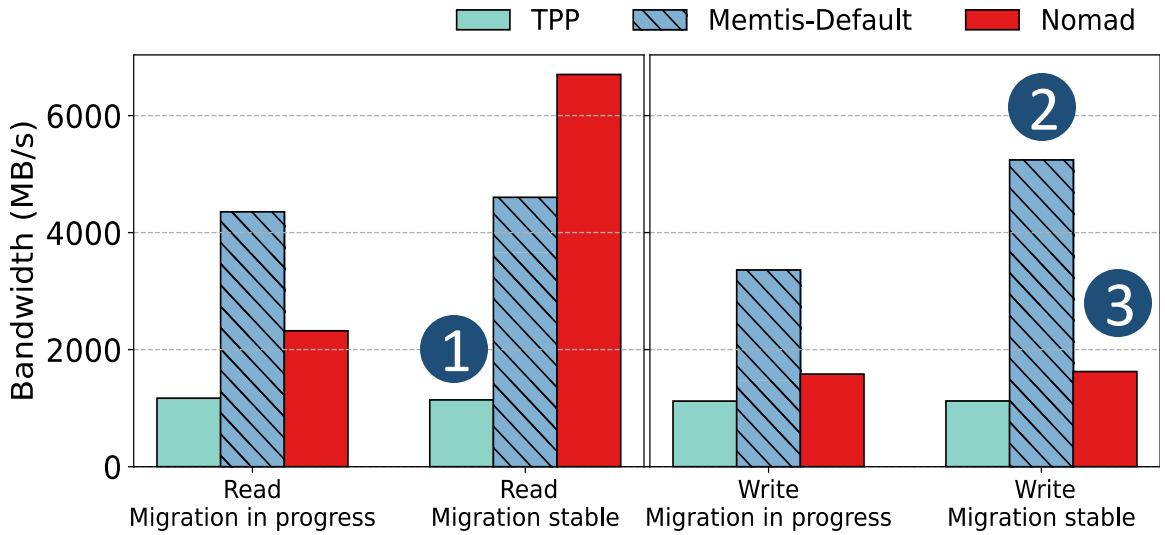
Migration stable

UNIVERSITY OF TEXAS ARLINGTON

Important tradeoffs

Testbed: CXL-FPGA





Small WSS

Large WSS

1. NOMAD significantly outperforms TPP during active migration and for large WSS

2. Sampling-based approach (Memtis) achieves stable performance during thrashing but fails to optimally place hot data in fast memory

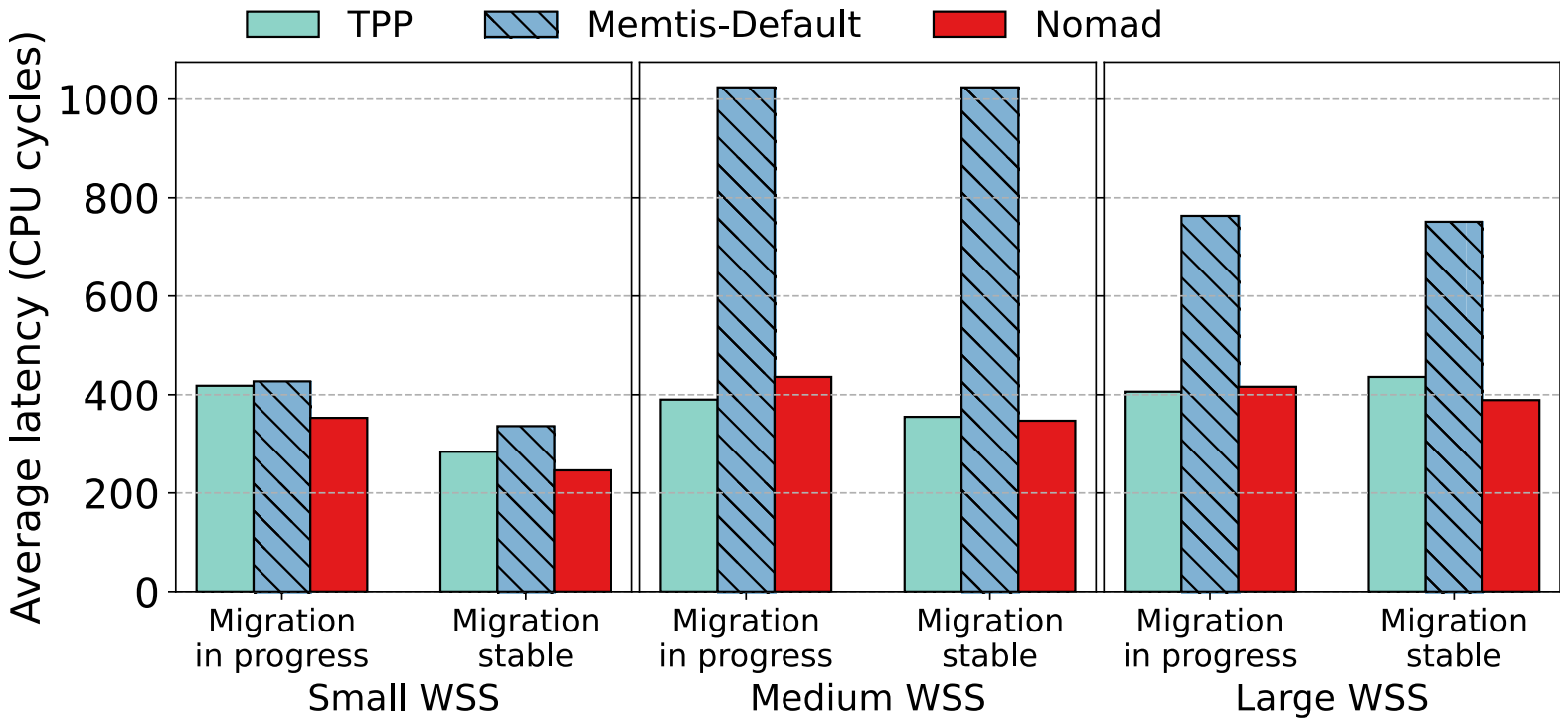
3. NOMAD is more effective for read-only workloads and suffers from migration abortions for write-intensive workloads

UNIVERSITY OF TEXAS ARLINGTON

Sampling-based vs. page fault-based approaches

Testbed: PMem

Lower is better



Although the sampling-based approach maintains high throughput during thrashing thanks to a lack of migrations, its latency is sub-optimal, suggesting page migration is ineffective

UNIVERSITY OF TEXAS ARLINGTON

Conclusions

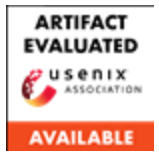
NOMAD is a tiered memory management mechanism that features

- Transactional page migration
- Page shadowing
- Non-exclusive memory tiering

Results show that **NOMAD** is significantly **more efficient** than the state-of-the-art tiered memory management scheme in Linux but call for more research on

- The optimal strategy to enable/disable page migrations under high memory pressure

Open sourced at: https://github.com/lingfenghsiang/Nomad



UNIVERSITY OF TEXAS ARLINGTON