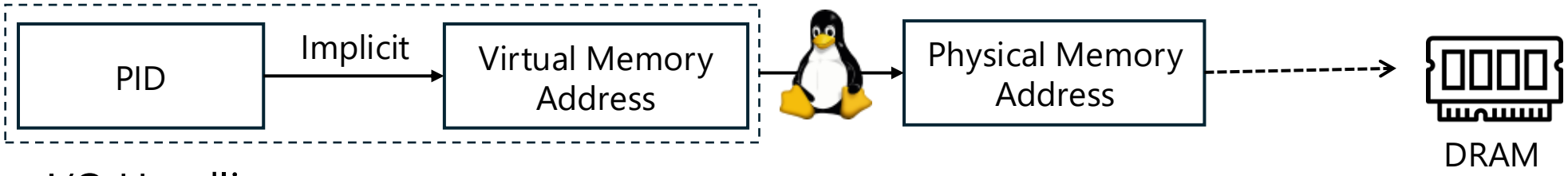
Virtual-Memory Assisted Buffer Management **In Tiered Memory**

Yeasir Rayhan and Walid G. Aref
Purdue University
West Lafayette, Indiana, USA

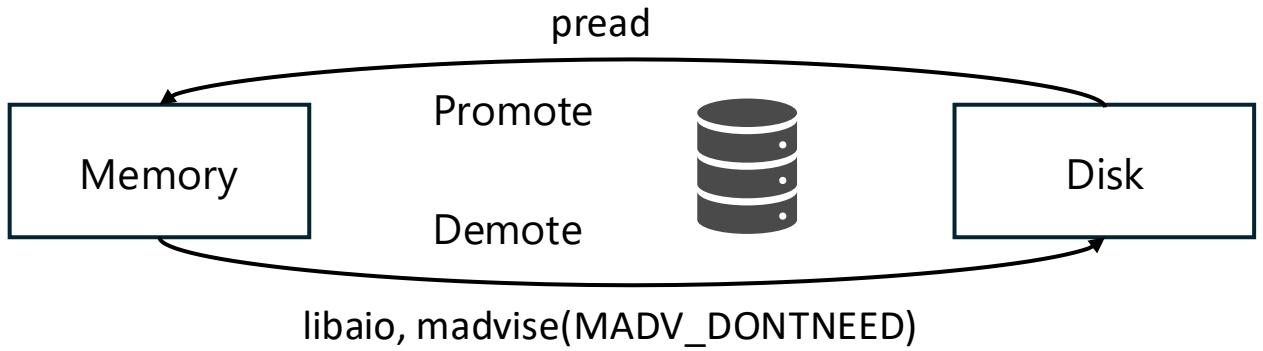
PURDUE UNIVERSITY

vmcache: A Virtual-Memory Assisted Buffer Pool

- ***vmcache*** [1]: proposed in 2023 by Leis et al.
  - Page Translation: PID → Physical Location
    - Leverages the Page-Table data structure of the OS.



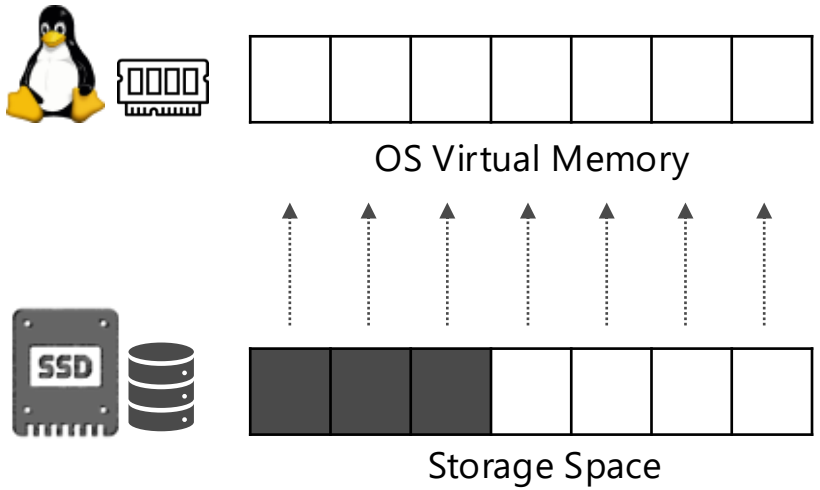
  - I/O Handling
    - DBMS controls page faults and page eviction.
      - Page Promotion: pread
      - Page Demotion: libaio (dirty page), madvise(MADV_DONTNEED) (clean page)



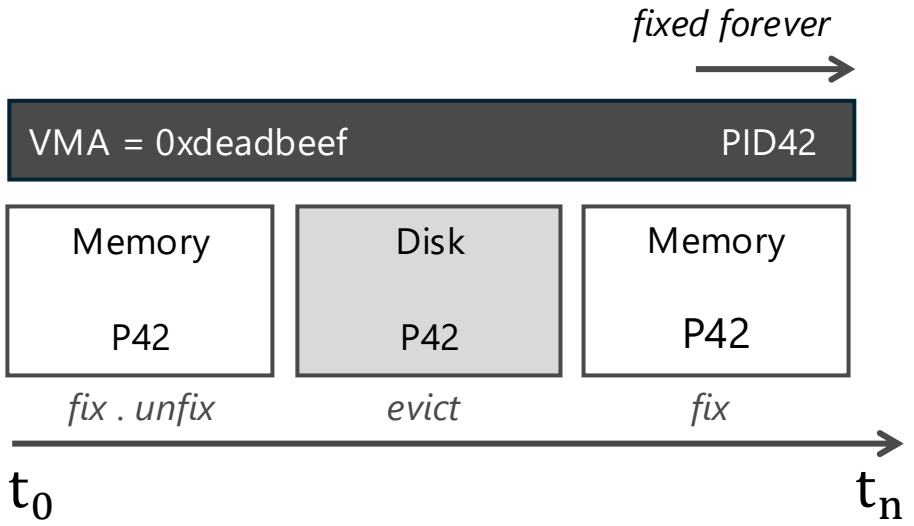
[1] Viktor Leis, Adnan Alhomssi, Tobias Ziegler, Yannick Loeck, Christian Dietrich. Virtual-Memory Assisted Buffer Management. In SIGMOD, 2023

Core Principle of *vmcache*

1. Stable Virtual Addressing.
   - The virtual memory address of a page remains fixed throughout its lifetime.
     - Pre-reserves virtual memory for every possible page (PID) upfront.
     - A 1:1 correspondence between PID and its associated virtual memory address.



A 1:1 mapping between a PID and its VMA.



A single page (P42) over time.

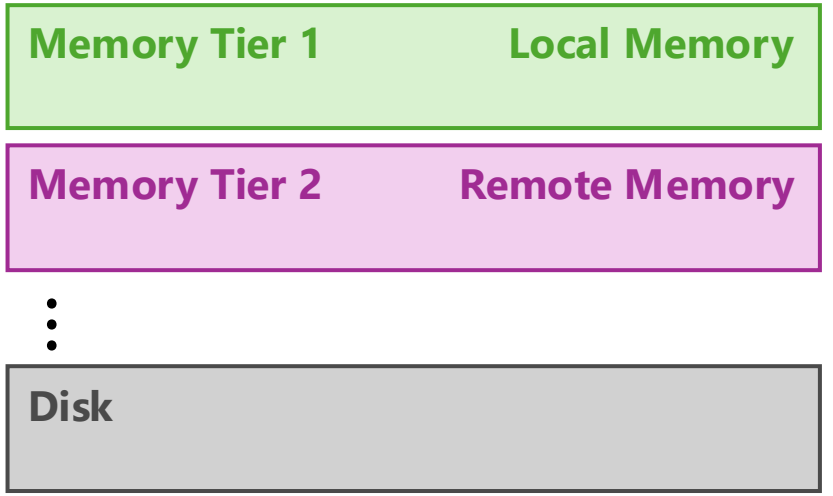
From 2-Tier to n-Tier Virtual-Memory–Assisted Buffer Pools

- *vmcache*:
  - A 2-tier virtual-memory assisted buffer pool

A 2-tier setting

How do virtual-memory assisted buffer pools extend to an n-tier memory hierarchy?

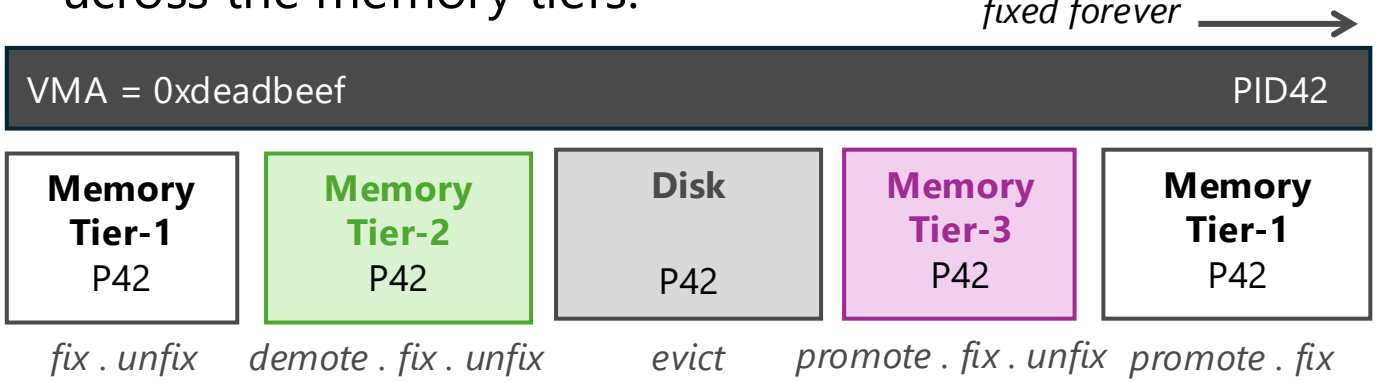
***vmcache*^{n}**: An n-tier virtual-memory assisted buffer pool



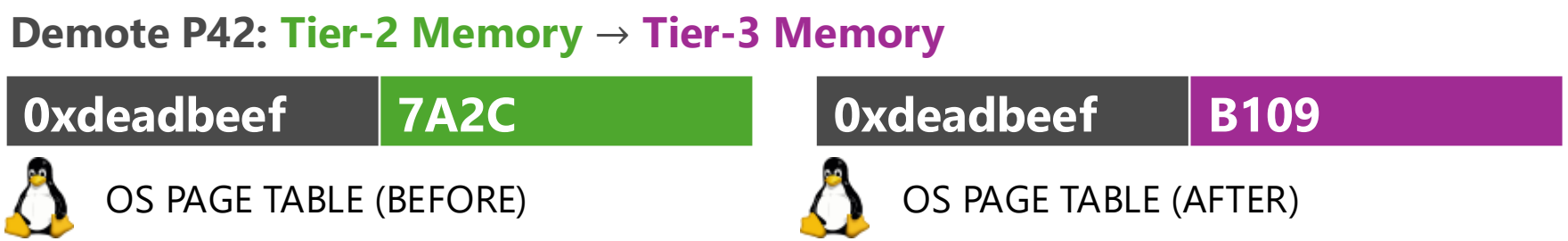
An n-tier setting

Core Principles of *vmcache*^{n}

1. Stable Virtual Addressing.
   - Virtual address of a page remains fixed with dynamic physical frame mappings across the memory tiers.

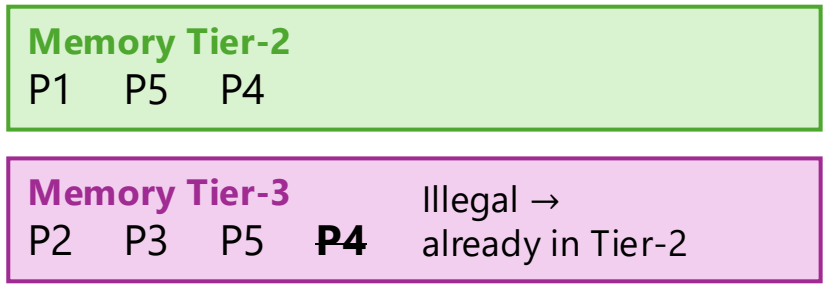


2. Migration via OS Page-Table Remapping.
   - Page migration updates the OS page table while preserving the virtual address
   - OS primitives: `mbind` and `move_pages` (No `memcpy` as it requires changing virtual address).

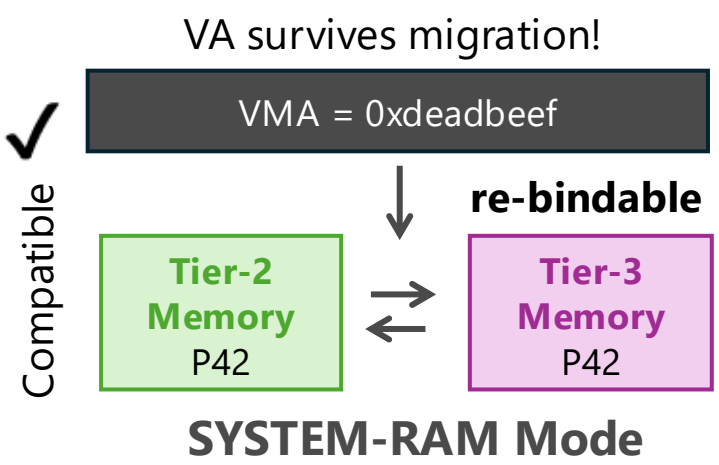


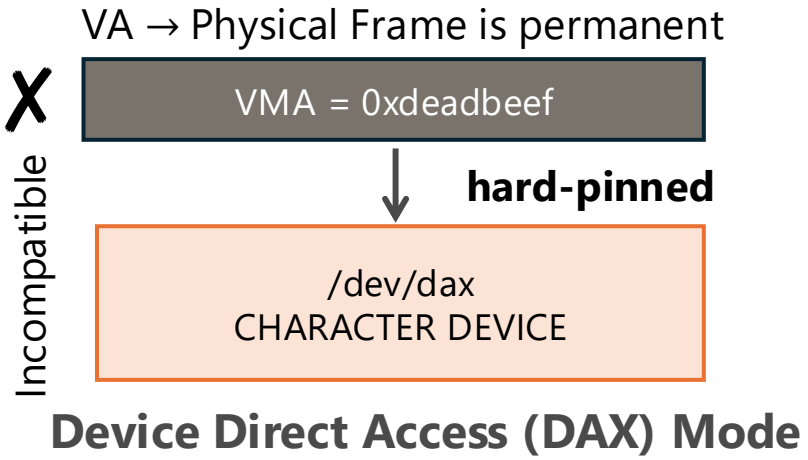
Core Principles of *vmcache*^{n} (cont'd)

3. Single-Tier Residency.
   - Each page resides in exactly one memory tier; no cross-tier replication.



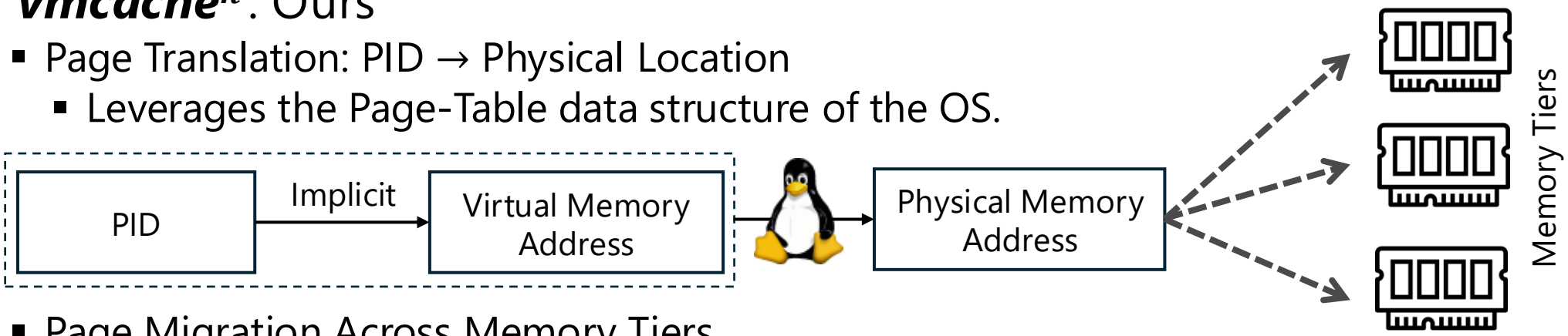
4. System-RAM Requirement (DAX mode is incompatible).
   - vmcache^n requires memory tiers in system-RAM mode, where virtual addresses remain remappable across tiers.



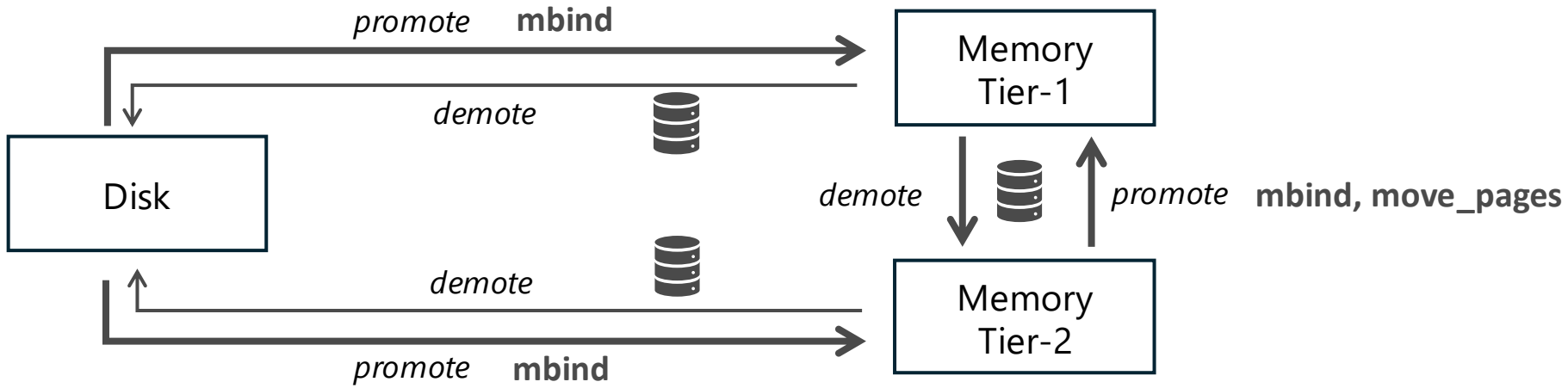


vmcache^n : A Virtual-Memory Assisted Buffer Pool In Tiered Memory

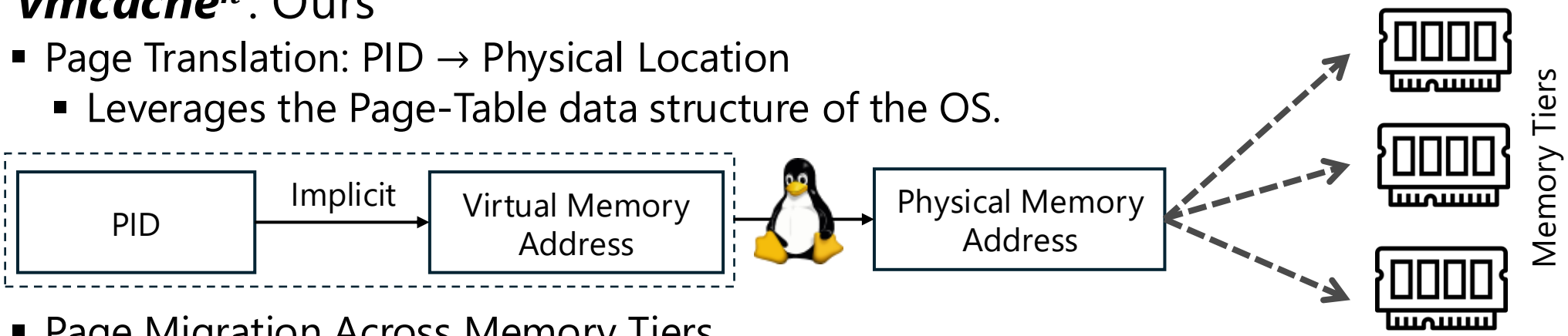
- ***vmcache***^{n} : Ours
  - Page Translation: PID → Physical Location
    - Leverages the Page-Table data structure of the OS.



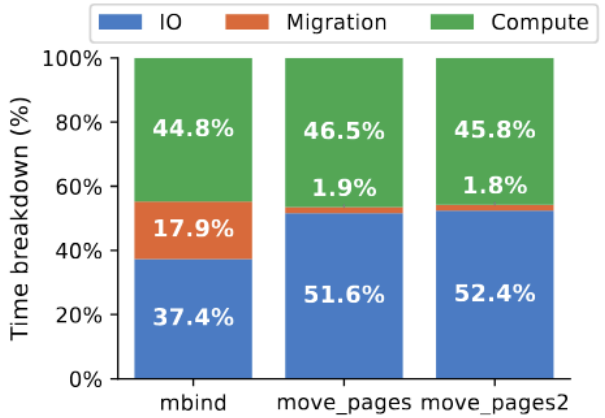
  - Page Migration Across Memory Tiers
    - DBMS controls page promotion and demotions across memory tiers.



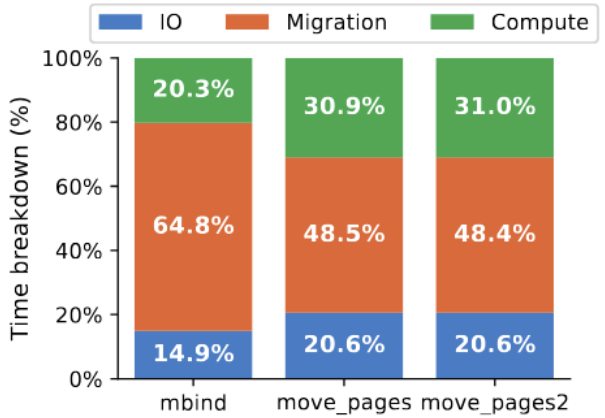
- ***vmcache***^{n} : Ours
  - Page Translation: PID → Physical Location
    - Leverages the Page-Table data structure of the OS.



  - Page Migration Across Memory Tiers
    - On the critical path of *vmcache*^{n} (18% of execution time in TPC-C)



(a) TPC-C workload.



(b) Random read workload.

Page Migration System Calls In **vmcache^n**

`mbind`		`move_pages`
1 page migration / call		n page migrations / call

Bottlenecks: High TLB shootdown overhead and Abort-on-Failure strategy.

Our Proposed System Call

`move_pages2`
n page migrations / call

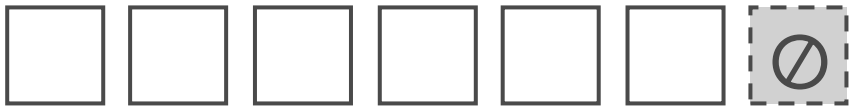
- Introduce 2 new parameters
  1. `nr_max_batched_migration`
  2. `migrate_mode`

```
long move_pages2 (unsigned long count, void *pages[.count],
    const int nodes[.count], int status[.count],
    enum migrate_mode mode, int nr_max_batched_migration);
```

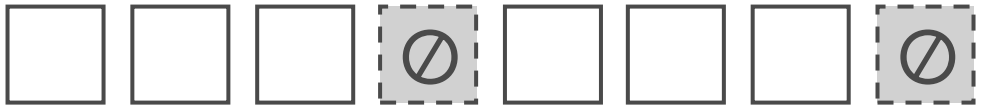
- `count`: The number of pages to process.
- `pages`: An array of page-pointers that need to be processed.
- `nodes`: An array of target NUMA node IDs.
- `status`: An array to store the migration status of each page.
- `mode`: The mode of migration.
- `nr_max_batched_migration`: The maximum number of pages that can be accumulated before TLB shootdown is invoked.

Design Principles of move_pages2

1. Tunable Batch Size
   - `nr_max_batched_migration` controls migration batch size and **amortizes TLB shootdown cost**.
   - Fixed batch size 512 in move_pages.



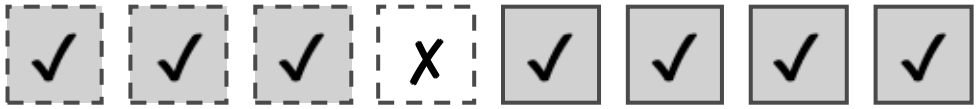
MOVE_PAGES2: 1 TLB SHOOTDOWN



MOVE_PAGES: 2 TLB SHOOTDOWNS

Moving > 512 pages

2. Optimistic Error Handling
   - `migrate_mode` introduces **optimistic error handling**.
   - Records failed pages and continues migration.



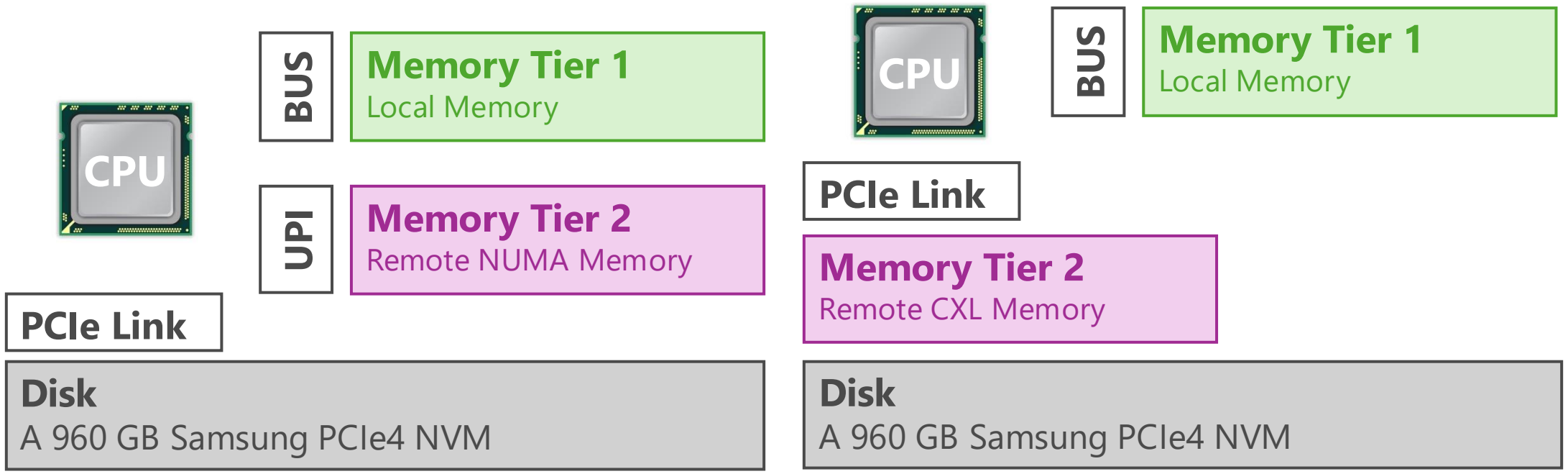
MOVE_PAGES2: 0 PAGE WASTED

MOVE_PAGES: 4 PAGES WASTED

PURDUE UNIVERSITY

Experiments: Setup

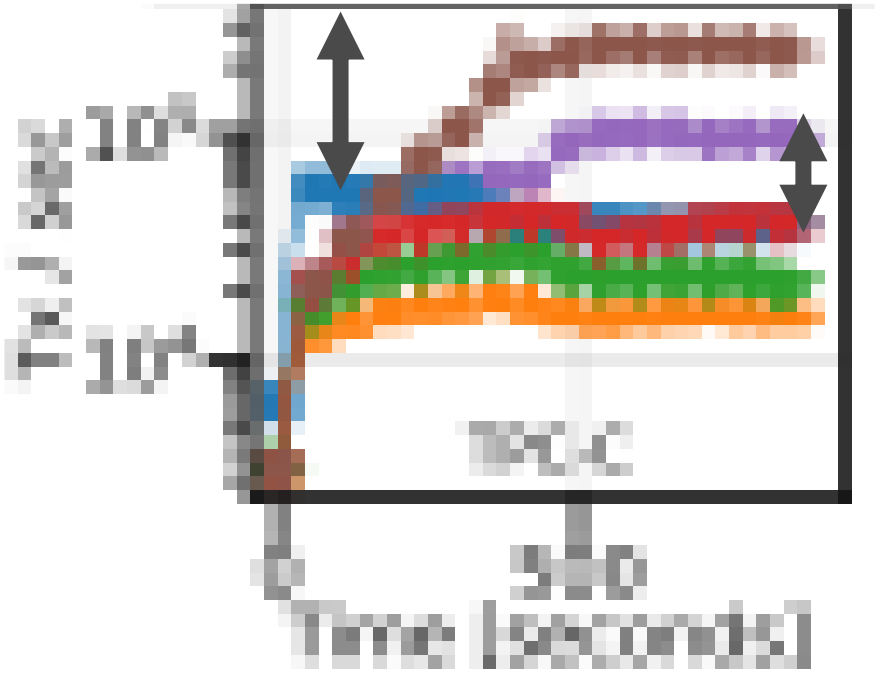
- Take the results with a grain of salt!



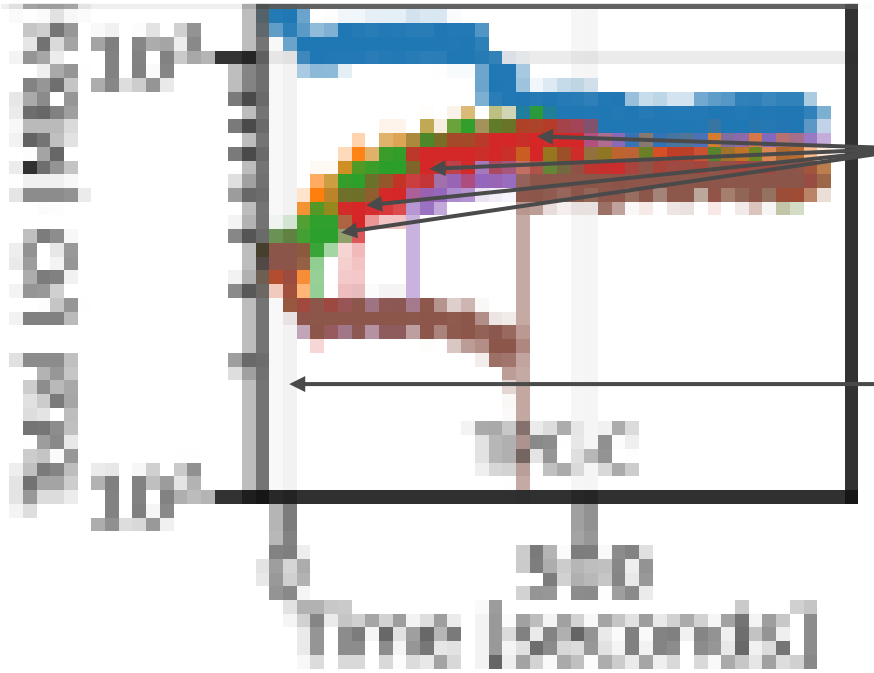
Experiments: vmcache vs vmcache^n

***vmcache^n* (X)**
Local Memory Tier = 32 GB
Remote Memory Tier = X GB

3.82×
Remote Tier =4× Local Tier



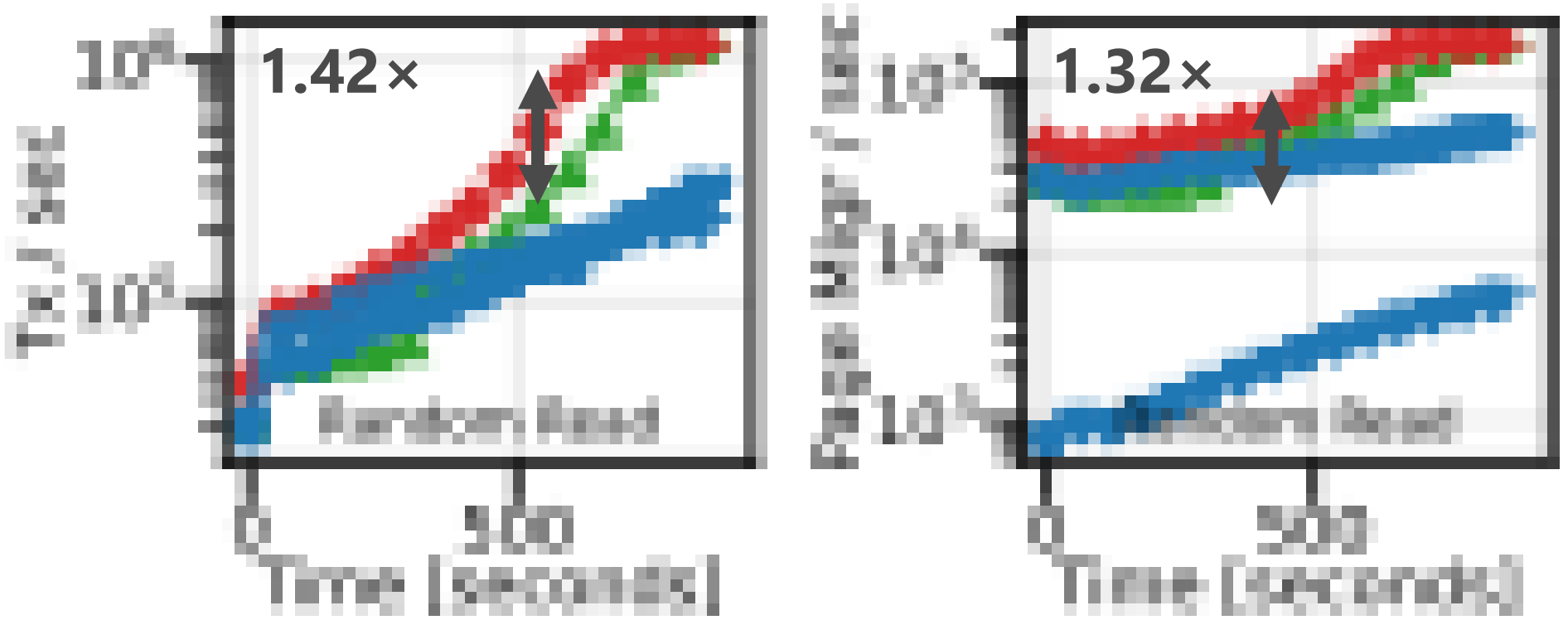
1.67×
Remote Tier = 2× Local Tier



[2] Xinjing Zhou, Joy Arulraj, Andrew Pavlo, David E. Cohen. Spitfire: A Three-Tier Buffer Manager for Volatile and Non-Volatile Memory. In SIGMOD, 2021.

Experiments: move_pages2 vs Alternate OS System Calls

For n page migrations,
mbind (1) = n syscalls
mbind = 1 syscall



Takeaways

- For virtual-memory assisted buffer pools,
  - Remote memory has a cost break-even point at roughly 1-2× DRAM capacity.
  - Page migration is the primary bottleneck.
    - For TPC-C and Random Read, page migration across memory tiers can account for up to 18% and 49% of execution cost, respectively.

Thank You!

Questions?

Read The Paper!

Get in touch!

PURDUE UNIVERSITY