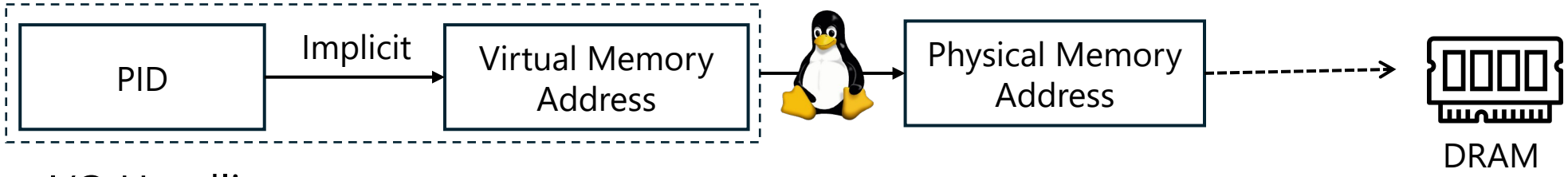
# Virtual-Memory Assisted Buffer Management **In Tiered Memory**

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

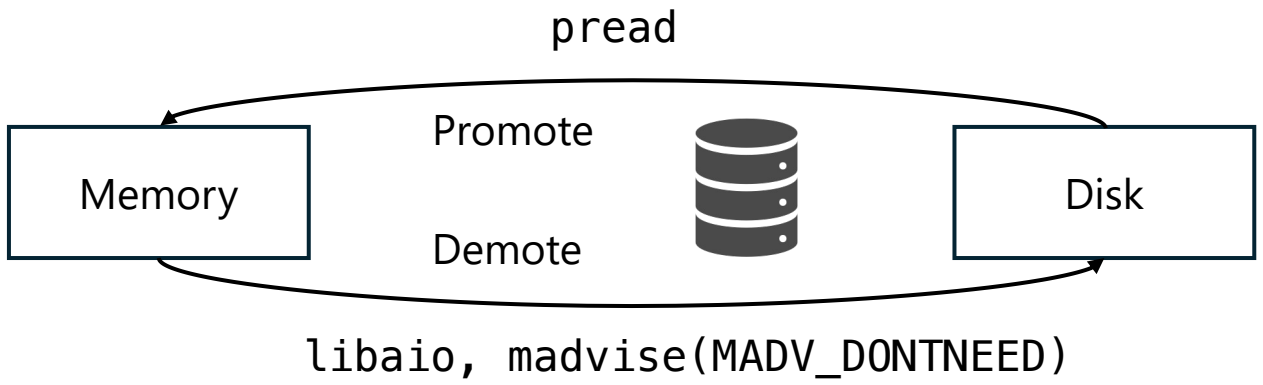
PURDUE UNIVERSITY

# vmcache: A Virtual-Memory Assisted Buffer Pool

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

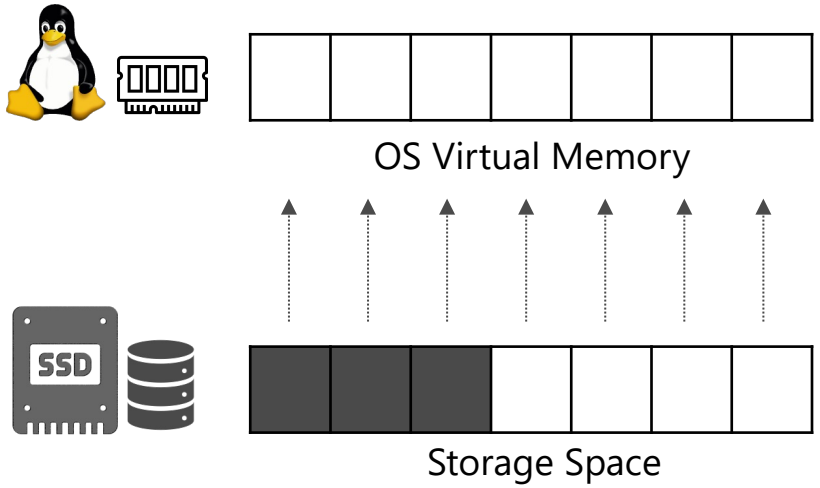



  - I/O Handling
    - DBMS controls page faults and page eviction.
      - Page Promotion: `pread`
      - Page Demotion: `libaio` (dirty page), `madvise(MADV_DONTNEED)` (clean page)

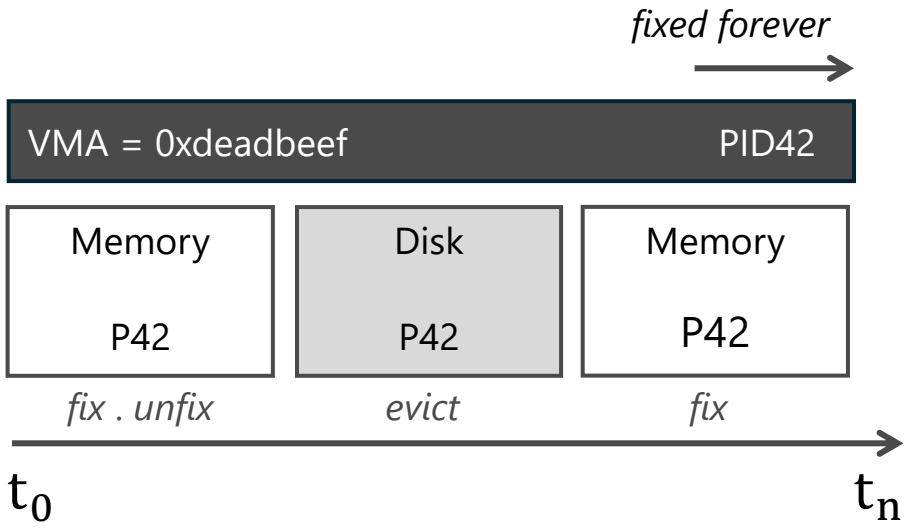


# Core Principle of vmcache

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

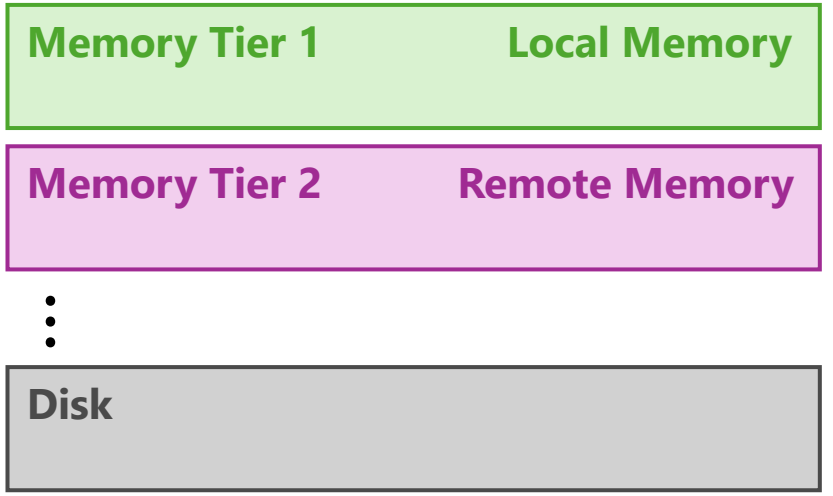
- vmcache:
  - A 2-tier virtual-memory assisted buffer pool




A 2-tier setting

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

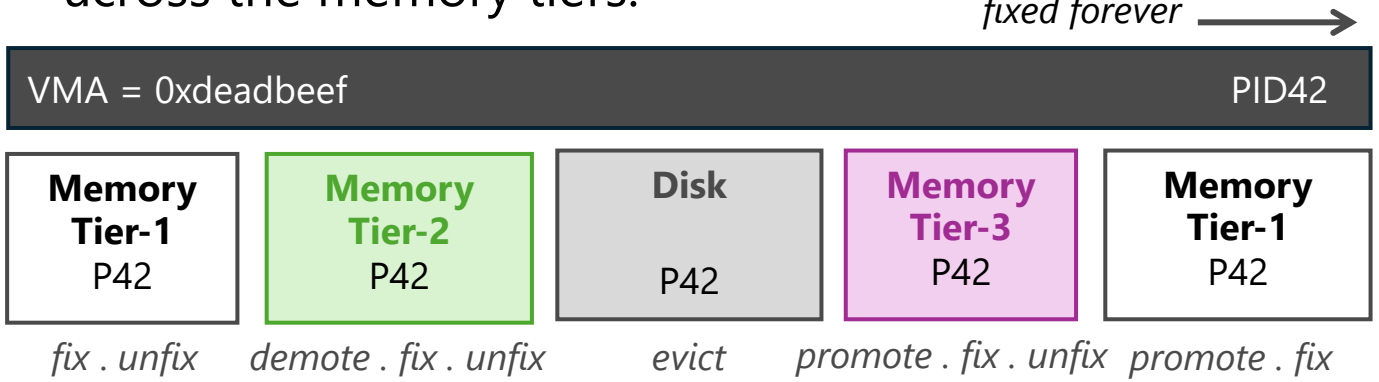
**vmcache$^n$**: An $n$-tier virtual-memory assisted buffer pool




An n-tier setting

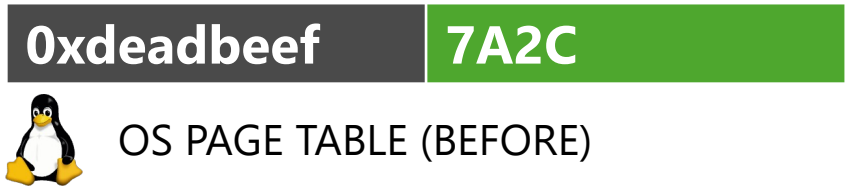
# Core Principles of vmcache$^n$

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 across tiers updates the virtual-to-physical page-table mapping while preserving the virtual address (`mbind`, `move_pages`).
   - No `memcpy` (it requires changing virtual address).

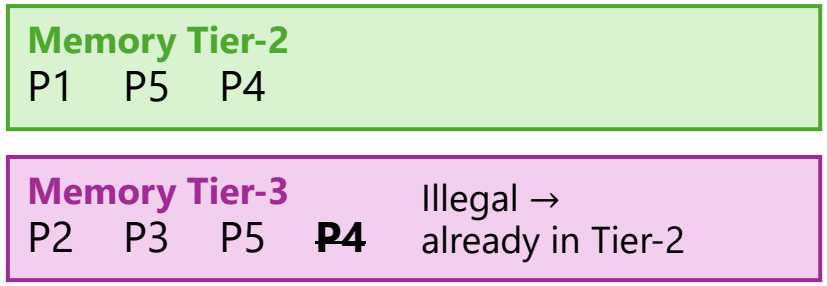
Demote P42: Tier-2 Memory → Tier-3 Memory

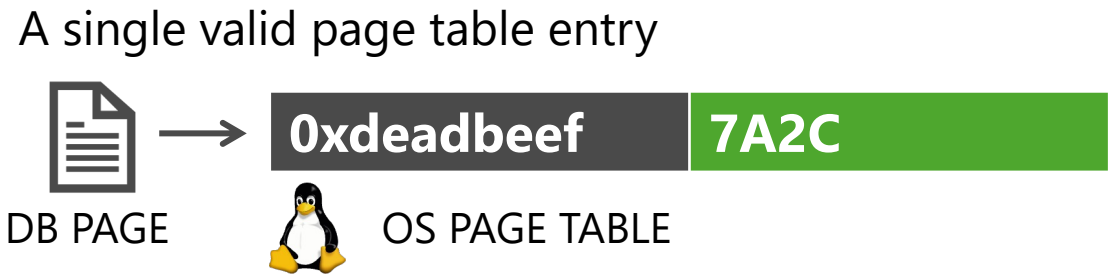




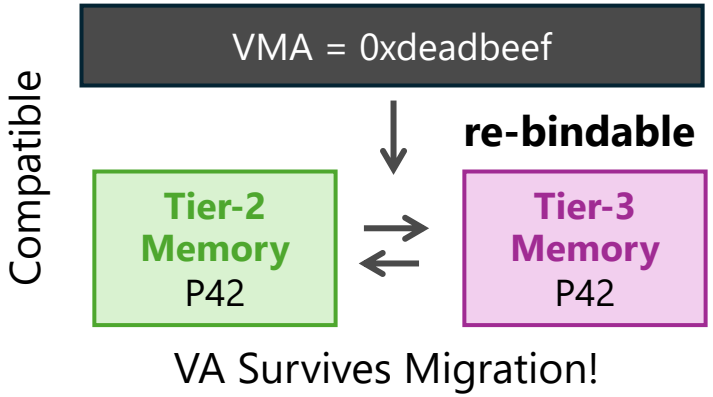

# Core Principles of vmcache$^n$ (cont'd)

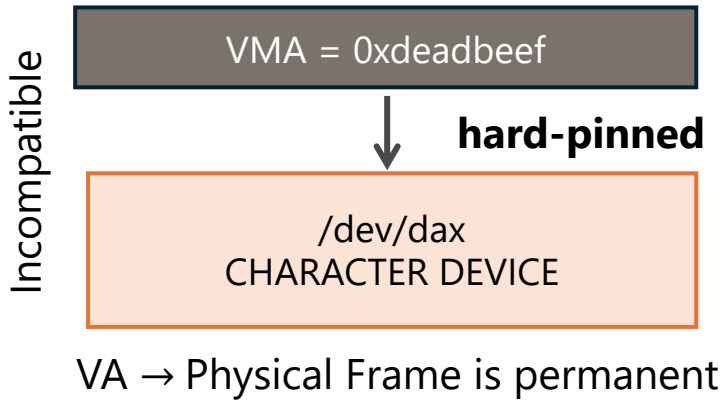
3. Single-Tier Residency.
   - Each page resides in exactly one memory tier; no cross-tier replication.



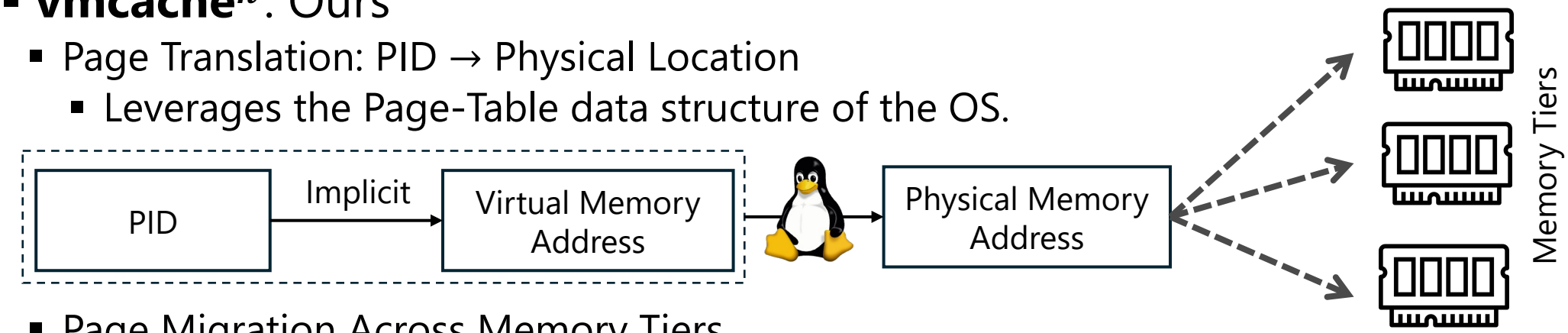




4. System-RAM Requirement.
   - Virtual address must be remappable.
   - Only tiers in system-RAM mode support this virtual-memory based design.

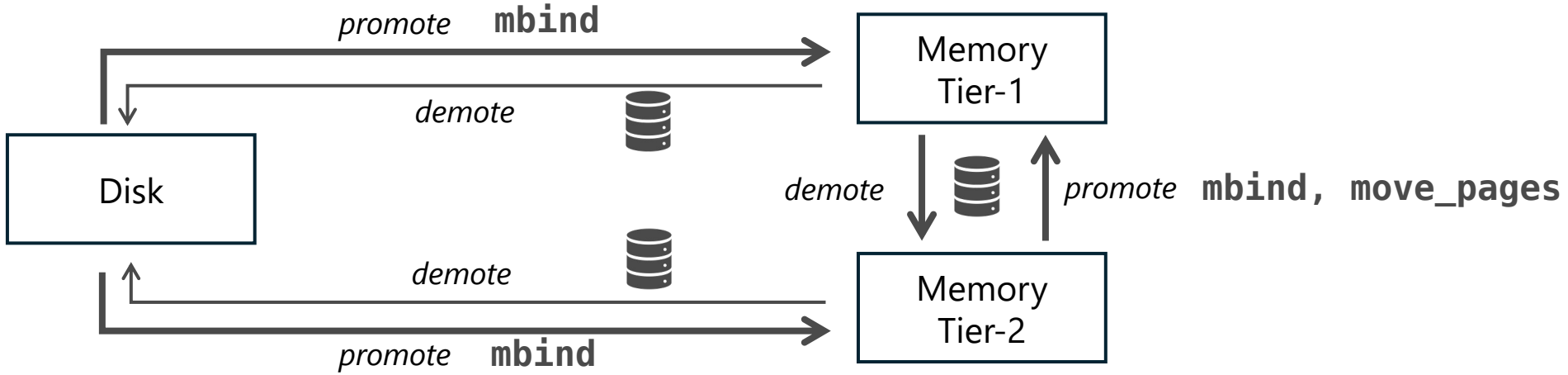


- **vmcache$^n$** : Ours
  - Page Translation: PID → Physical Location
    - Leverages the Page-Table data structure of the OS.




  - Page Migration Across Memory Tiers
    - DBMS controls page promotion and demotions across memory tiers.
      - 2 System calls: `mbind`, `move_pages`



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

- Page Migration across memory tiers is on the critical path.
  - For TPC-C, up to 18% of execution time is spent in page migration.

| `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

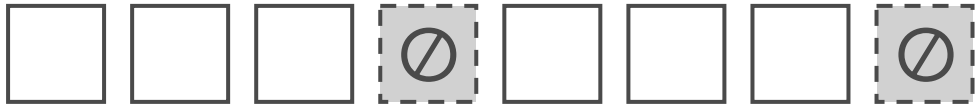
- Introduce 2 new parameters
  - `migrate_mode` and
  - `nr_max_batched_migration`

```
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);
```

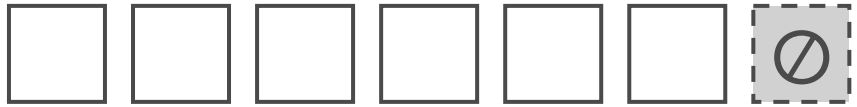
- `count`: The number of pages to process.
- `pages`: An array of page-pointers that need to be processed.
- `nodes`: An array of target NUMA node IDs.
- `status`: An array to store the migration status of each page.
- `mode`: The mode of migration.
- `nr_max_batched_migration`: The maximum number of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

# Design Principles of move_pages2

- Amortizing TLB Shootdown Overhead over `NR_MAX_BATCHED_MIGRATION` pages
  - `NR_MAX_BATCHED_MIGRATION:` Tunable in `move_pages2`
  - `NR_MAX_BATCHED_MIGRATION:` Fixed at 512 in `move_pages`



BEFORE (MOVE_PAGES): 2 TLB SHOOTDOWNS



AFTER (MOVE_PAGES2): 1 TLB SHOOTDOWN

- Optimistic Error Handling
  - Record failed pages and continue migration.



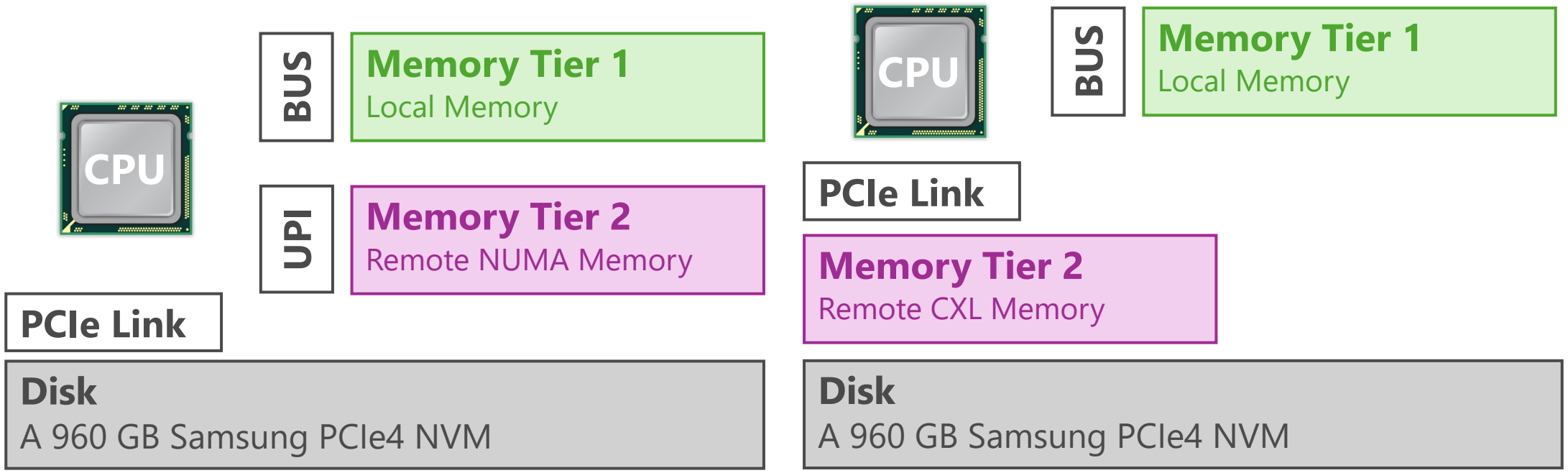
BEFORE (MOVE_PAGES): 7 PAGES WASTED



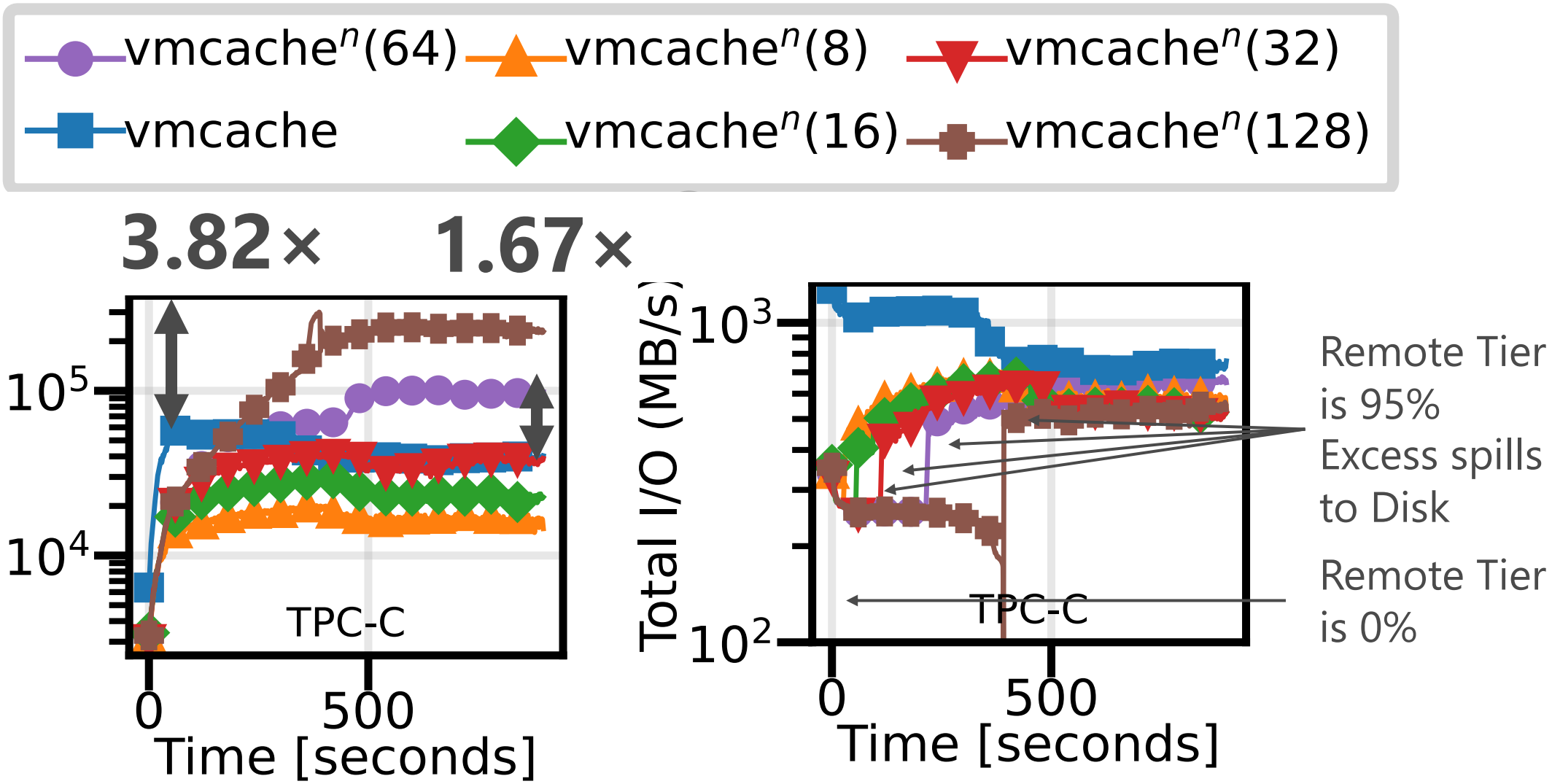
AFTER (MOVE_PAGES2): 1 PAGE WASTED

# Experiments: Setup

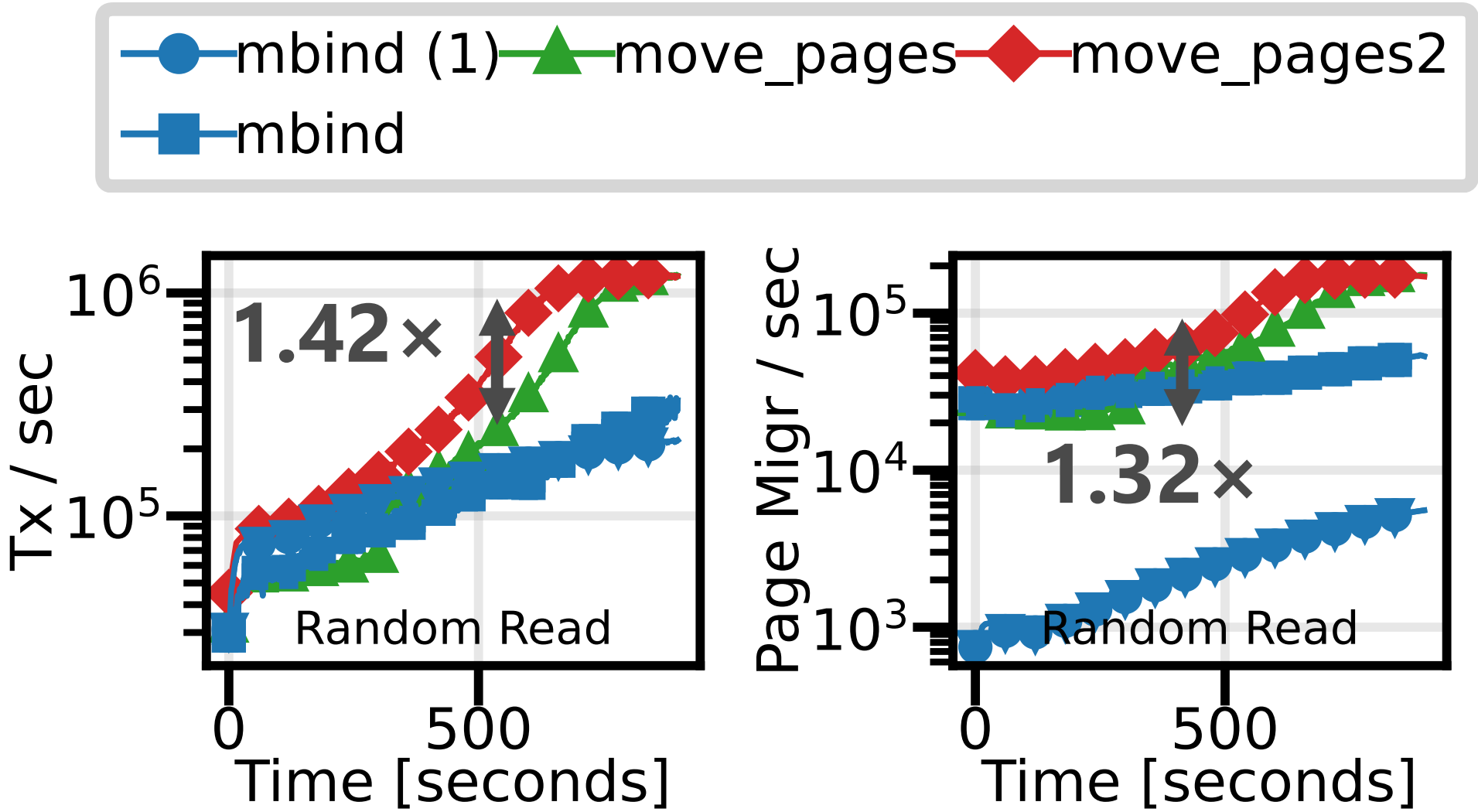
- Take the results with a grain of salt!



# Experiments: vmcache vs vmcache$^n$



# Experiments: move_pages2 vs Alternate OS System Calls



# Conclusion

- 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.

Read The Paper!

Get In Touch!

# Thank You!

Questions?

Read The Paper!

Get In Touch!

PURDUE UNIVERSITY