INTERNATIONAL CONFERENCE ON PARALLEL PROCESSING
54th International Conference on Parallel Processing (ICPP)
September 8 –11, 2025

Leave No One Behind: Towards Fair and Efficient Tiered Memory Management for Multi-Applications

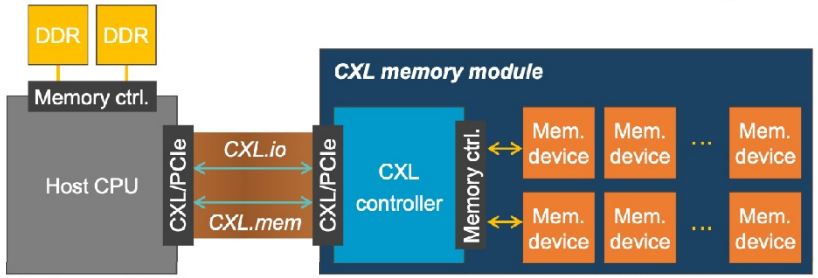
Wenda Tang, Yiduo Wang, Yanwen Wang and Jie Wu

China Telecom Cloud Computing Reserch Institute, Beijing, China

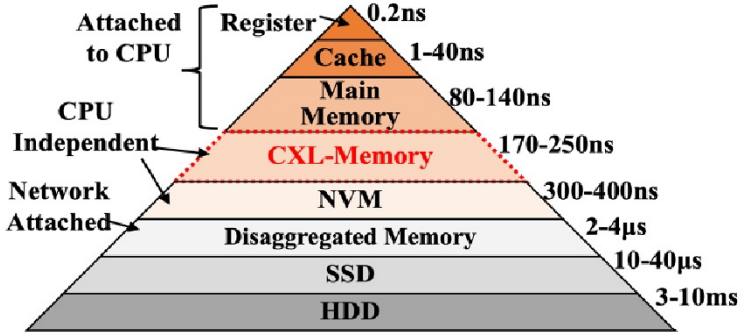
Contents

Introduction of CXL Memory and Memory Tiering

CXL & Performance

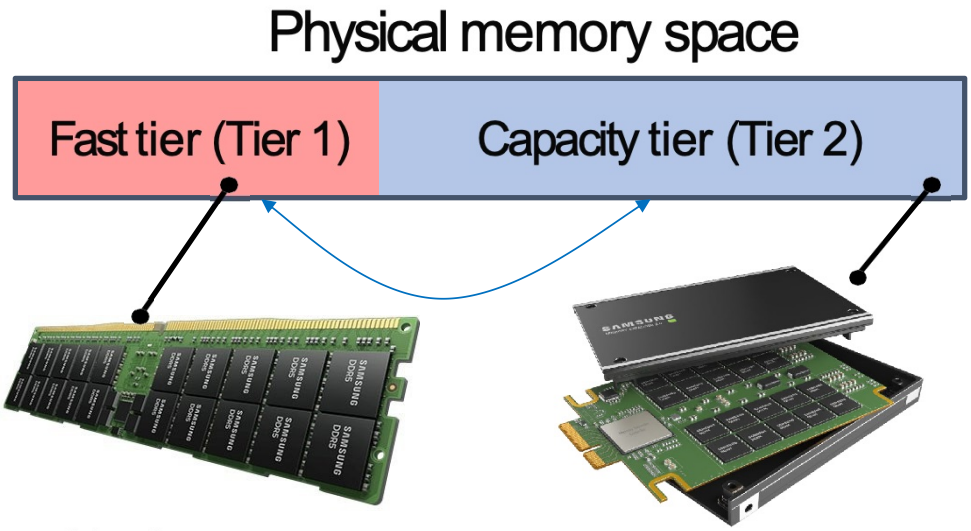


Using CXL memory as a bandwidth expander.
—— Caption, Intel [MICRO'23]



Latency characteristics of memory technologies.
—— TPP, Meta [ASPLOS'23]

Memory Tiering



- Monitor memory accesses
- Decide which pages are hot
- Migrate hot pages to the fast tier

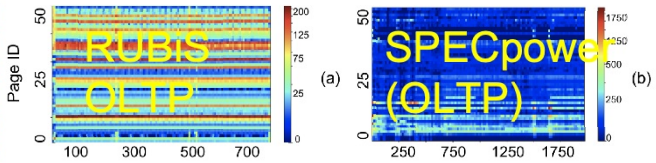
Goal: maximize the utilization of fast tier memory with hot pages

Brief introduciton of Key techniques of Tiered Memory Management

Profiling mechanism

Memory profiling helps
- identify hot and cold data
- guiding efficient data placement and migration.

- **NUMA Hinting Faults**: Injects page faults...
- **Page Table Scanning**: Checks page table access bits
- **CPU PMU:** Monitors cache misses



Workloads exhibit diverse and changing hot/cold data patterns

Migration mechanism

- **Steps:** Kernel trap, PTE lock/unmap, TLB shootdown, copy, remap

- **Modes:**
  - Synchronous (blocks execution)
  - Asynchronous (off critical path)

- **Optimizations:**
  - Huge page & multi-page migration [ASPLOS'19,SOSP'23]
  - DMA/DSA acceleration [SOSP'21, ATC'25]
  - Transactional async migration [OSDI'24]

Migration policy

Decides when and which pages to migrate for optimal performance

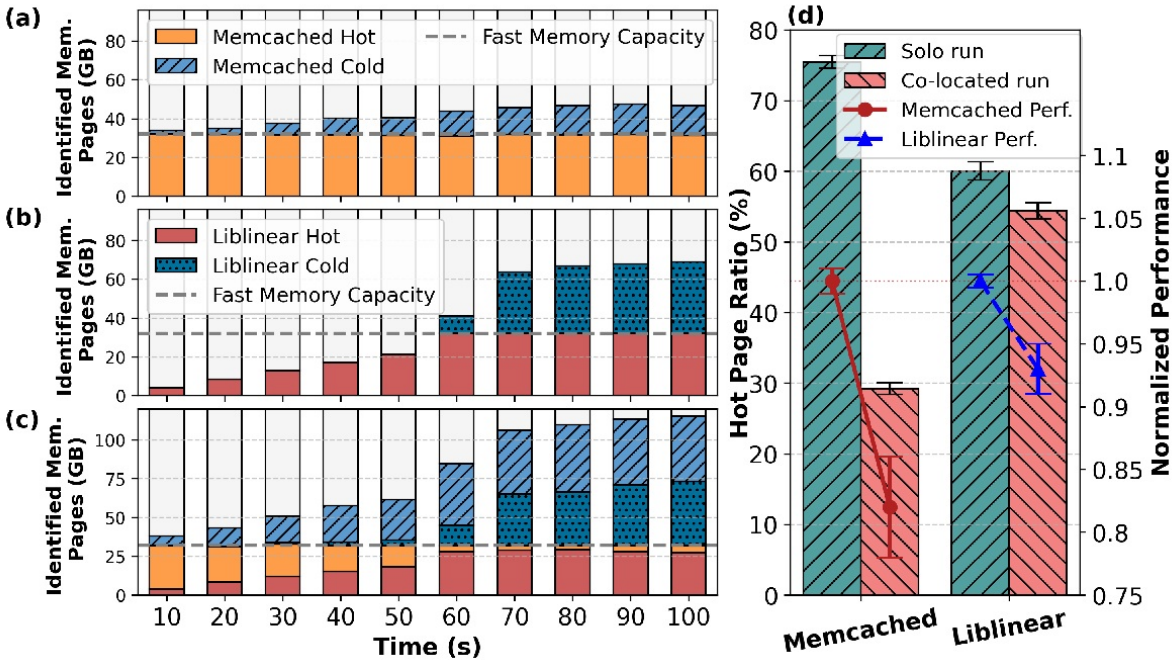
What, When, Where

Common Strategies:
- Proactive page reclamation when fast memory is low (Linux, TPP[ASPLOS'23, AutoTiering[ATC'21])
- Adaptive balancing of hot pages based on access latency/performance gains (SOSP'24,OSDI'25)

Key observations of current memory tiering solution

Observation #1

Workload Diversity Causes the "Cold Page Dilemma"

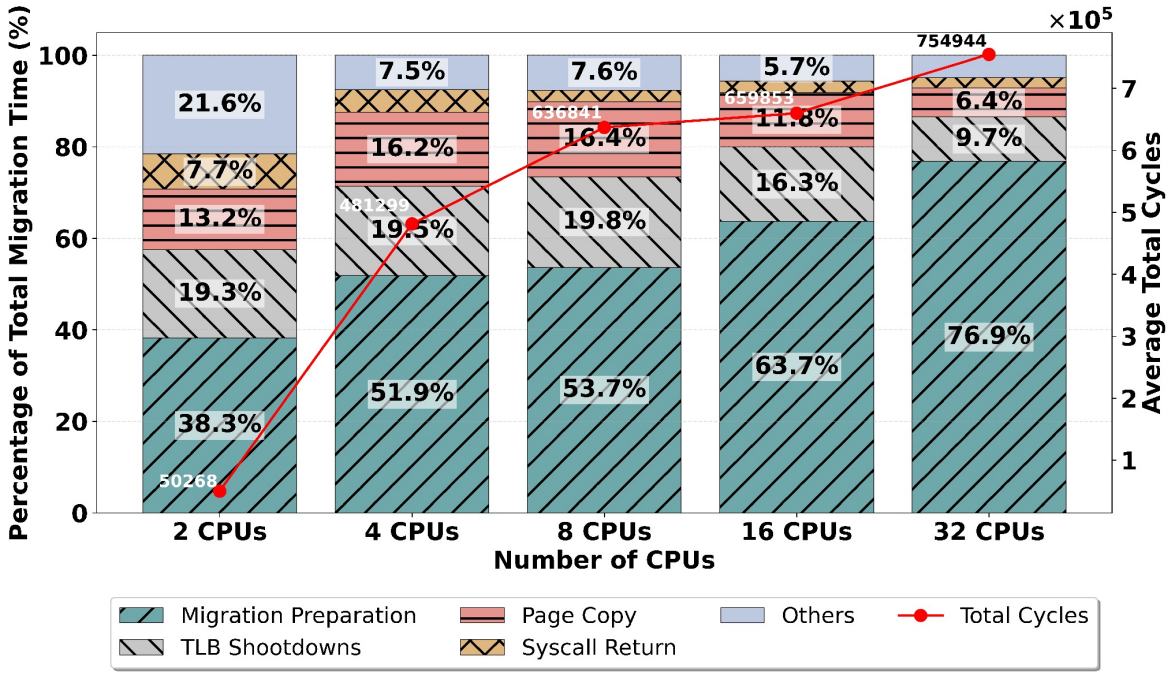


"cold page dilemma" —— current migration policies can unfairly evict cold-but-important pages from latency-sensitive workloads, resulting in performance loss.

Key observations of current memory tiering solution

Observation #2

Migration Overhead is Underestimated on Multi-Core Systems

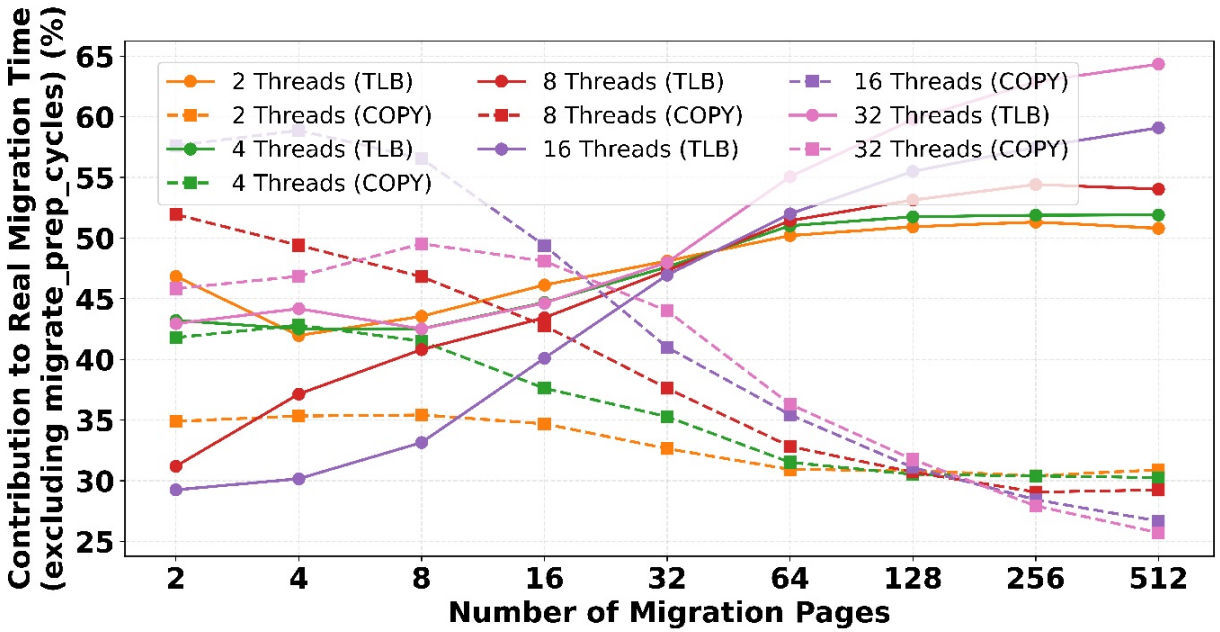


Page migration introduces substantial overhead, especially due to **synchronization** and **TLB shootdown.**

Key observations of current memory tiering solution

Observation #3

Managing TLB coherence efficiently is critical for scalable page migration on many-core systems.

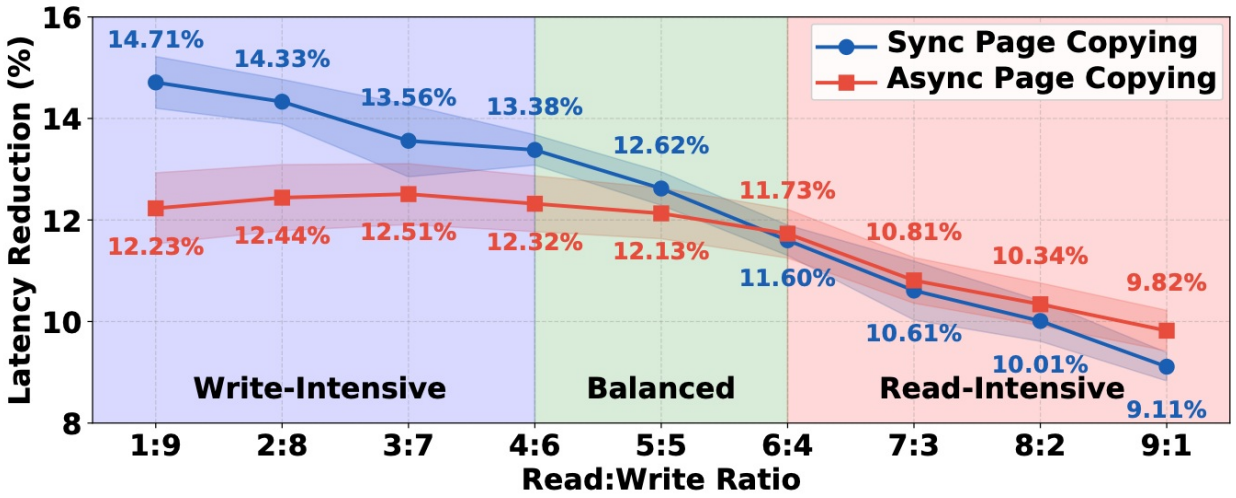


The cost of TLB shootdown increases rapidly as more pages are migrated, limiting scalability.

Key observations of current memory tiering solution

Observation #4

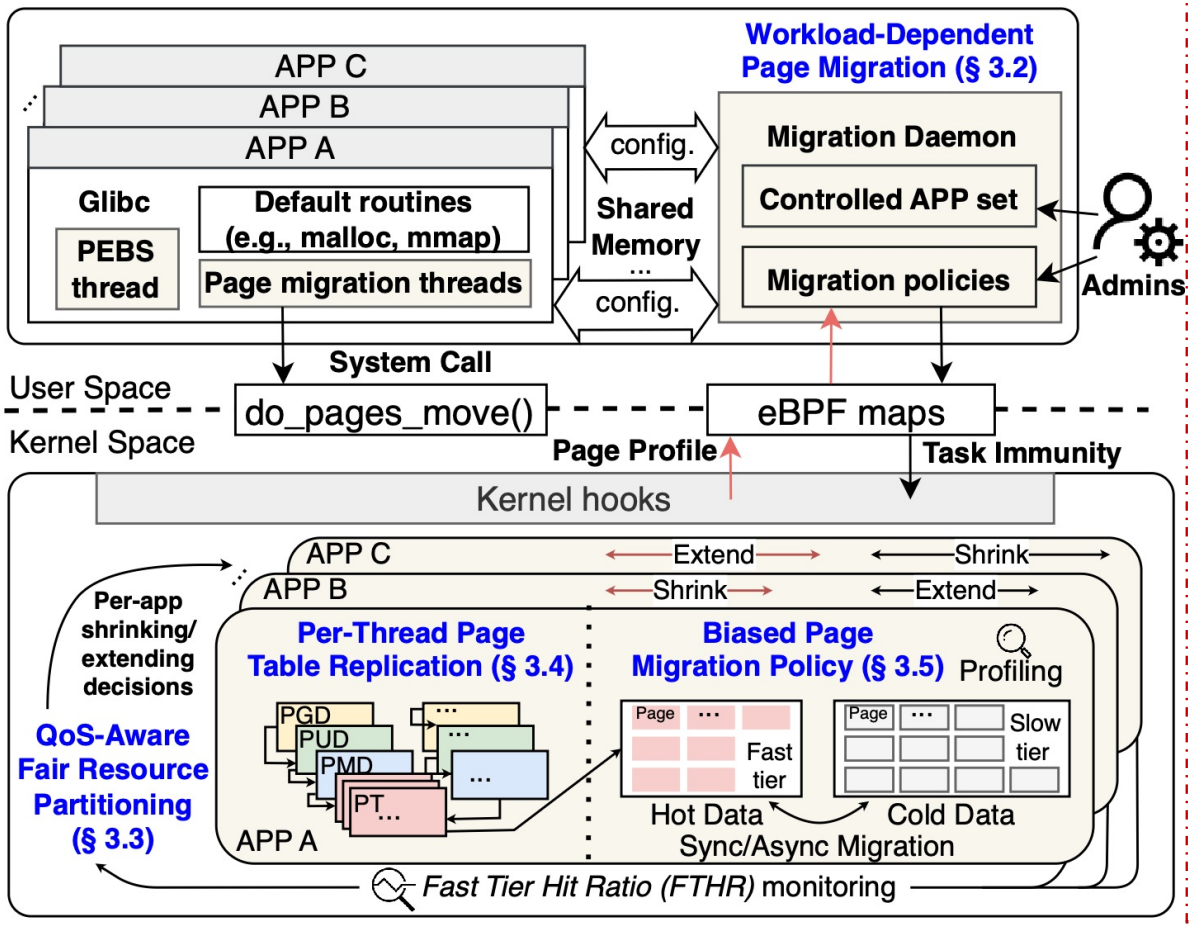
Sync vs. Async Migration Impact Varies by Access Pattern



The performance impact of **synchronous** and **asynchronous** page migration differs depending on the page's access pattern (e.g., read-intensive vs. write-intensive).

Vulcan Design

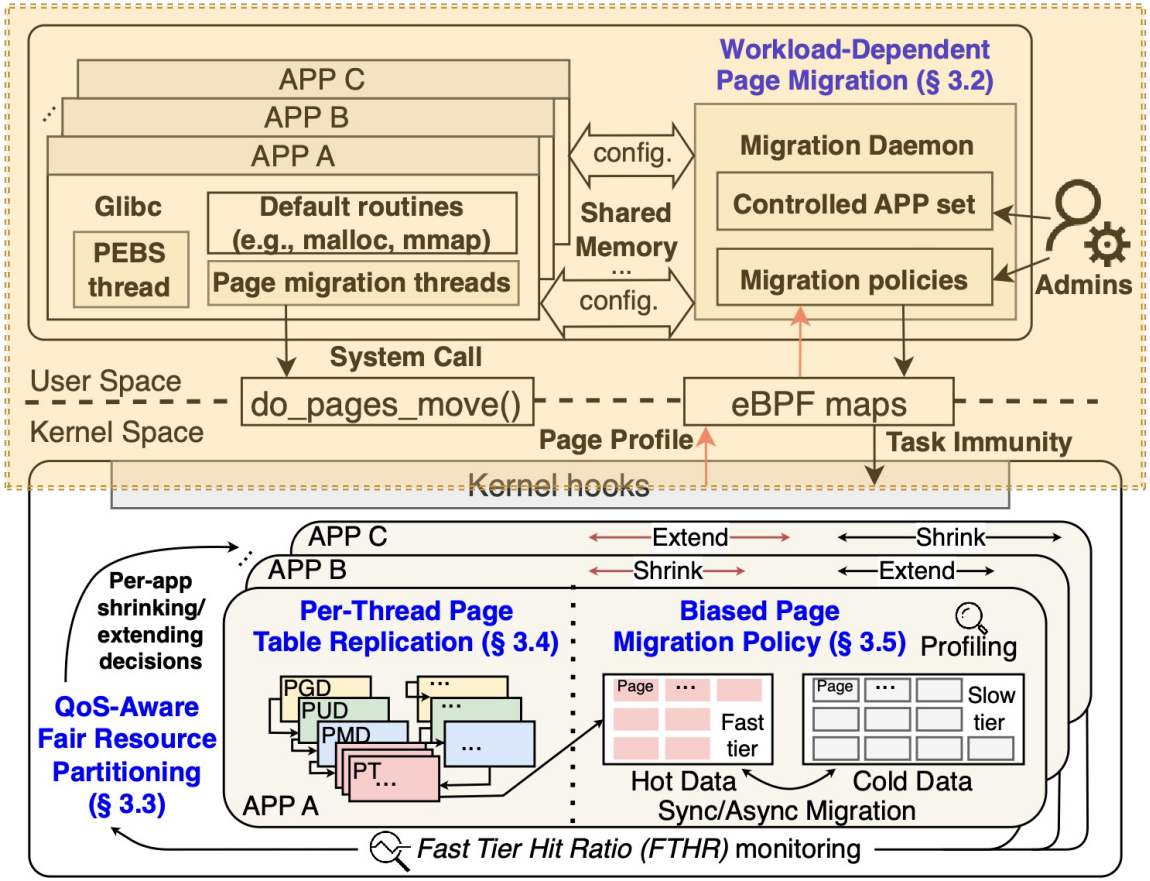
Overview of *Vulcan*



Key Features of *Vulcan*

- Workload-Dependent Page Migration.
  - ✓ dedicated migration threads per application
  - ✓ flexible profiling (PEBS, hinting faults, etc.)
- QoS-Aware Fair Resource Partitioning
  - ✓ Dynamically monitor Fast-Tier Hit Ratio (FTHR)
  - ✓ Credit-Based Fair Resource Partitioning (CBFRP), prioritizing LC workloads
- Per-Thread Page Table Replication
  - ✓ Reduces TLB shootdown overhead
- Biased Page Migration Policy
  - ✓ Categorizes pages by access pattern and ownership (read/write, private/shared)

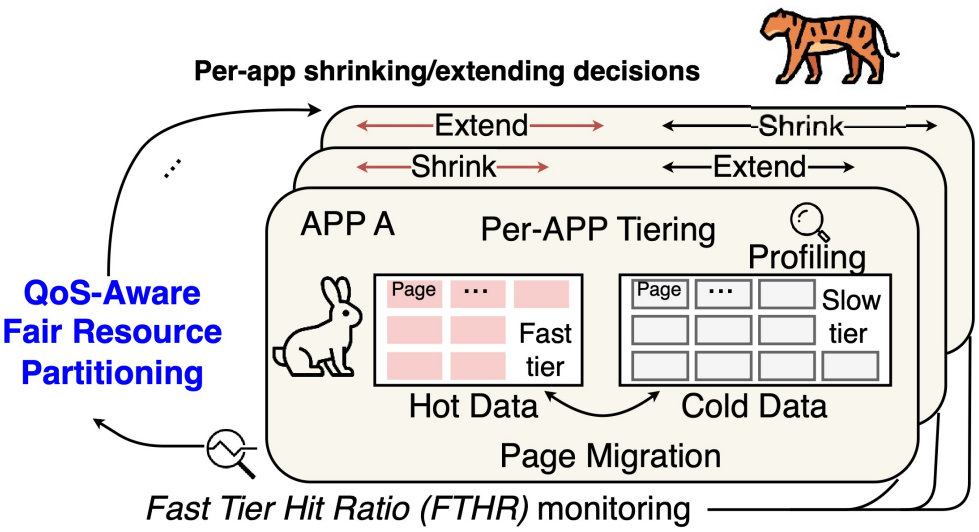
Workload-Dependent Page Migration



Main points:

1. Migration daemon restricts operations to whitelisted apps for security.
2. eBPF enables flexible, decoupled page profiling methods.
3. Hybrid profiling combines counters and page faults to improve accuracy.
4. Migration runs periodically and in parallel, eliminating global sync.

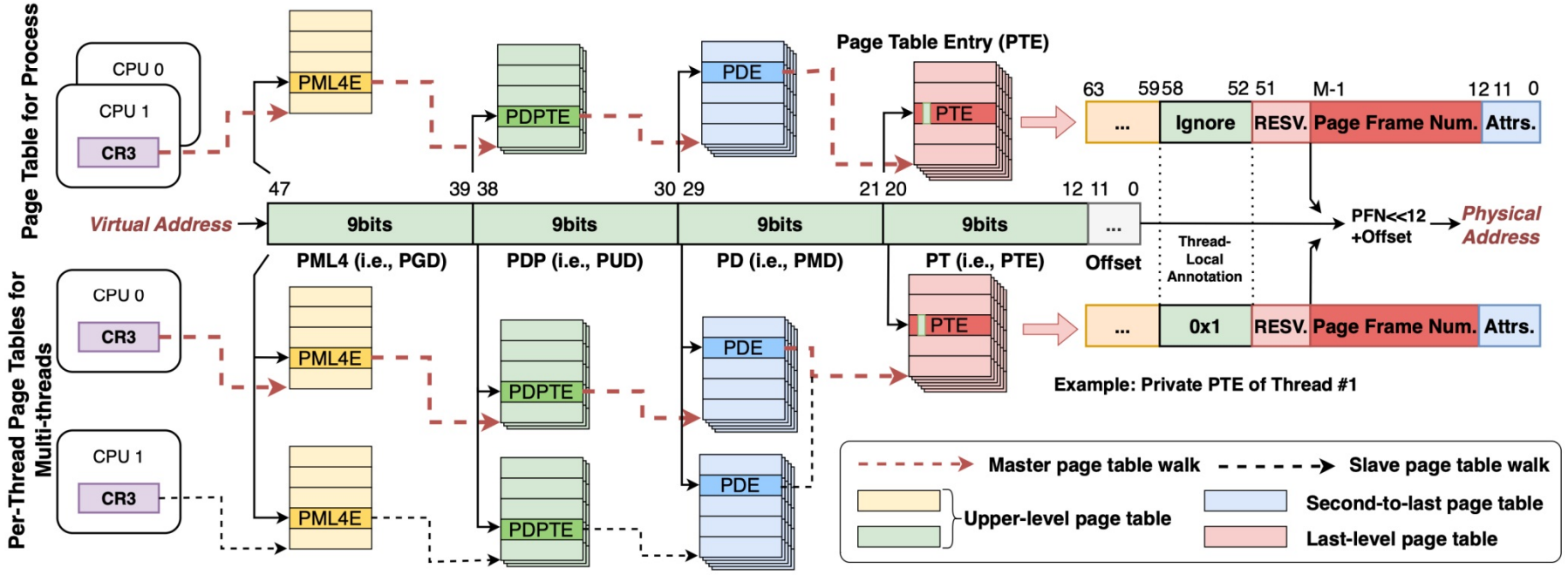
QoS-Aware Fair Resource Partitioning



Main points:

1. Dynamically allocates/adjusts fast memory among workloads
2. Monitors each workload's fast memory usage for fairness guarantee
3. Prioritizes latency-critical applications when allocating resources

Per-thread page table replication



Main points:

1. Maintains separate upper-level page tables for each thread
2. Reduces TLB shootdown overhead during page migration
3. Improves multi-threaded memory access efficiency

Biased migration policy

Multi-Level Feedback Queue (MLFQ)

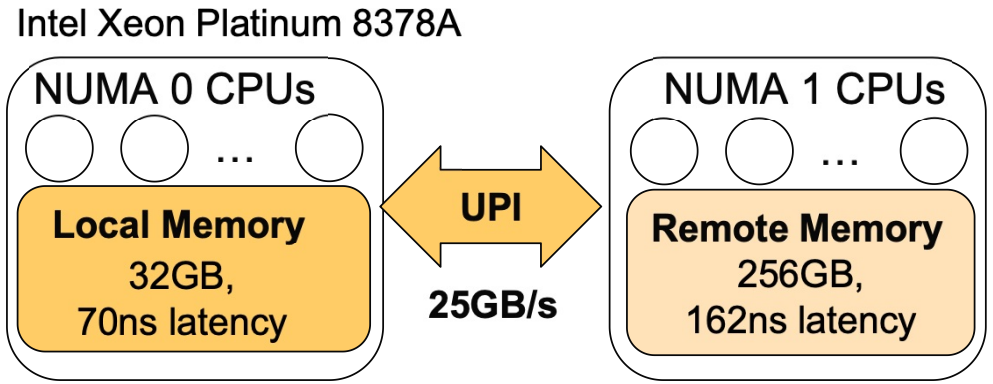
check by PTEs

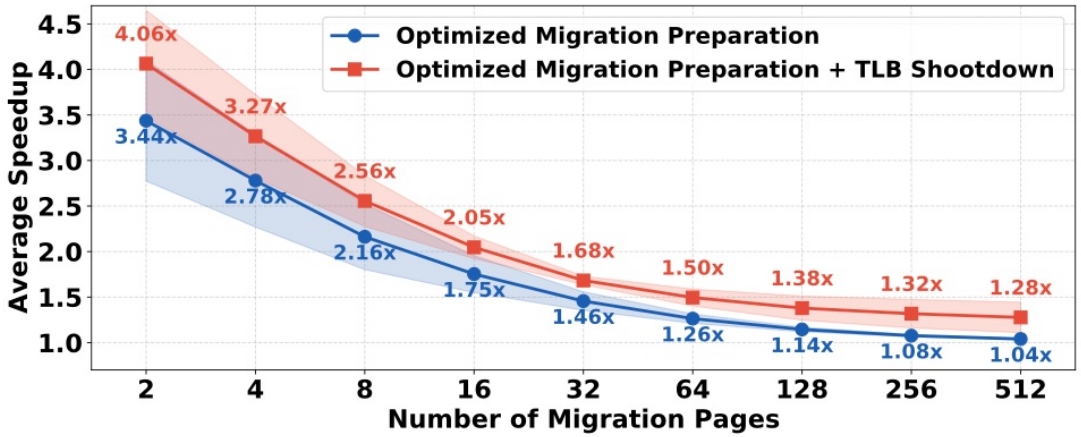
Page Type	Read/Write Pattern	Priority	Strategy
Shared	Read-intensive	★★★	Async copy
Shared	Write-intensive	★	Sync copy
Private	Read-intensive	★★★★	Async copy
Private	Write-intensive	★★	Sync copy

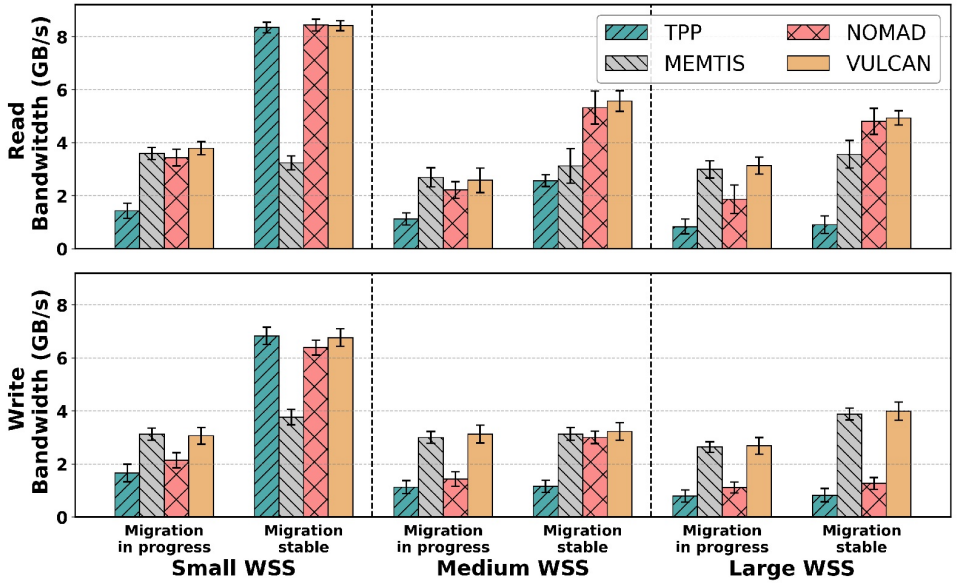
Main points:

1. Differentiates between read-intensive and write-intensive pages
2. Migrates read-intensive/private pages asynchronously to minimize disruption
3. Migrates write-intensive/shared pages synchronously to ensure data consistency
4. Optimizes migration speed and reduces application performance impact

Microbenchmarks





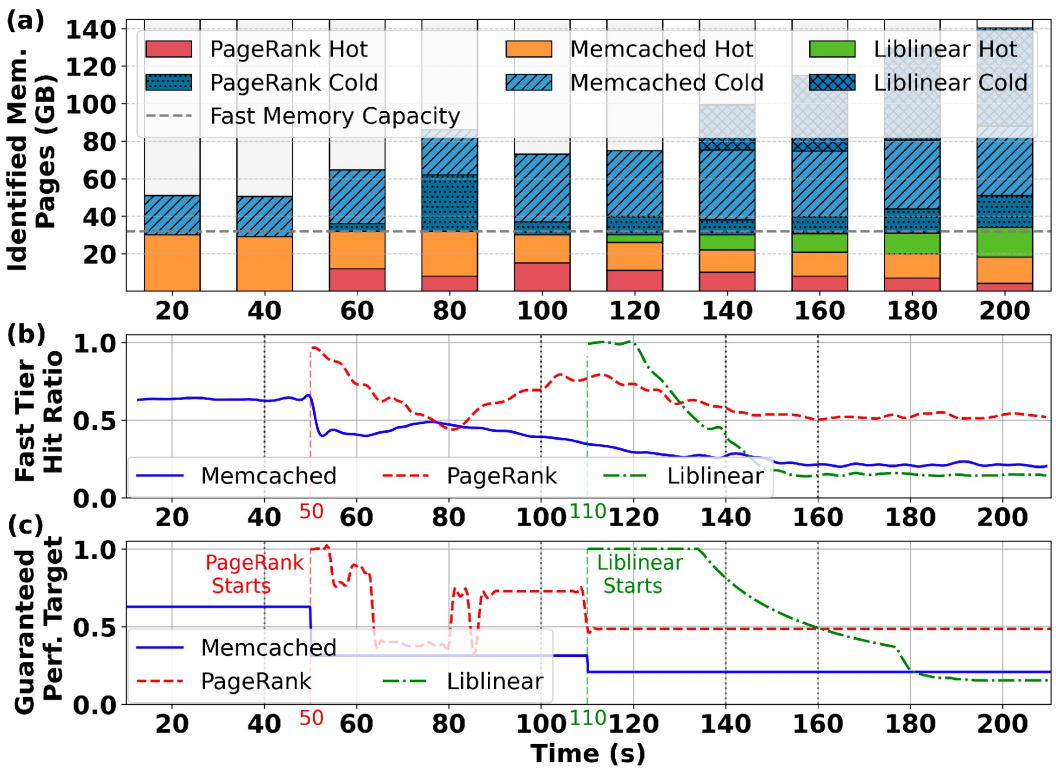


Migration performance (higher is better)

Main results:

1. Vulcan speeds up migration by up to 4×
2. Reduces TLB shootdown
3. Effective for both read- and write-intensive loads

Real-World Applications Study



Dynamic memory allocation and measurement of memory tiering performance of co-located workloads.

App	Workload	RSS
Memcached	In-memory database engine using YCSB-C	51 GB
PageRank	Compute the PageRank score of Web pages	42 GB
Liblinear	Linear classification of KDD12 dataset [33]	69 GB

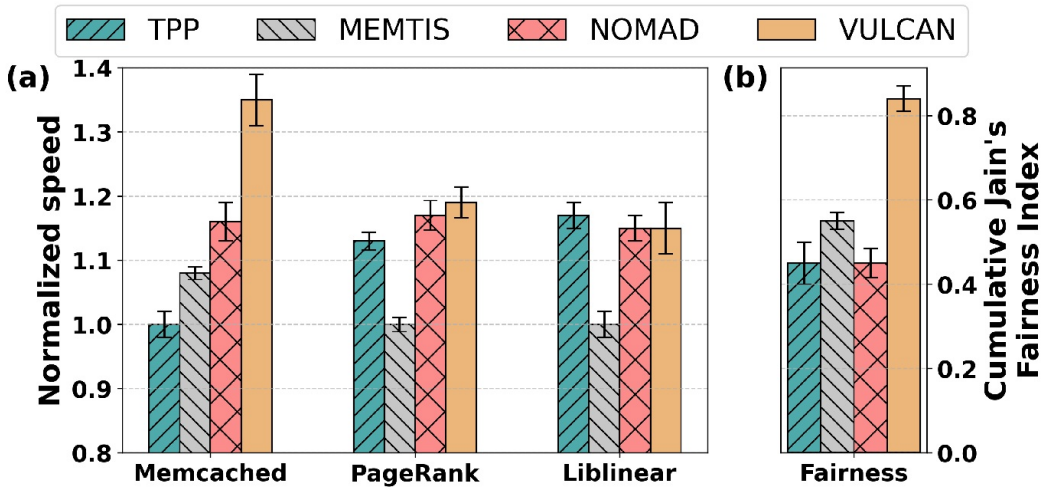
Main results:

1. Vulcan adapts fast memory allocation as workloads change
2. Maintains high performance for latency-critical applications
3. Achieves balanced resource sharing across all workloads

Real-World Applications Study

FTHR-weighted Cumulative Jain's Fairness Index (CFI)

$$\mathrm{CFI} = \left(\sum_{i=1}^{N} X_i\right)^2 \Big/ \left(N \cdot \sum_{i=1}^{N} X_i^2\right)$$



Performance and fairness comparisons (higher is better).

Main results:

1. Proposed CFI: a new fairness metric for dynamic multi-tenant memory allocation
2. Vulcan improves average performance by 12.4% and fairness by 75.3% over baselines
3. Delivers superior resource sharing and performance across diverse workloads

Conclusions

- Multi-tenant environments demand both fair and efficient tiered memory management.
- Vulcan introduces workload-aware migration and the novel CFI metric to optimize both performance and fairness.
- Extensive evaluation shows Vulcan consistently outperforms state-of-the-art solutions.
- Vulcan achieves up to 12.4% higher performance and 75.3% better fairness across diverse workloads.

Thank you!

Any questions?

Email: tangwd1@chinatelecom.cn