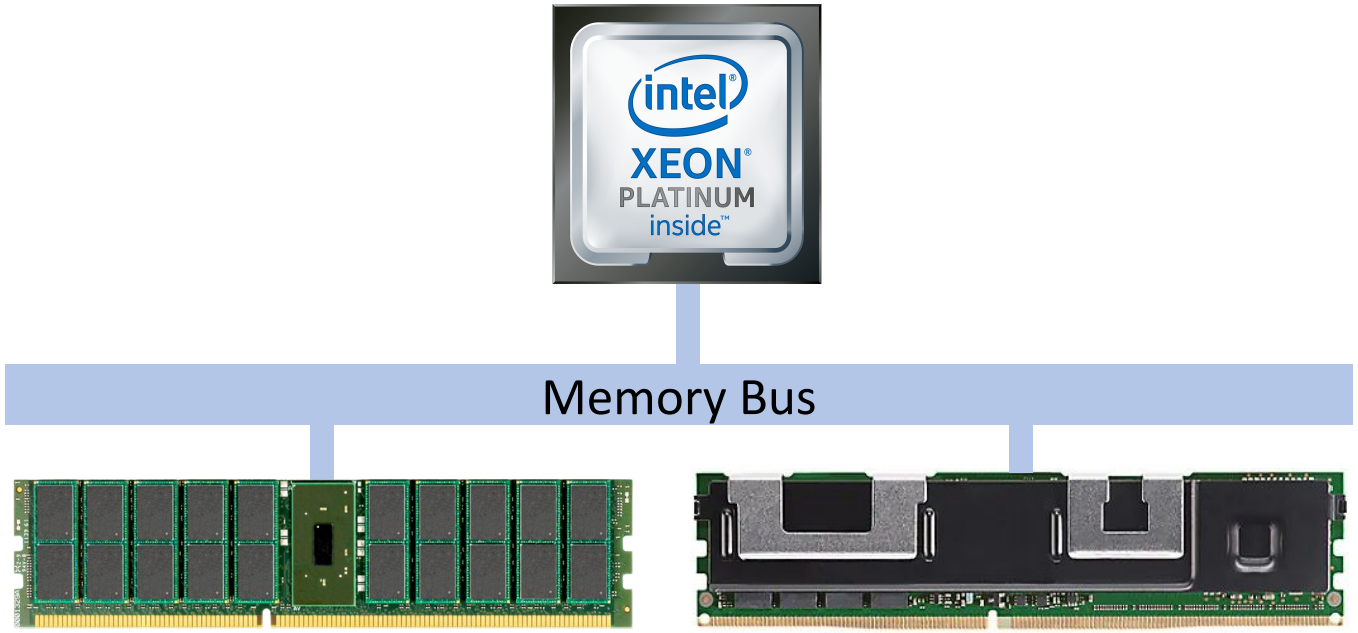
# **HeMem**: Scalable Tiered Memory Management for Big Data Applications and Real NVM

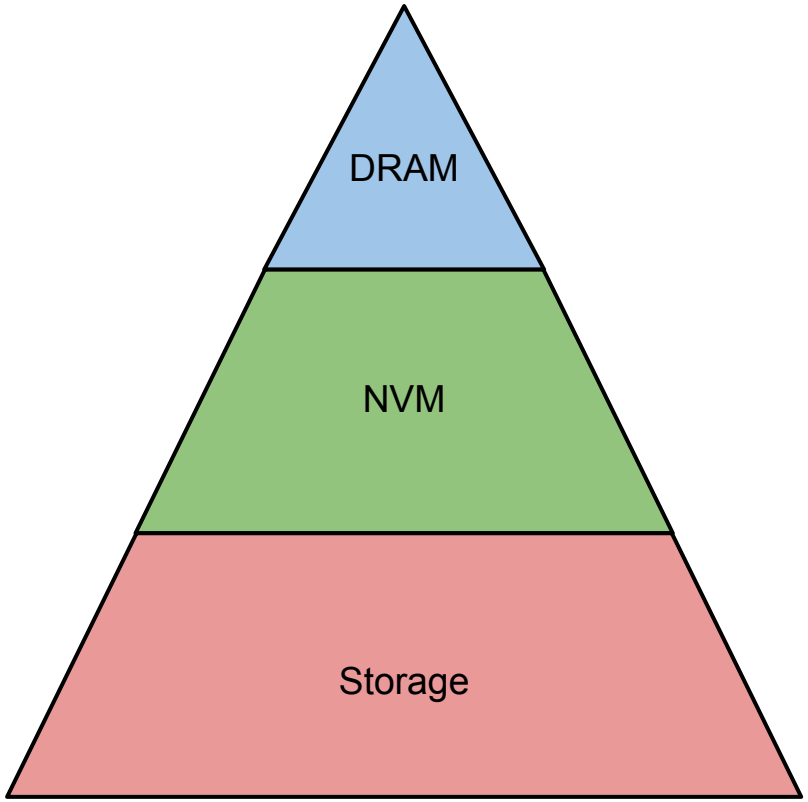
**Amanda Raybuck**, Tim Stamler, Wei Zhang, Mattan Erez, and Simon Peter



# DRAM + NVM tiered memory

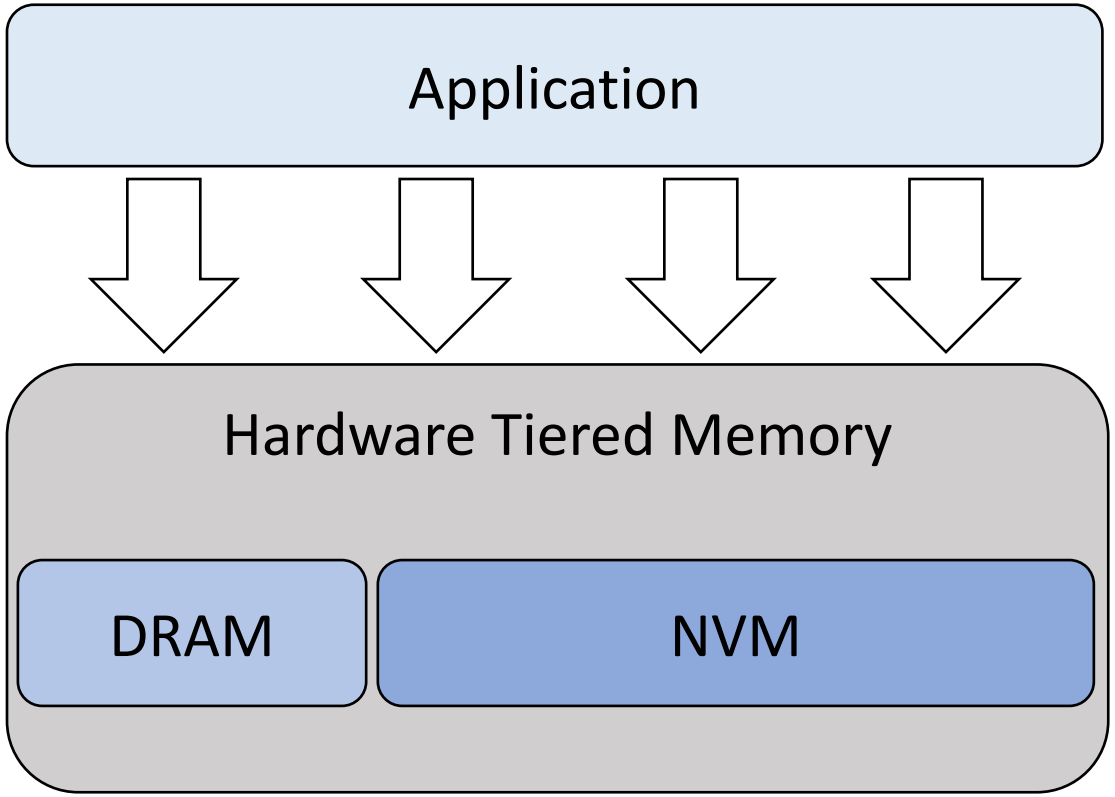



DRAM NVM




- 8x capacity
- 2x latency
- Asymmetric read/write bandwidth
- High overhead for small accesses

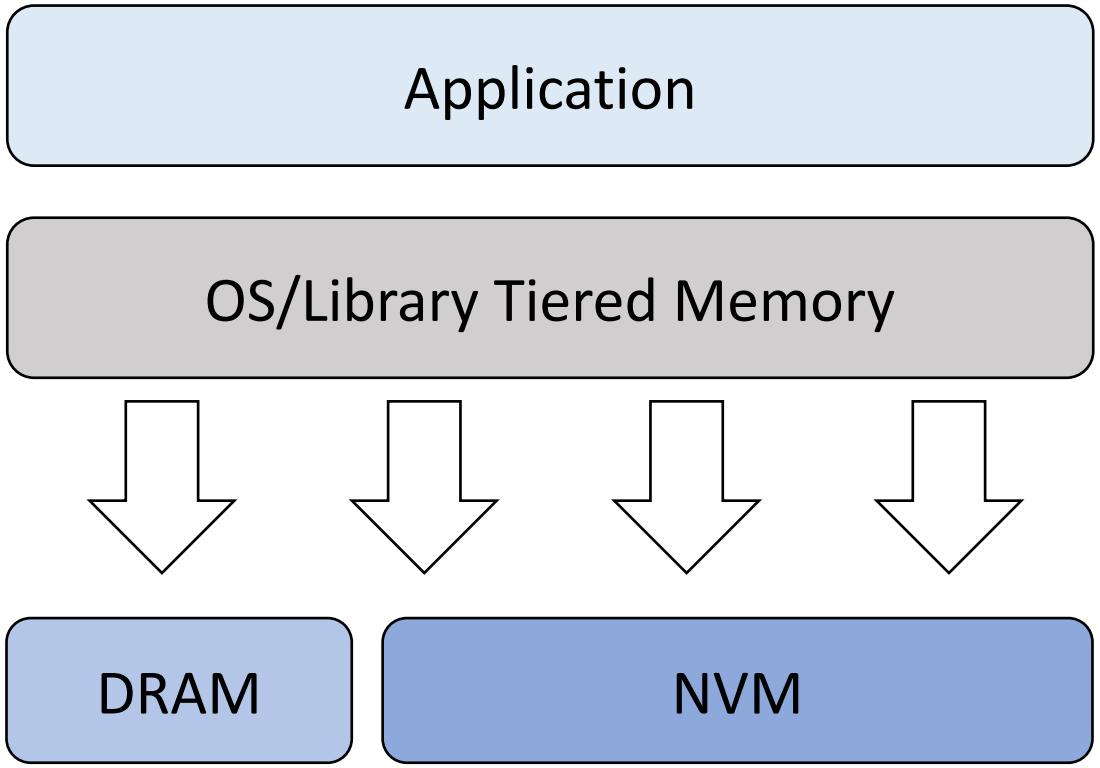
# Hardware tiered memory




Example: Intel Optane Memory Mode

- ✓ No OS support needed
- ✓ Low overhead
- ✗ No visibility into apps
- ✗ Limited to simple management techniques

# Existing software tiered memory




Examples: HeteroOS [ISCA '17], Nimble Page Management [ASPLOS '19]

✓ Insights into applications

✓ Supports complex policies

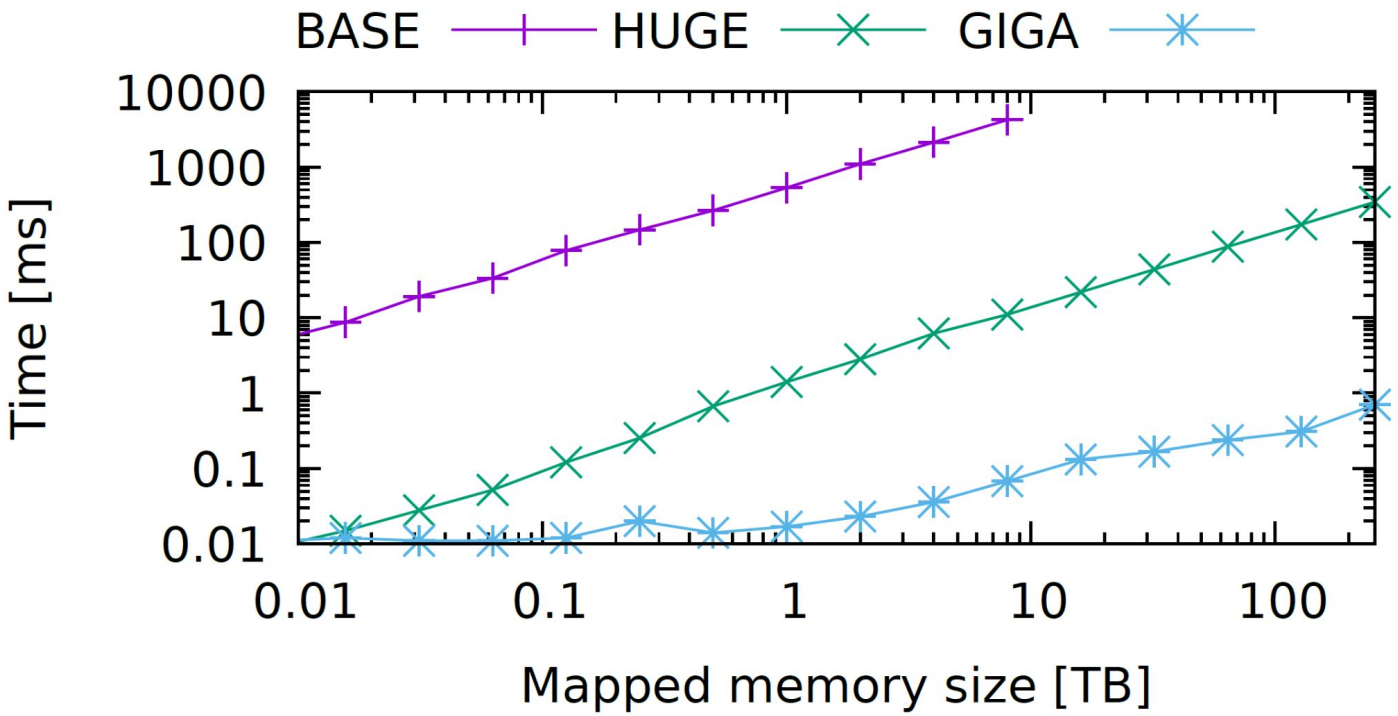
Evaluated only on emulated NVM:

✗ Does not scale to NVM capacity
- Due to page table overheads

✗ No support for asymmetric read/write bandwidth

✗ Limited flexibility

# Why not access/dirty bits?




- Not scalable
- Takes seconds to scan large memories with base pages
- Overhead of TLB shootdowns to clear bits

# HeMem:

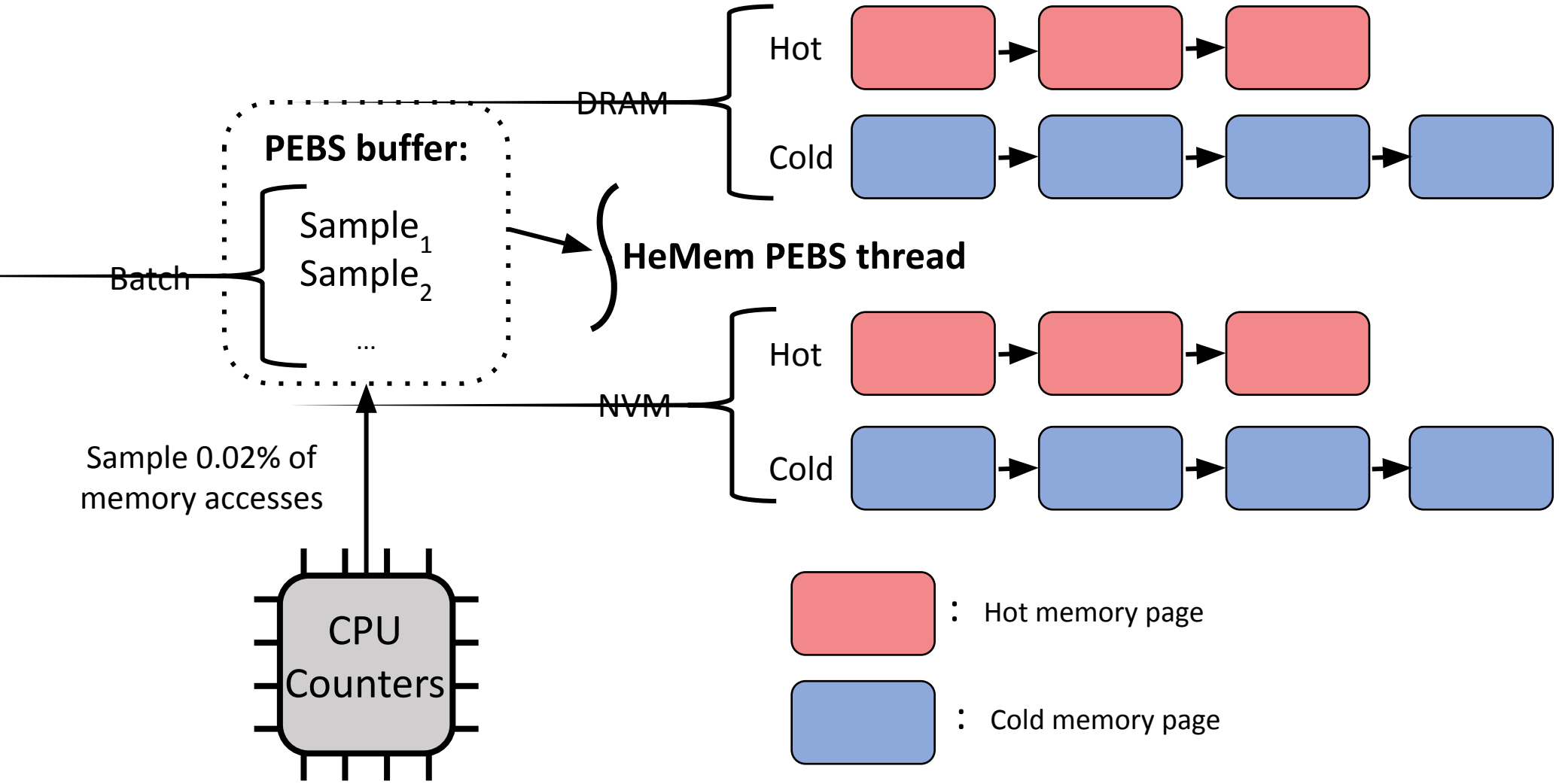
Scalable software tiered memory management system designed for real NVM

- **Design principles:**
  - Asynchronous memory access sampling with CPU performance counters
  - Asynchronous memory migration with DMA offload
  - Data scalability awareness
  - Focus on asymmetric NVM bandwidth
  - Flexibility

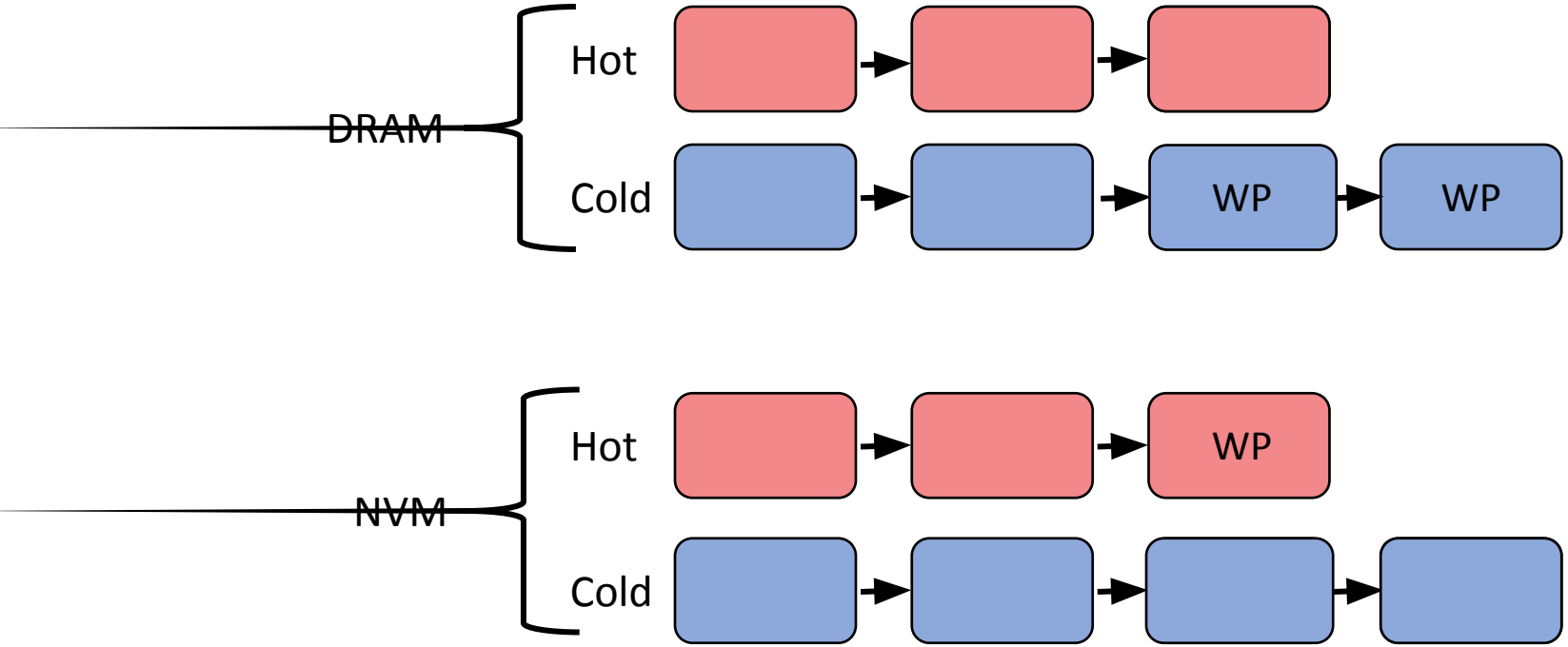
# PEBS memory access sampling

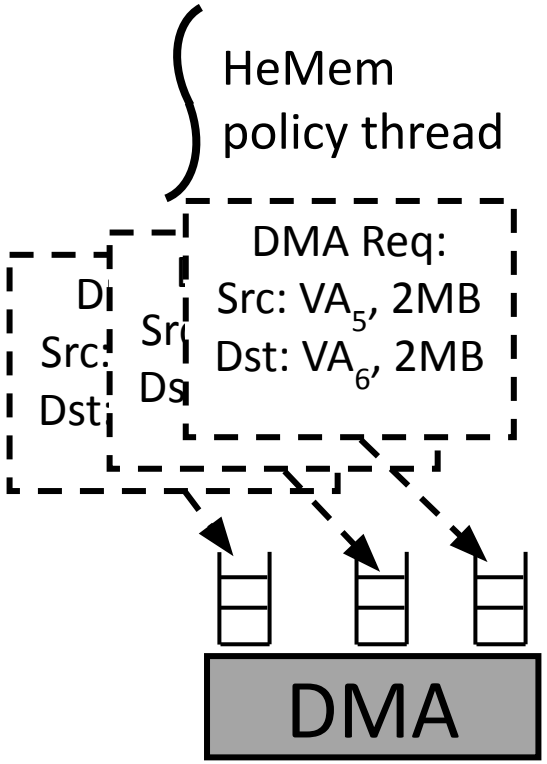
- PEBS: processor event-based sampling
  - Supported in modern Intel processors
- Processor records samples of load/store virtual memory address
  - Records are stored in a memory buffer
- We measure DRAM loads, NVM loads, and all stores
  - Instead of using page table access/dirty bits
- Sampling 0.02% of all memory accesses provides sufficient fidelity

# Asynchronous hot/cold classification



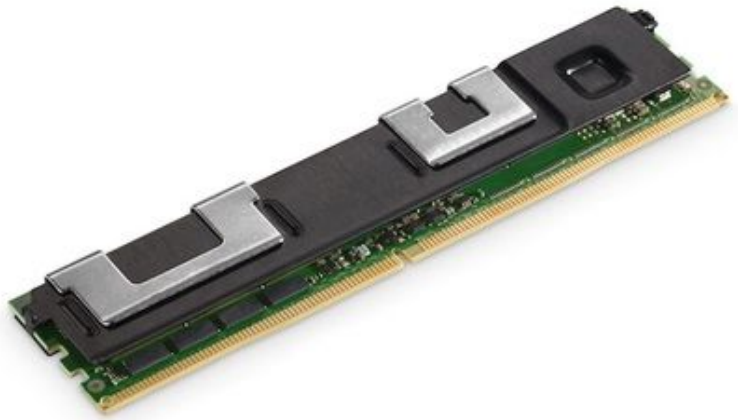
# Asynchronous memory migration



# Optimize for real NVM

- Limit writes to NVM to avoid write bandwidth bottleneck
  - Migrate and keep frequently written pages to DRAM
- Keep small objects in DRAM
  - Avoid the small random reads from NVM that suffer overheads
  - Small, ephemeral objects remain in DRAM



# Flexible user space mechanisms

- HeMem is implemented as a user-level library
  - Can be modified to better suit applications
  - Can more closely integrate with managed runtimes to further optimize
  - Userfaultfd for handling of page and write-protection faults
- Monitors application allocations and access patterns with low overhead
  - Intercepts mmap calls to learn size of allocations
  - PEBS for access patterns
- Works with unmodified applications

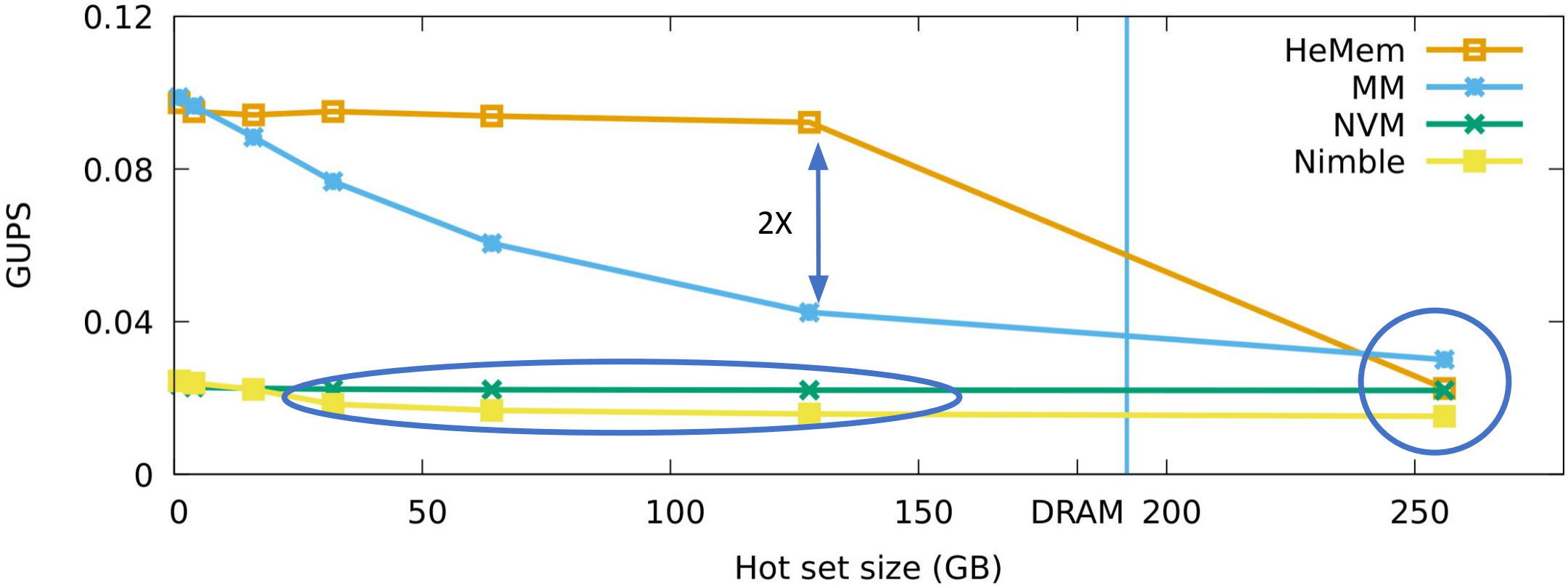
# Evaluation

# Evaluation setup

- Cascade Lake-SP w/ 24 cores, 192 GB DRAM, 768 GB NVM
  - All DIMMs populated, leveraging all 6 memory channels
- Comparisons:
  - Intel Memory Mode
  - Linux nimble tiered memory management [ASPLOS ‘19]

# Hot set identification

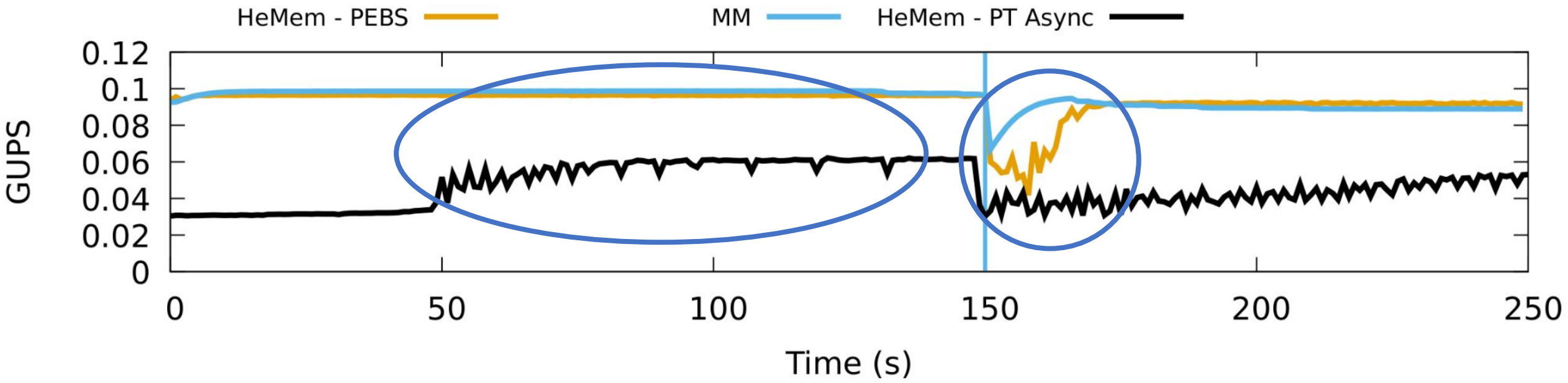
GUPS microbenchmark with hot set (512 GB working set)
8 byte accesses, non-contiguous hot set



# Dynamic hot set identification

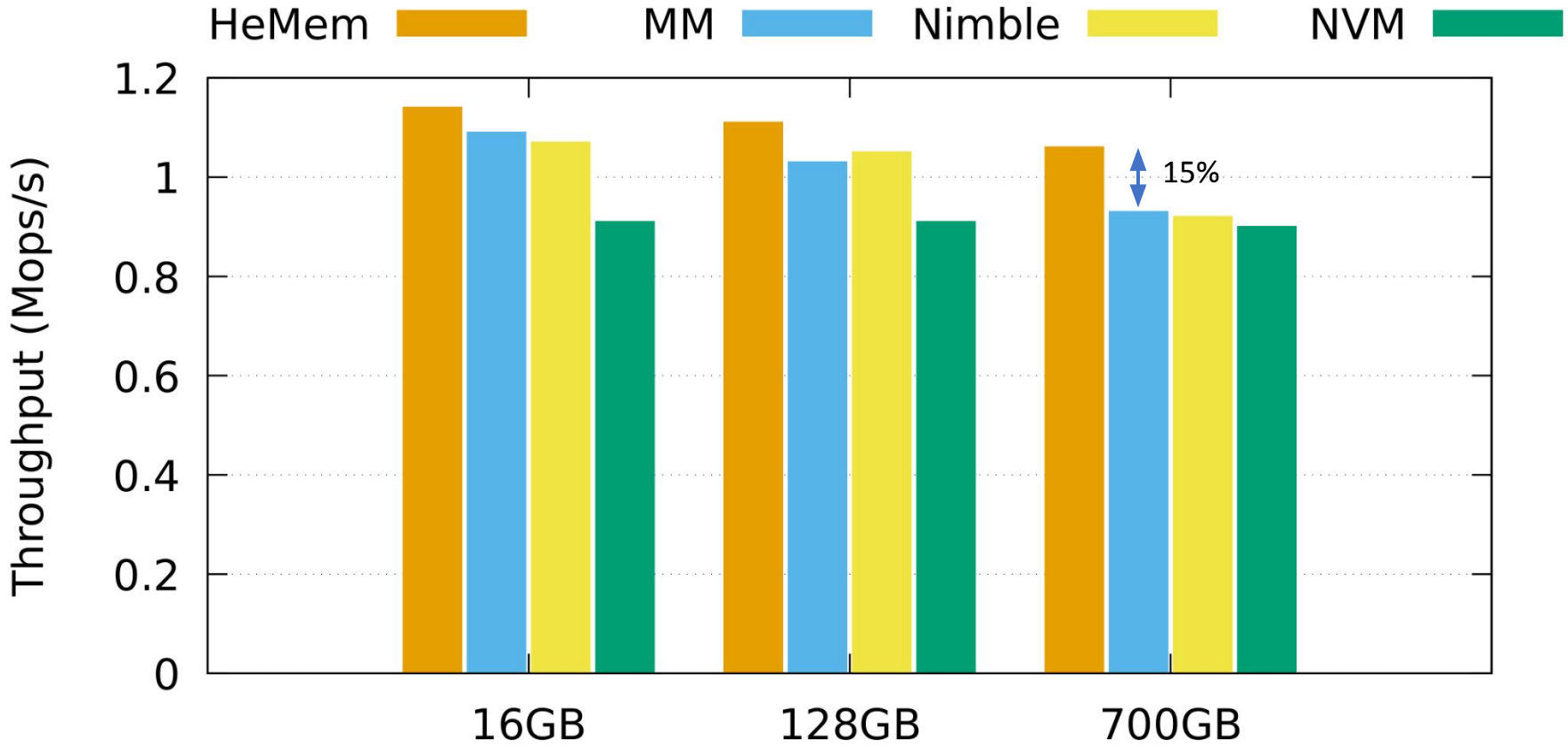
GUPS with a 512 GB working set and a 16 GB hot set

At time t=150, shift hot set over by 4 GB

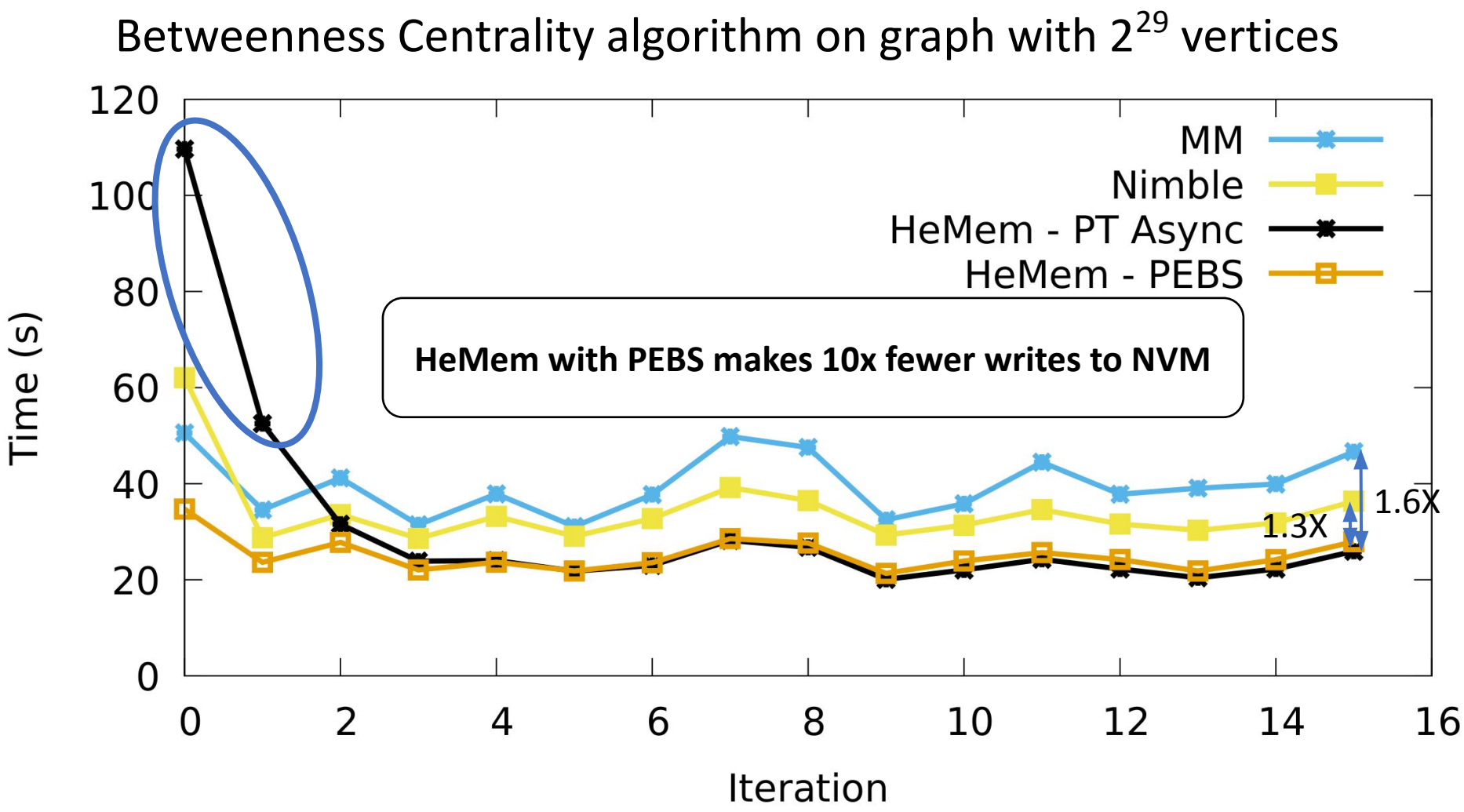


# FlexKVS key-value store throughput

4KB value size, 90% GET, 10% SET, 20% hot keys accessed 90% of time



# GAPbs execution time



# Summary

- Tiered memory systems need to support real NVM
  - Need to scale to large capacities
  - Need to support unique NVM performance features

- **HeMem:** redesign of tiered memory management with real NVM
  - Sampling-based memory access monitoring without page tables
  - Asynchronous memory migration in batches with DMA offload
  - Accurately distinguishes hot from cold memory

- Up to 1.6x GAPbs speedup, 2x GUPS, 10x fewer NVM writes

Source code: https://bitbucket.org/ajaustin/hemem/src/sosp-submission/