Good morning!

CS 744: HEMEM

Shivaram Venkataraman

Spring 2024

ADMINISTRIVIA

Last research paper!

Midterm 2, April 25th

- Papers from SCOPE to HeMem
- Similar format as first midterm
- Details on Piazza

Regrades M1 pending → Canvas today / tomm

TA reviews, discussion

MOTIVATION: MEMORY DEMANDS

Large memory applications

- Key Value Stores
- ML, Graph analytics?

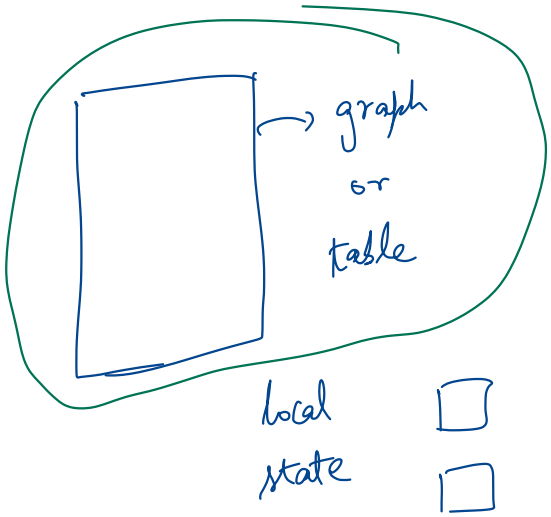
→ Benchmarks

SILO

PageRank, Connected Components

Usage pattern

- Bimodal allocations
- Allocate a large region that lives throughout
- Small ephemeral allocations



INTEL OPTANE DC NVM

→ NV storage

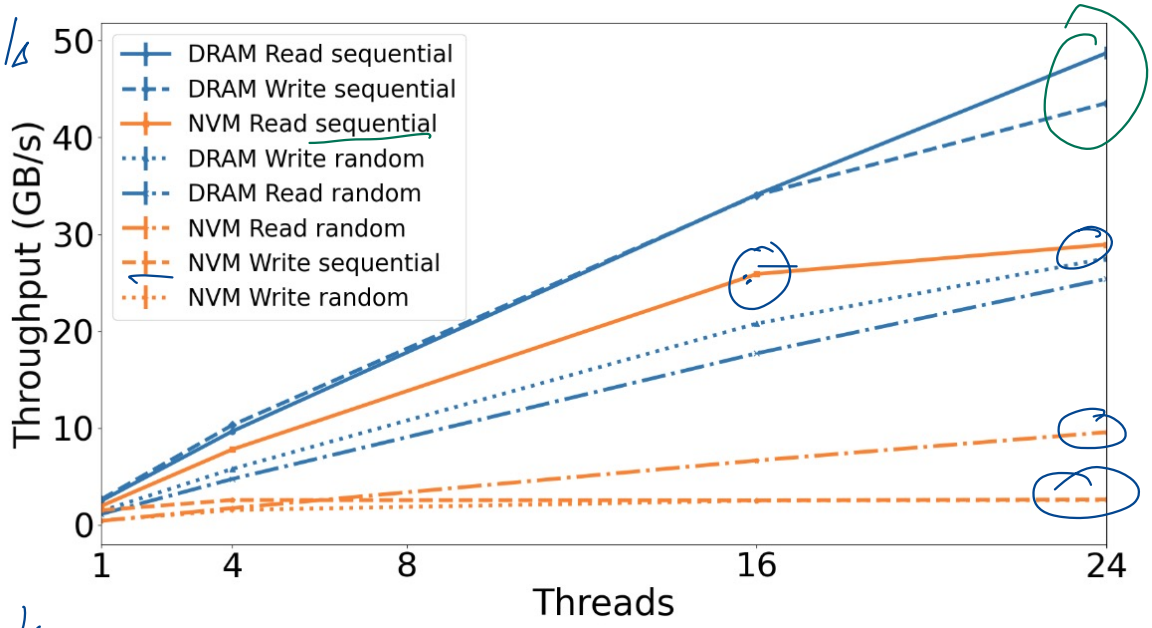
→ DRAM extension | → This paper

DRAM

- read / write are comparable at ~40-50 GB/s
- random is slower than sequential

NVM

- Read is good! ~ 25 GB/s
- Writes are much slower! ~ 2 GB/s
- Random reads ~ 10 GB/s



MEMORY MODE (MM)

① No visibility of app access patterns

② Conflicts → frequent DRAM misses

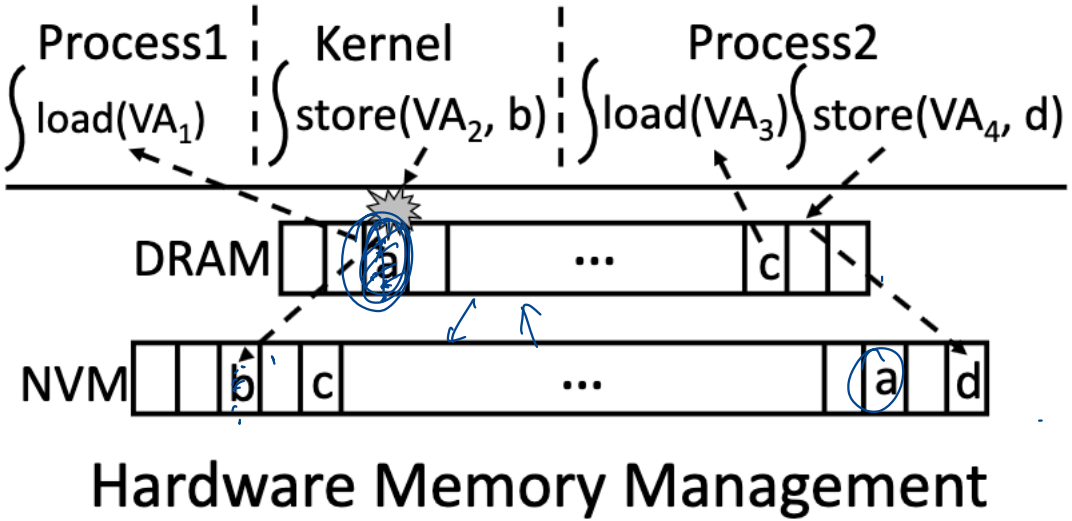
Transparent to apps

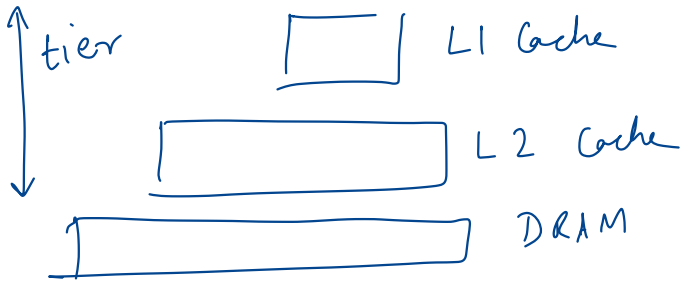
DRAM is a cache!

Hardware managed

Cache-line size (64 B)

↳ fine granular





HEMEM: GOALS

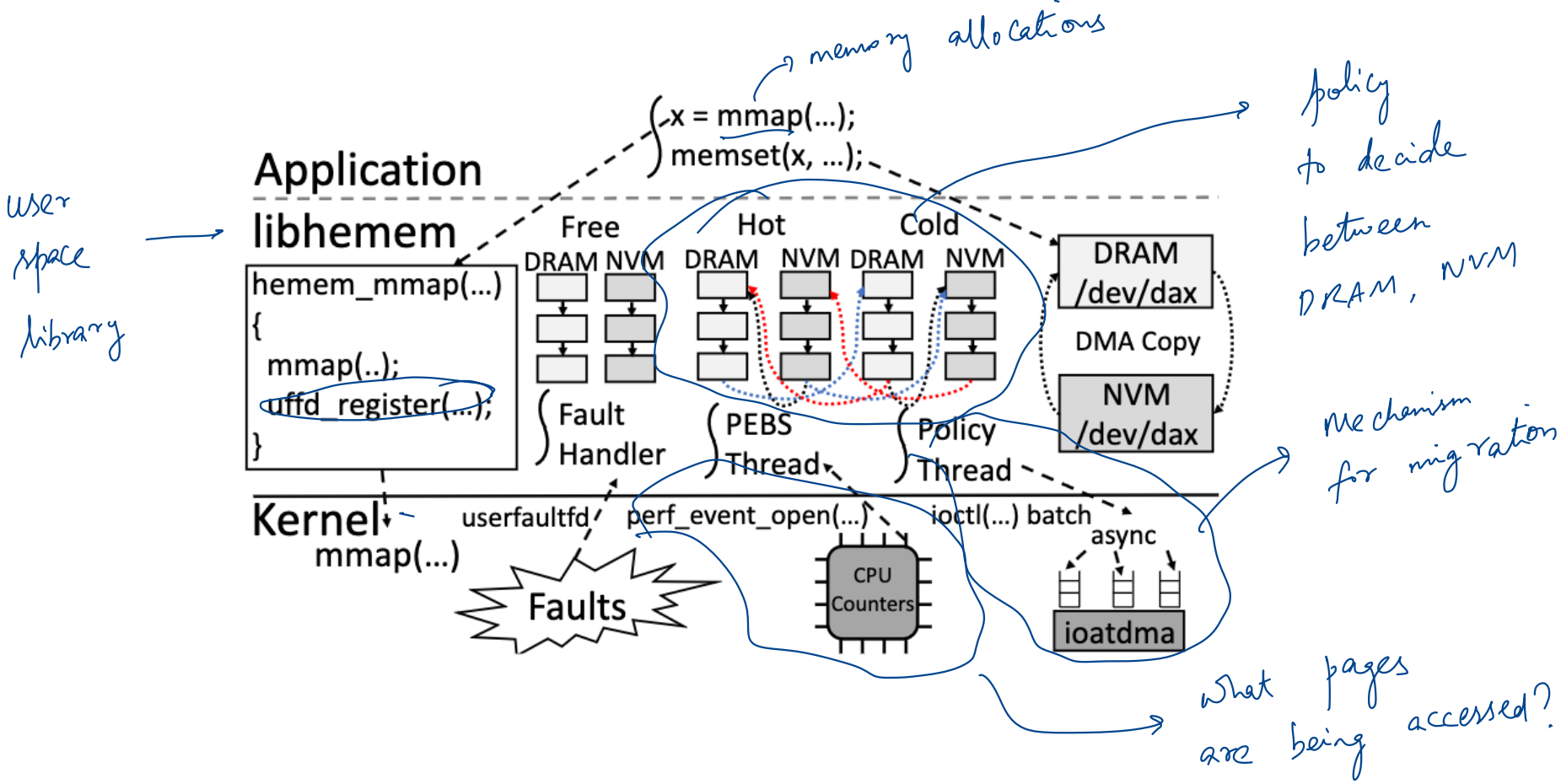
Asynchronous memory management → apps don't block when accessing memory

Handle asymmetric NVM bandwidth → write BW limits of NVM

Asynchronous memory access sampling

Flexibility – per-application policies → user / administrator control

HEMEM: DESIGN



HEMEM POLICIES

heuristic:

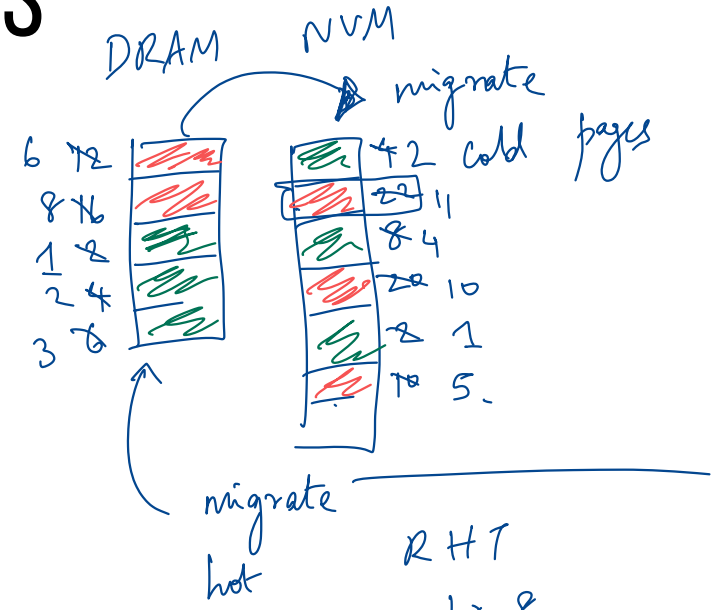
accessed frequently

Track hot, cold pages for NVM and DRAM

Separately track read, write hot using thresholds

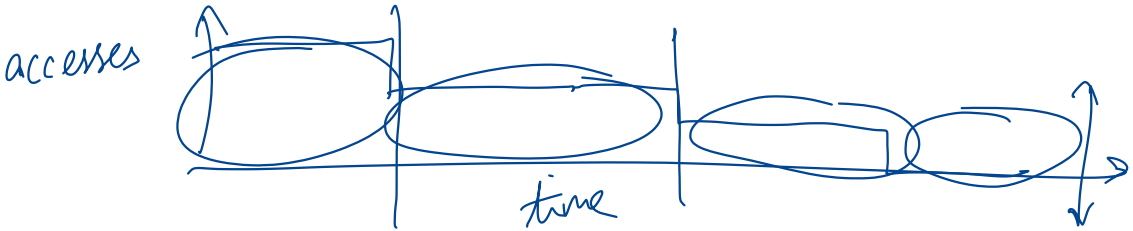
two counters for each page

one read, write



Periodic cooling

- Halve all ~~pages~~ Counters once one page reaches threshold ?! -



HEMEM POLICIES

Allocation → small allocations (< 1G)
↳ skip HeMem altogether

- Use DRAM if available
- When free DRAM is less than 1G → for migrations
  - Allocate new pages on NVM

Migration → background or asynchronously

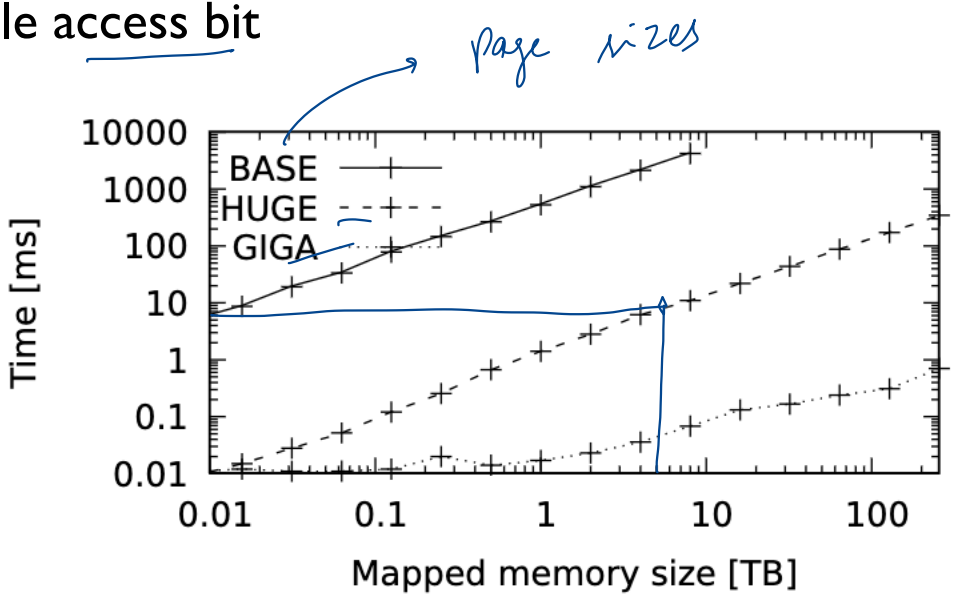
- Migrate cold DRAM pages to NVM
  hot NVM pages to DRAM
- Prefer write-heavy pages. Why?
  ↳ NVM write BW is low

MEASURING MEMORY ACCESS

Challenge: Track which page has been used and how often

Periodically

Scan the page table access bit



MEASURING MEMORY ACCESS

Processor event based sampling (PEBS)

Processor makes a note when perf counter overflows

MEM_LOAD_RETIRED, MEM_INST_RETIRED etc.

Periodically getting triggered

triggers an event, has address at that point in time

↳ fine grained

Sampling frequency trade-off? → very rarely

↳ inaccurate hot pages

very frequently

↳ overhead

Granularity of tracking

↳ Page granularity

Coarse → migrations efficient

Fine → hot set appropriately

MIGRATION MECHANISM

Background thread to migrate from DRAM from/to NVM

- Mark ~~thread~~ page as write protected → avoids writes while page is being migrated.
  - ↳ Blocking
- Use DMA engine to do the copy (batch these calls)
  - ↳ avoids overheads
  - → here are 4 pages to migrate → all of them in parallel

SUMMARY

New hardware support to extend DRAM

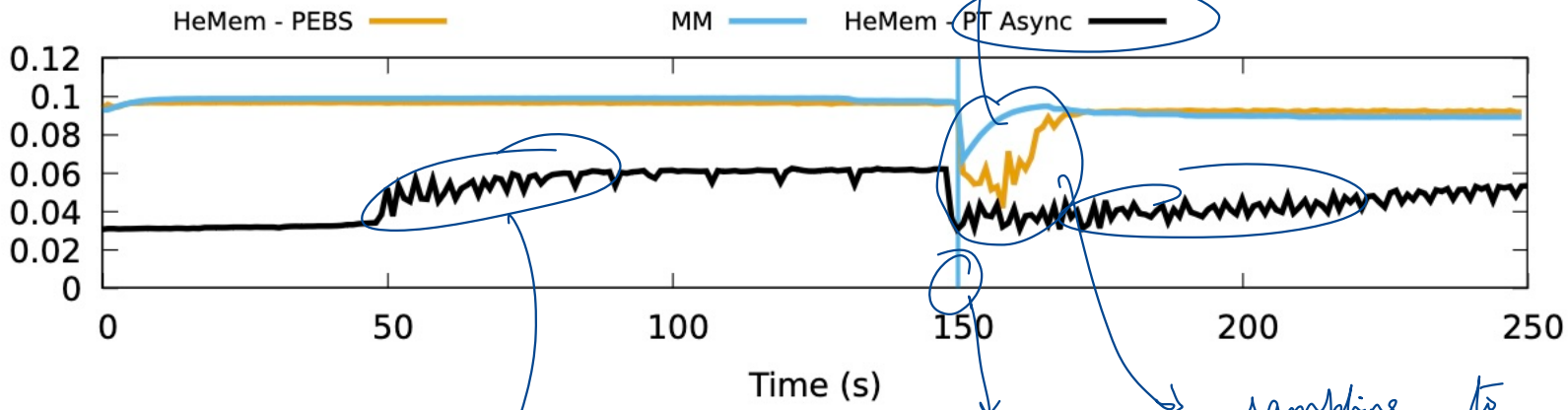
Need for systems to manage migrations

HeMem:

- PEBS based memory access tracking
- Hot, Cold lists for DRAM, NVM
- Background migration

DISCUSSION

https://forms.gle/Gh5gaCmhCXUmjG7R9



MM → Hardware mode recovers better

Page table scanning slows down application

blocking

sampling to detect hot pages quickly catches up.

What are ways in which a memory tiering system like HeMem is similar to Marius/BagPipe and in what ways is it different?

Marius / BagPipe

↳ you know what will be accessed in future

↳ even more application knowledge

→ Data access patterns → both hot and cold set

→ Async

HeMem

↳ recent read/writes → read/writes in near future

GENERACITY

→ Infer both hot/cold set

→ Async

NEXT STEPS

Midterm 2 next!