6.5810: TMO

Adam Belay <abelay@mit.edu>

Logistics

- Project meetings tomorrow

Why search for alternatives to local RAM?

- RAM is very fast, compared to other storage mediums
- However, it is expensive in a $/byte sense
- And its price has been volatile over time

What are some alternatives to local RAM?

- Compressed RAM (i.e., zswap)
- Solid state disks (i.e., swap)
- Stranded RAM on another machine
  - Exposed through RDMA or normal Ethernet
- Nonvolatile RAM
- Disaggregated RAM (e.g., CXL)

Problem! All these options are slower…

- How can we preserve performance?

Problem! All these options are slower…

- How can we preserve performance?
- Solution: Tiering; store cold objects in slower memory

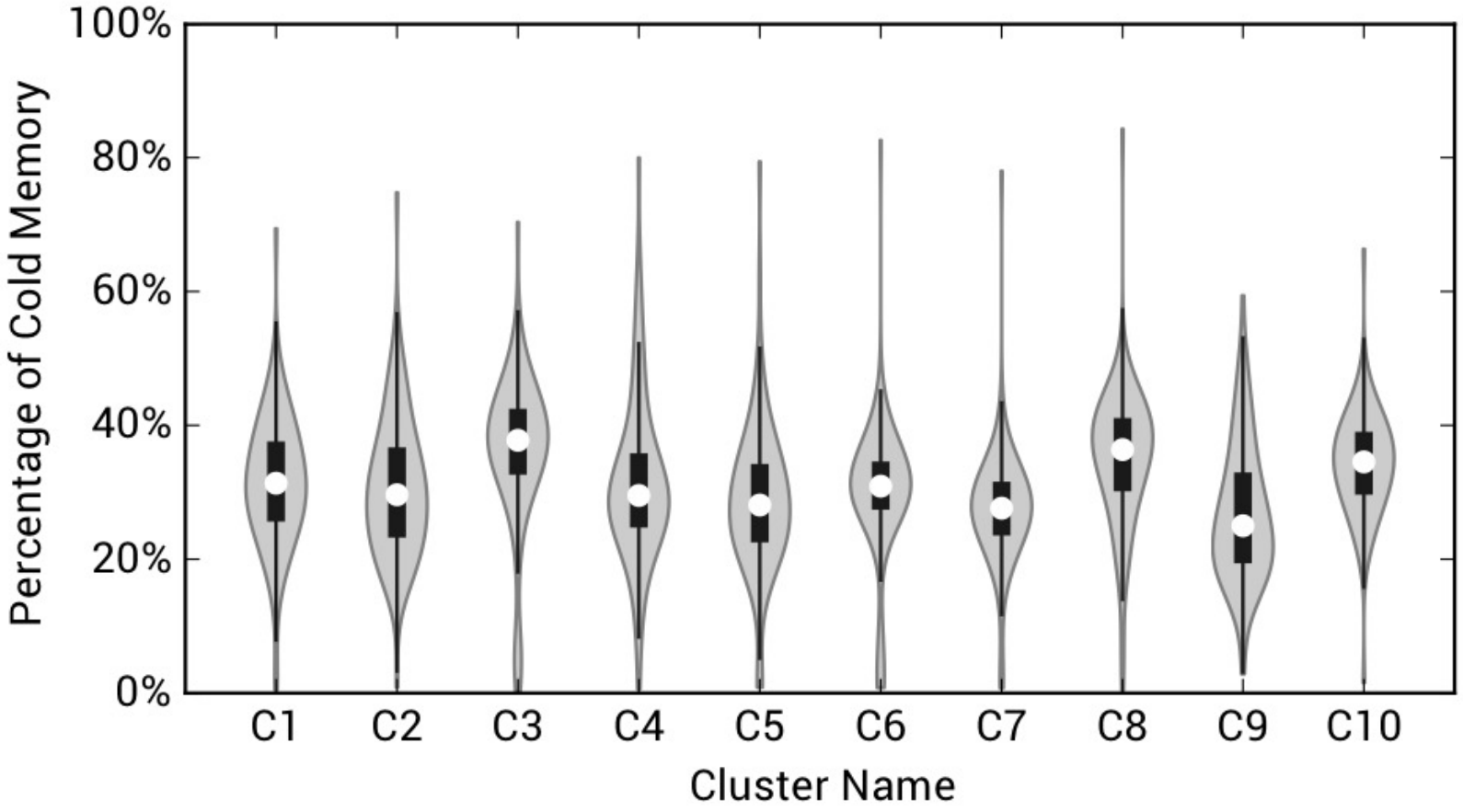
Solution: Store cold pages in slow memory

- But what is a cold page?
- Measure the *age* of the page; the time since the page was last accessed
- e.g., if page was not accessed for the last 10 seconds, it is cold
- The *promotion rate* is the rate of access to cold memory stored in a slower tier
- Assumption: A lower promotion rate should have less perf. impact

Software-defined far memory [ASPLOS'19]

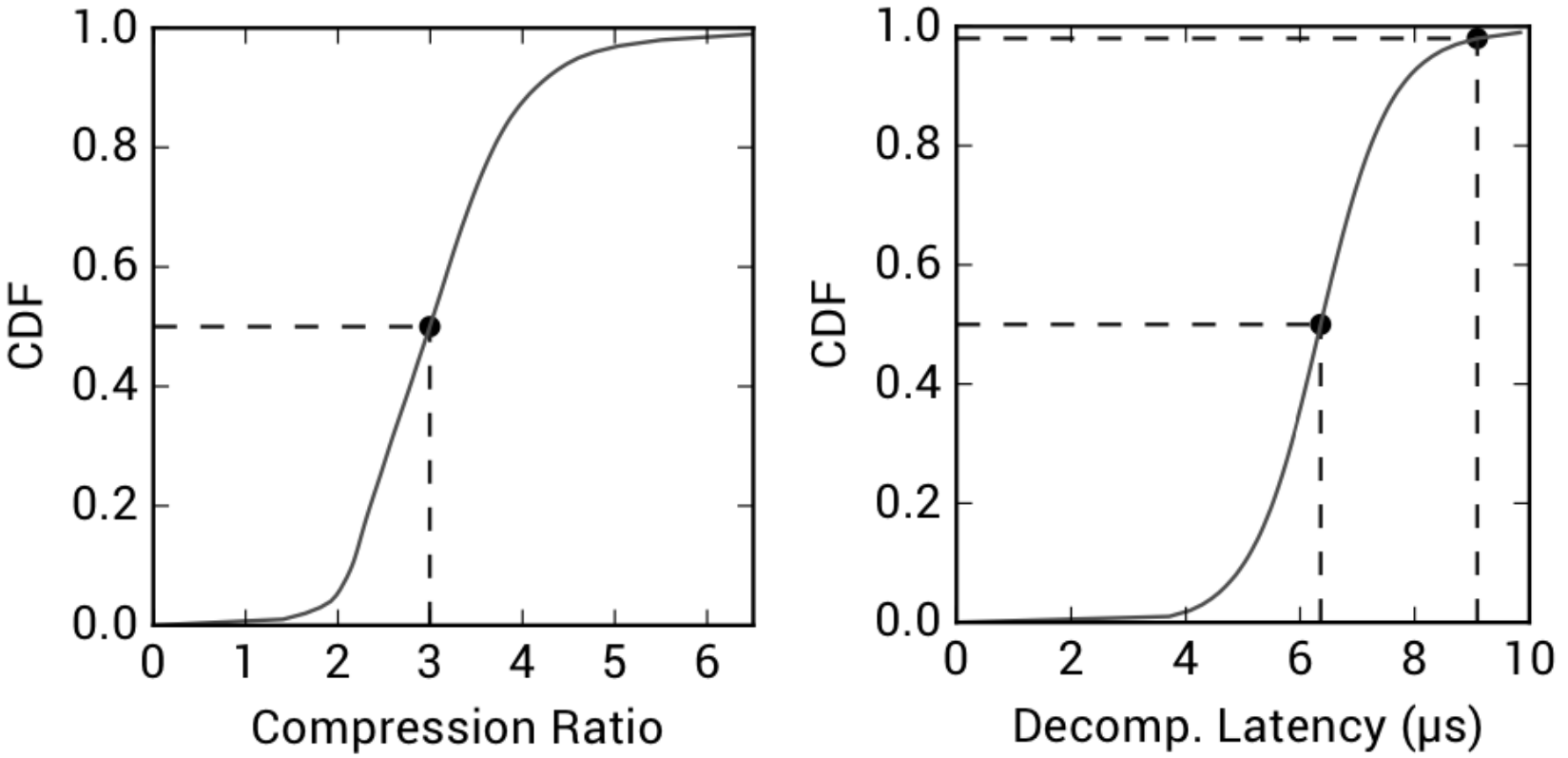
- Prior to the TMO paper, Google published their results on using zswap (i.e., memory compression) to reduce the cost of storing cold memory
- Goal: Free lunch, no performance impact on workloads
- Insight: Move cold pages proactively, before memory pressure
- Insight: Use promotion rate threshold to determine how much memory to swap

Cold memory is surprisingly abundant



Software-Defined Far Memory in Warehouse-Scale Computers [ASPLOS'19]

Opportunity for compression of cold pages



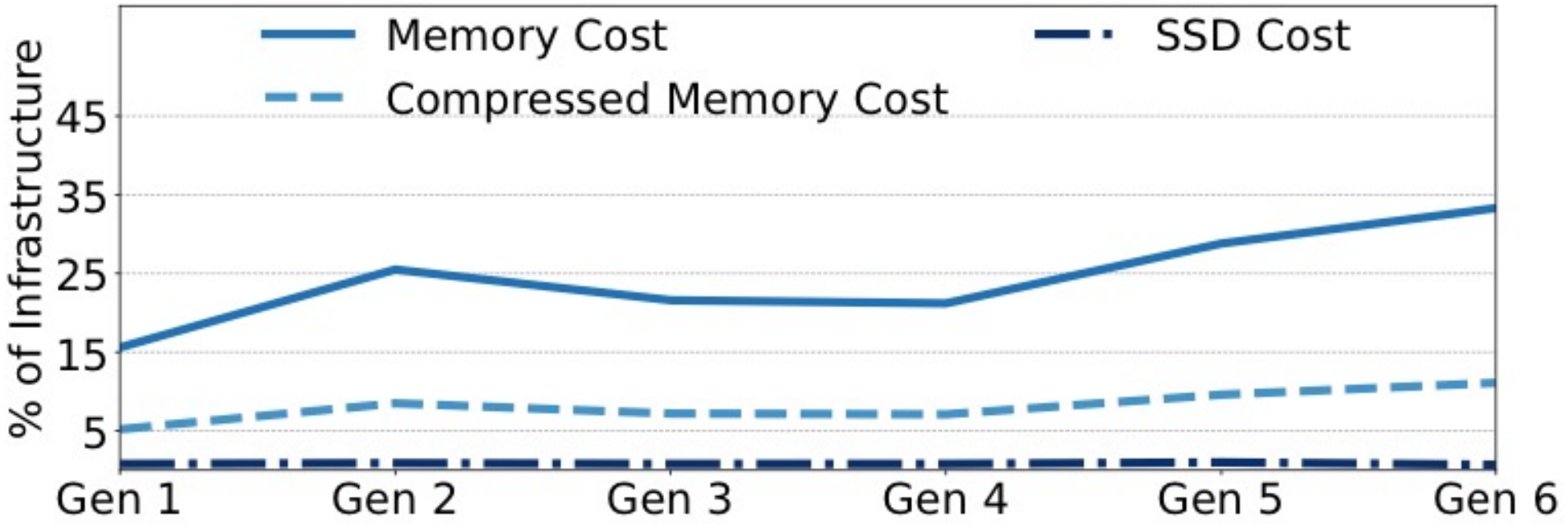
(a) Compression ratio. **(b)** Average decompression latency.

Software-Defined Far Memory in Warehouse-Scale Computers [ASPLOS'19]

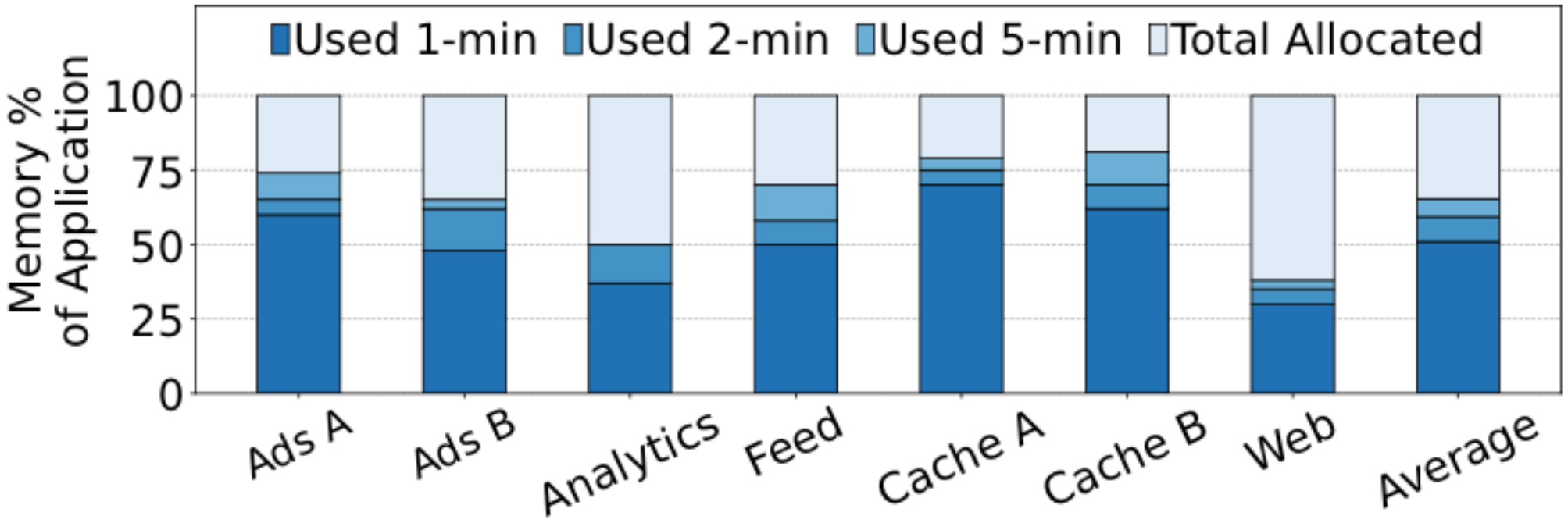
Transparent memory offload (TMO)

- Used in production in Meta's datacenters, saving 20-32% of RAM!
- Improvements over software-defined far memory
  - Supports more than a single type of far mem (both compression and flash)
  - Instead of setting a promotion rate threshold, stall time is used

Trend in memory cost

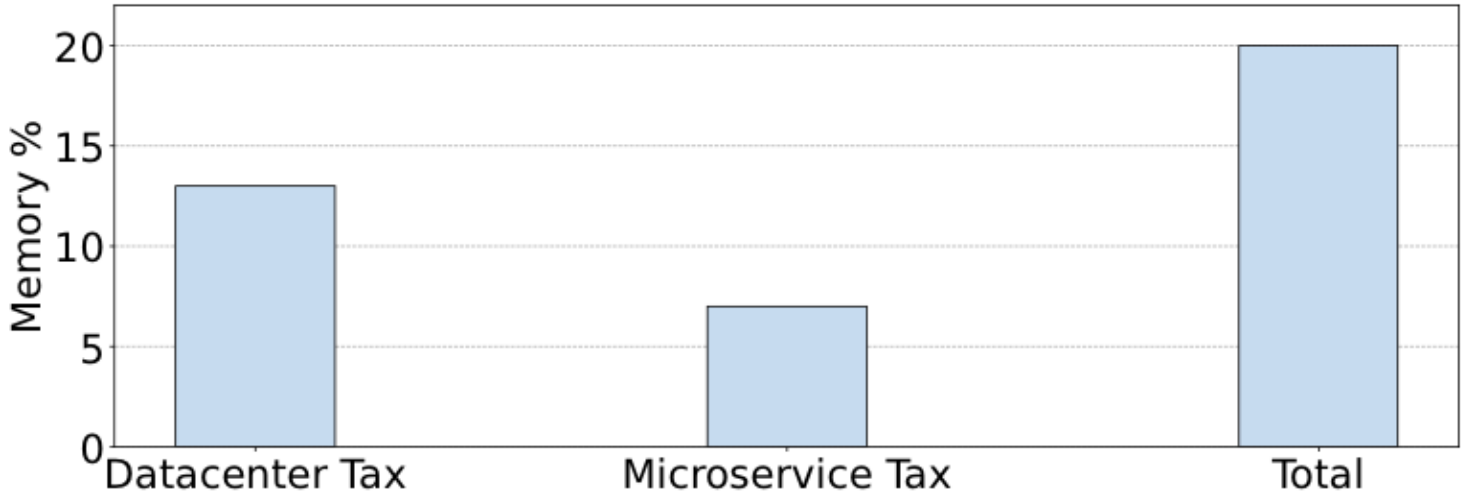


Cold memory in meta’s workloads



The memory tax

- Meta's services rely on a package of shared common infrastructure
- *Datacenter tax*: software packaging, logging, profiling, etc.
- *Microservice tax*: Sidecars (proxies like Envoy), RPC routing etc.



Linux manages two types of memory

- *Anonymous memory*: Allocating by applications, not backed by file or device
- *File-backed memory:* A memory mapping backed by a file, stored in the kernel's page cache (to prevent slow disk reads)

TMO answers two key questions

- *How much memory to offload?*
  - New pressure stall information (PSI) metric
- *What memory to offload?*
  - Balances the file cache v.s. anonymous memory
  - Considers the relative priorities of containers

How much memory to offload?

- Goal: Minimize performance slowdown
  - Major fault: Fetch page from slower memory
  - Minor fault: install page from physical memory to page table
- PSI metric: Measures the amount of time a container is stalled, blocking on major faults

Debate: Why is PSI better than promotion rate?

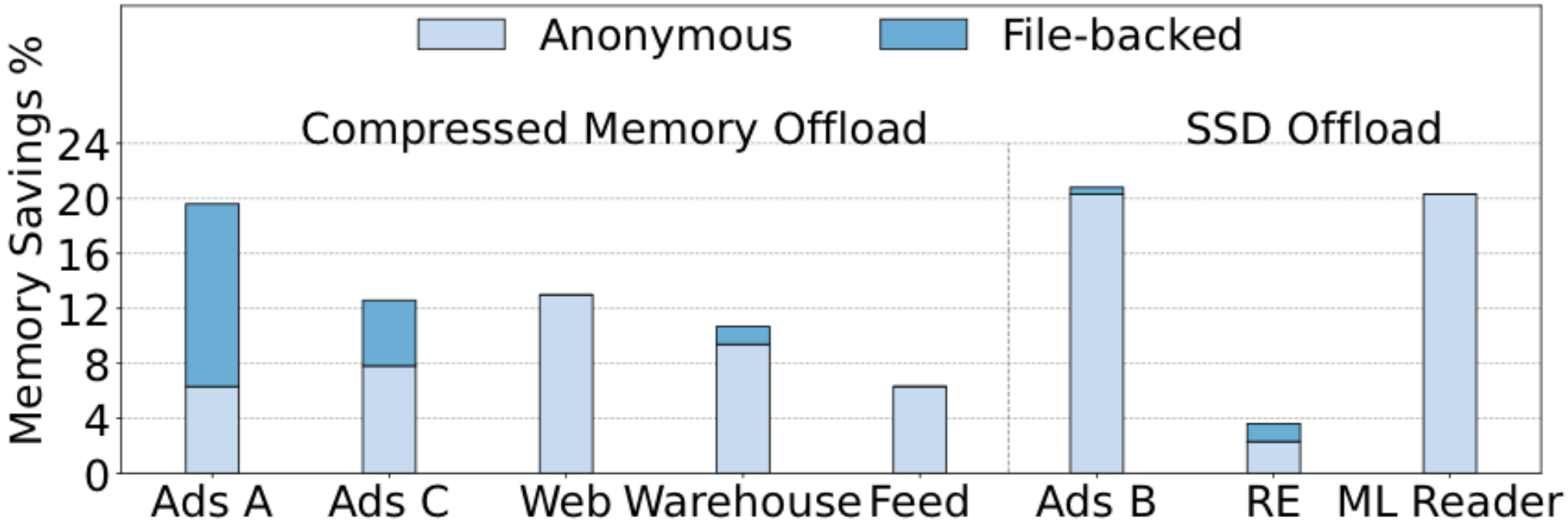
The page cache demotion imbalance issue

- Historically, disks were awfully slow
- Consequence: swapping anonymous memory is for emergencies only, focus on evicting the page cache first
- Now: Flash is fast enough, that anonymous memory swapping could be a better choice than demoting the page cache
- TMO tunes this to achieve a better balance
  - *Refault distance*: # of page faults - # of page faults when page evicted
  - If < current memory footprint, a refault occurred
  - Can detect first-time file page faults, so they don’t count toward PSI
  - Start swapping anonymous memory as soon as refaults start to occur

Q: When should we use zswap vs. swap?

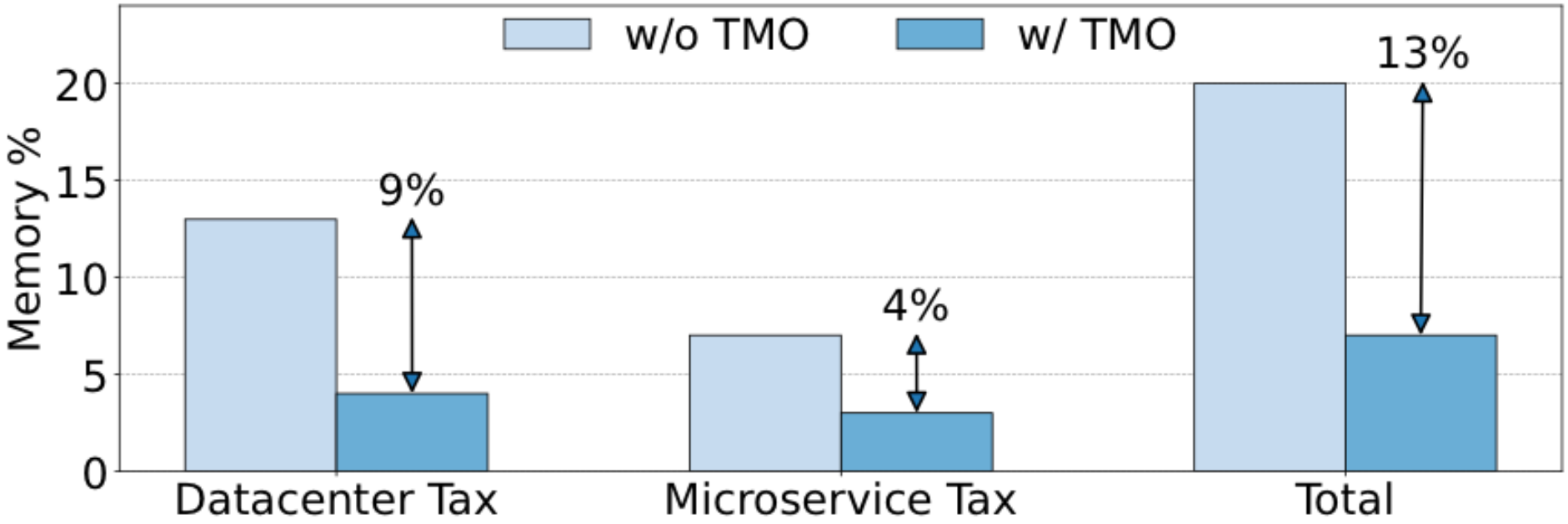
- Does TMO answer this question?

How much memory can TMO save?

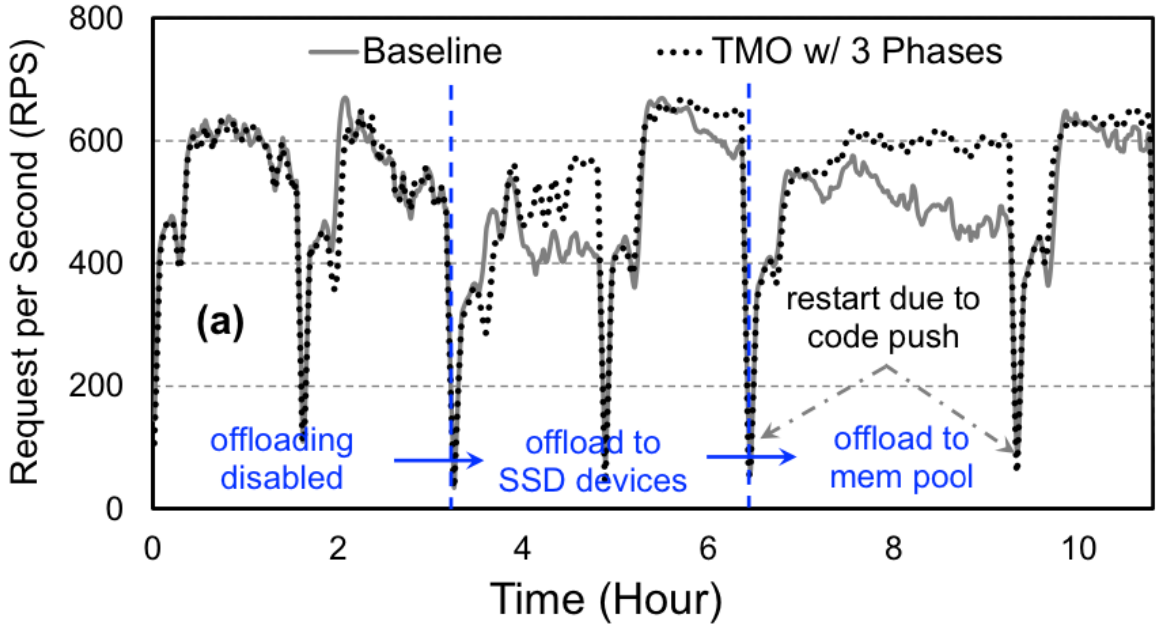


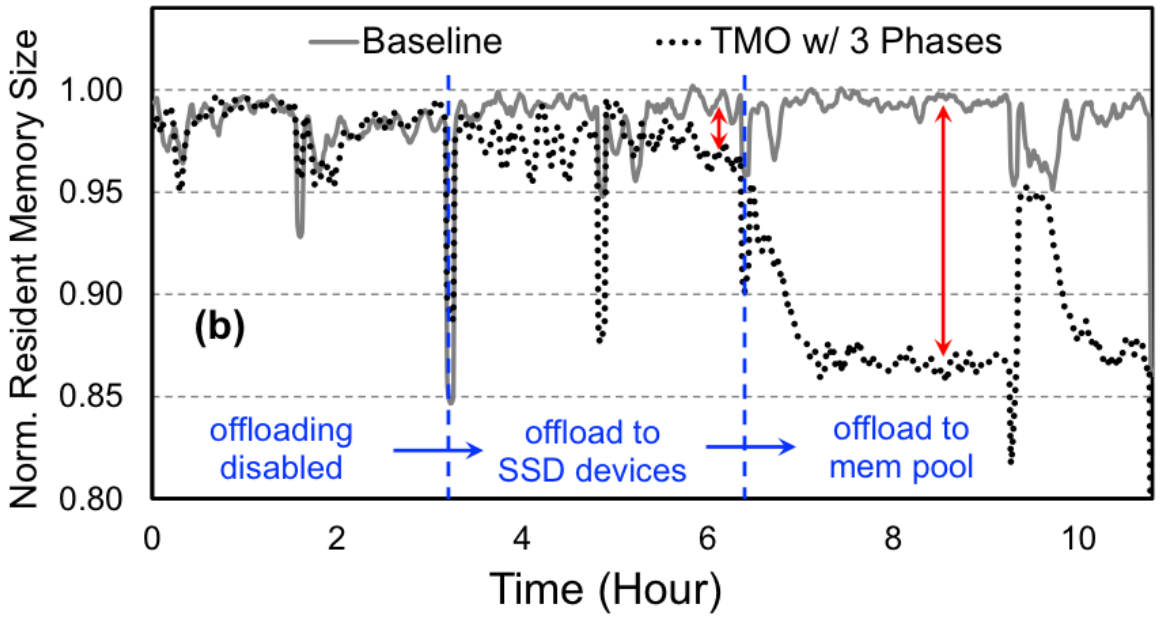
Q: Why does SSD offload appear to save more?

What about the datacenter tax?



How does TMO impact app performance?





Future challenges

- How can we support many different tiers of remote memory simultaneously?
- How can we make memory swapping mechanisms faster?
- How can we better hide the latency of major faults?
- Should we use pages at all? Does their granularity cause amplification?
- How can we modify applications to arrange their memory better for remote paging?