# CS 477:

# Advanced Operating Systems

## Virtual Memory II & TMO

# Administrivia

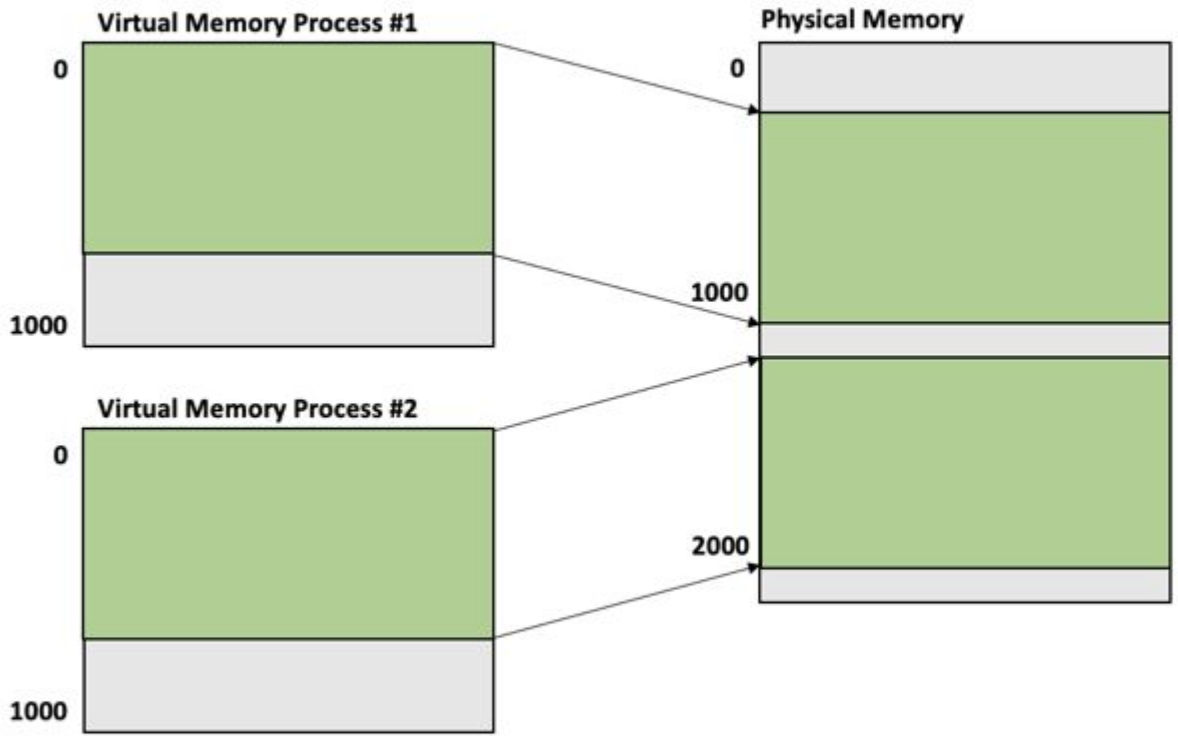
# Feedback discussion

# This week

- Virtual Memory and TLB Recall
- Memory Disaggregation
- Swapping in Virtual Memory
- TMO (Transparent Memory Offloading)

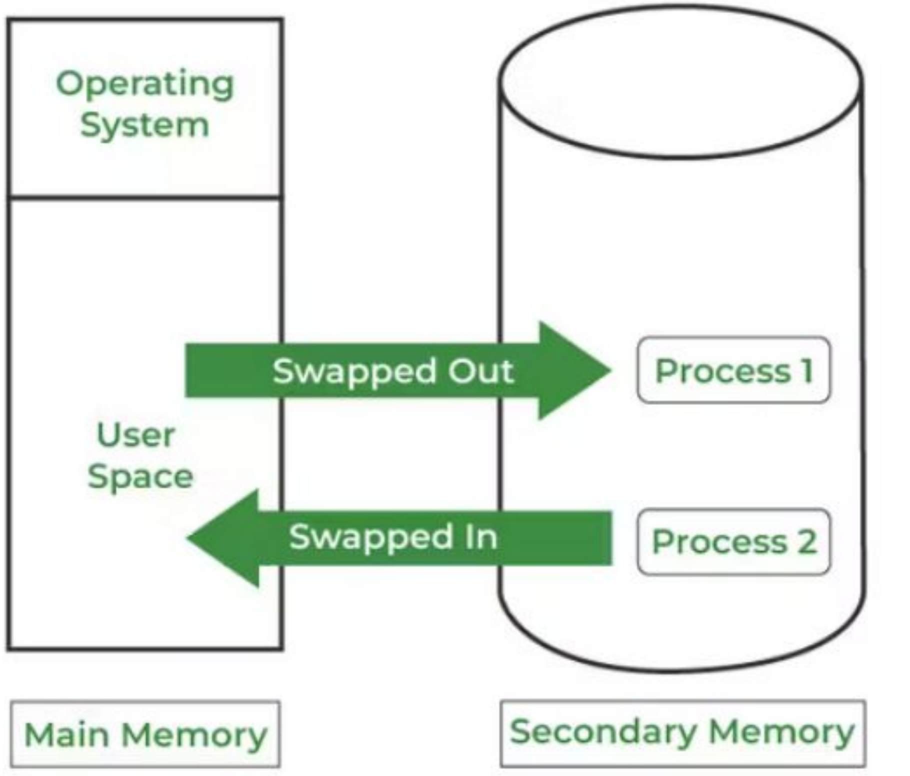
# Virtual memory recall

- Provide isolation and protection across processes
  - Each process has its own address space for the physical memory
  - Each process cannot directly access memory via physical address



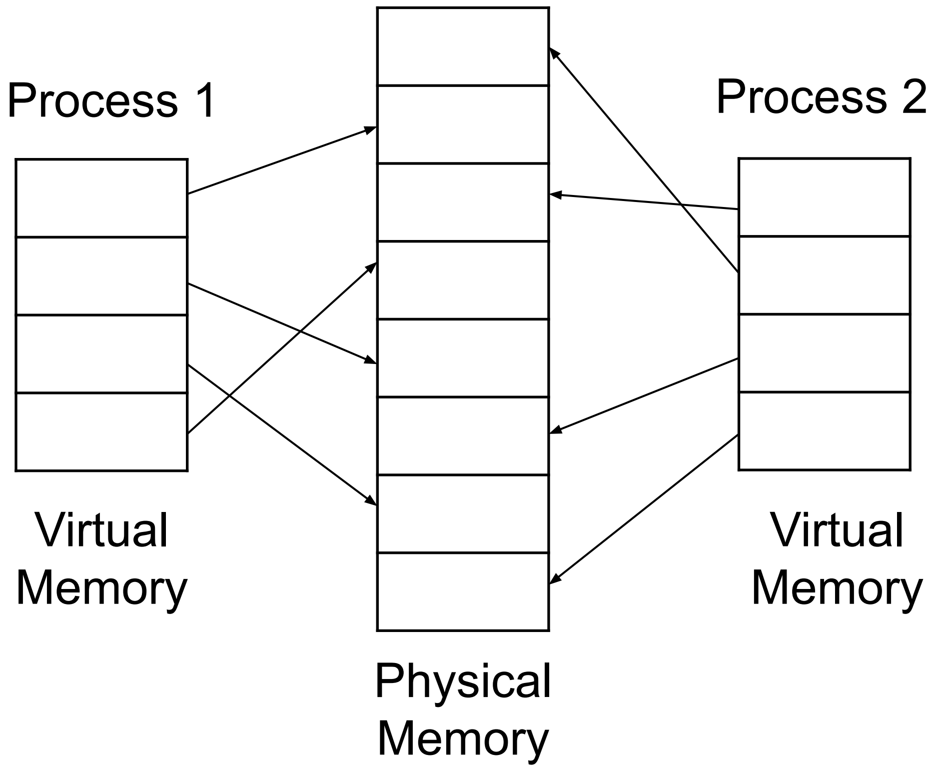
# Virtual memory recall

- Enlarge the physical memory capacity with external storage in a transparent way
  - OS swaps unused data from physical memory to external storage (e.g., disk or remote memory)



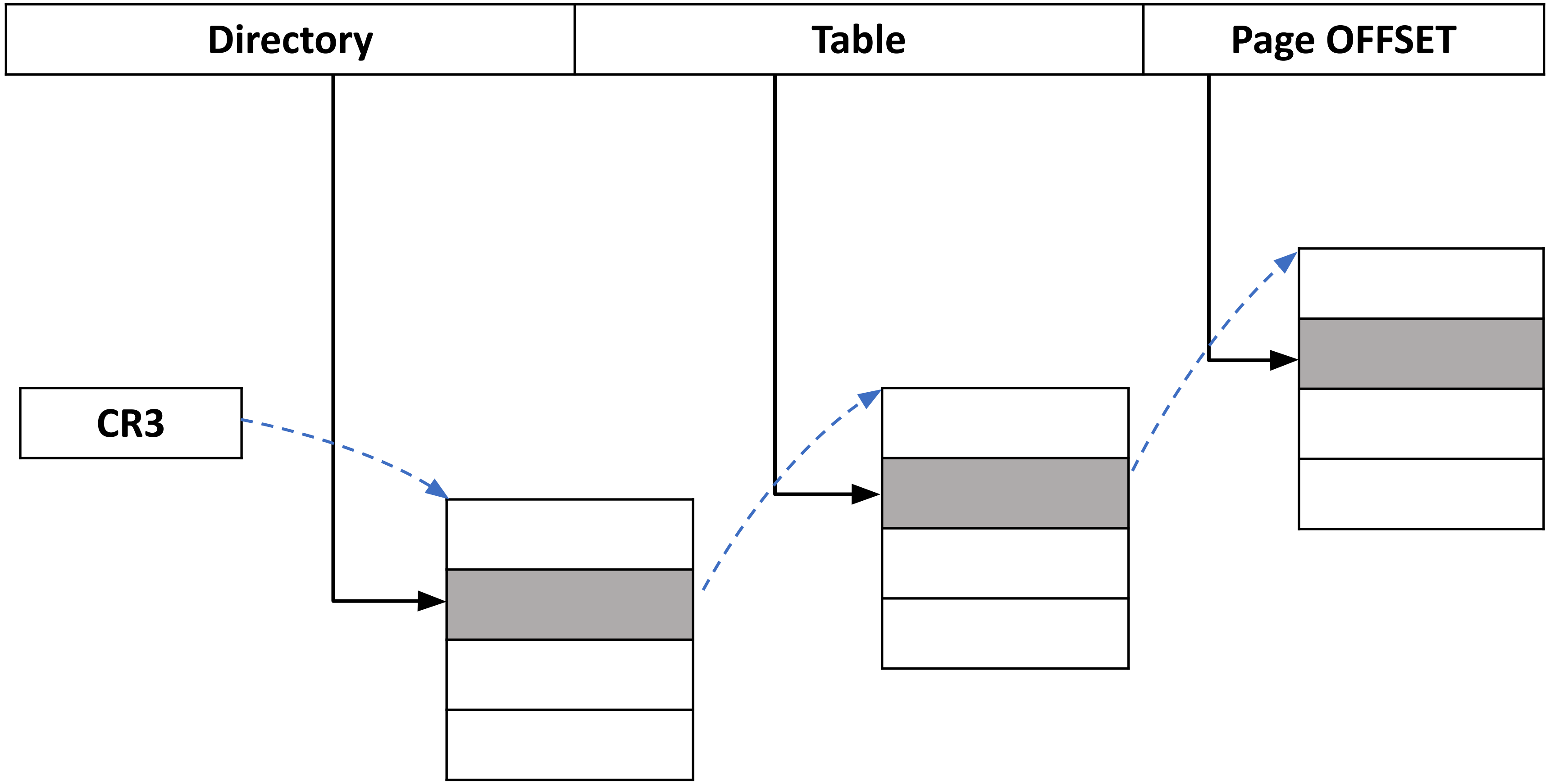
# Virtual memory – Paging

- Physical memories are partitioned into chunks of fixed sizes (Physical Page Frames)
- Eliminates the need for contiguous physical memory
- Paging is used more often than segmentation in modern OSes



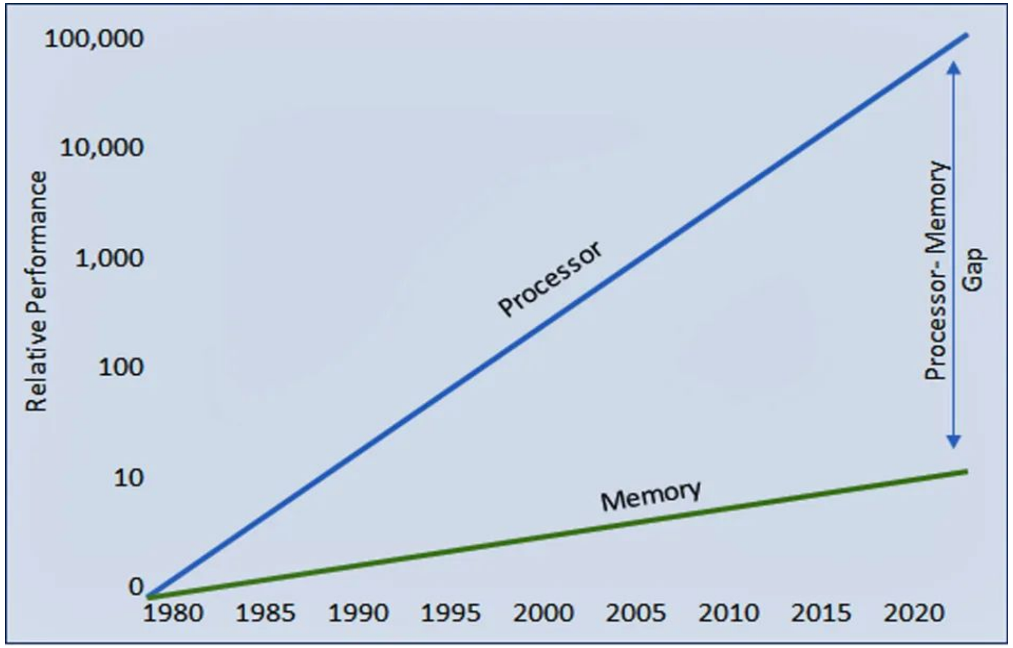
# Paging – Multi-level Page Table

**LD [VA], R1**



# Memory wall

- The performance increase in processors shows dominant trend over that in DRAM
  - DRAM speed becomes slower and slower than CPU (Speed wall)
  - DRAM capacity desired by applications becomes higher and higher (Capacity wall)



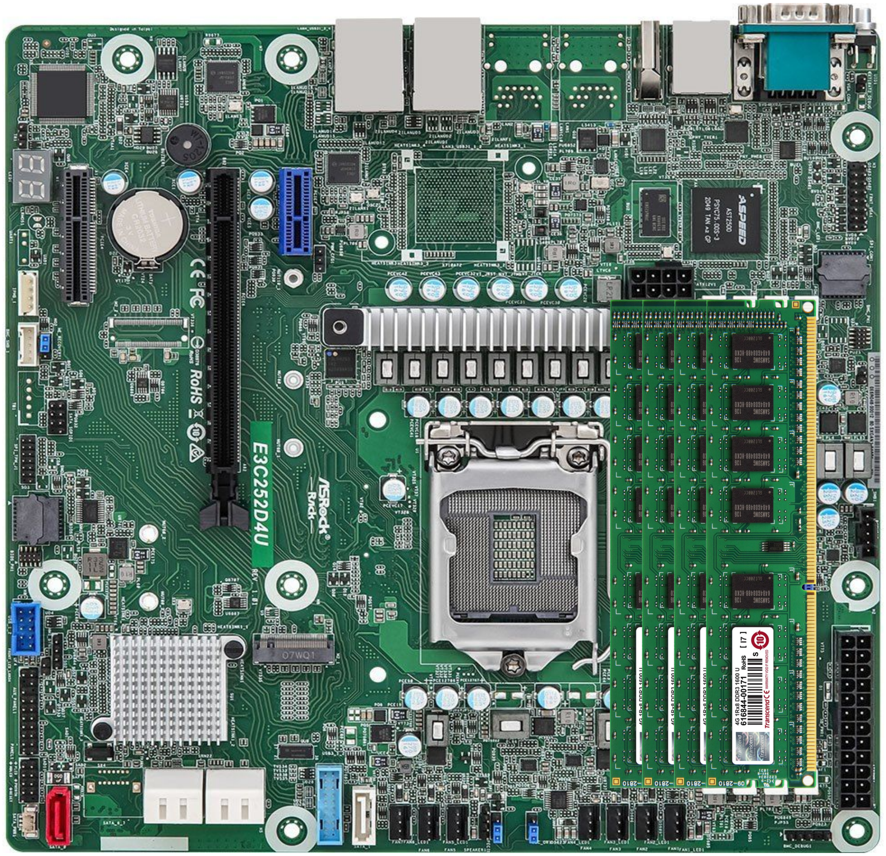

1. https://www.ednasia.com/generative-ai-and-memory-wall-a-wakeup-call-for-ic-industry/

# Memory capacity wall

- Why not enlarge the physical memory size?

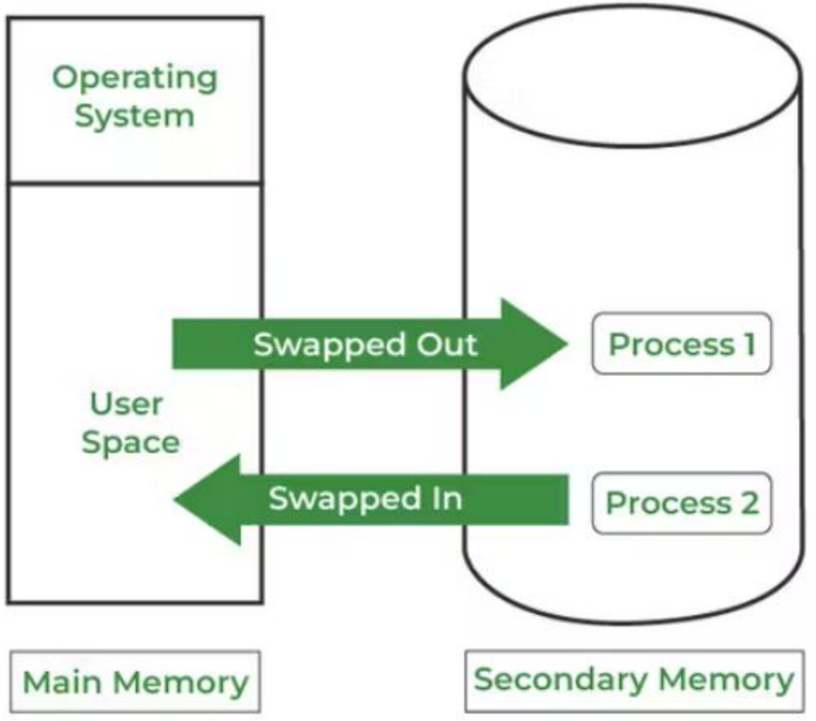
  - **Hardware limitation:** Only limited DRAM slots on each CPU

  - **Cost efficiency:** Installing large physical DRAM leads to DRAM underutilization



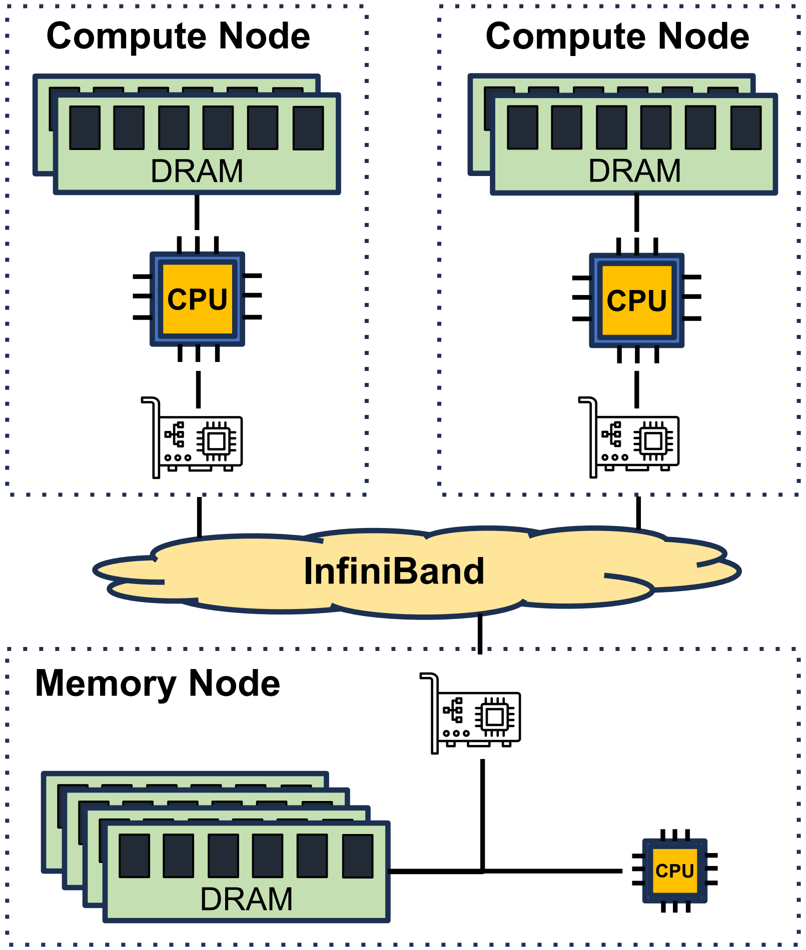
# Extending available memory

- Use storage devices to extend the DRAM
    - The communication between DRAM and storage is via disk I/O operation
    - The performance depends on the speed of the storage device



# Extending available memory

- Use DRAM on another machine to extend the local DRAM
  - The communication between local machine and remote memory is via NIC and RDMA
  - RDMA defines a set of primitives to access DRAM on another machine
  - The performance depends on the speed of the NIC

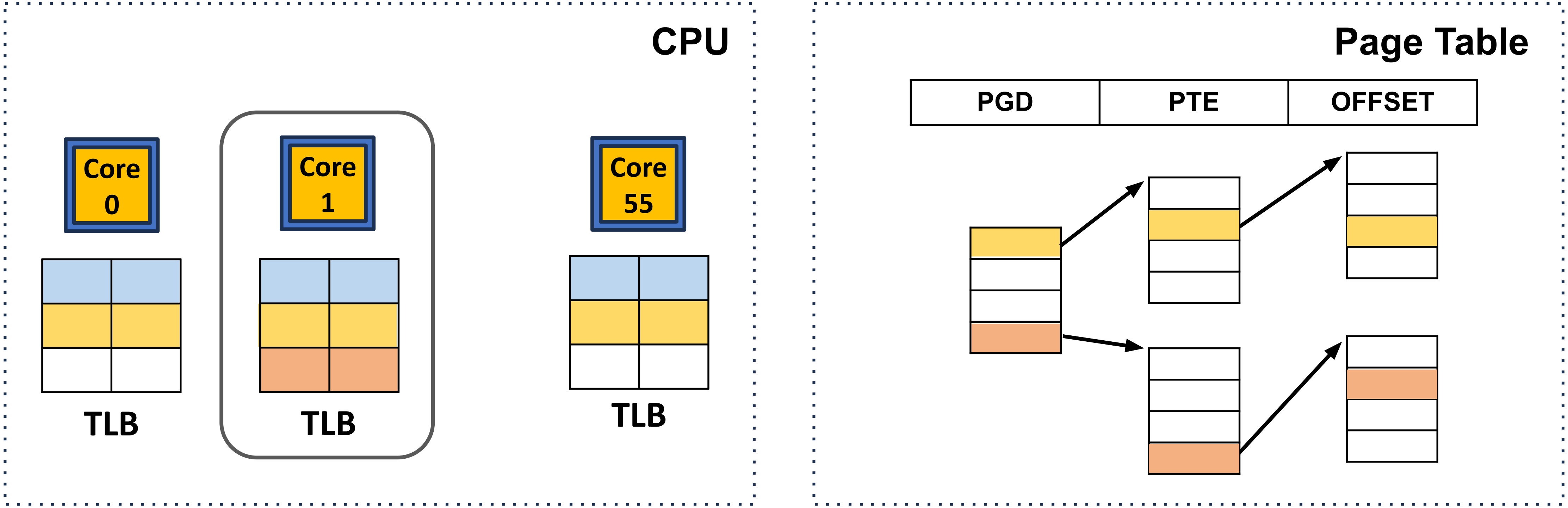


# Paging mechanism recall

**LD [VA0], R1**

**ST [VA1], R2**

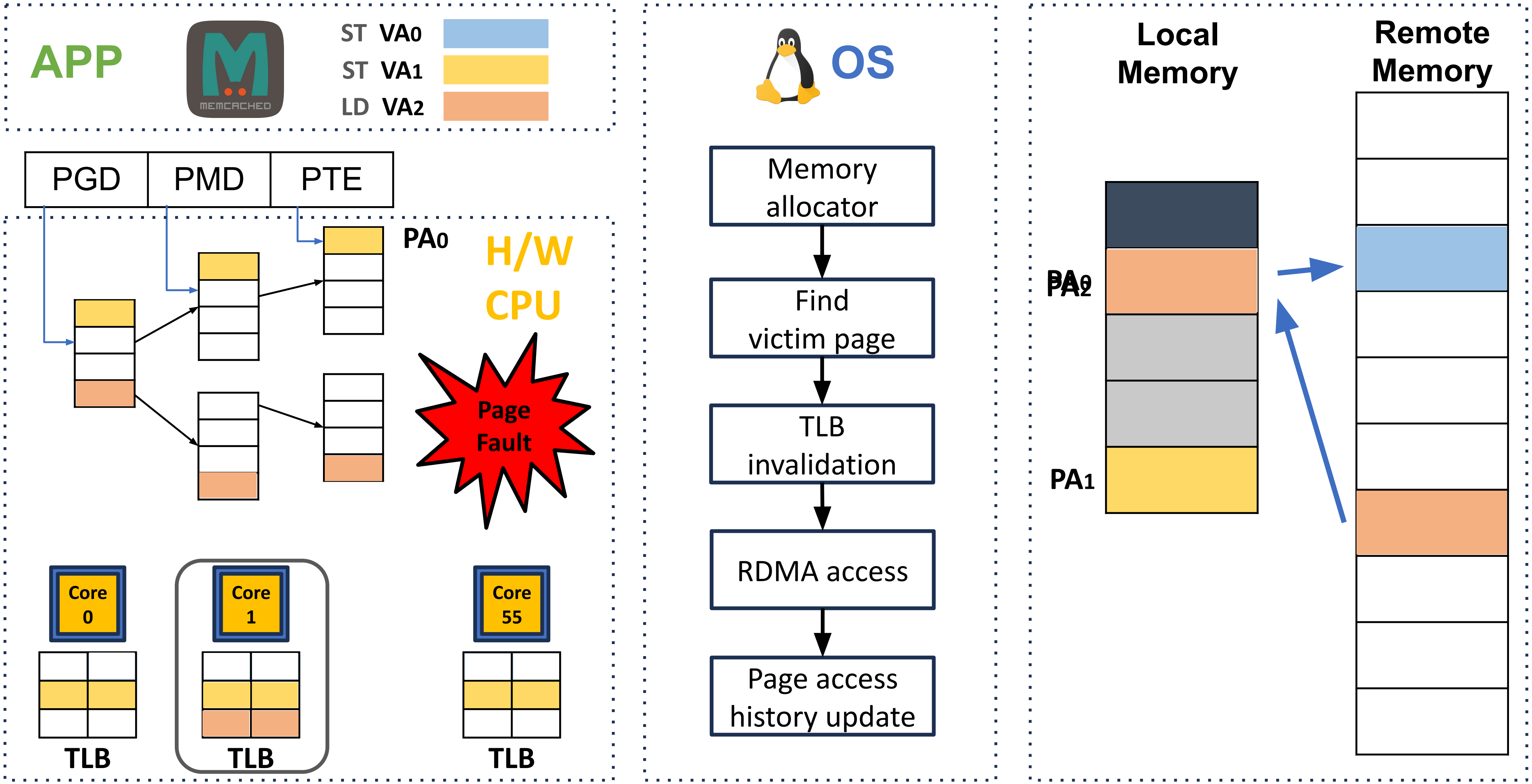
**LD [VA2], R3**



# Mechanism of swapping

- The OS swaps out unused data in a transparent way
  - Applications do not need to handle the page swapping
  - The OS swaps out pages during memory pressure and swap pages in on demand
- The OS utilizes the page faults to swap data between DRAM and external memory
  - A page fault is triggered when a page is not present in DRAM
  - The OS performs swapping during the page fault handler

# Swapping page between local and remote DRAM



# Policies of swapping

- What are the set of pages to swap out?
    - The OS needs to select the victims to be swapped out
- How many pages to swap out?
    - The OS needs to quantify the deficiency of the DRAM
- When to swap them?
    - The OS needs to control the right time to swap them

# Identifying the pages to swap out

- Application pages
  - The data allocated by application
  - Code segments, heap and stack regions

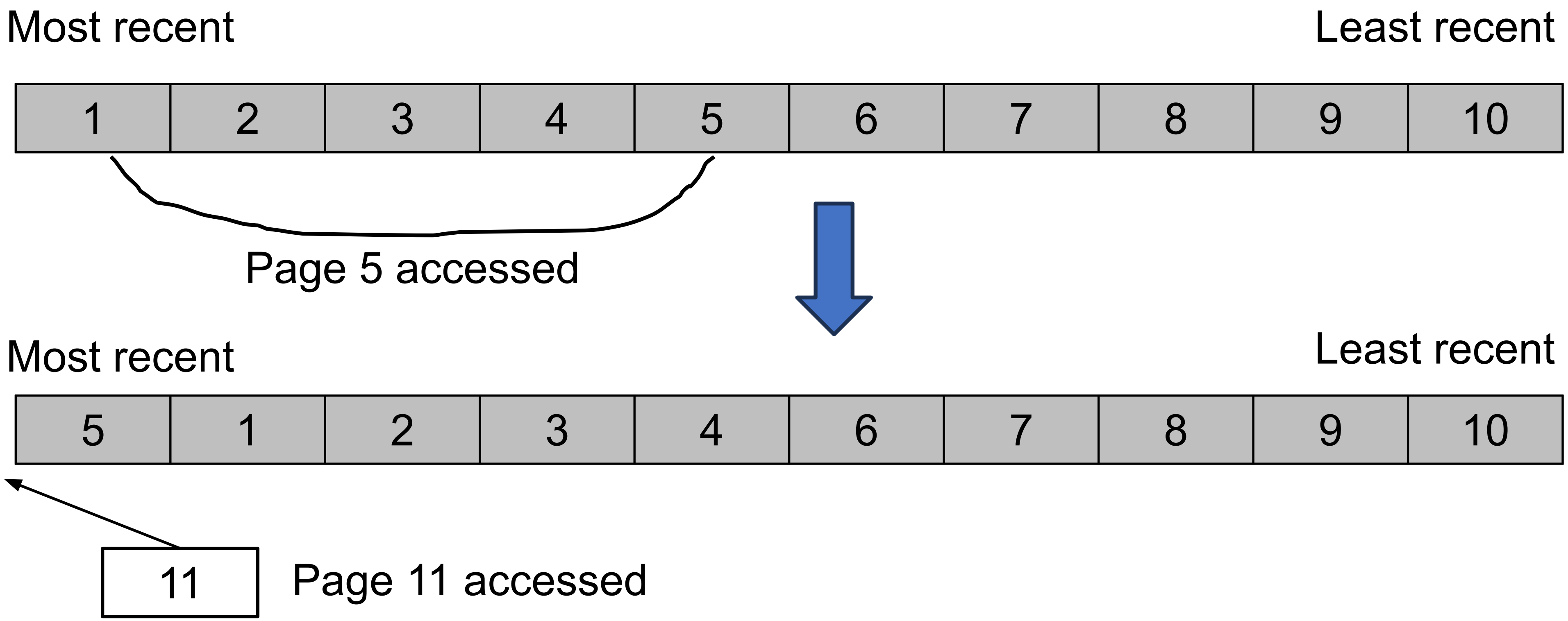
- OS pages
  - The page cache – pages caching files on storage
  - The OS kernel memory objects

# Identifying the pages to swap out

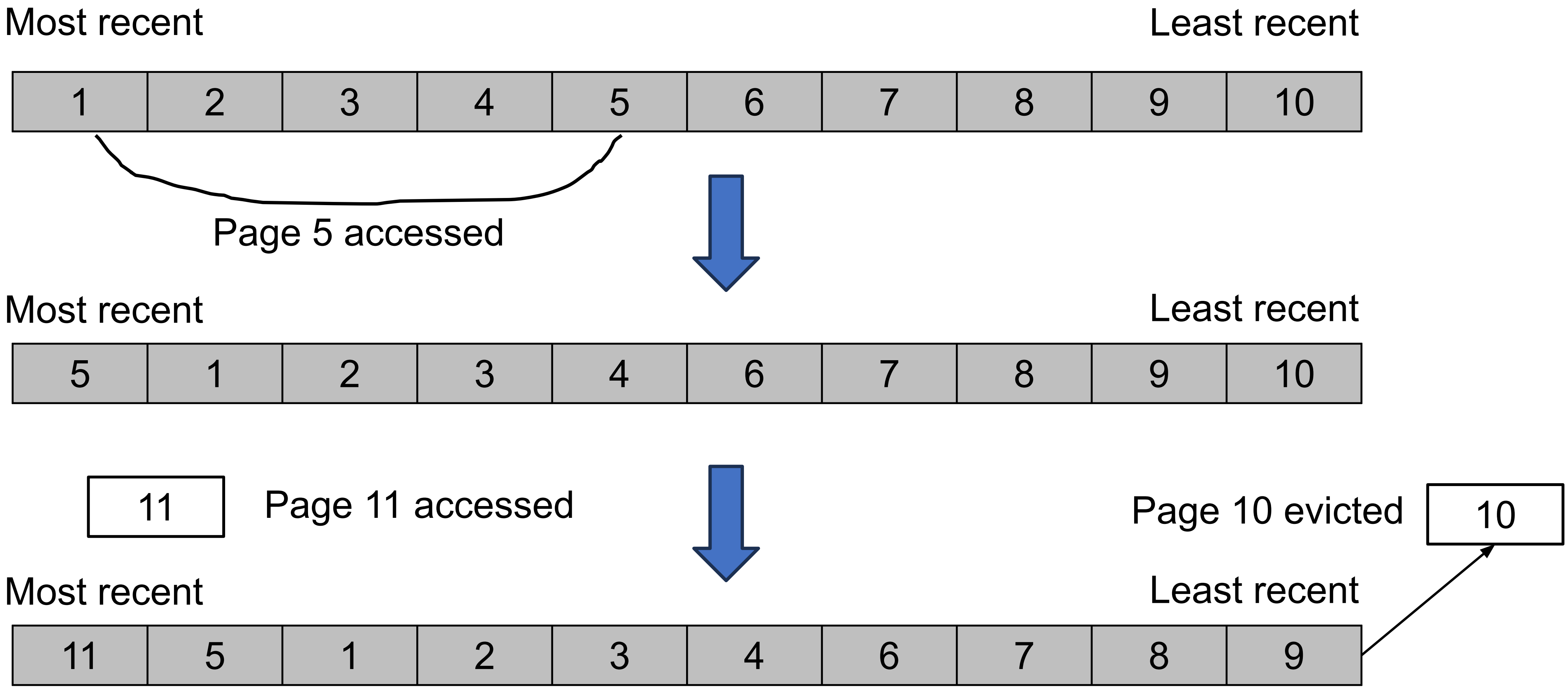
- Two distinct types of applications’ pages
  - Anonymous pages allocated by mmap(MAP_PRIVATE) and populated by page fault, must be swapped out first (if at all possible) to reclaim
  - File pages (a.k.a. page cache) created by file operations or mmap(…, fd) – can be immediately discarded when clean, or after write-out when dirty

# Identifying the pages to swap out

- Linux uses LRU (Least Recently Used) policy to select the victims



# Identifying the pages to swap out



# Identifying the pages to swap out

- Splitting anonymous pages and cached file pages
  - Clean file pages can be just evicted, anonymous have to be swapped out at least once
  - Historically, reclaim has been biased towards file pages more than anonymous

- Split LRU lists into those categories
  - Balance the size of each

# How many pages to swap out

- Detecting memory pressure with a set of ***watermarks***
  - **WMARK_MIN:** The minimum level of free memory that the system should maintain. When free memory falls below this level, the system aggressively tries to free up pages
  - **WMARK_LOW:** A low level of free memory, where the kernel starts reclaiming pages in a more moderate fashion to avoid reaching critical levels
  - **WMARK_HIGH:** The desired level of free memory. If free memory is above this level, no action is taken to reclaim pages

# When to swap out pages

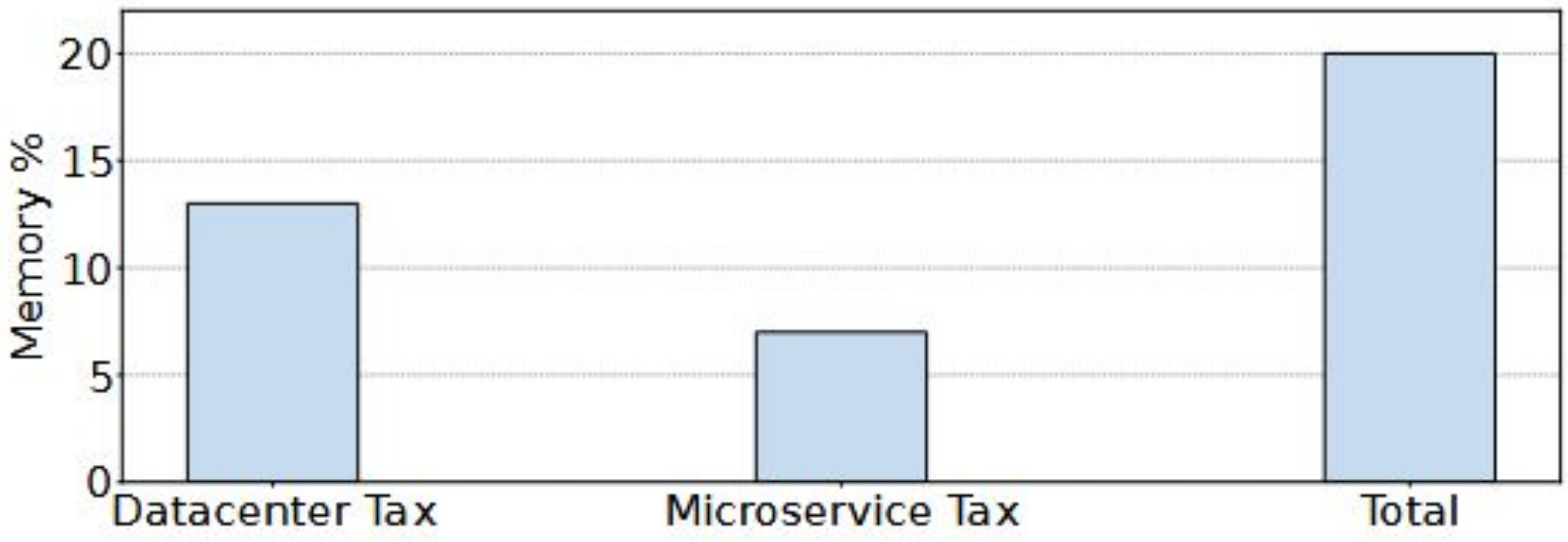
- ***Watermarks*** also guides the time to swap pages

- When the free pages is below the low watermarks, the ***kswapd*** daemon (the Linux kernel swap kernel thread) kicks in

- The ***kswapd*** daemon keeps freeing out pages until the watermark beyond the high watermark.
  - Swapping anonymous pages or dirty file pages to disk or remote memory
  - Reclaim clean file pages directly

# Memory usage patterns in data centers

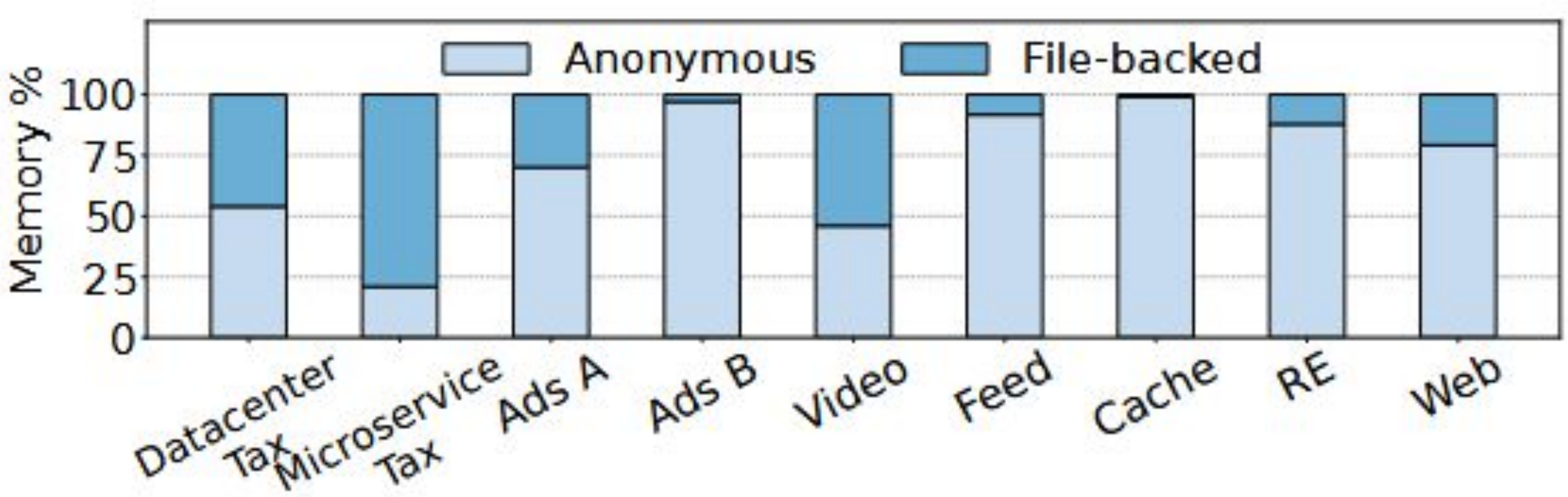
- Data centers pays extra taxes in using memory




  - Software packages, profiling, loggings etc.

# Memory usage patterns in data centers

- Different clients (VMs) presenting diverse memory usage patterns



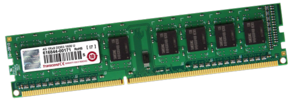

  - Some using anonymous memory more, while others uses file pages more

# Tiered Memory - architecture in data centers

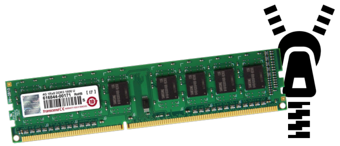
- A set of available memory products to select
  - They have diverse performance characteristics
  - They have different prices tags
  - Use them in a coordinated way helps

    improving cost-efficiency for data centers

DRAM



Compressed DRAM

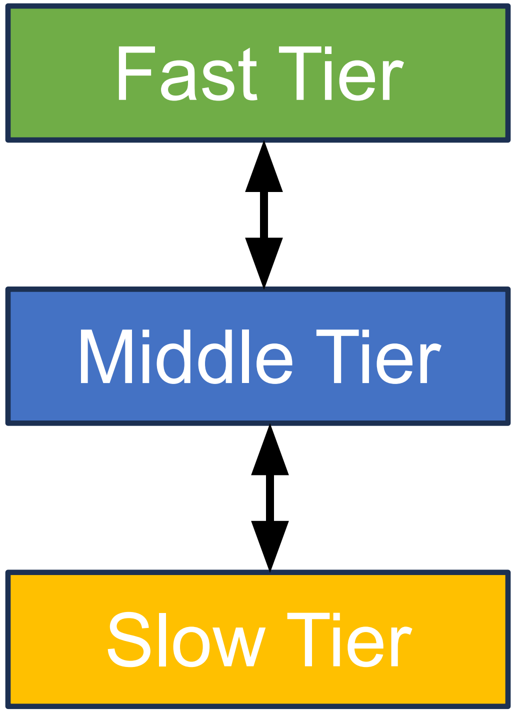


PMEM

NVMe

# Memory architecture in data centers

- Data centers use ***tiered memory*** to achieve cost-efficiency
  - Place fast memory on top of slower ones
  - Keep hot pages in fast memory
  - Move cold pages in slower ones



# Memory requirements in data centers

- Data centers require huge memory to support applications

- Massive growth in memory needs of emerging applications

- Data centers need smart decisions to provision memory resource

# Challenges in swapping pages in tiered memory

- Different memory media presents diverse characteristics
  - Existing solutions like g-swap support only a single slow memory tier
  - Some workload cannot use compressed DRAM

- Determine the page promotion and demotion is not straightforward in data center
  - Offline application profiling cannot reflect runtime memory situation
  - Need more accurate information for VMs (containers) during runtime

# TMO (Transparent Memory Offloading)

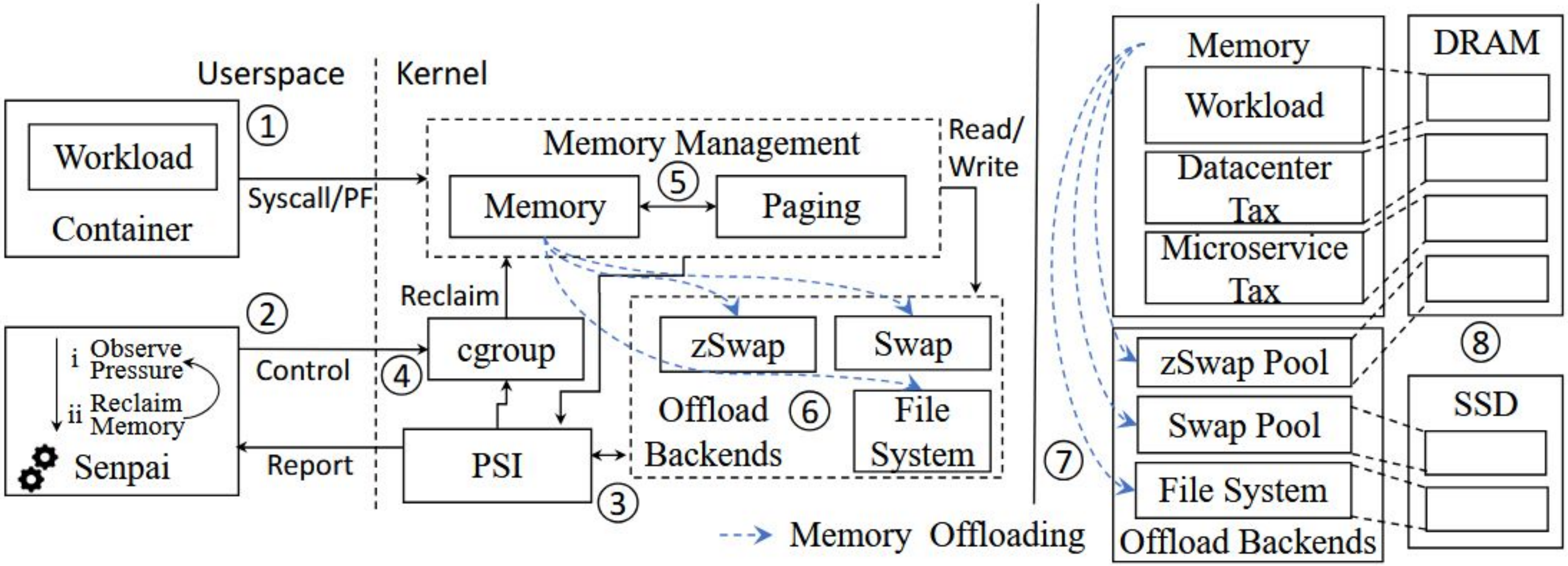
- Introducing PSI (Pressure Stall Information) to overcome the skewness of Linux page reclamation that prioritize file cache over anonymous memory

- Introducing Senpai, a userspace agent with online memory pressure detection mechanism

- Reporting experience with deploying TMO in the production in millions of servers in Meta Inc.

# TMO (Transparent Memory Offloading)

- TMO high-level architecture



# Defining resource pressure

- Existing OS solutions heavily rely on event counters
  - E.g., major/minor page fault counts
  - They cannot accurately capture the application situation

- High major page fault counts could indicate the workload initialization

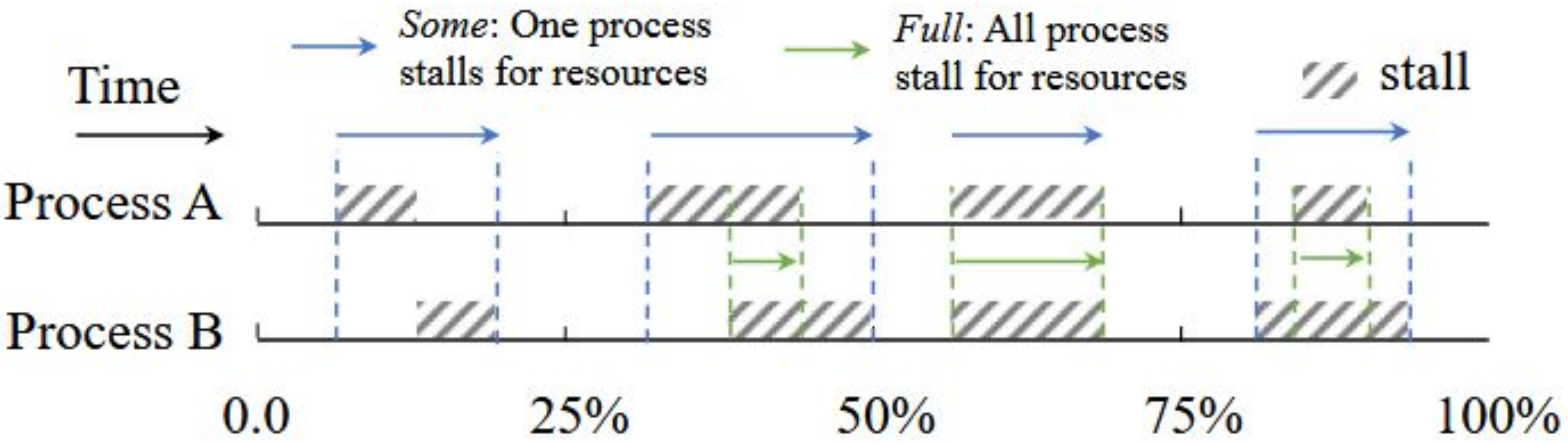
- High minor page fault counts could indicate the workload and memory access pattern transition

# PSI (Pressure Stall Information)

- PSI metrics - overview
  - Specify the amount of lost work due to the lack of a resource
  - PSI is the proportion of compute potential that is unproductive due to resource stalls
- PSI metrics - "***some***" metric
  - the percentage of time in which at least one process within the domain is stalled waiting for the resource
- PSI metrics - "***full***" metric
  - the percentage of time in which all processes are delayed simultaneously

# PSI (Pressure Stall Information)

- PSI metrics - “***some”*** and “***full”*** metric



# PSI (Pressure Stall Information)

- PSI metrics use cases

- "***some"*** metric measures the impact of resource insufficiency in each individual VMs

- "***full"*** metric measures the unacceptable losses of productivity that require immediate remediation

# Determine memory requirements for VMs

- Estimating the memory requirement for a workload is challenging

- Especially hard to estimate the memory used for populating the page caches via file system operations.

- Applications do not know if their memory are overprovisioned

# Determine memory requirements for VMs

- ***Senpai*** – use PSI to determine the memory requirements

$$reclaim\_mem = current\_mem \times reclaim\_ratio \times max(0, 1 - \frac{PSI_{some}}{PSI_{threshold}})$$

- If $\text{PSI}_{\text{some}}$ is high, it means this VM requires more resources to complete.

- If $\text{PSI}_{\text{some}}$ is low, it means this VM might be overprovisioned, so the TMO can reclaim some unused pages from this VM

# Balancing anonymous pages and file pages

- Two types of pages to consider as victims to swap out
    - Linux favors reclaiming file pages than anonymous pages

## Identifying the pages to swap out

- Splitting anonymous pages and cached file pages
    - Clean file pages can be just evicted, anonymous have to be swapped out at least once
    - Historically, reclaim has been biased towards file pages more than anonymous
- Split LRU lists into those categories
    - Balance the size of each

20

# Balancing anonymous and file pages

- Linux swapping subsystem favors reclaiming file pages than anonymous pages

- Some clean file pages can be dropped directly while anonymous pages needs to be swapped anyway

- Some file pages will be accessed again but reclaimed

# Identifying the reused file pages

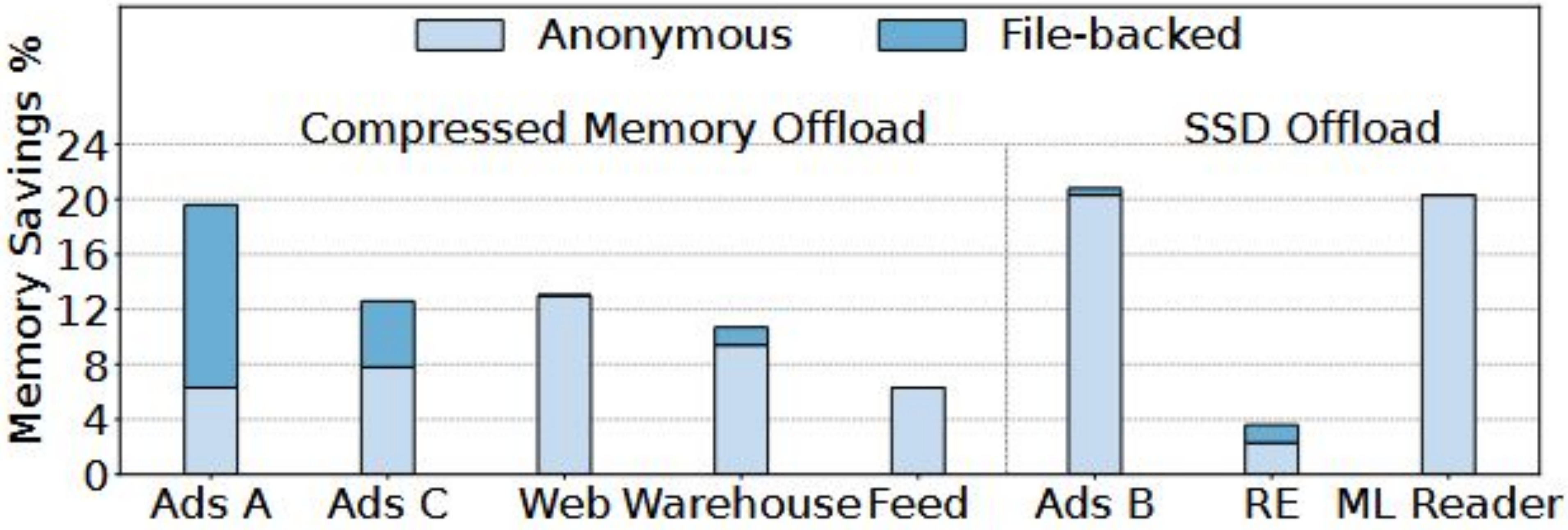
- ***Refault distance*** - the number of accesses to unique page made since the last reference to the requested page

- Count the eviction for a file page ($Count_{eviction}$) and compare with the page fault count ($Count_{fault}$).

- If $Count_{eviction}$ < $Count_{fault}$), it means more faults happens after evicting file pages. Then TMO tries to reduce reclamation from file pages

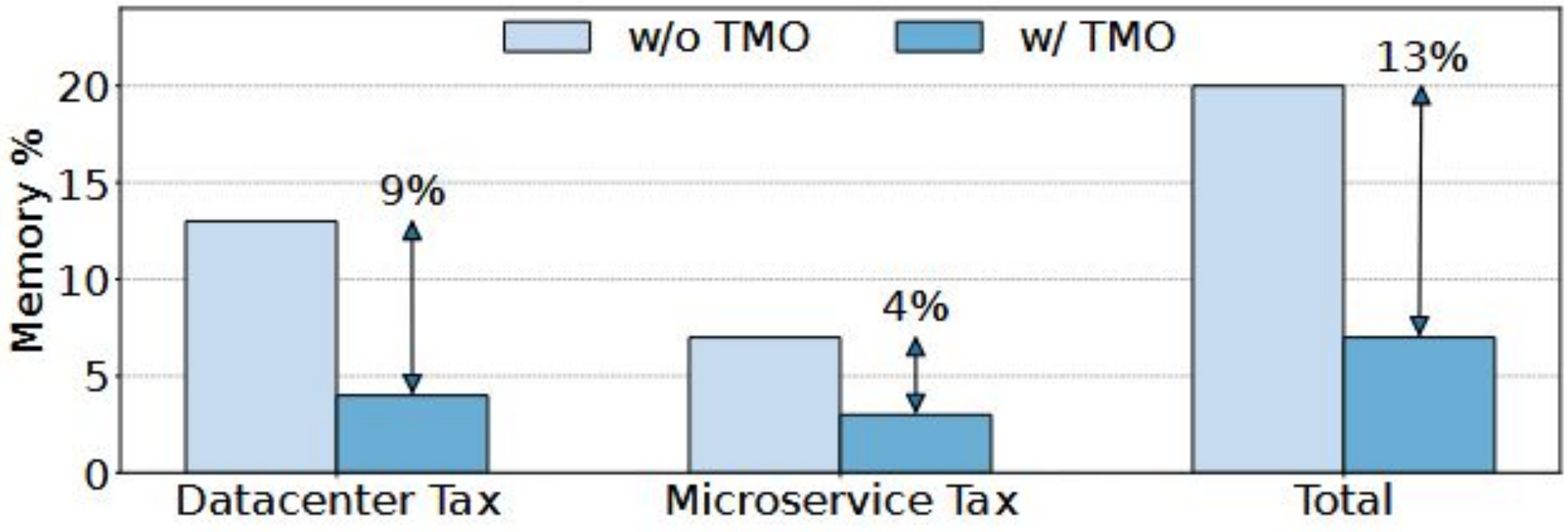
# TMO – Evaluation

- Memory savings



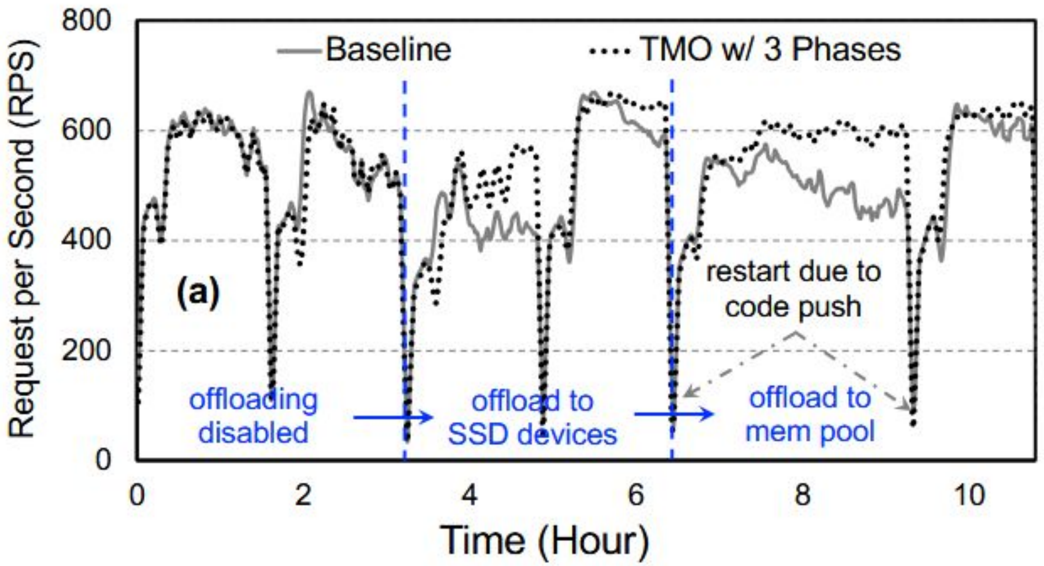
# TMO – Evaluation

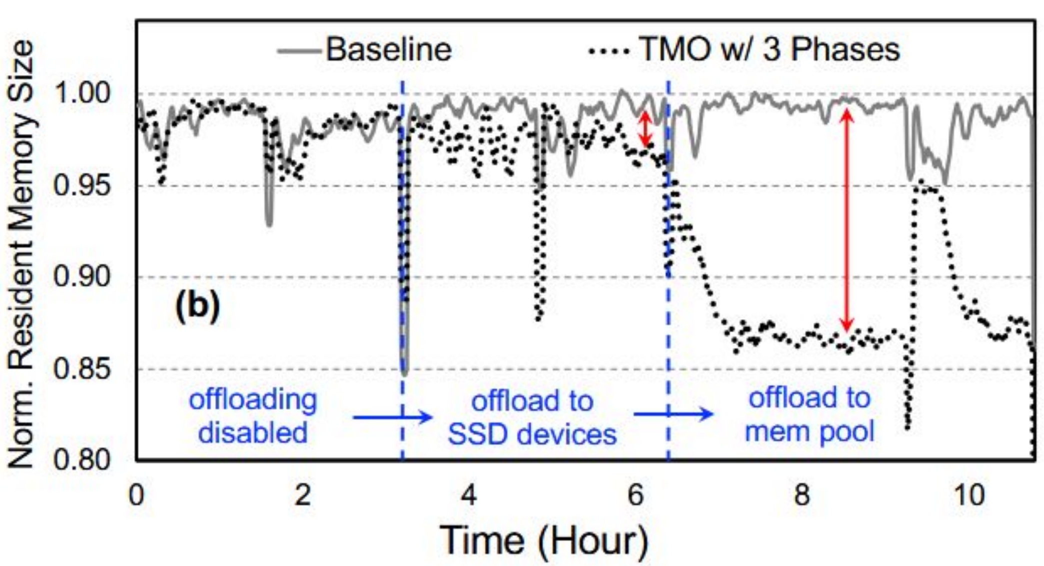
- Memory Tax savings



# TMO – Evaluation

- Case study – Web application on memory-bound hosts



# TMO – Summary

- Swapping is challenging in tiered memory setup in data centers
- PSI is a reliable metric to identify the resource shortage in resources (memory, CPU) in both per-application level and global level
- Calculating refault distance is important to understand the accurate information of how many file pages are accessed and not