

SAY-Go: Towards Transparent and Seamless Storage-As-You-Go with Persistent Memory

Hyeonho Song, **Sam H. Noh**

UNIST

HotStorage 2018

SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

Center for Innovative System Software Research

NECSST

Next-generation Embedded / Computer System Software Technology

Contents

- **Persistent Memory**
- **Motivation**
- **SAY-Go**
  - Design
  - Implementation
- **Evaluation**
- **Summary**
- **Future work**

Persistent memory

Persistent Memory Features

- Non-volatility
- Byte-level random access
- Fast access time (*nanoseconds*)

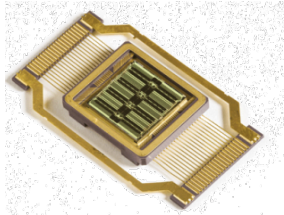
SAMSUNG EVERSPIN TECHNOLOGIES The MRAM Company Micron SK hynix intel



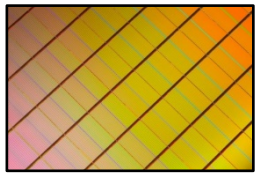
Storage



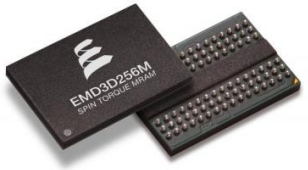
Memory



PRAM



3D XPoint



STT-MRAM

Persistent memory

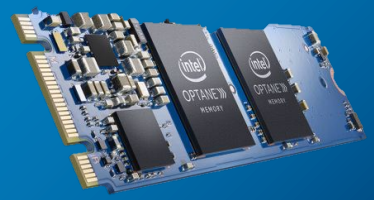
1H’ 2017



Intel® Optane SSD DC P4800X

Block device

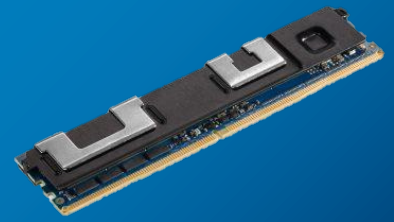
1H’ 2017



Intel® Optane Memory

Block device

Coming soon 2H’ 2018



DIMM Based Optane memory

Character device

Intel Displays 512GB Optane DC Persistent Memory DIMMs

7 COMMENTS

by Paul Alcorn May 30, 2018 at 6:00 PM

Intel held its Memory and Storage day today at its Santa Clara headquarters to announce its Optane DC Persistent Memory DIMMs. The new DIMMs slot into the DRAM interface, just like a normal stick of RAM, but come in three capacities of 128, 256, and 512GB. That's a massive capacity increase compared to the industry-leading 128GB DDR4 memory sticks. Intel designed the DIMMs to bridge both the performance and pricing gap between storage and memory, so the new DIMMs should land at much lower price points than typical DRAM.

Persistent memory

1H' 2017



Intel® Optane SSD DC P4800X

Block device

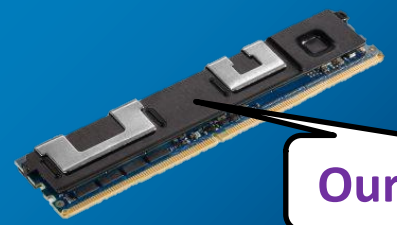
1H' 2017



Intel® Optane Memory

Block device

Coming soon 2H' 2018



DIMM Based Optane memory

Character device

Our target

Intel Displays 512GB Optane DC Persistent Memory DIMMs

7 COMMENTS

by Paul Alcorn May 30, 2018 at 6:00 PM

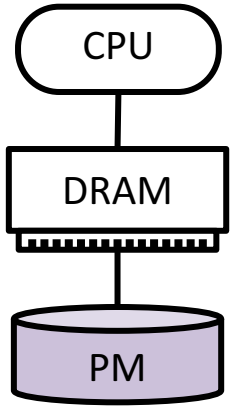
Intel held its Memory and Storage day today at its Santa Clara headquarters to announce its Optane DC Persistent Memory DIMMs. The new DIMMs slot into the DRAM interface, just like a normal stick of RAM, but come in three capacities of 128, 256, and 512GB. That's a massive capacity increase compared to the industry-leading 128GB DDR4 memory sticks. Intel designed the DIMMs to bridge both the performance and pricing gap between storage and memory, so the new DIMMs should land at much lower price points than typical DRAM.

Persistent memory

1H' 2017



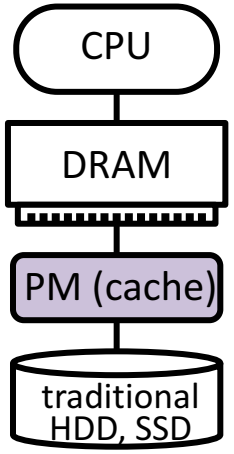
Intel® Optane SSD DC P4800X



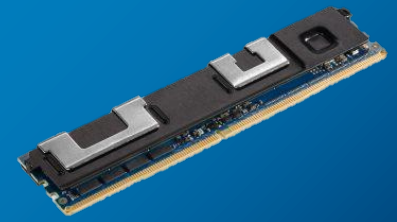
1H' 2017



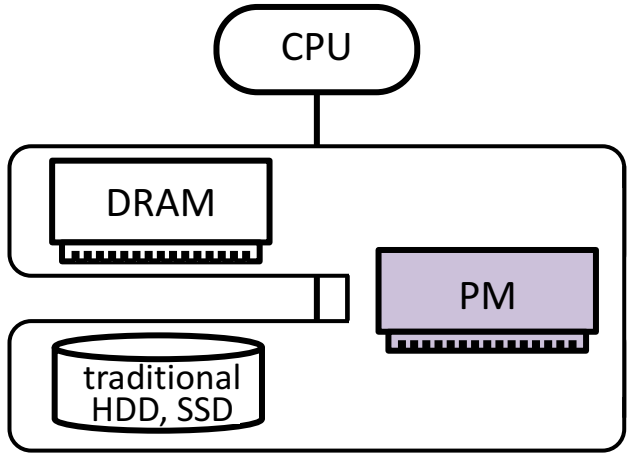
Intel® Optane Memory



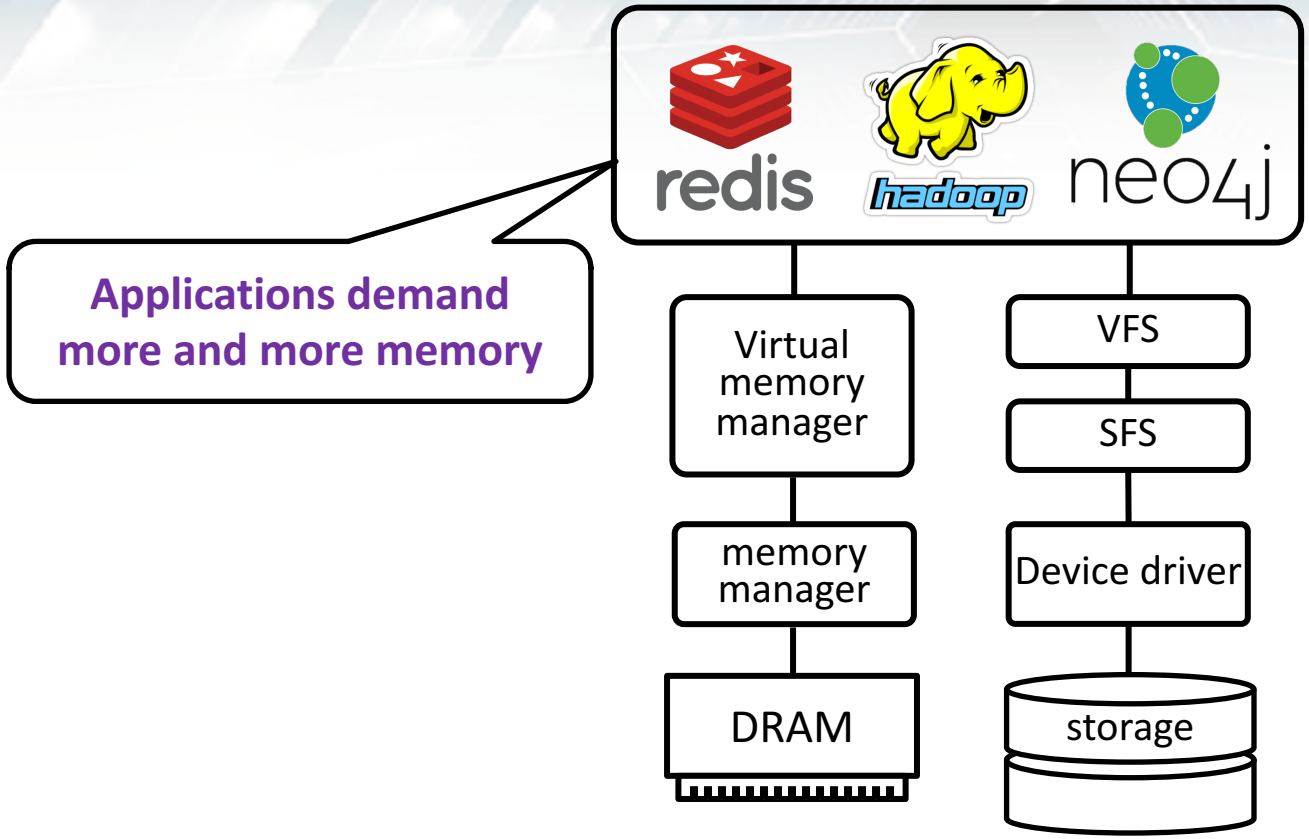
Coming soon 2H' 2018



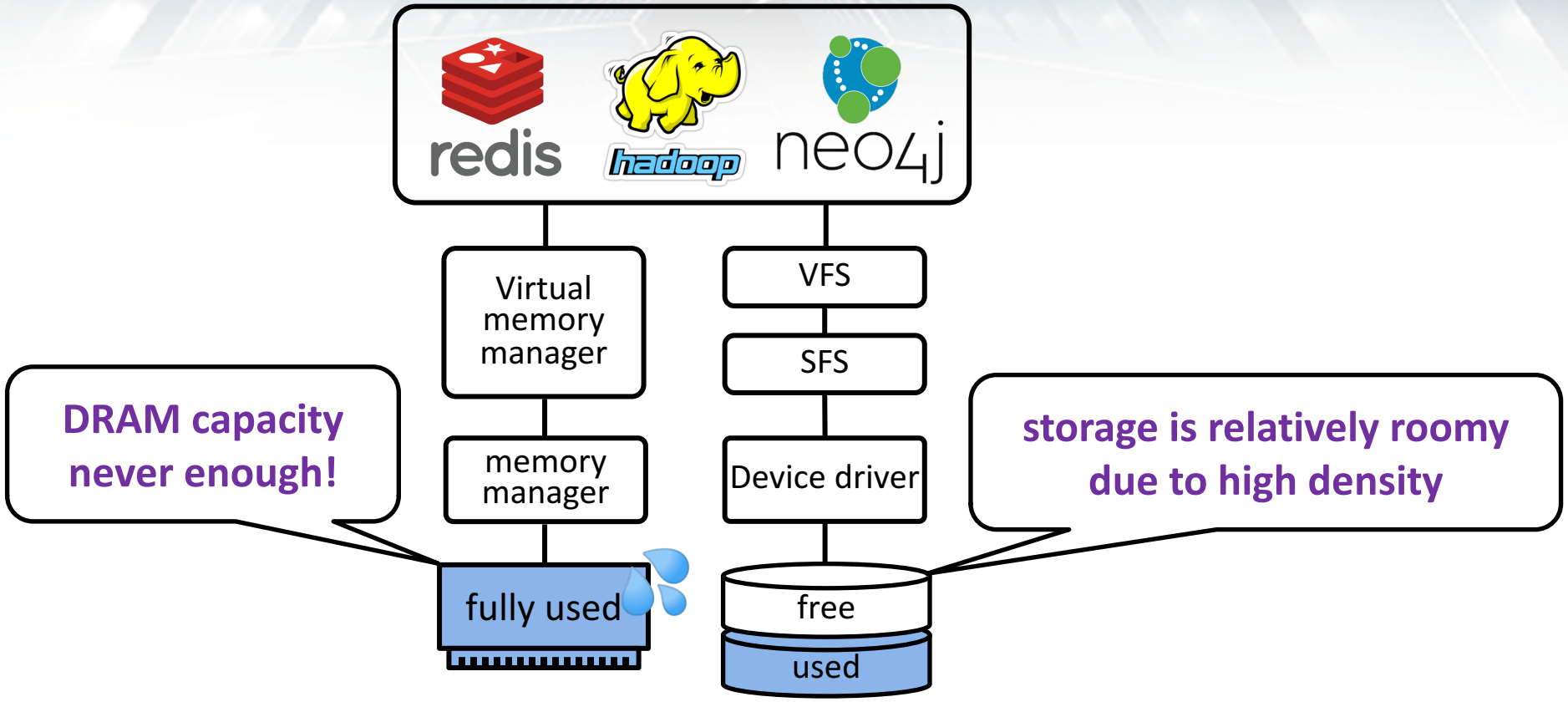
DIMM Based Optane memory



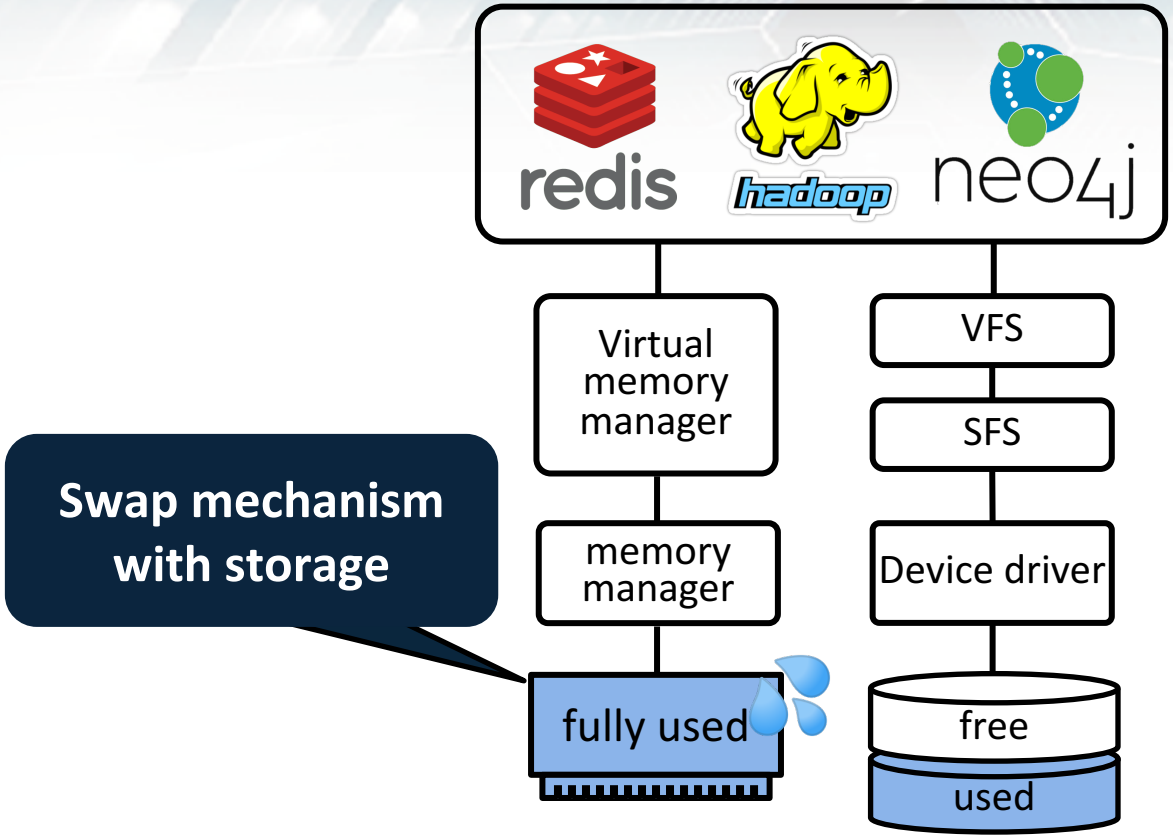
Motivation


Applications demand more and more memory
redis
hadoop
neo4j
Virtual memory manager
memory manager
DRAM
VFS
SFS
Device driver
storage

Motivation

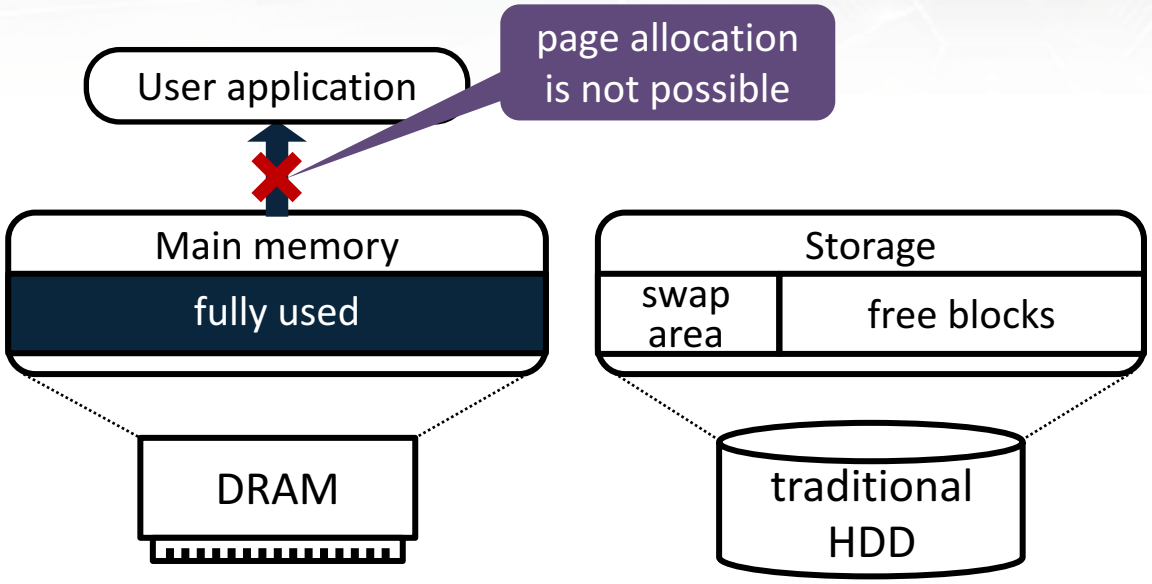

redis
hadoop
neo4j
Virtual memory manager
memory manager
VFS
SFS
Device driver
DRAM capacity never enough!
storage is relatively roomy due to high density
fully used
free
used

Traditional solution



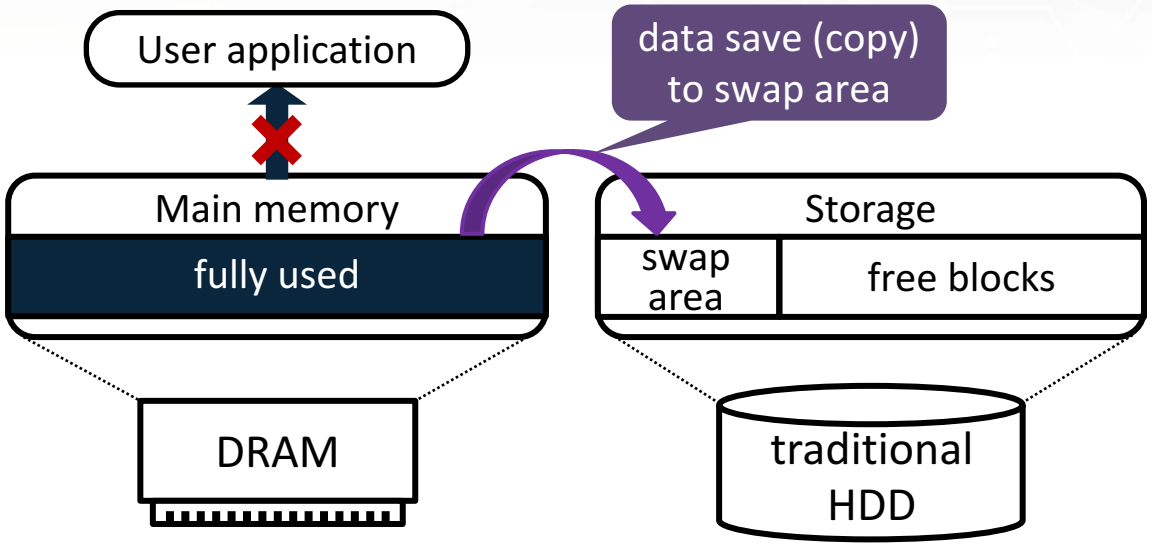
Swap mechanism

- **When DRAM runs out**



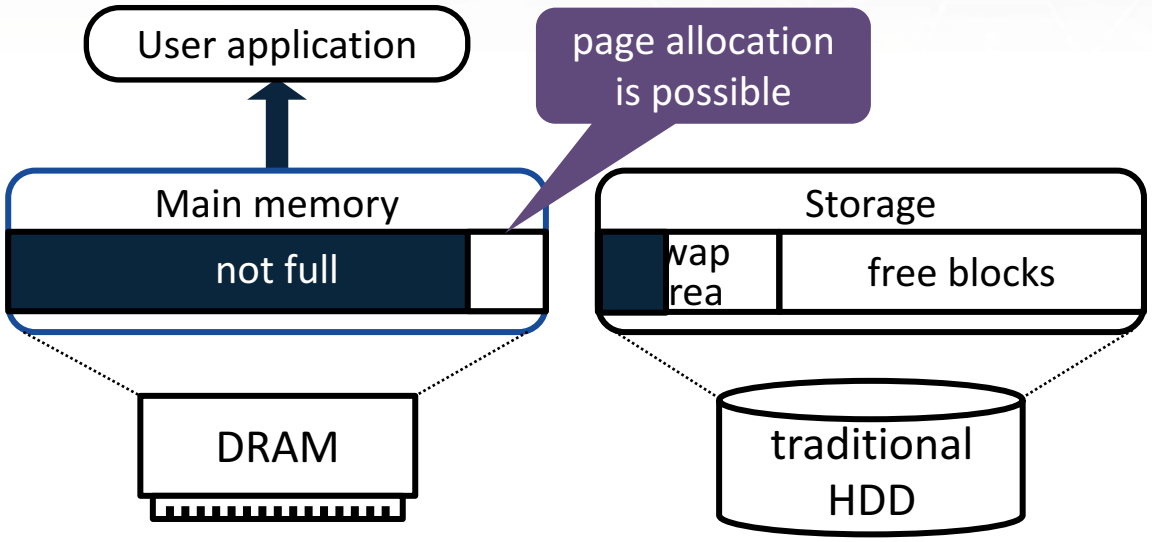
Swap mechanism: swap-out

- **Secure free space in DRAM**



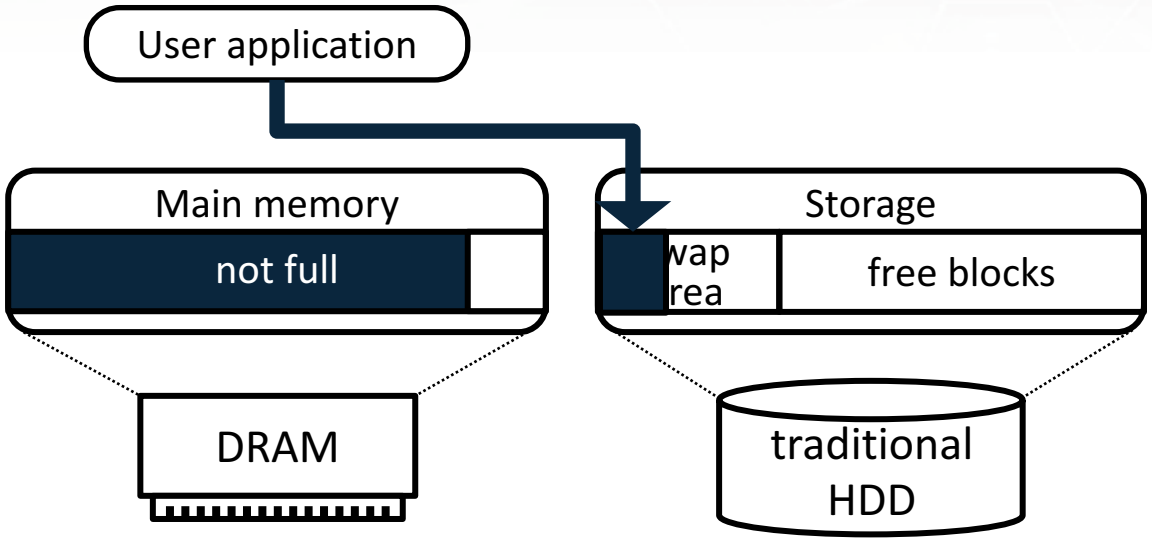
Swap mechanism: swap-out

- **Secure free space in DRAM**



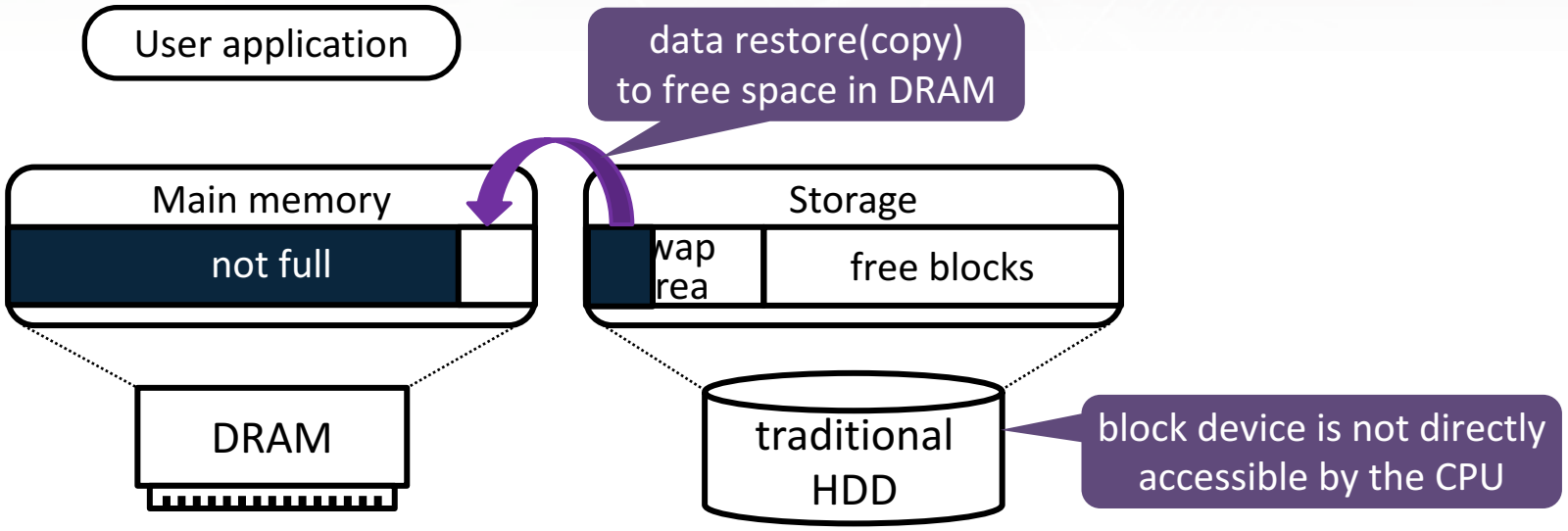
Swap mechanism: swap-in

- **When application wants to access data in swap area**



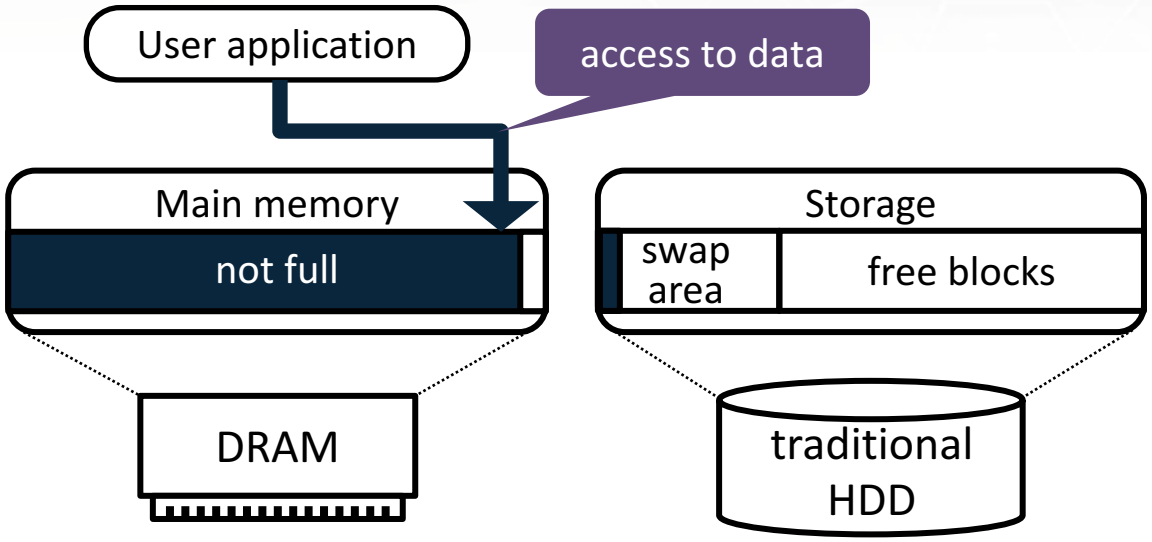
Swap mechanism: swap-in

- **When application wants to access data in swap area**



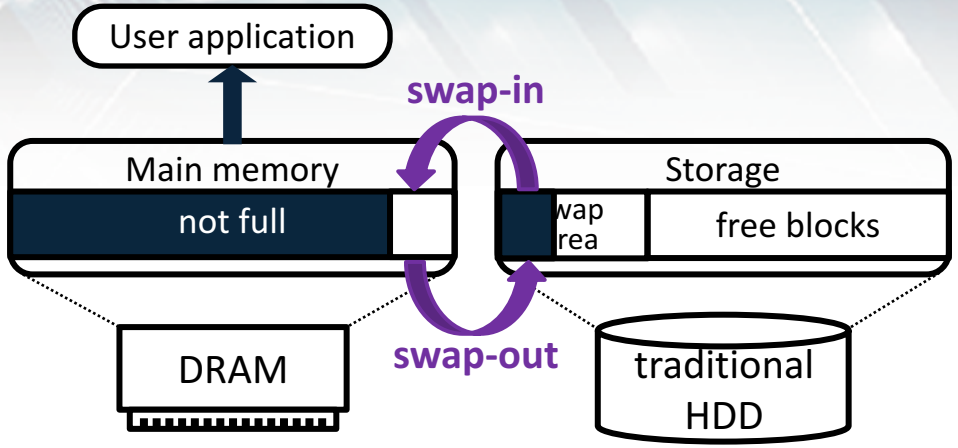
Swap mechanism: swap-in

- When application wants to access data in swap area

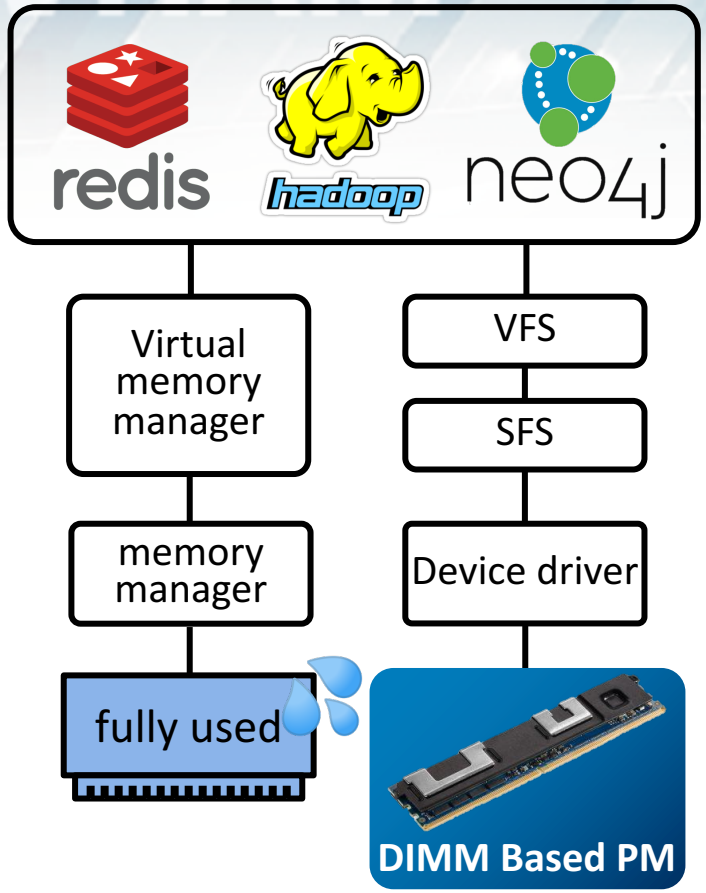


Weakness of traditional swap mechanism

- **Large overhead with slow storage**
  - Data copy is essential
  - Selecting pages to be swapped out is a complex process
- **Swap area cannot be used as storage space**
  - Static partition of file system

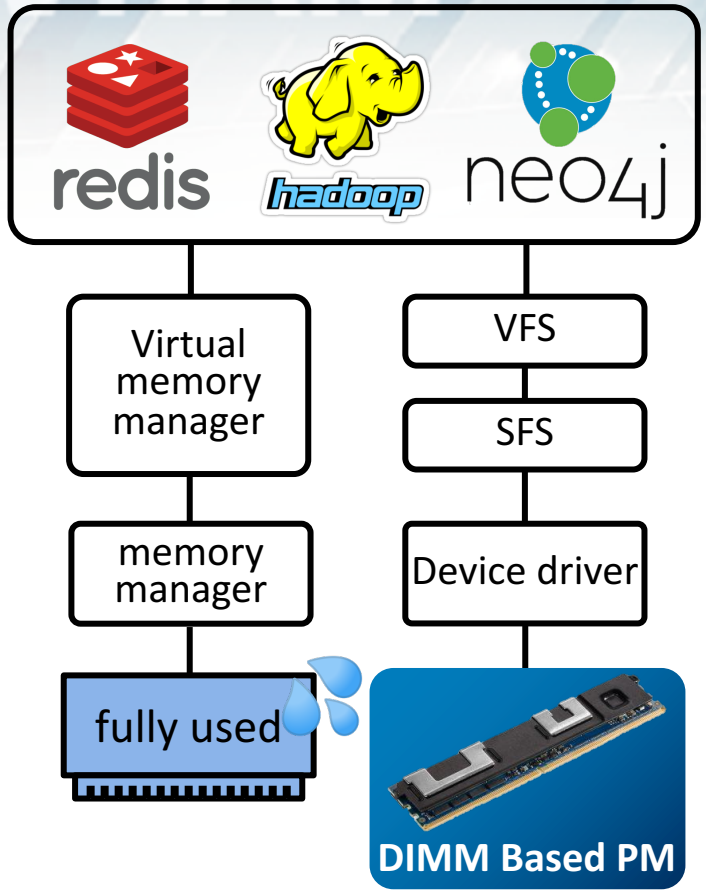


Simple solution: swap with PM storage



- **PM performance is better than traditional storage**
  - Reduced data copy time
  - Overhead of the swap mechanism is also reduced
- **Does this suffice?**

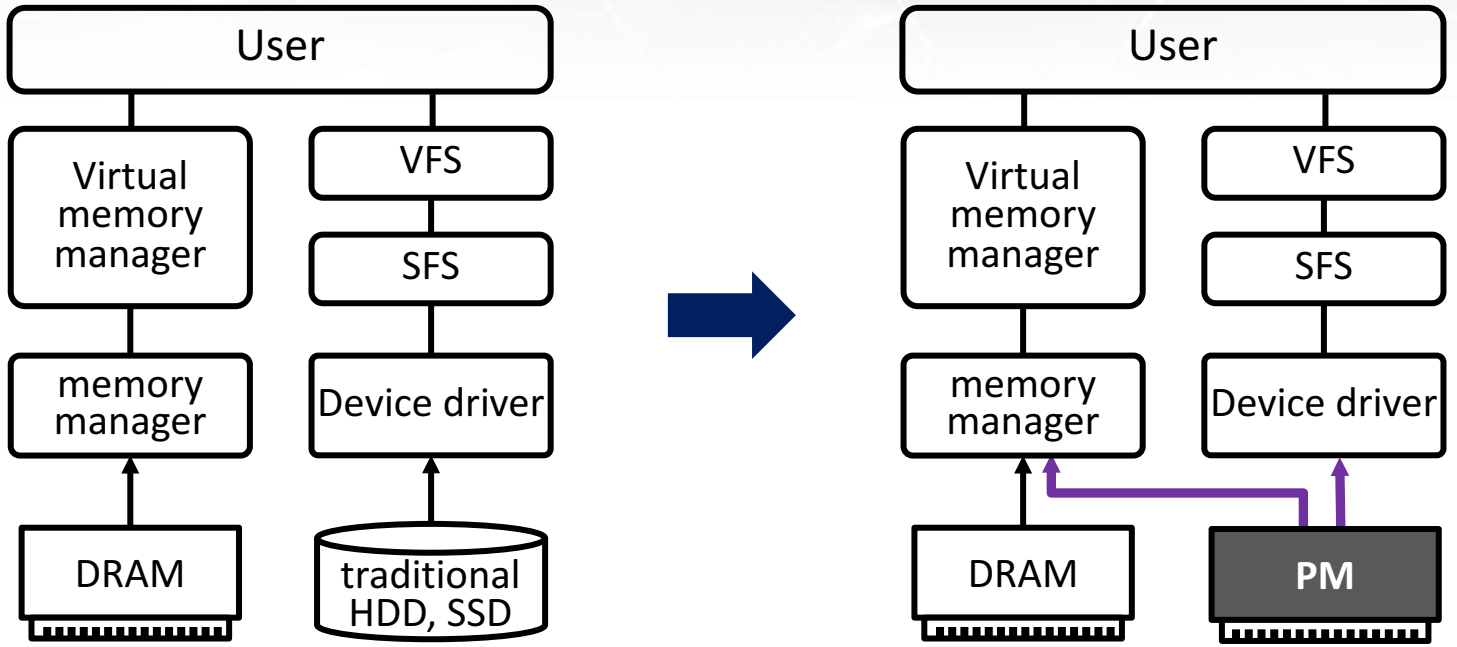
There should be a better way?



- **PM storage can be accessed directly from the CPU**
- **PM as both main memory and storage**
  - Dynamically use PM
  - Increase capacity of the working memory when needed

Our Goal

- **Dynamic adjustment of the memory and storage boundary**

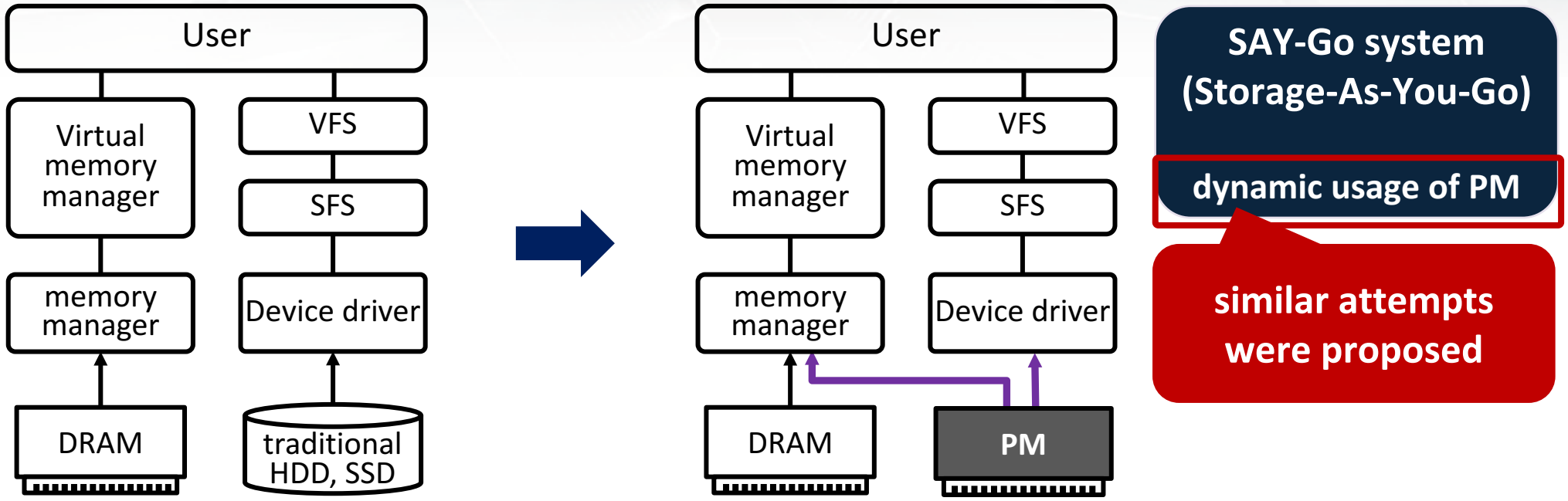


SAY-Go system (Storage-As-You-Go)

dynamic usage of PM

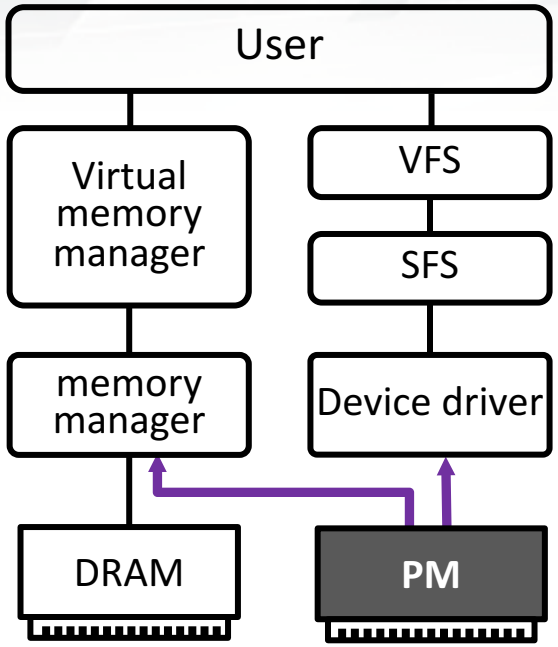
Our Goal

- **Dynamic adjustment of the memory and storage boundary**



Related work

- **Dynamic usage of PM**



Memorage[1] – ICS 2013
: Working memory expansion

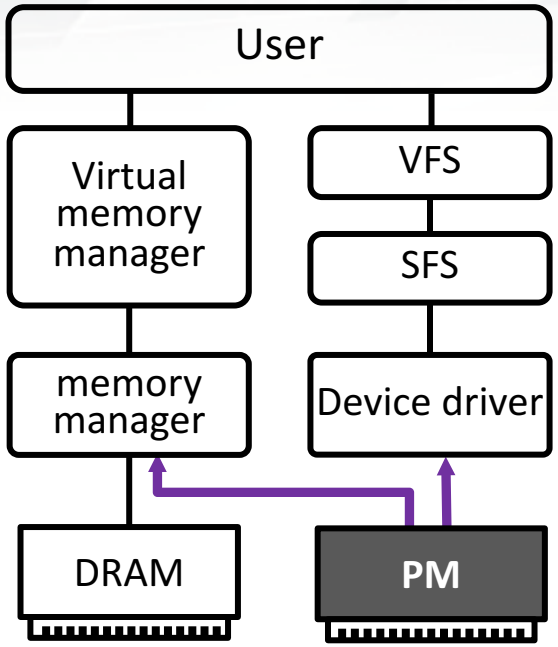
pVM[2] – EuroSys 2016
: Direct allocation of PM page to working memory

1) Jung, J. Y. and Cho, S. "Memorage: Emerging persistent ram based malleable main memory and storage architecture." In Proceedings of the International ACM Conference on International Conference on Supercomputing (ICS) (2013).

2) Kannan, S., Ada, G., and Karsten, S. "pVM: persistent virtual memory for efficient capacity scaling and object storage." In Proceedings of the European Conference on Computer Systems (EuroSys) (2016).

Related work

- **Dynamic usage of PM**



Memorage[1] – ICS 2013
: Working memory expansion

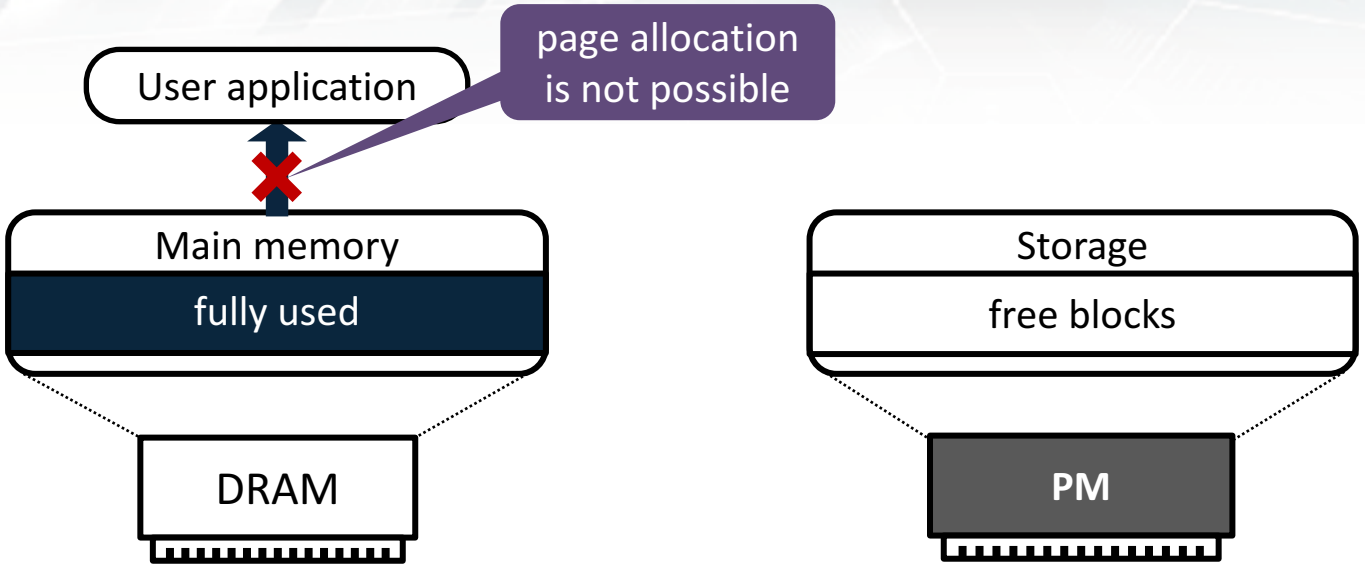
pVM[2] – EuroSys 2016
: Direct allocation of PM page to working memory

1) Jung, J. Y. and Cho, S. "Memorage: Emerging persistent ram based malleable main memory and storage architecture." In Proceedings of the International ACM Conference on International Conference on Supercomputing (ICS) (2013).

2) Kannan, S., Ada, G., and Karsten, S. "pVM: persistent virtual memory for efficient capacity scaling and object storage." In Proceedings of the European Conference on Computer Systems (EuroSys) (2016).

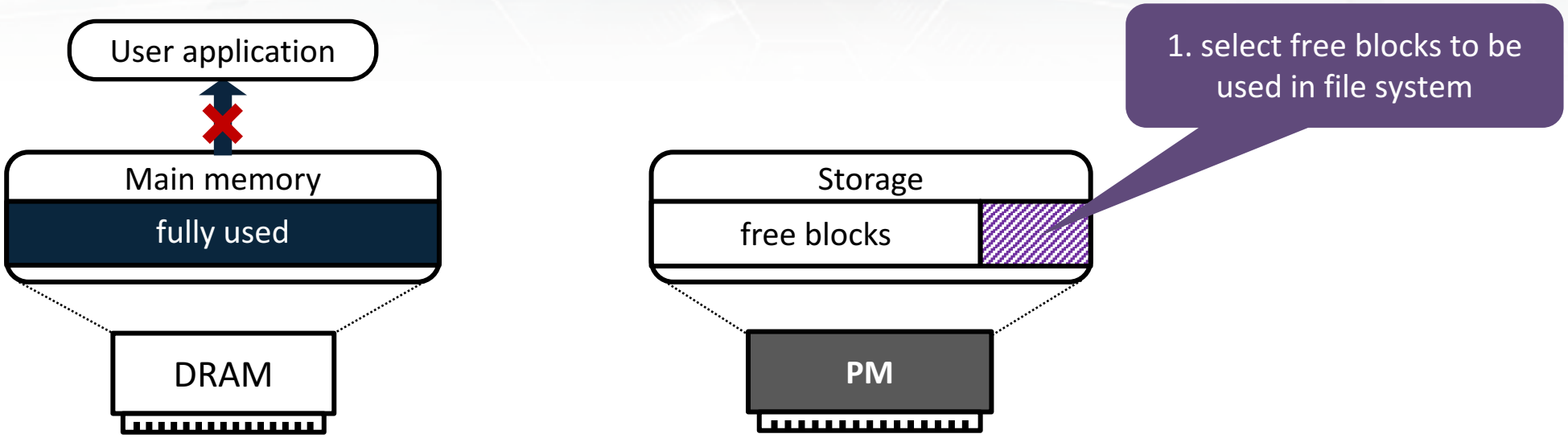
Memorage

- **When DRAM runs out**



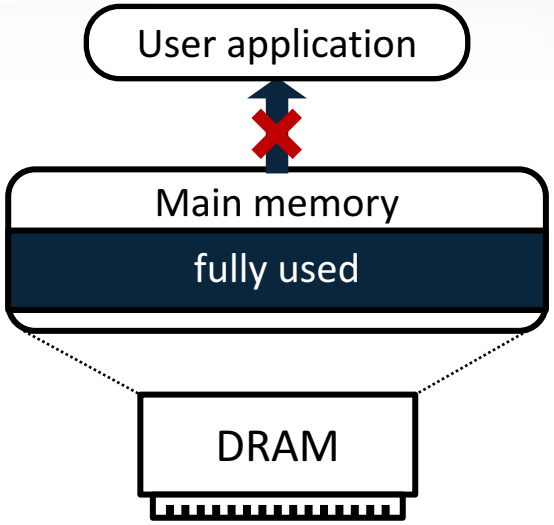
Memorage

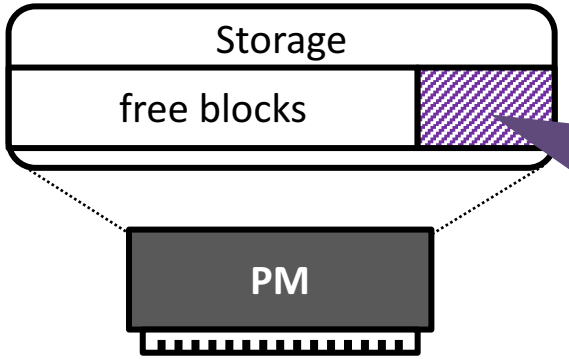
- **Working memory expansion**



Memorage

- **Working memory expansion**



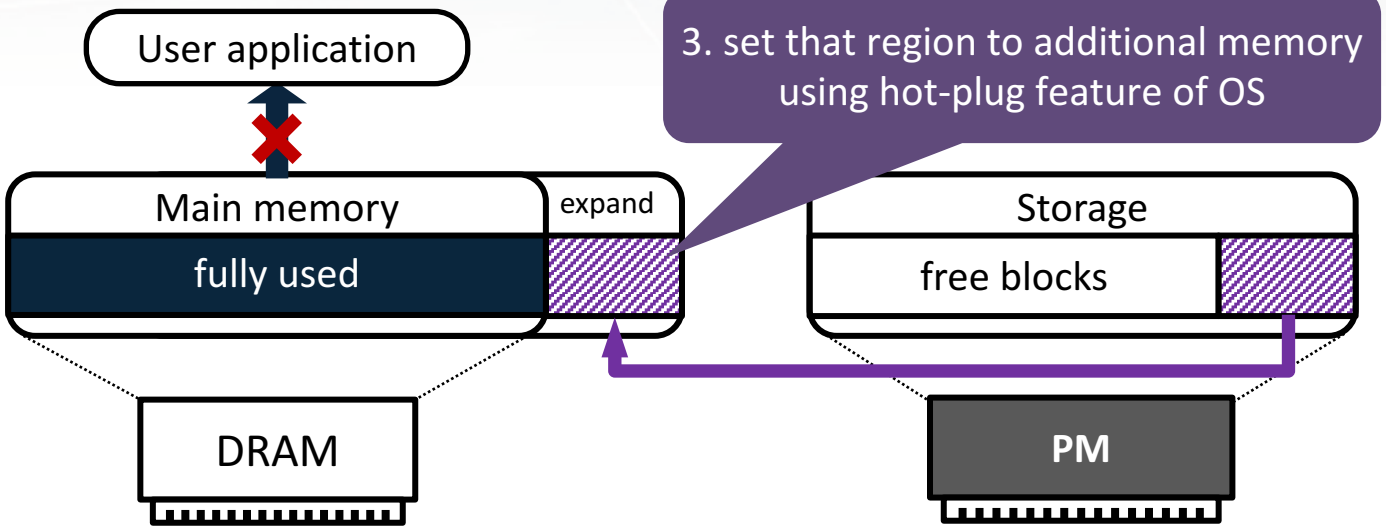


1. select free blocks to be used in file system

2. blocks are transformed into allocatable free pages by reorganizing them as data structures used in the OS memory management layer

Memorage

- **Working memory expansion**



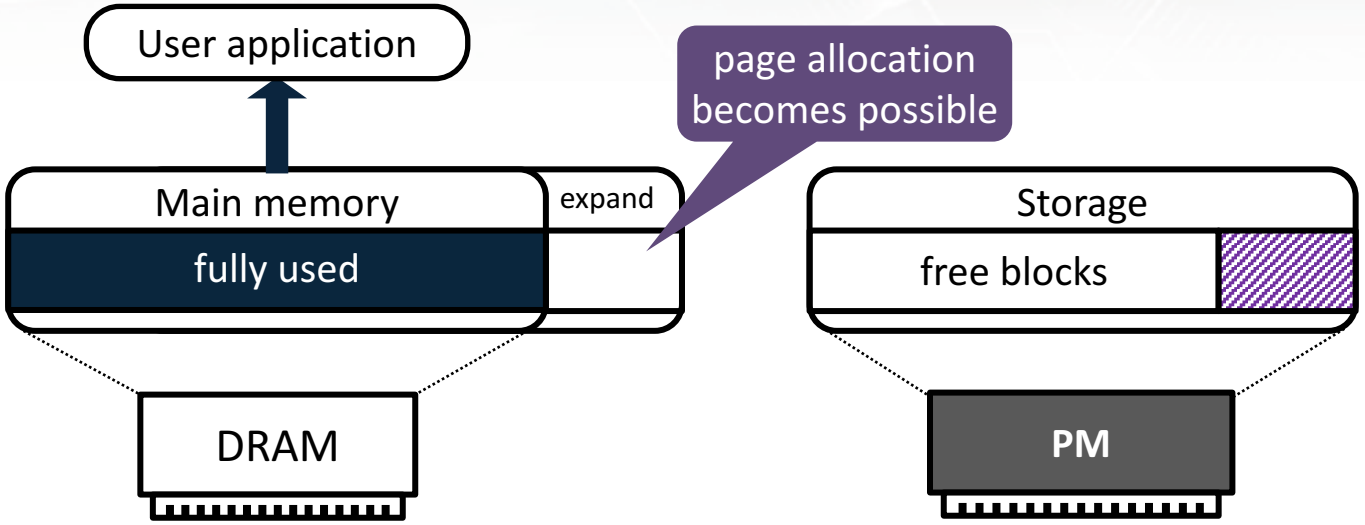
3. set that region to additional memory using hot-plug feature of OS

1. select free blocks to be used in file system

2. blocks are transformed into allocatable free pages by reorganizing them as data structures used in the OS memory management layer

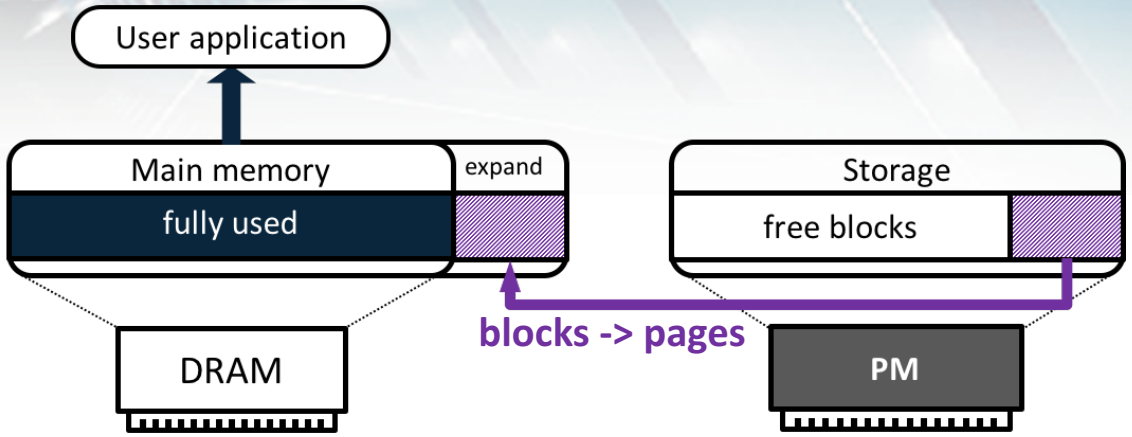
Memorage

- **Working memory expansion**



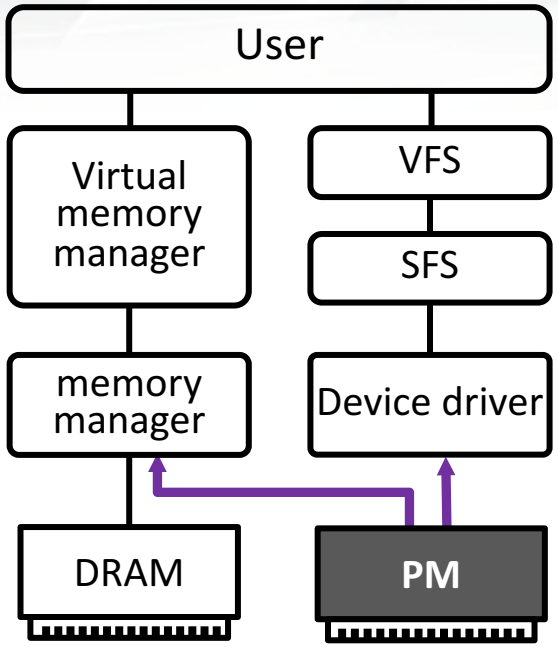
Weakness of Memorage

- **Large runtime overhead**
  - Process of changing free blocks into pages proceeds at runtime
- **Memory hot-plug feature is required**
- **Overlooks the issue of consistency**
  - Consistency of page allocator is not considered
  - Memory leak can occur with faults



Related work

- **Dynamic usage of PM**



Memorage[1] – ICS 2013
: Working memory expansion

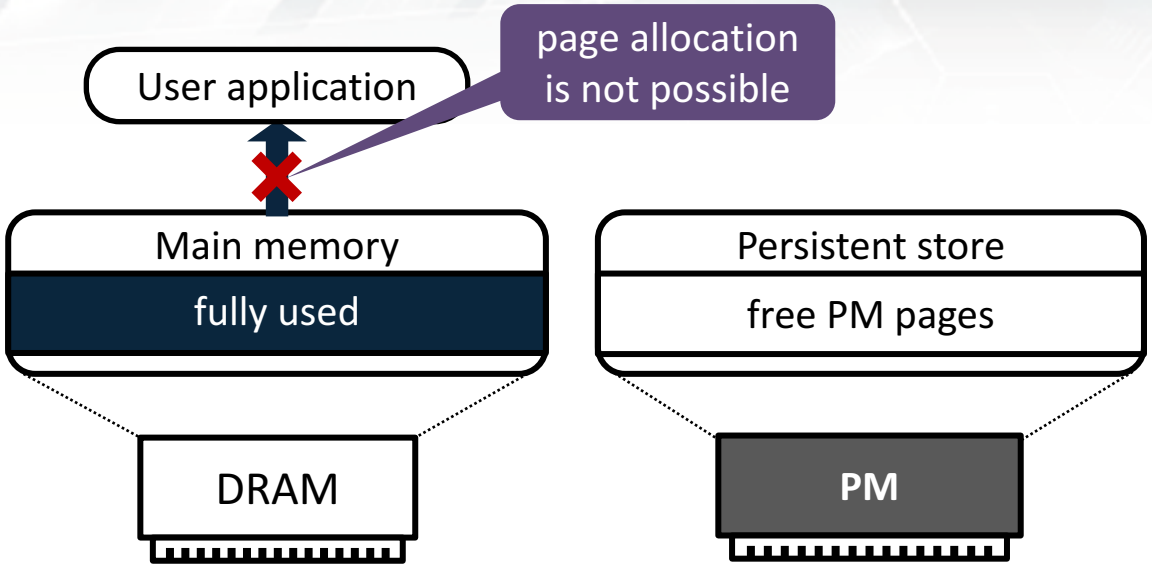
pVM[2] – EuroSys 2016
: Direct allocation of PM page to working memory

1) Jung, J. Y. and Cho, S. "Memorage: Emerging persistent ram based malleable main memory and storage architecture." In Proceedings of the International ACM Conference on International Conference on Supercomputing (ICS) (2013).

2) Kannan, S., Ada, G., and Karsten, S. "pVM: persistent virtual memory for efficient capacity scaling and object storage." In Proceedings of the European Conference on Computer Systems (EuroSys) (2016).

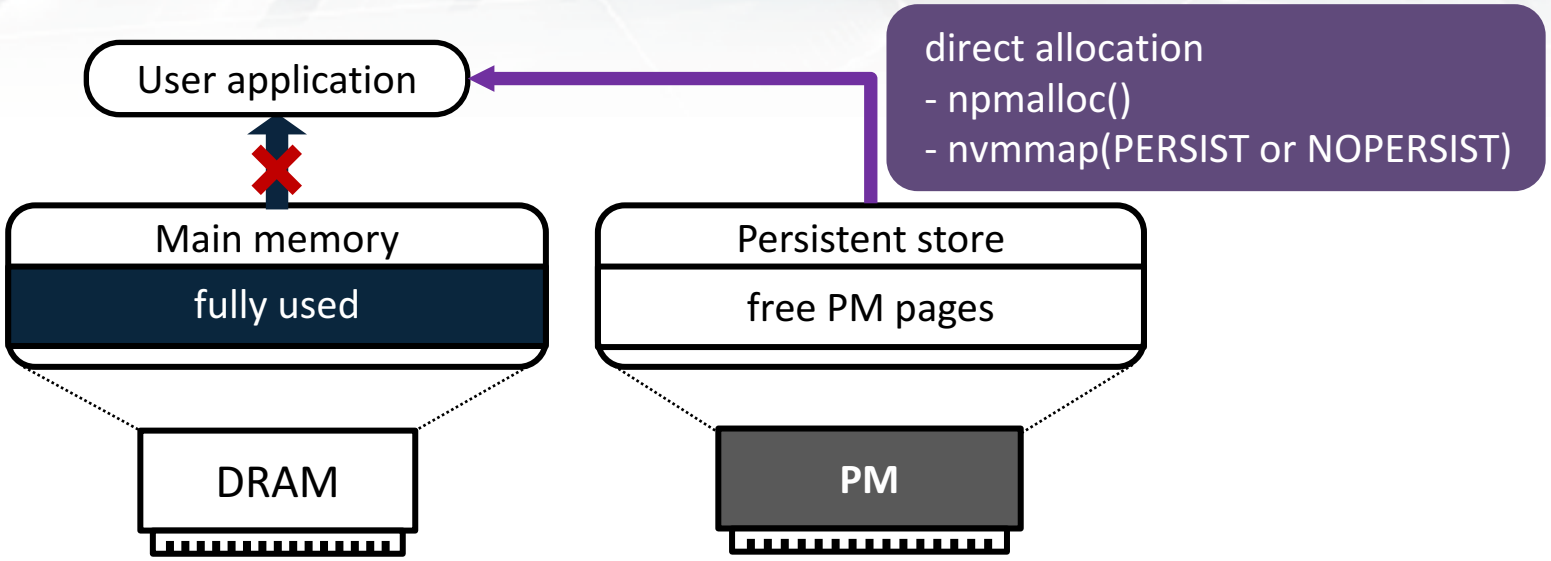
pVM

- **When DRAM runs out**



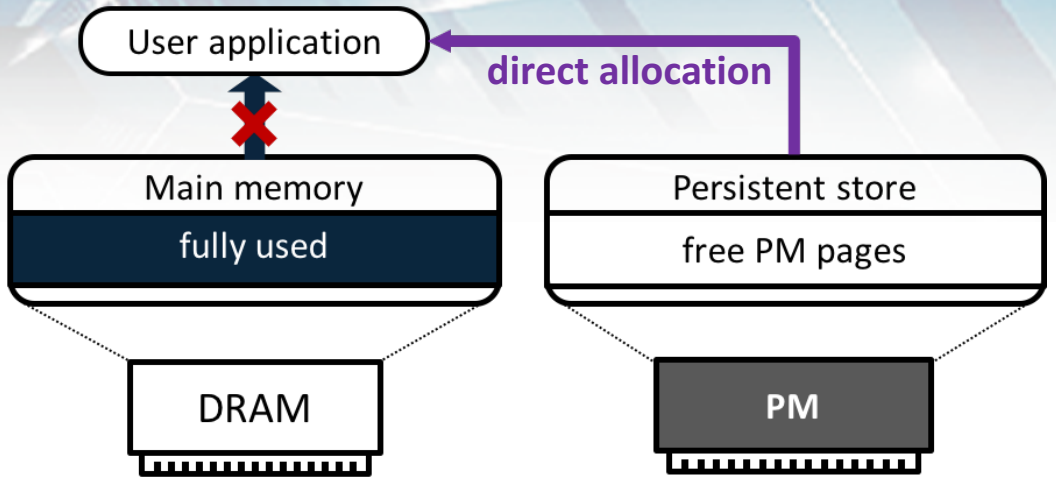
pVM

- **When DRAM runs out**



Weakness of pVM

- **Requires application modification**
  - Explicit library function calls need to be made
    - npmalloc(), nvmmap()
- **Similar approach to previous work**
  - NV-Heap
  - Mnemosyne
  - NVM Duet

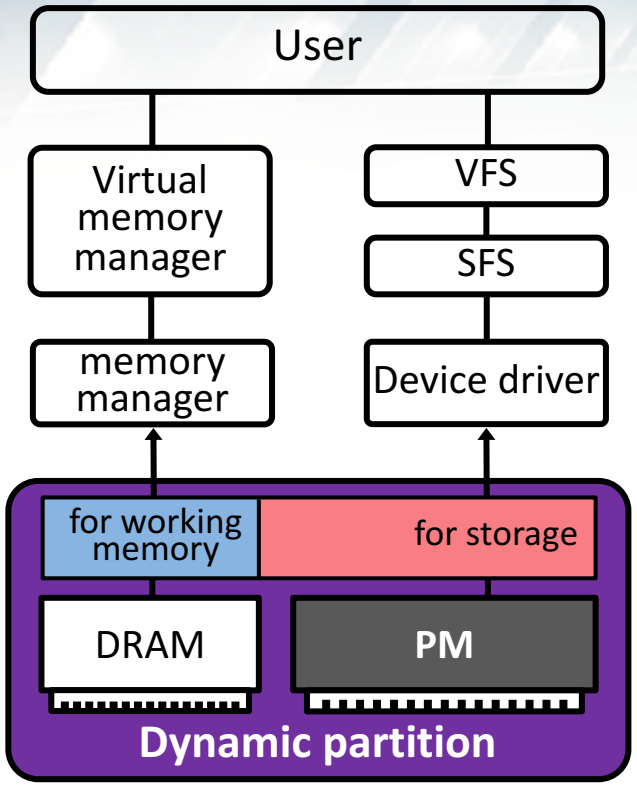


Comparison of related works

strength
weakness

	Swap	Memorage	pVM
Goal	free up working memory space	memory expansion	persistent store
Memory-Storage division	fixed	fixed	fixed
Consistency	not considered	not considered	yes
Transparency	yes	yes	no
Runtime overhead	yes	yes	no
Direct access to data	no	yes	yes

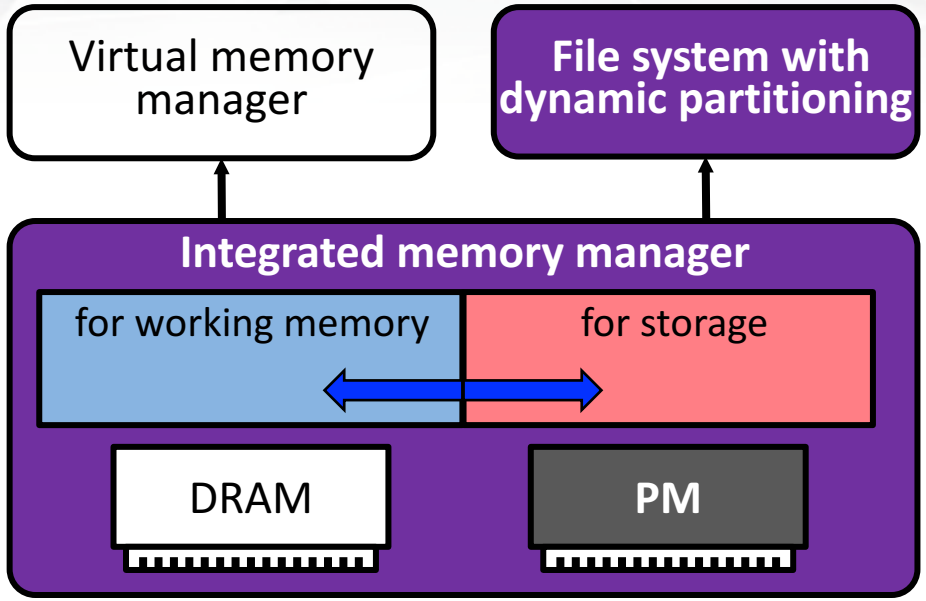
SAY-Go design goals



- **Seamless integration**
  - PM role is dynamically adjusted as need be
- **Transparency**
  - Automatic scaling of working memory
  - Without modification of the application
- **Consistency**
  - Ensure consistency of the PM allocator
  - Prevent memory leaks

SAY-Go components

- **Consists of two parts**

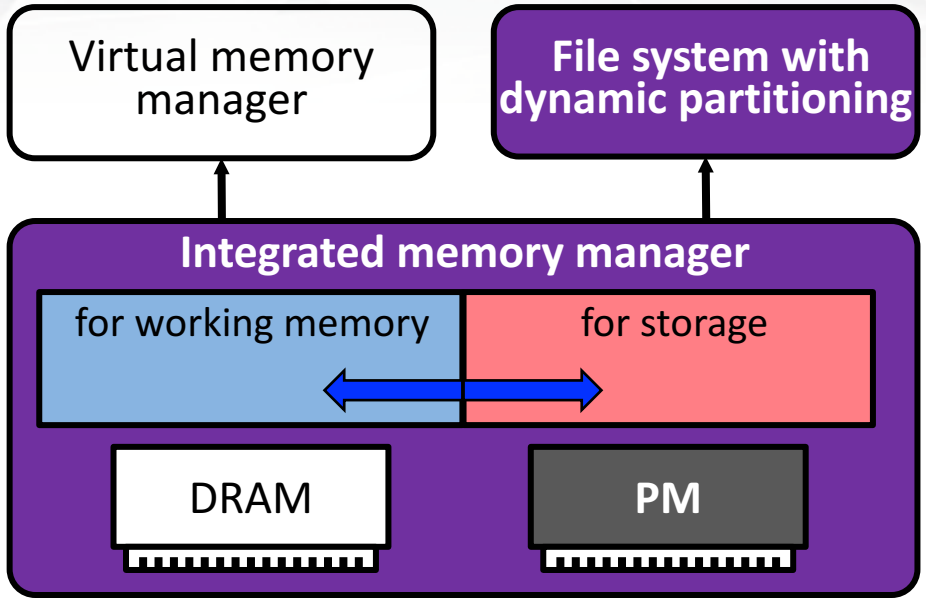


1. File system with dynamic partition support
   - Support dynamic layout
   - Region managed in page unit

2. Integrated memory manager
   - Page allocator for both working memory and storage
   - Ensure consistency of memory allocator

Layout

- **Consists of two parts**



1. File system with dynamic partition support
   - Support dynamic layout
   - Region managed in page unit

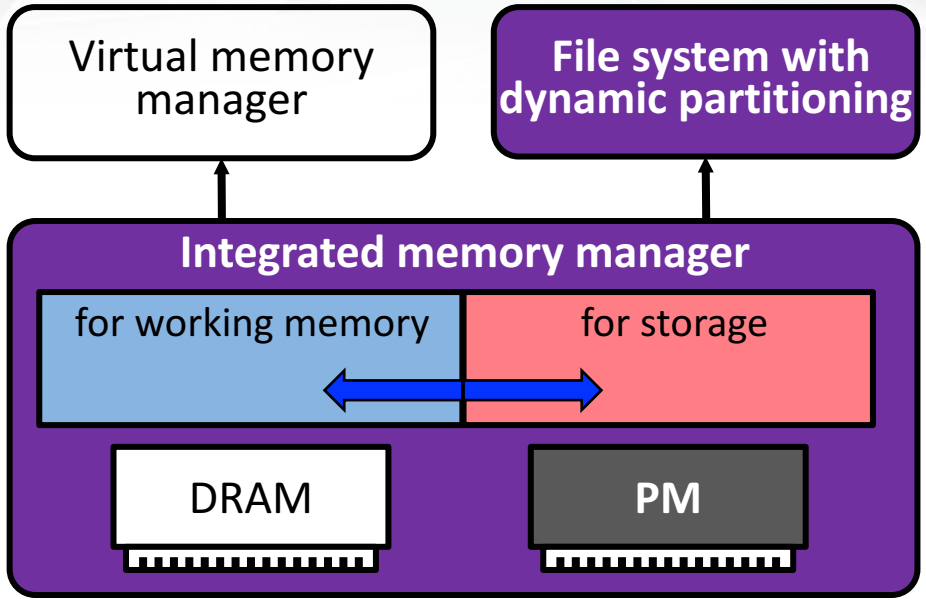
   Dynamic Partition File System (DPFS), in progress

2. Integrated memory manager
   - Page allocator for both working memory and storage
   - Ensure consistency of memory allocator

   Persistent Memory Buddy (PMB), HotStorage '18

Layout

- **Consists of two parts**



1. File system with dynamic partition support
   - Support dynamic layout
   - Region managed in page unit

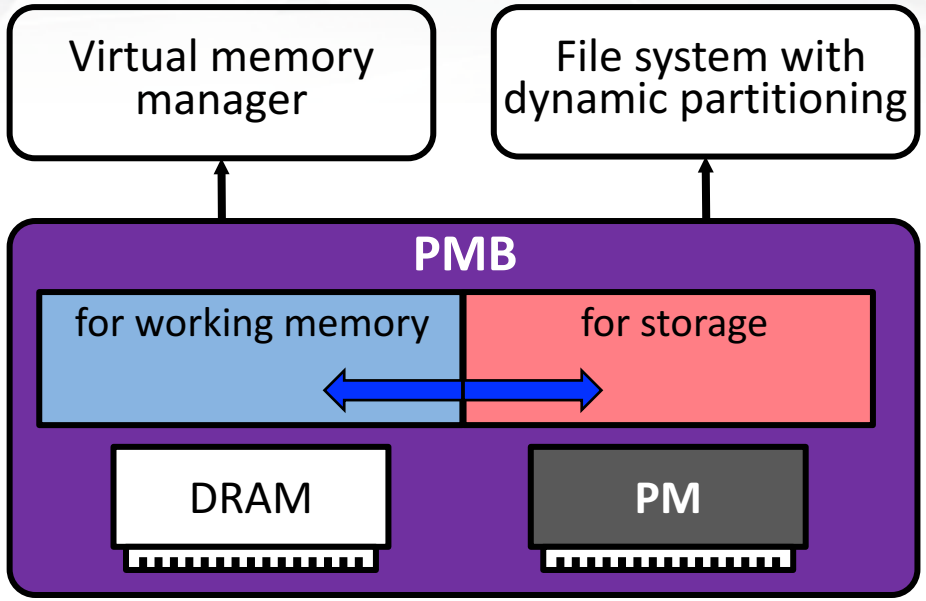
   Dynamic Partition File System (DPFS), in progress

2. Integrated memory manager
   - Page allocator for both working memory and storage
   - Ensure consistency of memory allocator

   Persistent Memory Buddy (PMB), **HotStorage '18**

Implemented in Linux

PMB implementation features

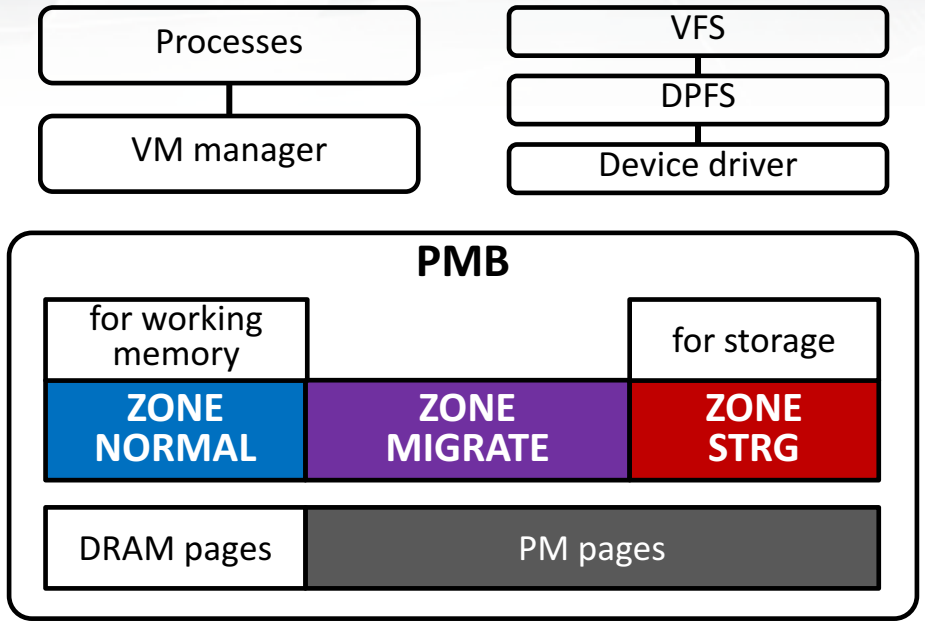


Persistent Memory Buddy (PMB)

- **Seamless integration**
  - PM role is dynamically adjusted as need be
- **Transparency**
  - Automatic scaling of working memory
  - Without modification of the application
- **Consistency**
  - Ensure consistency of the PM allocator
  - Prevent memory leaks

PMB designed with SAY-Go goal in mind

PMB Layout in Linux



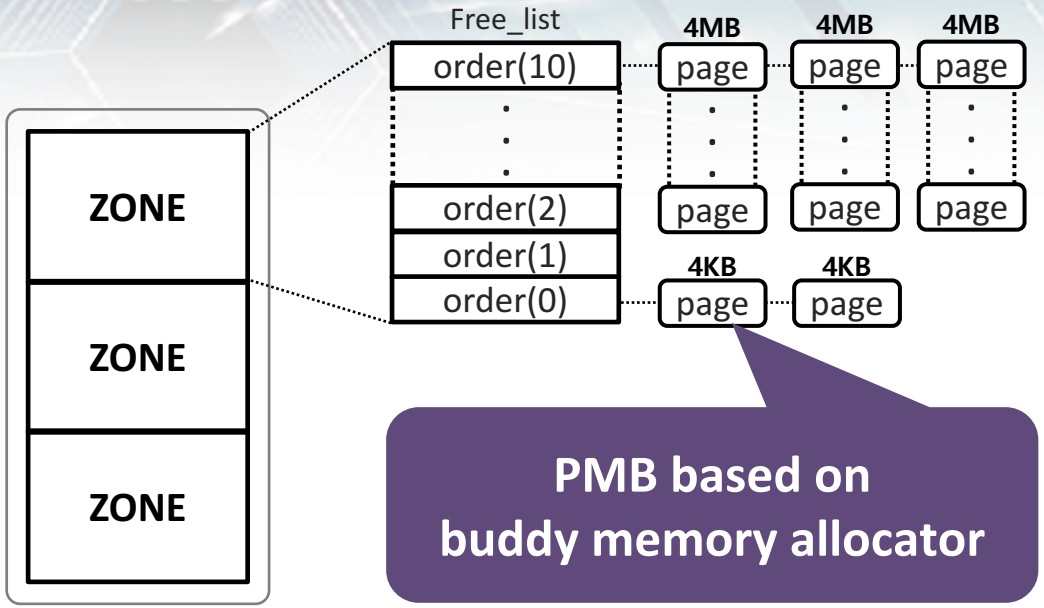
- DRAM and PM are managed in ZONE units
  - **ZONE_NORMAL**
  - **plus 2 new ZONEs**
    - * **ZONE_MIGRATE and ZONE_STRG**
- Each ZONE has free pages
  - **buddy memory allocator**

Buddy memory allocator

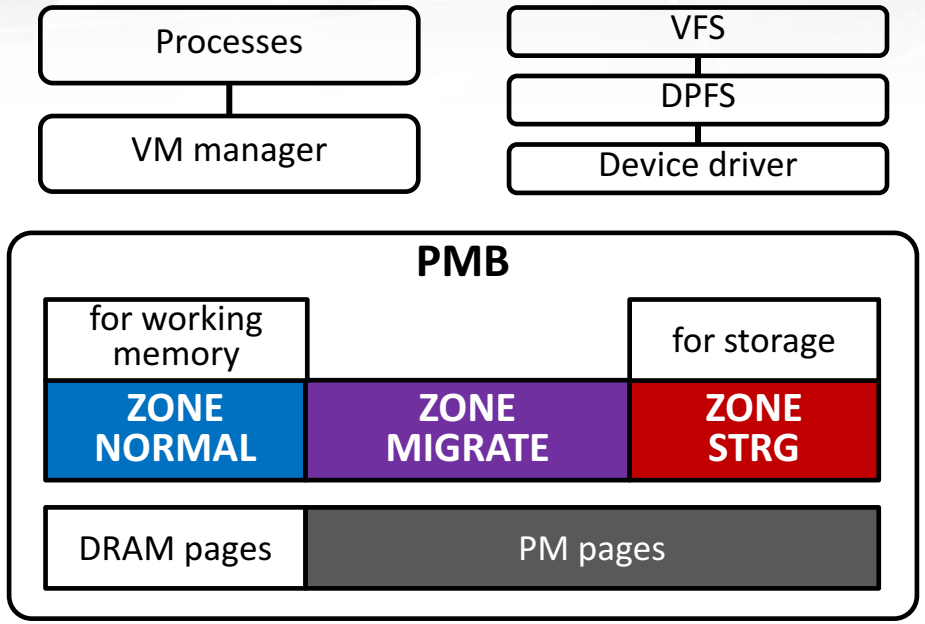
- **Characteristics in Linux**
  - Unsorted
  - Circular linked list
  - Push and pop operation

    occurs only at the end of the list
  - Push and pop operations

    are implemented by atomic operations

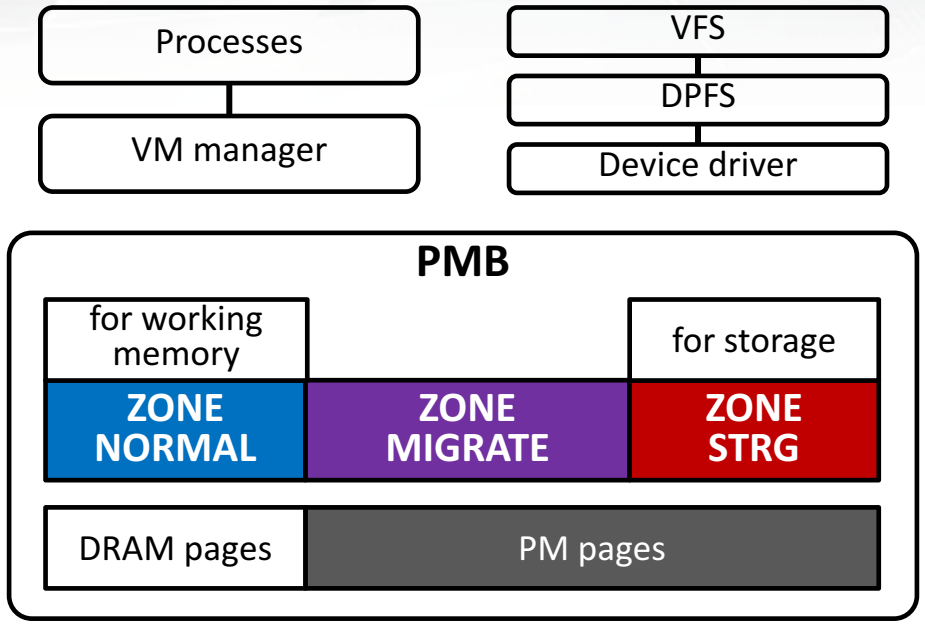


PMB Layout in Linux



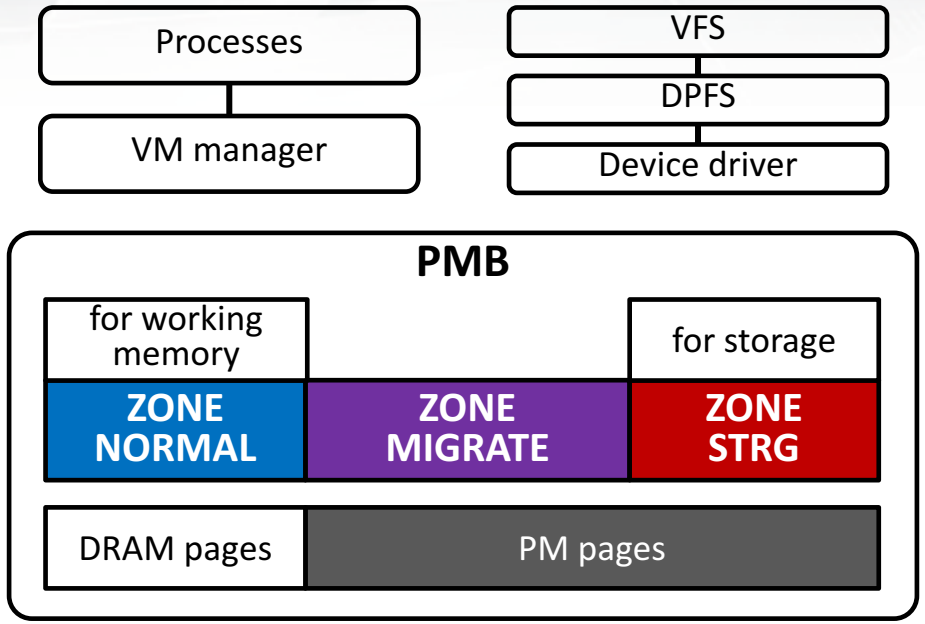
- DRAM and PM are managed in ZONE units
  - **ZONE_NORMAL**
  - **plus 2 new ZONEs**
    - * **ZONE_MIGRATE and ZONE_STRG**
- Each ZONE has free pages
  - **buddy memory allocator**
- Each ZONE has different role

Role of each ZONE



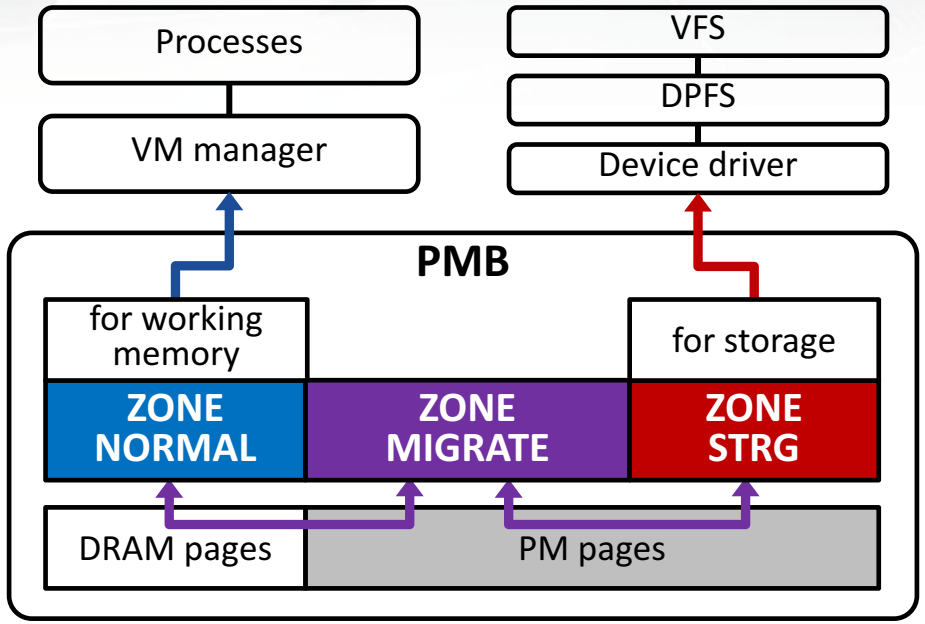
- **ZONE_NORMAL**
  - DRAM is covered by this ZONE
- **ZONE_STRG**
  - Minimum storage area
  - Persistent metadata saved to this area
    - Persistent page descriptor
    - Bitmap that maintains the allocation state
    - Log table for PMB consistency

Role of each ZONE



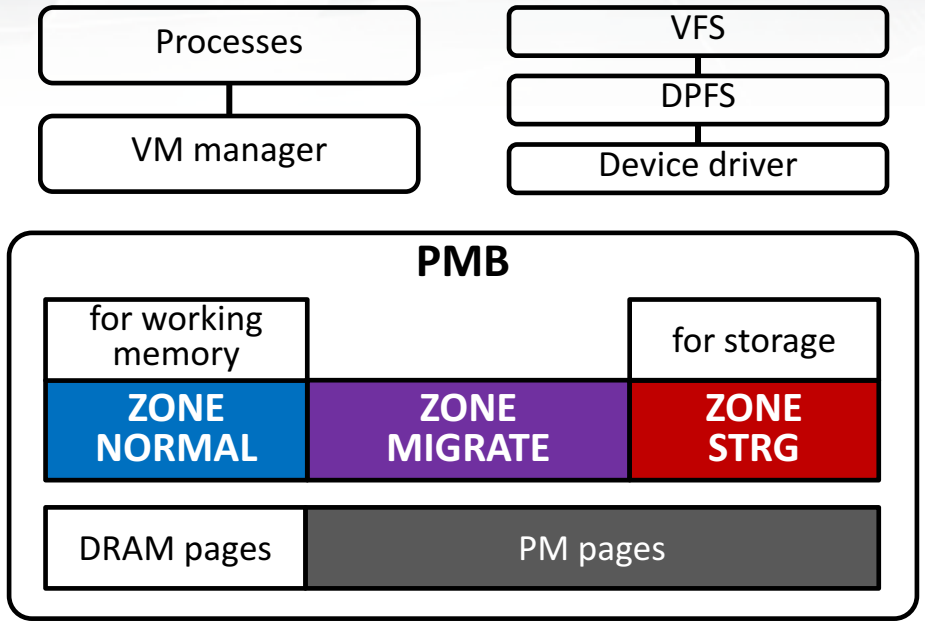
- **ZONE_MIGRATE**
  - Key component in PMB
  - Either allocated depending on its use
    * as persistent (storage) page
    * as non-persistent (memory) page

Page allocation



- **ZONE_NORMAL**
  - **Allocate working memory**
- **Storage allocated from ZONE_STRG**
  - **Allocate storage**
- **ZONE_MIGRATE**
  - **Allocate PM pages to both zones**
  - Allocation done using the existing interface
  - Allocation done by kernel thread in the background

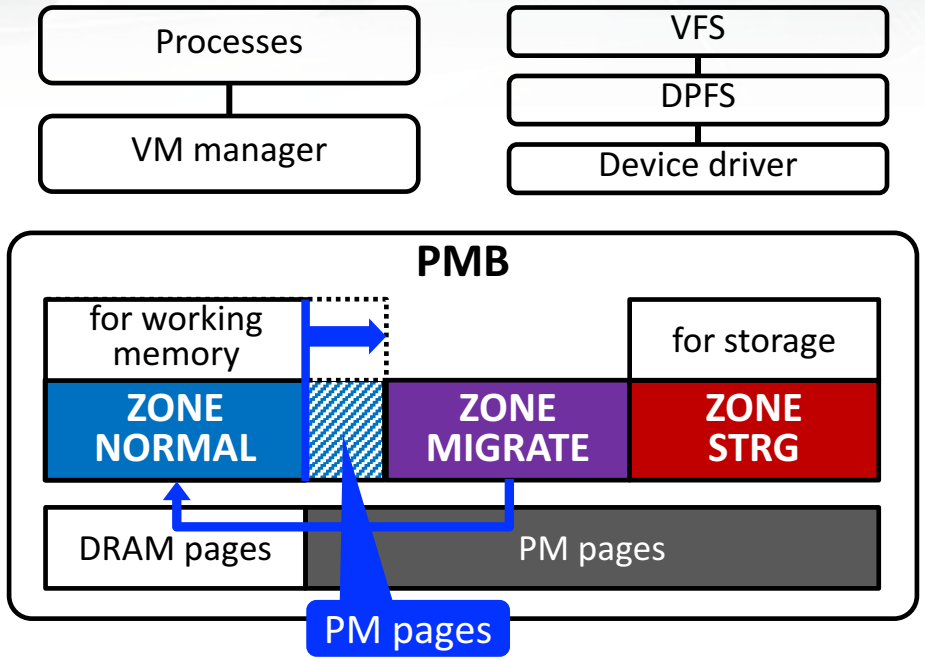
ZONE capacity



- **ZONE_NORMAL**
  - Initially, only DRAM pages
  - Will grow with demand
- **ZONE_STRG**
  - Initially, essential storage area
  - Will grow with demand

Page movement

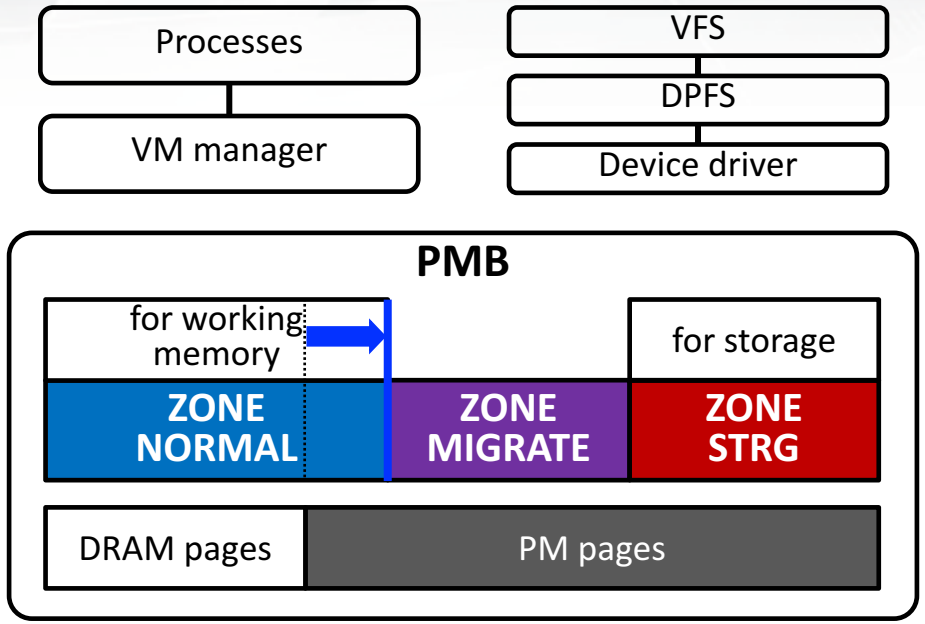
- **ZONE NORMAL expansion**



- **ZONE_NORMAL**
  - Initially, only DRAM pages
  - Will grow with demand
  - **PM page migration**

Page movement

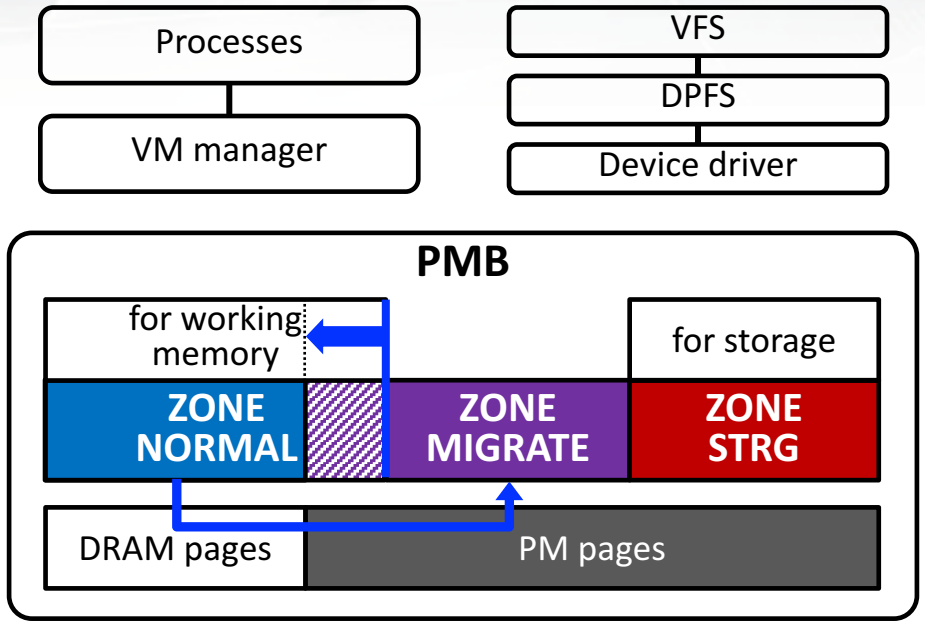
- **ZONE NORMAL expansion**



- **ZONE_NORMAL**
  - Initially, only DRAM pages
  - Will grow with demand
  - **PM page migration**
  - **Increased ZONE_NORMAL size**

Page movement

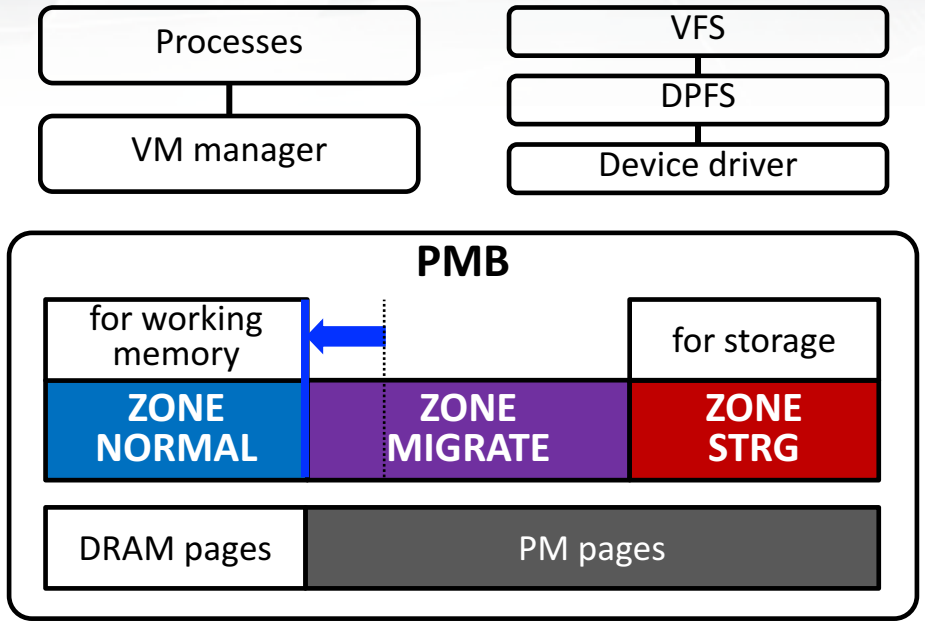
- **ZONE NORMAL retraction**



- **ZONE_NORMAL**
  - Initially, only DRAM pages
  - Will grow with demand
  - **PM page retrieval**

Page movement

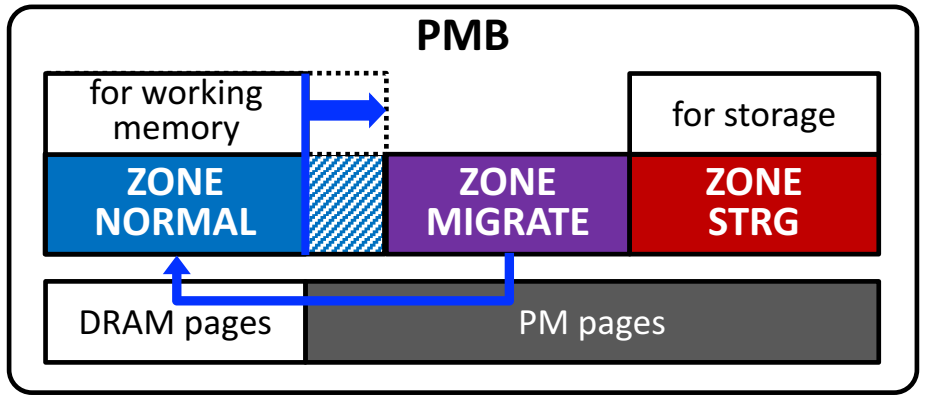
- **ZONE NORMAL retraction**

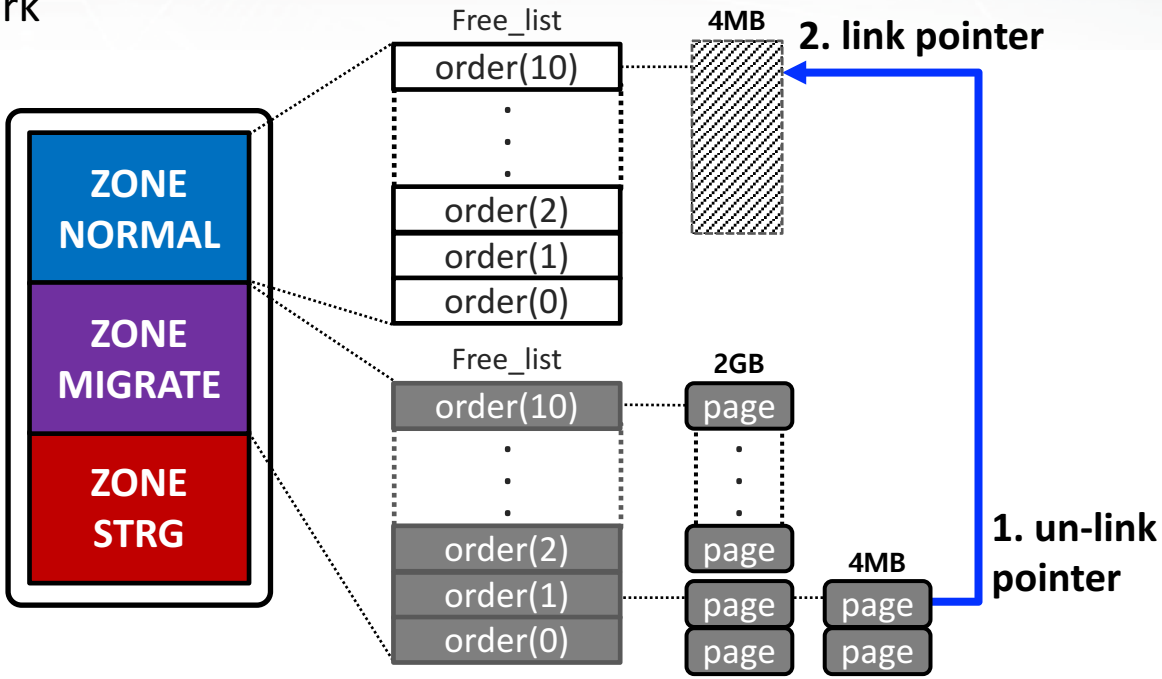


- **ZONE_NORMAL**
  - Initially, only DRAM pages
  - Will grow with demand
  - **PM page retrieval**
  - **Reduced ZONE_NORMAL size**

Page migration

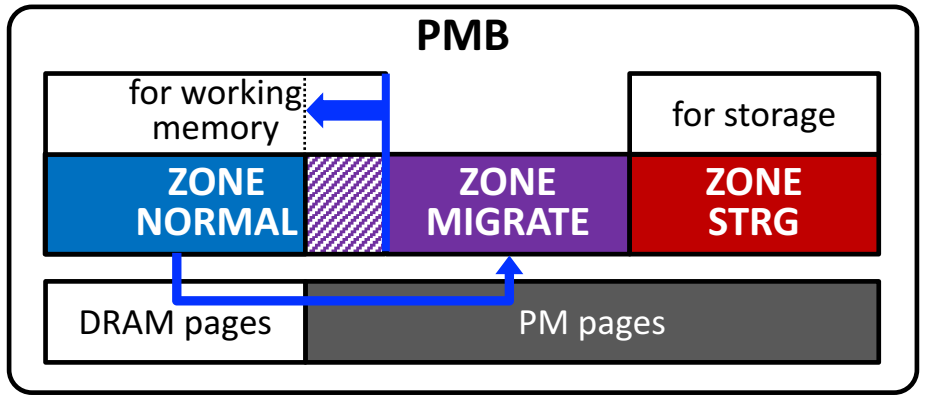
- ZONE_MIGRATE to ZONE_NORMAL
  - Both zones managed with Buddy memory allocator
  - When number of free pages falls below watermark
  - Transfers 4MB contiguous pages
    - **simple pointer manipulation**

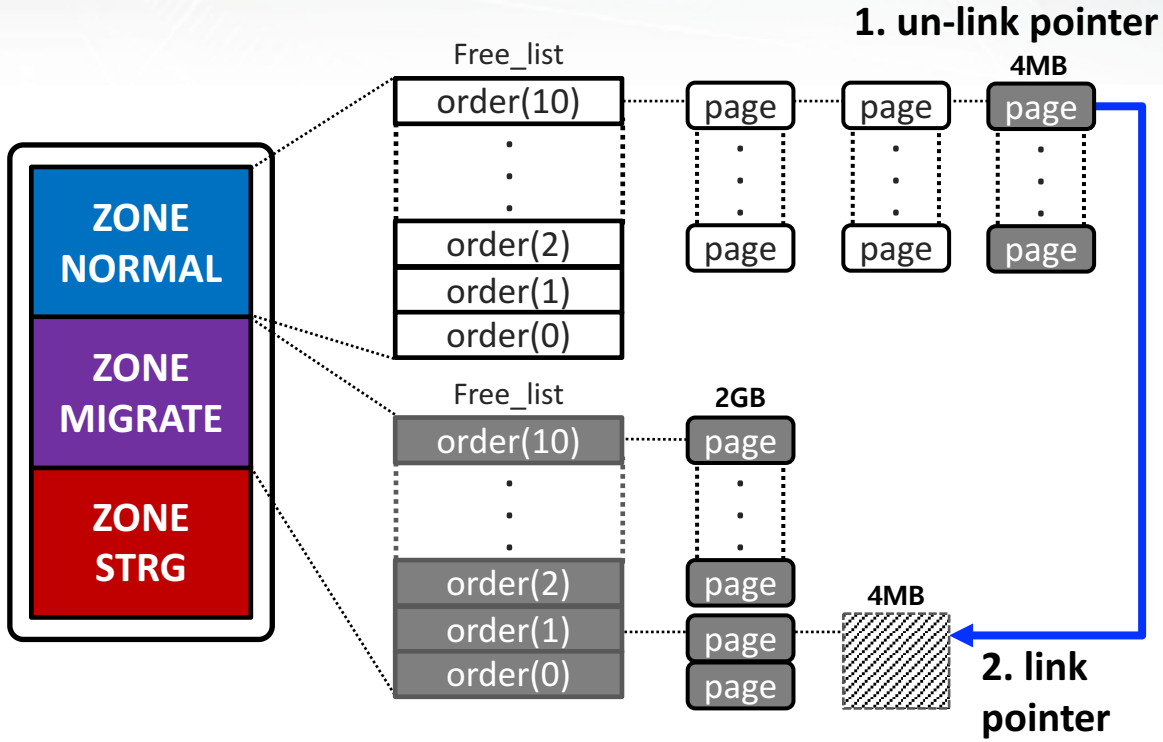




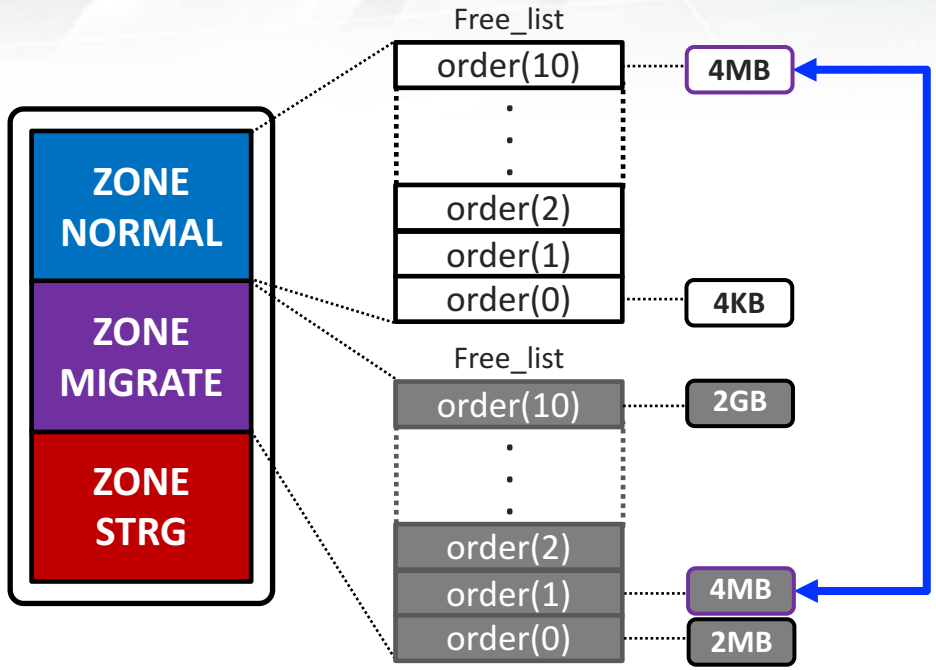
Page retrieval

- **ZONE_NORMAL to ZONE_MIGRATE**
  - No watermark
  - Retrieved released by application
    - **simple pointer manipulation**





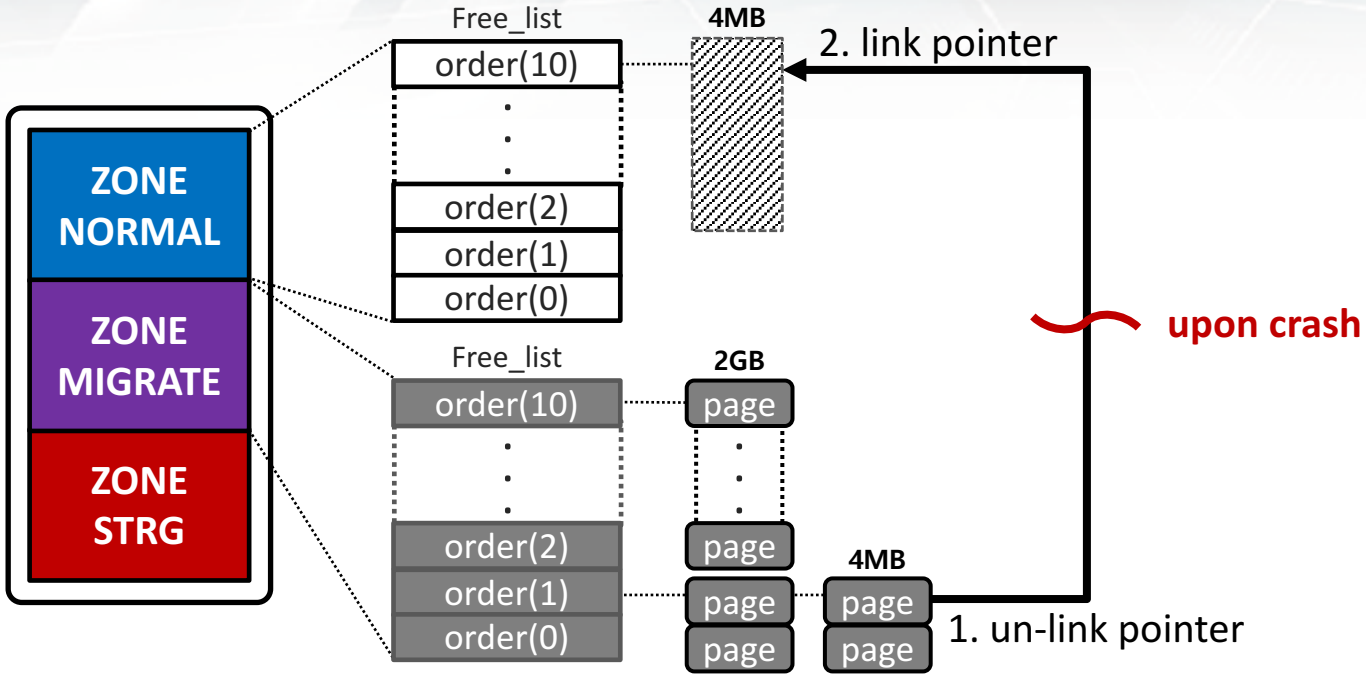
Management unit



- PMB takes the 2MB huge page size
- Contiguous memory space (2^0 ~ 2^{10} pages)
  - ZONE_NORMAL: 4KB ~ 4MB
  - ZONE_MIGRATE, ZONE_STRG: 2MB ~ 2GB
- Page movement unit is 4MB contiguous page
  - Performed between ZONE_NORMAL (order 10) and ZONE_MIGRATE (order 1)

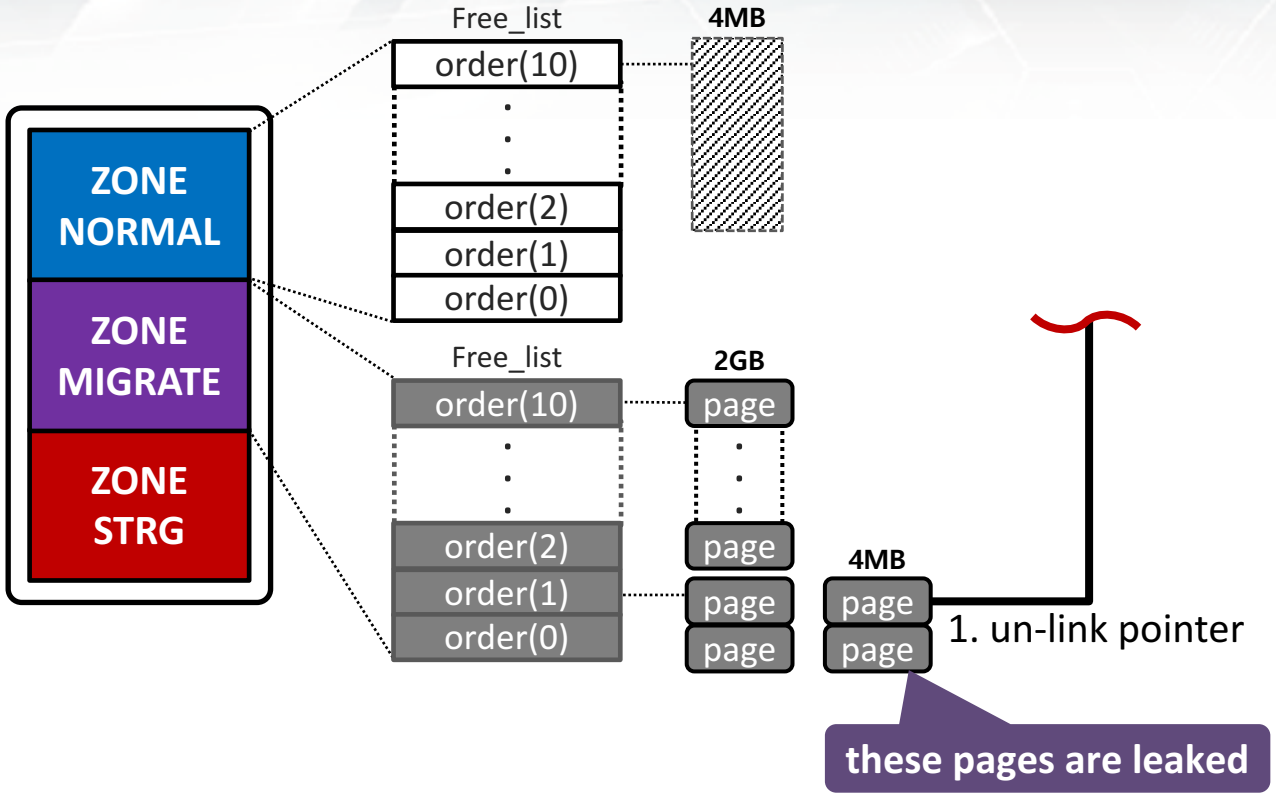
Consistency

- **Memory leak may occur in page moving process**



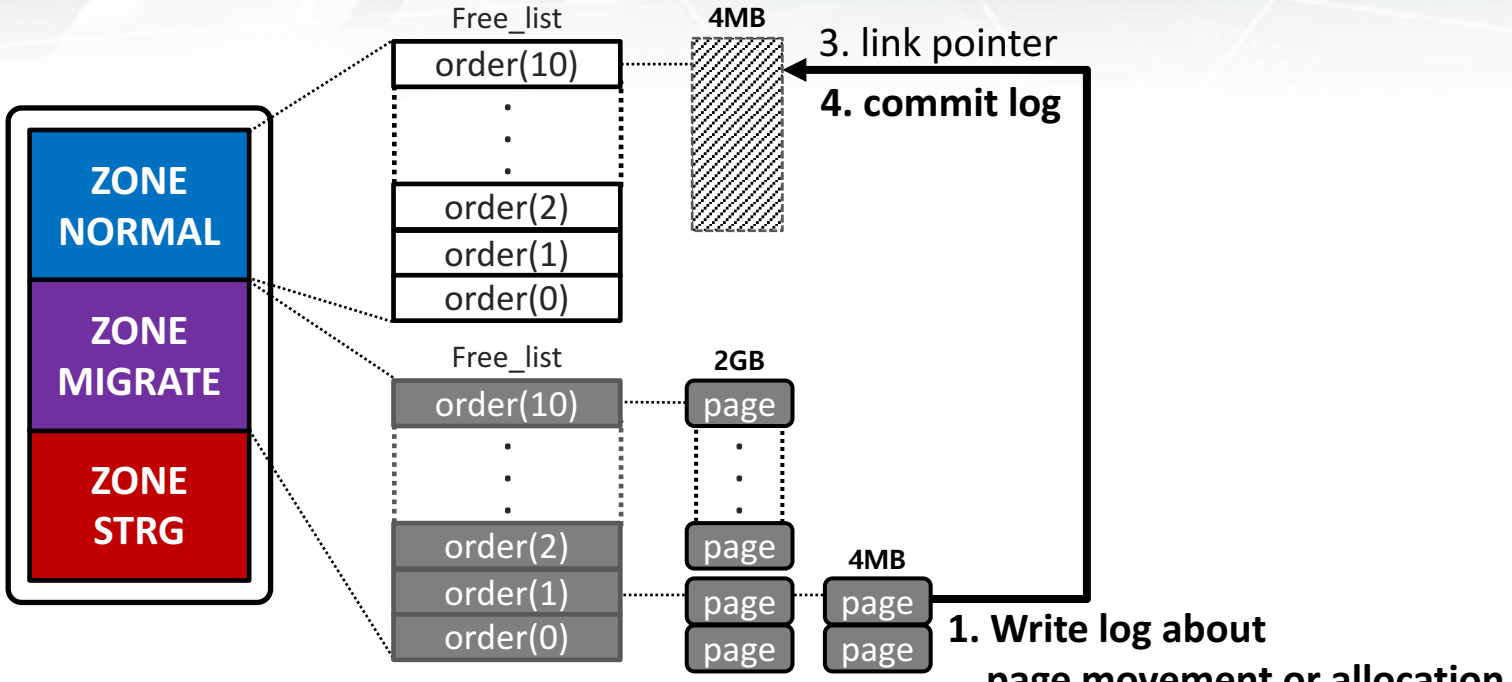
Consistency

- **Memory leak may occur in page moving process**



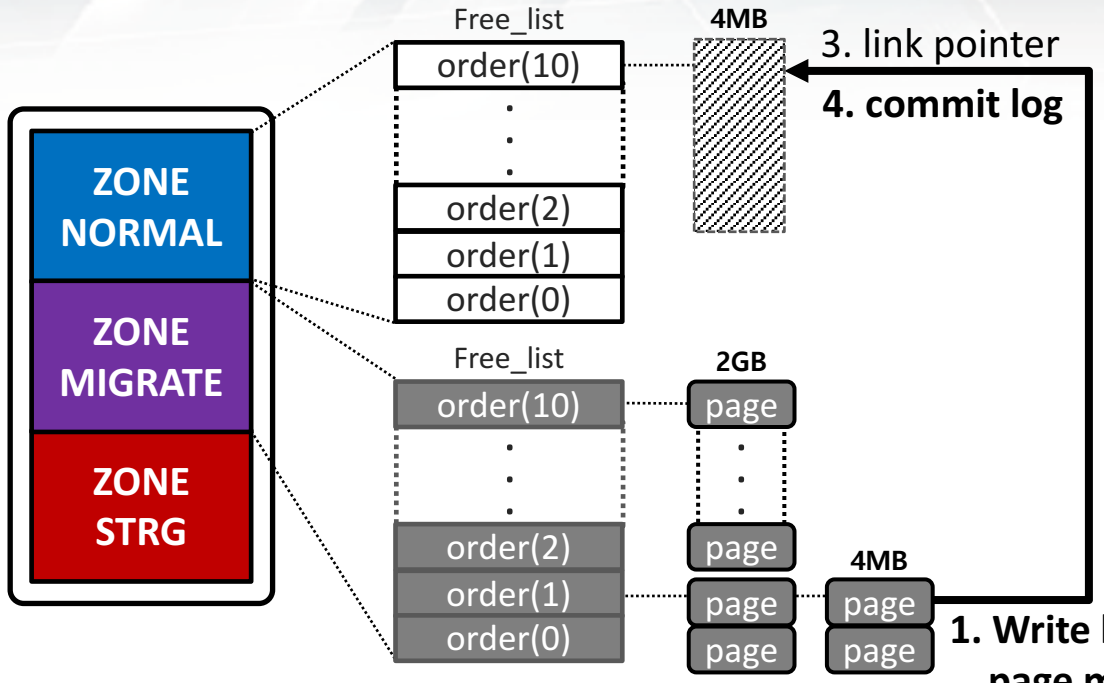
Consistency

- **Use logging to prevent memory leak**



Consistency

- **Use logging to prevent memory leak**



- Log entry fields
  - Page structure address
  - Operation type
  - Destination of page
  - Commit bit
- Log entry size < 64bit
- Use *mfence* and *clflush* instructions for logging

Previous work and SAY-Go

strength
weakness

	Swap	Memorage	pVM	SAY-Go
Goal	free up working memory space	memory expansion	persistent store	**efficient use of resources**
Memory-Storage division	fixed	fixed	fixed	**dynamic**
Consistency	not considered	not considered	yes	**yes**
Transparency	yes	yes	no	**yes**
Runtime overhead	yes	yes	no	**no**
Direct access to data	no	yes	yes	**yes**

Evaluation

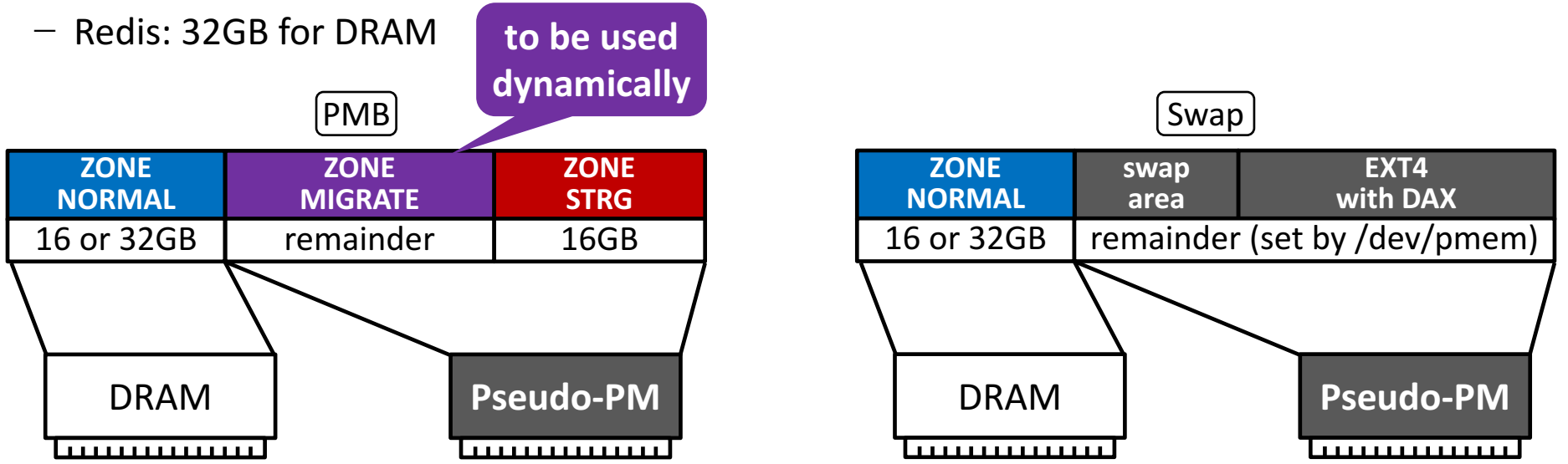
- **Test platform**
  - Machine specification
    - Intel Xeon E5-2620v3 2.4GHz (24 cores)
    - 256 DRAM
      - Divide to DRAM and Pseudo-PM
    - Linux v4.11.1

- **Workloads characteristics**

	FFT	Redis
Scale	Memory intensive application	In-memory database
Domain	Signal processing	Key-value store
Benchmark suite	Splash2x in Parsec 3.0	YCSB
Input	Native (largest)	1:1 (read:write)
Max memory usage	**12GB**	**20GB**

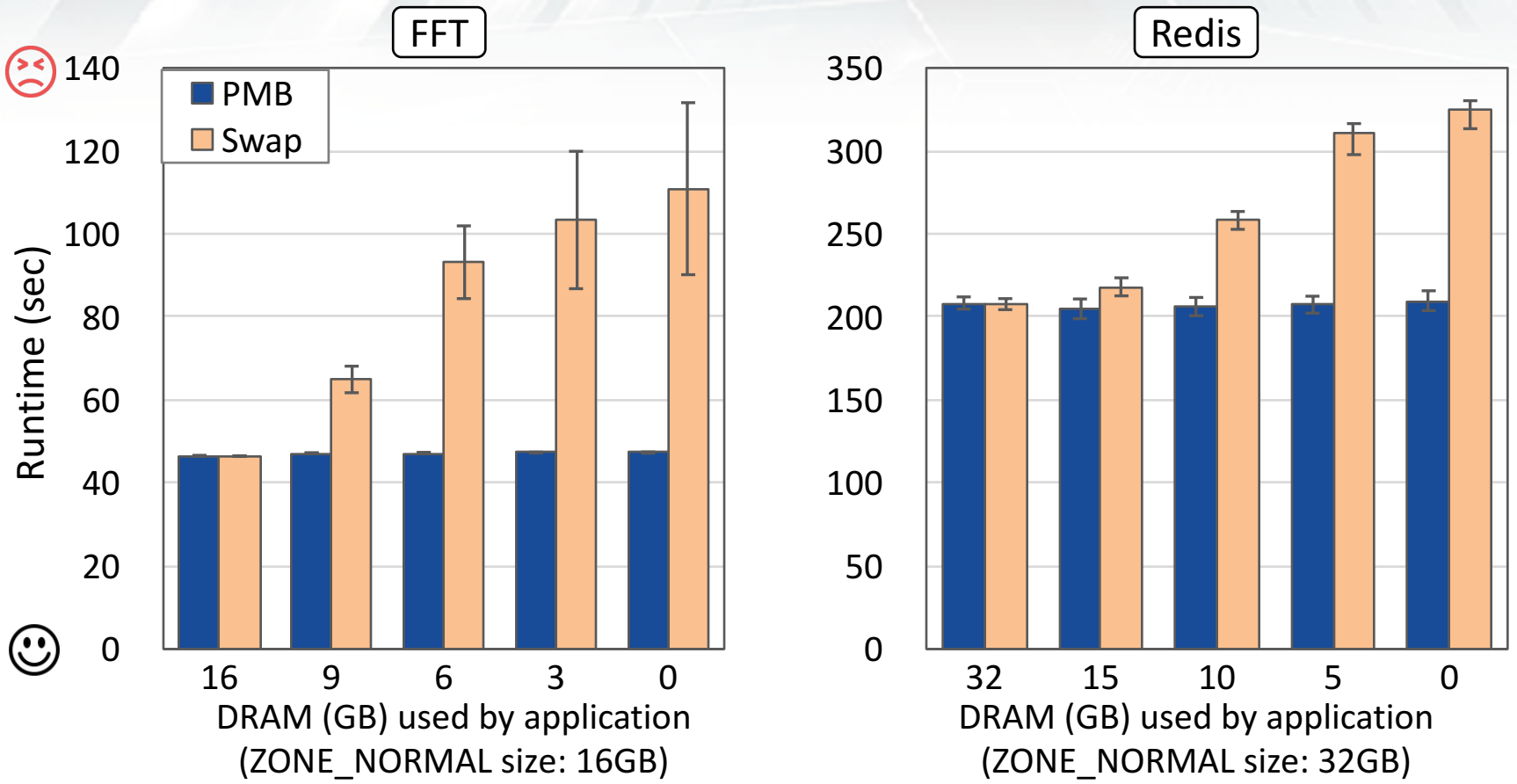
Evaluation

- **Comparison with swap**
  - Memory configuration
  - ZONE_NORMAL capacity varied based on workload
    - FFT: 16GB for DRAM
    - Redis: 32GB for DRAM



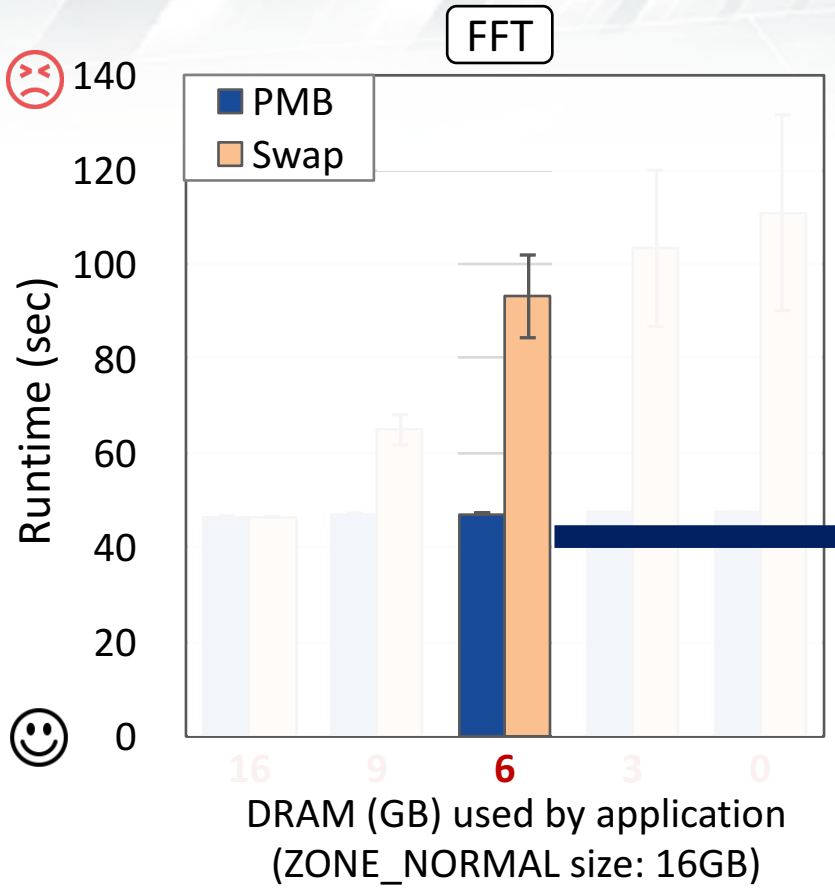
Evaluation

- **Comparison with swap**

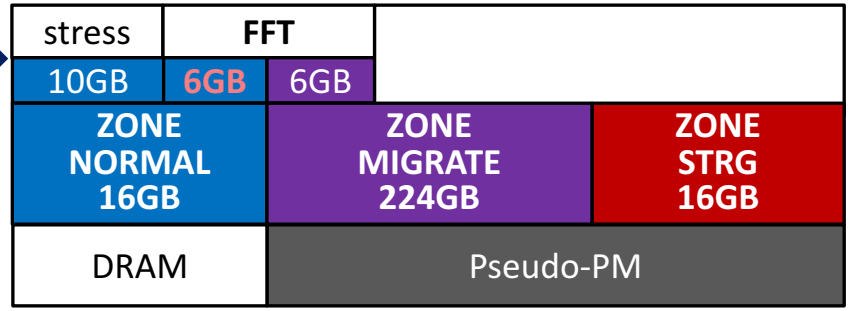


Evaluation

- **Measurement method: FFT example**

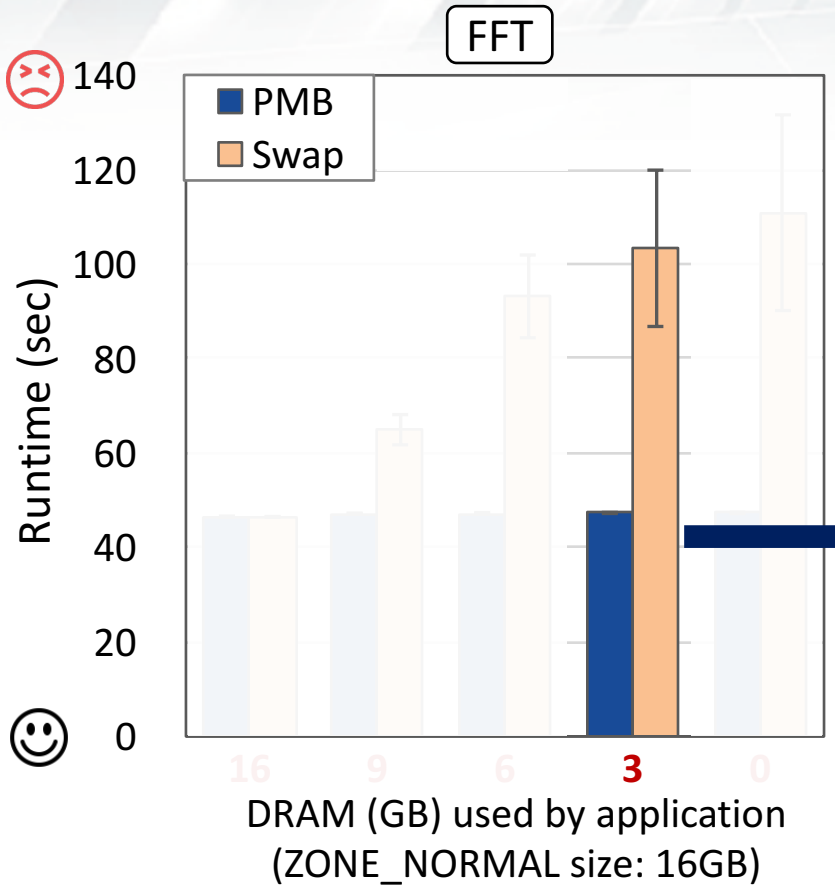


- FFT maximum working set size: 12GB
- Limit amount of main memory used by FFT
  - Use **stress tool** to consume main memory
- *x*-axis refers to the maximum DRAM used by FFT

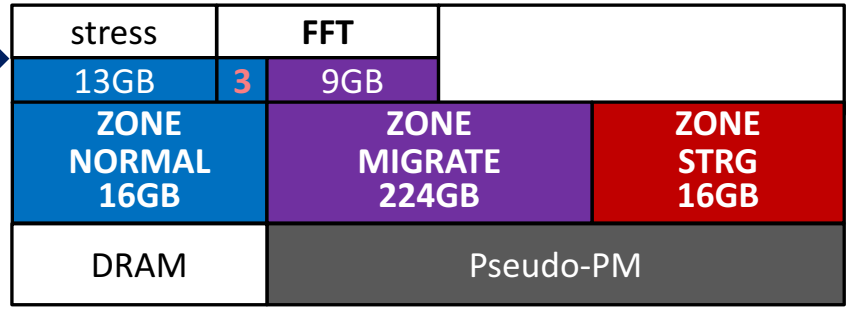


Evaluation

- **Measurement method: FFT example**

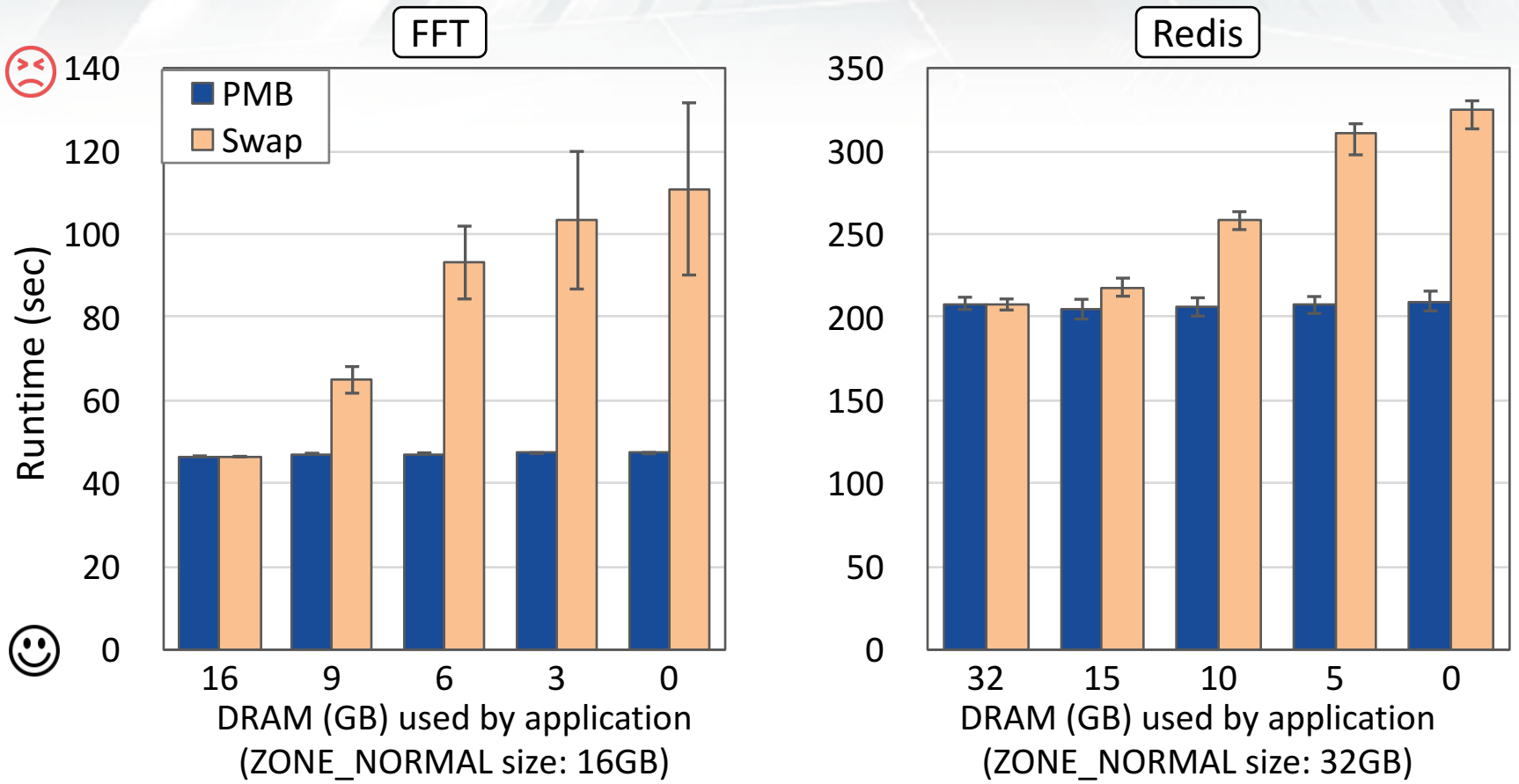


- FFT maximum working set size: 12GB
- Limit amount of main memory used by FFT
  - Use **stress tool** to consume main memory
- *x*-axis refers to the maximum DRAM used by FFT



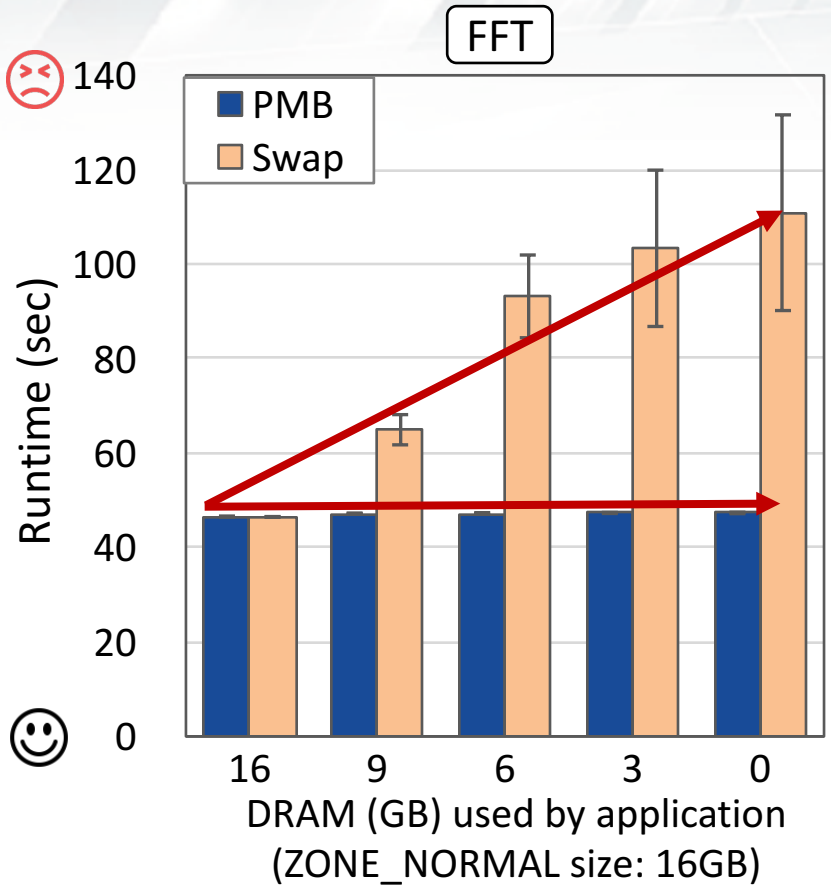
Evaluation

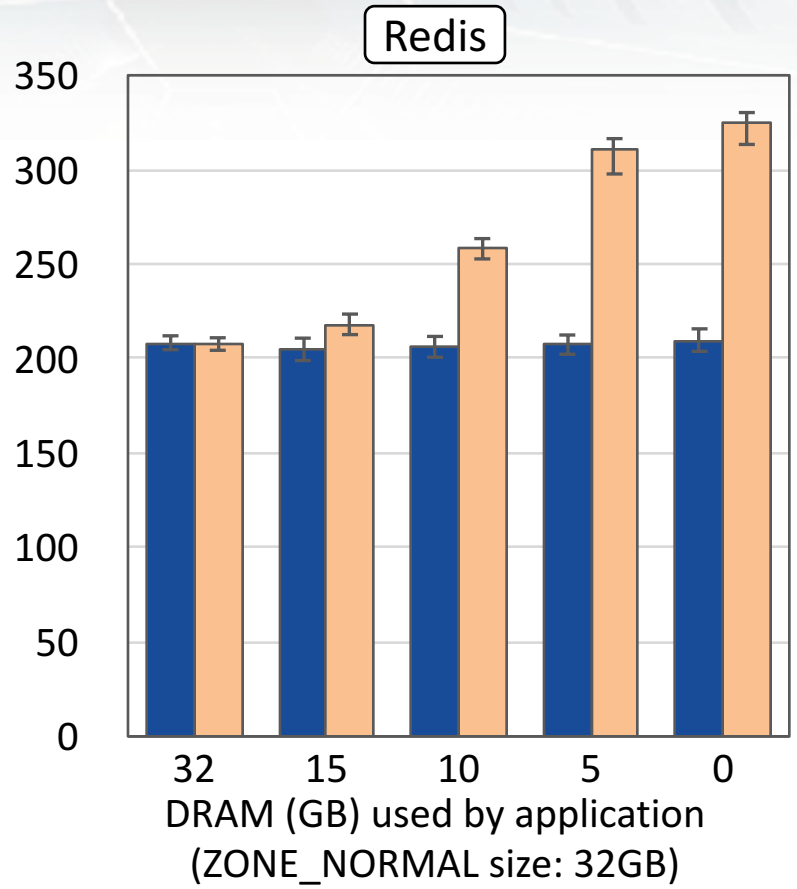
- **Comparison with swap**



Evaluation

- **Comparison with swap**



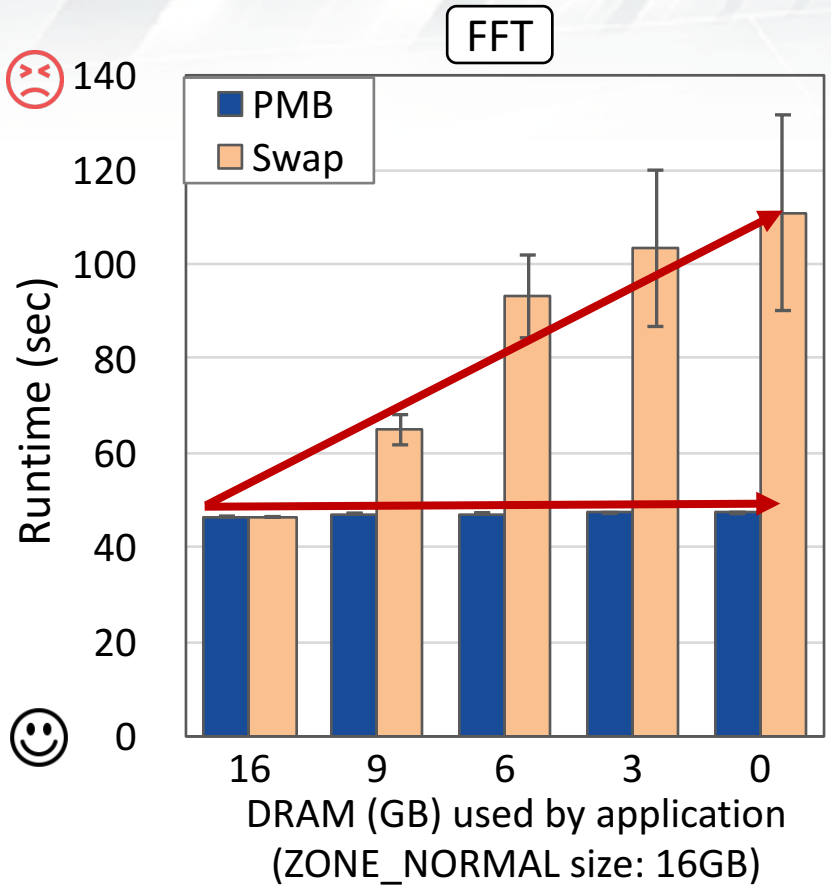


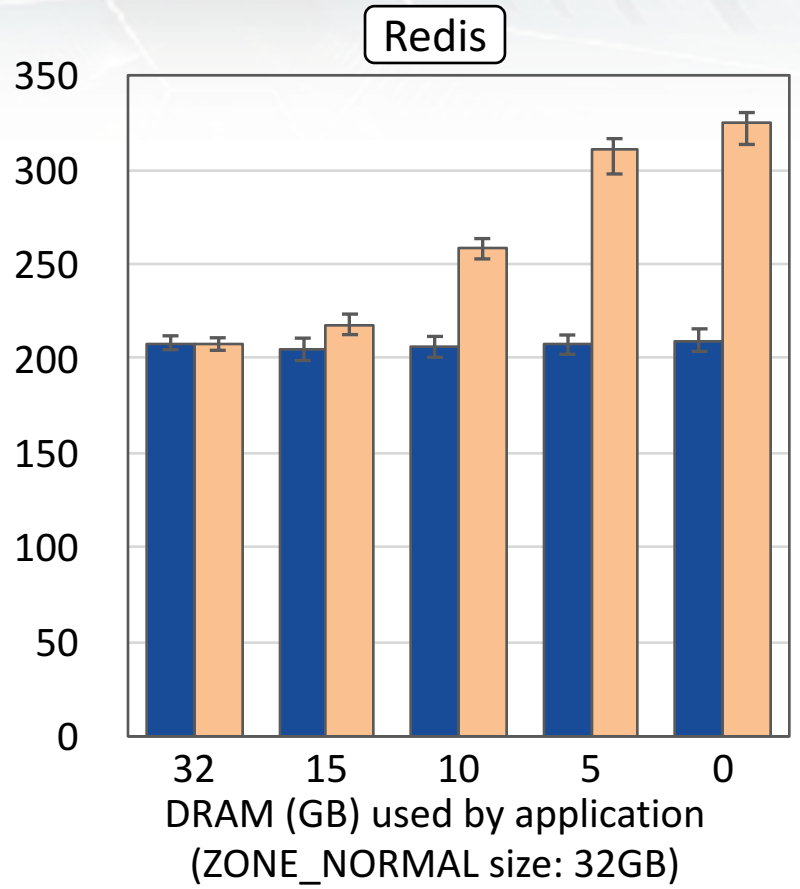
- Swap performance degrades considerably
- PMB performance remains stable

Two reasons behind results

Evaluation

- **Comparison with swap**

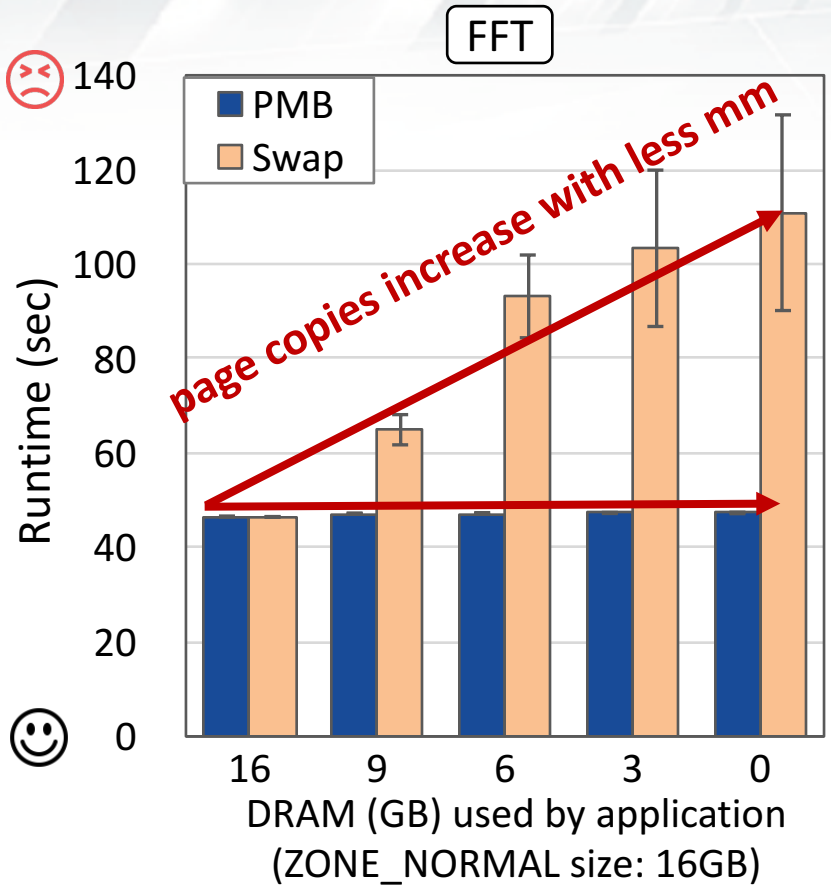


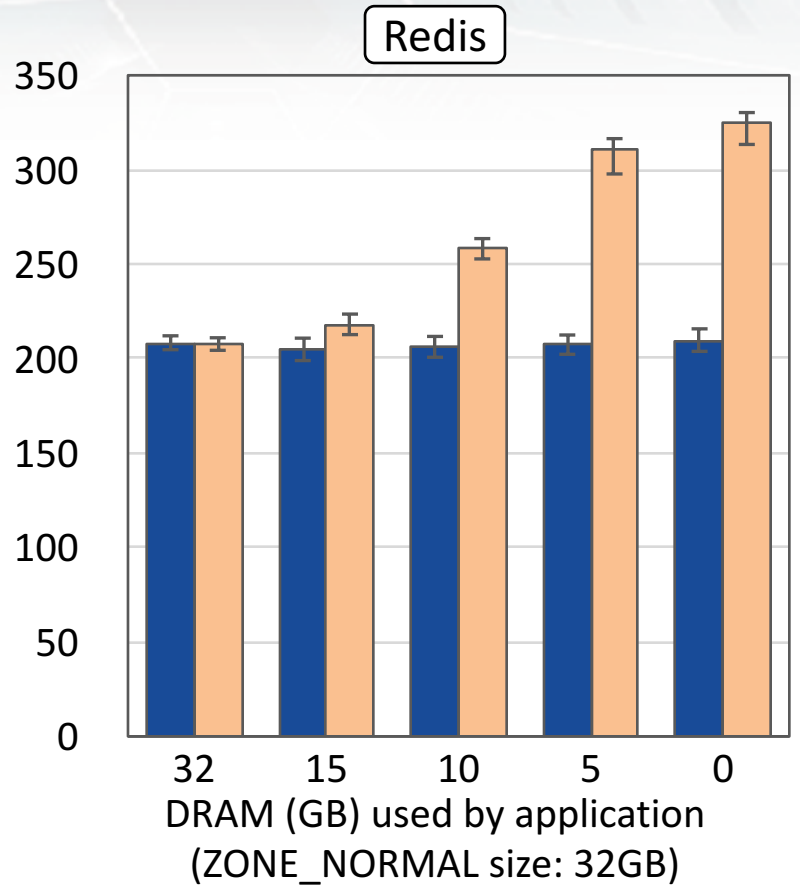


- **Mechanism overhead**
  - PMB migration and retrieval : ~ 400 ns
  - Swap-in / out : ~ 6 / 41 us

Evaluation

- **Comparison with swap**

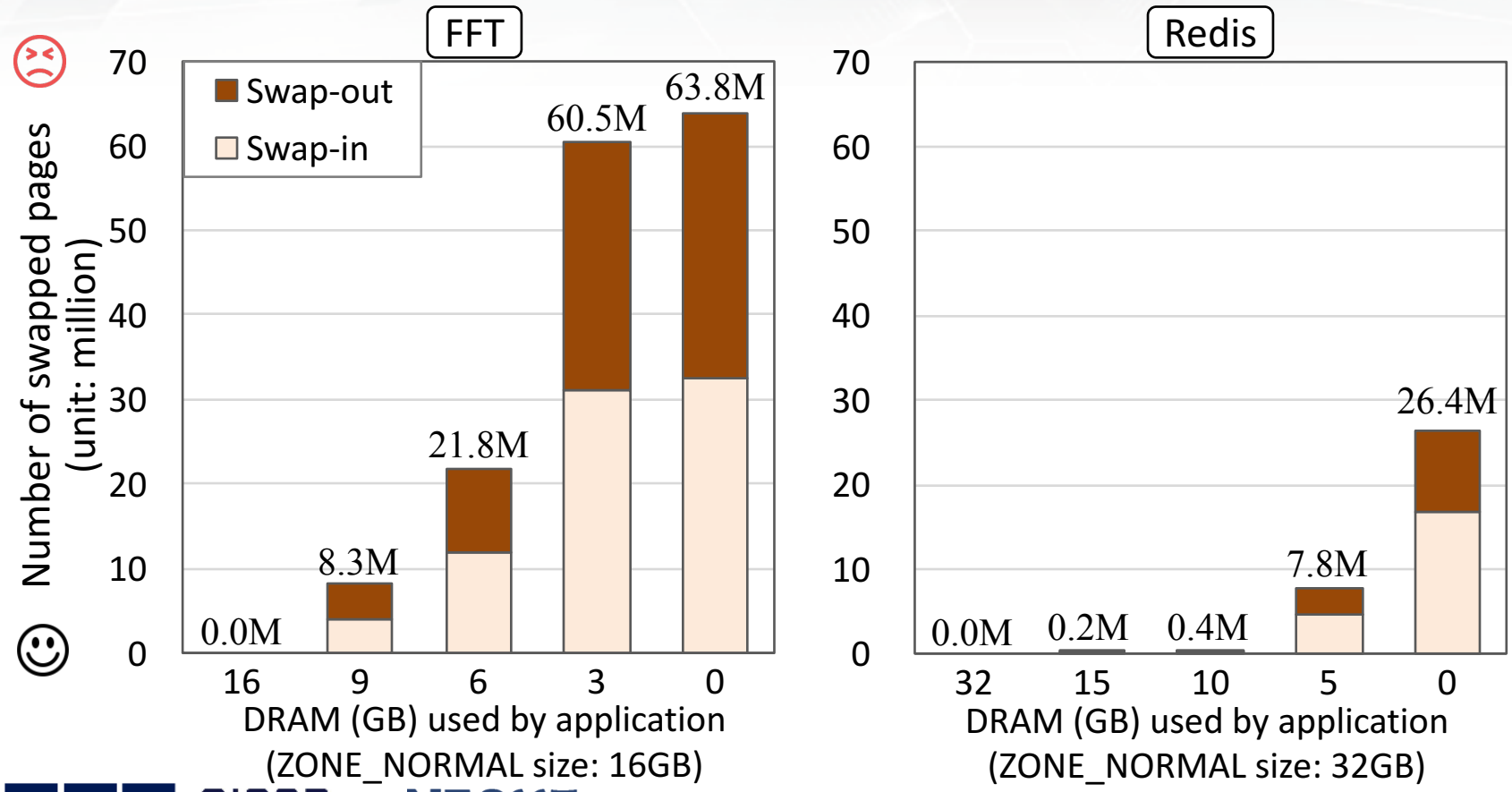




- **PMB "unaffected" by DRAM size**
  - swap page copies increase with reduced DRAM size

Evaluation

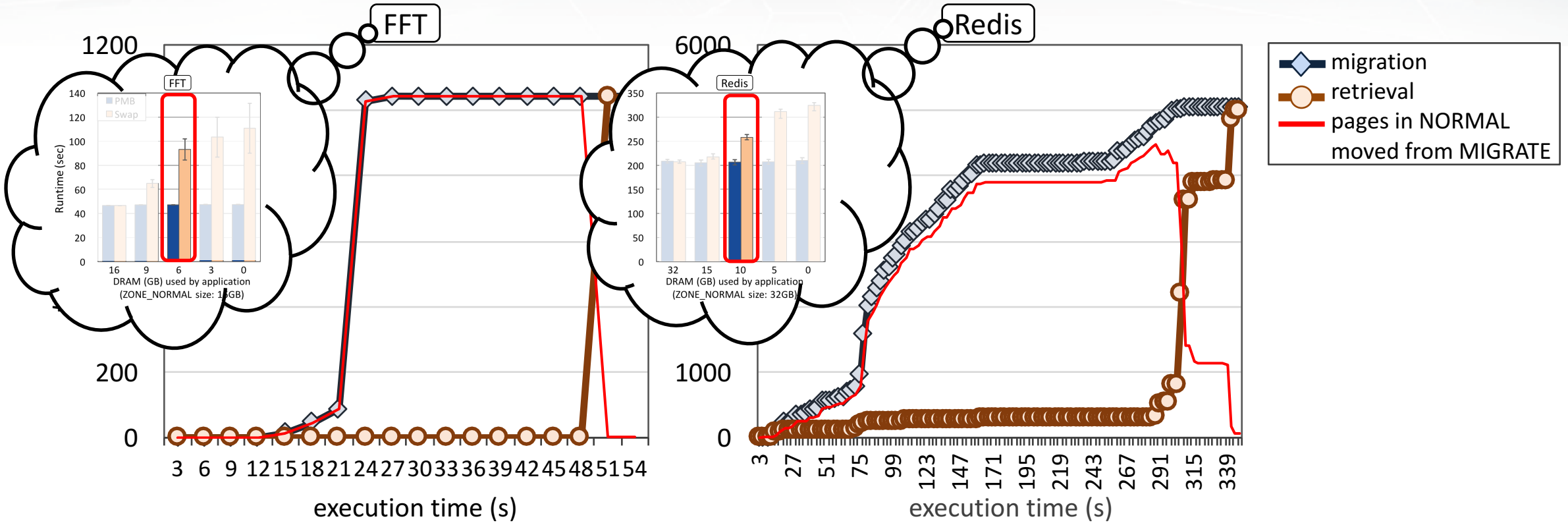
- **Number of copied pages**



- Reported by */proc/vmstat*

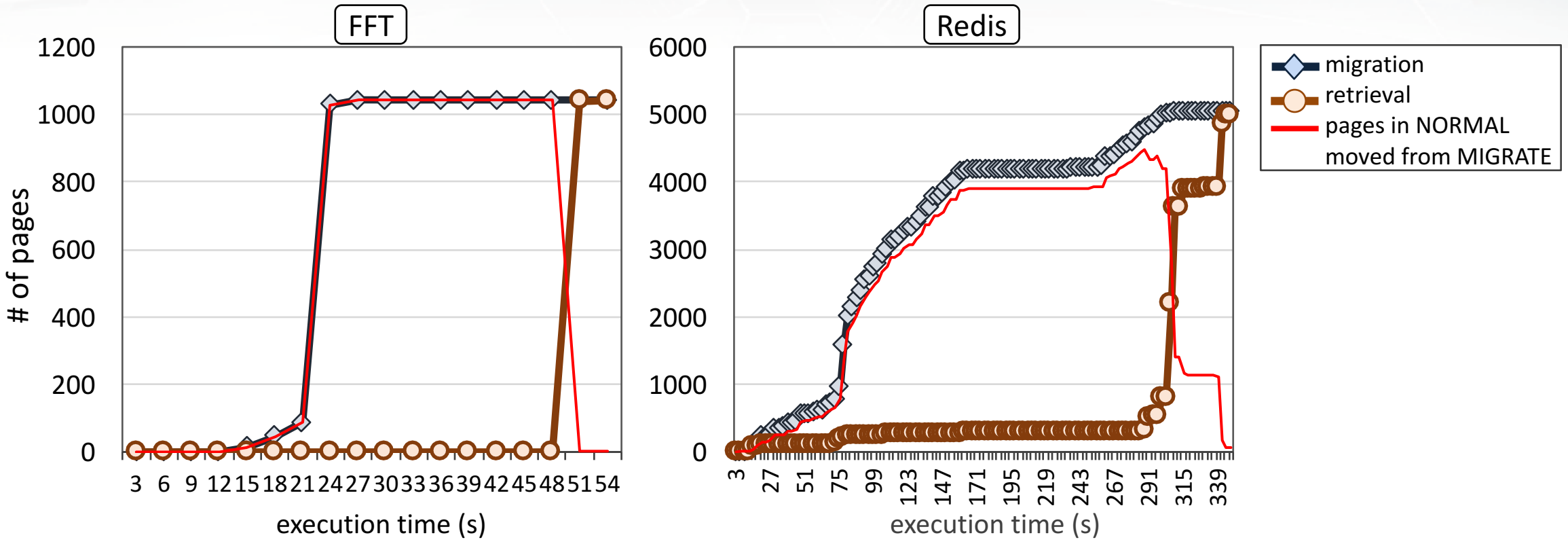
Evaluation

- **PM page migration and retrieval behavior**
  - Measured per 3s interval



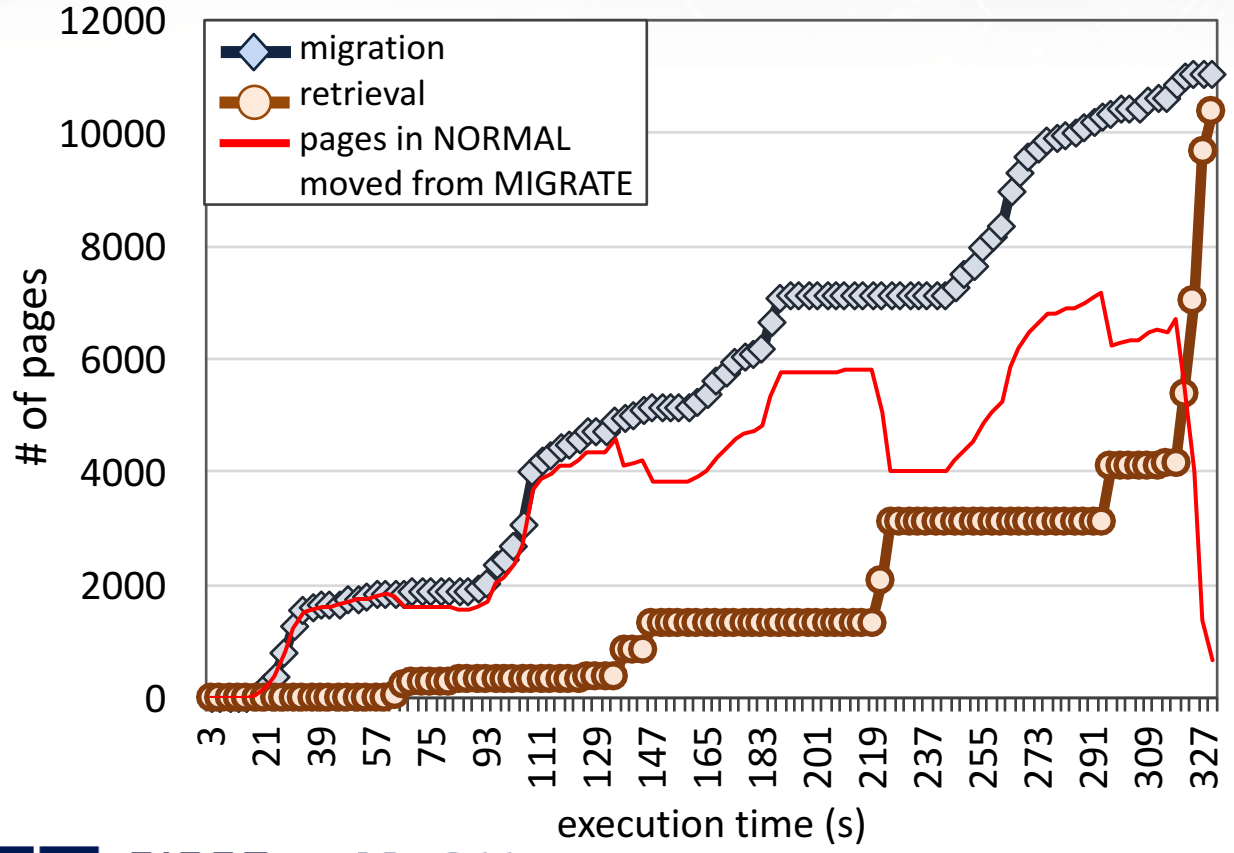
Evaluation

- **PM page migration and retrieval behavior**
  - Measured per 3s interval



Evaluation

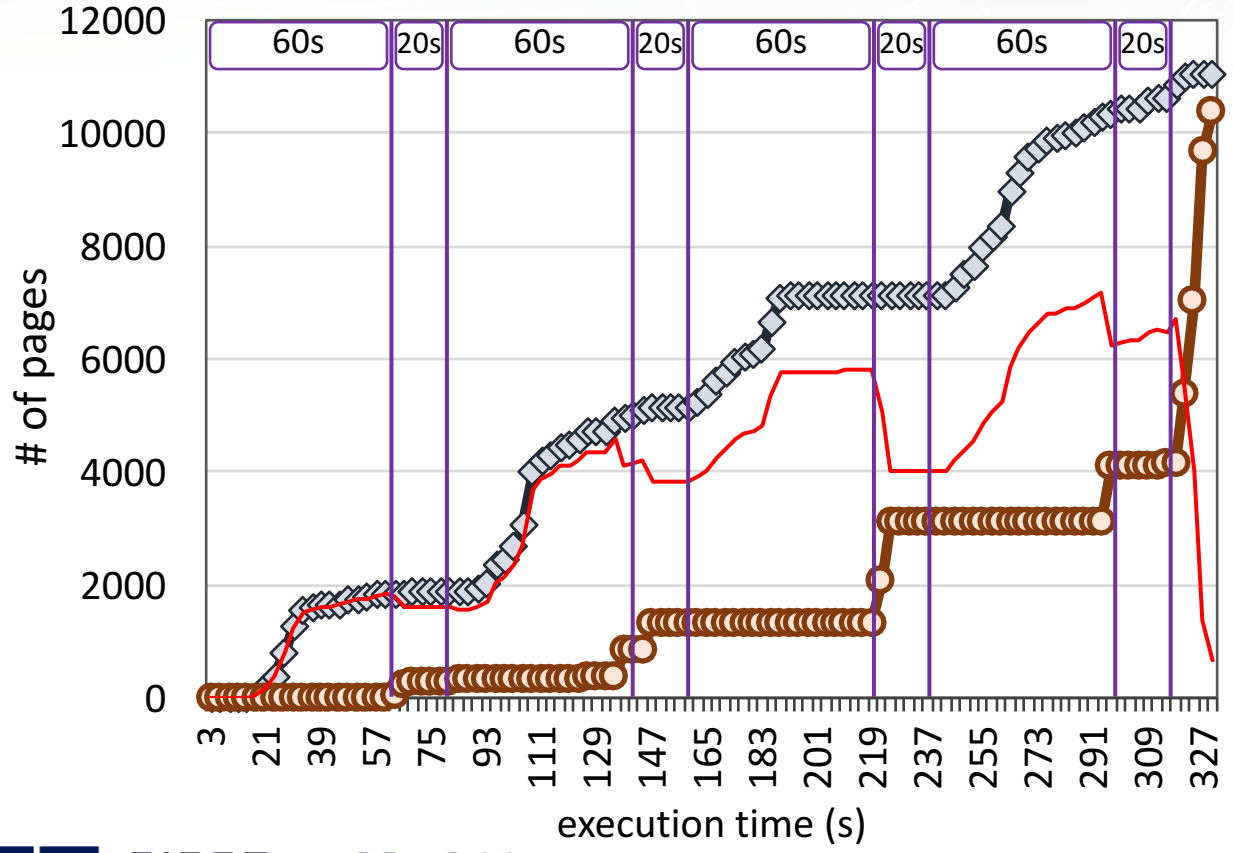
- **PM page migration and retrieval behavior**
  - FFT and Redis executed simultaneously



- Redis runs once
- FFT running time: 60 seconds
  - Restarted upon termination after a 20 second lapse

Evaluation

- **PM page migration and retrieval behavior**
  - FFT and Redis executed simultaneously



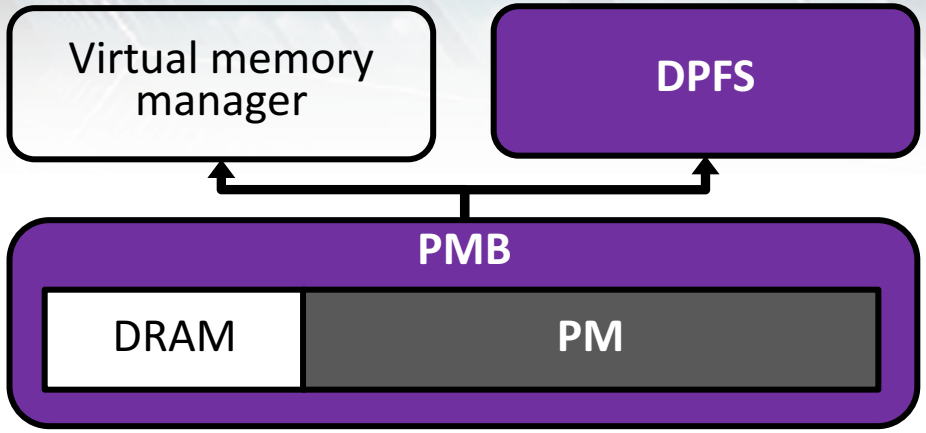
- Redis runs once
- FFT running time: 60 seconds
  - Restarted upon termination after a 20 second lapse
  - Pages are being freed and retrieved in the vicinity of 60, 140, 220, 300 seconds

Summary

- **Proposed a system called Storage-As-You-Go (SAY-Go)**
  - Transparently adjusts the use of PM
  - PM can be used as memory as well as storage as need be
  - Break the boundary between memory and storage
- **Persistent Memory Buddy (PMB)**
  - Memory allocation service that can freely grow and shrink working memory
  - Experimental results show PMB performs better than traditional swap

Future work

- **DPFS**
  - Requires dynamic partition resizing support
  - In the design and early implementation phase
- **PMB**
  - More detailed policies for managing page
  - Various optimizations



Q & A