MFence : Defending Against Memory Access Interference in a Disaggregated Cloud Memory Platform

Jinhoon Lee, Yeonwoo Jung, Suyeon Lee, Safdar Jamil, Sungyong Park,
Kwangwon Koh, Hongyeon Kim, Kangho Kim, Youngjae Kim

Presenter: Yeonwoo Jeong

DISCOS
Data-Intensive Computing & Systems Laboratory

ETRI
Electronics and Telecommunications Research Institute

Department of Computer Science and Engineering
Sogang University, Seoul
South Korea

Agenda

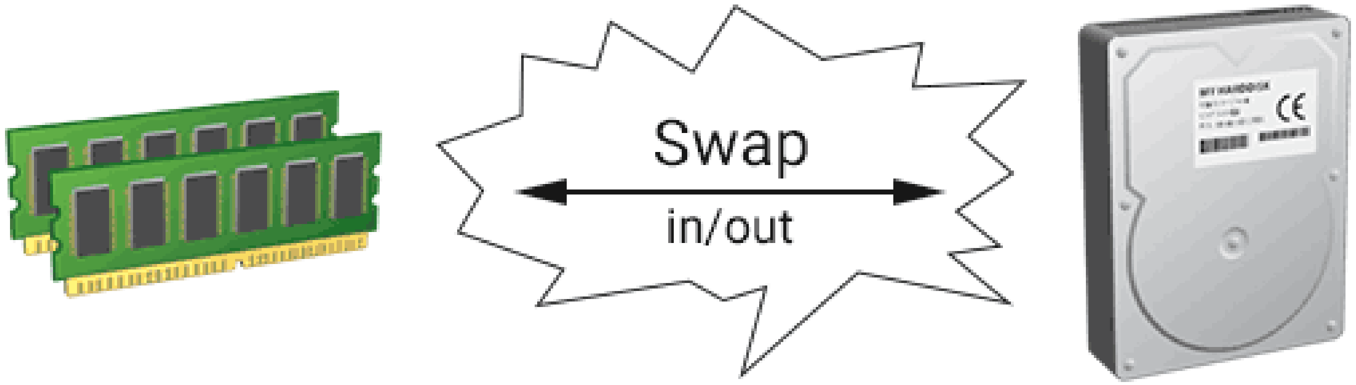
- Introduction & Background
- Motivation
- Design
- Evaluation

Memory-Intensive Applications

- Demand for processing memory-intensive applications is high
  - ✓ Machine learning, graph processing, KV Store
- Bigdata applications require a high memory and cause memory shortages

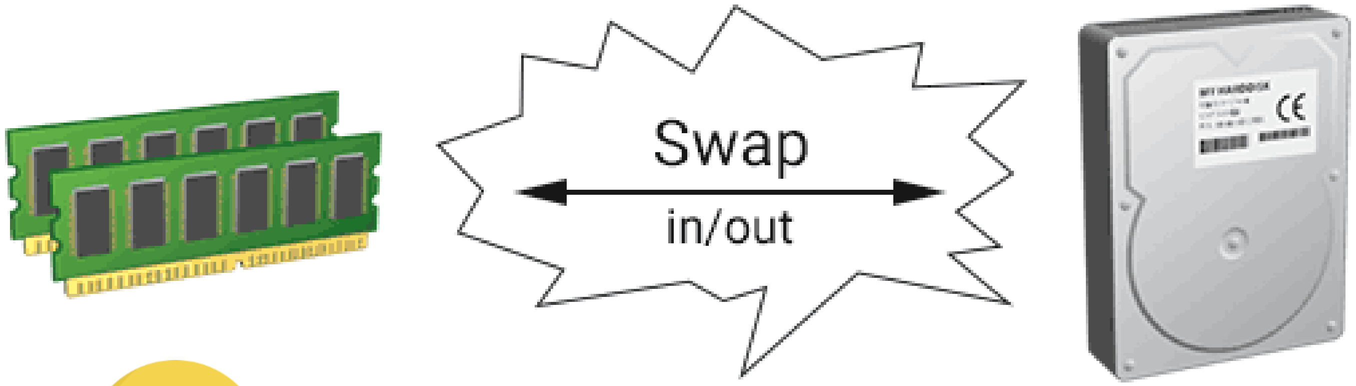
Memory-Intensive Applications

- Limited memory capacity per machine
  - ✓ OOM (Out of memory), application failures..
- Traditional solution for big memory
  - ✓ Disk swap



Memory-Intensive Applications

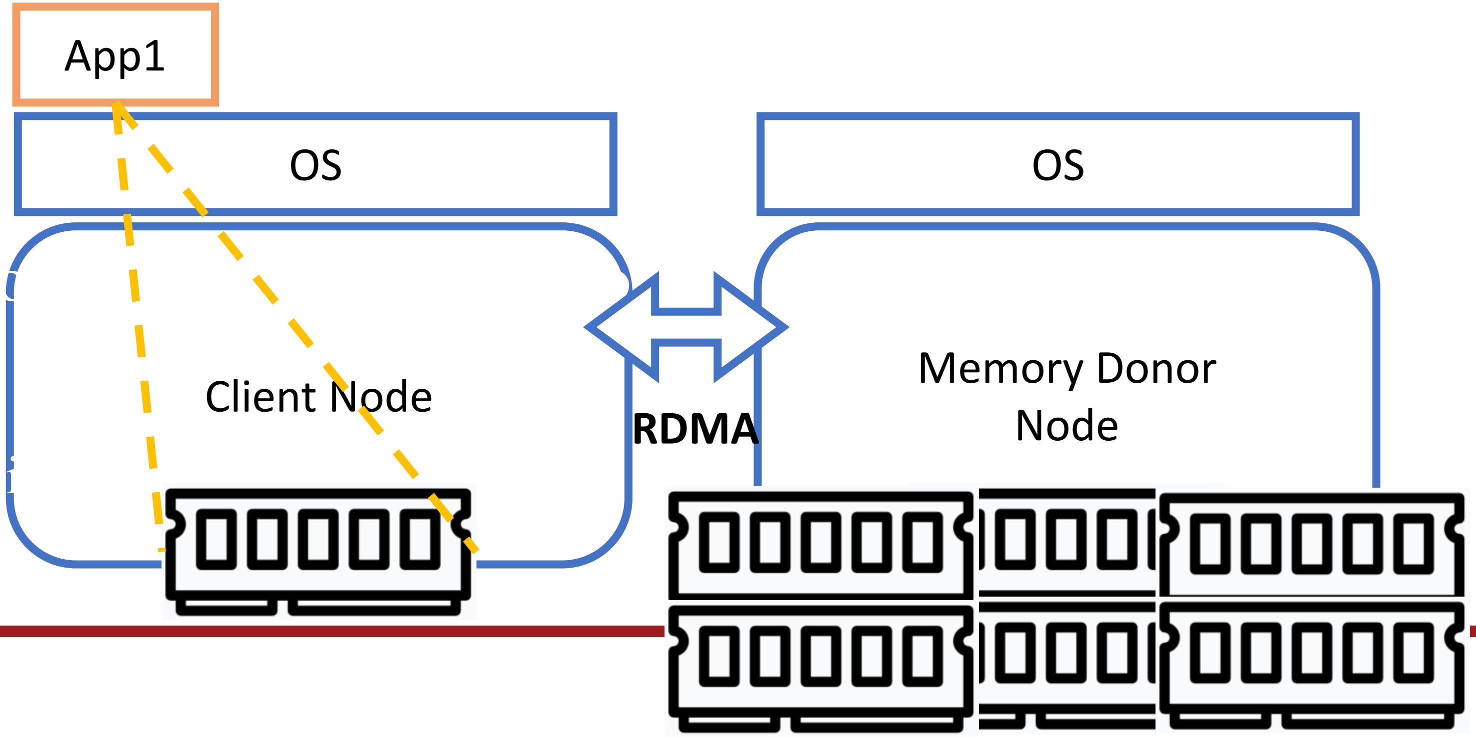
- Limited memory capacity per machine
  - ✓ OOM (Out of memory), application failures..
- Traditional solution for big memory
  - ✓ Disk swap



Higher access latency in disk swap!

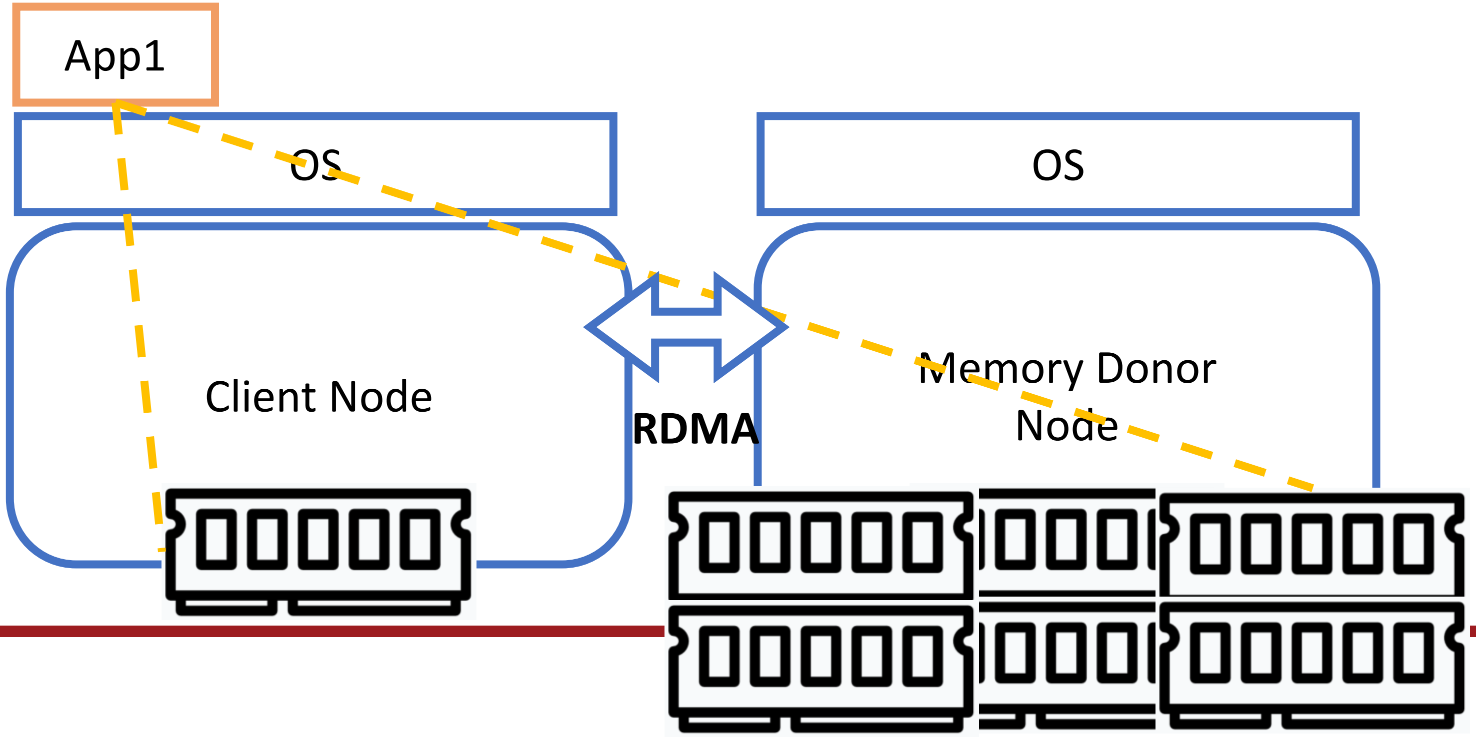
Disaggregated Memory Platform

- Memory of a remote server as an extension of limited local memory
  - ✓ One machine **borrows** memory from remote machine with high-speed network
- VM-based remote memory solution
  - ✓ **Client machine** (borrows the memory)
  - ✓ **Donor machine** (provides the memory)



Disaggregated Memory Platform

- Memory of a remote server as an extension of limited local memory
  - ✓ One machine **borrows** memory from remote machine with high-speed network
- VM-based remote memory solution
  - ✓ **Client machine** (borrows the memory)
  - ✓ **Donor machine** (provides the memory)



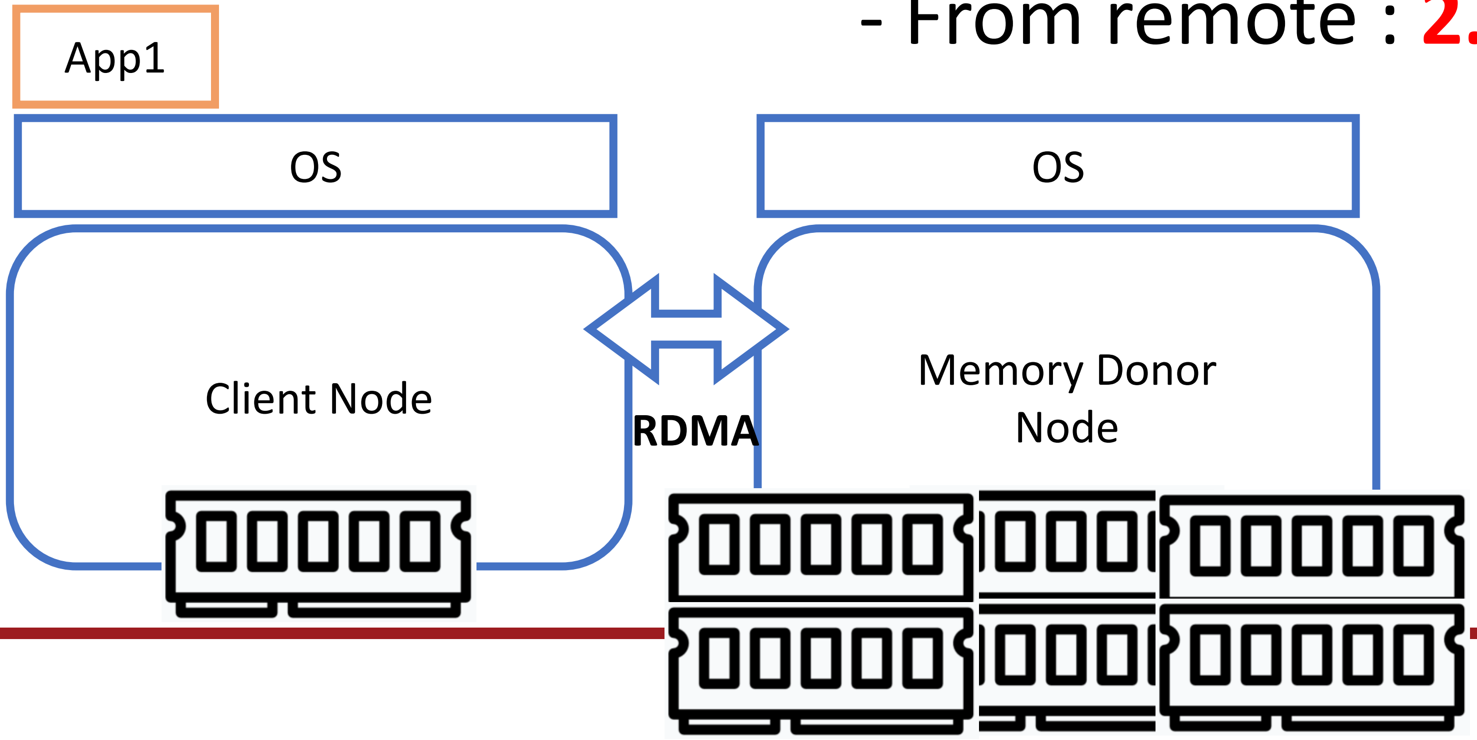
Disaggregated Memory Platform

- Memory of a remote server as an extension of limited local memory
  - ✓ One machine **borrows** memory from remote machine with high-speed network
- VM-based remote memory solution
  - ✓ **Client machine** (borrows the memory)
  - ✓ **Donor machine** (provides the memory)

Memory Access Time

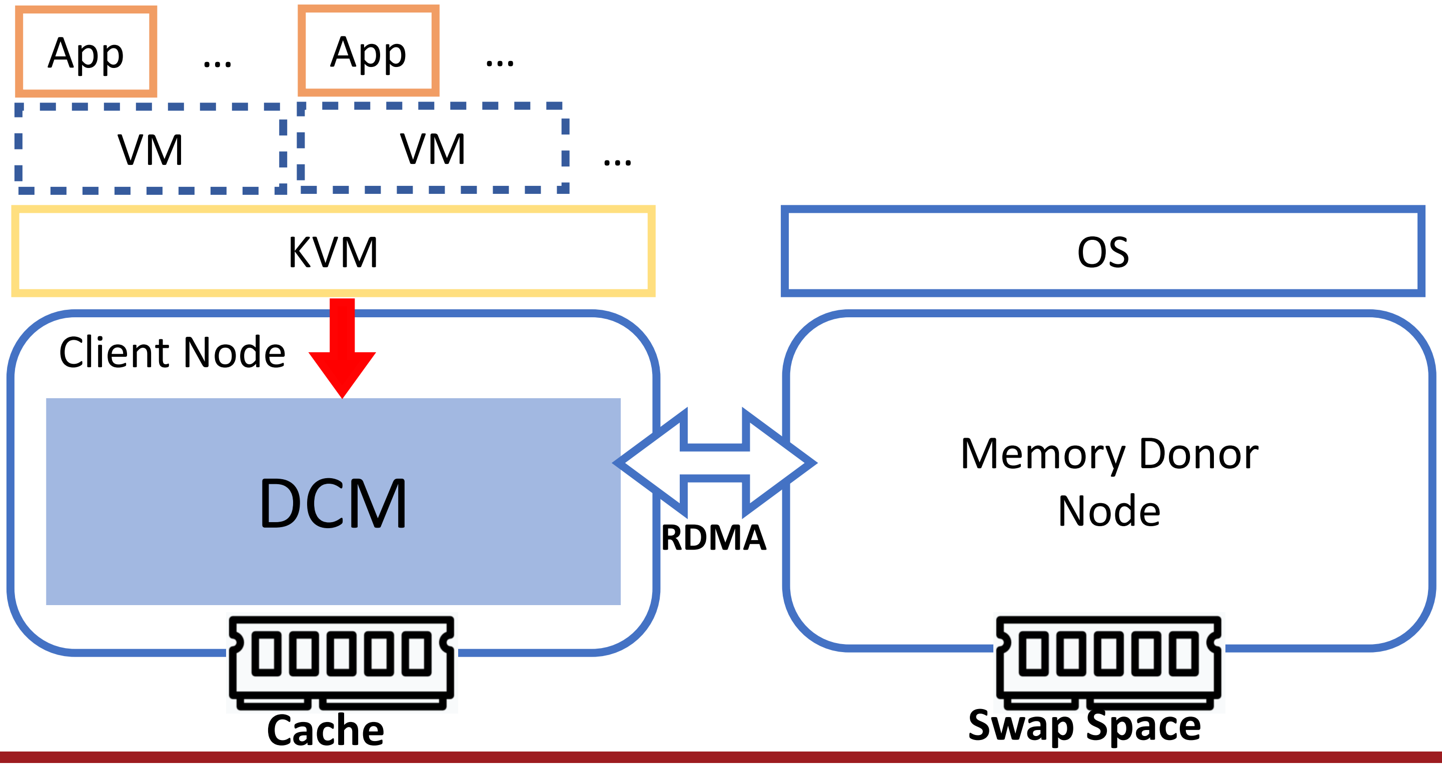
- From local : 10~256 ns

- From remote : **2.8 us**



Disaggregated Memory Platform

- DCM[TC'19]* uses **local memory** as an **inclusive cache** and maximizes the **hit rate** to reduce fetching from remote memory
- Using a VM, donor's memory can be perceived as its own memory space



* Koh, K. Disaggregated cloud memory with elastic block management. *IEEE Transactions on Computers 68*, 1 (2019)

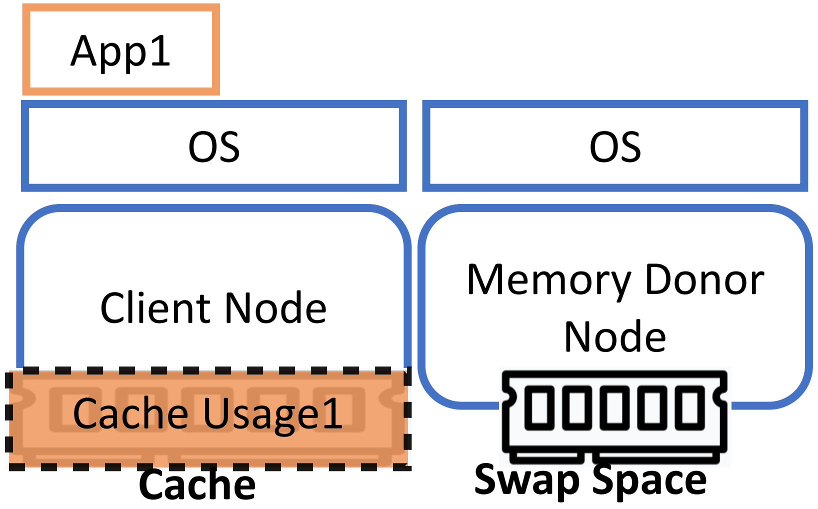
SOGANG UNIVERSITY

Agenda

- Introduction & Background
- **Motivation**
- Design
- Evaluation

Cache Contention on DCM

- Processes in VM use the shared cache concurrently, a memory race condition occurs

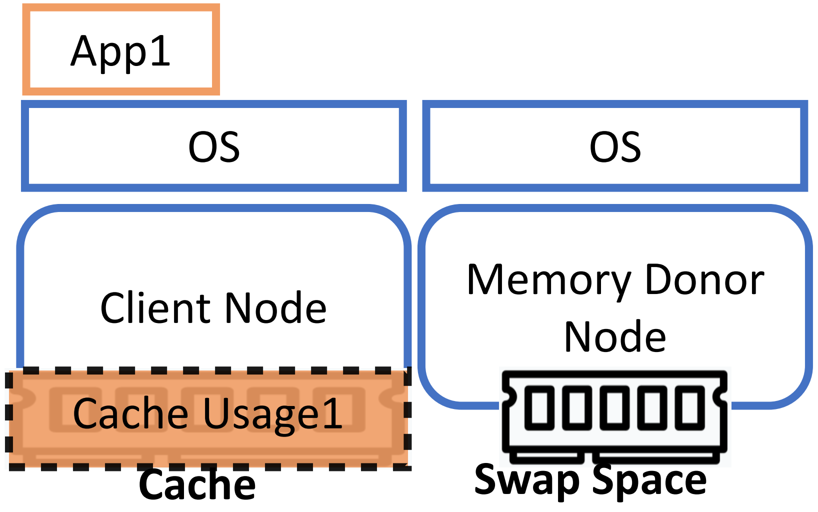


Single run

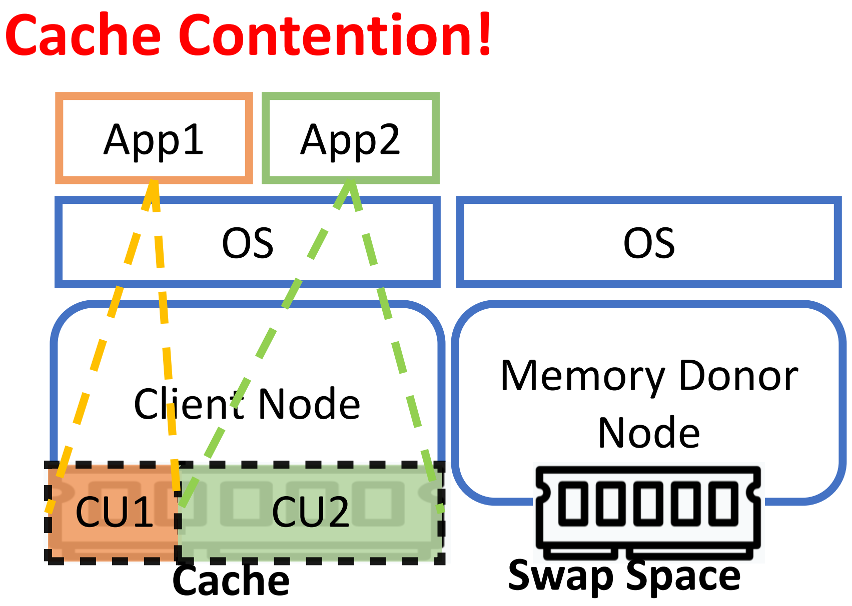
SOGANG UNIVERSITY

Cache Contention on DCM

- Processes in VM use the shared cache concurrently, a memory race condition occurs



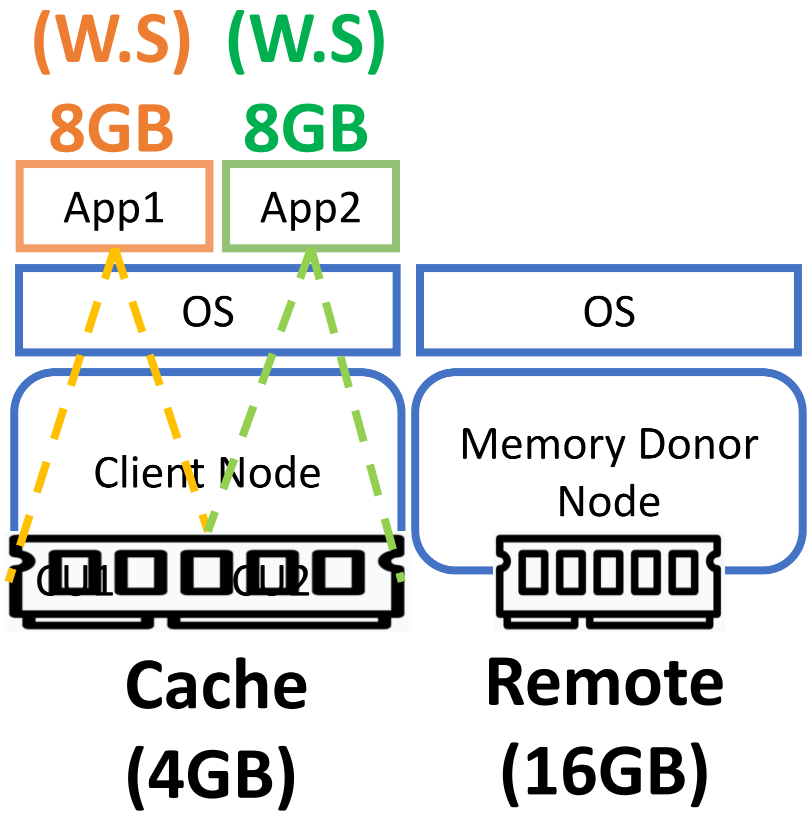
Single run



Shared run

Cache Contention on DCM

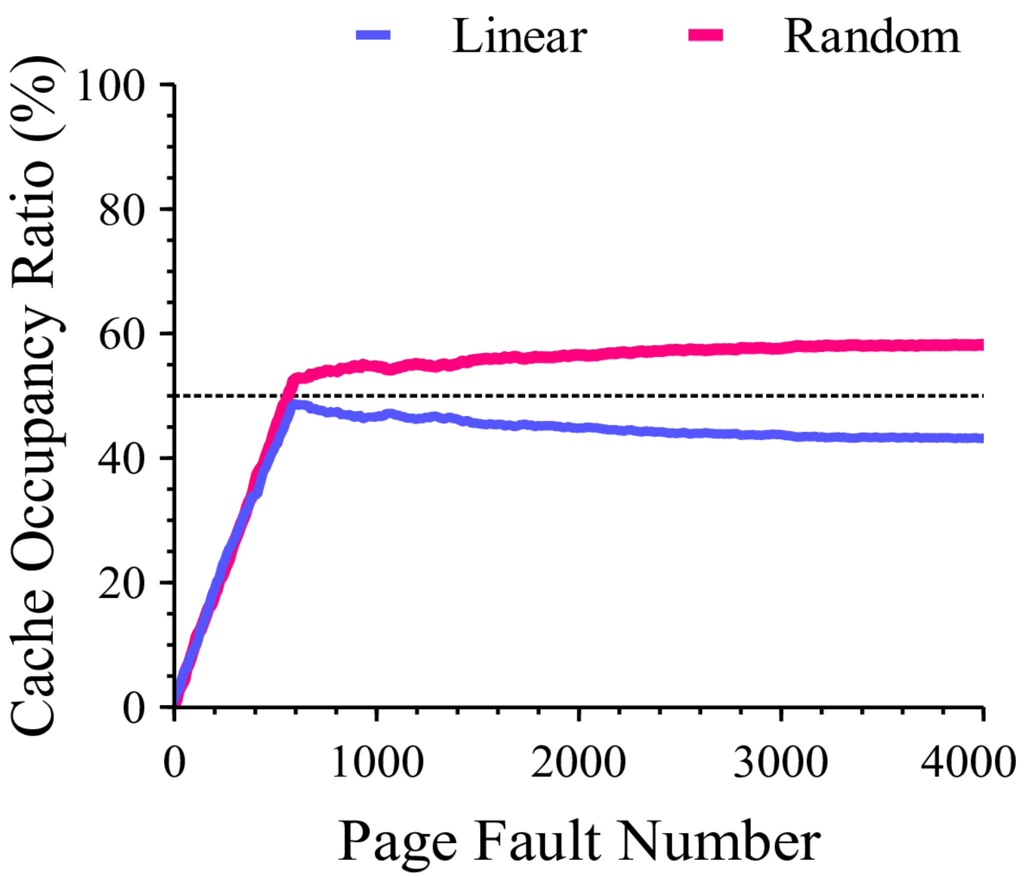
- Evaluation with micro memory benchmark(PmBench*) on DCM
  - ✓ Workload : 1 linear/random memory-access
  - ✓ All process memory footprint : 8GB
  - ✓ Local memory capacity in VM : 4GB



*Jisoo, Y. Pmbench: A micro-benchmark for profiling paging performance on a system with low-latency ssds. In Proceedings of the Information Technology New Generations (2018)

Cache Contention on DCM

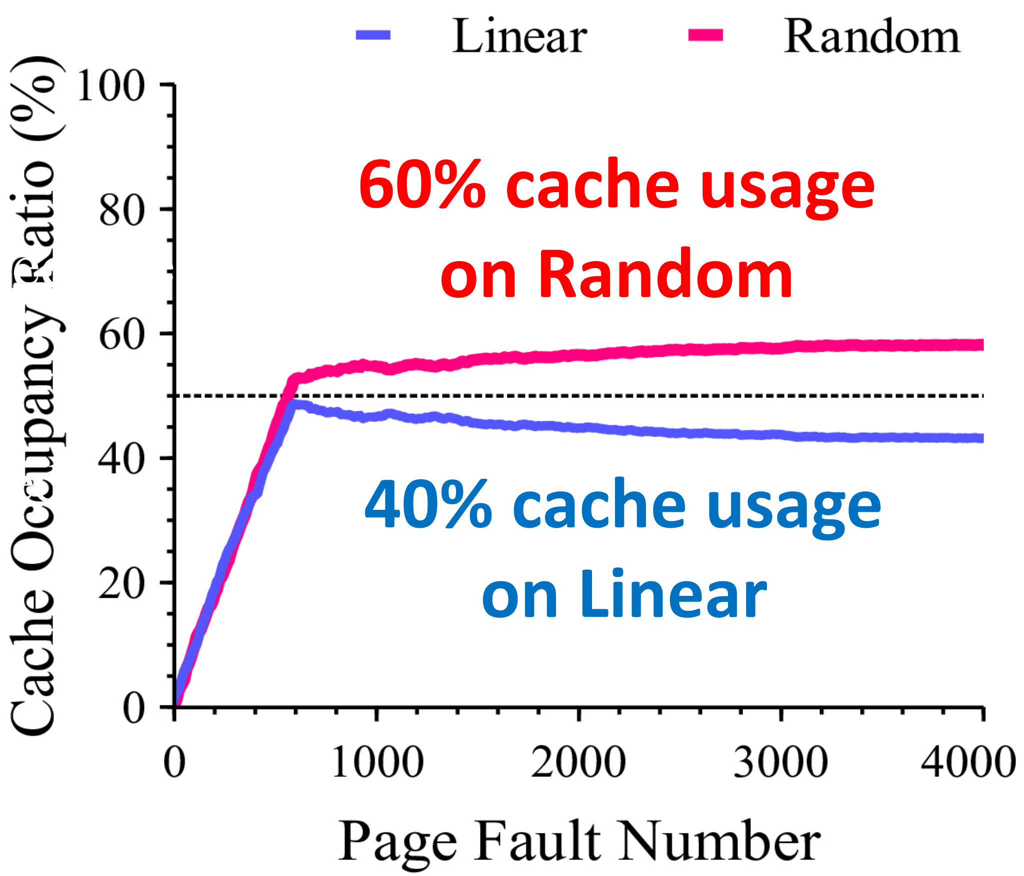
- Evaluation with micro memory benchmark(PmBench*) on DCM



*Jisoo, Y. Pmbench: A micro-benchmark for profiling paging performance on a system with low-latency ssds. In Proceedings of the Information Technology New Generations (2018)

Cache Contention on DCM

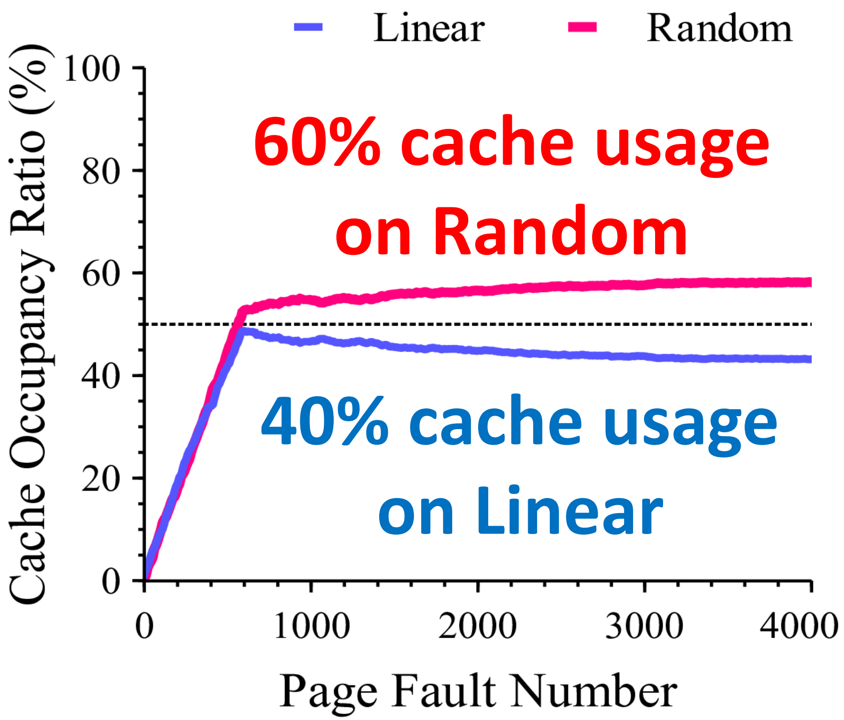
- Evaluation with micro memory benchmark(PmBench*) on DCM



*Jisoo, Y. Pmbench: A micro-benchmark for profiling paging performance on a system with low-latency ssds. In Proceedings of the Information Technology New Generations (2018)

Cache Contention on DCM

- Evaluation with micro memory benchmark(PmBench*) on DCM

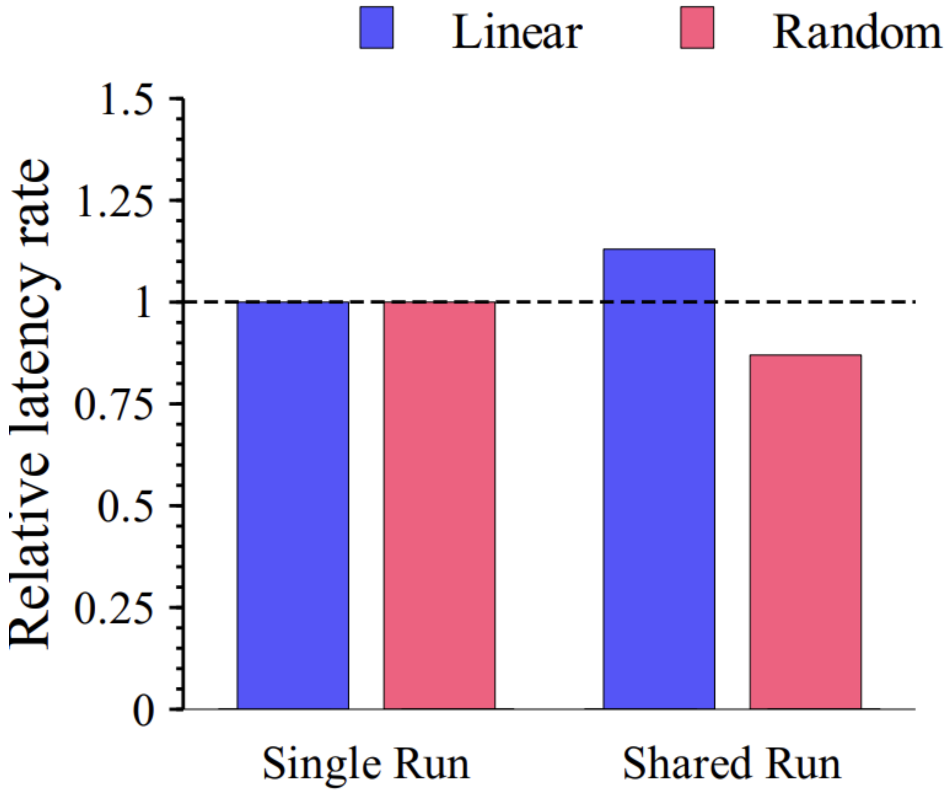


Unfair cache occupancy per process in DCM

*Jisoo, Y. Pmbench: A micro-benchmark for profiling paging performance on a system with low-latency ssds. In Proceedings of the Information Technology New Generations (2018)

Cache Contention on DCM

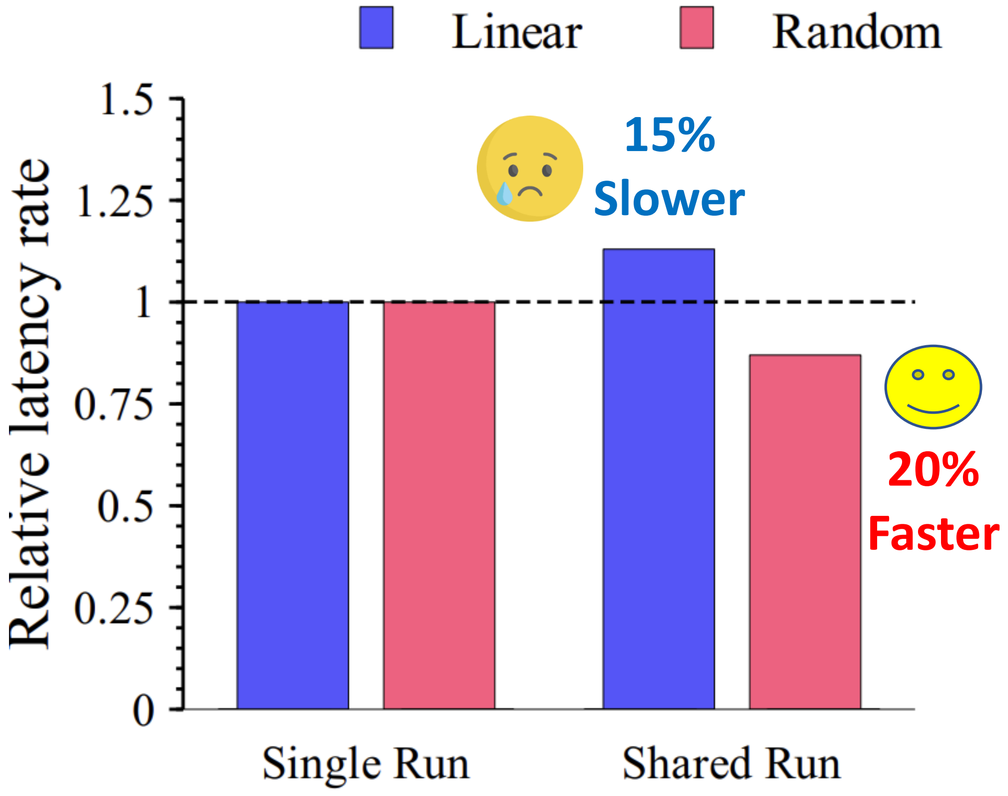
- Evaluation with micro memory benchmark(PmBench*) on DCM



*Jisoo, Y. Pmbench: A micro-benchmark for profiling paging performance on a system with low-latency ssds. In Proceedings of the Information Technology New Generations (2018)

Cache Contention on DCM

- Evaluation with micro memory benchmark(PmBench*) on DCM

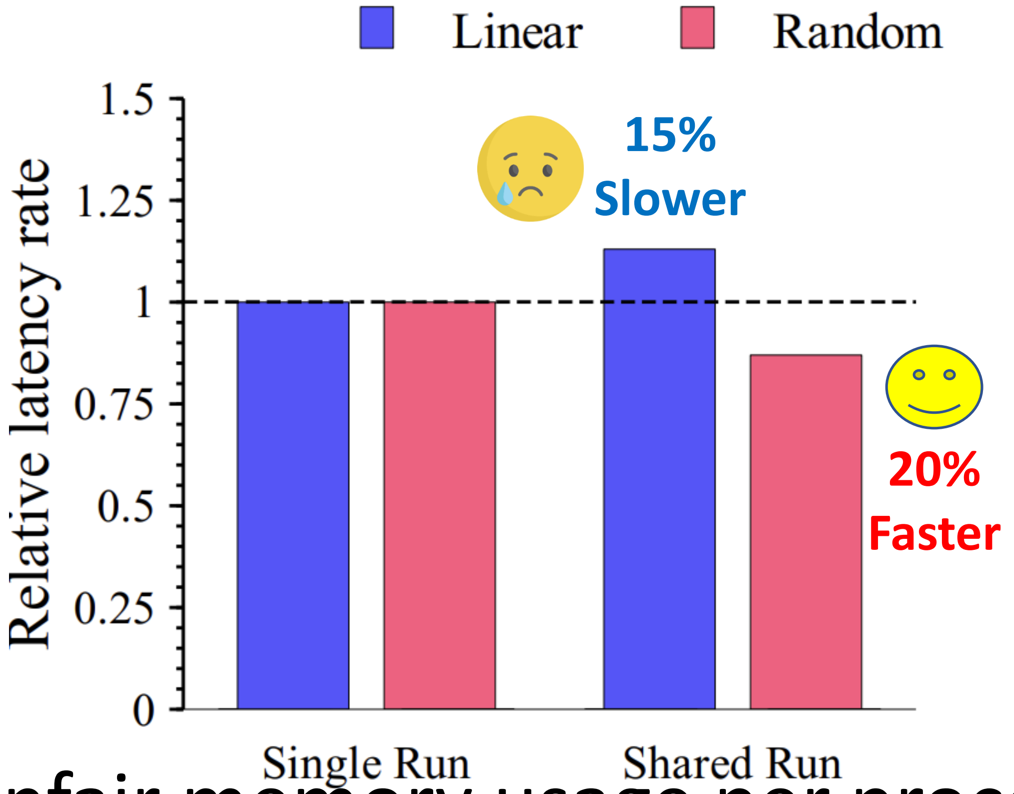


*Jisoo, Y. Pmbench: A micro-benchmark for profiling paging performance on a system with low-latency ssds. In Proceedings of the Information Technology New Generations (2018)

SOGANG UNIVERSITY

Cache Contention on DCM

- Evaluation with micro memory benchmark(PmBench*) on DCM



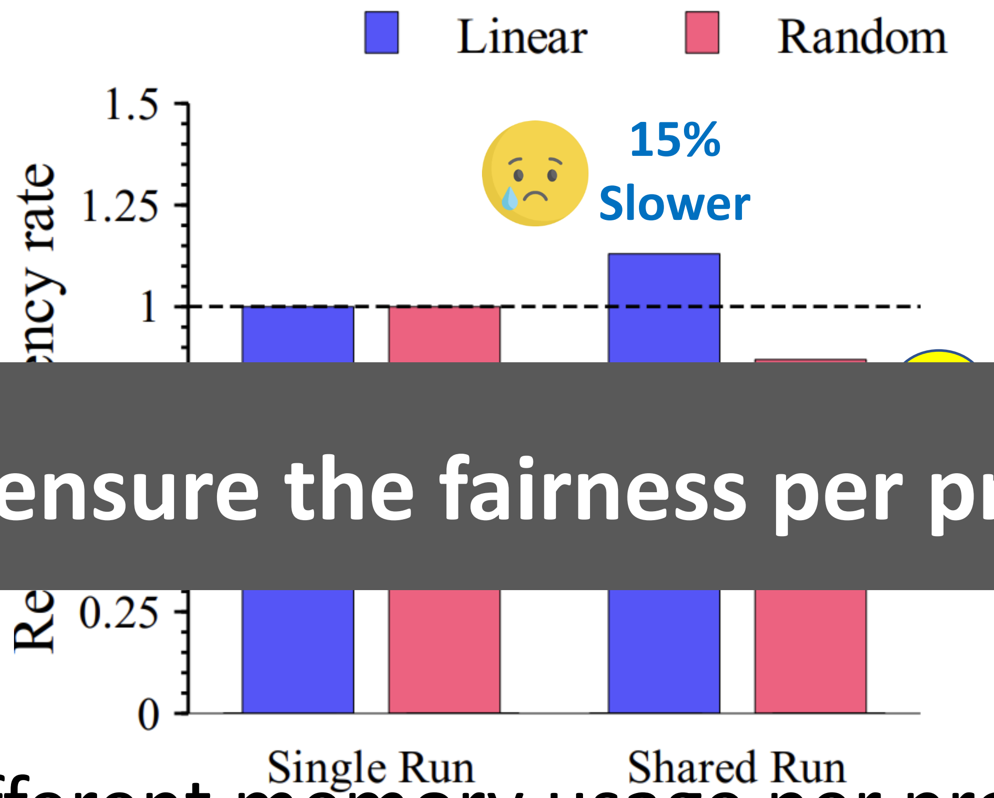
Unfair memory usage per process

sharing local memory affects the performance

*Jisoo, Y. Pmbench: A micro-benchmark for profiling paging performance on a system with low-latency ssds. In Proceedings of the Information Technology New Generations (2018)

Cache Contention on DCM

- Evaluation with micro memory benchmark(PmBench*) on DCM



Can we ensure the fairness per process?

Different memory usage per process

sharing local memory affects the performance

*Jisoo, Y. Pmbench: A micro-benchmark for profiling paging performance on a system with low-latency ssds. In Proceedings of the Information Technology New Generations (2018)

Opportunity with Cache Partitioning

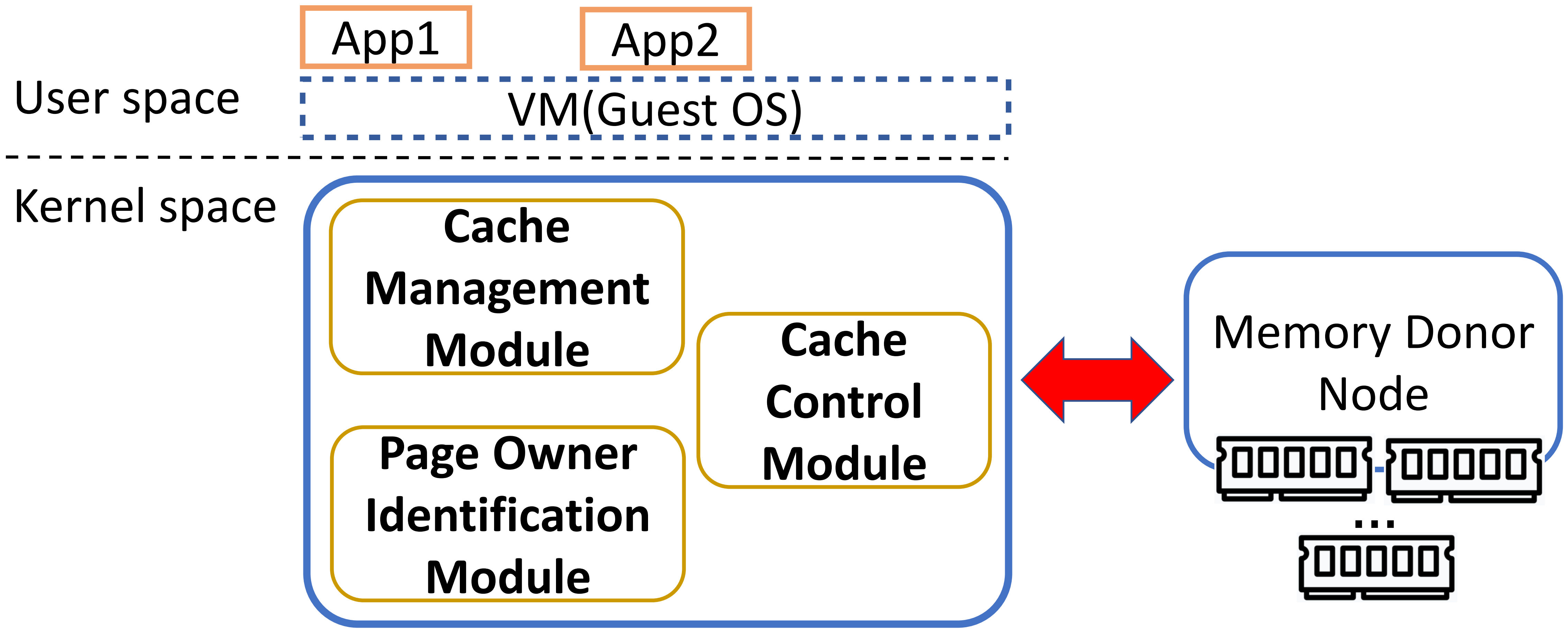
- **Cache partitioning** can overcome the unfair utilization of local cache between memory-greedy processes

- **Challenges**
  - ✓ Host kernel cannot directly know page information of the process running on guest OS/VM
  - ✓ Overhead of page identification per process between host/guest area is not trivial

Agenda

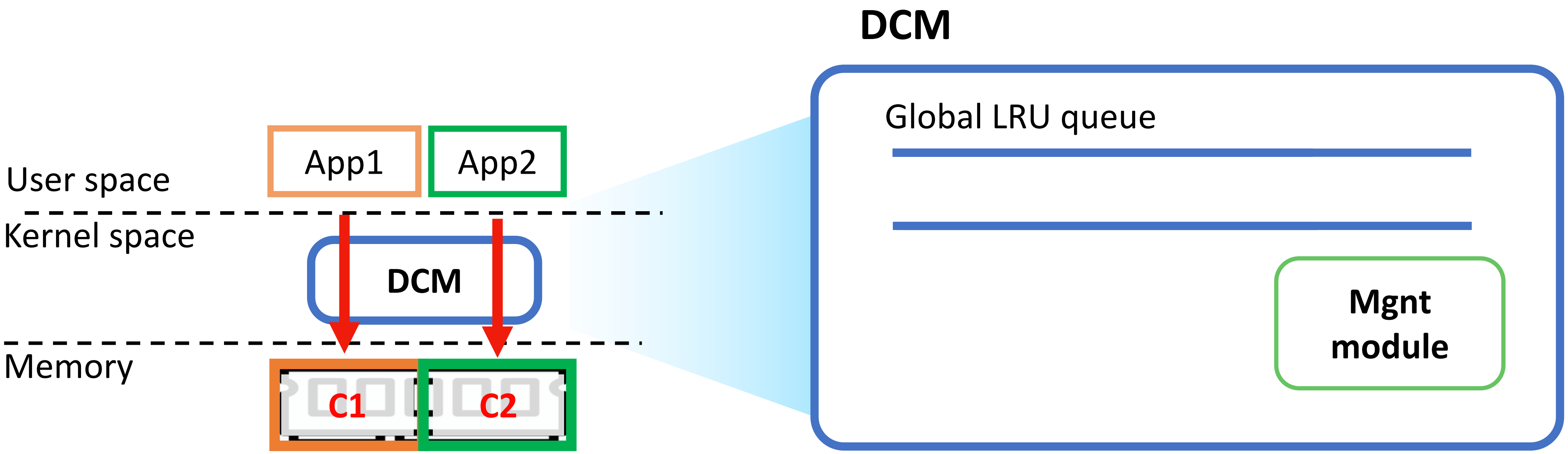
- Introduction & Background
- Motivation
- **Design**
- Evaluation

MFence


App1
App2
User space
VM(Guest OS)
Kernel space
Cache Management Module
Page Owner Identification Module
Cache Control Module
Memory Donor Node
...

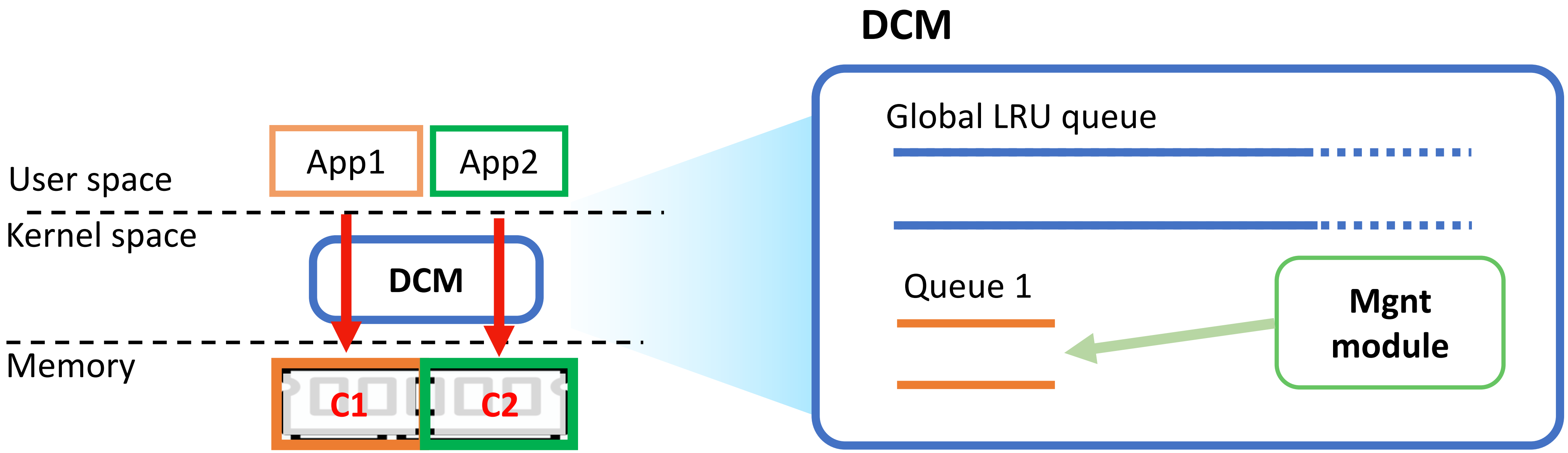
MFence

- **Cache Management Module**
  - ✓ MFence manages a per-process LRU queue, providing local memory partitions of varying sizes
  - ✓ The split queue (cache) has a unique ID[group ID (**GID**)] is assigned



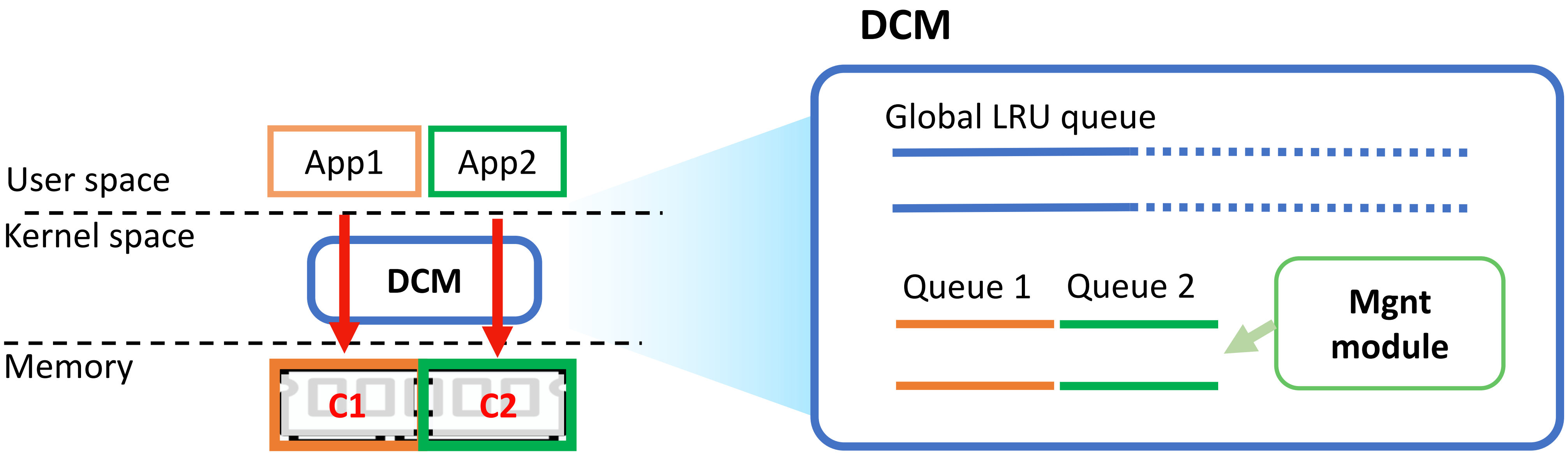
MFence

- **<u>Cache Management Module</u>**
  - ✓ MFence manages a per-process LRU queue, providing local memory partitions of varying sizes
  - ✓ The split queue (cache) has a unique ID[group ID (**GID**)] is assigned



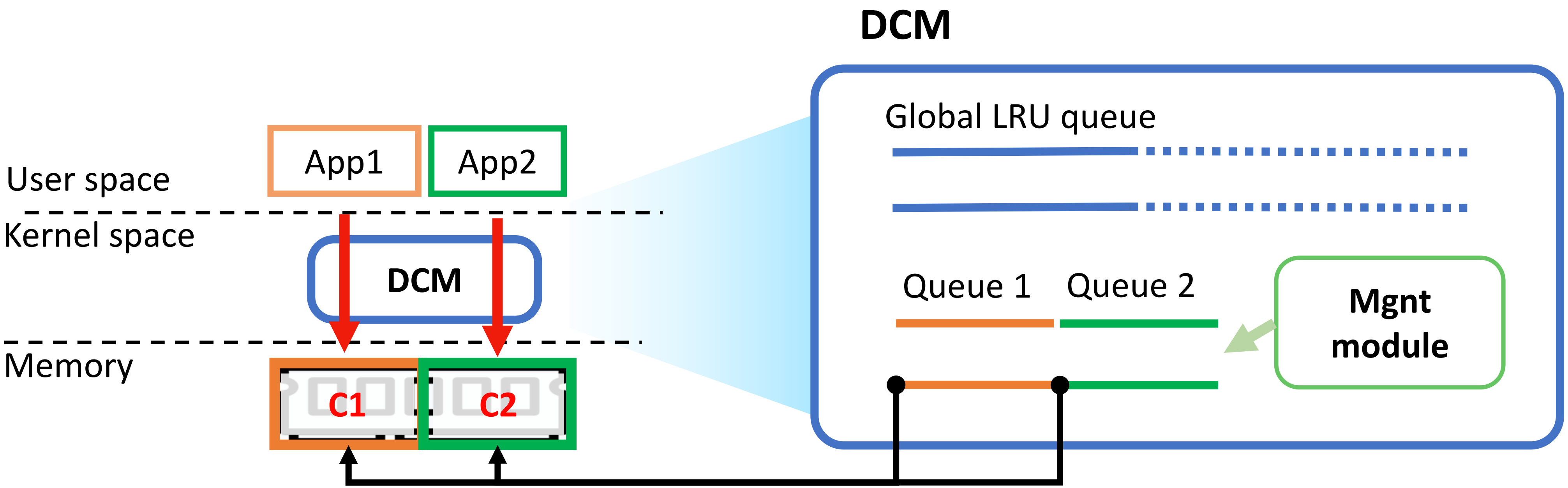
MFence

- **<u>Cache Management Module</u>**
  - ✓ MFence manages a per-process LRU queue, providing local memory partitions of varying sizes
  - ✓ The split queue (cache) has a unique ID[group ID (**GID**)] is assigned



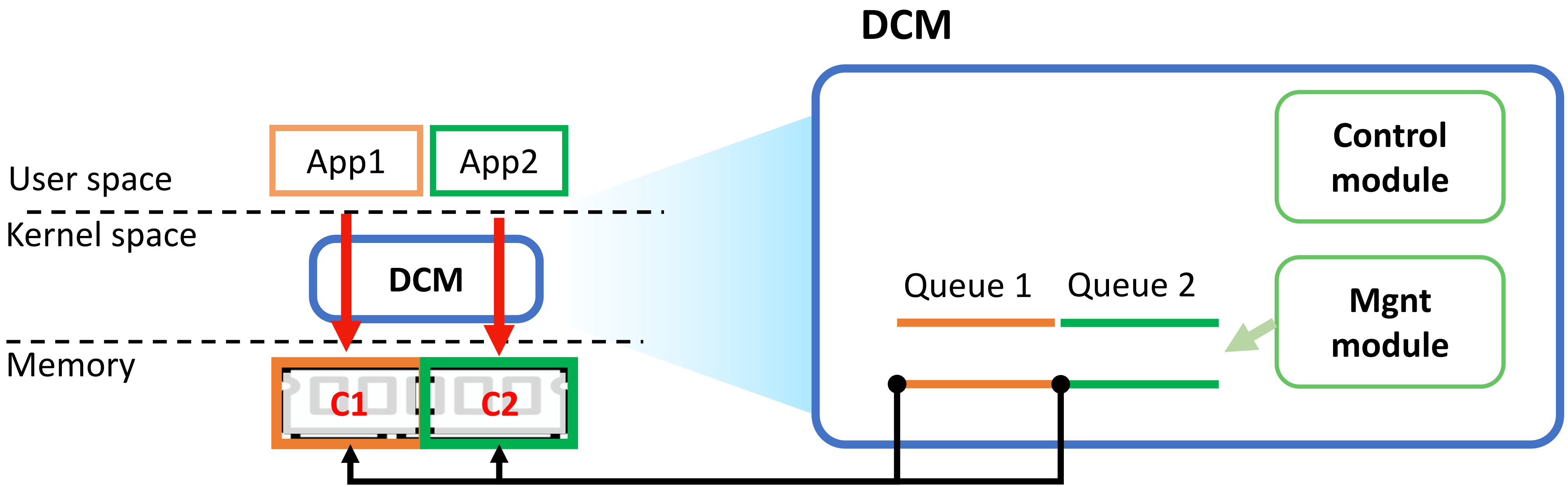
MFence

- **<u>Cache Management Module</u>**
  - ✓ MFence manages a per-process LRU queue, providing local memory partitions of varying sizes
  - ✓ The split queue (cache) has a unique ID[group ID (**GID**)] is assigned



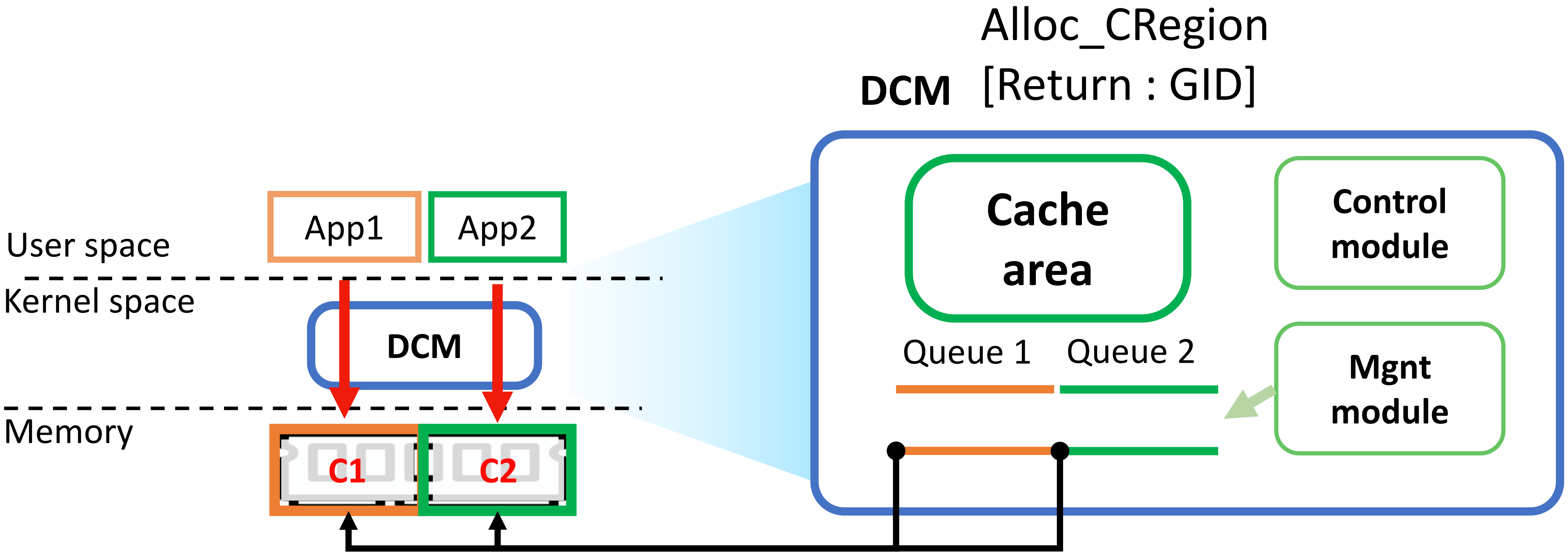
MFence

- **<u>Cache Control Module</u>**
  - ✓ MFence allows the user to manage the cache of the process
  - ✓ Cache group allocation, cache area creation, cache area release via user API



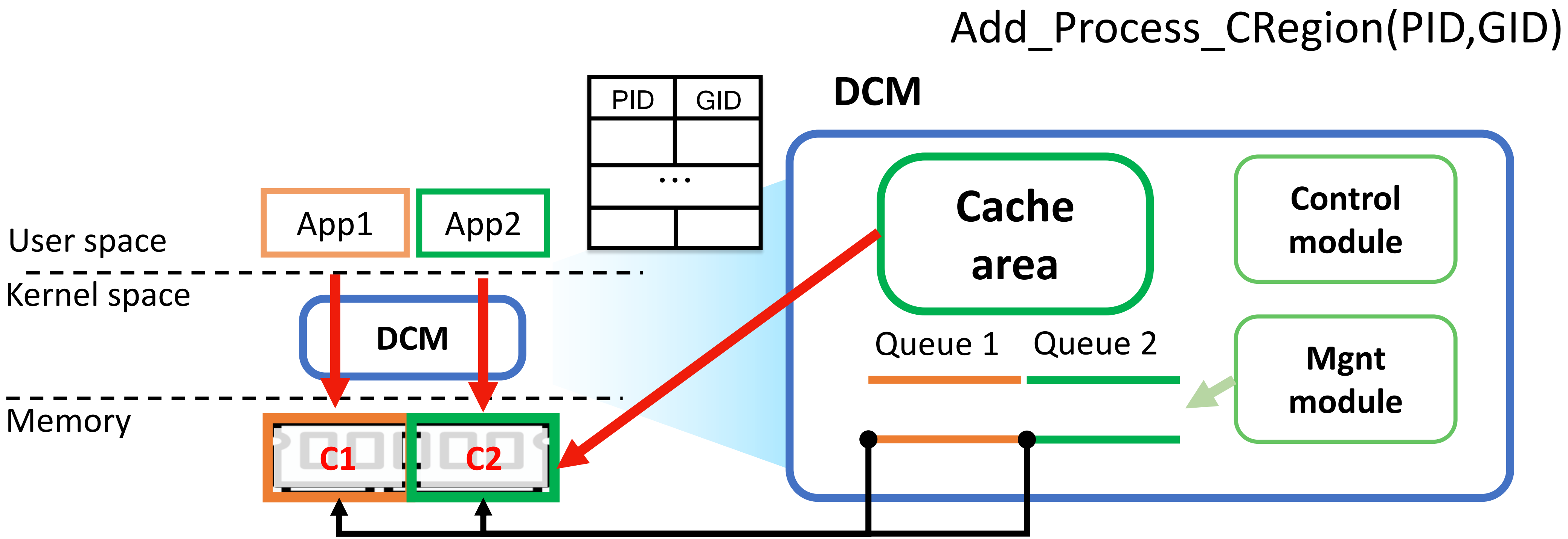
MFence

- **Cache Control Module**
  - ✓ MFence allows the user to manage the cache of the process
  - ✓ Cache group allocation, cache area creation, cache area release via user API



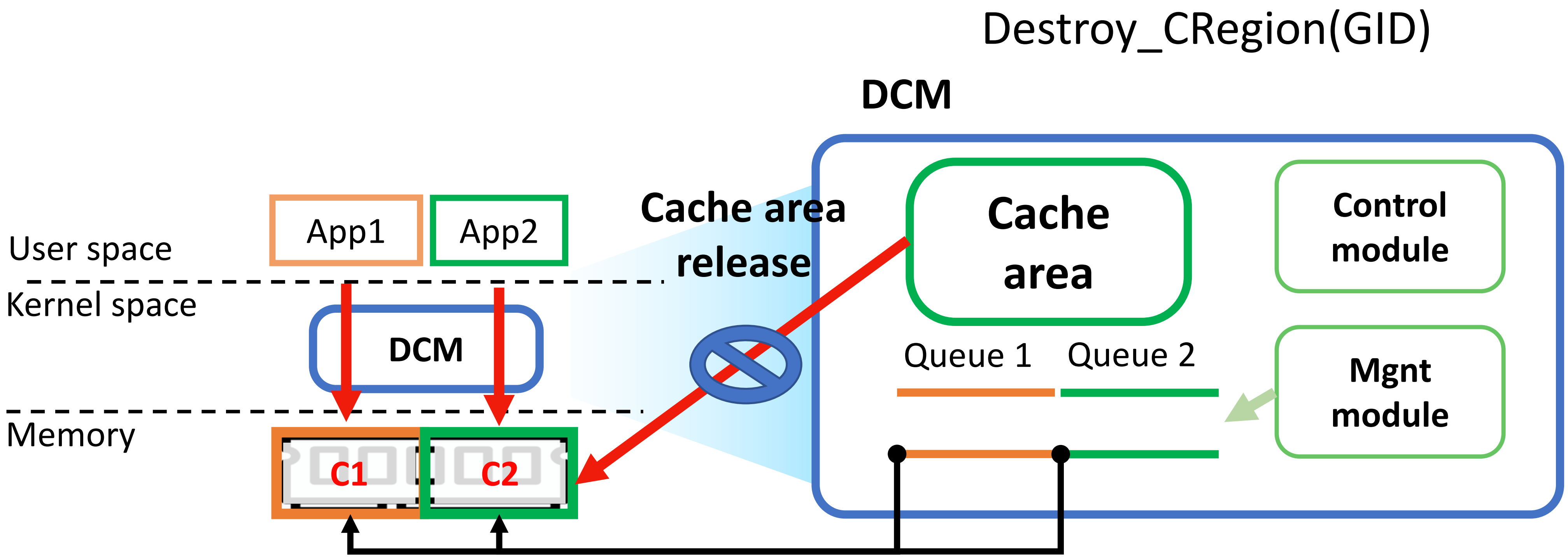
MFence

- **Cache Control Module**
  - ✓ MFence allows the user to manage the cache of the process
  - ✓ Cache group allocation, cache area creation, cache area release via user API



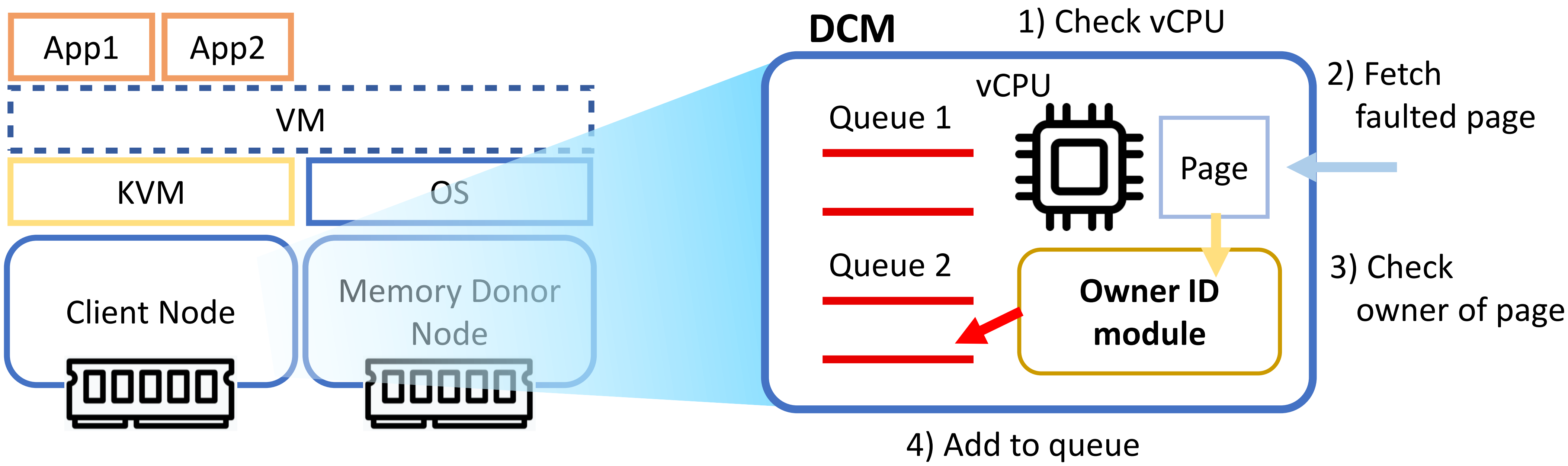
MFence

- **Cache Control Module**
  - ✓ MFence allows the user to manage the cache of the process
  - ✓ Cache group allocation, cache area creation, cache area release via user API



MFence

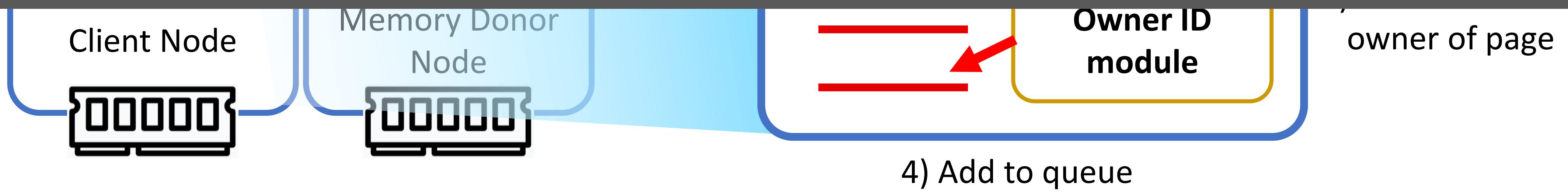
- **<u>Page Owner Identification Module</u>**
  - ✓ Each LRU queue caches only pages that belong to the process
  - ✓ Determine the queue into which to insert a page fetched from a remote server



MFence

- **Page Owner Identification Module**
  - ✓ Each LRU queue caches only pages that belong to the process
  - ✓ Determine the queue into which to insert a page fetched from a remote server

- ✓ **Kernel cannot be aware of upper layer's processes**
- ✓ **Kernel cannot confirm page's owner and perform memory separation between processes**

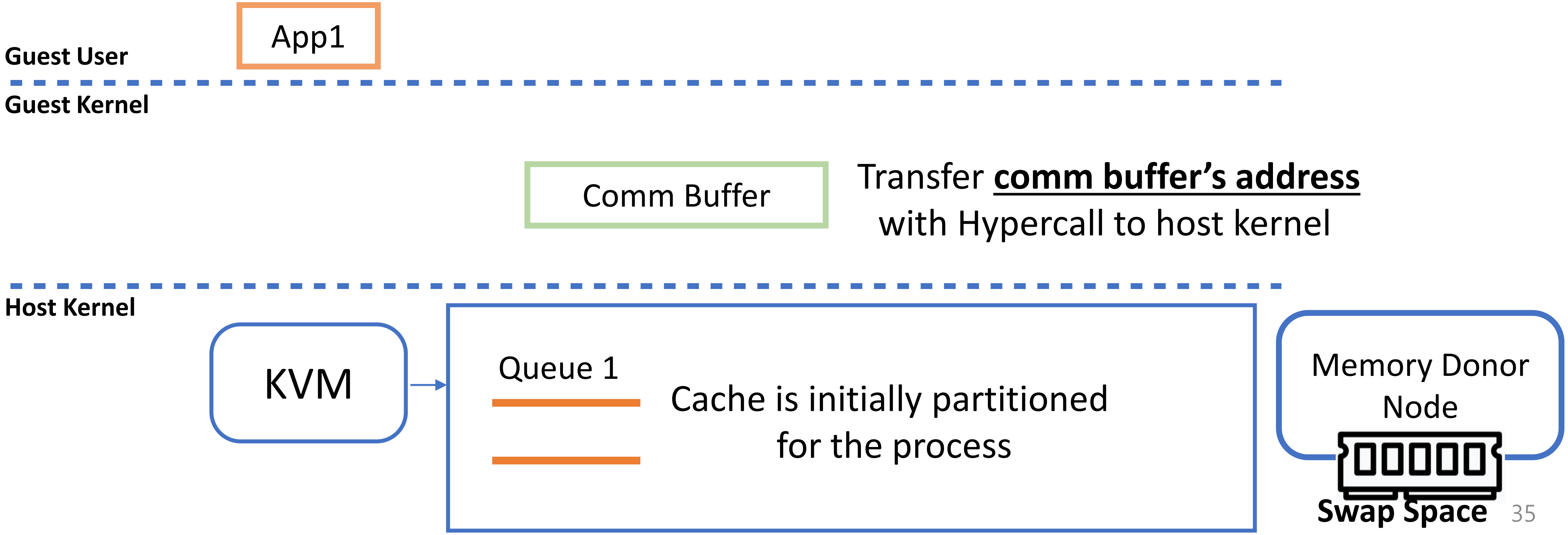


MFence

- **Page Owner Identification Module**
  - Hypercall method
  - gCR3 method

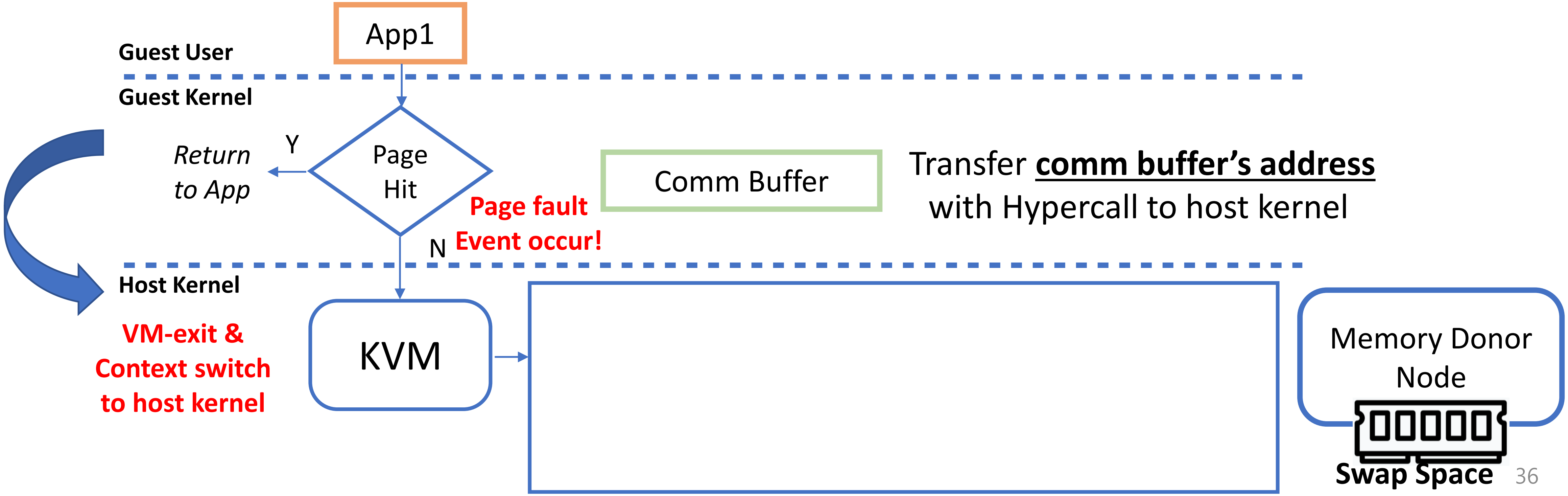
MFence

- **Page Owner Identification Module (Hypercall)**
  - It communicates with the guest kernel to find PID of a fetched page
  - Requires one Hypercall handshake to establish communication buffer



MFence

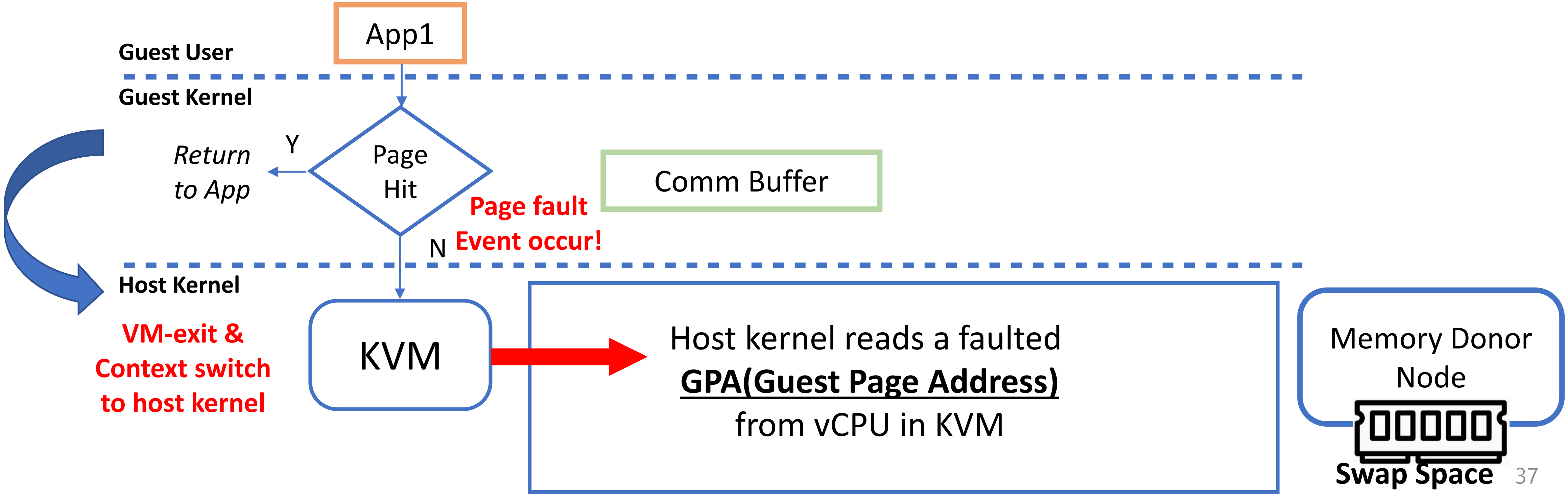
- **Page Owner Identification Module (Hypercall)**
  - It communicates with the guest kernel to find PID of a fetched page
  - Requires one Hypercall handshake to establish communication buffer



SOGANG UNIVERSITY

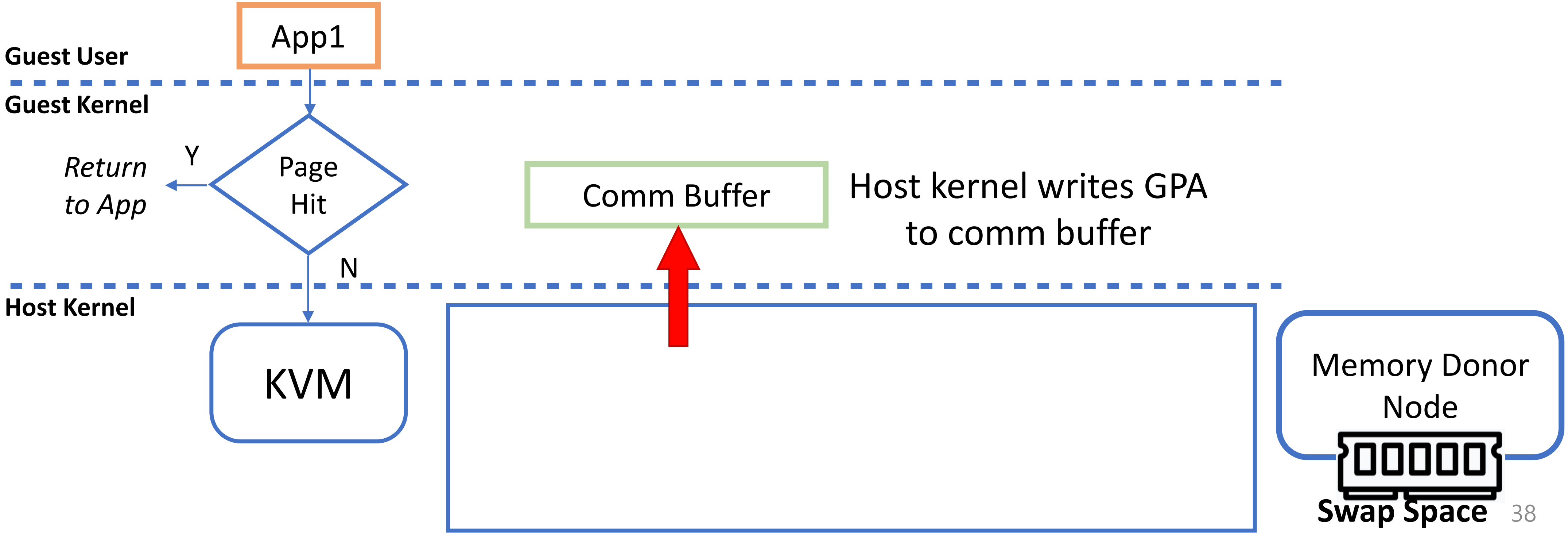
MFence

- **Page Owner Identification Module (Hypercall)**
  - It communicates with the guest kernel to find PID of a fetched page
  - Requires one Hypercall handshake to establish communication buffer



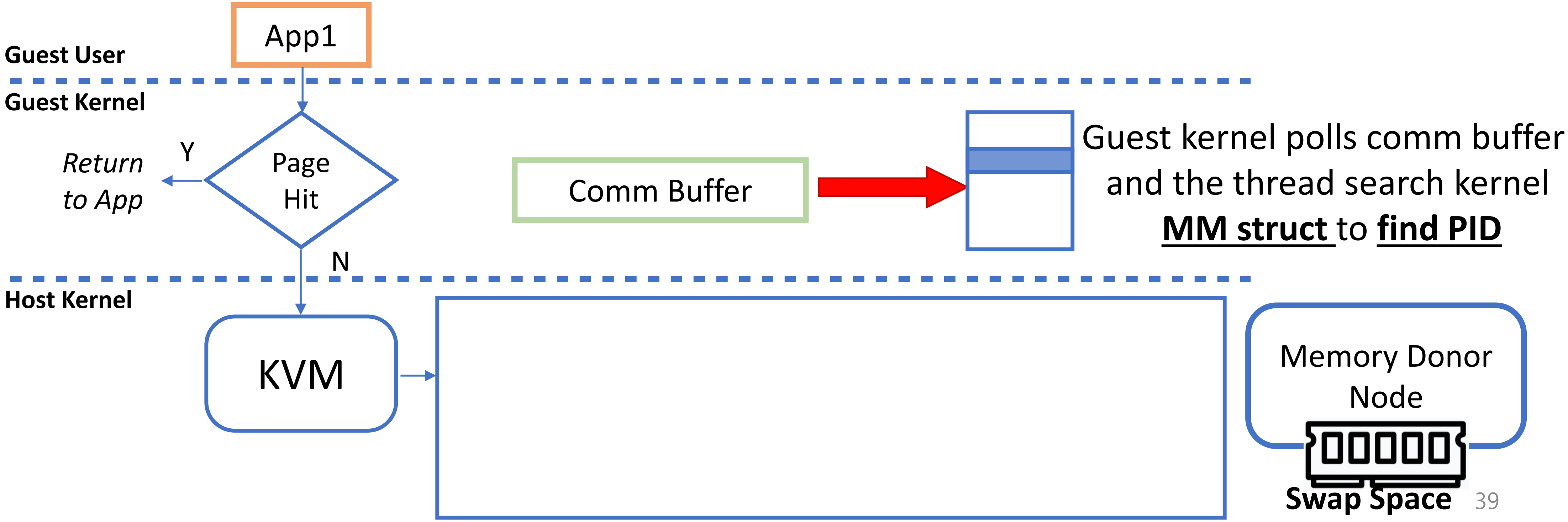
MFence

- **Page Owner Identification Module (Hypercall)**
  - It communicates with the guest kernel to find PID of a fetched page
  - Requires one Hypercall handshake to establish communication buffer



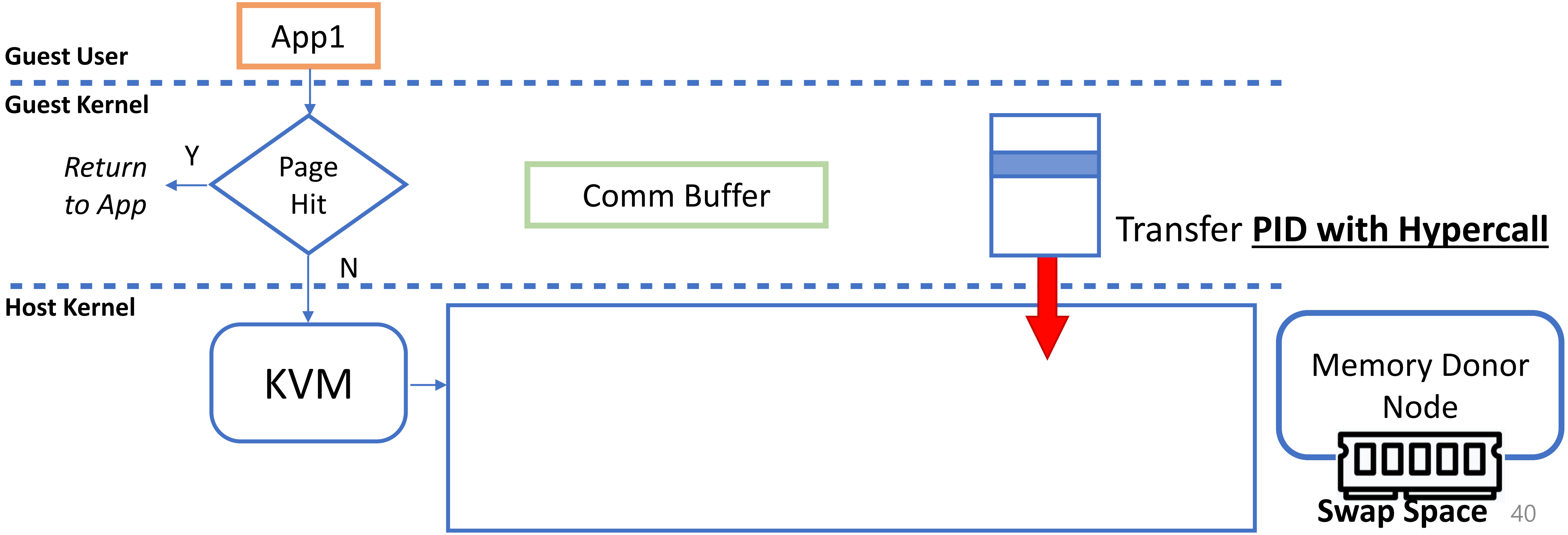
MFence

- **Page Owner Identification Module (Hypercall)**
  - It communicates with the guest kernel to find PID of a fetched page
  - Requires one Hypercall handshake to establish communication buffer



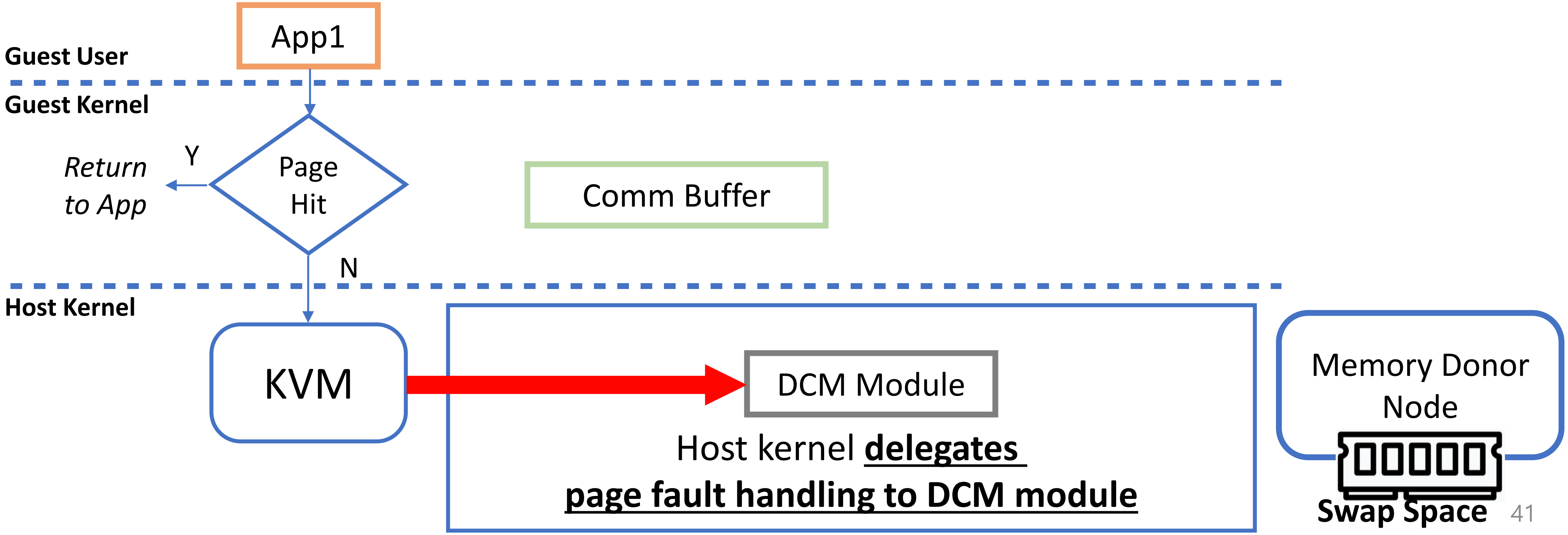
MFence

- **Page Owner Identification Module (Hypercall)**
  - It communicates with the guest kernel to find PID of a fetched page
  - Requires one Hypercall handshake to establish communication buffer



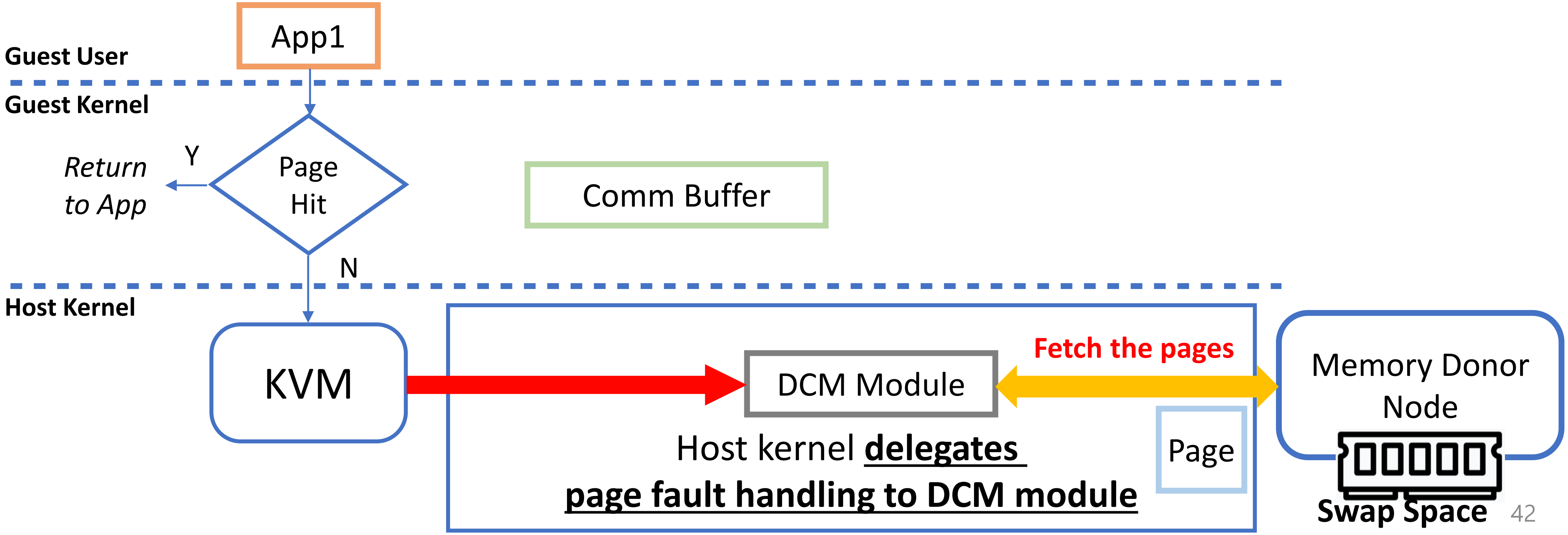
MFence

- **Page Owner Identification Module (Hypercall)**
  - It communicates with the guest kernel to find PID of a fetched page
  - Requires one Hypercall handshake to establish communication buffer



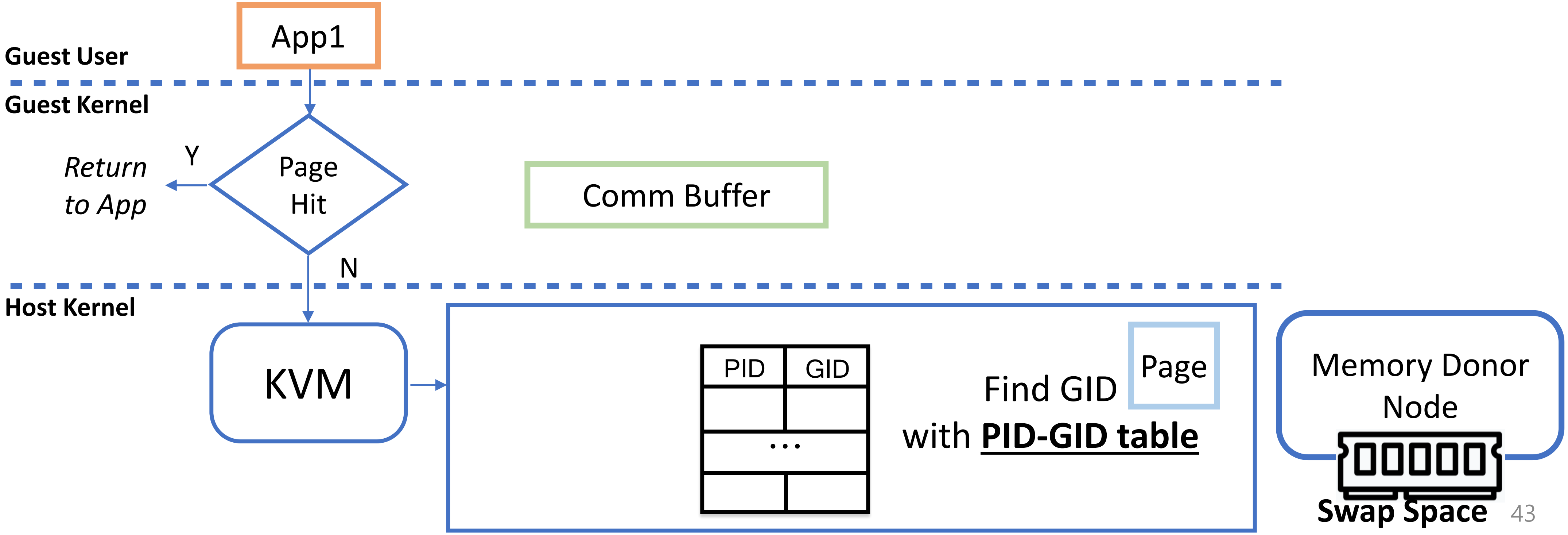
MFence

- **Page Owner Identification Module (Hypercall)**
  - It communicates with the guest kernel to find PID of a fetched page
  - Requires one Hypercall handshake to establish communication buffer



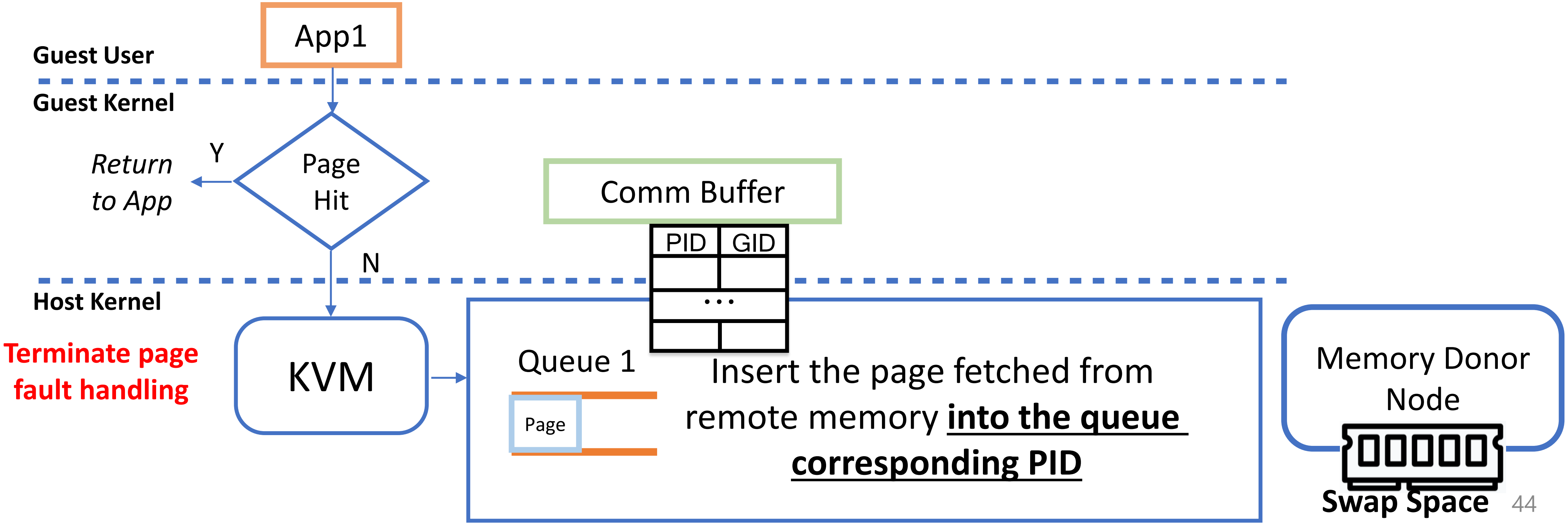
MFence

- **Page Owner Identification Module (Hypercall)**
  - It communicates with the guest kernel to find PID of a fetched page
  - Requires one Hypercall handshake to establish communication buffer



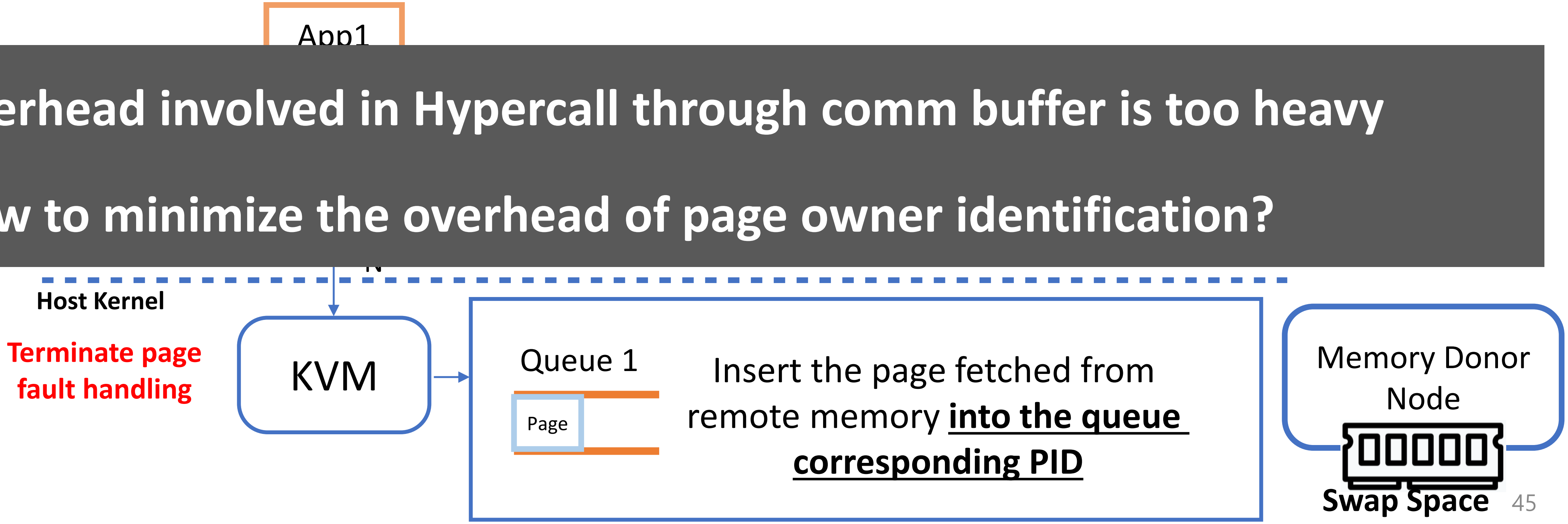
MFence

- **Page Owner Identification Module (Hypercall)**
  - It communicates with the guest kernel to find PID of a fetched page
  - Requires one Hypercall handshake to establish communication buffer



MFence

- **Page Owner Identification Module (Hypercall)**
  - It communicates with the guest kernel to find PID of a fetched page
  - Requires one Hypercall handshake to establish communication buffer

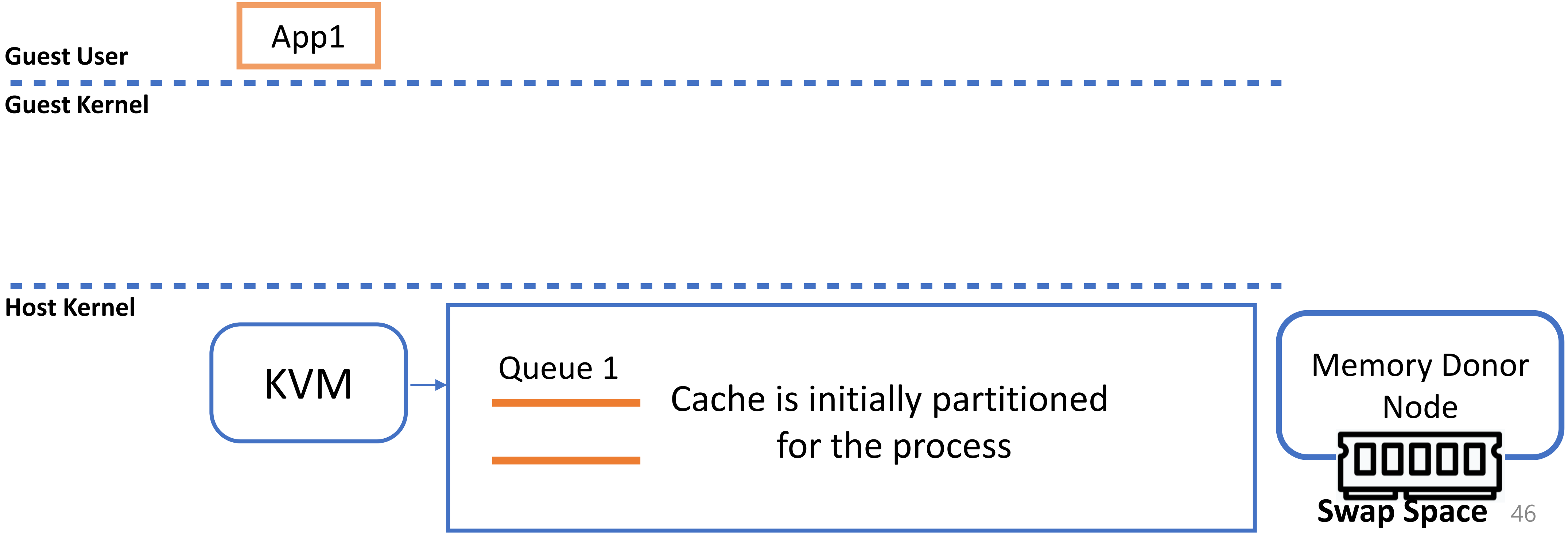


✓ **Overhead involved in Hypercall through comm buffer is too heavy**

✓ **How to minimize the overhead of page owner identification?**

MFence

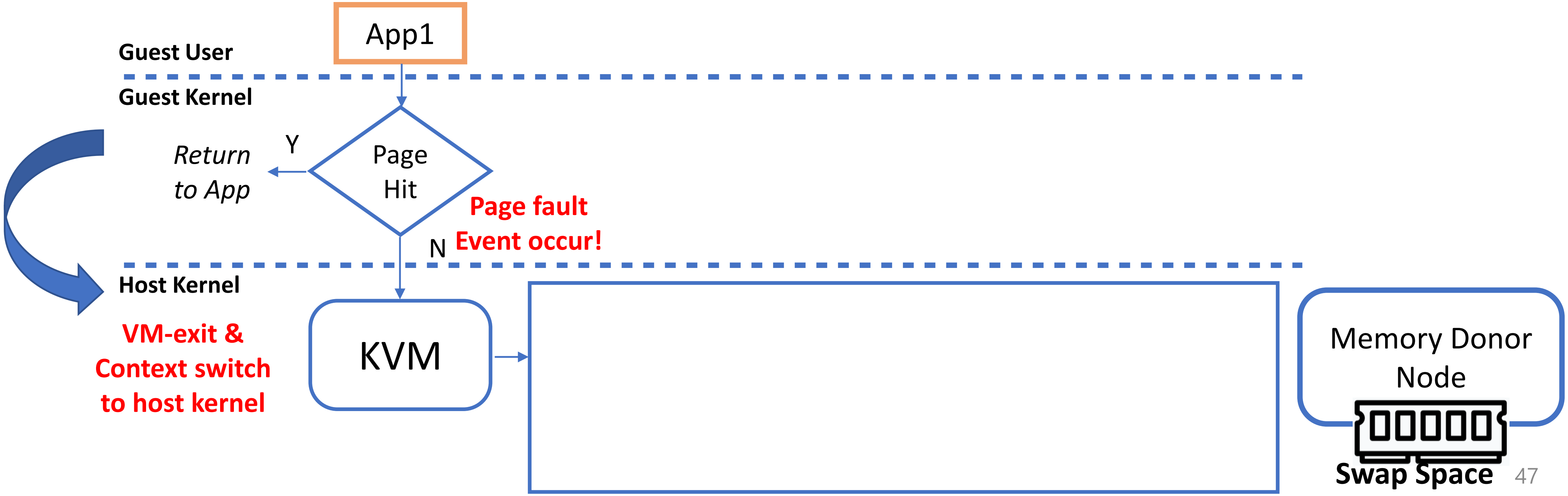
- **<u>Page Owner Identification Module (gCR3)</u>**
  - It uses PGD (Page Global Directory), the address of the page table, as a PID
  - PGD can be obtained by reading the vCPU's guest CR3 register value



SOGANG UNIVERSITY

MFence

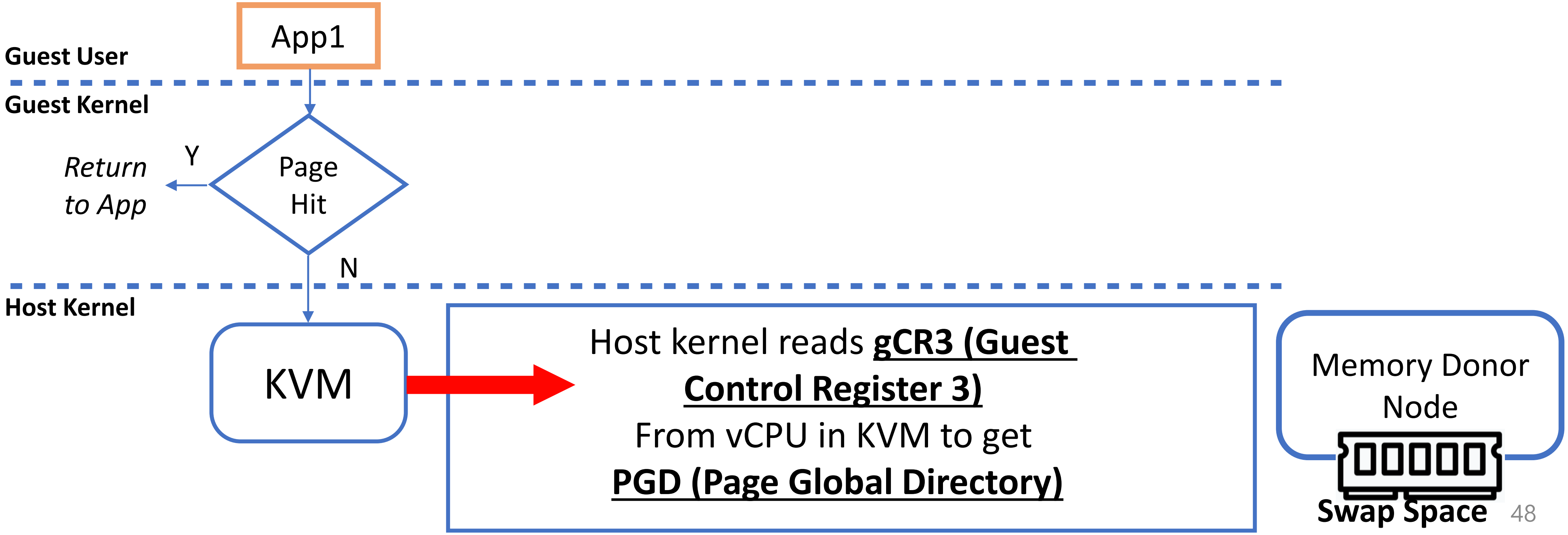
- **Page Owner Identification Module (gCR3)**
  - It uses PGD (Page Global Directory), the address of the page table, as a PID
  - PGD can be obtained by reading the vCPU's guest CR3 register value



SOGANG UNIVERSITY

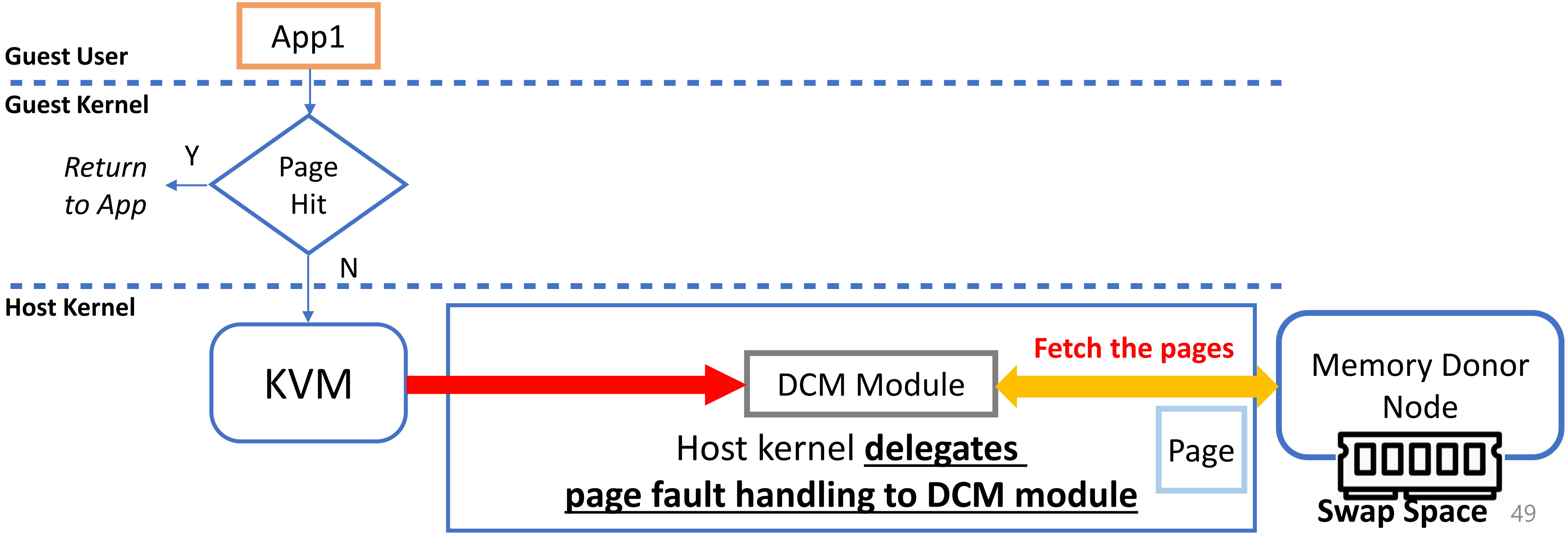
MFence

- **Page Owner Identification Module (gCR3)**
  - It communicates with the guest kernel to find PID of a fetched page
  - Requires one Hypercall handshake to establish communication buffer



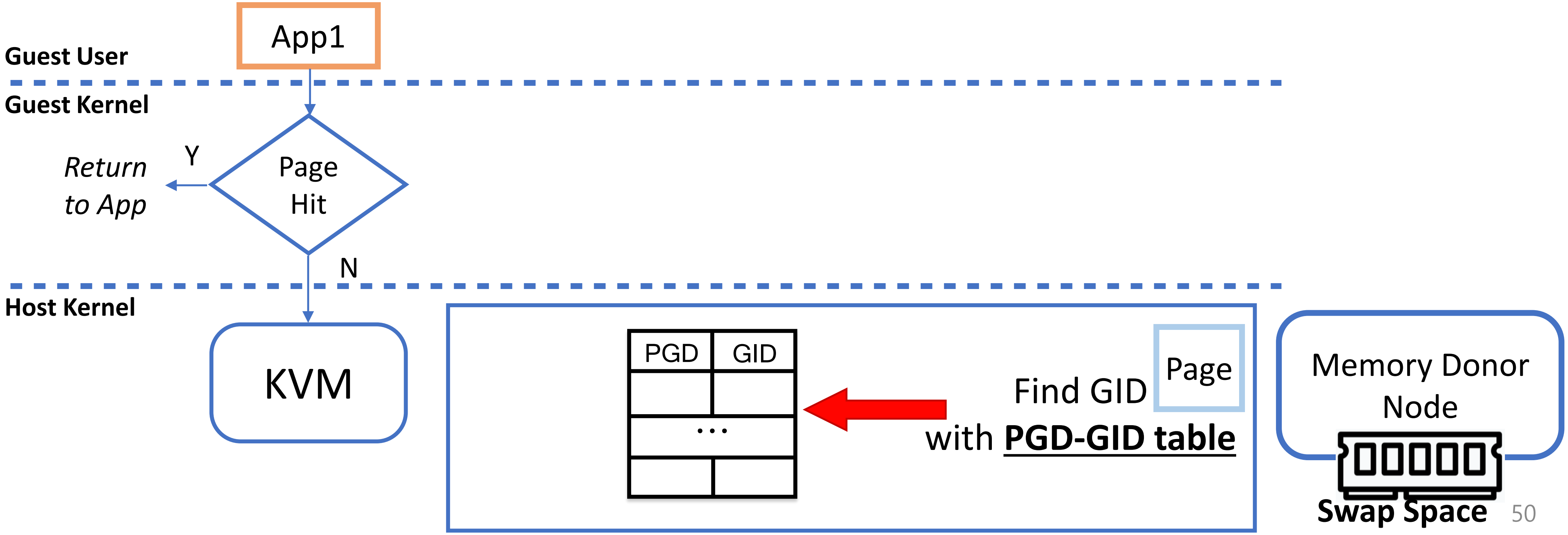
MFence

- **Page Owner Identification Module (gCR3)**
  - It communicates with the guest kernel to find PID of a fetched page
  - Requires one Hypercall handshake to establish communication buffer



MFence

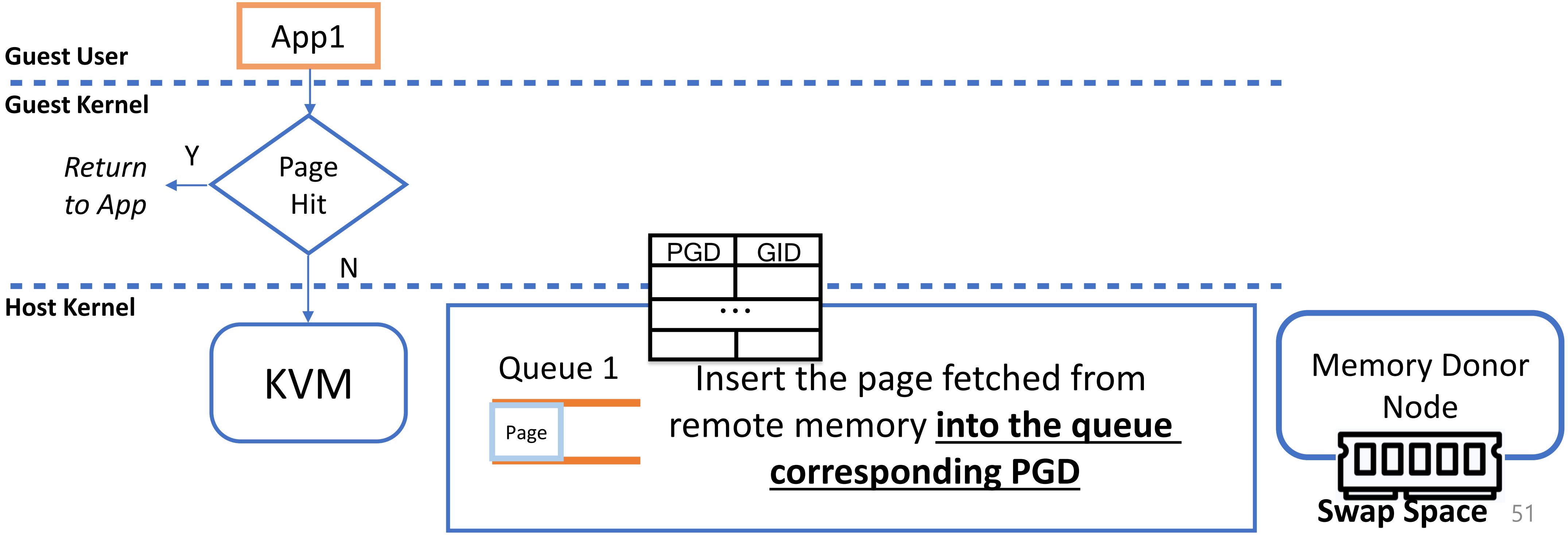
- **Page Owner Identification Module (gCR3)**
  - It communicates with the guest kernel to find PID of a fetched page
  - Requires one Hypercall handshake to establish communication buffer



SOGANG UNIVERSITY

MFence

- **Page Owner Identification Module (gCR3)**
  - It communicates with the guest kernel to find PID of a fetched page
  - Requires one Hypercall handshake to establish communication buffer



SOGANG UNIVERSITY

Agenda

- Introduction & Background
- Motivation
- Design
- **Evaluation**

Evaluation

- Evaluation Setup

CPU	Intel® Xeon Gold 6330, 2.00 GHz 28core * 2
Memory	16GB (DDR4, 3200MHz) * 8
Network	Mellanox ConnectX-5 100Gb/s EDR HCA
SSD	Intel SSD 750 (Read: 2.2 GB/s, Write: 900 MB/s)
OS	Linux kernel-4.18.0-240.10.1.el8

- Comparisons
  - ✓ DCM(A): DCM where the workload is running alone
  - ✓ DCM(S): DCM where the workload is executed in combination with other workload, which can cause interference between workload
  - ✓ MFence(**gCR3**): DCM with cache partitioning adopting gCR3
  - ✓ MFence(Hypercall): DCM with cache partitioning adopting the Hypercall

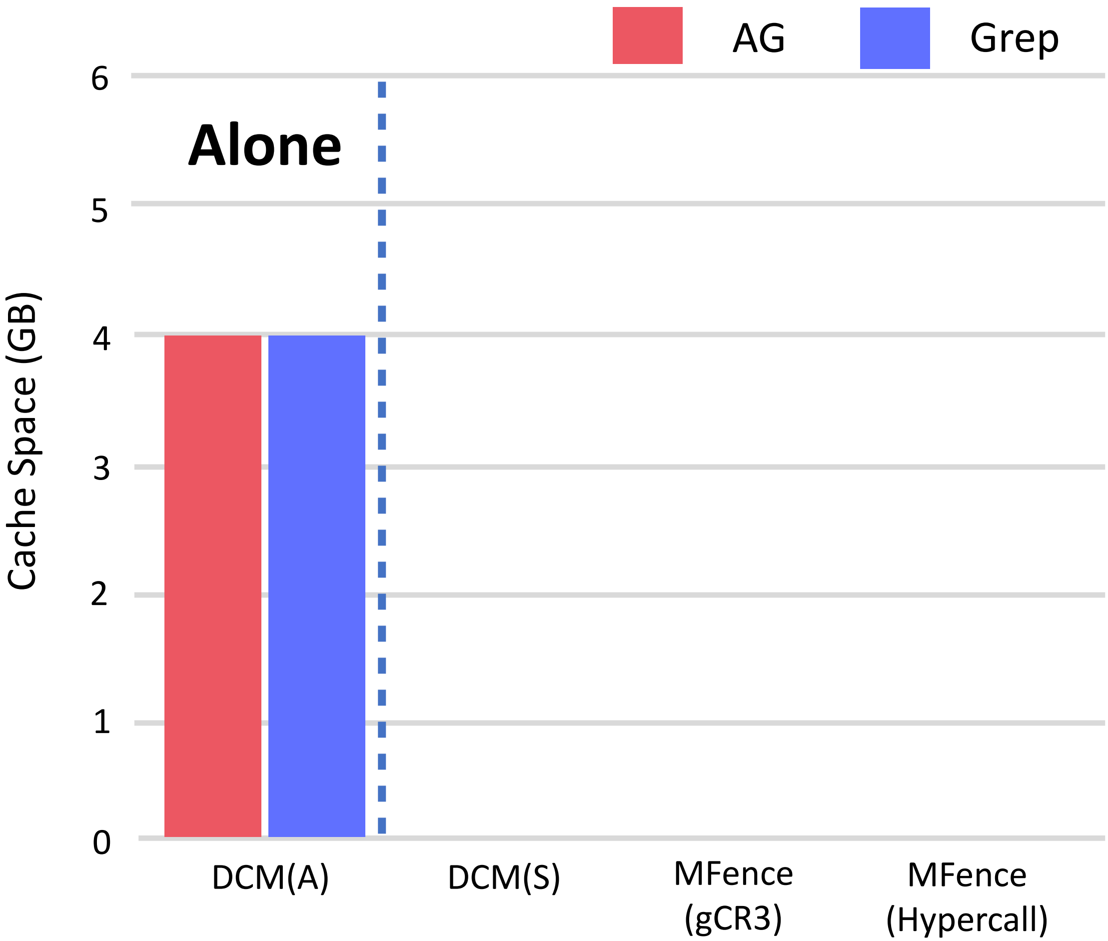
Evaluation

- Workloads (Bigdata kernel applications)
  - ✓ Grep
  - ✓ AG(Aggregation)
  - ✓ GAG(Group By Aggregation)

- Evaluation Setting
  - ✓ Memory footprint 8GB for each workload
  - ✓ Local cache size is half the sum of the workload's footprints for each comparison

Evaluation

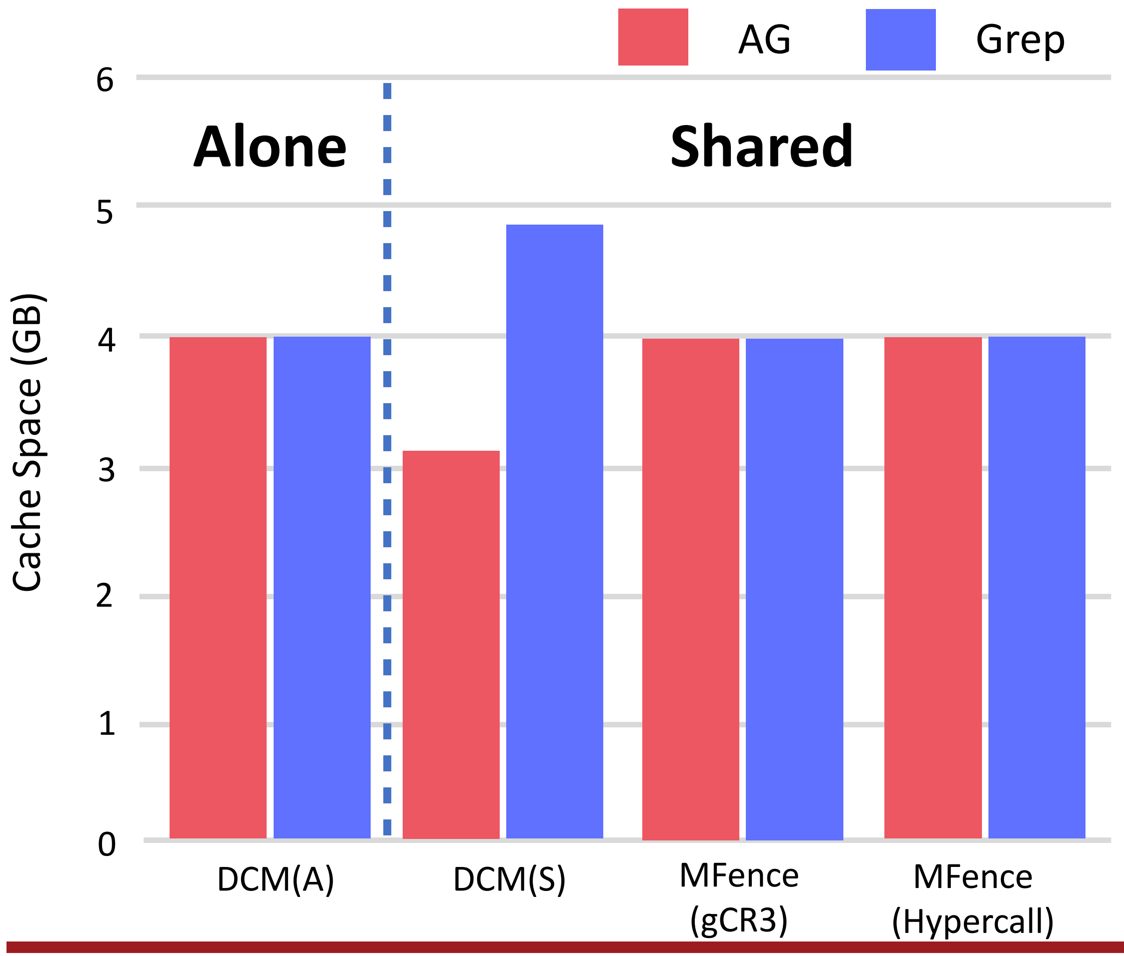
- MFence performance of cache partitioning with workload combination



SOGANG UNIVERSITY

Evaluation

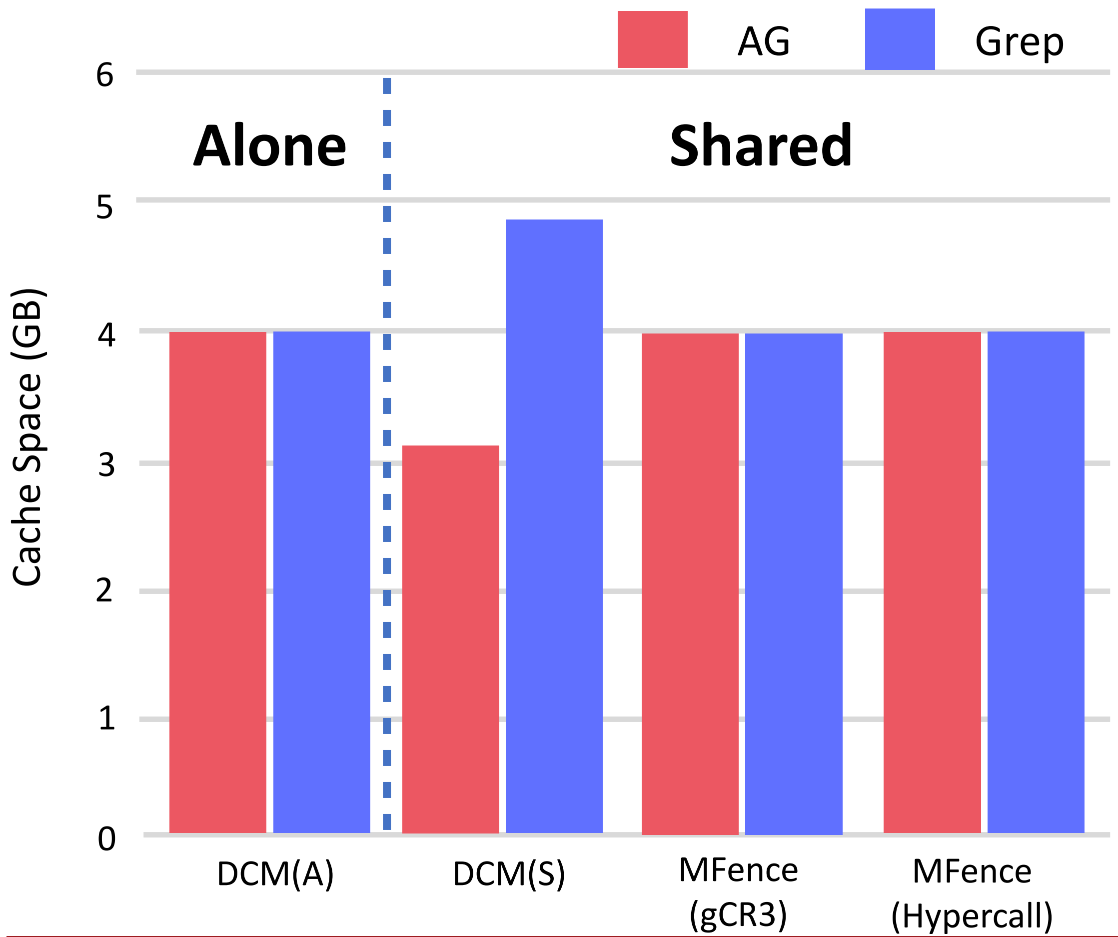
- MFence performance of cache partitioning with workload combination

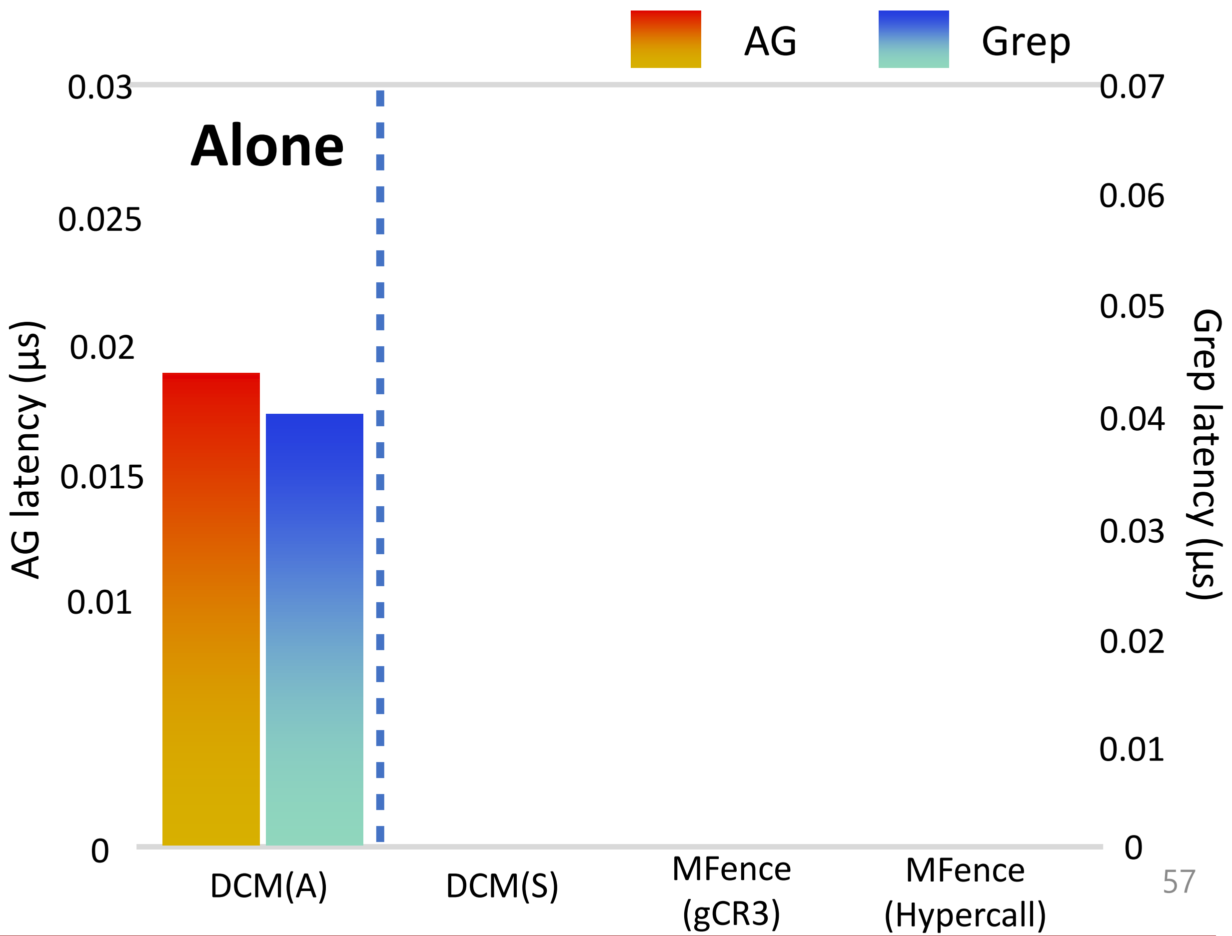


SOGANG UNIVERSITY

Evaluation

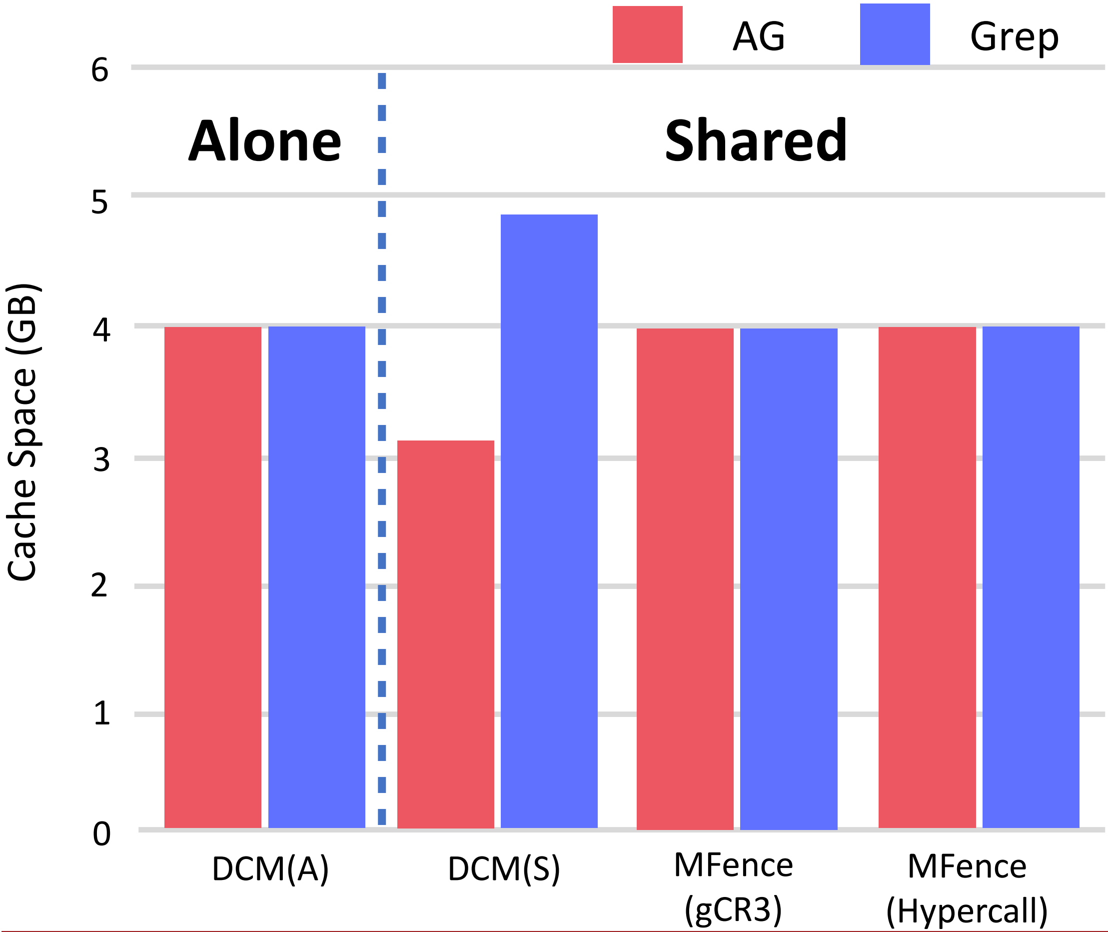
- MFence performance of cache partitioning with workload combination

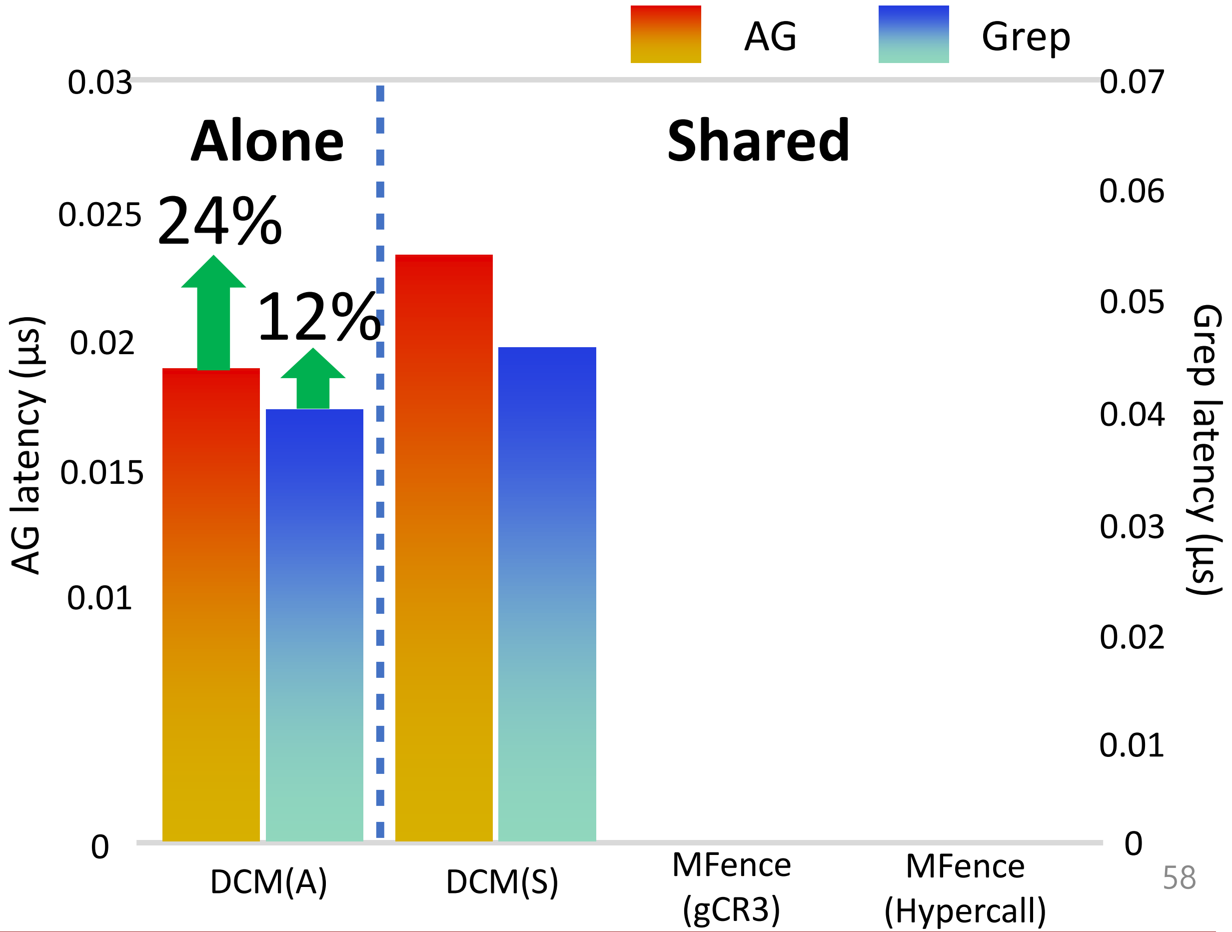




Evaluation

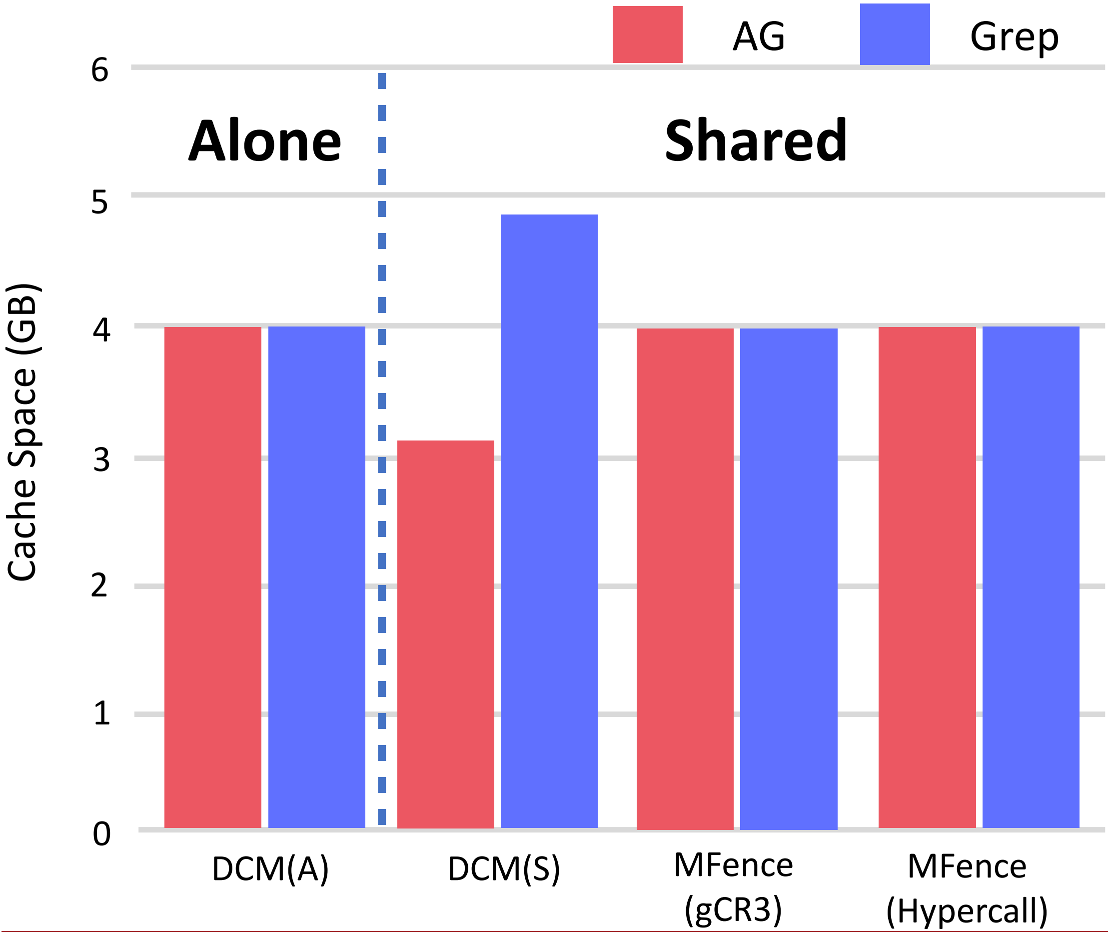
- MFence performance of cache partitioning with workload combination

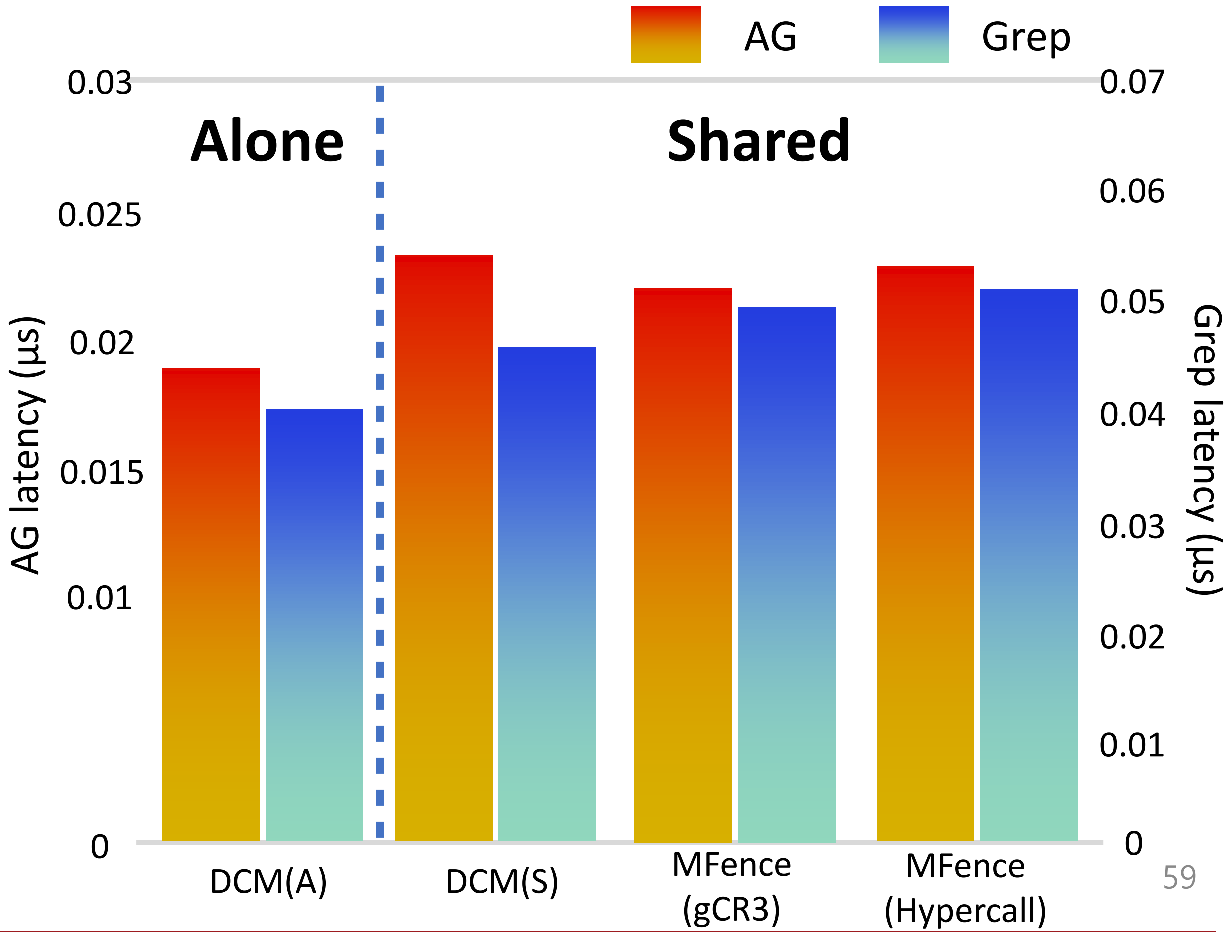




Evaluation

- MFence performance of cache partitioning with workload combination





Conclusion

- Memory races between processes on disaggregated memory platform causes memory imbalance and ultimately causes a difference in performance
- We proposed MFence, a mechanism to prevent memory race on a VM-based disaggregated memory platform (DCM)
- MFence devised a lightweight identification module (gCR3) to get the page information of the process running on guest OS from Host OS
- The experiment showed that there was no memory race when each workload combination provided half the memory of the local cache, and the corresponding change in performance was measured

Questions?

Yeonwoo Jeong

akssus12@sogang.ac.kr

DISCOS Lab, Sogang University