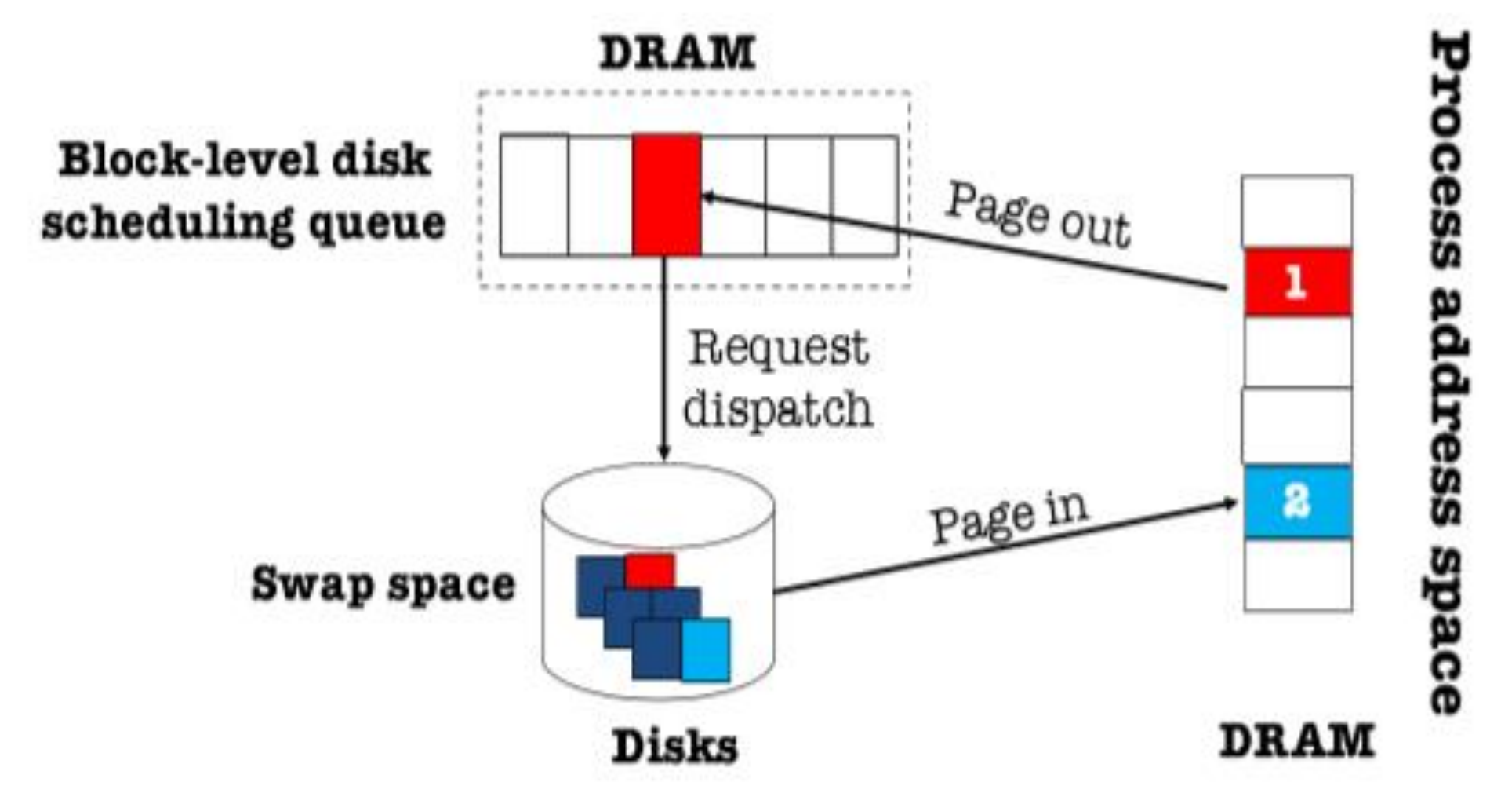
NVSwap: Latency-Aware Paging using Non-Volatile Main Memory

Yekang Wu   Xuechen Zhang

WASHINGTON STATE UNIVERSITY VANCOUVER

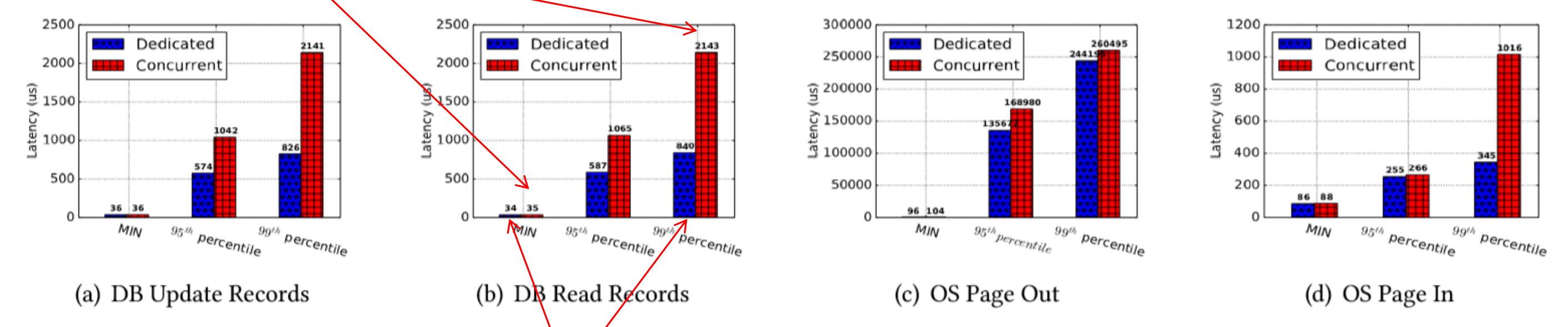
What is a paging system?



- In the Linux operating system, paging is designed to extend the main memory capacity using the space of secondary storage devices.
- A secondary storage device is a non-volatile device that holds data until it is deleted or overwritten, such as a hard disk drive or SSD.
- The system selects some pages and swaps them out to the swap space through the time of memory shortage.
- The system swaps in the pages from the swap space to DRAM if these pages were swapped out before are accessed.

Existing Problem

With concurrent accesses to the swap space, the 99th percentile latency of read operations (2143 us) is 61X higher than its minimum latency (35 us).



(a) DB Update Records (b) DB Read Records (c) OS Page Out (d) OS Page In

The 99th percentile latency of DB read operations (840 us) is 25X longer than its minimum latency (34 us).

Observations:

- During paging, both of the DB read and update operations may have a long tail latency.
- During paging, page-out requests have an extremely long tail latency.
- The Linux paging system is not able to enforce latency bounds.
- The OS page-in latency has a critical impact on the latency of DB operations at the application level.

In summary, while the paging systems have been well implemented to provide high I/O bandwidth during paging, they are not latency-sensitive.

Related Work

Technique	*In-situ Paging-in*	*Write Awareness*	*Latency Enforcement*	*Swap Device Type*	*Swapping Unit*
Linux	No	No	No	Disk	Page
FlashVM	No	Yes	No	Flash	Page
SmartSwap	No	No	Some	Flash	Application
Mars	No	Some	No	Flash	Application
Memorage	Yes	No	No	NVMM	Page/block
Dr. swap	Yes	No	No	NVMM	Page
Refinery swap	Some	Yes	No	NVMM	Page

Our Design: NVSwap

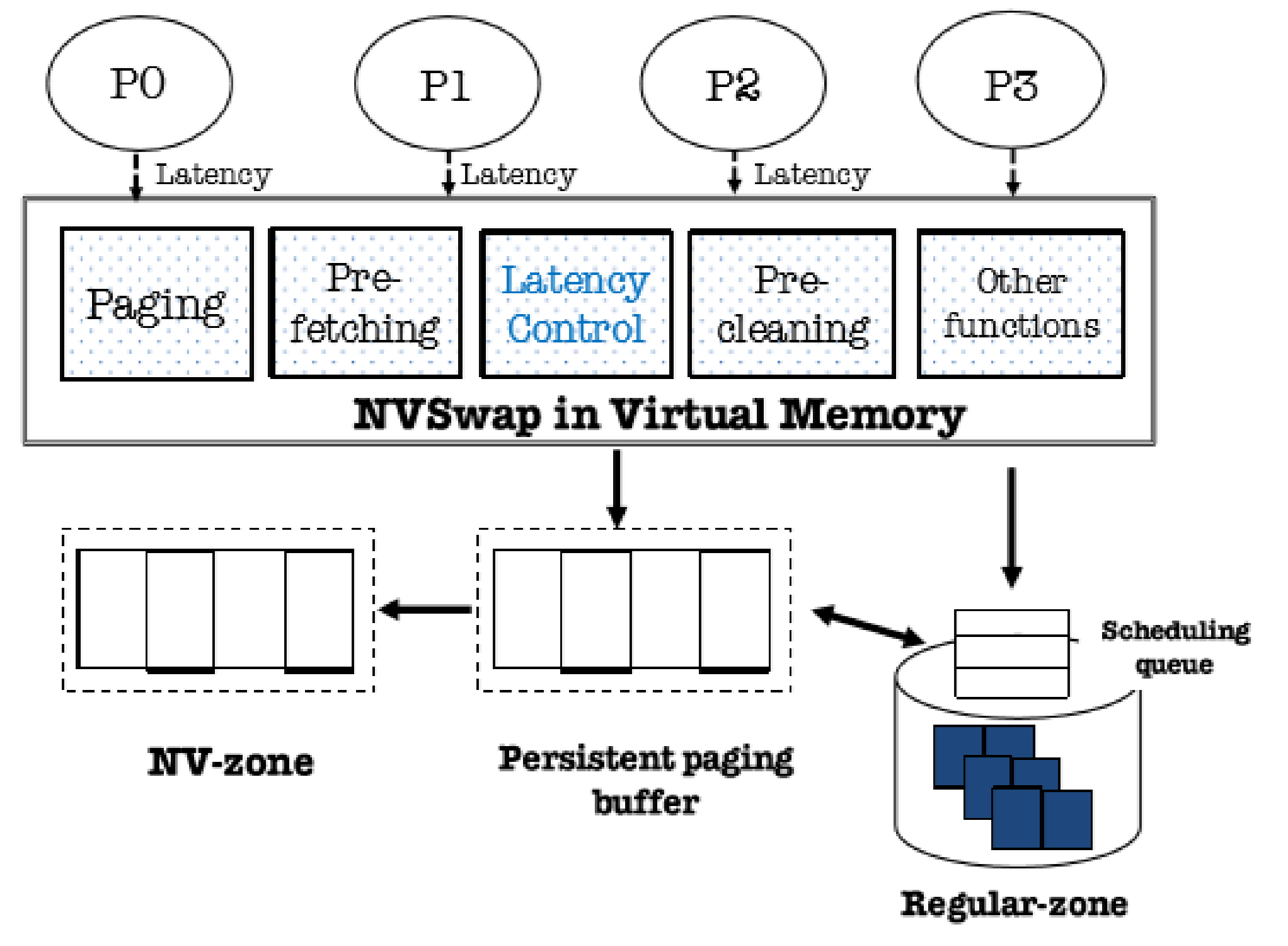
NVSwap	**Yes**	**Supported**	**Yes**	NVMM+Flash	Page

The New Design: NVSwap

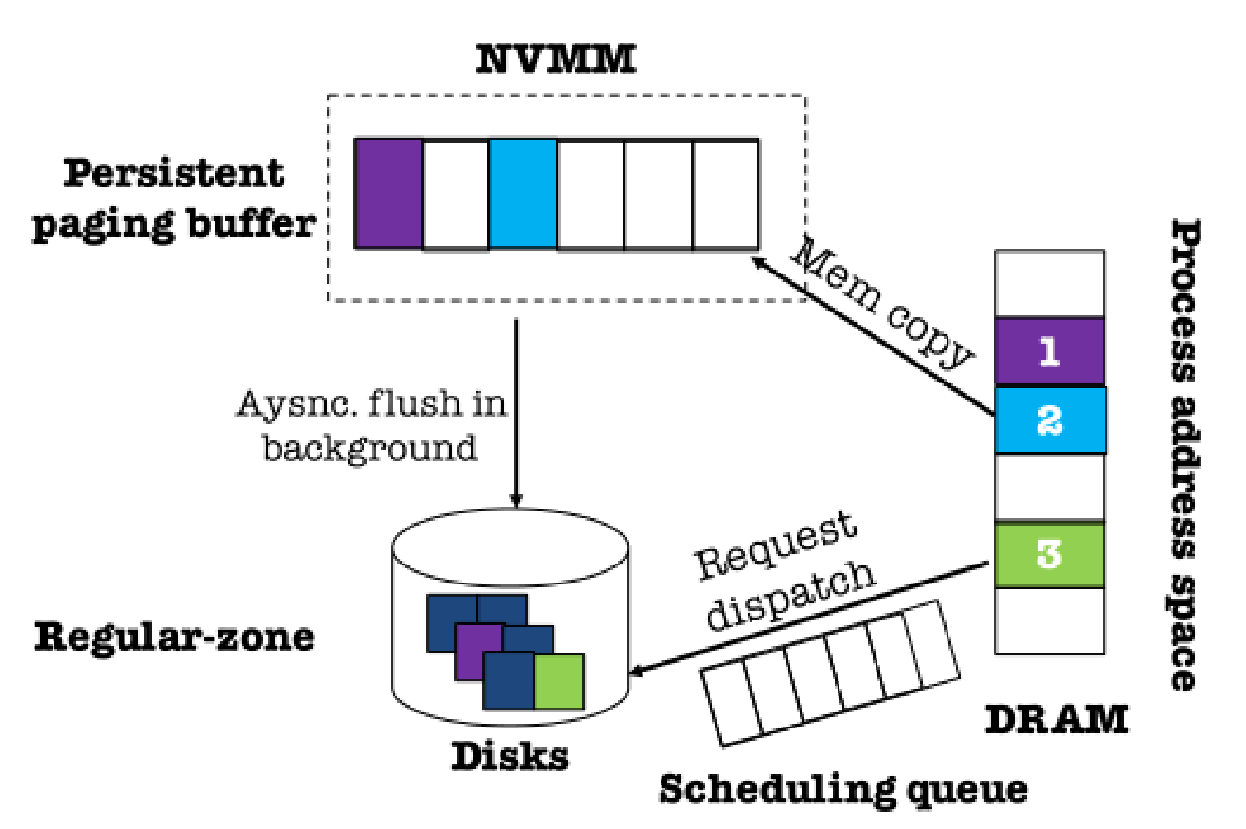
NVSwap: A latency-aware paging mechanism

Four main components: a regular-zone, an NV-zone, a persistent paging buffer and a shadow mapping table.

- The regular-zone, hosted on block storage devices, serves paging requests dispatched from a disk scheduling queue.
- The NV-zone and persistent paging buffer, hosted in NVMM, serve paging requests to enforce the latency bounds.
- The NV-zone consists of NVMM page frames that can be directly accessed in process address spaces.
- The persistent paging buffer stores swapped-out pages from latency-sensitive processes and prefetched pages from the regular-zone.
- NVSwap asynchronously flushes pages to the regular-zone in background when the buffer is full.
- The new disk location of the flushed pages in the regular-zone is recorded in the shadow mapping table.
- The page-in requests to access the flushed pages will be served using the new disk addresses in the shadow mapping table.



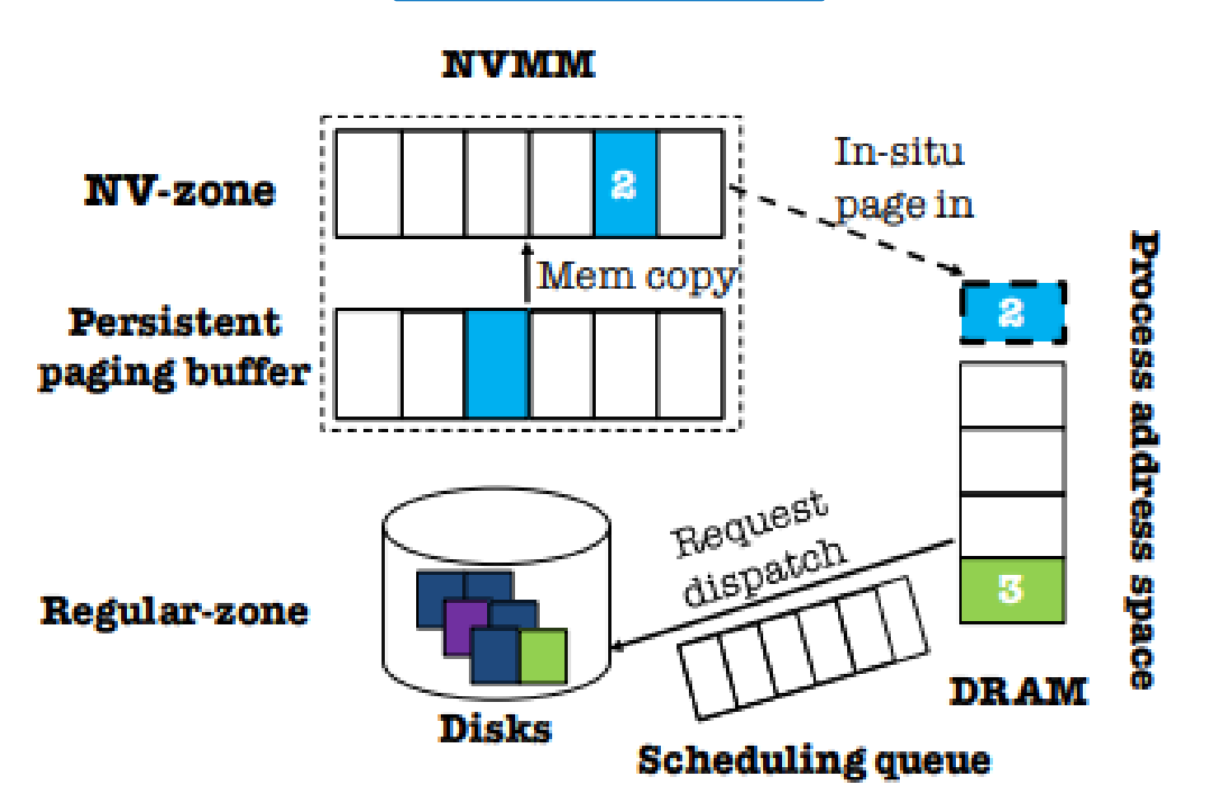
Page out



Three page-out paths of NVSwap:

- Page (1):
  1. It is paged out to the persistent paging buffer first and asynchronously flushed to the regular-zone when the buffer is full.
  2. NVSwap copies the page to be swapped out from DRAM to a new page frame in NVMM.
  3. A page-out request can be considered complete once it is sent to the main memory extension.
- Page (2): It is paged out to NVMM.
- Page (3) : It is paged out to the regular-zone.

Page in



Two paths for serving the page-in request:

- If the page is stored in the regular-zone:
  1. NVSwap issues a read request to read the page from the block device to a new DRAM page frame.
  2. By updating the page table entry of the process, it sets up the PTE mapping from the virtual address to the physical address in DRAM.
- If the page is stored in the persistent paging buffer:
  1. NVSwap allocates a page frame in the NV-zone.
  2. It sets up the PTE mapping from the virtual address to the physical address in NVMM.
  3. It copies the data from the persistent paging buffer to the NVMM page frame in the NV-zone.

Experimental Setup

Hardware

- CPU: 6-core Intel Xeon CPU X5670 2.93 GHz CPU
- DRAM: 32 GB
- Disk:
  (1) 1TB hard disk (Seagate Barracuda 7200.12) (host the operating system)
  (2) 128 GB SSD (OCZ-VERTEX 4) (host the regular-zone)

Kernel

- Kernel version: kernel-3.16.74

Main Memory & NVMM

- Main Memory: 5 GB
- NVMM: 10 GB

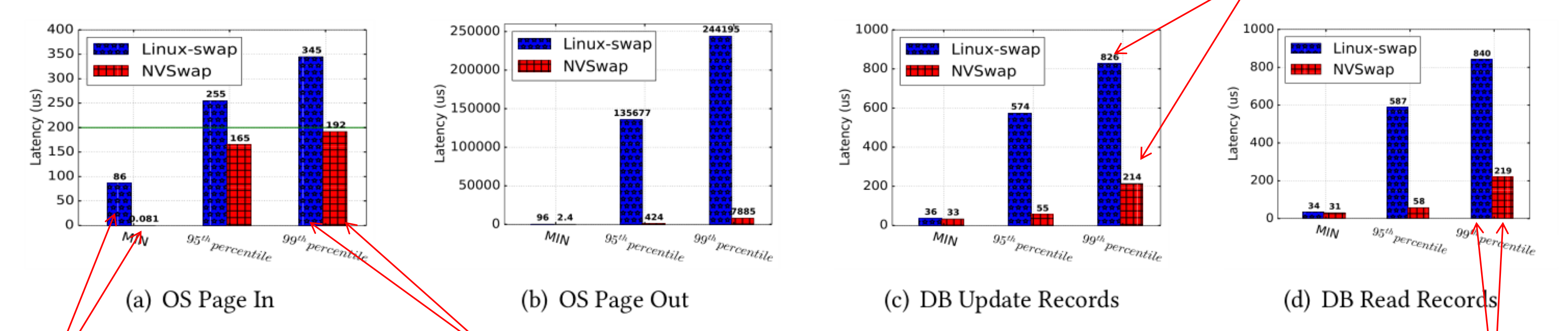
Benchmark

- **Memcached**: Free & open source, high-performance, distributed memory object caching system.
- **YCSB**: an open-source specification and program suite for evaluating retrieval and maintenance capabilities of computer programs.

Evaluation

Single Workloads

The 99th percentile latency: 826 us -> 214 us



(a) OS Page In (b) OS Page Out (c) DB Update Records (d) DB Read Records

The minimum latency: 86 us -> 0.081 us

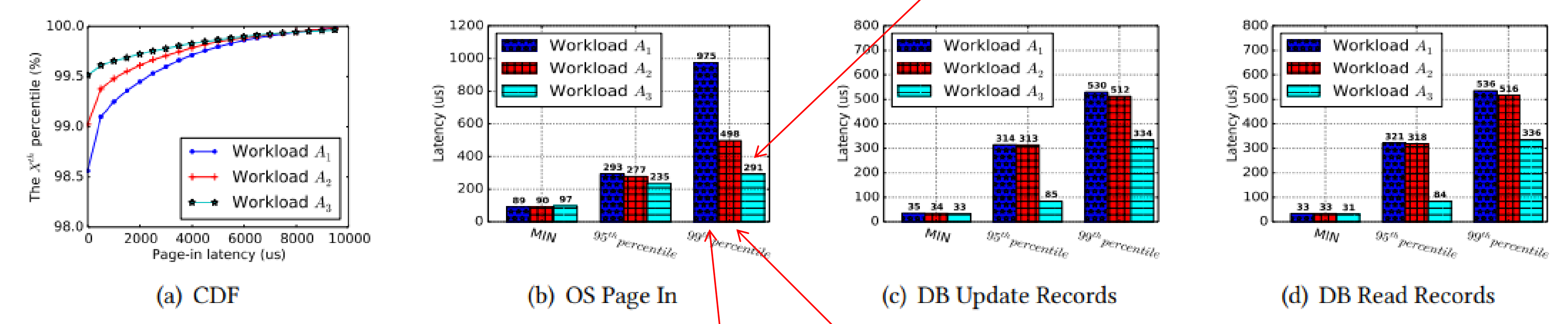
The 99th percentile latency: 345 us -> 192 us, below the latency bound 200 us

The 99th percentile latency: 840 us -> 219 us

We study the effectiveness of latency control using NVSwap with a single memcached server accessing the swap space in the experiment. The result shows the effectiveness of NVSwap in enforcing latency bounds.

Concurrent Homogeneous Workloads

The 99th percentile latency: 291 us, below the latency bound 300 us



(a) CDF (b) OS Page In (c) DB Update Records (d) DB Read Records

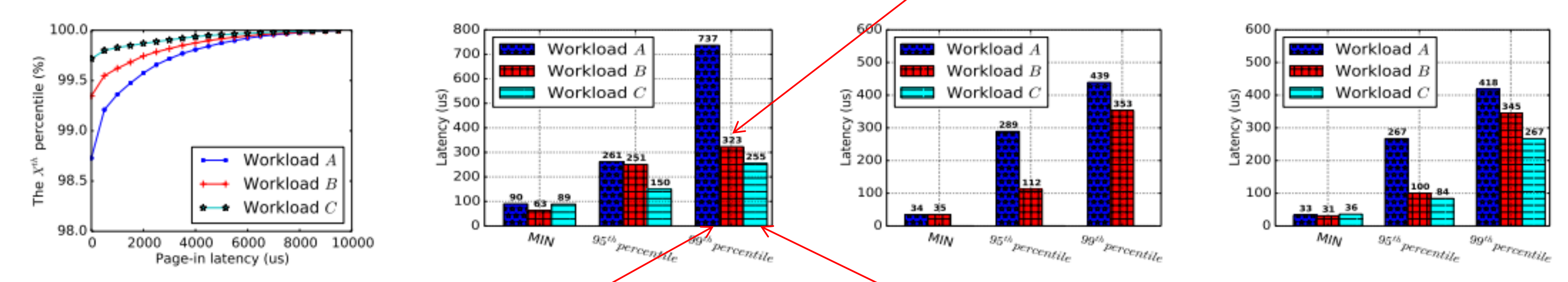
The 99th percentile latency: 975 us, below the latency bound 1000 us

The 99th percentile latency: 498us, below the latency bound 500 us

We study the effectiveness of NVSwap with three concurrent homogeneous memcached servers accessing the swap space. The result shows that all workloads meet the latency requirements.

Concurrent Heterogeneous Workloads

The 99th percentile latency: 323 us, below the latency bound 500 us



(a) CDF (b) OS Page In (c) DB Update Records (d) DB Read Records

Workload	*Read%*	*Update%*	*99% Latency Bound*
A	50%	50%	1000 us
B	95%	5%	500 us
C	100%	0%	300 us

Configurations of heterogeneous Workload A, B, C.

The 99th percentile latency: 737 us, below the latency bound 1000 us

The 99th percentile latency: 255 us, below the latency bound 300 us

We study the effectiveness of NVSwap with three concurrent heterogeneous memcached servers accessing the swap space. The result shows that all workloads meet the latency requirements.

Conclusion

- NVSwap provides a cost-effective and hybrid swap space using both NVMM and SSD.
- NVSwap allows the setting of Xth percentile page-in latency bound for a single process or a group of processes.
- NVSwap can enforce the tail latency while providing strong performance isolation for latency-sensitive processes.

Submission ID: 2168