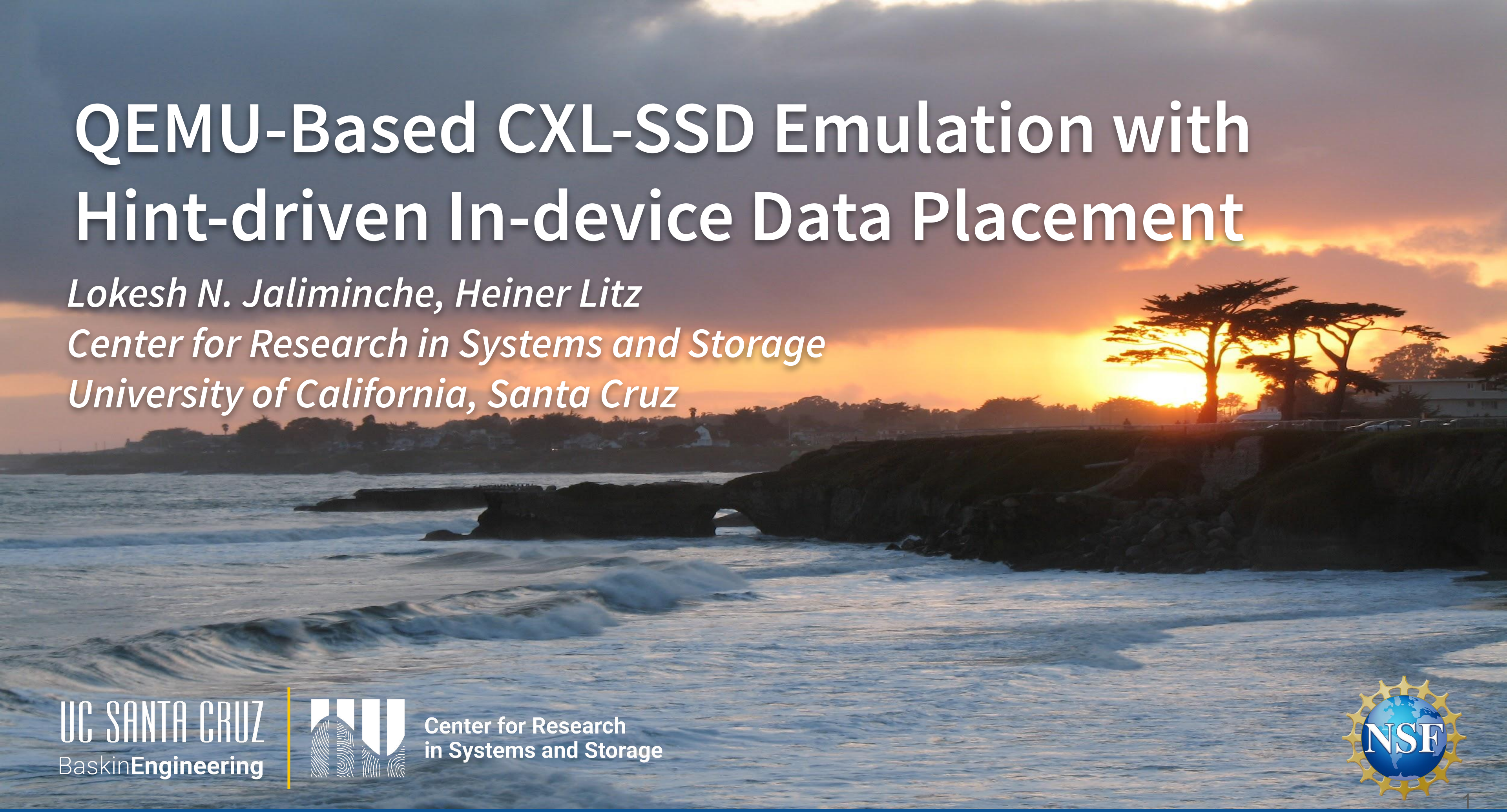


QEMU-Based CXL-SSD Emulation with Hint-driven In-device Data Placement

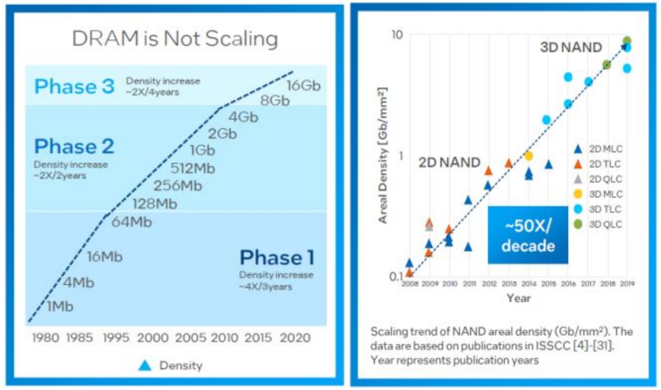
Lokesh N. Jaliminche, Heiner Litz
Center for Research in Systems and Storage
University of California, Santa Cruz

UC SANTA CRUZ
BaskinEngineering

Center for Research in Systems and Storage

Background and Motivation

- **Memory Demands of Modern Applications:** Applications like large language models (LLMs) and deep learning recommendation models (DLRMs) require substantial memory resources, **hitting memory capacity wall.**
- **Solution Exploration:** Researchers are investigating **hybrid devices as an alternative for expanding working memory capacity**.
- **Hybrid Device Approach with CXL:** These devices pair a small, high-speed DRAM cache with large-capacity storage media (e.g., NAND Flash) and are a**ccessible over a CXL interconnect with memory-like semantics.**

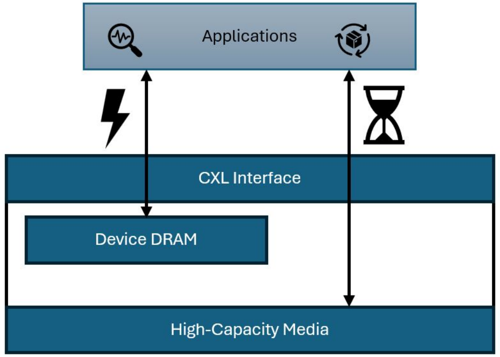


Storage Technology Density Scaling Trends

Source:https://infohub.delltechnologies.com/l/new-vmware-vsphere-memory-techniques-in-dell-emc-hyperconverged-infrastructure-solutions/technology-background

Background and Motivation

- While these devices can **address capacity and TCO concerns**, **direct accesses** from Flash media **can degrade application performance.**
- It becomes **critical to research data placement strategies** that help avoid Flash Accesses (Access slower media) in critical path of the application.
- However since these devices are still in its early stages they are difficult to obtain and building a real prototype can be time consuming.
- **QEMU based emulation of CXL-SSD**
  - Hardware independent
  - Easy to use and distribute



Performance Gap Between DRAM and Flash

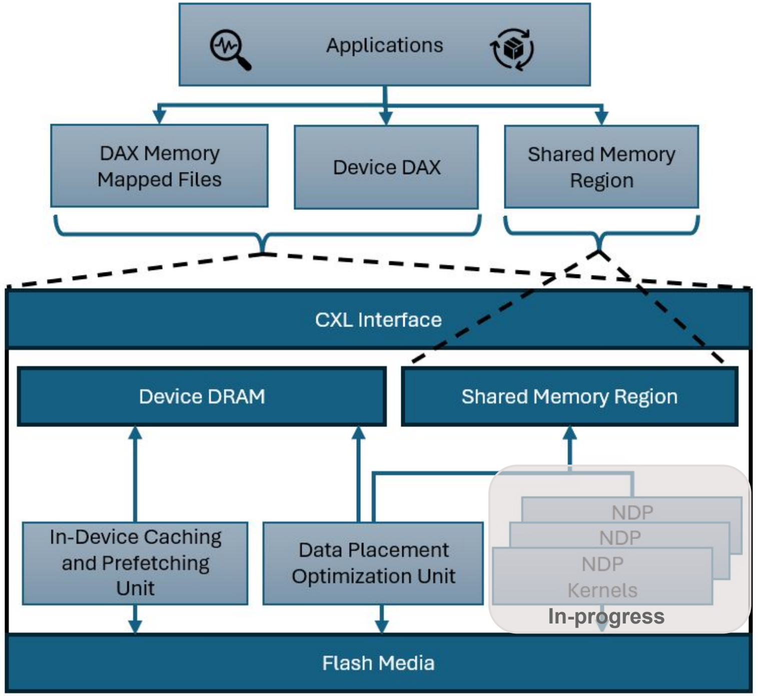
Key Research Questions

- ❖ **Device-Side:**
  - ➢ What are the optimal **caching and prefetching** policies?
  - ➢ Are in-device data placement strategies sufficient?
  - ➢ How **in-device data placement strategies leverage application hints/Context**?

- ❖ **Application-Side:**
  - ➢ What Kind of **Performance impact applications** should expect?
  - ➢ Should applications have **explicit control over data placement** within the device?
  - ➢ How can **applications aid in-device data placement strategies**?

Current Progress

- **In-Device Caching and Prefetching Unit**
  - Telemetry unit to observe memory accesses
  - Implements Caching and Prefetching Unit
- **Data Placement Optimization Unit**
  - Can receive hints from Host and drive corresponding optimizations
- **Shared Memory Region**
  - Passing hints and instantiating NDP kernels, passing arguments



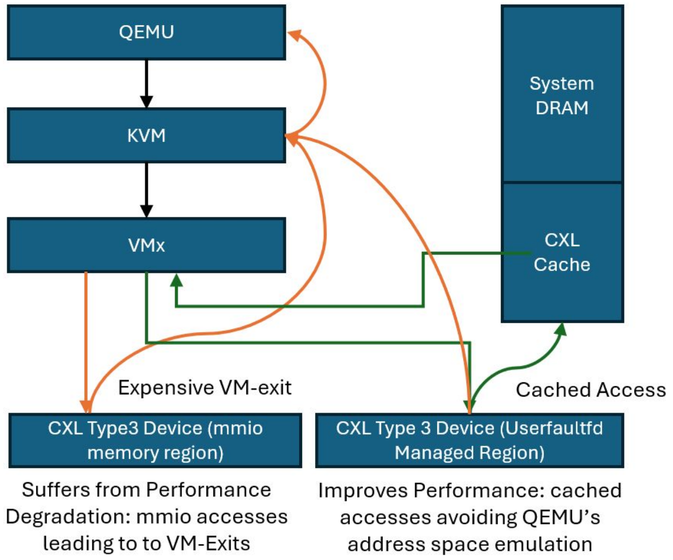
HIgh Level Device Architecture

CRSS

LRU Caching Policy

Recap : LRU Implementation Challenges

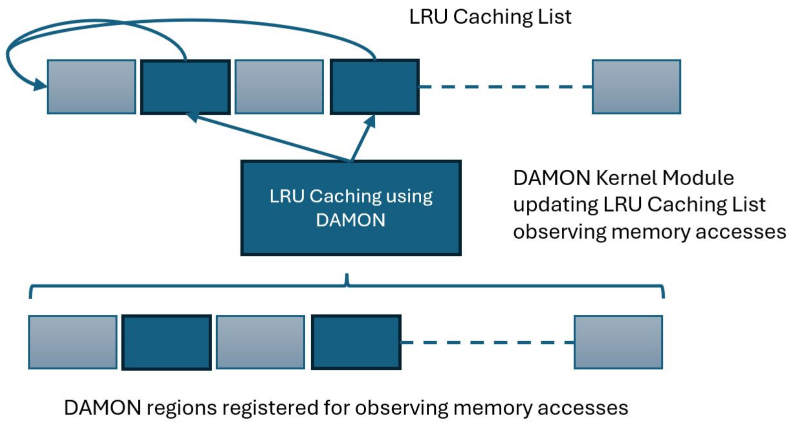
- **Performance Workaround:** Mitigate performance degradation from VM-exits on each CXL access, a core limitation of QEMU.
- **Solution:** Bypass QEMU's address space emulation by implementing a Userfaultfd-managed memory region.
- **Direct Cache Access:** Allow VMx to access the CXL cache directly, avoiding VM-exits for cached accesses.
- **Limitation:** While performance improves, this approach prevents intercepting memory accesses, restricting feedback-based caching techniques like LRU.


QEMU
KVM
VMx
System DRAM
CXL Cache
Expensive VM-exit
Cached Access
CXL Type3 Device (mmio memory region)
CXL Type 3 Device (Userfaultfd Managed Region)
Suffers from Performance Degradation: mmio accesses leading to to VM-Exits
Improves Performance: cached accesses avoiding QEMU's address space emulation

Addressing Performance Issues with QEMU

LRU Implementation with DAMON

- **Monitor with DAMON:** Utilize DAMON, a Linux tool, to efficiently monitor memory access patterns with low overhead.
- **Cache Miss Handling:** Each access triggers addition of the corresponding uncached page to the LRU list and registration with DAMON to observe memory accesses.
- **Update LRU Position:** At each sampling interval, move accessed pages to the front of the LRU list.


LRU Caching List
LRU Caching using DAMON
DAMON Kernel Module updating LRU Caching List observing memory accesses
DAMON regions registered for observing memory accesses

LRU Caching with DAMON Framework

Experiment Setup

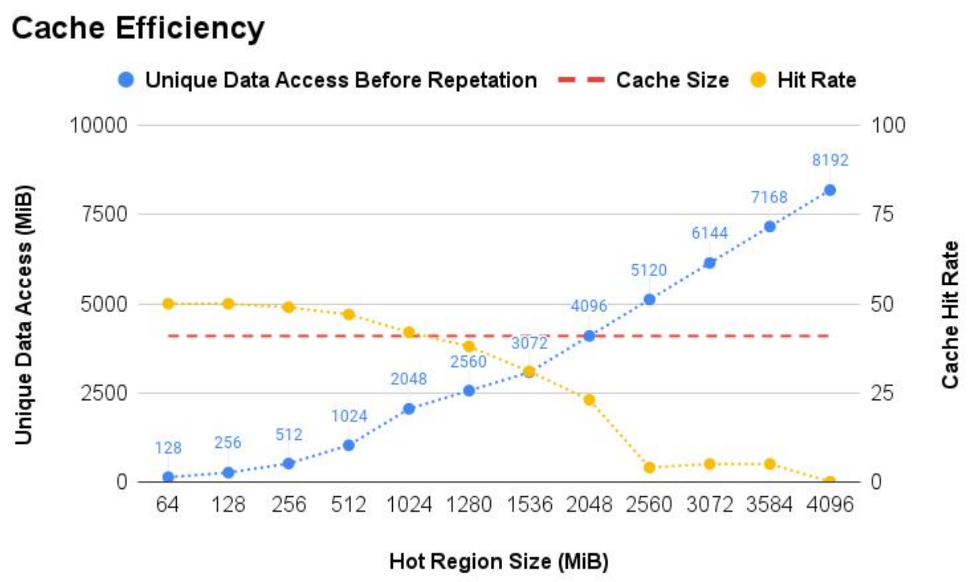
- **CXL-SSD Configuration**
  - Device DRAM : 4GB
  - Backend Store : 32 GB

- **Workloads**
  - **Sequential workload simulating hot regions**
    - Hot Workload : 10 GiB (repetitively access data only from hot region)
    - Streaming Workload : 10GiB

- **Random workload simulating hot regions with different access distributions**
  - Normal (Gaussian) Distribution (Scale Std. Deviation 0 - 100)
  - Zipf (Scale Zipf Theta 0.1 to 2.5)
  - Pareto (Scale Pareto Power 0.1 to 0.9)
  - Zoned Distribution

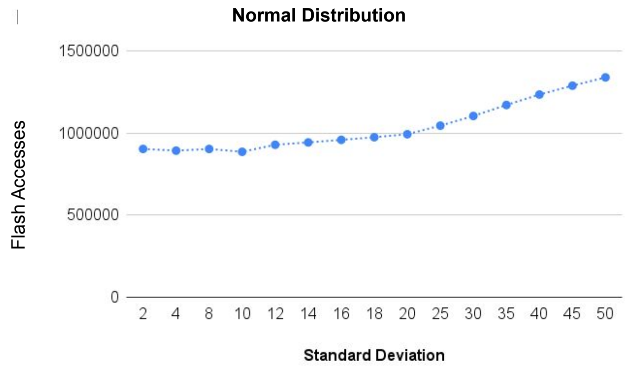
- **Observe if hot pages are being retained in the cache**

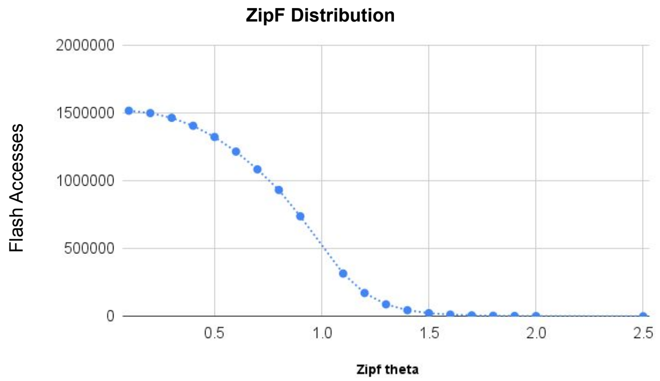
LRU Cache Validation

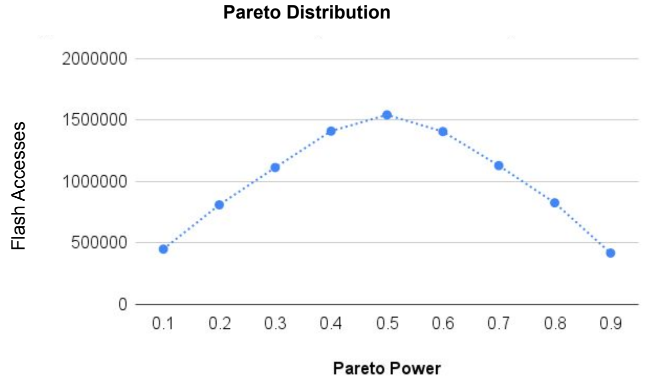


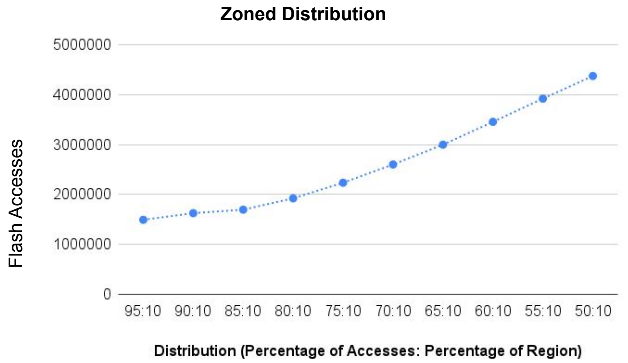
Observation : Cache Hit Rate Decreases as unique data access or hot region size increases, demonstrating effectiveness of the Implemented LRU policy

LRU Cache Validation





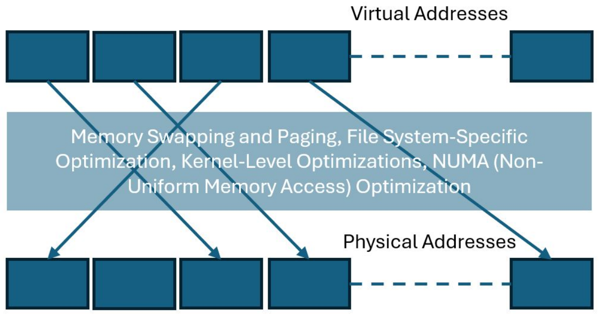




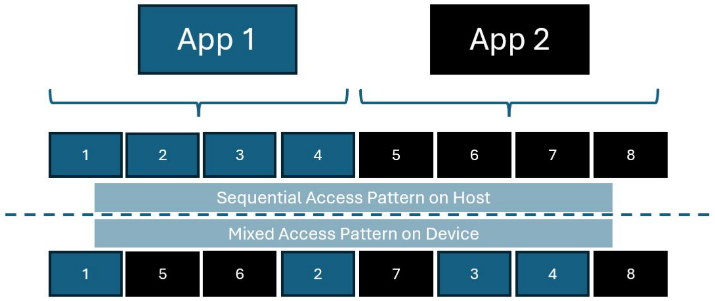
Intent-Based Device Hints

In-Device Data Placement Challenges

- **Address Mapping and optimizations:** Non-linear virtual-to-physical mappings and various system level optimizations create irregular access patterns.
- **Obfuscated Patterns in Mixed Workloads:** Running multiple workloads generates complex, obfuscated access patterns.
- **Lack of Application Context:** Absence of application-level context makes optimizing data placement within the device challenging.


Virtual Addresses
Memory Swapping and Paging, File System-Specific Optimization, Kernel-Level Optimizations, NUMA (Non-Uniform Memory Access) Optimization
Physical Addresses

Non-linear Virtual-Physical Address Mappings


App 1
App 2
1 2 3 4 5 6 7 8
Sequential Access Pattern on Host
Mixed Access Pattern on Device
1 5 6 2 7 3 4 8

Mixed Access Patterns

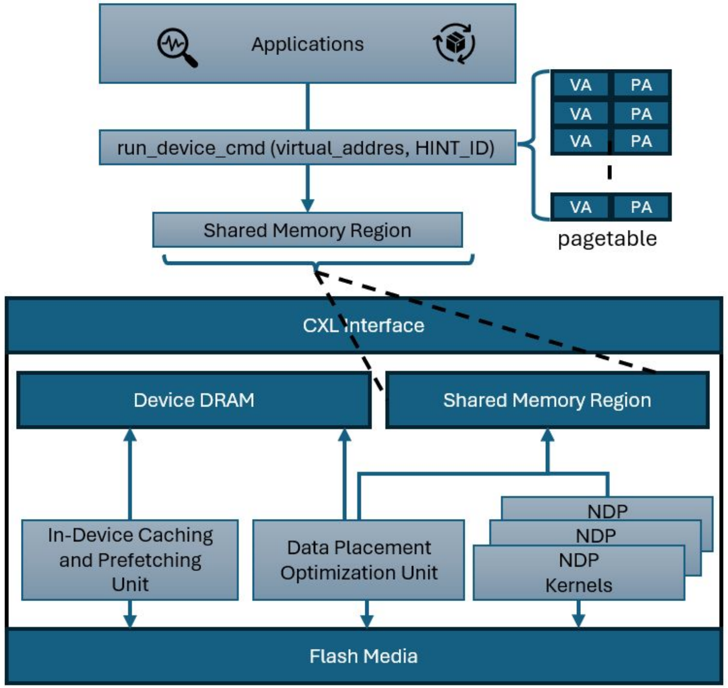
Device-hints with Virtual Addresses

- **Simple API to compose optimizations**
  run_device_cmd(virtual_address, HINT_ID);

- Address non-linear address mappings issue by **working with virtual addresses**

- **Applications** can utilize its context to **aid in-device Data Placement** with hints.

- **Supported Hints**
  - PREFETCH
  - LIFETIME (STREAMING)
  - UNMAP



Hint Framework

CRSS

Evaluation

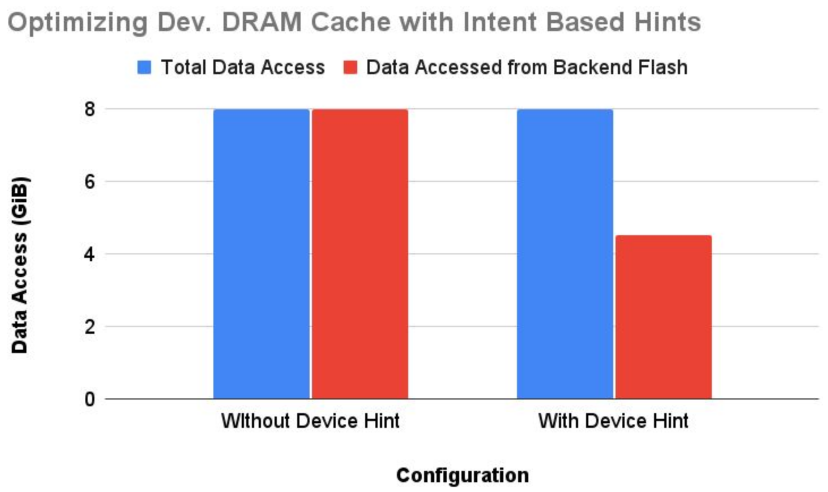
Data Lifetime : Avoid Cache Pollution

- **Experiment Setup:**
  - **Device DRAM:**1GiB
  - **Application**

    1. Hot region: 512 MiB
    2. Streaming Workload: 1 GiB
    3. Total Data Written: 2 * 4 GiB

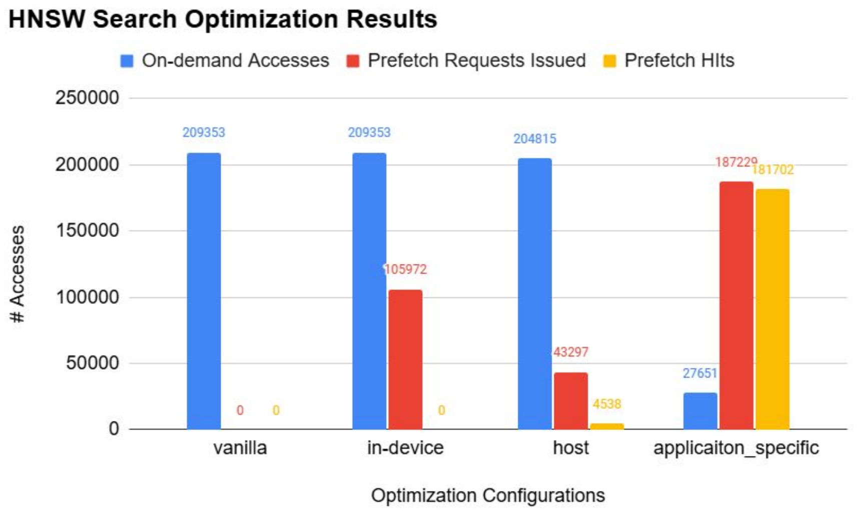
- **Observation**

  The X-axis represents data accessed from the backend Flash; without hints, cache efficiency drops to zero due to cache pollution.



HNSW Optimization: Prefetch Efficiency

- **CXL-SSD Configuration**
  - Device DRAM : 4GB
  - Backend Store : 32 GB
- **Workloads**
  - HNSW Search
  - Vector Dimensions : 96
  - # Vector : 10 million
  - # Queries : 10240
- **Observations**
  - With prefetch hints device utilizes application context to improve prefetch efficiency



Conclusion

- **Implemented extensible QEMU-Based CXL-SSD Emulator with Device Hints**
  - Experiment different **collaborative in-device and host driven data placement** strategies
  - Useful to **prototype CXL-enabled** applications.
  - **Exposes useful statistics** for optimizations

- **Demonstrated Benefits of Device Hints based on virtual addresses**
  - Tackles issues with address mapping
  - Reduce Cache Pollution
  - Improve In-device Cache efficiency

- **Real-World Evaluation**
  - Implemented application specific prefetcher for HNSW search

- **Next Steps**
  - Evaluate More Applications
  - Implement **NDP kernels offloading** HNSW distance calculation to the device
  - Evaluate **optimizations on real device**

Thank You

CRSS

Feedback / Questions?

Contact: Lokesh Jaliminche
email: ljalimin@ucsc.edu

CRSS

Thank you to our sponsors!

arm

cerabyte

intel®

MARVELL™

NUTANIX

NSF