Heterogenous Memory Management

Presented by

Om Rameshwar Gatla

IOWA STATE UNIVERSITY

Intro…

- Om Rameshwar Gatla
  - PhD Student at ISU
  - Advisor: Mai Zheng
  - Research Emphasis: Data storage, Non-volatile memory, File systems
    - File Systems (FS)
      - Linux FS [TOS ‘18, FAST ‘18, HotStorage ‘17]
      - Large scale FS [MSST ‘18]
    - Non Volatile Memory
      - Persistent Memory Devices [FAST ’19 - Poster] (Current research emphasis)

Research Teams

- Western Digital:
  - Cyril Guyot (Director), Adam Manzanares, Filip Blagojevic, Robert Mateescu, Damien Kah, Yongjune Kim, Qing Li
  - Team was part of CTO department
  - Prominent people
    - Christoph Hellwig – Developer and maintainer of XFS
    - Matias Bjorling – Author of Open Channel Spec. 1.2 & 2.0 and maintainer of Linux Kernel, ZNS
    - Zvonimir Bandic – Director for RISC-V project

- Data Storage Lab @ Iowa State University (ISU):
  - Faculty: Mai Zheng
  - 6 Graduate students

Related Concepts

- Page Management
  - Swap space
- IO Interface
  - Block IO
    - Access storage devices (HDD, SSD)
    - Managed by block layer in kernel
    - Minimum IO size = Page size (e.g.: 4KB)
    - Multiple interfaces: SATA, SCSI, NVMe, etc.
  - Memory Mapped IO (MMIO)
    - Byte addressable IO
    - Access to DRAM

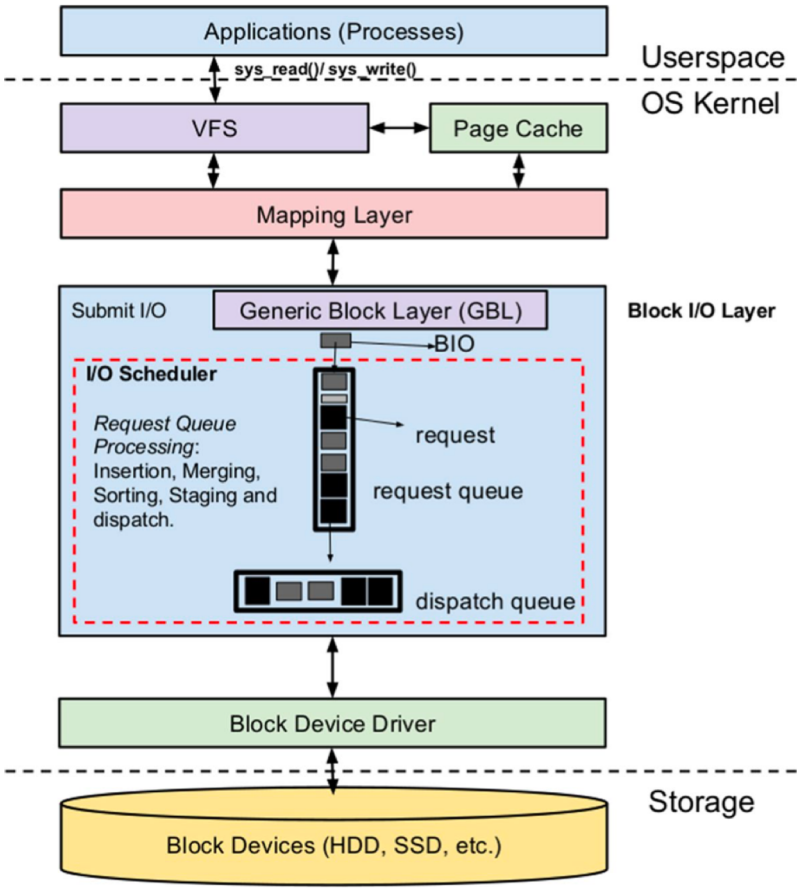


*Image taken from Misra & Somani, Journal of Big Data 2017

Motivation

- Applications demand more memory:
  - E.g.: In-memory database, Graph processing, etc.
  - Store most data in-memory to reduce latency

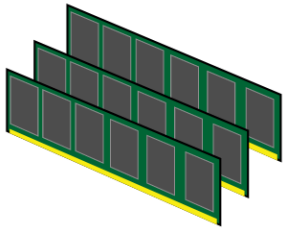
Atlas developed by ISU

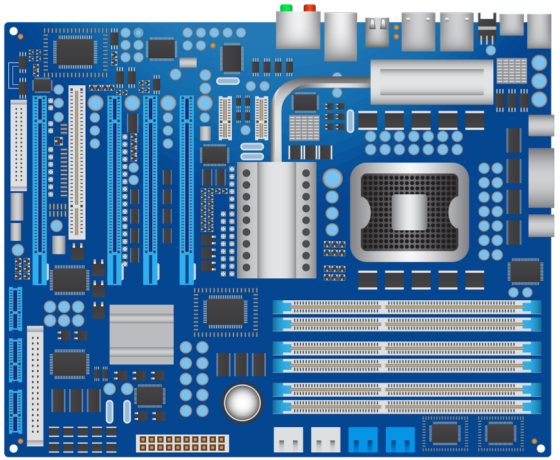
- Static software analysis tool
  - Maintains a graph structure in memory
  - May require 100+ GB memory

IOWA STATE UNIVERSITY

Motivation

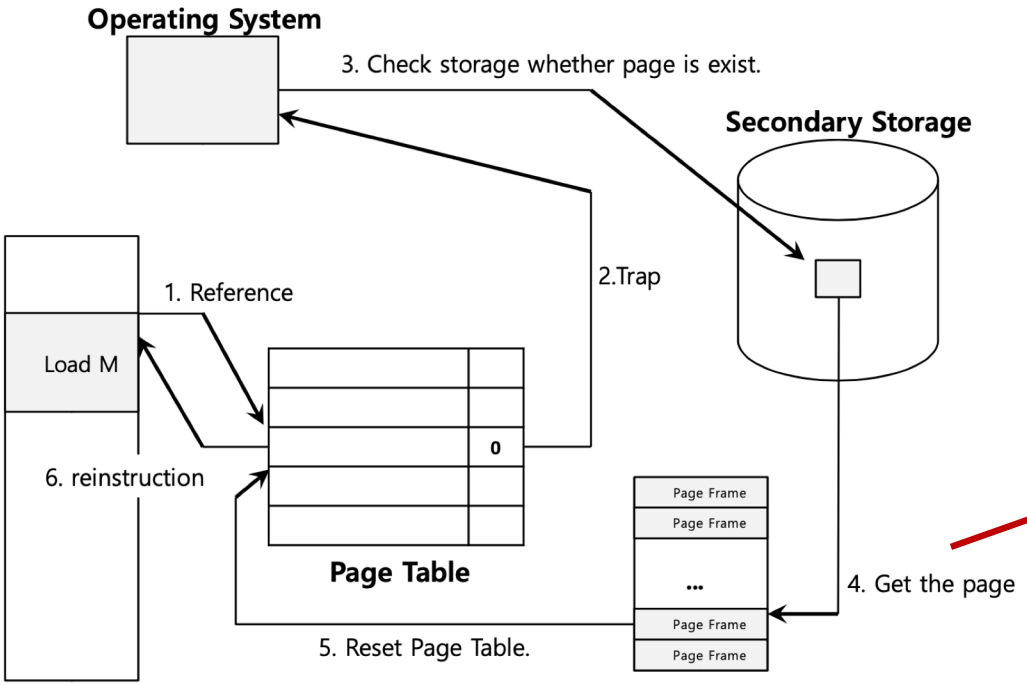
- Existing memory/storage architecture is not optimal for the demand
  - Price/GB for DRAM is high
  - Limited DIMM slots (4) per node
  - Access to swap space on disk is slow





IOWA STATE UNIVERSITY

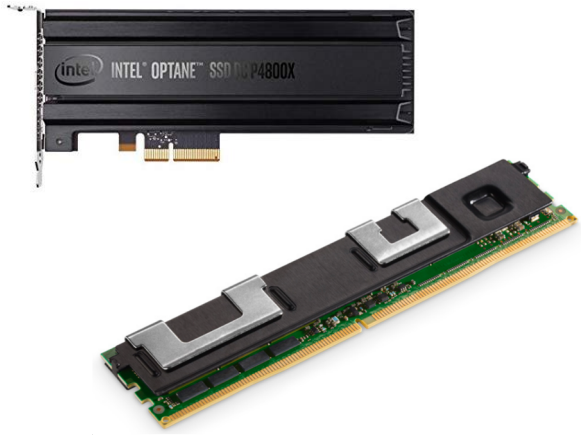
Recap on Swap Space

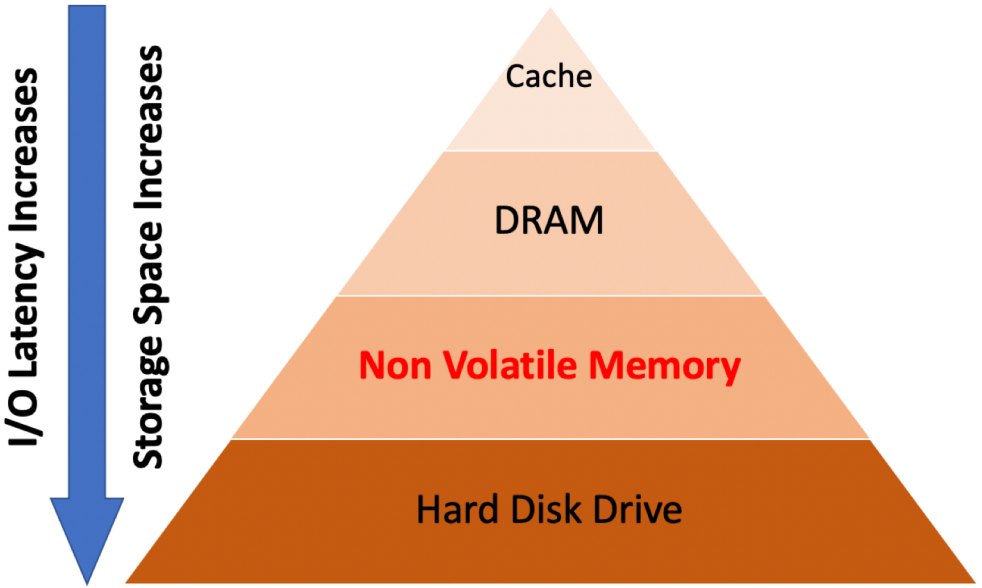


- Happens over Block IO
- Low access time

Motivation

- The line b/w memory & storage is blurring
  - Advent of NVM technologies
    - Flash based devices (NVMe SSD)
    - Intel 3D XPoint
  - Provide low storage latencies





IOWA STATE UNIVERSITY

Our Idea

- Using NVMs as Main Memory
  - One flash-based NVMe SSD can scale to Terabytes
    - much larger compared to one DRAM device
  - Price/GB for flash-based NVMe SSD is much lower
  - Can provide data persistence

IOWA STATE UNIVERSITY

Our Idea

- Using NVMs as Main Memory
  - One flash-based NVMe SSD can scale to Terabytes
    - much larger compared to one DRAM device
  - Price/GB for flash-based NVMe SSD is much lower
  - Can provide data persistence

- Key Challenges
  - How to provide unified address space on heterogenous devices?
  - What applications benefit the most?
  - How to provide transaction support for applications?

IOWA STATE UNIVERSITY

Outline

~~0. Motivation & Introduction~~

1. How to provide unified address space on heterogenous devices
2. What applications benefit the most
3. How to provide transaction support for applications

Providing Unified Address Space

- Problem: DRAM & NVMe SSD have different characteristics
  - Read/Write granularity is different: Byte VS Page (E.g.: 4KB)
  - Access latency is different

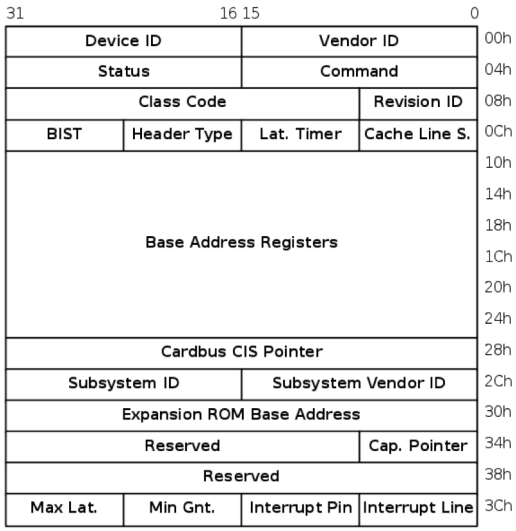
Category	Read Latency (*ns*)	Write Latency (*ns*)	Endurance (*# of writes per bit*)
SRAM	2-3	2-3	∞
DRAM	15	15	10^{18}
STT-RAM	5-30	10-100	10^{15}
PCM	50-70	150-220	10^{8}-10^{12}
Flash	25,000	200,000-500,000	10^{5}
HDD	3,000,000	3,000,000	∞

Table from NV-Tree [FAST '15]

Intel Optane NVMe SSD offers Read latency < 10 µs

Making NVMe SSD Byte-Addressable

- Leverage Base Address Registers (BARs) in PCI Configuration Space
  - BARs enable device to be mapped into host's memory mapped address space
    - Typically for device control/configuration
  - Host allocates address window based on the size in BAR
    - I/O's to this address window are directed to the device
  - Use one BAR to expose SSD space as mmap-ed region



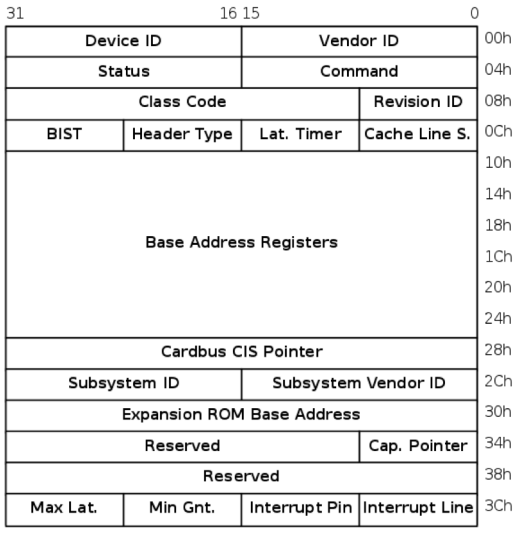
Standard registers of
PCI Configuration Space Header

IOWA STATE UNIVERSITY

Making NVMe SSD Byte-Addressable

- Leverage Base Address Registers (BARs) in PCI Configuration Space
  - BARs enable device to be mapped into host's memory mapped address space
    - Typically for device control/configuration
  - Host allocates address window based on the size in BAR
    - I/O's to this address window are directed to the device
  - Use one BAR to expose SSD space as mmap-ed region

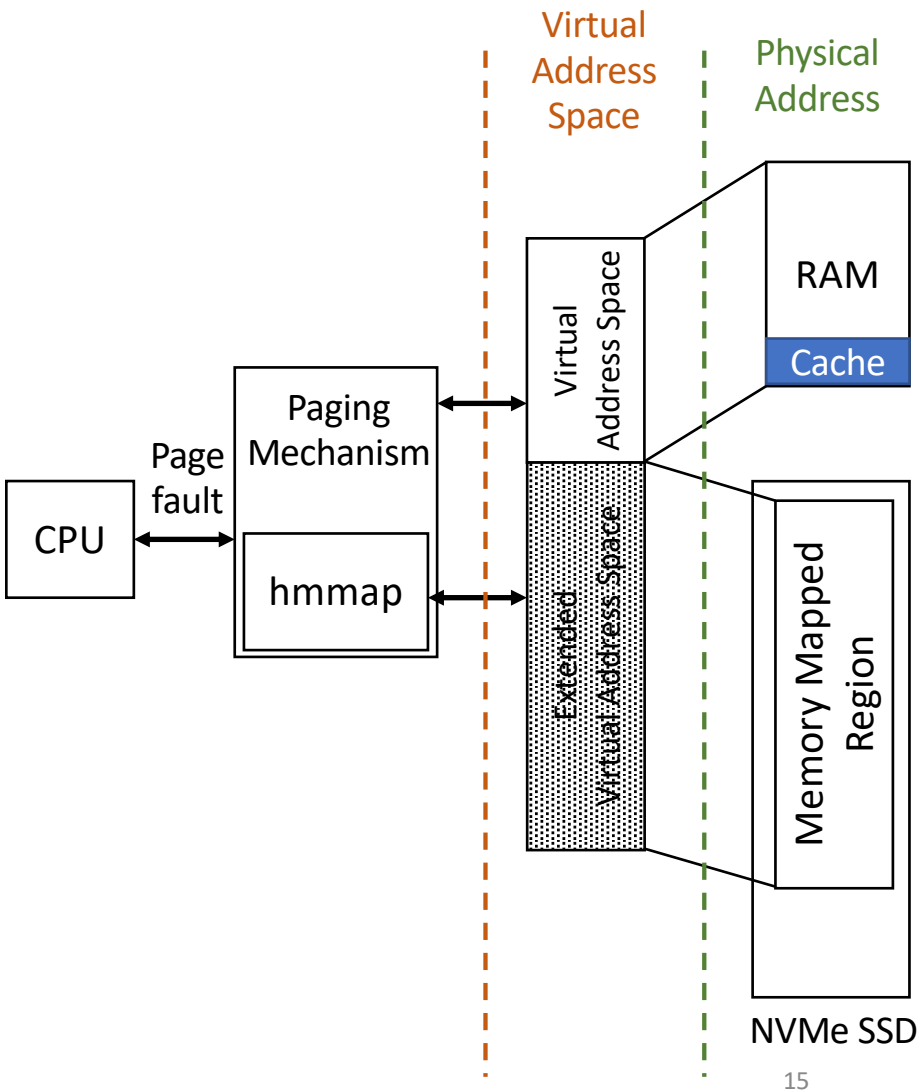
- Existing PCI NVMe SSD device expose a small region (8 KB)
- Team at WD has built a hardware prototype for this purpose
  - Exposes entire device size for MMIO (512 GB)



Standard registers of
PCI Configuration Space Header

Paging Support

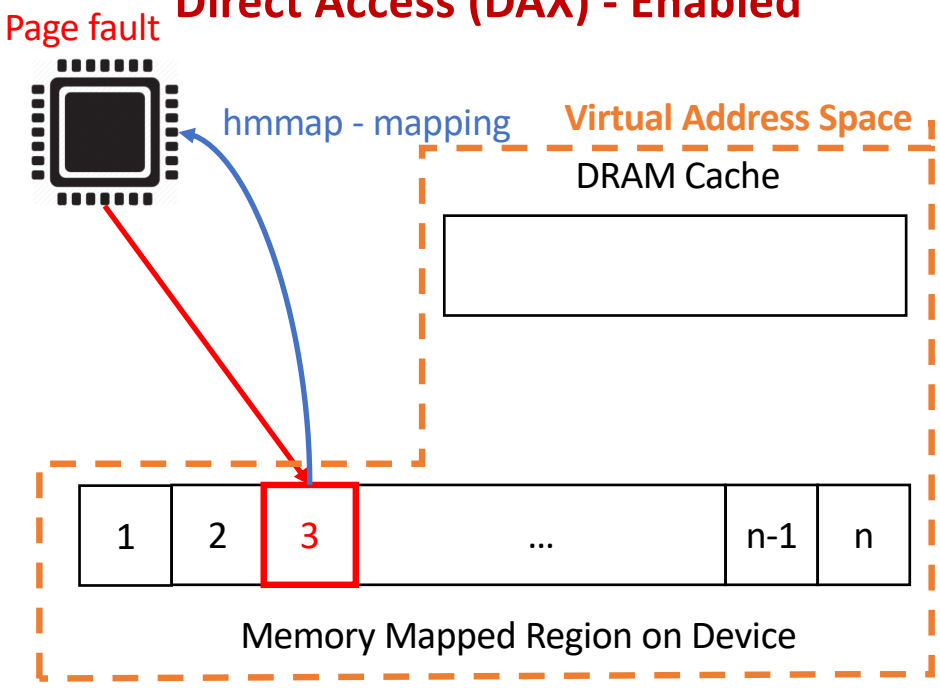
- Heterogenous Memory Mapper (hmmap)
  - Kernel Module
  - Provides mapping between mmap-ed SSD region to Virtual Address Space of application
  - Maintains a cache on the host DRAM
    - To reduce latency



Paging Support (hmmap)

- **Two Modes for load/read:**

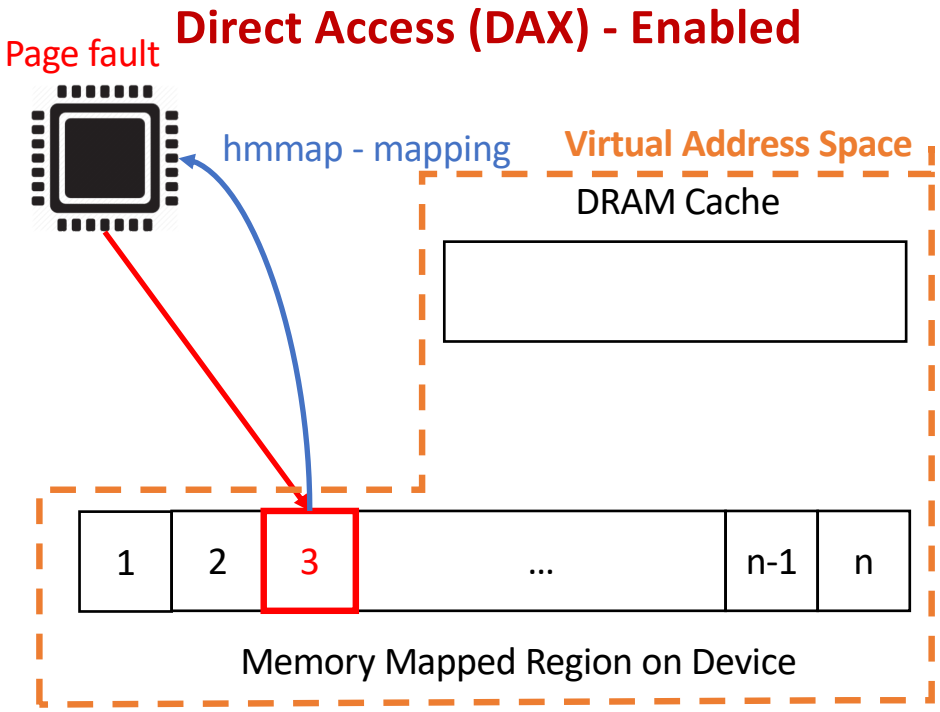
Direct Access (DAX) - Enabled

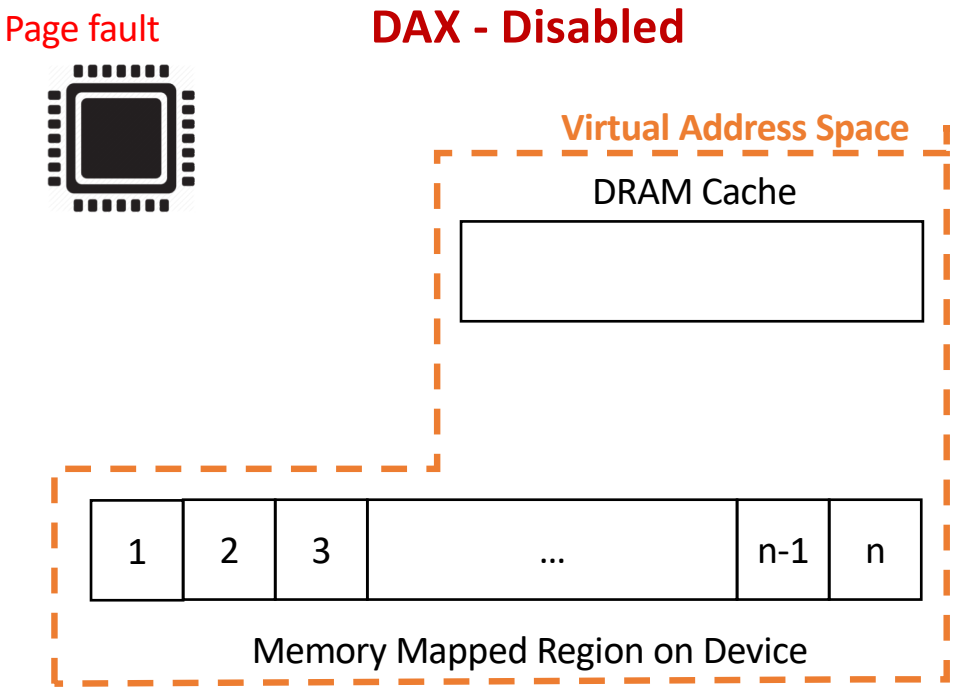


DAX - Disabled

Paging Support (hmmap)

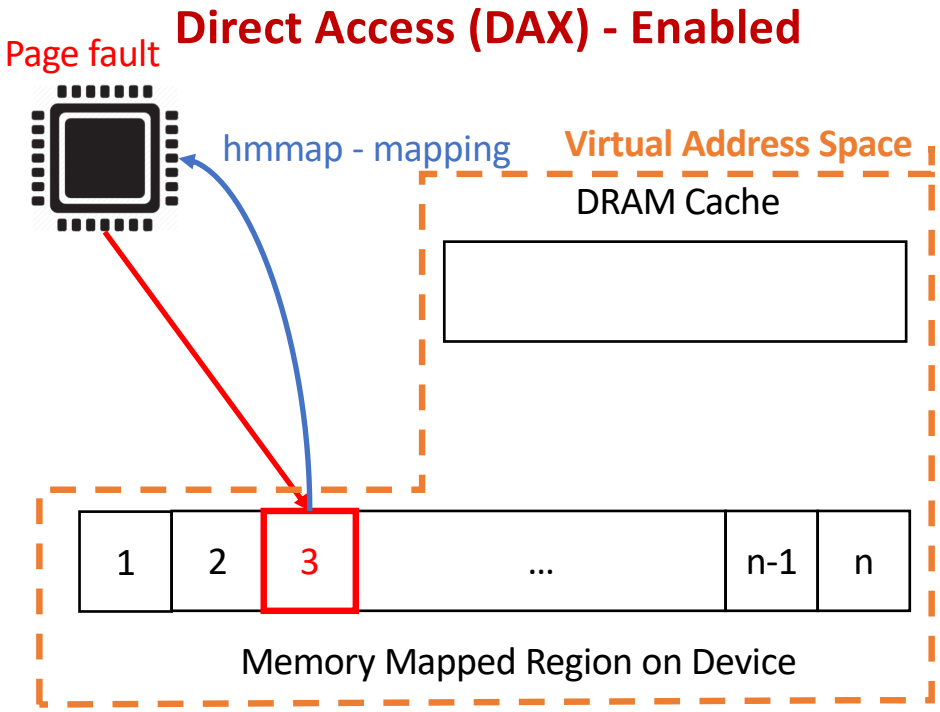
- **Two Modes for load/read:**

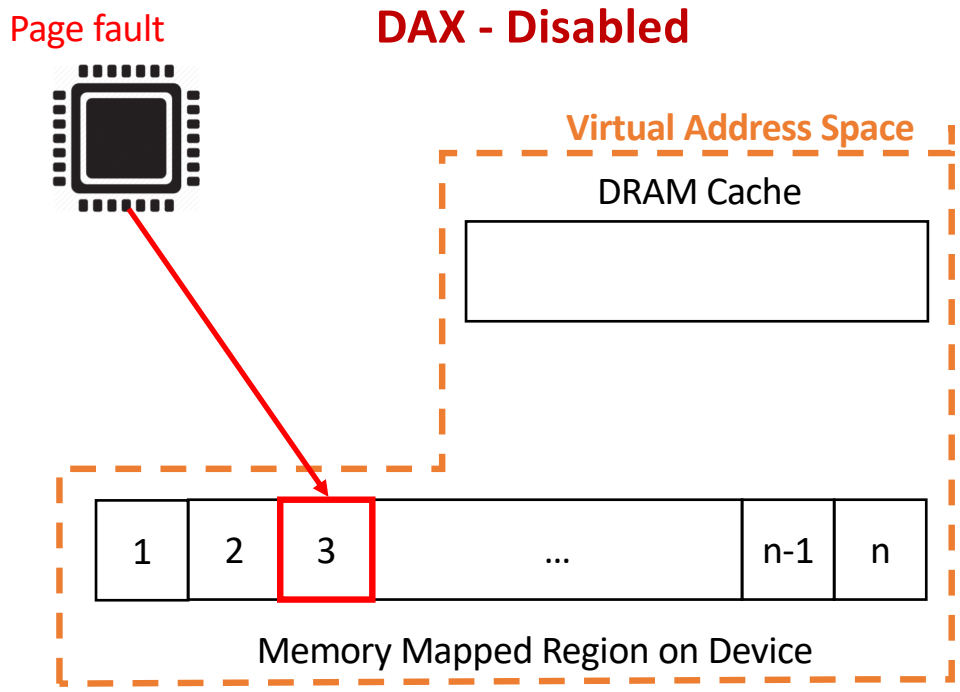




Paging Support (hmmap)

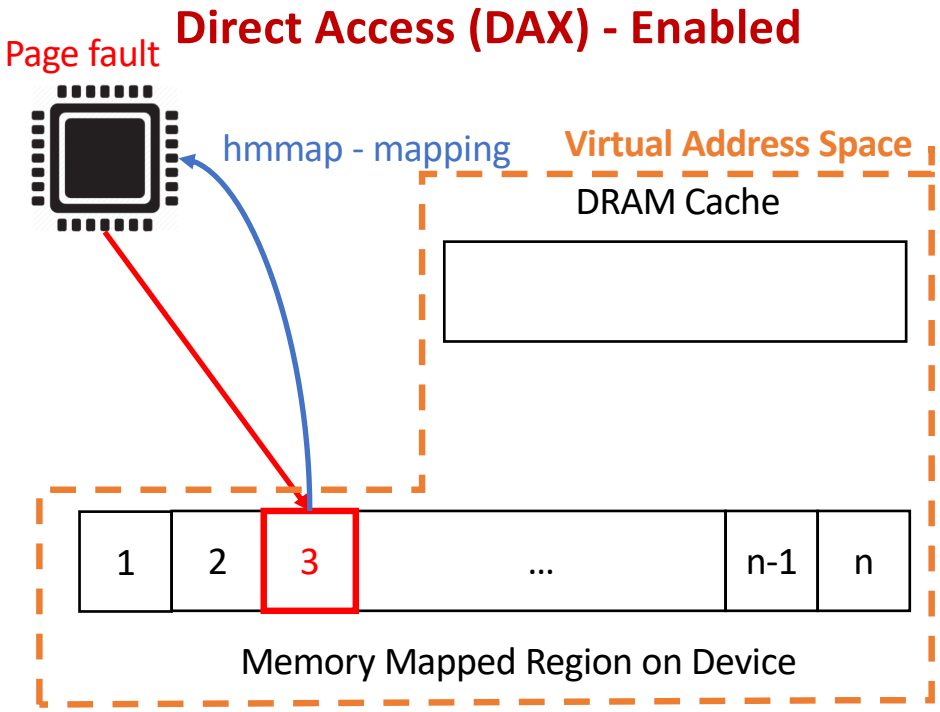
- **Two Modes for load/read:**

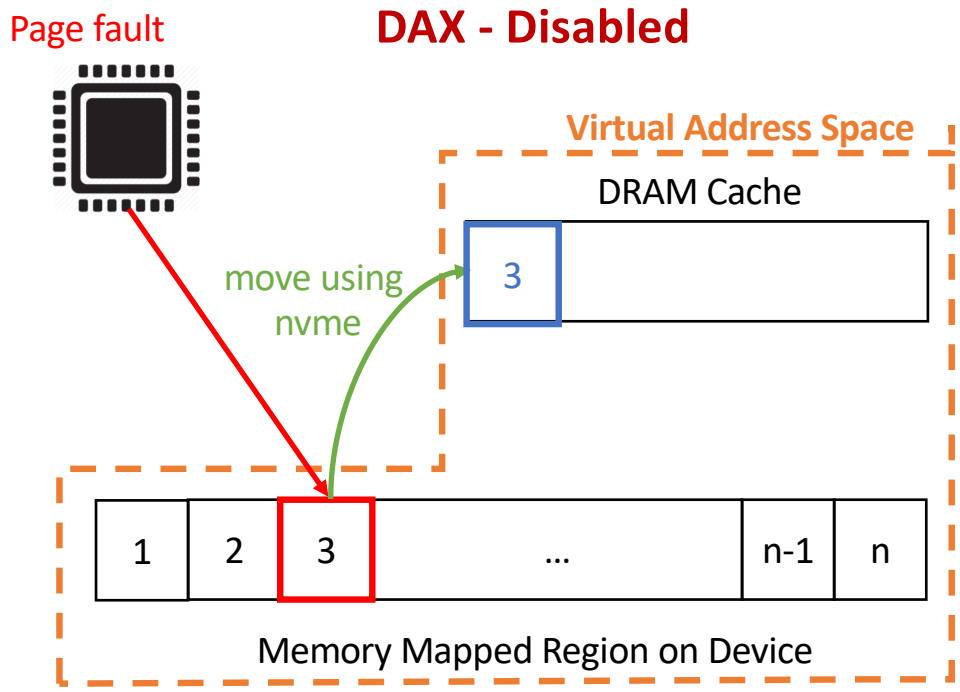




Paging Support (hmmap)

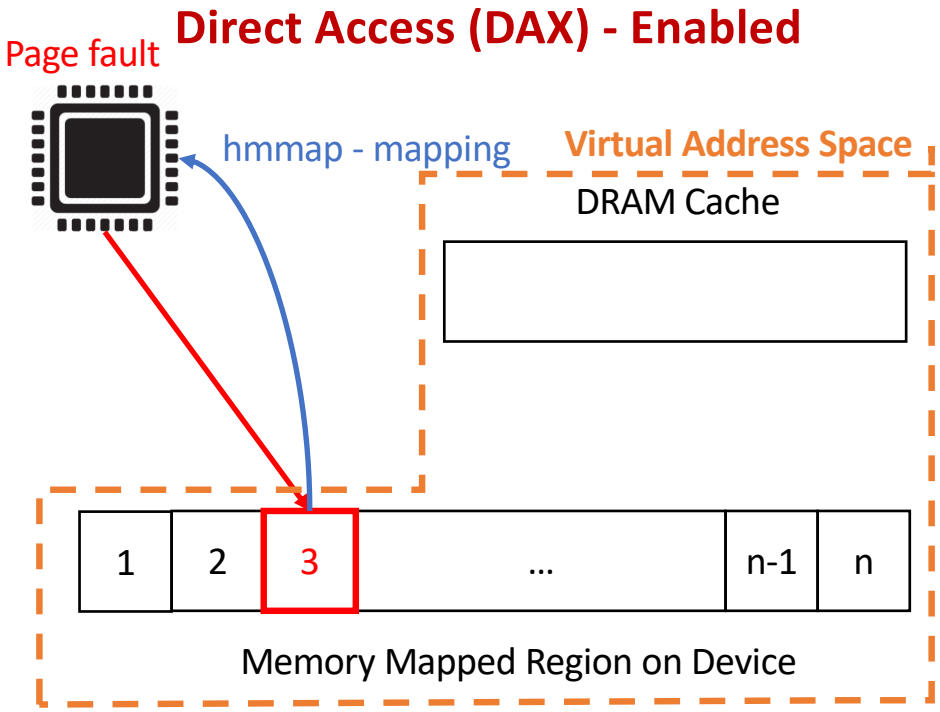
- **Two Modes for load/read:**

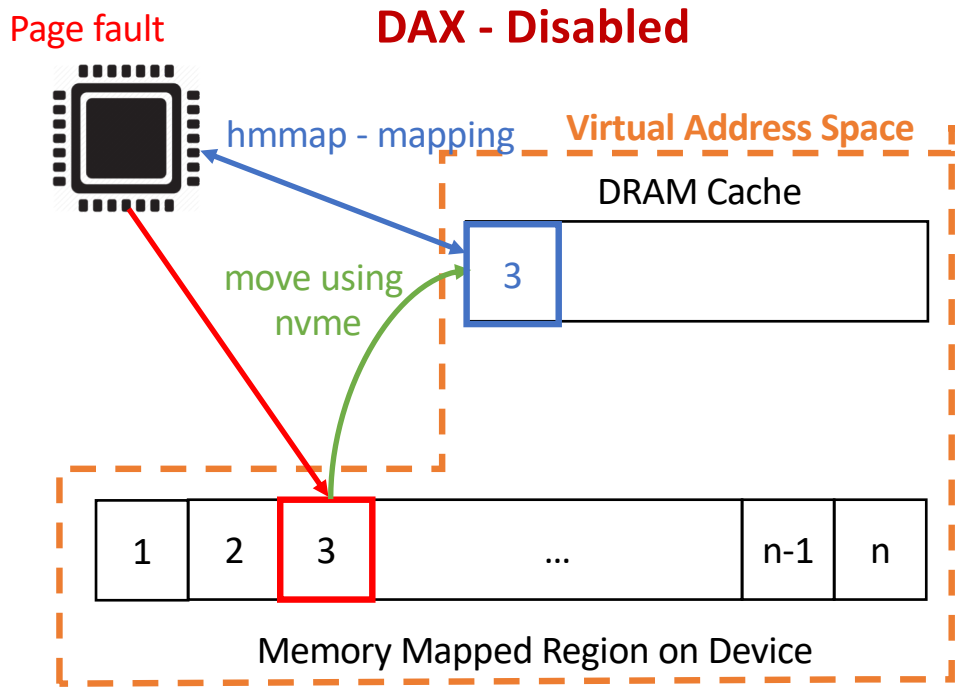




Paging Support (hmmap)

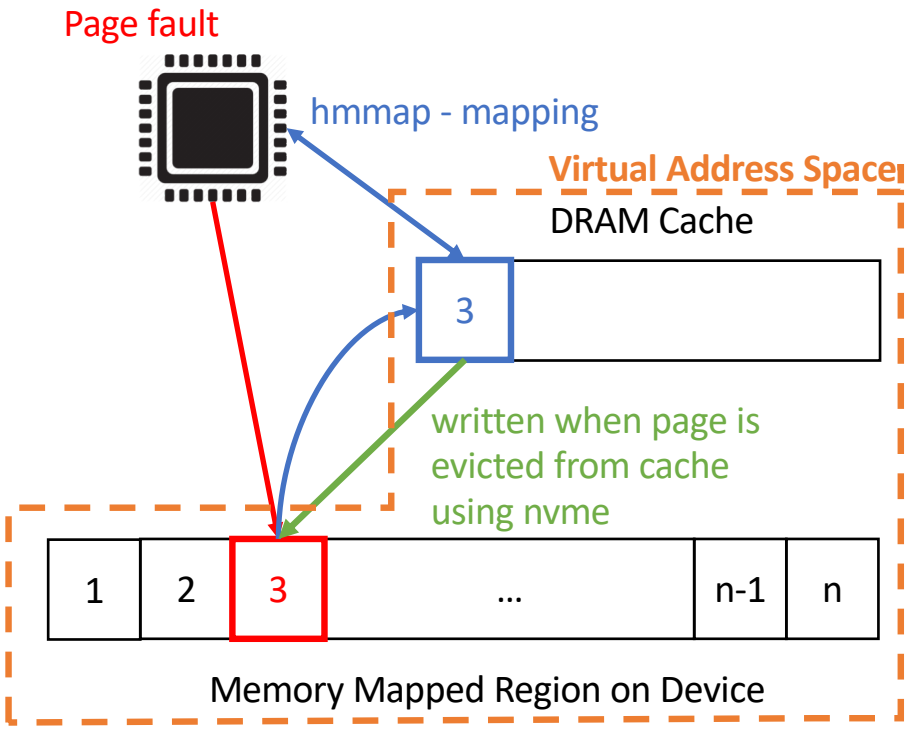
- **Two Modes for load/read:**





Paging Support (hmmap)

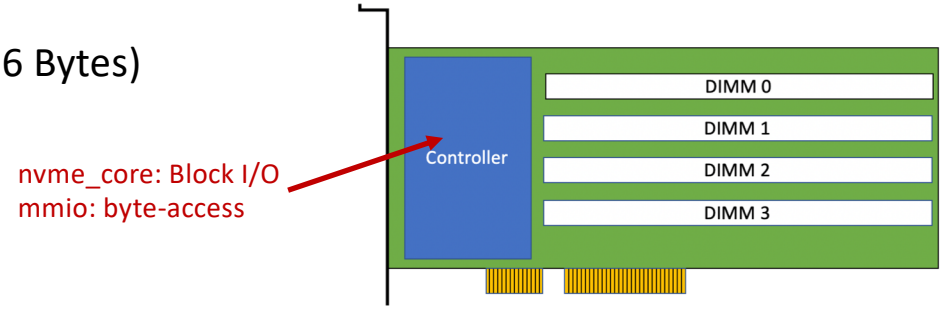
- Only one mode for stores/writes:
  - Small random writes result in low IO operations
    - High write latency
  - hmmap writes new data to DRAM cache
  - Flushes the page when it is evicted from cache
    - Use NVMe block IO interface



IOWA STATE UNIVERSITY

Experiments

- Application
  - **memc3**: in-memory KV store running YCSB workload
    - 95% Reads – Zipfian distribution
    - 6 Million Records (Key: 32 Bytes, Value: 256 Bytes)
    - 6 Million Transactions
- hmmap settings:
  - Mem-mapped region: 2 GB
  - DRAM cache: 8 MB
  - DAX enabled for reads



Hardware Prototype

Runtime on DRAM (in sec)	Runtime on HMMAP (in sec)	Runtime on Block Device (in sec)
70	146	240

IOWA STATE UNIVERSITY

Next Step

- How to reduce paging overhead
  - Possible solution:
    - Identify frequently accessed pages (hot pages) and move them to DRAM cache
  - Methods applied, but failed:
    - **Method 1**: Trace accesses to the memory mapped region using Performance Counters
      - Constant overhead due to Performance Counters
    - **Method 2**: Track Page Table Entries (PTE)
      - Too many pages to track

Method 1: Trace accesses using Performance Counters

- Hardware counters to track hardware events
  - Ex: Cache hit/miss, etc.
- Use perfmon lib to program a custom tracer
  - Use LLC_LOAD_MISS events
  - Tune sampling frequency
- Identify hot pages by counting accesses to each page
- Inform hmmap about hot pages
  - Interface through sysfs file

Method 1: Trace accesses using Performance Counters

- A constant performance overhead
  - Overhead due to many page faults by hmmap
  - Test with cgroups

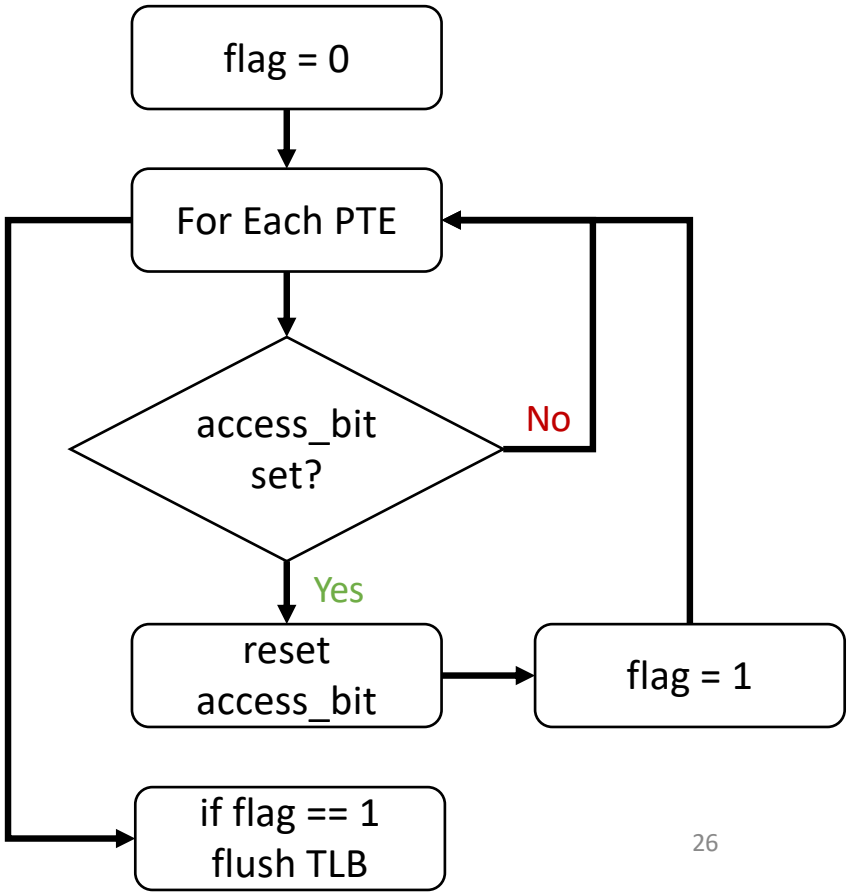
- Performance improves with feedback

Runtime without performance counters (sec)	Runtime with performance counters & without feedback (sec)	Runtime with performance counters & with feedback (sec)
146	168	155

Method 2: Track Page Table Entries (PTE)

- Kernel thread to walk page table
- Tunable sample & feedback time

- Drawbacks:
  - Many pages to track
  - Results in performance degradation



IOWA STATE UNIVERSITY

Outline

~~0. Motivation & Introduction~~

~~1. How to provide unified address space on Heterogenous devices~~

2. What applications benefit the most

3. How to provide transaction support for applications

IOWA STATE UNIVERSITY

Survey on Popular Applications

- Survey 48 research papers from top 7 systems/architecture conferences from 2009 – 2019
- Identify most commonly used applications that benefit from NVM

Conference	Related Papers
FAST	13
OSDI/SOSP	2
ASPLOS	14
USENIX ATC	10
MICRO	3
ISCA	3
HPCA	3
Total	48

IOWA STATE UNIVERSITY

Survey on Popular Applications

- Database workloads are predominant

Workload Type	Applications	Paper Count
NoSQL Database	Memcached, Redis, Cassandra, Aerospike, MongoDB	19
SQL Database	MySQL, SQLite	9
Graph Processing	PowerGraph, Neo4J	3

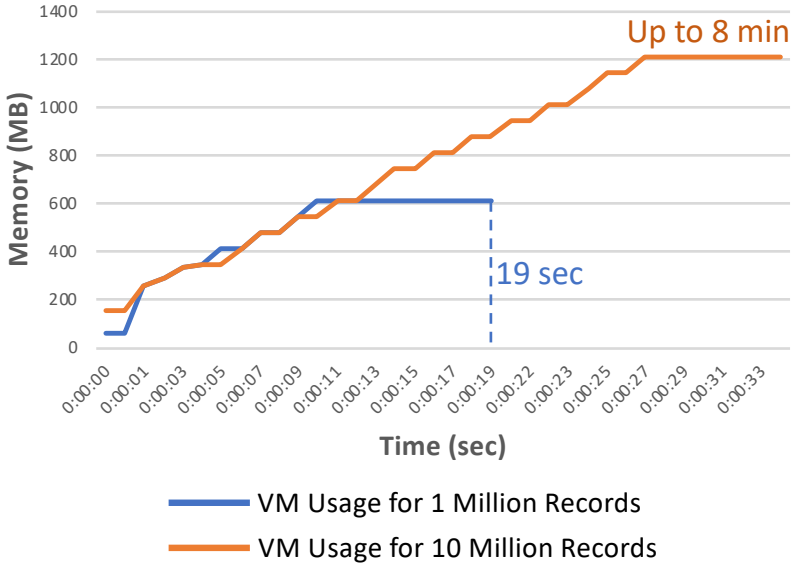
Tracking Memory Usage of Applications

- Understand memory usage characteristics of popular applications
  - Memory profiling
    - Provide aggregate information
    - Custom script based on **PS**
  - Memory tracing
    - Provide detailed access pattern
    - Intel **Pin** (Dynamic Analysis)
    - **Spindle** (Static Analysis) [USENIX ATC'18]
      - With customization
  - Cache Analysis
    - Study Impact of cache on memory accesses
    - Cache simulator - **DynamoRIO**
    - Performance Counters – **Perf tool**

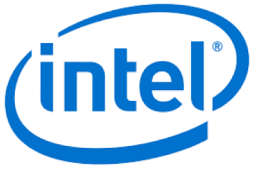
IOWA STATE UNIVERSITY

Tracking Memory Usage of Application

- Memory Profiling
  - Custom script to profile memory usage using **PS**
  - Use **Memcached** application
    - Memory restricted to 1GB
  - Run YCSB workload:
    - 1 & 10 Million records
    - Each record size 280 Bytes
  - Runtime increases significantly when given memory is insufficient



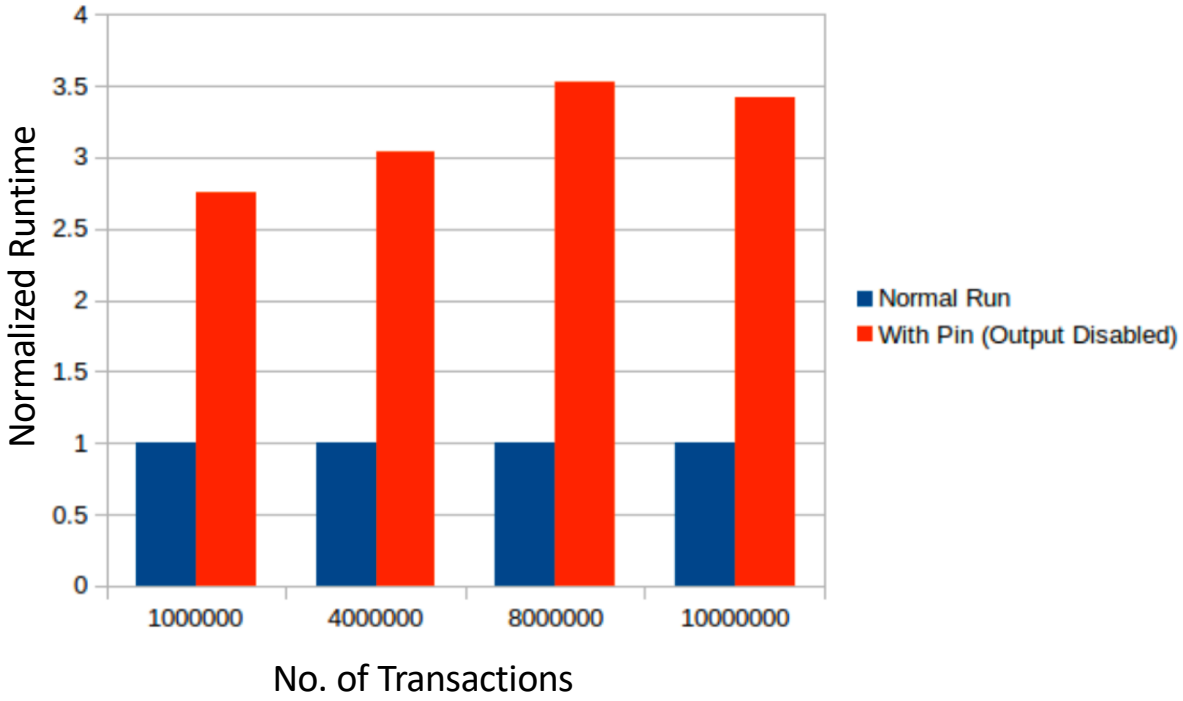
Tracking Memory Usage of Applications



- Memory Tracing
  - Intel Pin
    - Dynamic instrumentation tool
    - Customize tool to perform memory tracing
    - Results in high runtime overhead
  - Spindle [USENIX ATC'18]
    - A LLVM based static analysis framework
      - Uses static analysis data to perform memory access monitoring
    - Authors' code does not work on latest system
      - Re-implement based on the paper
        - Provide support on LLVM 8.0
        - Perform tracing across multiple functions by building call graph
        - Identify memory accesses by local variables

Tracking Memory Usage of Applications

Preliminary results on Pin:



IOWA STATE UNIVERSITY

Tracking Memory Usage of Applications

Preliminary results on spindle:

Source code:

```
void bubbleSort(int *arr, int n) {
  int i, j;
  for (i = 0; i < n - 1; i++) {
    for (j = 0; j < n - i - 1; j++) {
      if (arr[j] > arr[j + 1]) {
        swap(&arr[j], &arr[j + 1]);
      }
    }
  }
}
```

Static Trace:

```
function bubbleSort 2
loop ID: 0
Indvars: 0
loop ID: 1
Indvars: 0
block id: 6, loop_id:  1, type: 11
 successor: 3
dependence
```

Dynamic Trace:

```
// value Id, function Id and address.
ov: 0, 2, 0x7ffe892184d8
ov: 1, 2, 0x7ffe892184e4
ov: 2, 2, 0x7ffe892184e8
ov: 3, 2, 0x7ffe892184ec
ov: 4, 2, 0x00e808b0
ov: 5, 2, 0x00e808b4
path: 6, 1
```

Tracking Memory Usage of Applications

Preliminary results on spindle:

Source code:

```
void bubbleSort(int *arr, int n) {
  int i, j;
  for (i = 0; i < n - 1; i++) {
    for (j = 0; j < n - i - 1; j++) {
      if (arr[j] > arr[j + 1]) {
        swap(&arr[j], &arr[j + 1]);
      }
    }
  }
}
```

Static Trace:

```
function bubbleSort 2
loop ID: 0
Indvars: 0
loop ID: 1
Indvars: 0
block id: 6, loop_id:  1, type: 11
 successor: 3
dependence
```

Dynamic Trace:

```
// value Id, function Id and address.
ov: 0, 2, 0x7ffe892184d8
ov: 1, 2, 0x7ffe892184e4
ov: 2, 2, 0x7ffe892184e8
ov: 3, 2, 0x7ffe892184ec
ov: 4, 2, 0x00e808b0
ov: 5, 2, 0x00e808b4
path: 6, 1
```

local variables

global variables (array)

Tracking Memory Usage of Applications

- Impact of Cache on Memory Access
  - Current tracing tools do not consider the effect of cache
  - A subset of memory references result in memory load/store
    - A great portion are served by the cache
    - L3 Cache load/store events
  - 2 methods to quantify the impact
    - Method 1: Use DynamoRIO cache simulator to identify load/stores
    - Method 2: Use “Perf tool” to trace L3 Cache Miss events

	DynamoRIO	Perf tool
% of Memory Accesses resulting in L3 Cache Miss event	8 - 10	35 - 40

Tracking Memory Usage of Applications

- Impact of Cache on Memory Access
  - Current tracing tools do not consider the effect of cache
  - A subset of memory references result in memory load/store
    - A great portion are served by the cache
    - L3 Cache load/store events
  - 2 methods to quantify the impact
    - Method 1: Use DynamoRIO cache simulator to identify load/stores
    - Method 2: Use “Perf tool” to trace L3 Cache Miss events

	DynamoRIO	Perf tool
% of Memory Accesses resulting in L3 Cache Miss event	8 - 10	35 - 40

cache simulator only runs one program

Tracking Memory Usage of Applications

- Impact of Cache on Memory Access
  - Current tracing tools do not consider the effect of cache
  - A subset of memory references result in memory load/store
    - A great portion are served by the cache
    - L3 Cache load/store events
  - 2 methods to quantify the impact
    - Method 1: Use DynamoRIO cache simulator to identify load/stores
    - Method 2: Use “Perf tool” to trace L3 Cache Miss events

System wide

	DynamoRIO	Perf tool
% of Memory Accesses resulting in L3 Cache Miss event	8 - 10	35 - 40

IOWA STATE UNIVERSITY

Outline

~~0. Motivation & Introduction~~

~~1. How to provide unified address space on Heterogenous devices~~

~~2. What applications benefit the most~~

3. How to provide transaction support for applications

IOWA STATE UNIVERSITY

Survey on Popular Applications

Previously…

- Database workloads are predominant
  - require ordering, durability & atomicity support

Workload Type	Applications	Paper Count
NoSQL Database	Memcached, Redis, Cassandra, Aerospike, MongoDB	19
SQL Database	MySQL, SQLite	9
Graph Processing	PowerGraph, Neo4J	3

Transaction support for applications

- Explore API support
  - 3 possible ways of accessing device:
    - Native Persistence: using CPU instructions such as clflush, sfence
    - Library Persistence: NV-Heaps, Mnemosyne, etc.
    - Filesystem Persistence: DAX Based Filesystems (EXT4-DAX, XFS-DAX)
  - Implementation on hmmap
    - E.g.: support for failure-atomic msync()

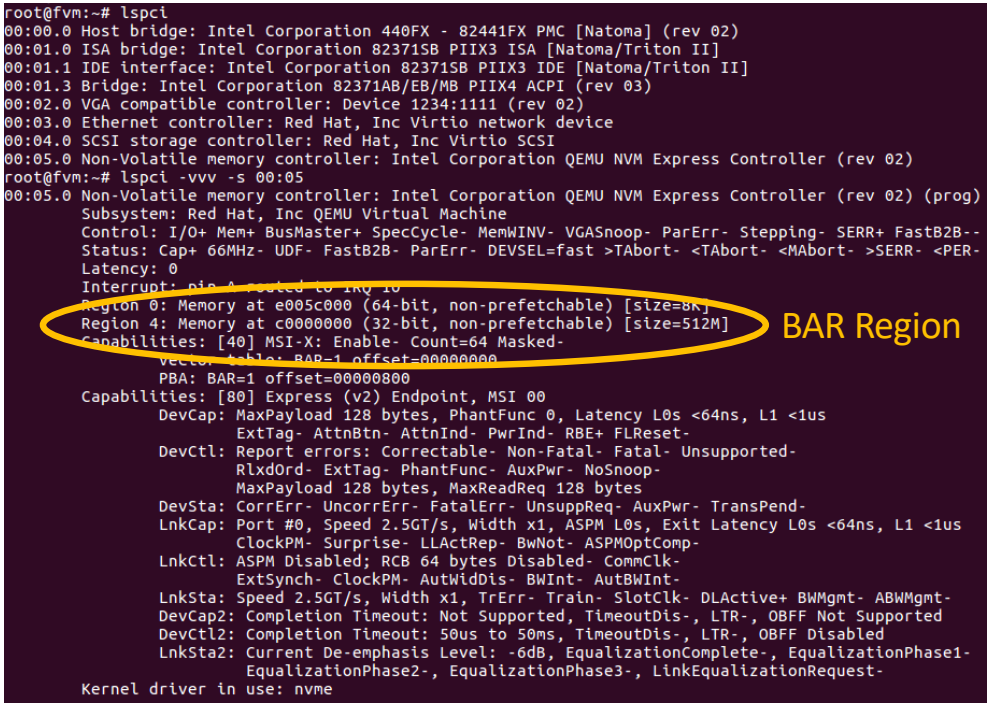
NV-Heaps [ASPLOS '11]
Mnemosyne [ASPLOS '11]
NV-Tree [FAST '15]
WHISPER [ASPLOS '17]

Transaction support for applications

- Explore Hardware support
  - Possible ways:
    - Implement Gates | 2B-SSD [ISCA '18]
      - Avoid conflicts between Block IO and MMIO
    - Write-verify Read operation | 2B-SSD [ISCA '18], FlatFlash [ASPLOS '19]
      - Enforce ordering of writes when CPU buffer is flushed
    - Maintain capacitors on board | 2B-SSD [ISCA '18]
      - Protection from power failure
    - Many more to explore…

Transaction support for applications

- **Explore co-design of hardware/software for durability**
  - Use QEMU virtual machine for easy prototyping
    - QEMU is a generic, open source machine emulator
    - Emulate multiple devices:
      - SATA HD, PCIe device, NVMe devices
    - Modify NVMe controller code
      - Create a BAR region to enable MMIO



IOWA STATE UNIVERSITY

Outline

~~0. Motivation & Introduction~~

~~1. How to provide unified address space on Heterogenous devices~~

~~2. What applications benefit the most~~

~~3. How to provide transaction support for applications~~

THANK YOU!

Questions?