Moneta:
A High-Performance Storage Architecture for Next-generation, Non-volatile Memories

Adrian M. Caulfield
Arup De, Joel Coburn, Todor I. Mollov, Rajesh K. Gupta, Steven Swanson

Non-Volatile Systems Laboratory,
Department of Computer Science and Engineering
University of California, San Diego

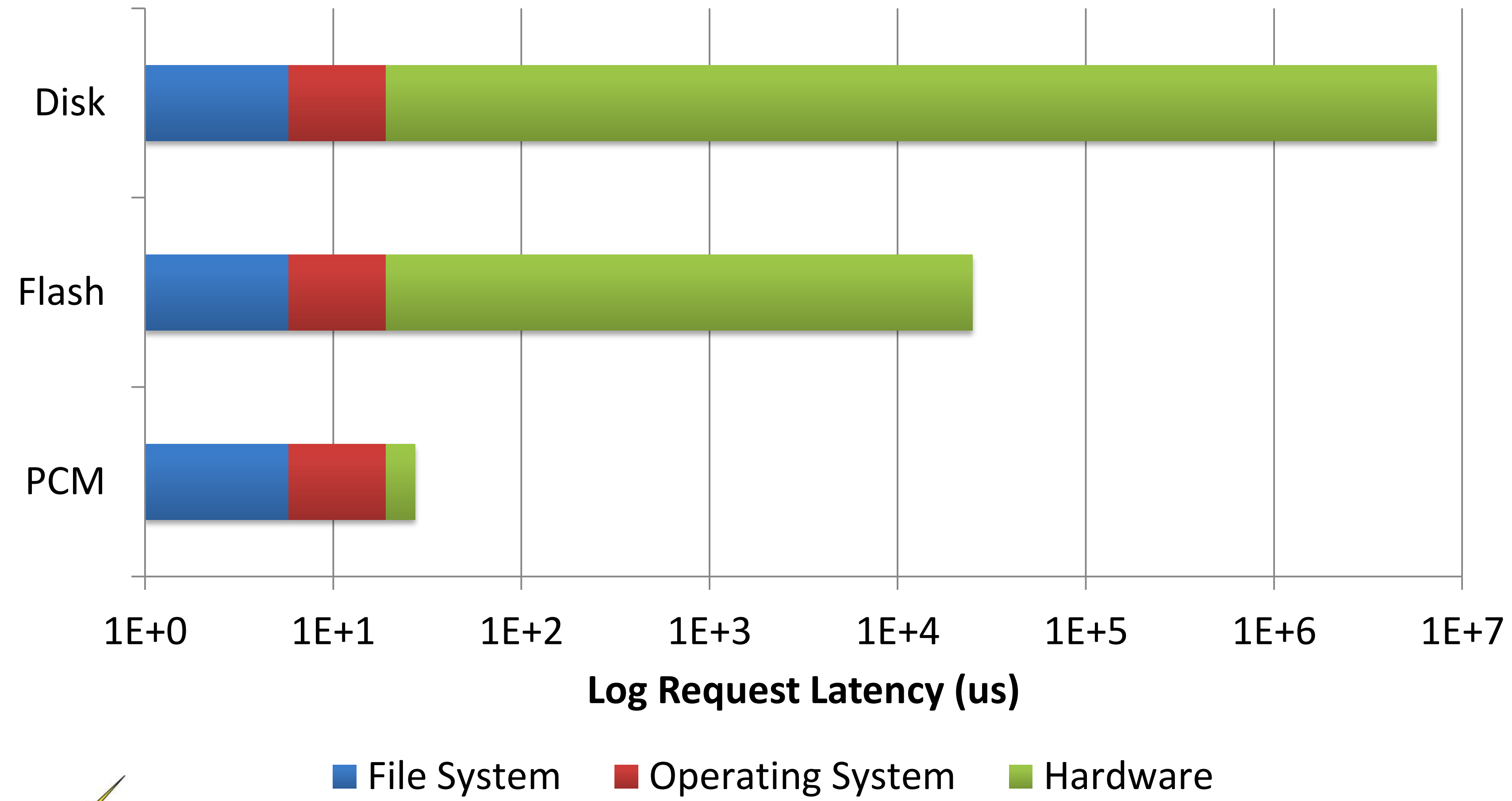
NVSL Non-volatile Systems Laboratory UCSDCSE Computer Science and Engineering

The Future of Storage

Hard Drives	PCIe-Flash 2007	PCIe-NVM 2013?	
			
Lat.: 7.1ms	68us	12us	
BW: 2.6MB/s	250MB/s	1.7GB/s	
1x	104x	591x	= 2.89x/yr
1x	96x	669x	= 2.95x/yr

*Random 4KB Reads from user space

Software Latency Costs



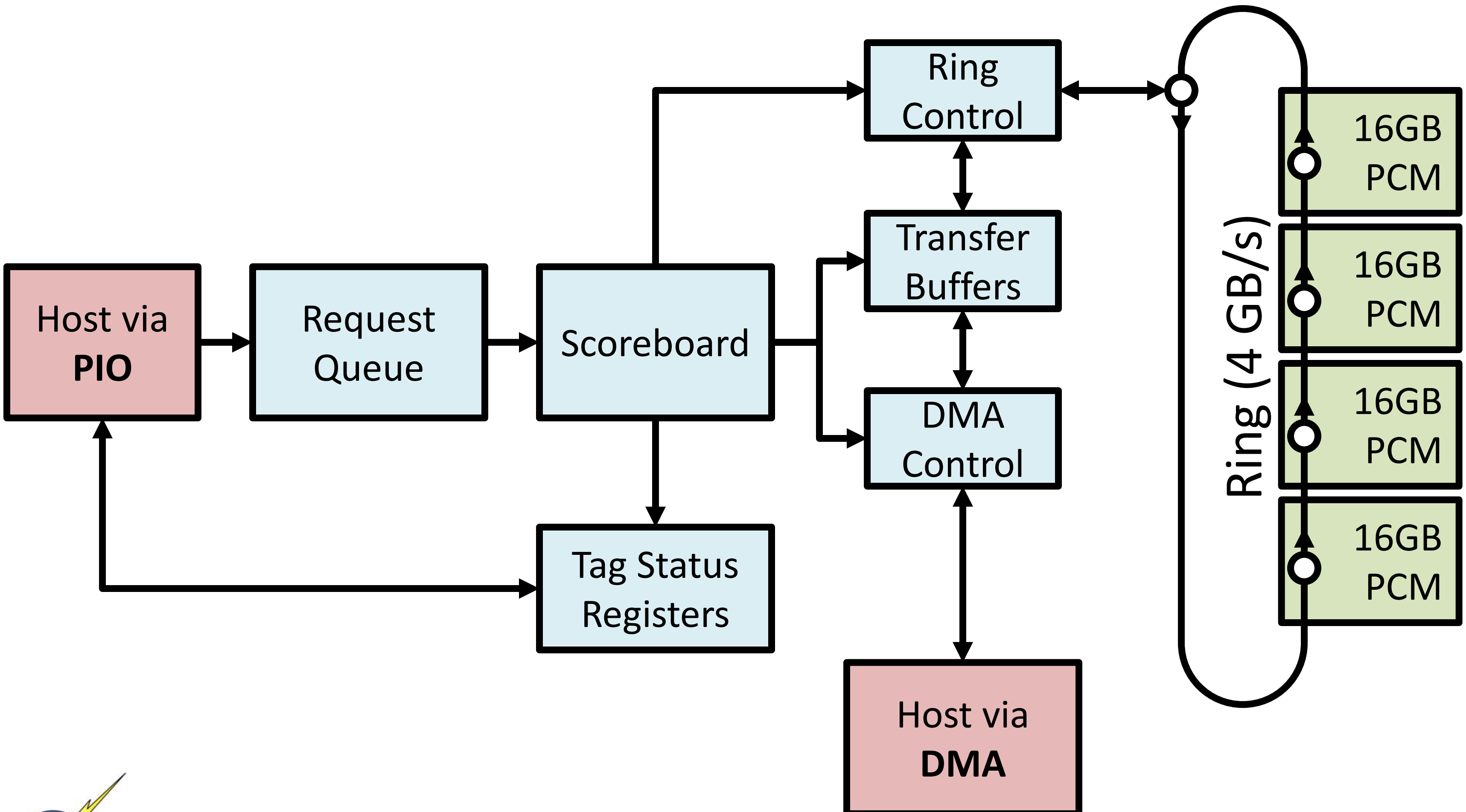
Architecting a High Performance SSD

- Hardware architecture and software layers limit performance
- HW/SW interface critical to good performance
- Careful co-design provides significant benefits
  - Increased bandwidth
  - Decreased latencies
  - Increased concurrency

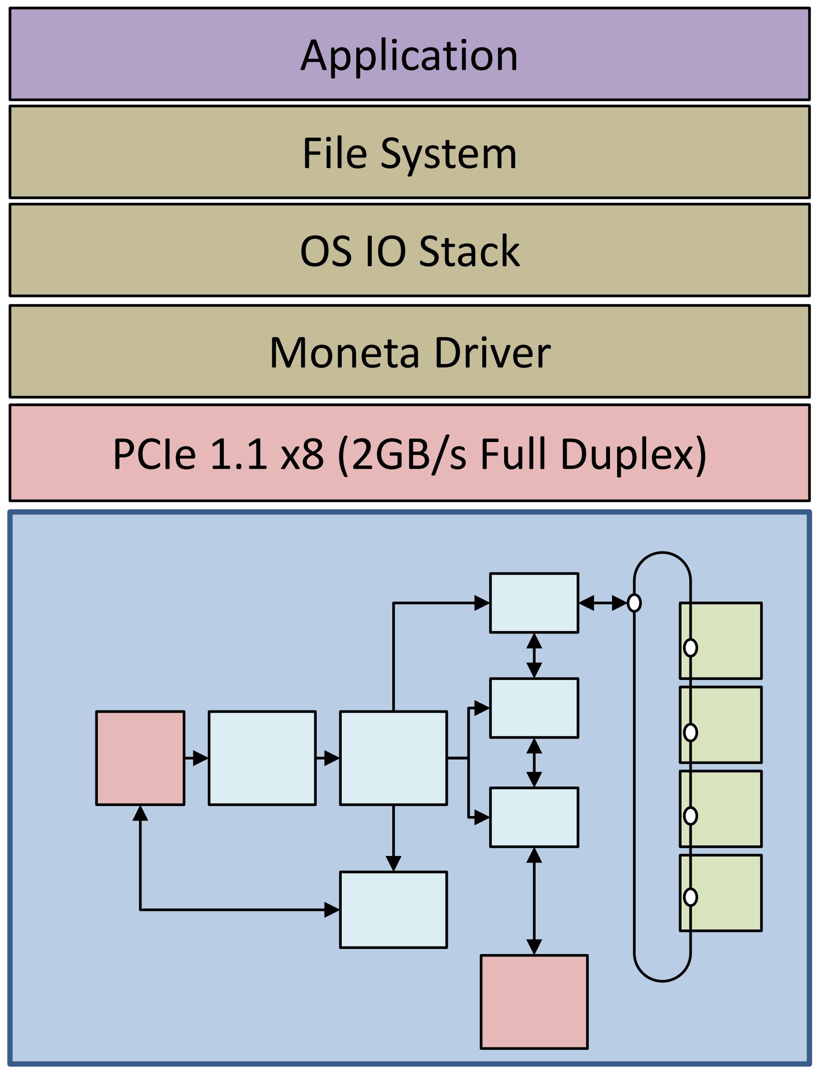
Overview

- Motivation
- **Moneta Architecture**
- Optimizing Moneta
- Performance Analysis
- Conclusion

Moneta Architecture



Moneta Architecture Overview



Advanced Memory Technology Characteristics

- Work with any fast NVM:
  - DRAM-like speed
  - DRAM-like interface
- Phase Change Memory
  - Coming soon
  - Simple wear leveling: Start Gap [Micro 2009]

Moneta: Modeling Advanced NVMs

- Built on RAMP's BEE3 board
- PCIe 1.1 x8 host connection
- 250MHz design
- DDR2 DRAM emulates NVMs
  - Adjust timings to match PCM
  - RAS-CAS Delay for reads
  - Precharge latency for writes

	Read	Write
Projected Latency	48 ns	150 ns

Latency projections from [B.C. Lee, ISCA'09]

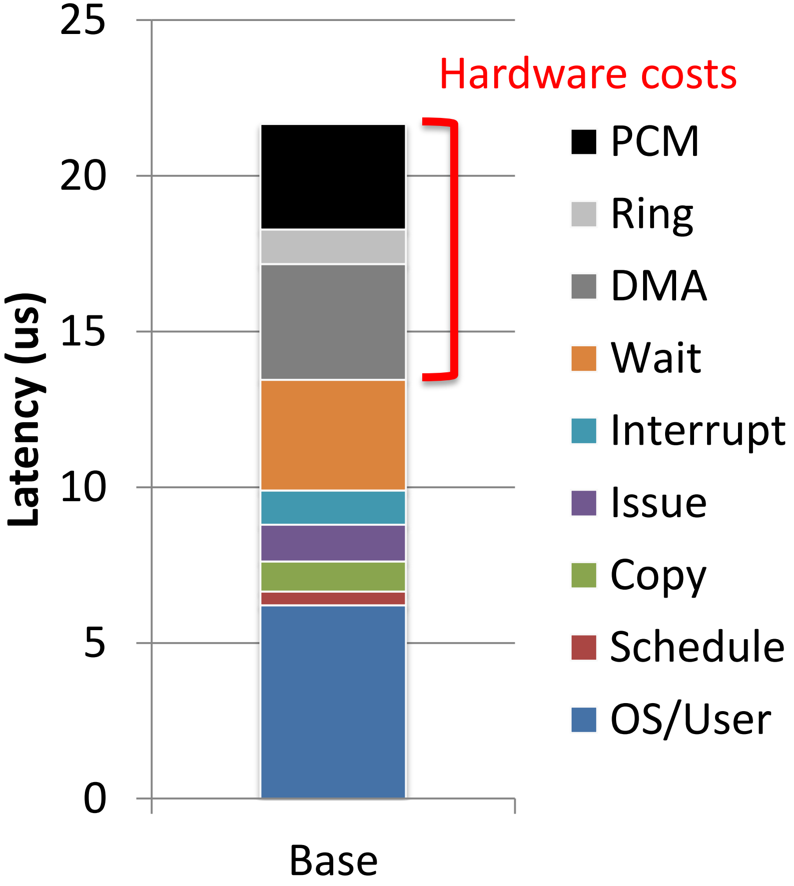
Overview

- Motivation
- Moneta Architecture
- **Optimizing Moneta**
  - **Interface & Software**
  - Microarchitecture
- Performance Analysis
- Conclusion

Optimized Software is Critical

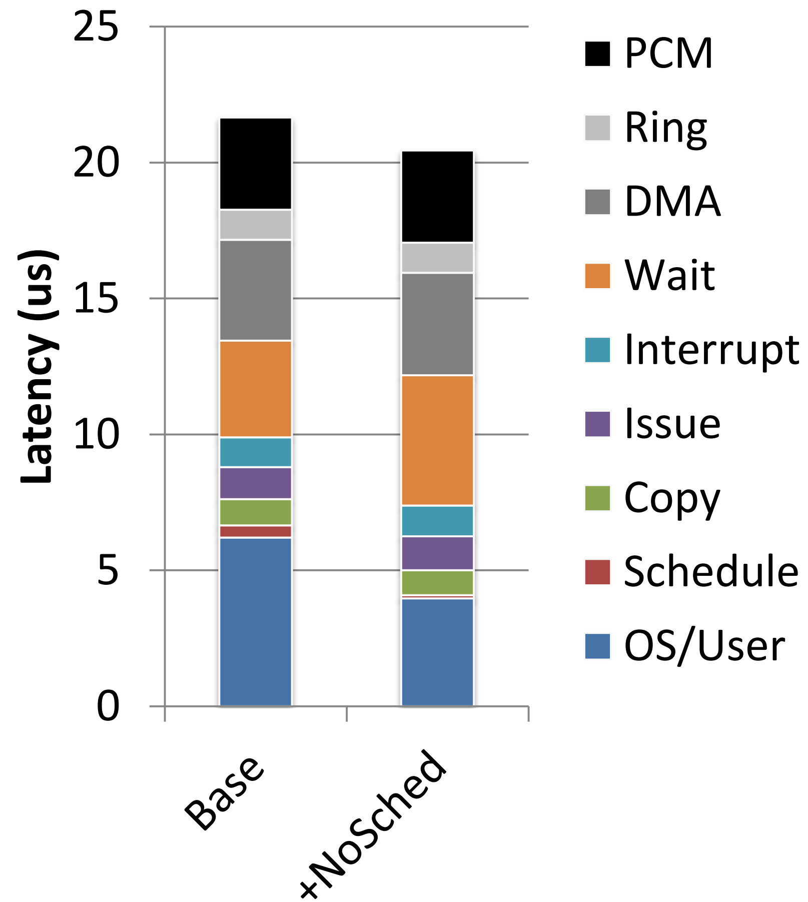
- Baseline Latency (4KB)
  - Hardware: 8.2 us
  - Software: 13.4 us

- Optimize hardware and software



Removing the IO Scheduler

- Reduces executed code
  - IO Sched code
  - Kernel request queuing
- Improves concurrency
  - Requests not serialized through IO scheduler
  - Multiple threads in driver
- 10% SW latency savings



Lock Free Tag Pool

- Compare-and-swap operations
- Tag indexed data structures
- Use processor ID as hint for where to search in tag structure
  - Reduces concurrency conflicts
  - Reduces cache-line misses

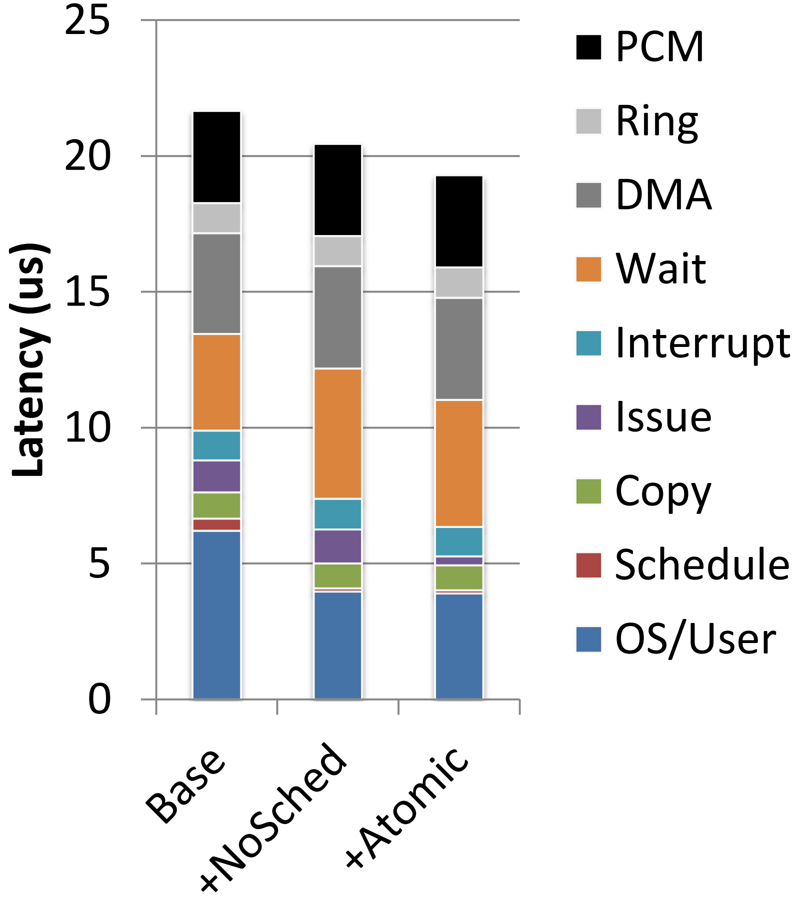
Co-design HW/SW Interface

- Baseline uses multiple PIO writes to send command
- Reduce command word to 64 bits
  - Allows a single PIO write to issue a request
  - No need for locks to protect request issue
- Remove DMA address from command
  - Pre-allocate buffers during initialization
  - Static buffer address for each tag

Atomic Operations

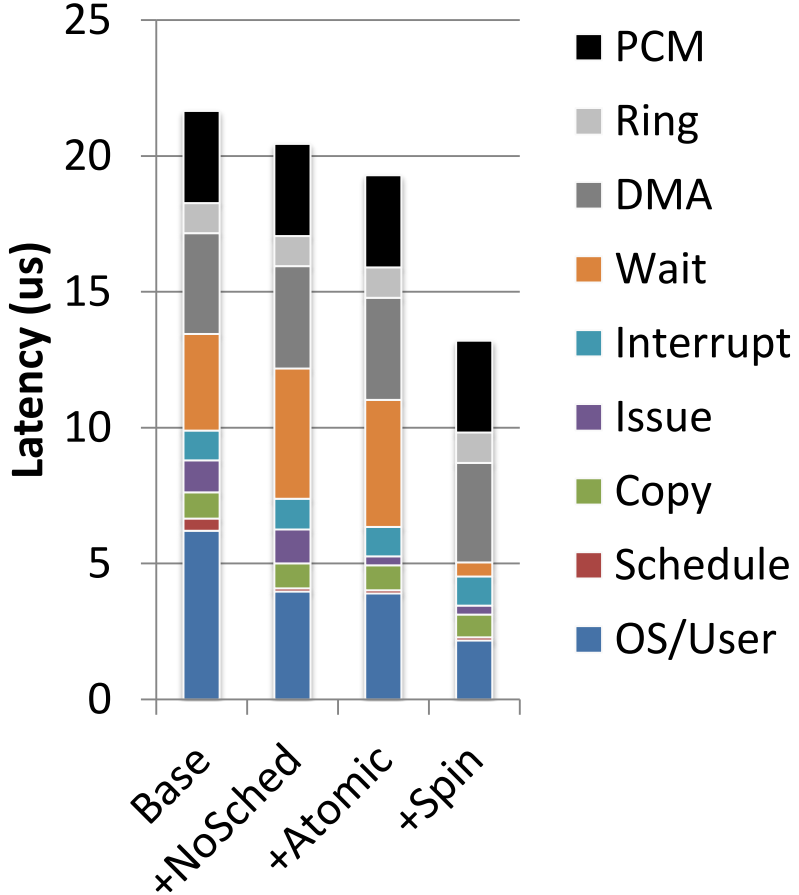
- Lock-Free Data Structures
- Atomic HW/SW Interface

- Increased concurrency
- 10% less latency vs. NoSched



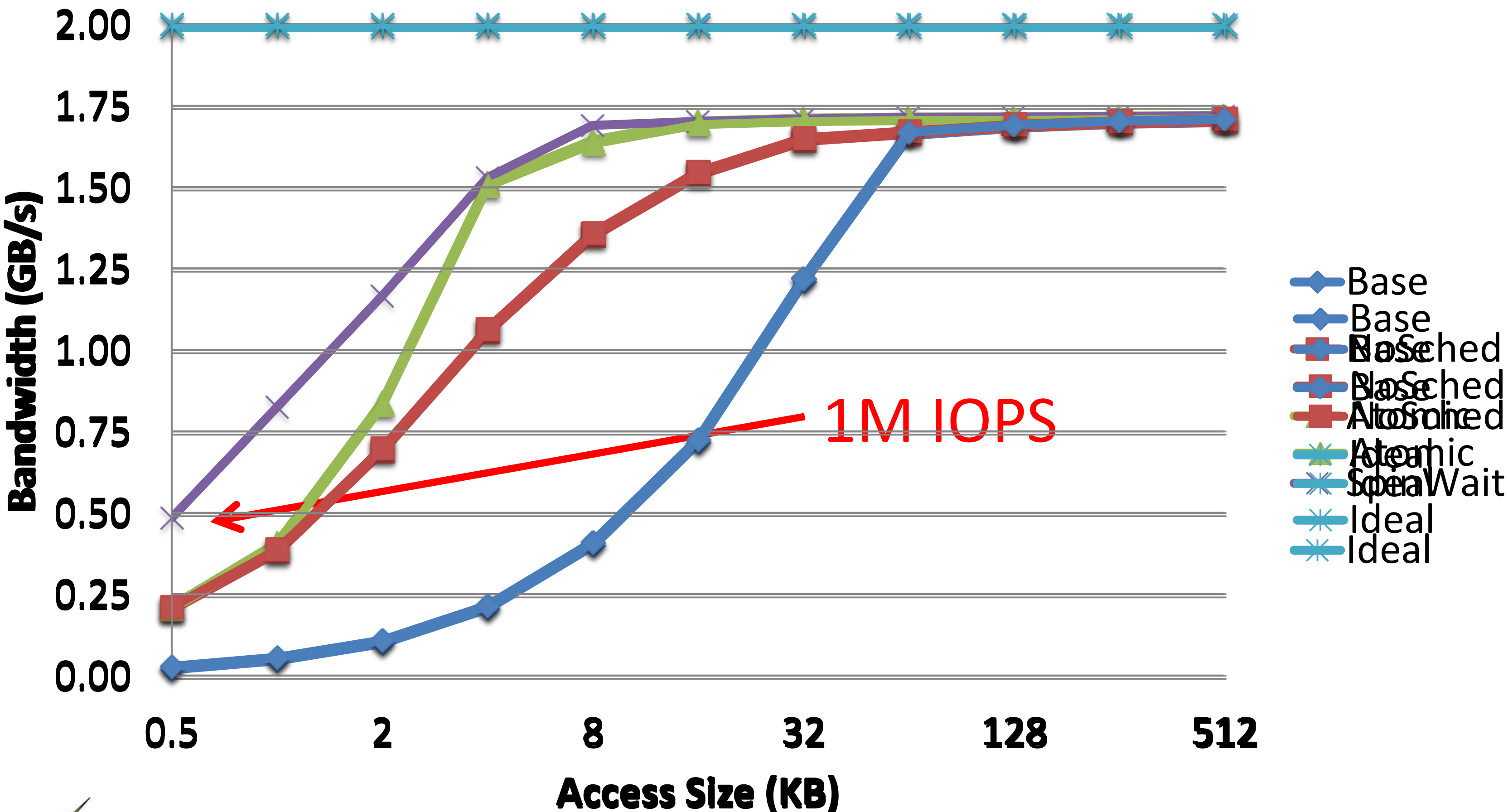
Add Spin-Waits

- Spin-waits vs. sleeping
  - Spin for < 4KB requests
  - Sleep for larger requests
- 5us of software latency
  - 54% less SW vs. Atomic
  - 62% vs. Base



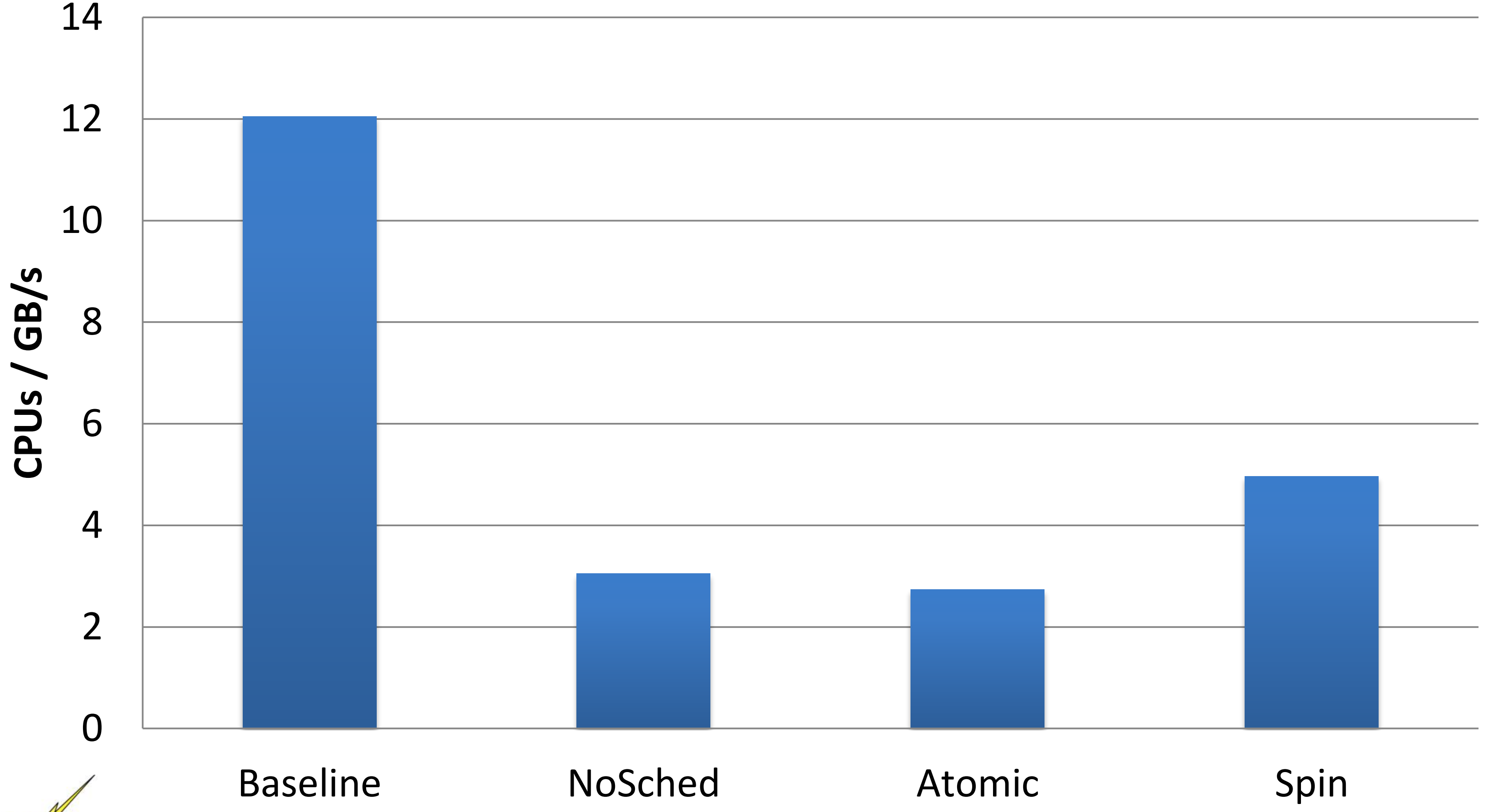
Moneta Bandwidth

Random Write Accesses



CPU Utilization

Random 4KB Read/Write Accesses



Zero-Copy

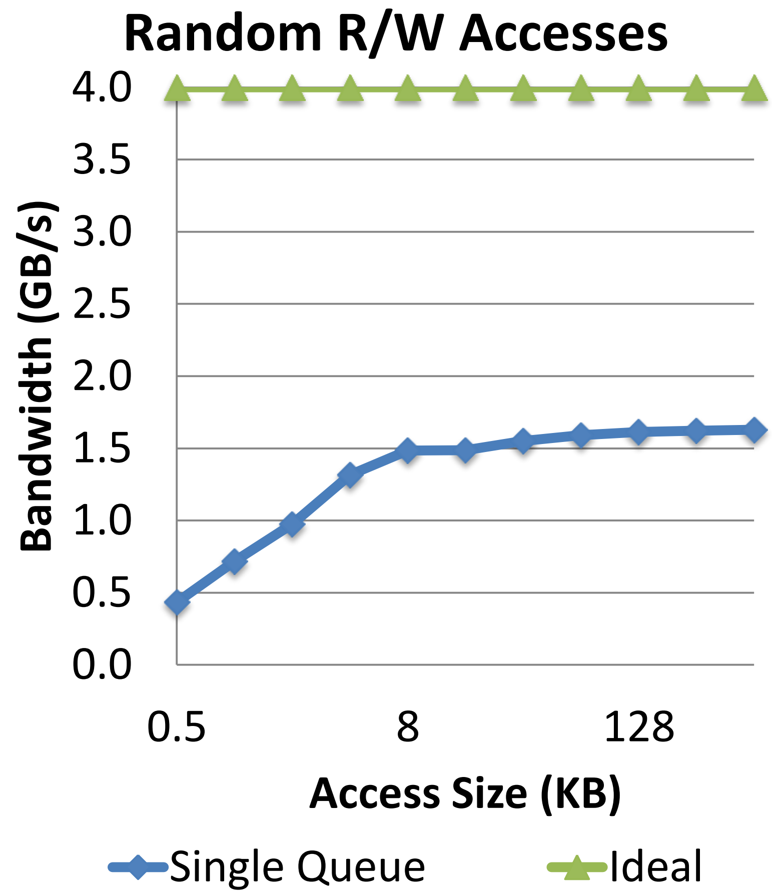
- Trade-off multiple PIO writes and zero-copy IO
- Currently faster for us to copy in the driver than it is to write multiple words to the hardware to issue a request
- 128-bit or cache-line sized PIO writes needed

Overview

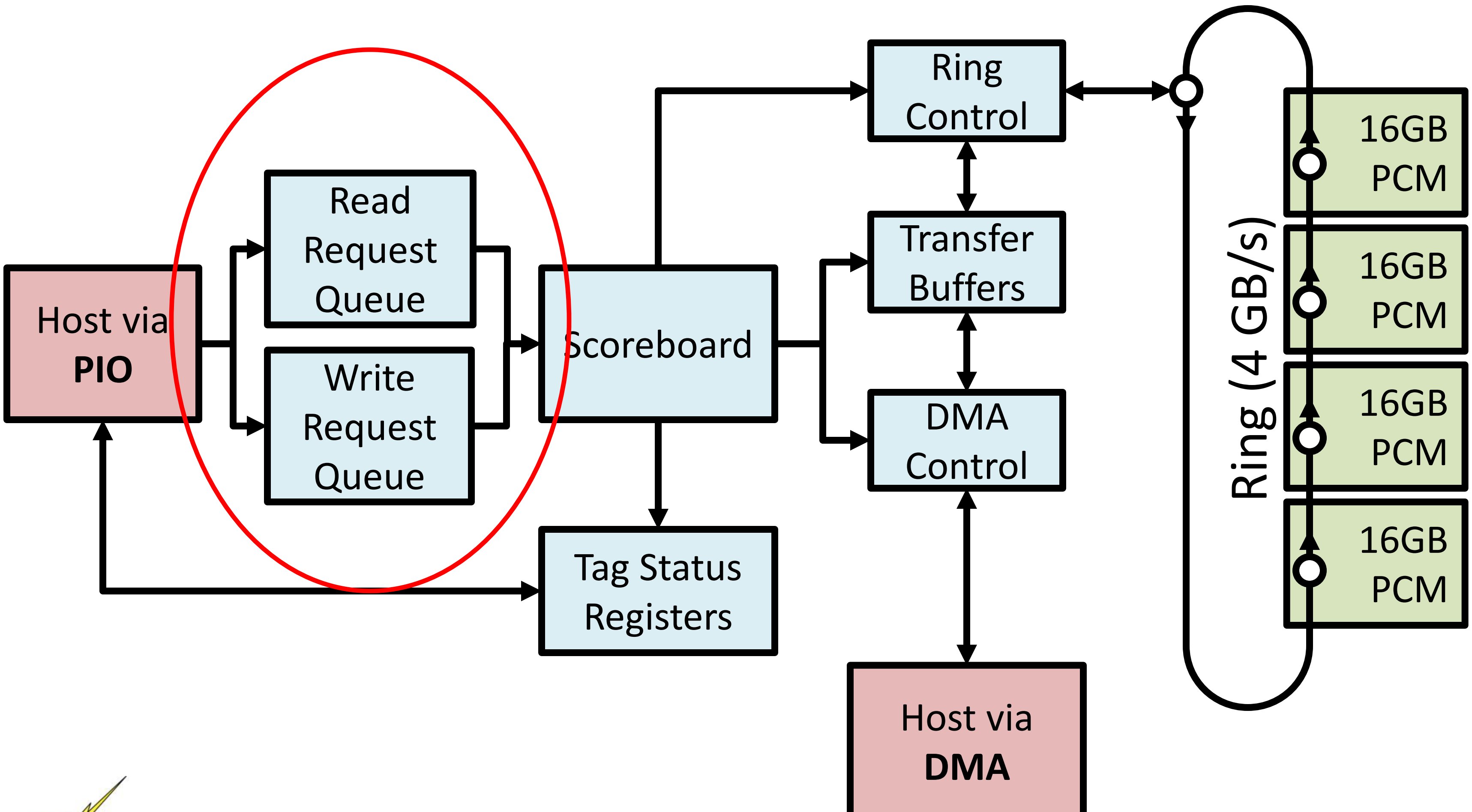
- Motivation
- Moneta Architecture
- Optimizing Moneta
  - Interface & Software
  - **Microarchitecture**
- Performance Analysis
- Conclusion

Balancing Bandwidth Usage

- Full duplex PCIe should see better R/W BW
- Smarter HW request scheduling = more BW
  - Two request queues: one for reads, one for writes
  - Alternate between Qs on each buffer allocation

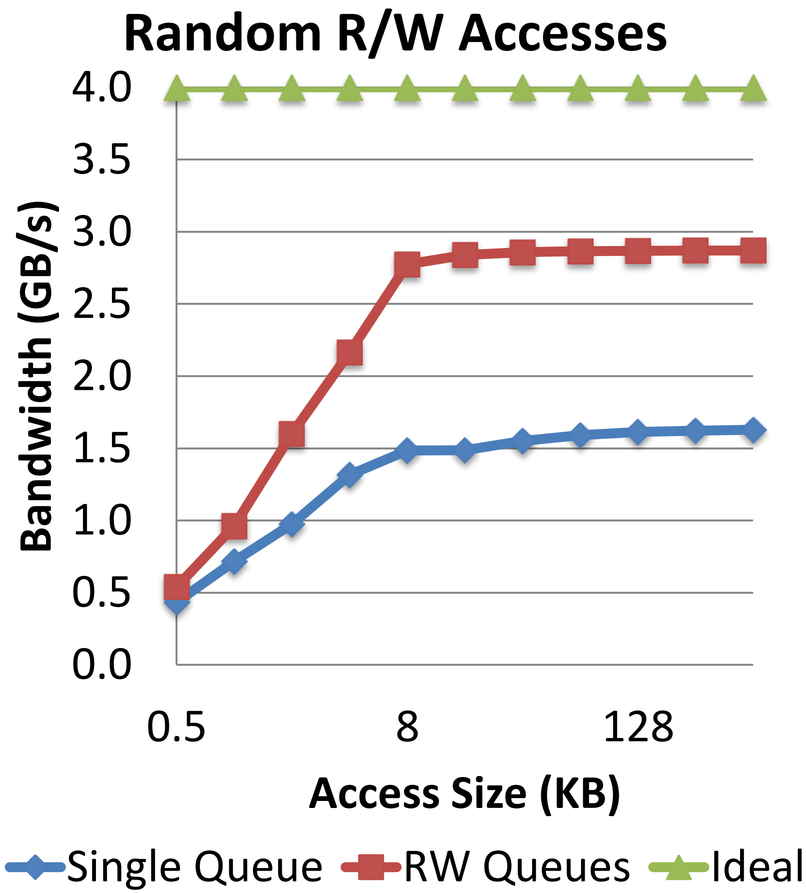


Moneta Architecture Changes



Balancing Bandwidth Usage

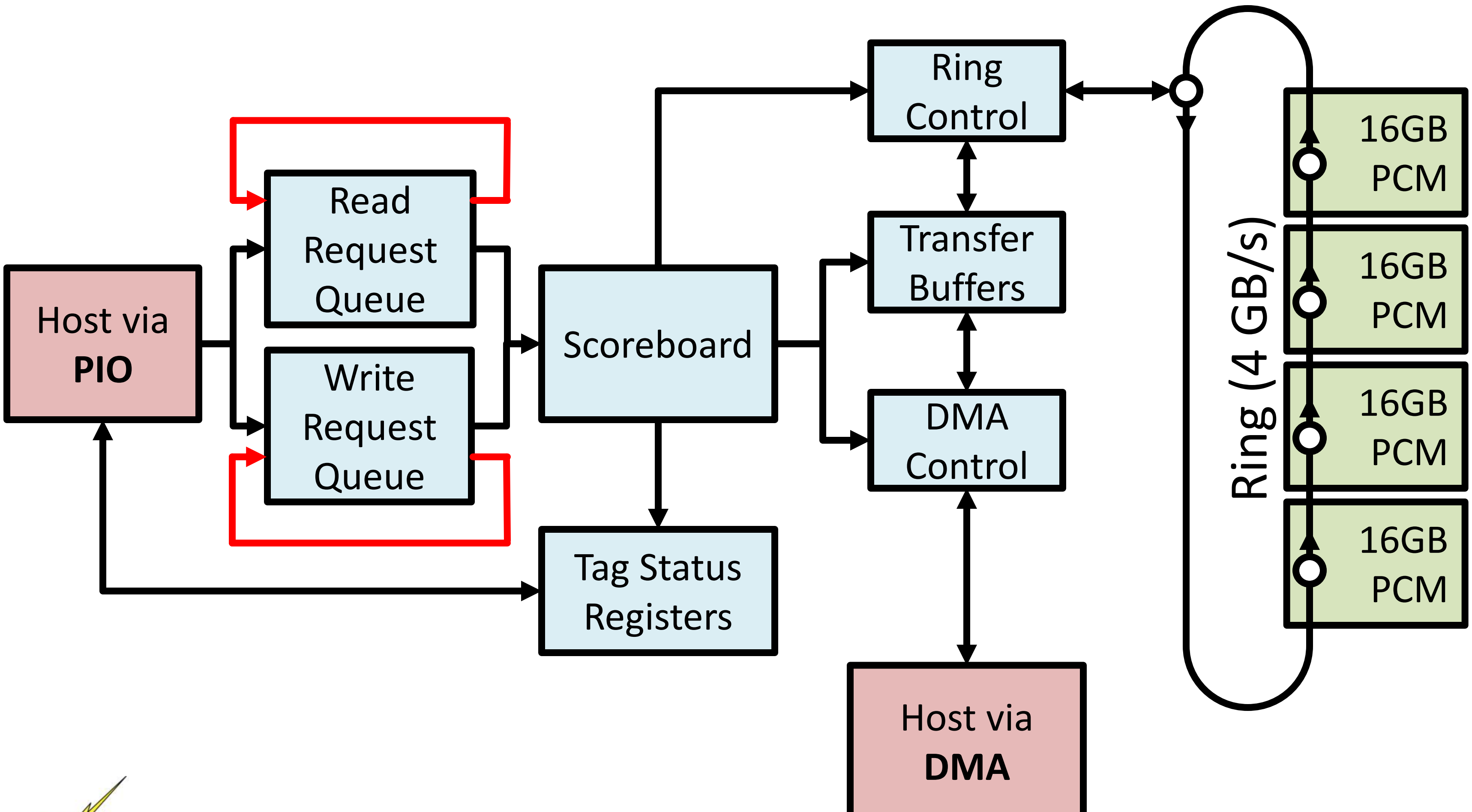
- Full duplex PCIe should see better R/W BW
- Smarter HW request scheduling = more BW
  - Two request queues: one for reads, one for writes
  - Alternate between Qs on each buffer allocation



Round-Robin Scheduling

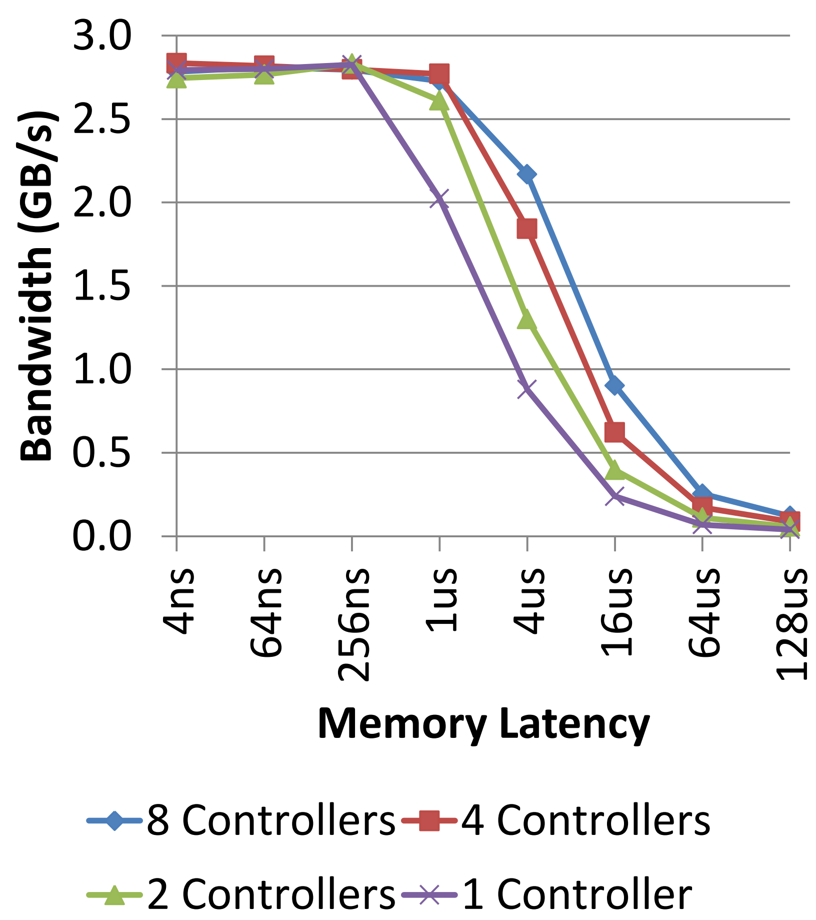
- Prevent requests from starving other requests
- Allocate a buffer and then put request at back of queue
- Attains much better small request throughput in the presence of large requests
- 12x improvement in small request BW

Moneta Architecture Changes



Effects of Memory Latency

- Moneta tolerates memory latencies up to 1us without BW loss
- Increased parallelism hides extra memory latency



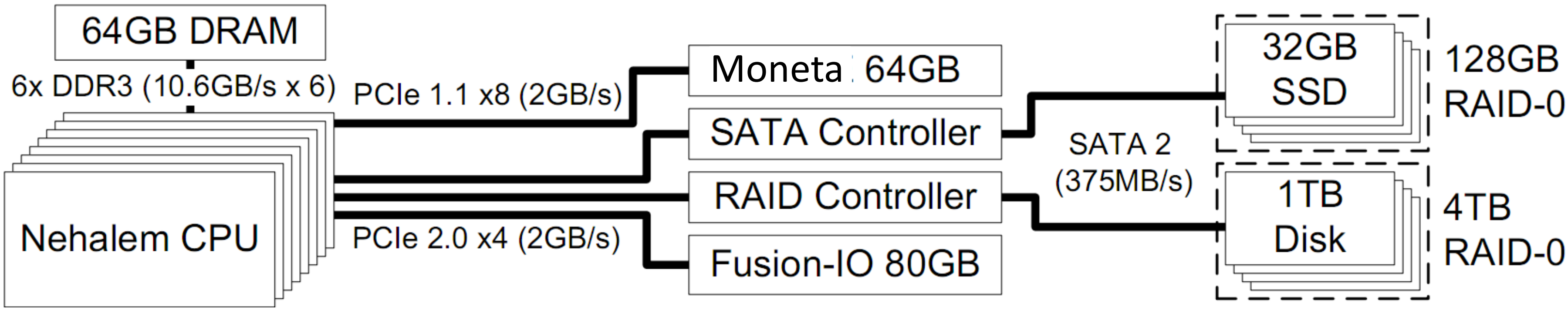
NVMs for Storage vs DRAM Replacement

- Write coalescing
  - Storage must guarantee durability by closing row
  - DRAM leaves row open to enable coalescing
- Row buffer size should match access size
  - Large accesses for storage
  - Cache-line sized for memory
- Peak memory activity limited by PCIe BW
- Storage and DRAM replacement are different and should be optimized differently

Overview

- Motivation
- Moneta Architecture
- Optimizing Moneta
- **Performance Analysis**
- Conclusion

System Overview



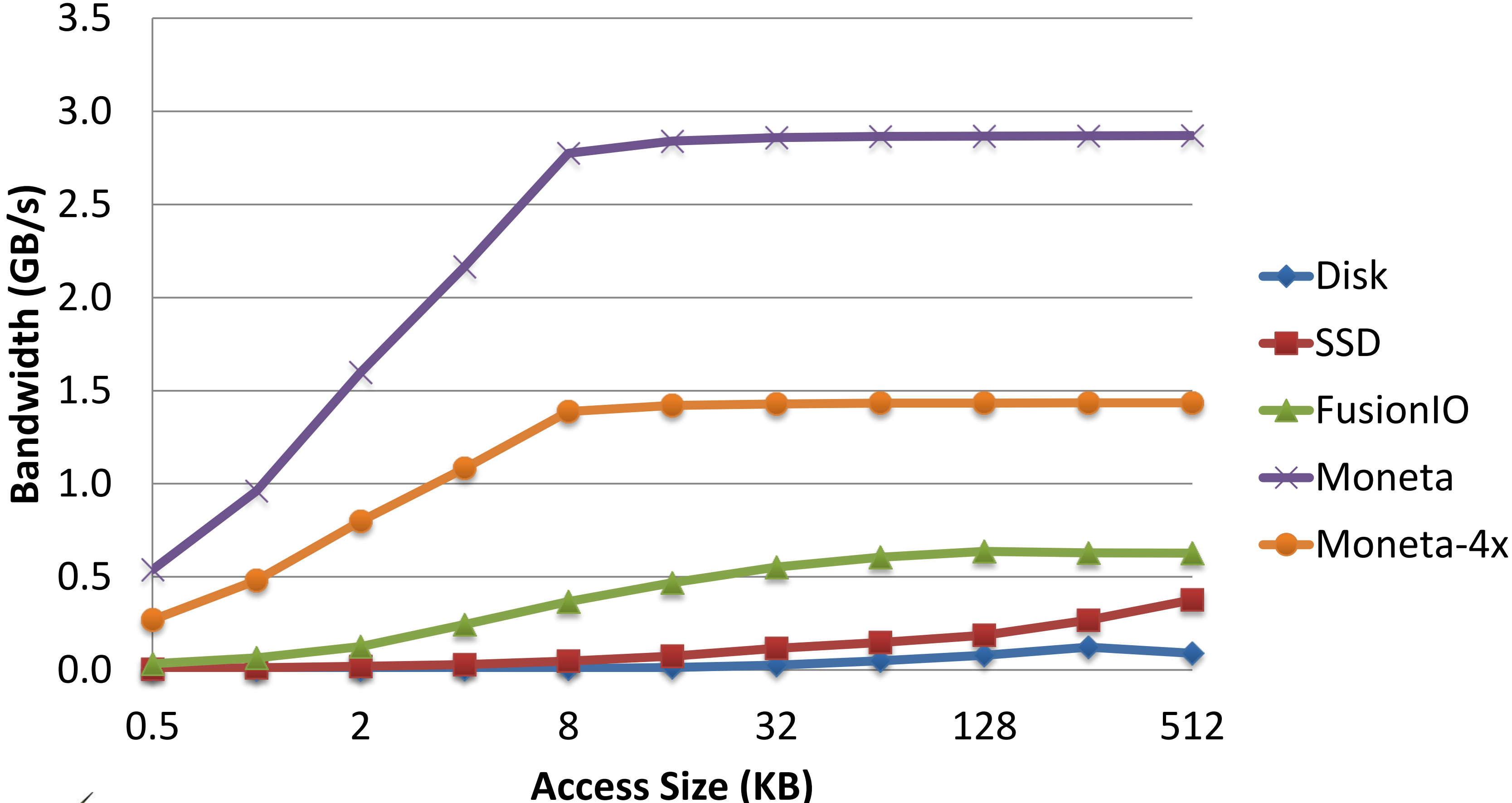
Memory and Device	Interconnect	Capacity
Fusion-IO IODrive	PCIe 4x	80GB
SLC NAND Flash SW RAID-0	PCIe 4x SATA 2 Controller	128GB
Disk HW RAID-0	PCIe 4x RAID Controller	4TB
Moneta	PCIe 8x	64GB
Moneta-4x	PCIe 4x	64GB

Workloads

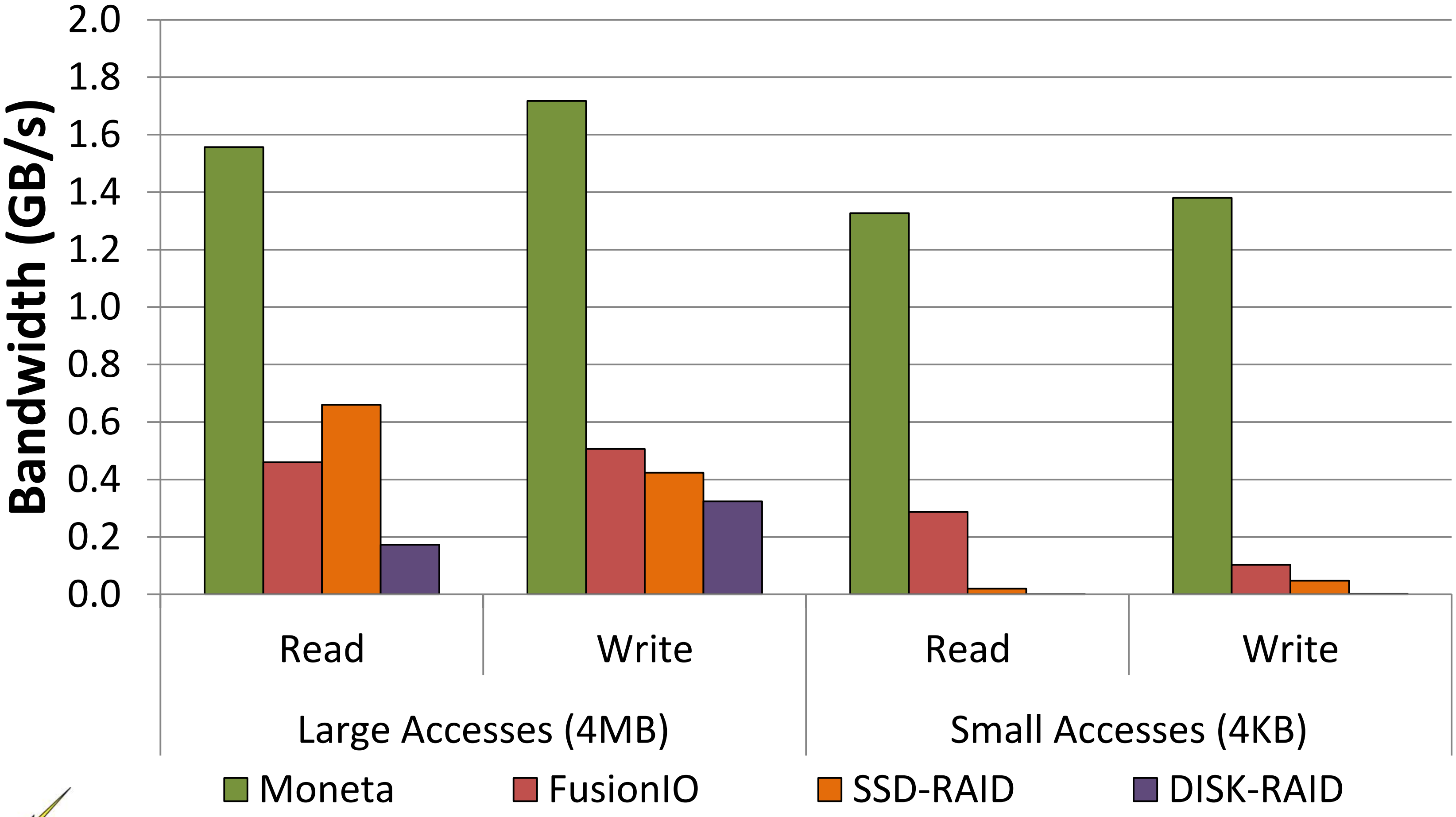
Name	Footprint	Description
Basic IO Benchmarks		
XDD NoFS	64 GB	Low-level IO performance without file system
XDD XFS	64 GB	XFS file system performance
Database Applications		
Berkeley-DB Btree	16 GB	Transactional updates to btree key/value store
Berkeley-DB HashTable	16 GB	Transactional updates to hash table key/value store
BiologicalNetworks	35 GB	Biological database queried for properties of genes and biological-networks
PTF	50 GB	Palomar Transient Factory sky survey queries
Memory-hungry Applications		
DGEMM	21 GB	Matrix multiply with 30,000 x 30,000 matrices
NAS Parallel Benchmarks	8-35 GB	7 apps from NPB suite modeling scientific workloads

XDD Bandwidth Comparison

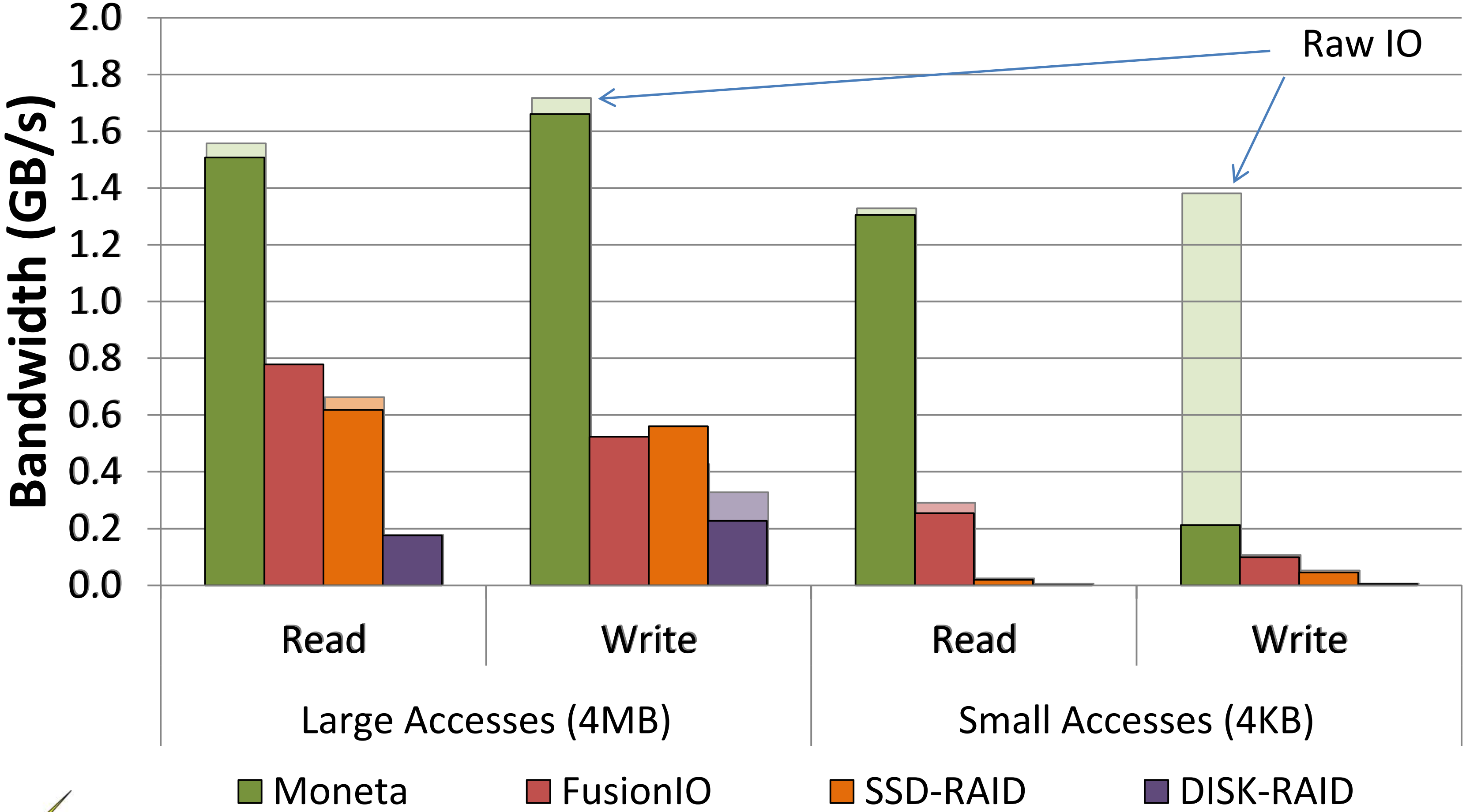
50/50 Read/Write Random Accesses



Bandwidth w/o File System

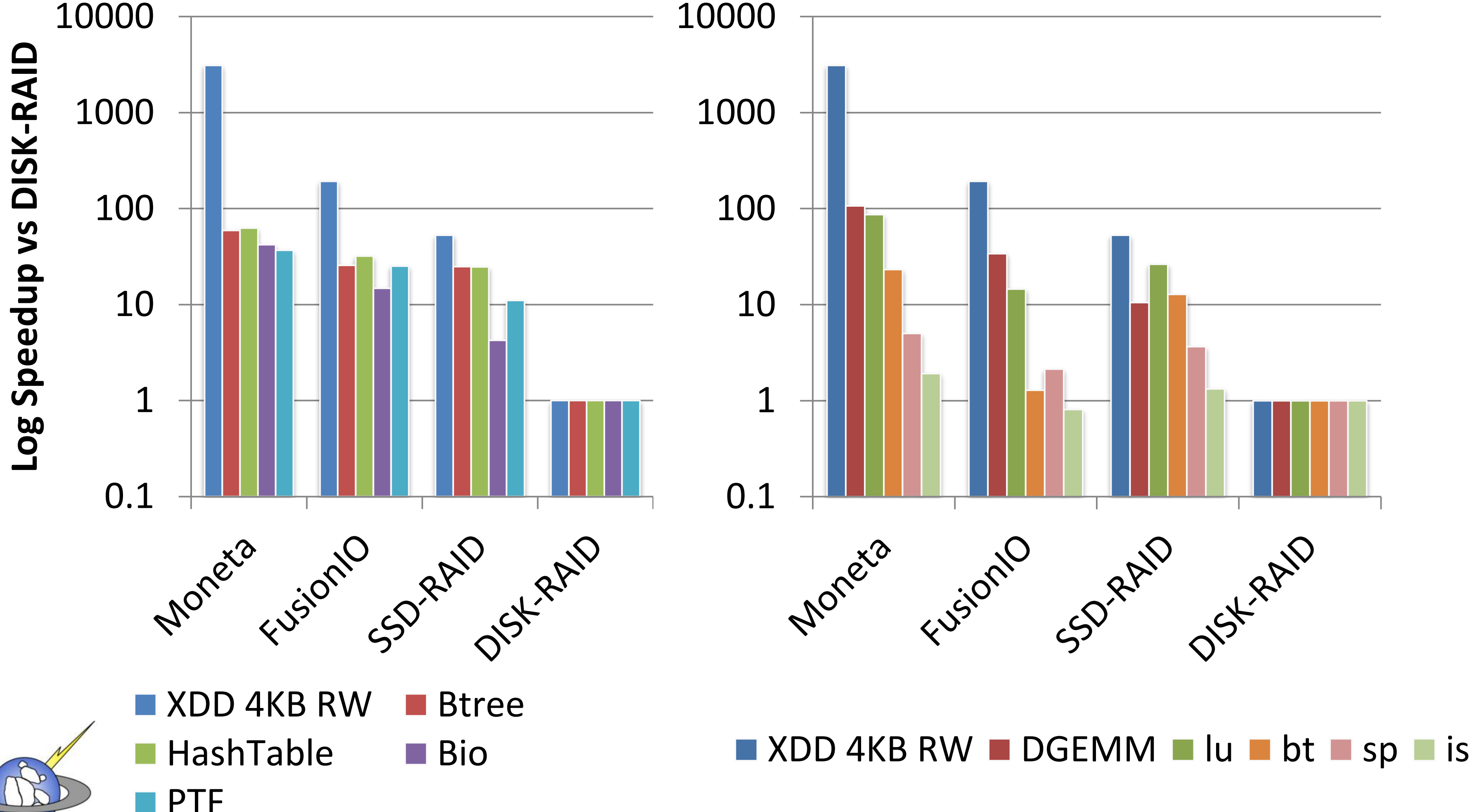


Bandwidth with XFS



Database & App Performance

An Opportunity for Leveraging Hardware Optimizations



Conclusion

- Fast advanced NVMs are coming soon
- We built Moneta to understand the impact of NVMs
  - Found that the interface and microarchitecture are both critical to getting excellent performance
  - Many opportunities to move software optimizations into hardware
- Many open questions exist about the architecture of fast SSDs and the systems they interface with

Thank You!

Any Questions?