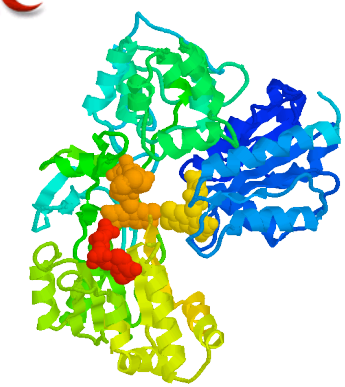
Redrawing the Boundary Between Software and Storage for Fast Non-Volatile Memories

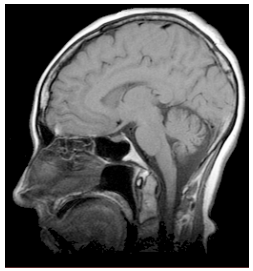
Steven Swanson

Director, Non-Volatile System Laboratory

Computer Science and Engineering

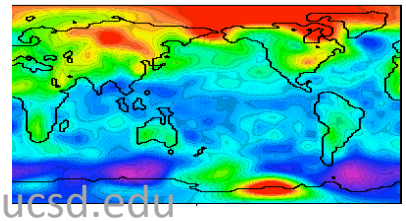
University of California, San Diego

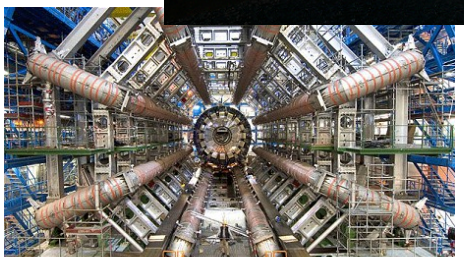


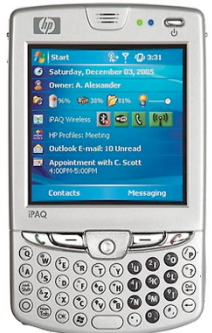


Hundreds of Gigabytes By 2008*

Welcome to the Data Age!



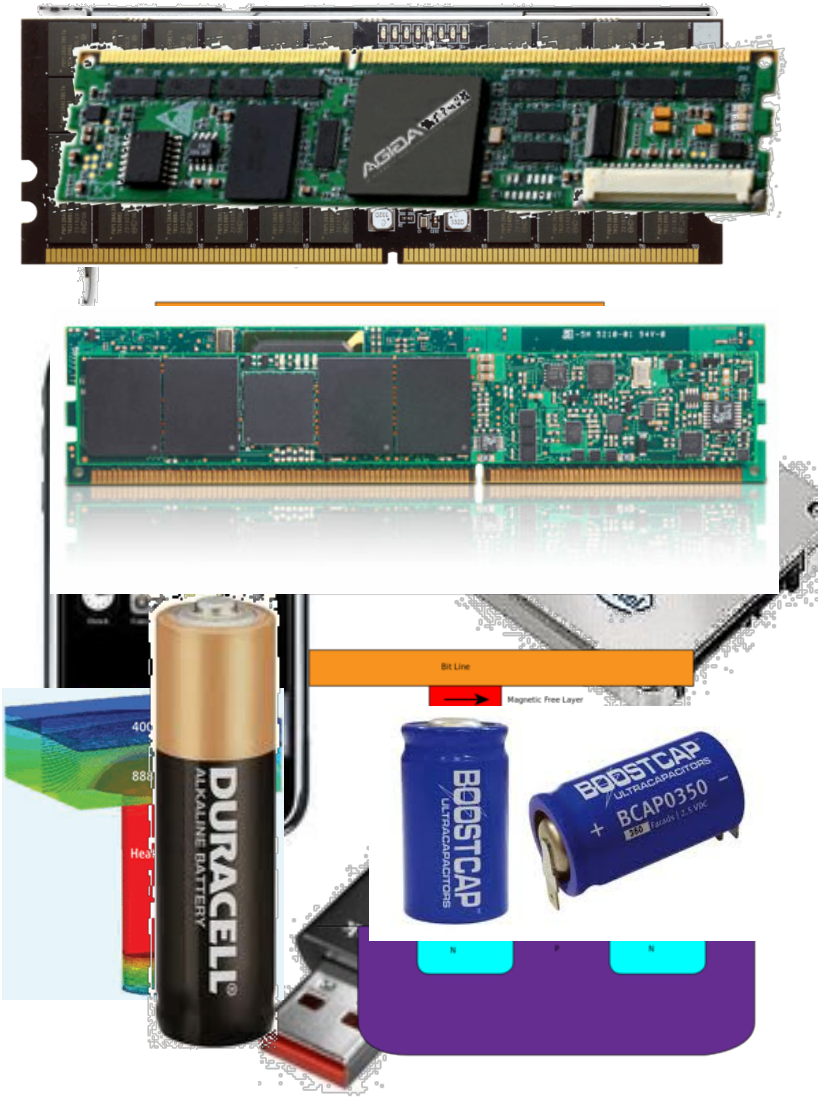




*http://hmi.ucsd.edu

Solid State Memories

- NAND flash
  - Ubiquitous, cheap
  - Sort of slow, idiosyncratic
- Phase change, Spin torque MRAMs, etc.
  - Not (yet) ready for prime time
  - DRAM-like speed
  - DRAM or flash-like density
- DRAM + Battery + flash
  - Fast, available today
  - Reliability?



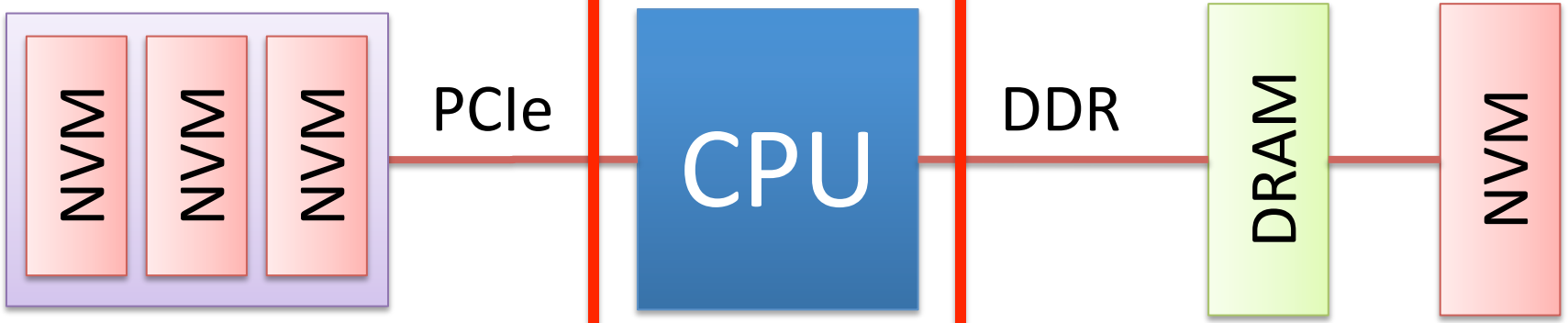
Integrating Them Into Systems

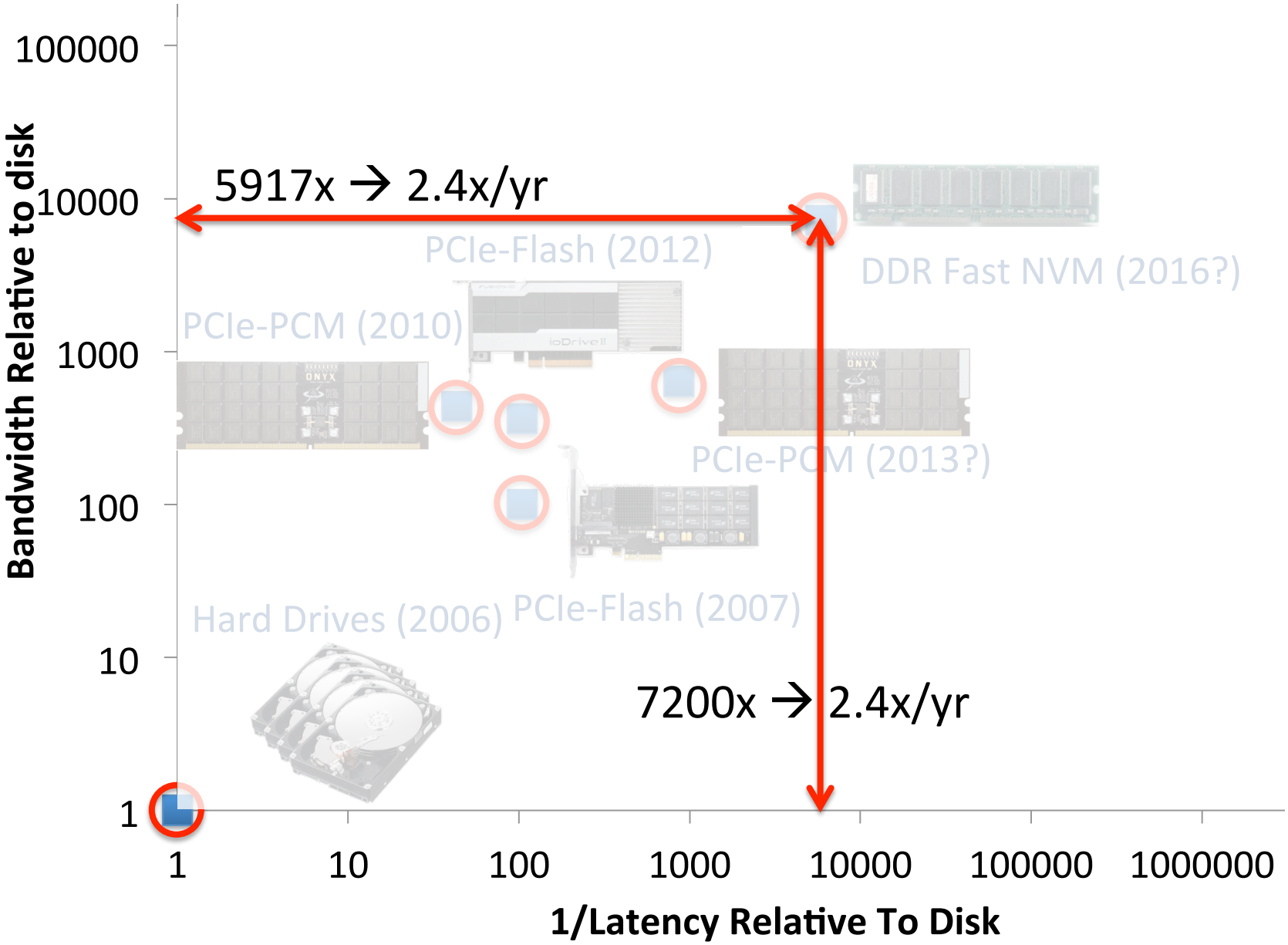
PCIe-attached NVMs

- Low Latency
- High bandwidth
- Flexible interface
- Flexible topology
- Scalable

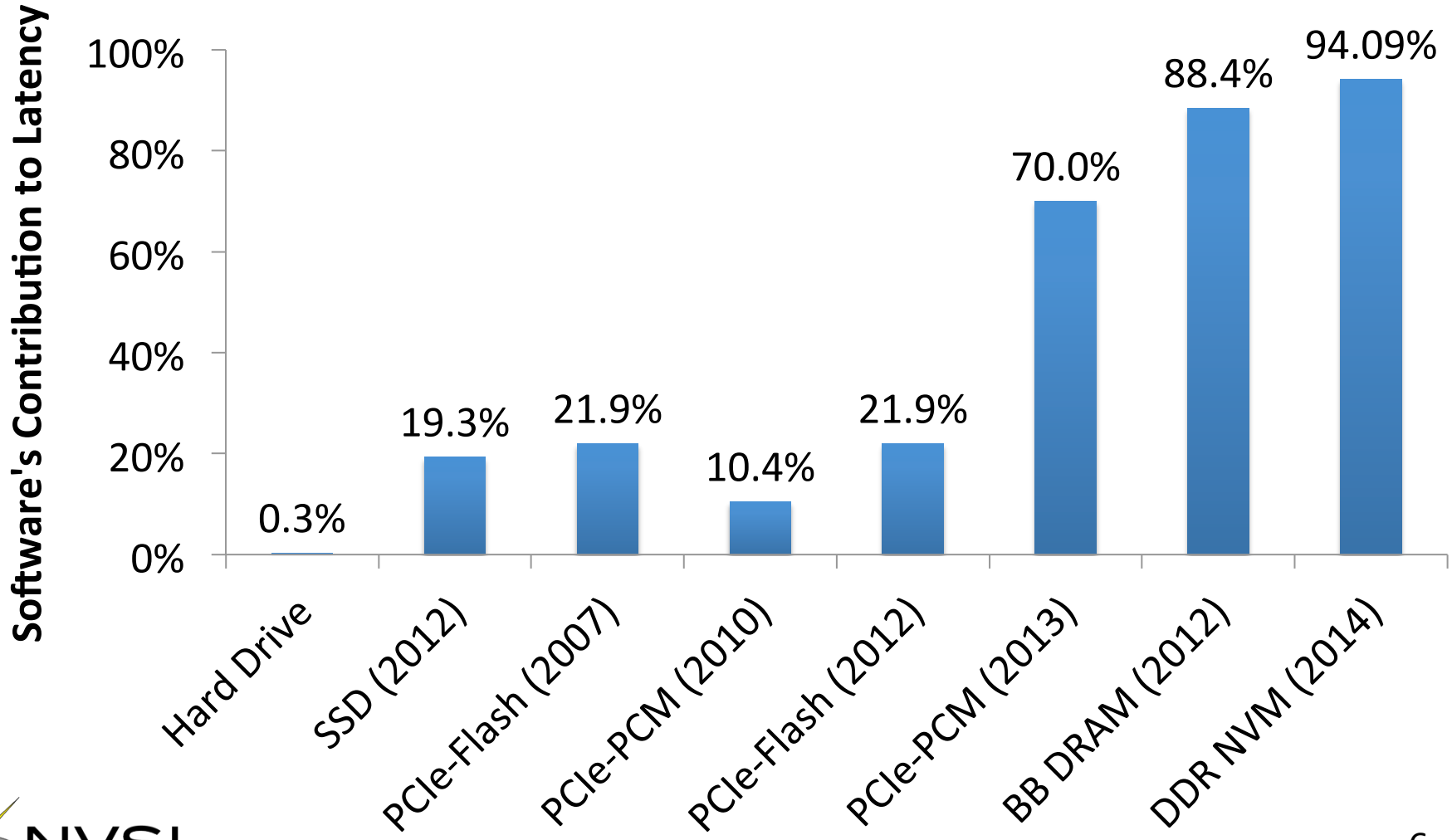
DDR-attached NVMs

- Lowest Latency
- Highest bandwidth
- Rigid interface
- Restricted topology
- Limited scalability



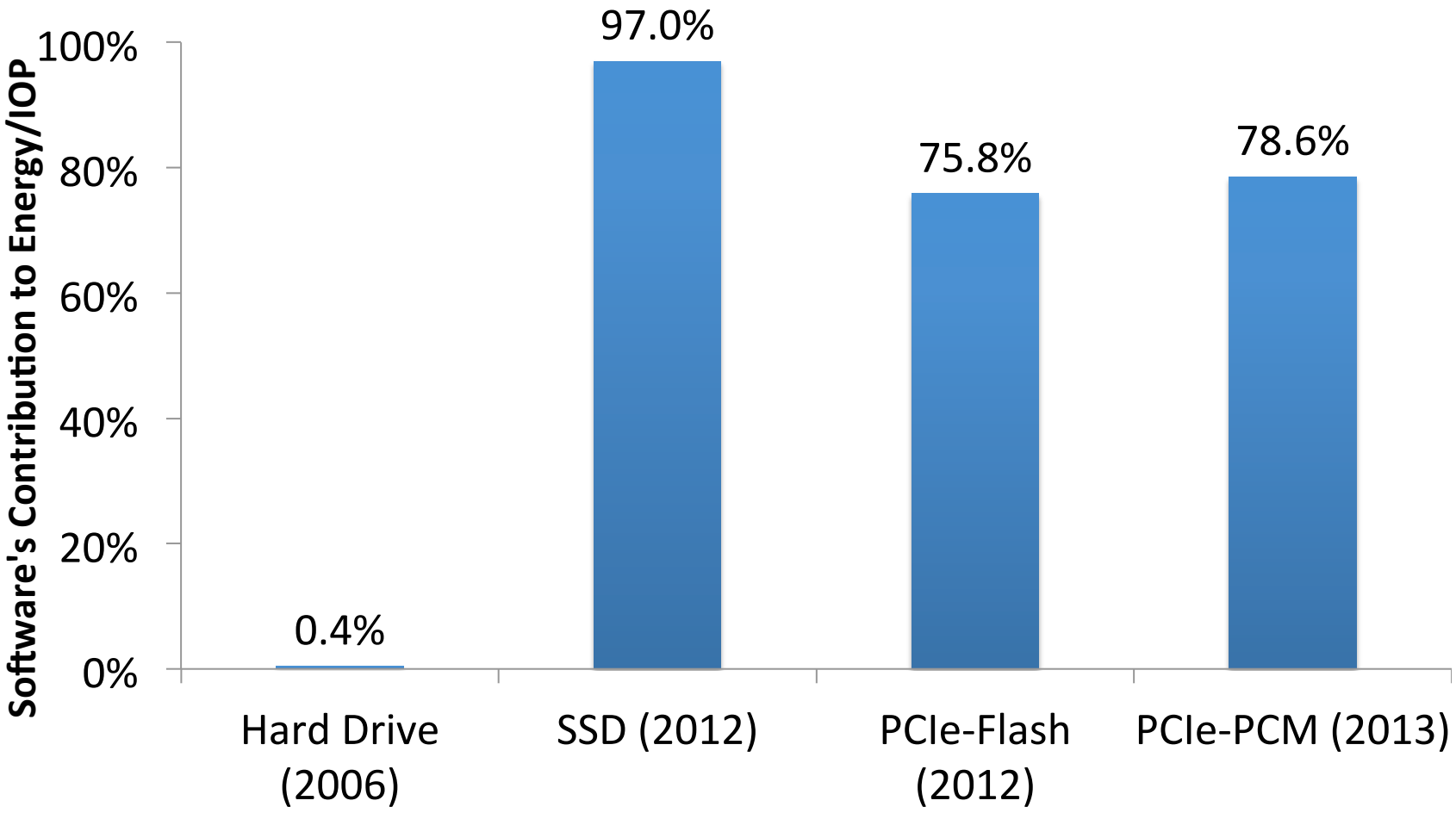

100000
10000
1000
100
10
1
Bandwidth Relative to disk
5917x → 2.4x/yr
PCIe-Flash (2012)
DDR Fast NVM (2016?)
PCIe-PCM (2010)
PCIe-PCM (2013?)
PCIe-Flash (2007)
Hard Drives (2006)
7200x → 2.4x/yr
1
10
100
1000
10000
100000
1000000
1/Latency Relative To Disk

Software Latency Will Dominate



NVSL
Non-volatile Systems Laboratory

Software Energy Will Dominate



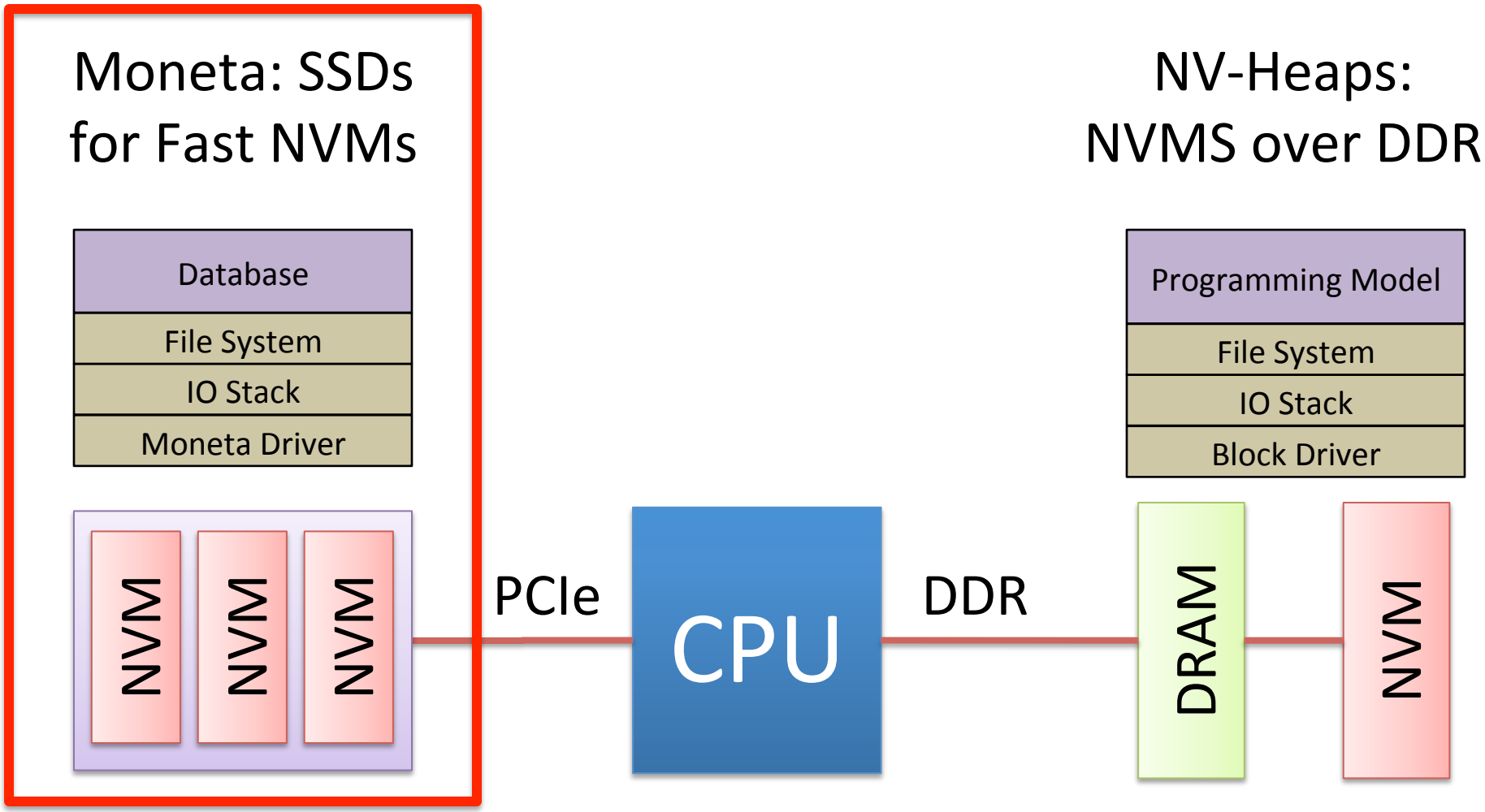
Software's New Role in Storage

For Disks: Adding Software Could Make the System Faster

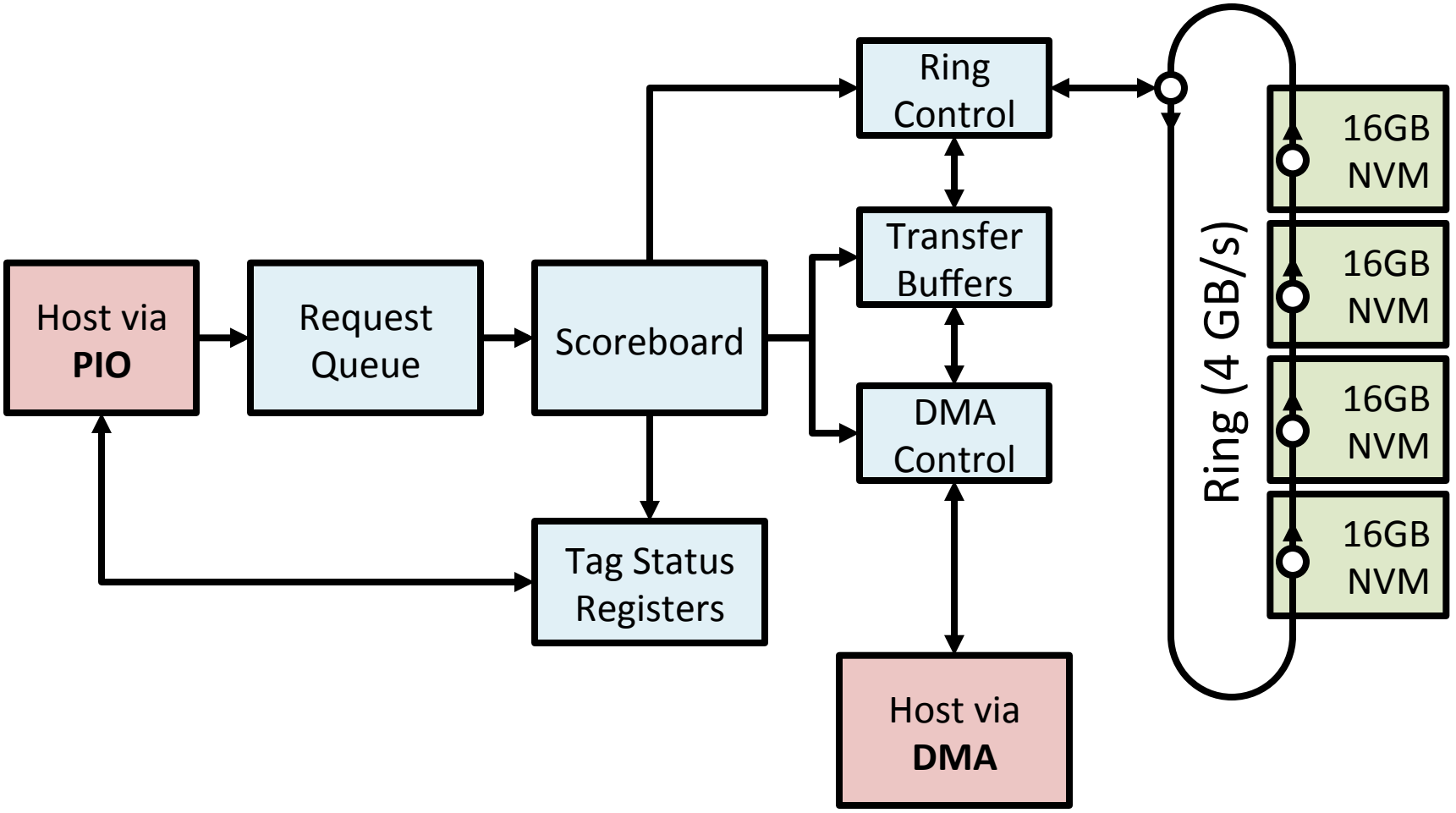
For NVMs: Adding Software Makes It Slower

- Reduce software interactions
- Remove disk-centric features
- Redesign IO software (OS, FS, languages, and apps)

Two Case Studies

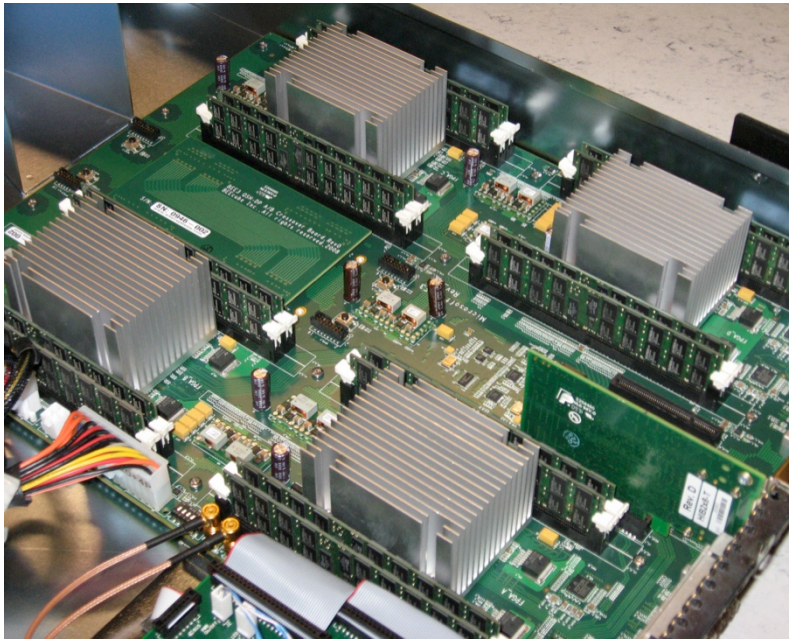


Moneta Internals: Performing a Write



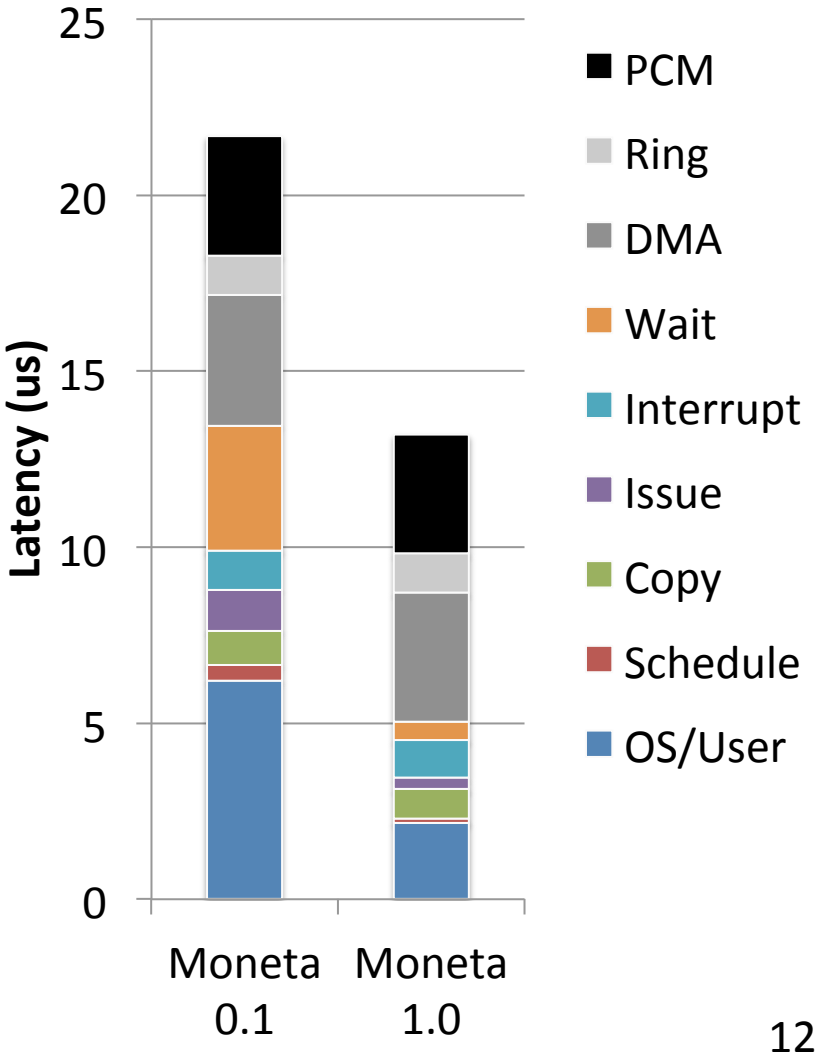
The Moneta Prototype

- FPGA-based implementation
- DDR2 DRAM emulates PCM
  - Configurable memory latency
  - 48 ns reads, 150 ns writes
  - 64GB across 8 controllers
- PCIe: 2 GB/s, full duplex

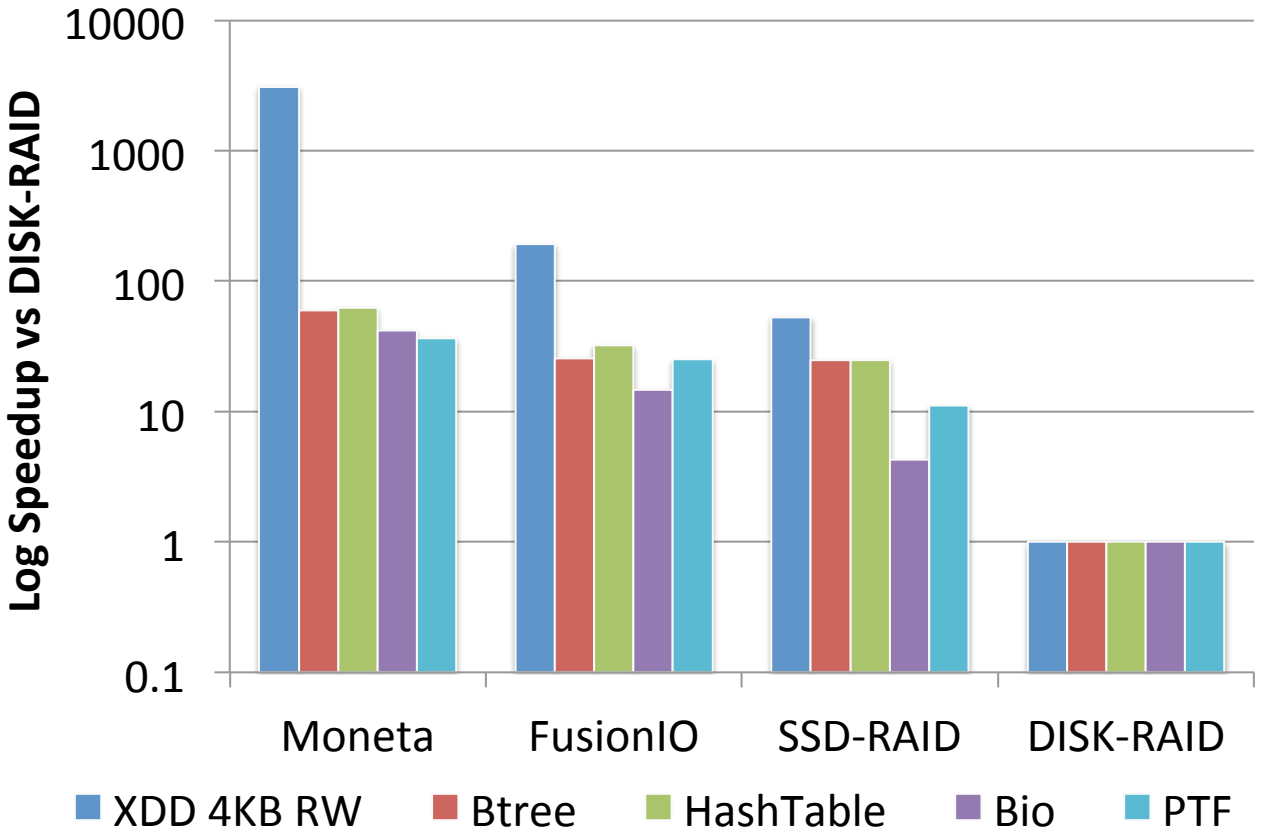


Optimizing Software for Moneta

- Optimizations
  - Remove IO scheduler
  - Redesigned HW interface
  - Remove locks
- SW latency drops from 13.4 us to 5us
  - 62% reduction in latency
  - Increased concurrency
- Bandwidth increases by up to 10x



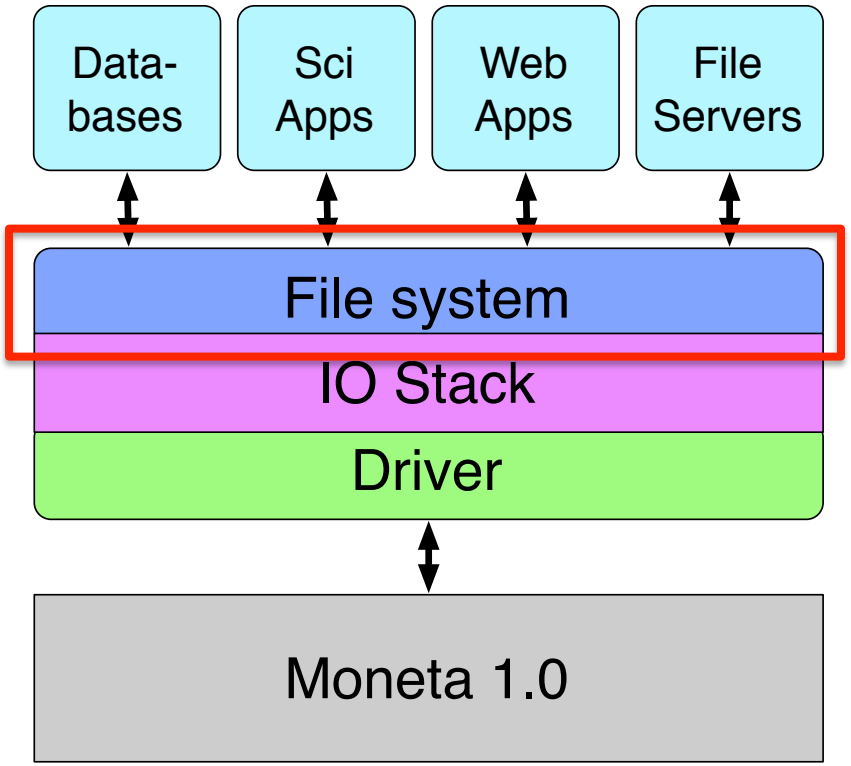
The App Gap

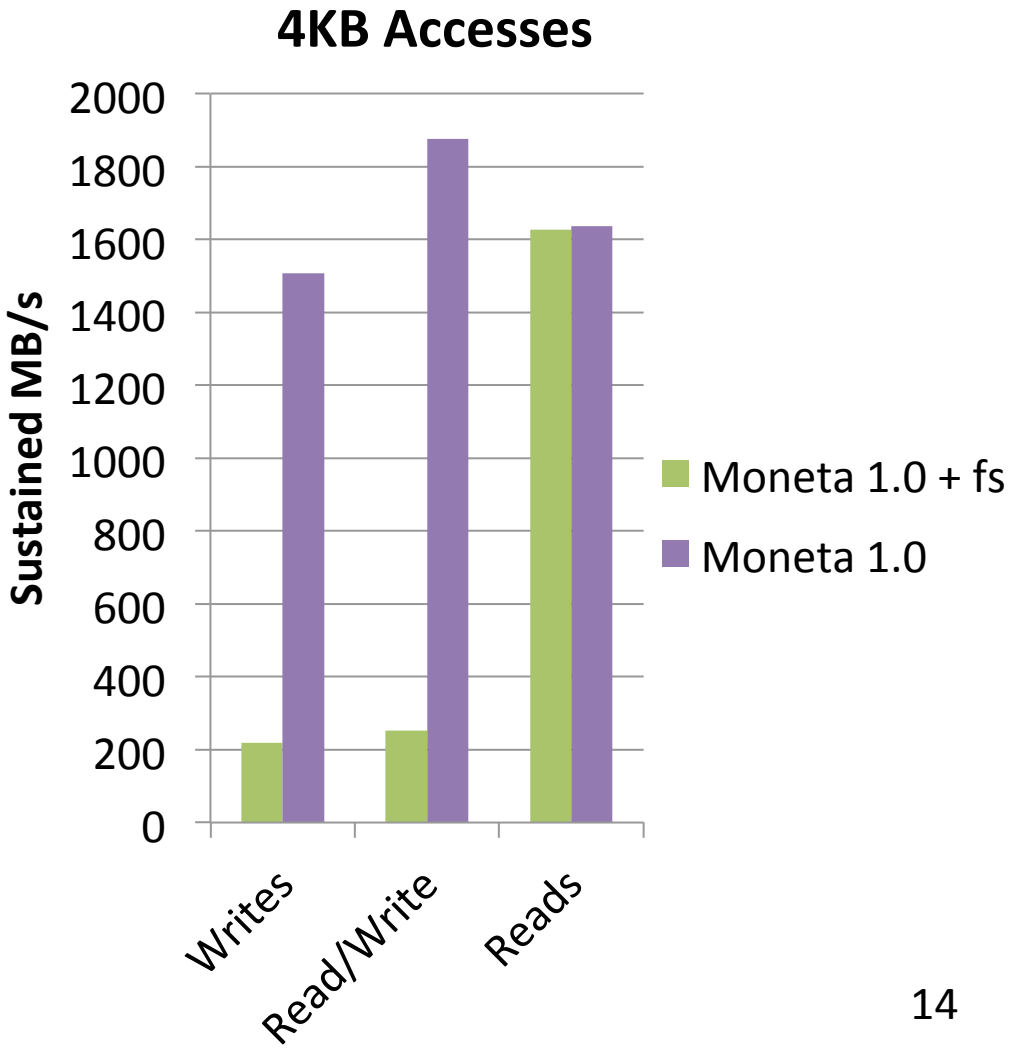


- We must close the App Gap for NVM success

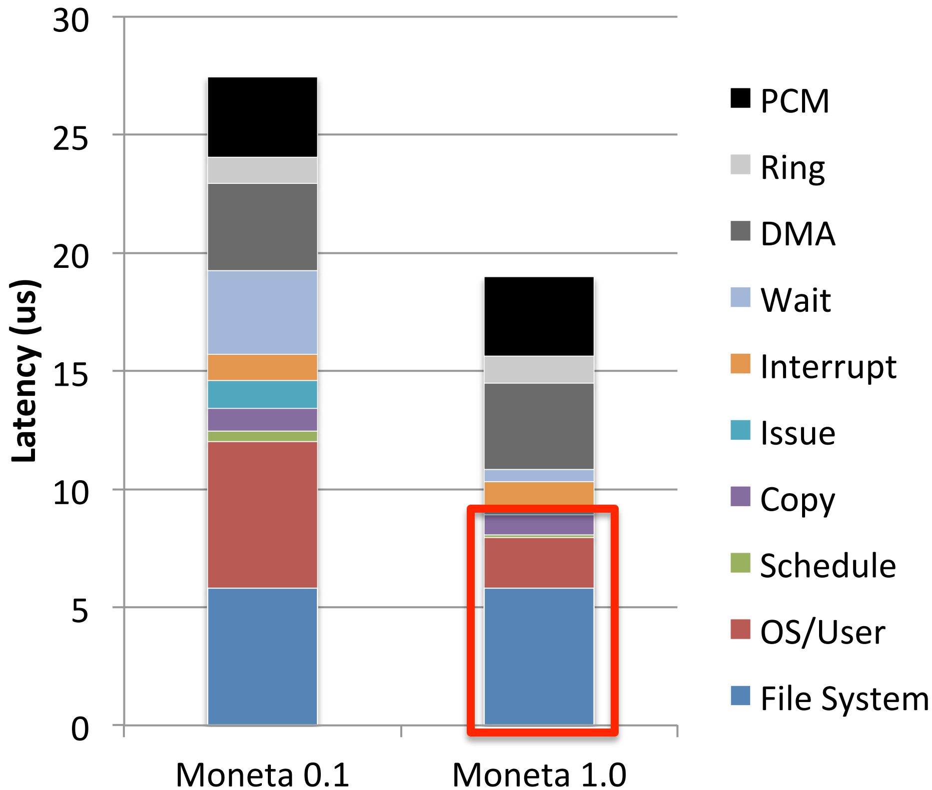
NVSL
Non-volatile Systems Laboratory

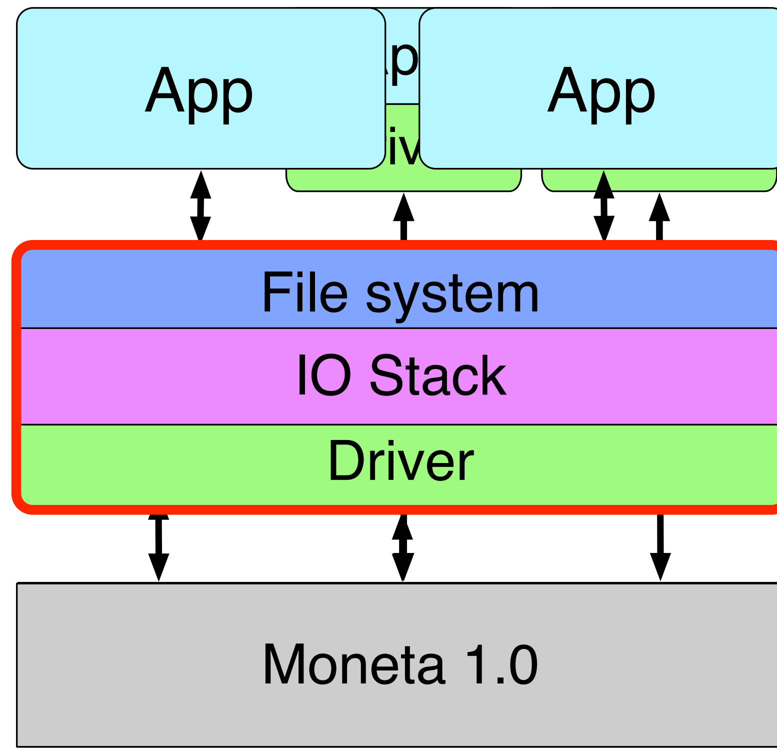
Where is the App Gap (Part I)?





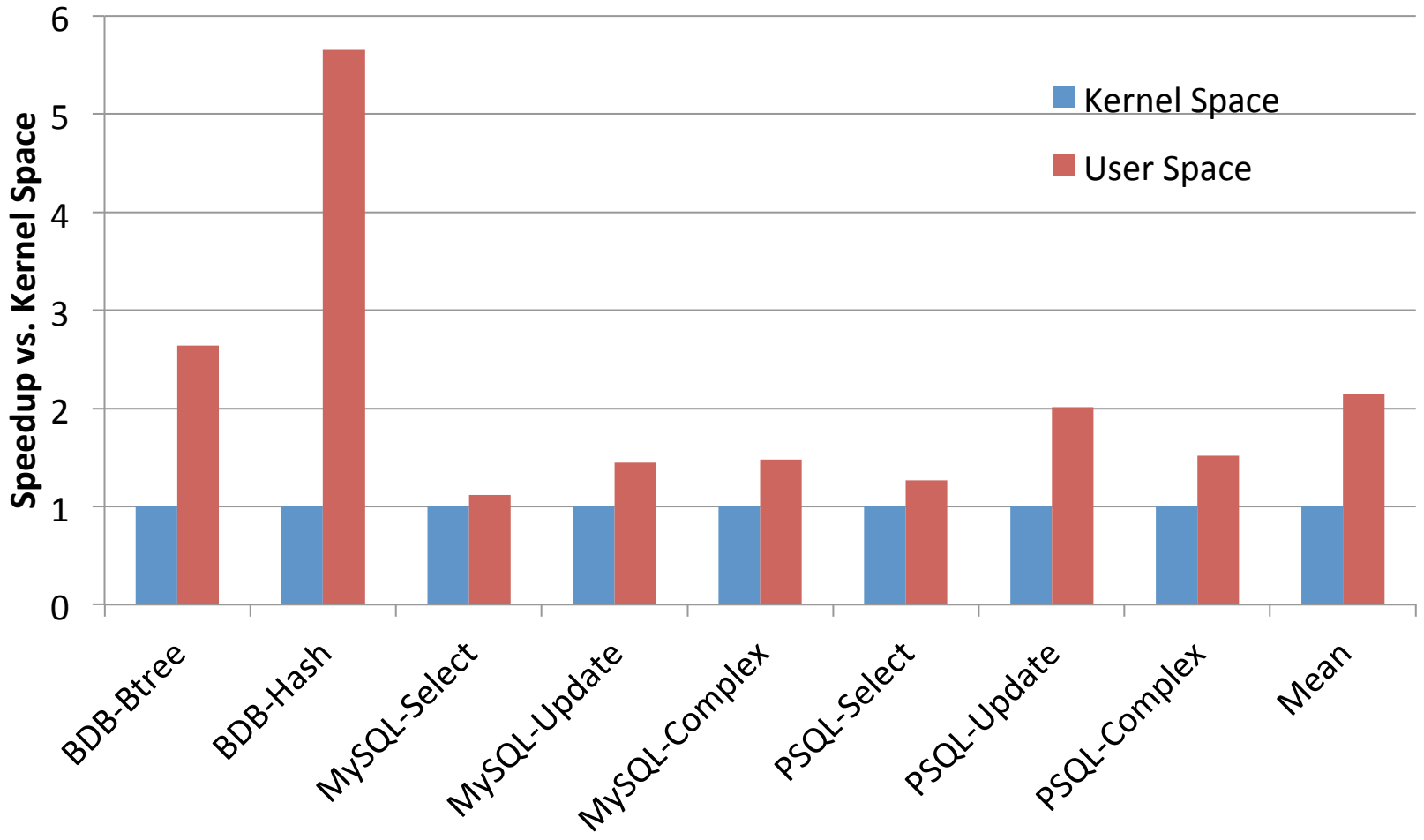
Eliminating FS and OS overheads





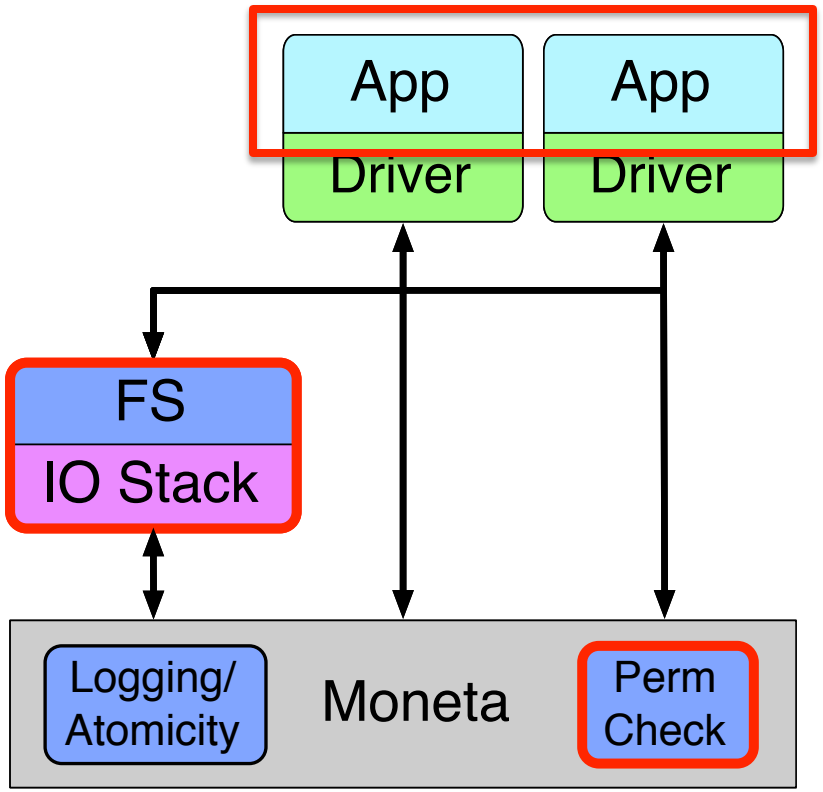
- [illegible]
  - Move protection checks to hardware
  - Allow applications to access Moneta directly

Some Progress on the App Gap



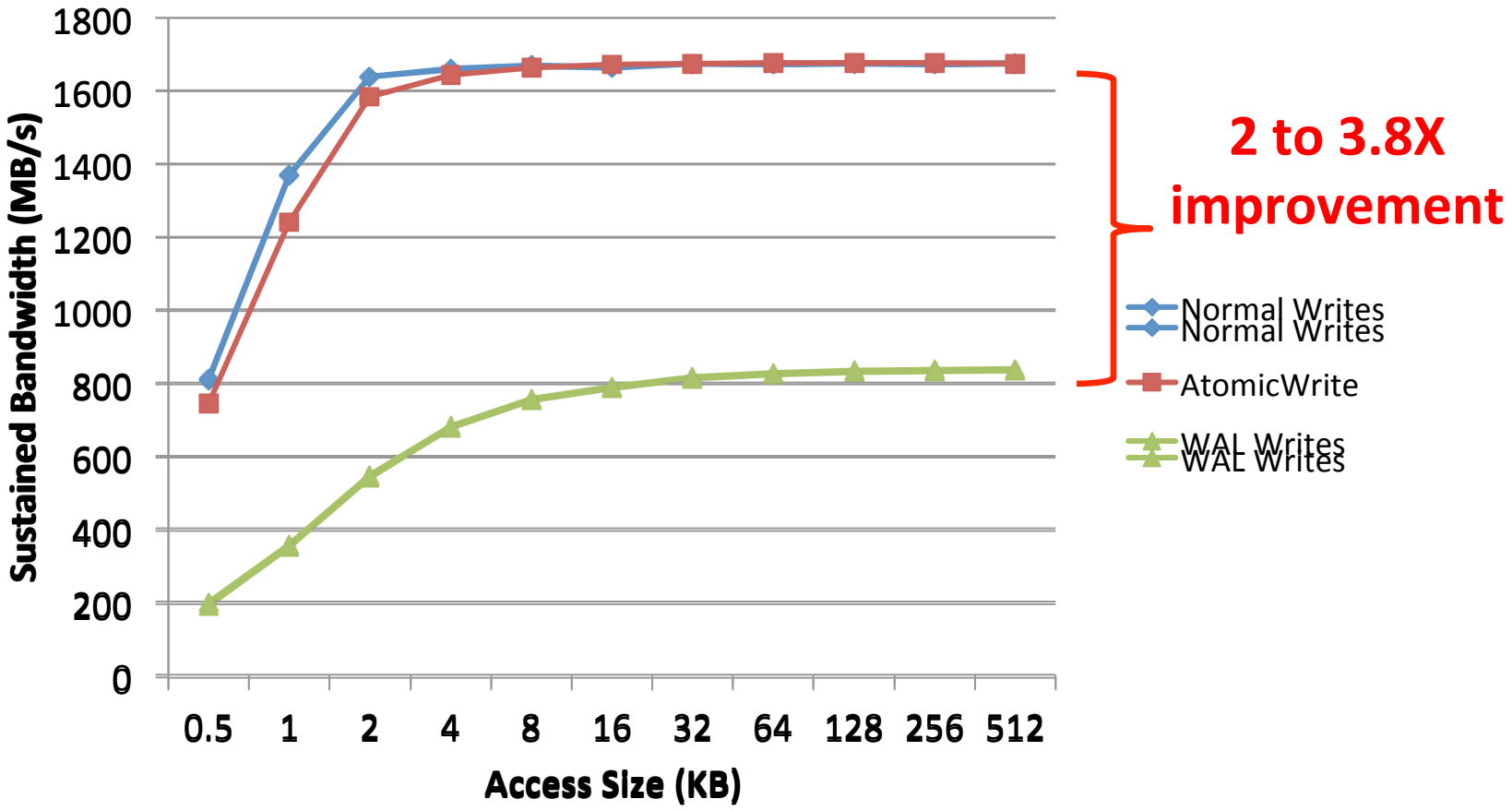
NVSL
Non-volatile Systems Laboratory

Where is the App Gap (Part II)?



- Write-ahead logging
  - e.g. ARIES
- Extra IO operations
  - Wasted software latency/power
  - Wasted IO performance
- The application can read and change the log contents

Bandwidth Comparison



NVSL
Non-volatile Systems Laboratory

ARIES – Write Ahead Logging for Databases on Disks

- ARIES provides useful database features
  - Atomicity
  - Durability
  - High Concurrency
  - Efficient recovery

We want to keep these

- But it makes disk-centric design decisions
  - Undo *and* redo logs to avoid random synchronous writes and provide IO scheduling flexibility
  - Page-based storage management to match disk semantics

Rethink these

ARIES' Disk-Centric Design Decisions

Design Decision	Advantages	Downsides
No-force	Eliminates synchronous random writes.	Requires synchronous writes to the redo log.
Steal	Longer sequential writes.	Requires an undo log.
Pages	Matches disk and OS page-based IO model.	Page granularity is not always ideal.

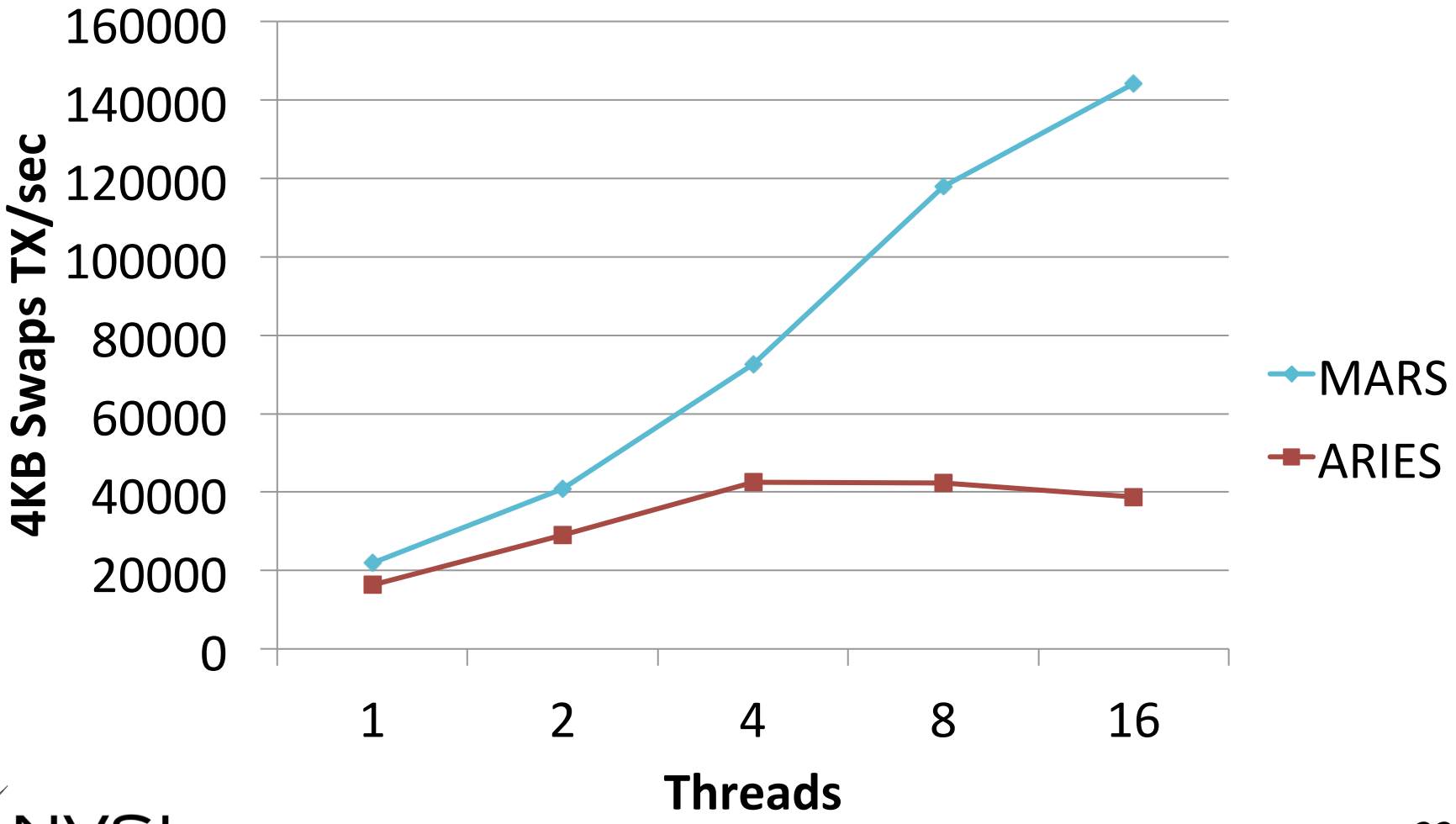
MARS: Rethinking ARIES for Moneta

No-force → Force updates on commit, but offload it to Moneta's memory controllers.

Steal → Get rid of undo logging.
Update redo log contents instead.

Pages → Software can choose object/page granularity.

ARIES vs MARS

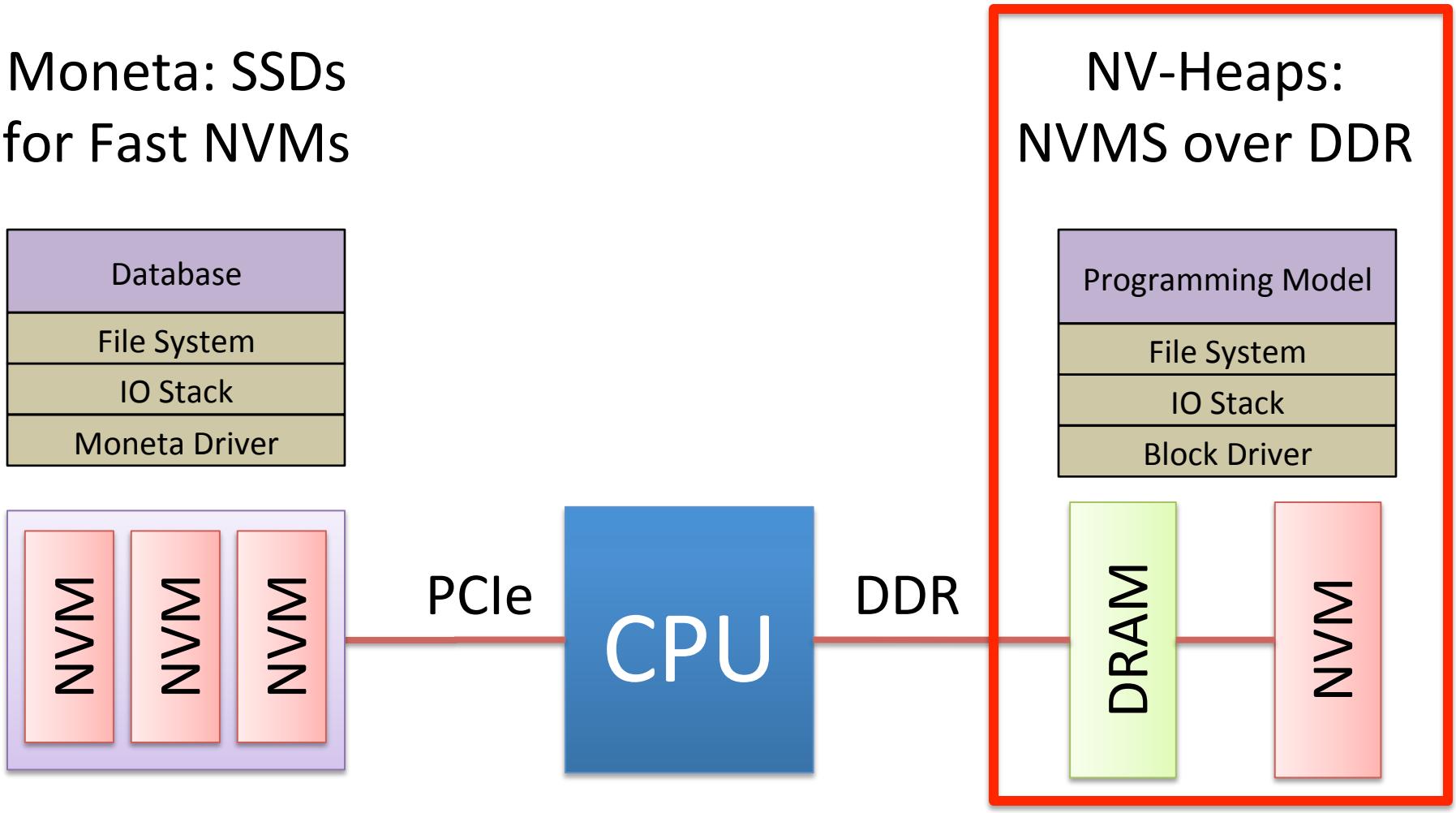


NVSL
Non-volatile Systems Laboratory

Software's Role in Moneta

- The basic Moneta hardware is (relatively) simple
- Software requirements drove the HW enhancements
- Reduce, Remove, Redesign
  - Simplify the kernel IO stack
  - Eliminate the OS and FS overheads for most accesses
  - Rethink write-ahead logging for new memories

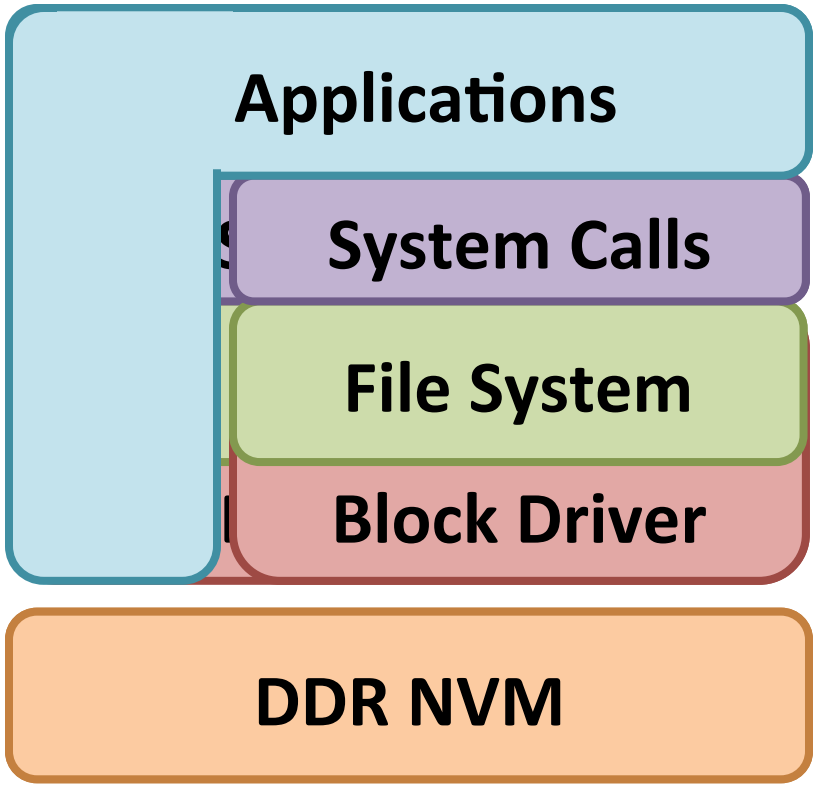
Two Case Studies



Disks and Files Are Passé

DDR NVM is a new kind of Memory

- Use VM to create a file
- Use load/store operations to access data
- Use pointers to data structures
- Leverage strong types
- But make it persistent



The Dangers of Non-Volatile Programming

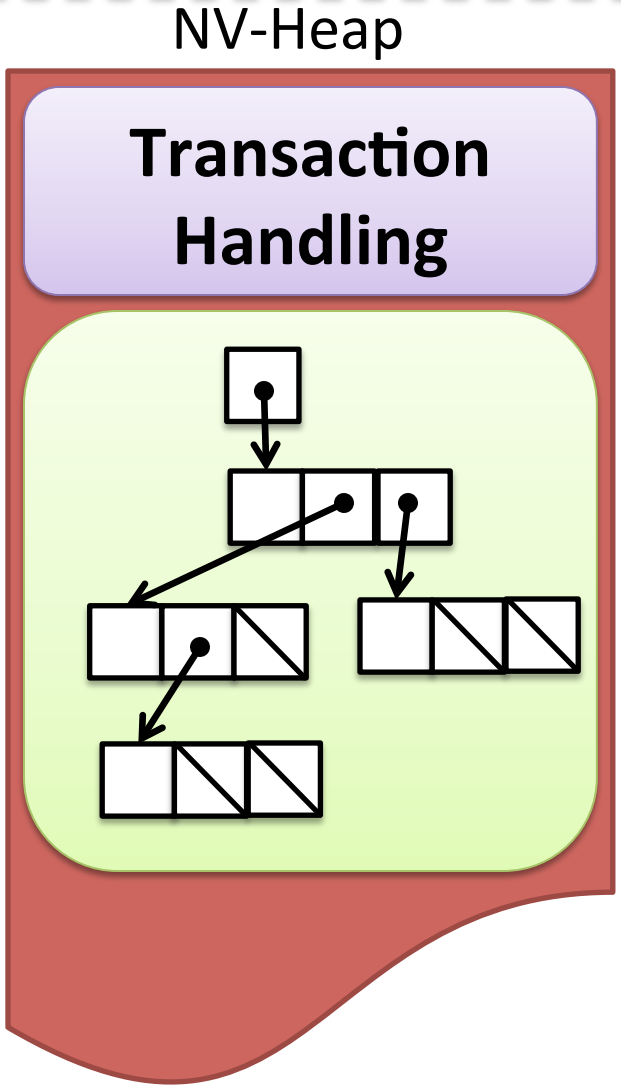
```
Dlist Insert(Foo x, Dlist head)
{
   Dlist r;
   r = new Dlist;
   r.data = x;
   r.next = head;
   r.prev = null;
   if (head) { head.prev = r; }
   return r;
}
```

What if the system crashes?

What if x and head are in different files?

NV-Heaps: Safe, Fast, Persistent Objects

- Container for NV data structures
  - Generic
  - Self-contained
- Access via loads and stores
  - Memory-like performance
  - Familiar semantics
- Strong safety guarantees
  - ACID transactions
  - Pointer safety

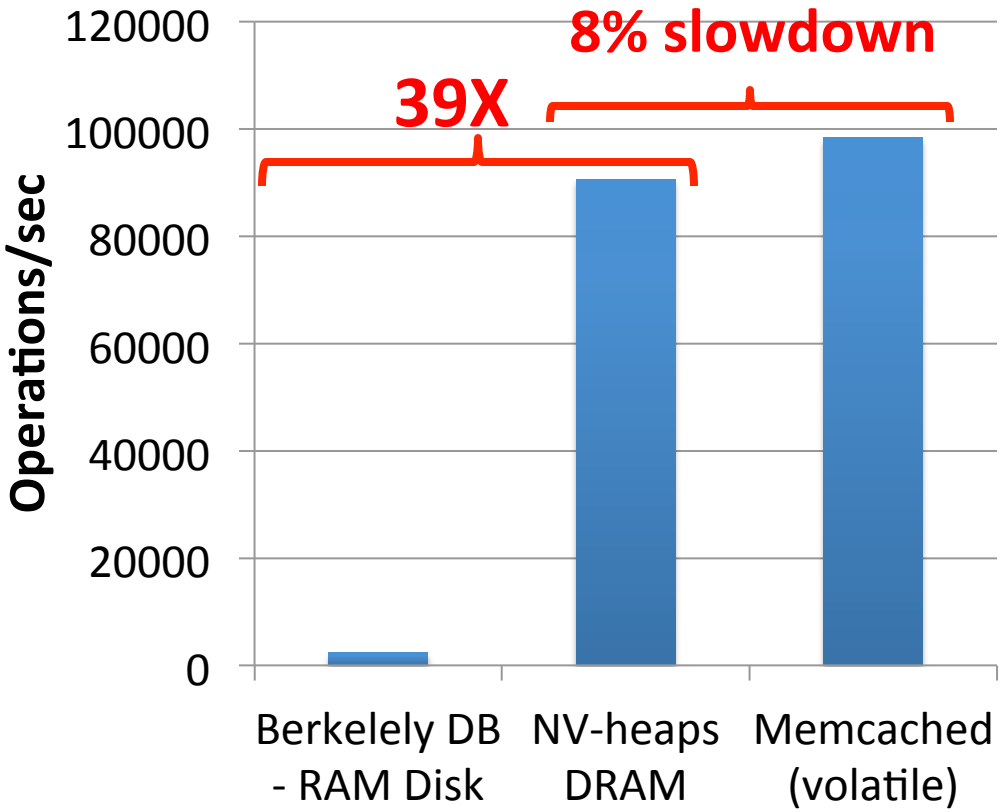


A Type System for NV-Heaps

1. Refine each type with a *heap-index*
2. Type rules ensure that the heaps match

```
Dlist<h> Insert(Foo<h> x, Dlist<h> head)
{
   atomic<h> {
      Dlist<h> r;
      r = new Dlist in heapof(r);
      r.data = x;
      r.next = head;
      r.prev = null in heapof(r);
      if (head) { head.prev = r; }
   }
   return r;
}
```

Application:
A Persistent Key-value Store



NV-heaps are much faster than the block-based systems

NV-heaps make the cost of persistence very low

NVSL
Non-volatile Systems Laboratory

NV-Heaps Wrap Up

- The hardware is here already.
- NV-heaps can change how programmers think about persistent state.
  - Careful system design can avoid the pitfalls
  - And the performance gains are huge

Other Efforts

- PCIe SSDs
  - FusionIO et. al.
  - Intel NVMExpress prototypes [FAST'12]
  - Texas Memory Systems SSDs
- Atomic Writes
  - Transactional Flash [OSDI'08]
  - FusionIO AtomicWrite [HPCA'11]
- NV-Heaps
  - Slow but safe
    - Stasis [OSDI 06]
    - Orthogonally persistent Java [SIGMOD 96]
    - Many others…
  - Fast but unsafe
    - Recoverable Virtual Memory [SOSP 93]
    - Rio Vista [SOSP 97]
  - Fast and safe
    - Mnemosyne [ASPLOS 11]

Open Challenges

How Can We Elegantly Enrich Storage Semantics?

- Atomic Writes help but,…
- Each applications has its own needs
  - Legacy system architectures
  - Novel data structures
- Can we build flexible semantics for storage?
  - Like shaders in graphics cards?

How do NVMs affect the data center?

- How should we organize a petabyte of NVMs?
  - Centralized? Distributed? Hybrid?
- NVM performance with large-scale system overheads
  - Replication
  - Network latency
  - Thick software layers (e.g. OpenStack, HadoopFS)

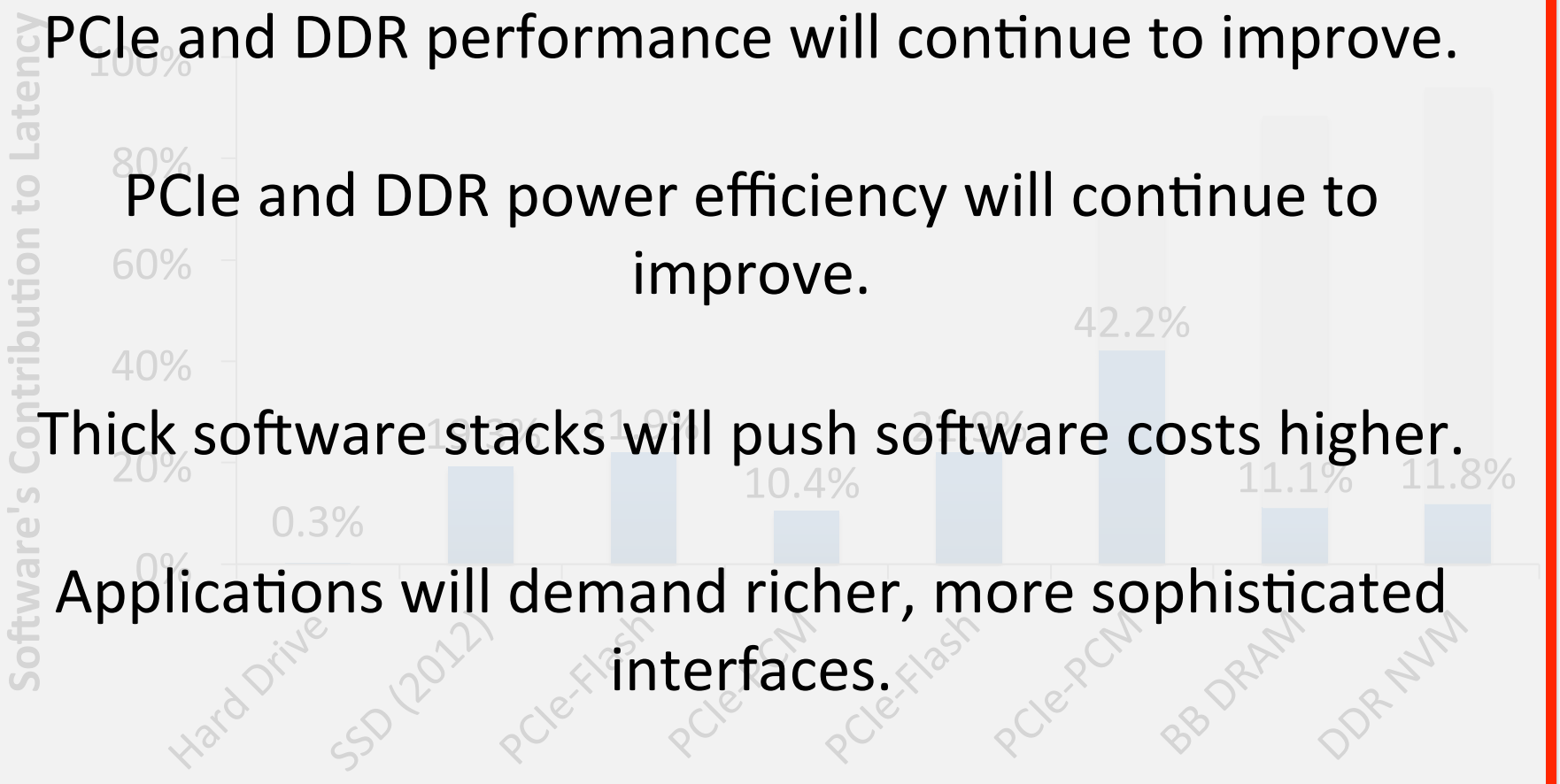
Software Costs Will Remain High

PCIe and DDR performance will continue to improve.

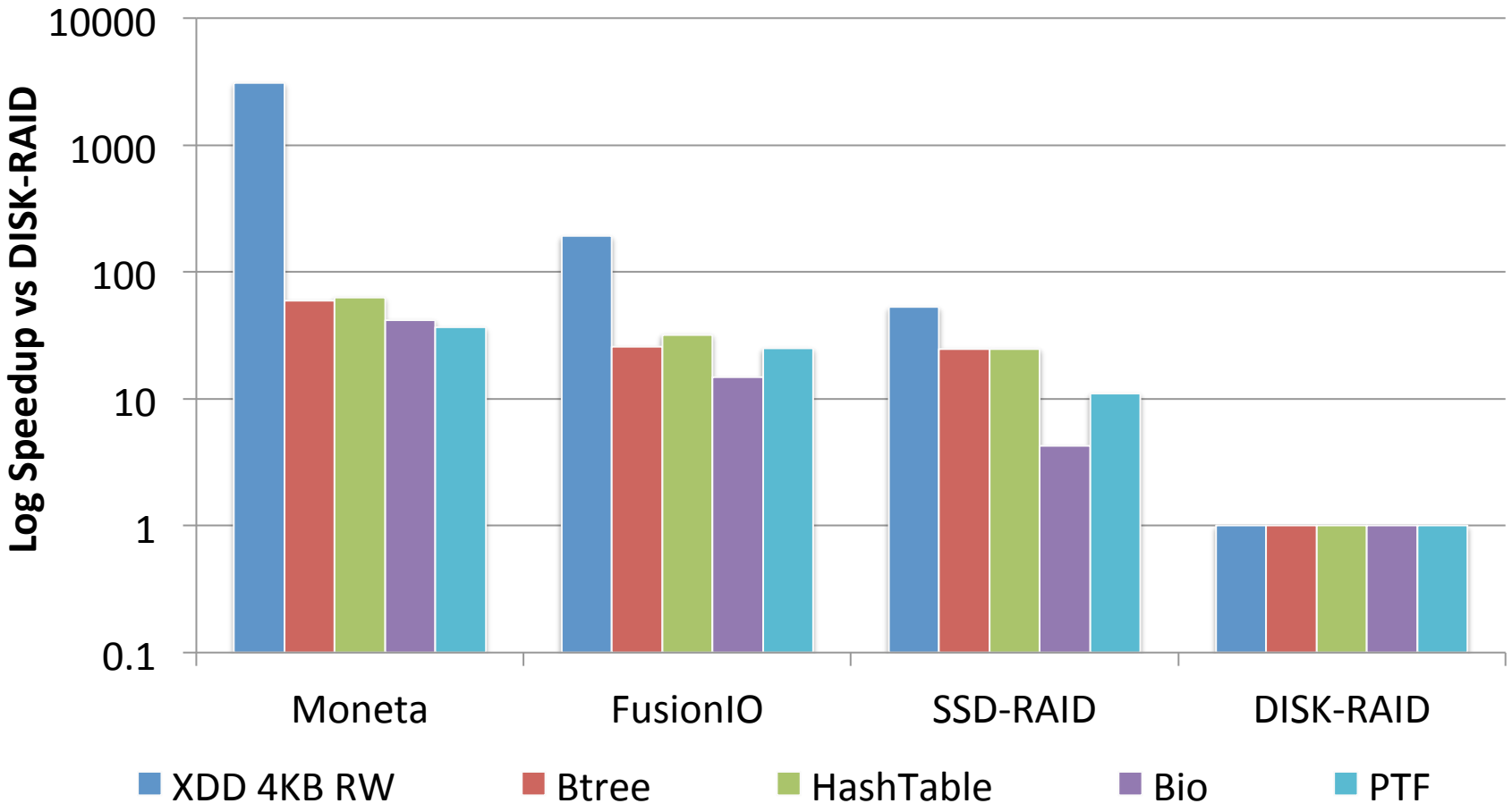
PCIe and DDR power efficiency will continue to improve.

Thick software stacks will push software costs higher.

Applications will demand richer, more sophisticated interfaces.



The App Gap Remains



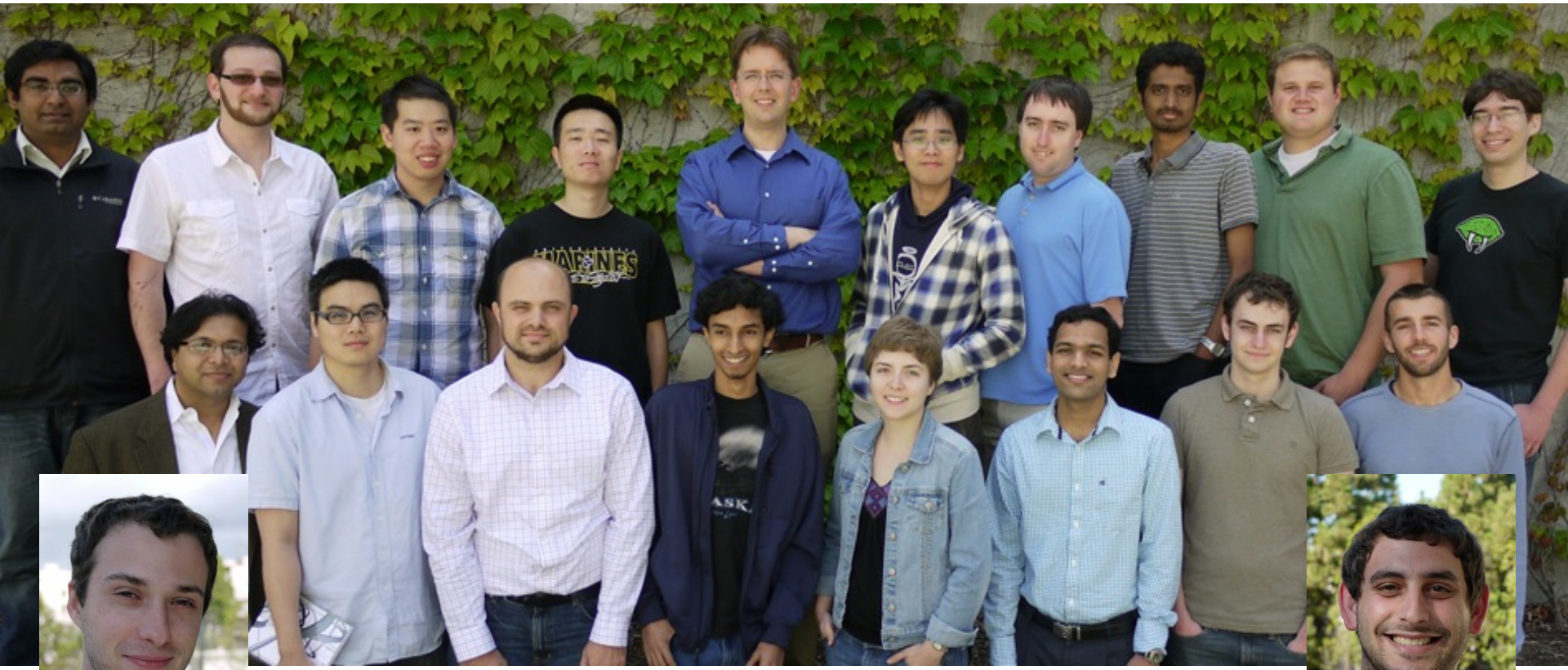
Solid State Storage In the Data Age

- Harnessing the data we collect requires vastly faster storage systems
- Fast, non-volatile memories can provide it, but…
- NVMs will force software's role to change
  - Software will become a drag on IO performance
  - Reduce, Remove, Redesign
- Leveraging NVMs quickly requires a coordinated research that spans system layers
- No aspect of the system is safe!

NVSL
Non-volatile Systems Laboratory

UCSDCSE
Computer Science and Engineering

Thanks!



NVSL
Non-volatile Systems Laboratory


SDSC
SAN DIEGO SUPERCOMPUTER CENTER


CMRR
Center for Magnetic Recording Research


HGST
a Western Digital company


WD
Western
Digital


NSF


it2
Calit2


Google


CISCO


Micron


DEPARTMENT OF DEFENSE
UNITED STATES OF AMERICA


BEEcube


XILINX

Questions?