FLASH IN THE DATACENTER

Nisha Talagala

NON VOLATILE MEMORY

Flash

- 100s GB to 10 TB per PCIe device
- Media trend – increase in density, reduction of write cycles, SLC/MLC/3BPC
- 100s of thousands to millions of IOPS, GB/s of bandwidth

PCM/MRAM/STT/Other NVMs

- Still in research
- Potential of extreme performance increase

HOW TO EFFECTIVELY USE FLASH?

FUSION-IO

Performance

- Closer to CPU – highest bandwidth, lowest latency
- Server (compute) side flash complements storage side flash

Hierarchy of DRAM, flash, disk

Disk displacement usages

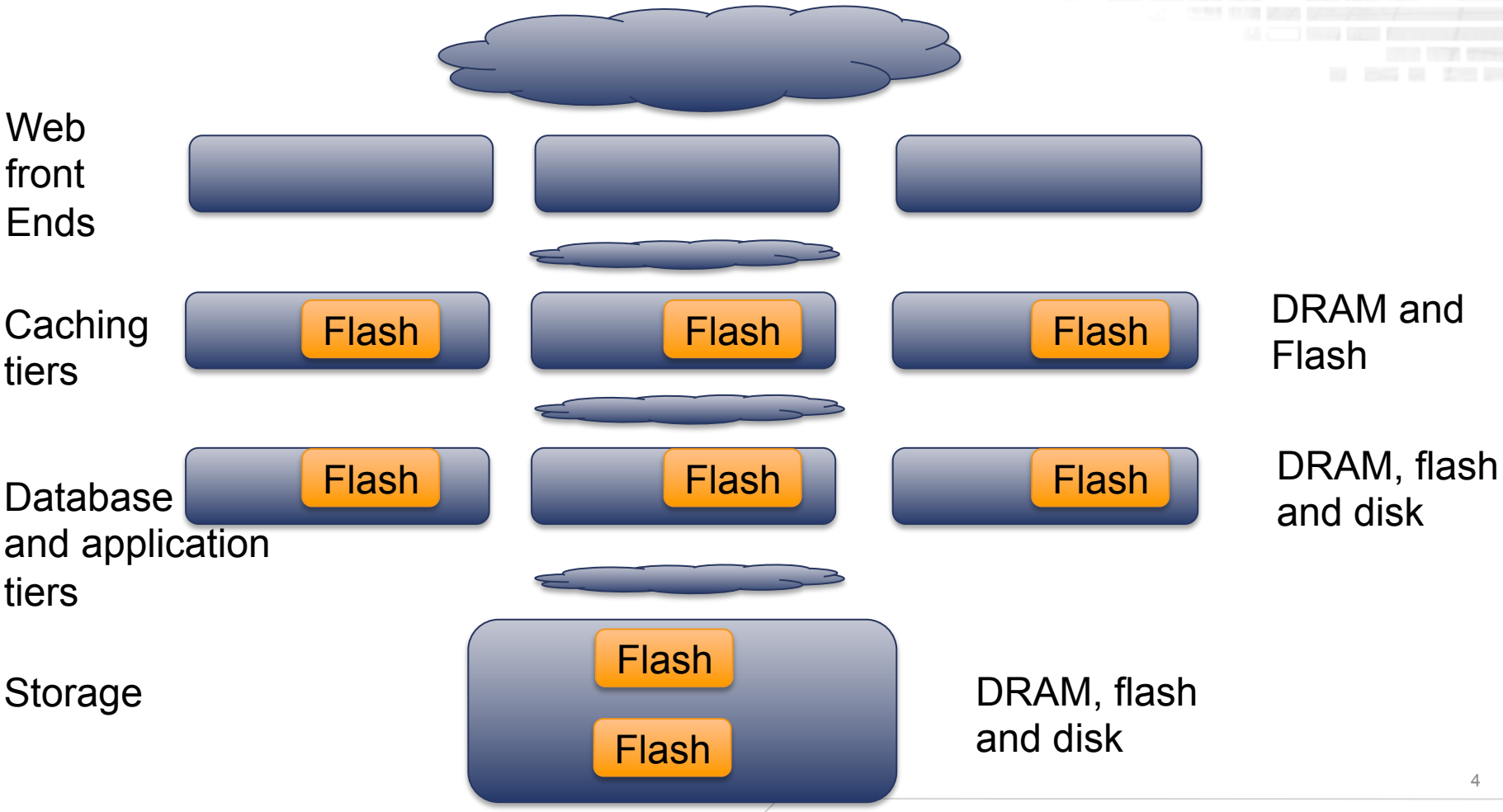
- Caches – server and storage side
- Scale out and cluster file systems
  - flash in metadata server
  - storage server
- Staging, checkpoint

DRAM displacement usages

- Improved paging, semi-external memory

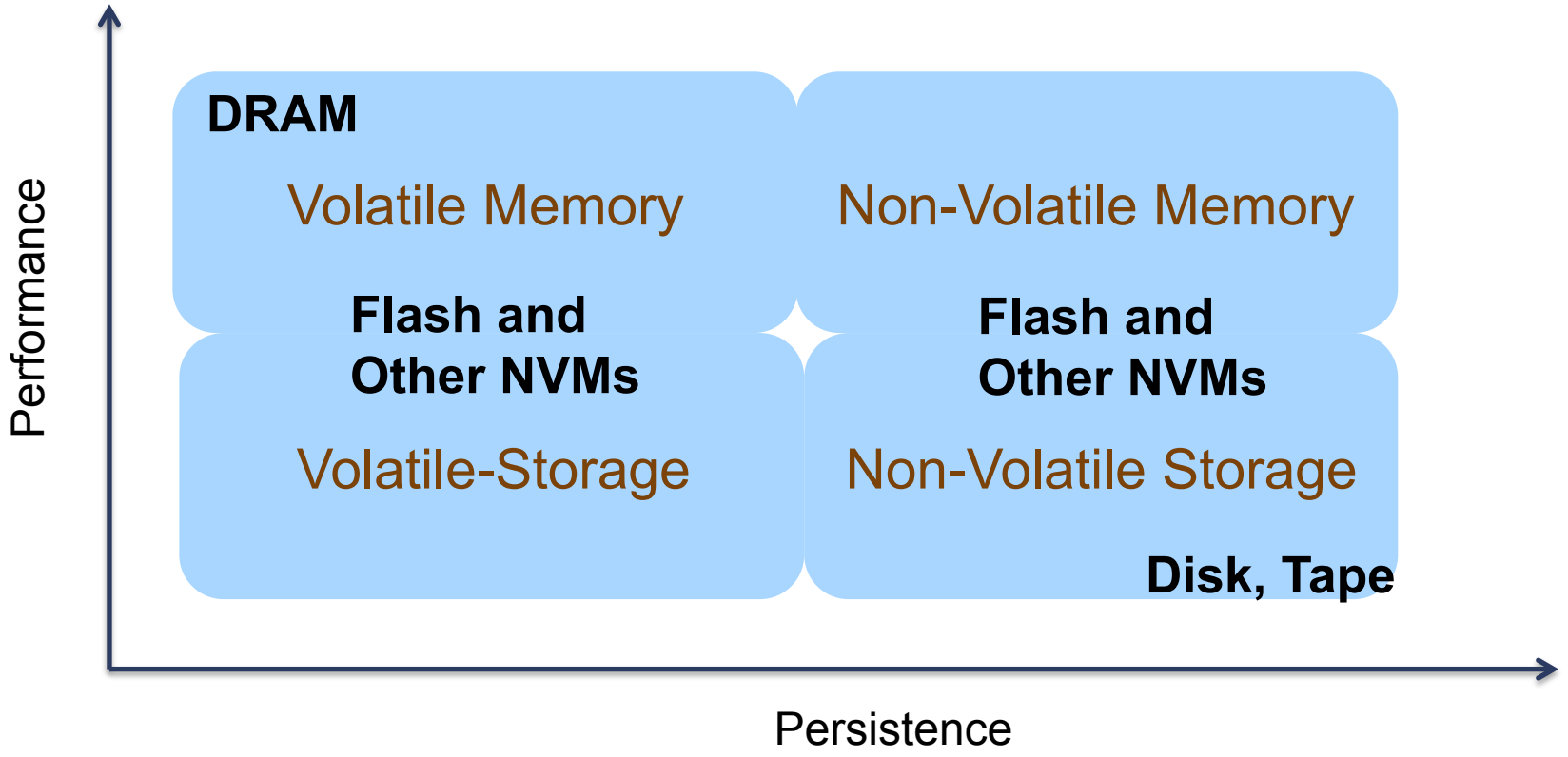
NVM IN THE DATA CENTER TODAY

FUSION-IO



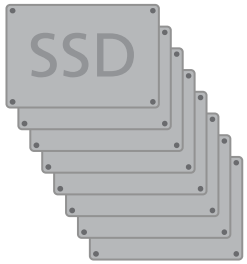
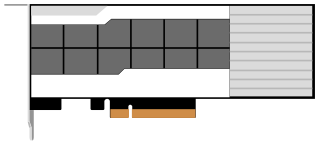
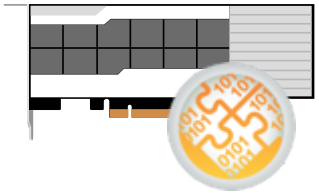
MEMORY STORAGE CONVERGENCE

FUSION-IO



EVOLUTION OF ENTERPRISE FLASH

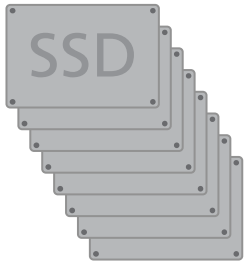
FUSION-IO

FLASH AS DISK	FLASH BEYOND DISK	FLASH AS MEMORY
		
Application	Application	Application
Application source code converts native data structures into block I/O	Application source code does I/O with native data structures	Application source code manipulates native memory data structures
Conventional I/O Access	Enhanced I/O	Memory Access
Block I/O	**Atomic I/O Transaction** / **Native primitives** / **Key-Value Transaction**	**Extended Memory** / **Persistent Memory**
NVM Devices/Media	NVM Devices/Media	NVM Devices/Media

EVOLUTION OF ENTERPRISE FLASH

FUSION-IO

FLASH AS DISK



Application

Application source code converts native data structures into block I/O

Conventional I/O Access

Block I/O

NVM Devices/Media

FLASH BEYOND DISK

Application

Application source code does I/O with native data structures

Enhanced I/O

Atomic I/O Transaction | Native primitives | Key-Value Transaction

NVM Devices/Media

FLASH AS MEMORY

Application

Application source code manipulates native memory data structures

Memory Access

Extended Memory | Persistent Memory

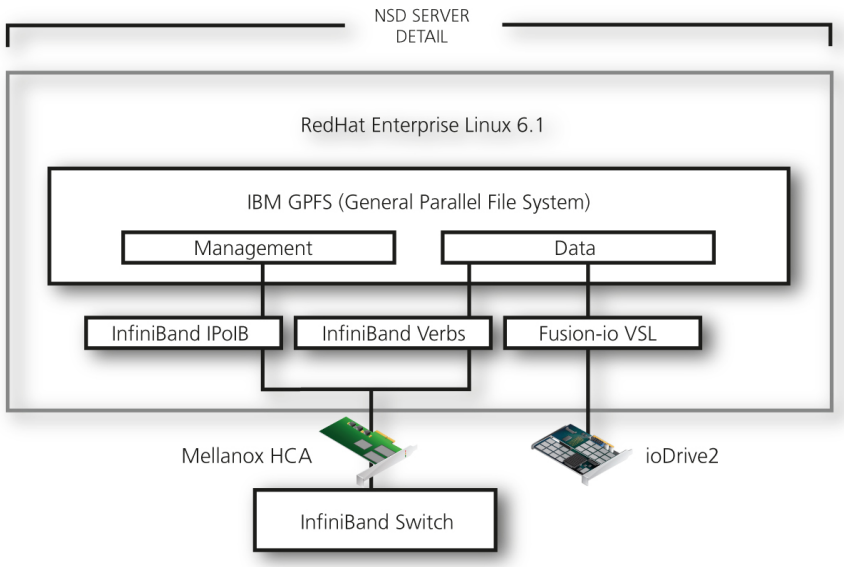
NVM Devices/Media

SC11 GENERAL PARALLEL FILE SYSTEM (GPFS) DEMO

FUSION-IO

- 24 uncompressed 1080p videos (up to 6 GB/s of data)
- Five Fusion Powered GPFS-based NSD servers
- Three visualization workstations

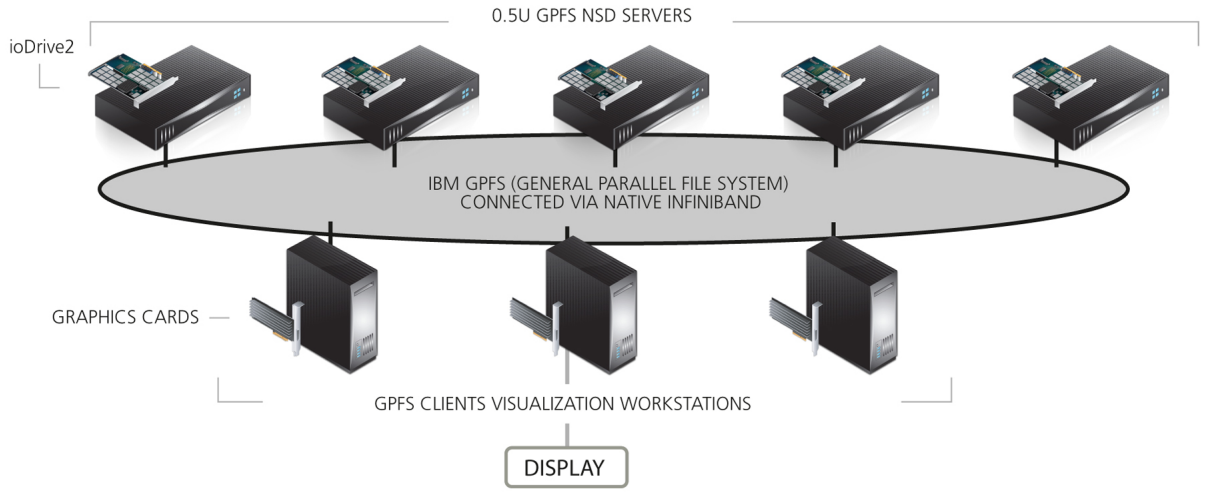
DEMO SOFTWARE SPECS



- RedHat Enterprise Linux 6.1 - InfiniBand Software Stack
- IBM GPFS (General Parallel File System) 3.4.0.8
- NVIDIA Linux Driver
- Fusion-io VSL 3

DEMO HARDWARE SPECS

FUSION-IO

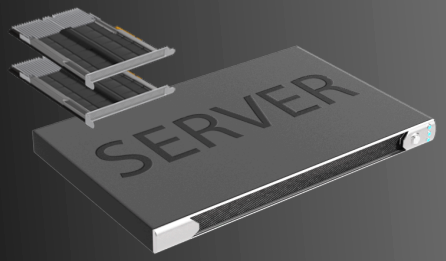


- Five 0.5U NSD Servers, each with six core dual socket CPUs, 12 GB RAM, InfiniBand HCA, and an ioDrive2
- 3 Visualization workstations, with an InfiniBand HCA and an NVIDIA graphics card
- 36-port QDR InfiniBand switch

ioN

Software

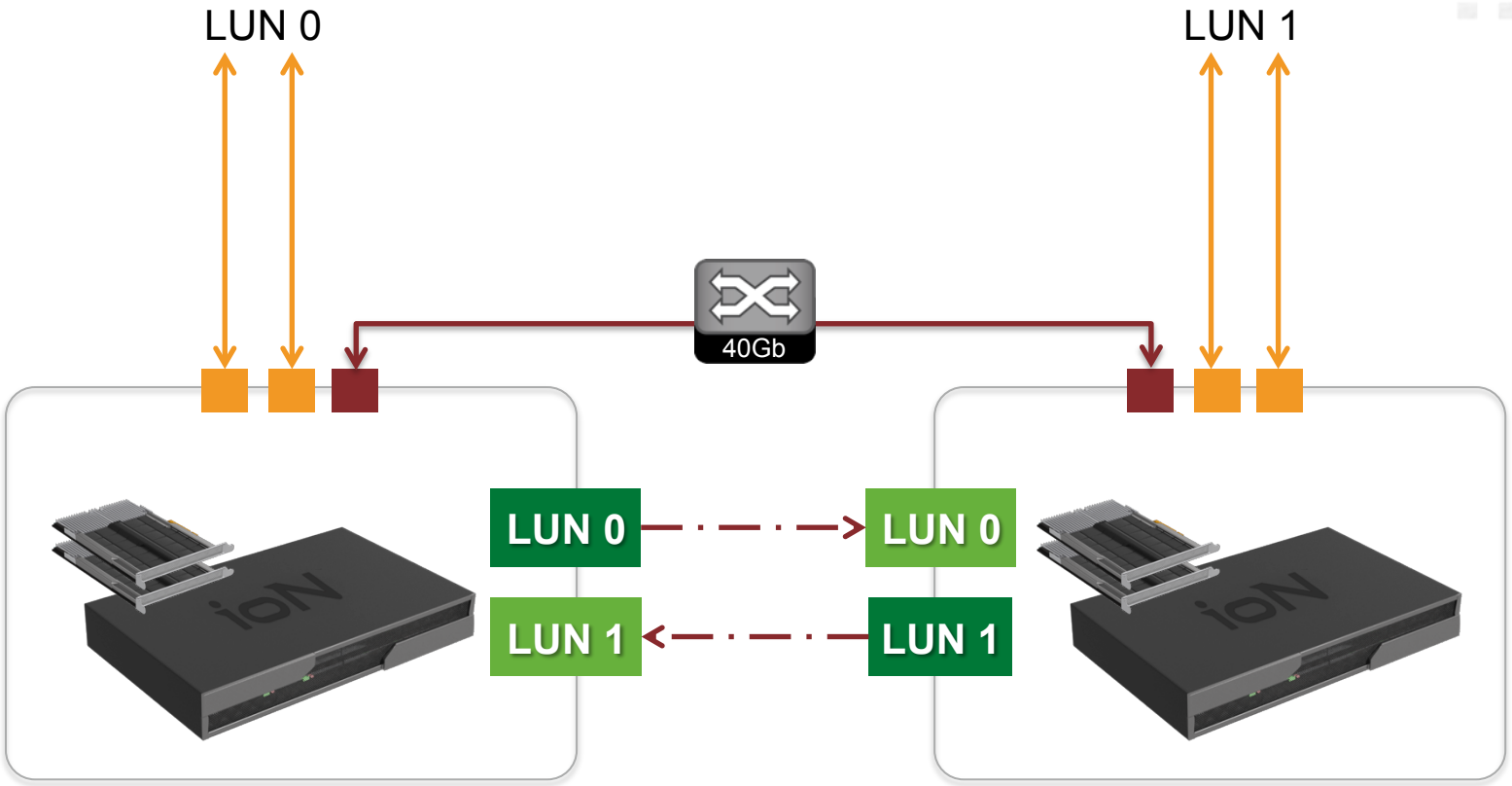
+



Your Server
Becomes a Shared
Flash Storage
Appliance

ION DATA ACCELERATOR – HIGH AVAILABILITY

FUSION-IO



ION – FREEDOM OF CHOICE

FUSION-IO

SOFTWARE

HP, IBM, DELL, CISCO, FUSION-IO

- Leverage your buying power
- Integrate
- Support
  - ION Software (via Fusion-io)
  - Server (via server OEM)
  - ioDrive (via your supplier)

FULLY INTEGRATED SOLUTION

HP, SUPERMICRO

- No hassles, partner integrated
- Support
  - End-to-End Fusion-io Support

EVOLUTION OF ENTERPRISE FLASH

FUSION-IO

FLASH AS DISK

Application

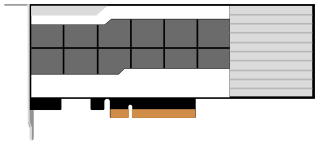
Application source code converts native data structures into block I/O

Conventional I/O Access

Block I/O

NVM Devices/Media

FLASH BEYOND DISK



Application

Application source code does I/O with native data structures

Enhanced I/O

Atomic I/O Transaction | Native primitives | Key-Value Transaction

NVM Devices/Media

FLASH AS MEMORY

Application

Application source code manipulates native memory data structures

Memory Access

Extended Memory | Persistent Memory

NVM Devices/Media

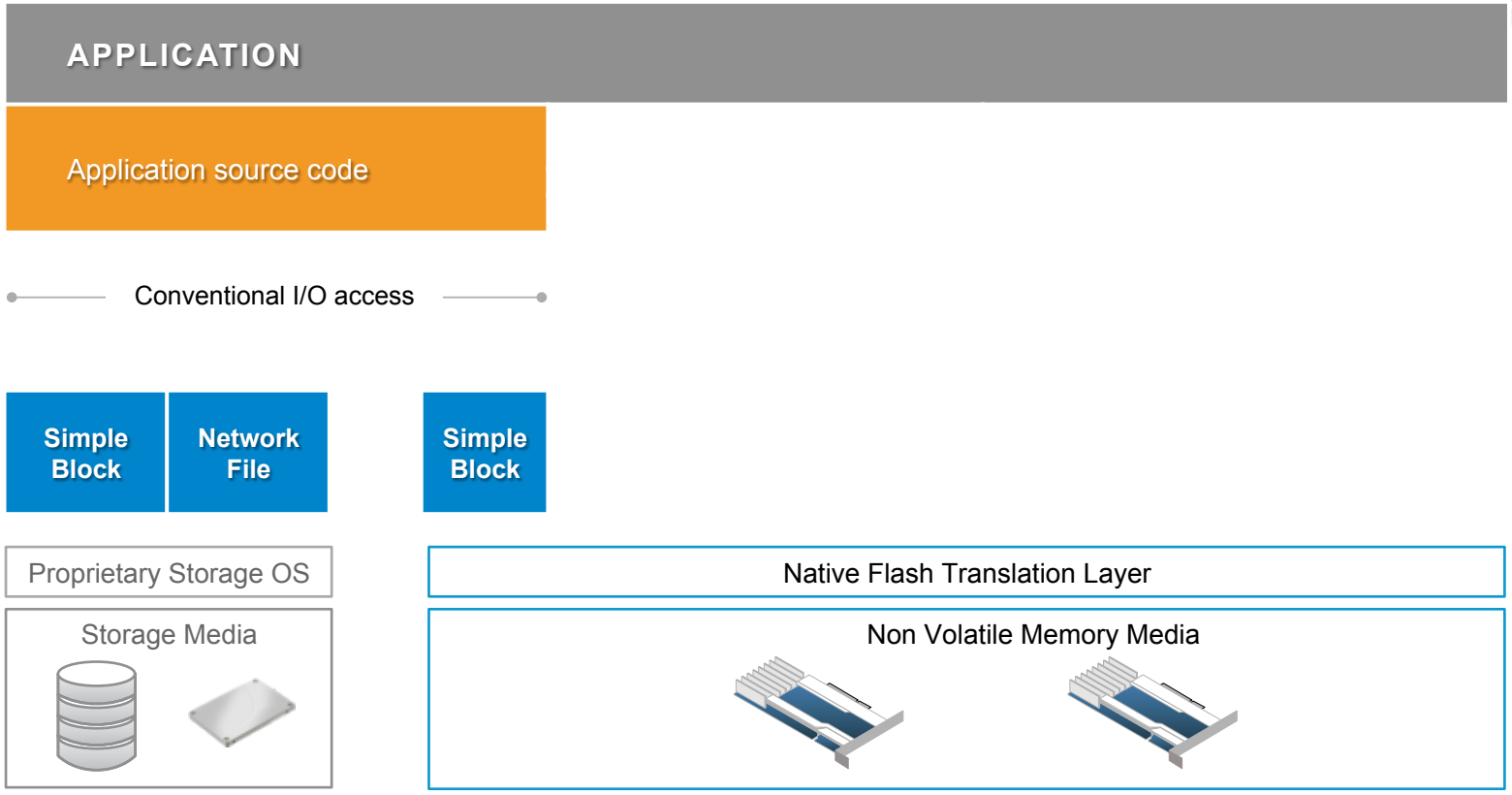
NVM (FLASH, OTHER) IS DIFFERENT FROM DISK

FUSION-IO

Area	Hard Disk Drives	Flash Devices
Logical to Physical Blocks	Nearly 1:1 Mapping	Remapped at every write
Read/Write Performance	Largely symmetrical	Heavily asymmetrical. Additional operation (erase)
Sequential vs Random Performance	100x difference. Elevator scheduling for disk arm	<10x difference. No disk arm – NAND die
Background operations	Rarely impact foreground	Regular occurrence. If unmanaged - can impact foreground
Wear out	Largely unlimited writes	Limited writes
IOPS	100s to 1000s	100Ks to Millions
Latency	10s ms	10s-100s us

CONVENTIONAL I/O ACCESS

FUSION-IO



MULTI-QUEUE I/O IN LINUX*

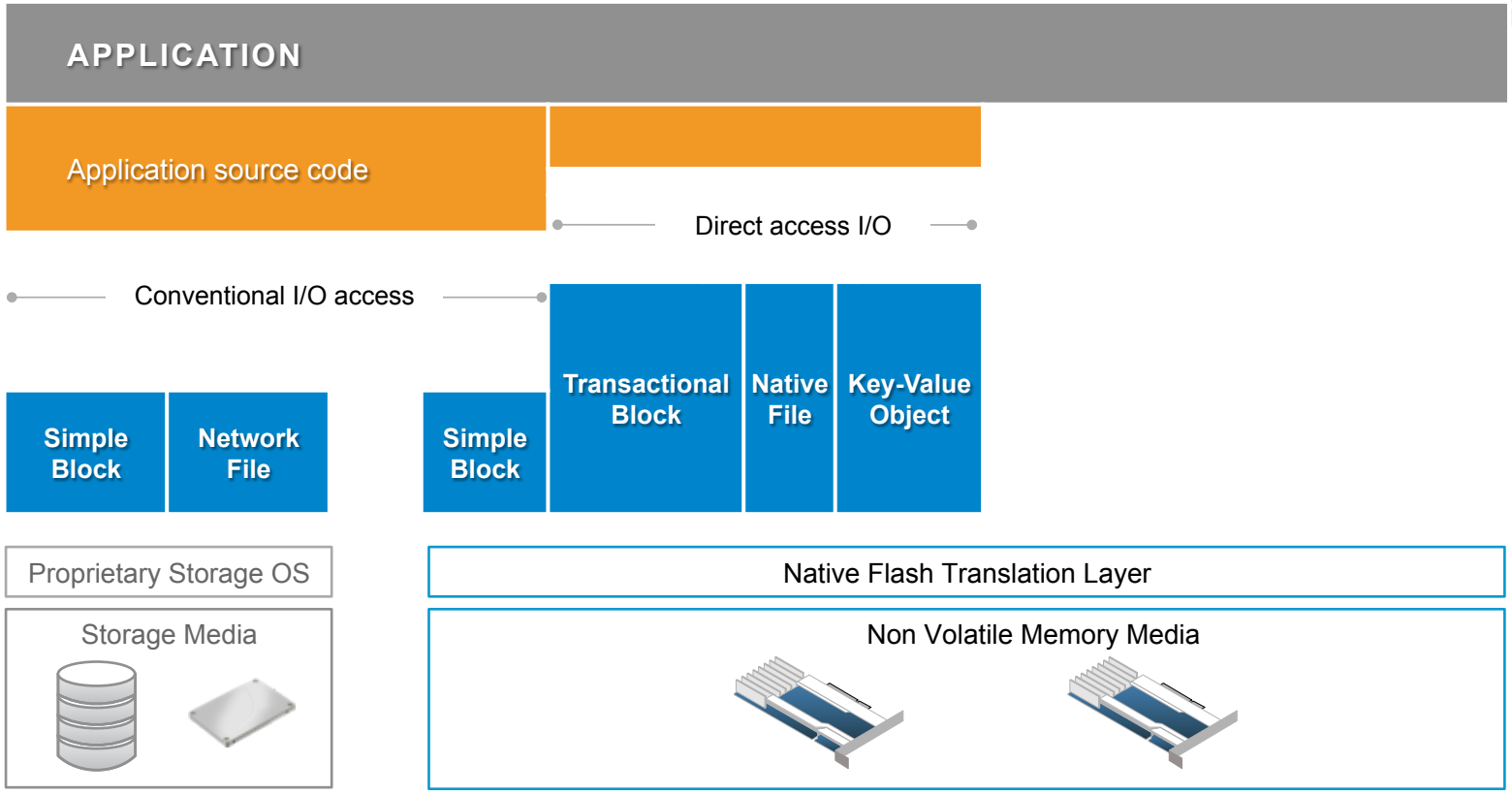
FUSION-IO

- Extending Linux block I/O to support NVM performance
- Multi-queue
  - Software queues, Hardware queues
  - Per CPU issue/completion, multi-socket scaling
  - Matches inherent parallelism in NVM devices and CPUs
  - Supports upcoming queue oriented standards models
- Performance
  - 3.5x – 10x increase in IOPS (from ~1M to 3.5-10M)
  - 10x – 38x reduction in I/O stack latency

*Linux Block I/O: Introducing Multiqueue SSD Access on Multicore Systems
Bjorling M., Axboe J., Nellans D., Bonnett P.
SYSTOR 2013
University of Copenhagen and Fusion-io

DIRECT-ACCESS I/O THROUGH NATIVE INTERFACES

FUSION-IO



FLASH PRIMITIVES: SAMPLE USES AND BENEFITS

FUSION-IO

Databases

Transactional Atomicity:
Replace various workarounds implemented in database code to provide write atomicity (MySQL double-buffered writes, etc.)

Filesystems

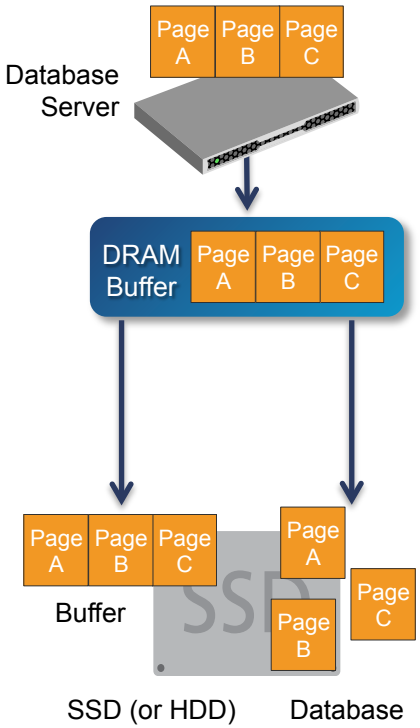
File Update Atomicity:
Replace various workarounds implemented in filesystem code to provide file/directory update atomicity (journaling, etc.)

- **98% performance of raw writes**
  Smarter media now natively understands atomic updates, with no additional metadata overhead.
- **2x longer flash media life**
  Atomic Writes can increase the life of flash media up to 2x due to reduction in write-ahead-logging and double-write buffering.
- **50% less code in key modules**
  Atomic operations dramatically reduce application logic, such as journaling, built as work-arounds.

ATOMIC WRITES – MYSQL EXAMPLE

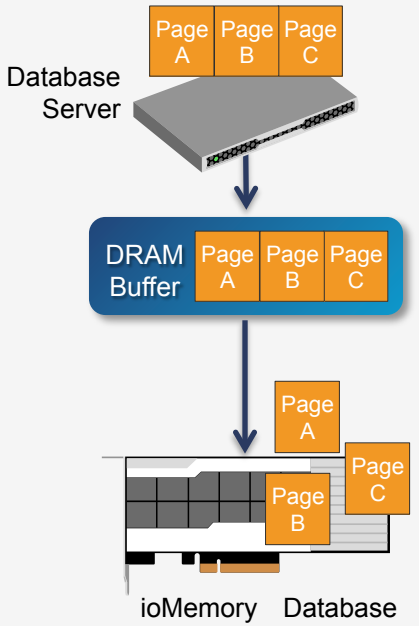
FUSION-IO

Traditional MySQL Writes



1. Application initiates updates to pages A, B, and C.
2. MySQL copies updated pages to memory buffer.
3. MySQL writes to double-write buffer on the media.
4. Once step 3 is acknowledged, MySQL writes the updates to the actual tablespace.

MySQL with Atomic Writes

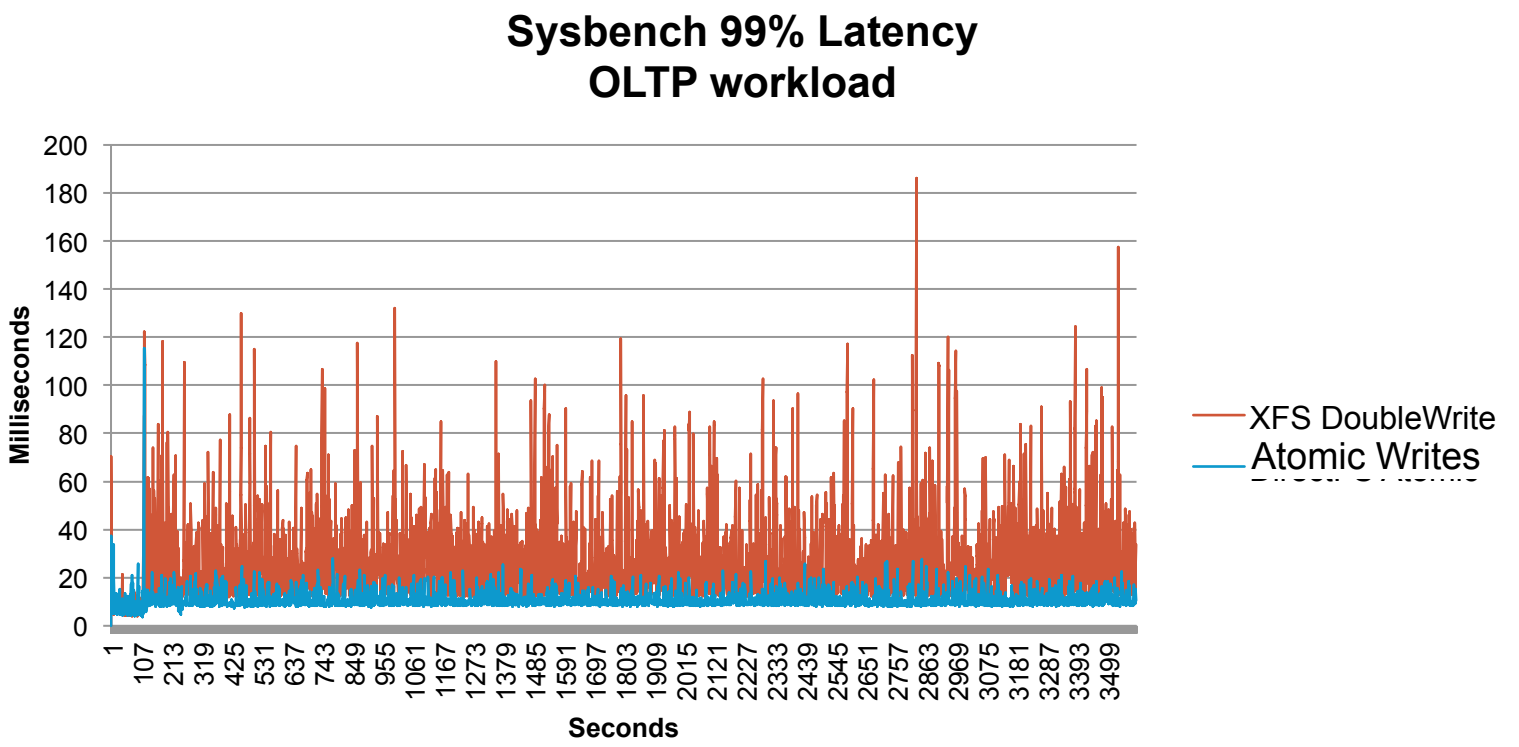


1. Application initiates updates to pages A, B, and C.
2. MySQL copies updated pages to memory buffer.
3. MySQL writes to actual tablespace, bypassing the double-write buffer step due to inherent atomicity guaranteed by the (intelligent) device.

MYSQL EXAMPLE: LATENCY IMPROVEMENT

FUSION-IO

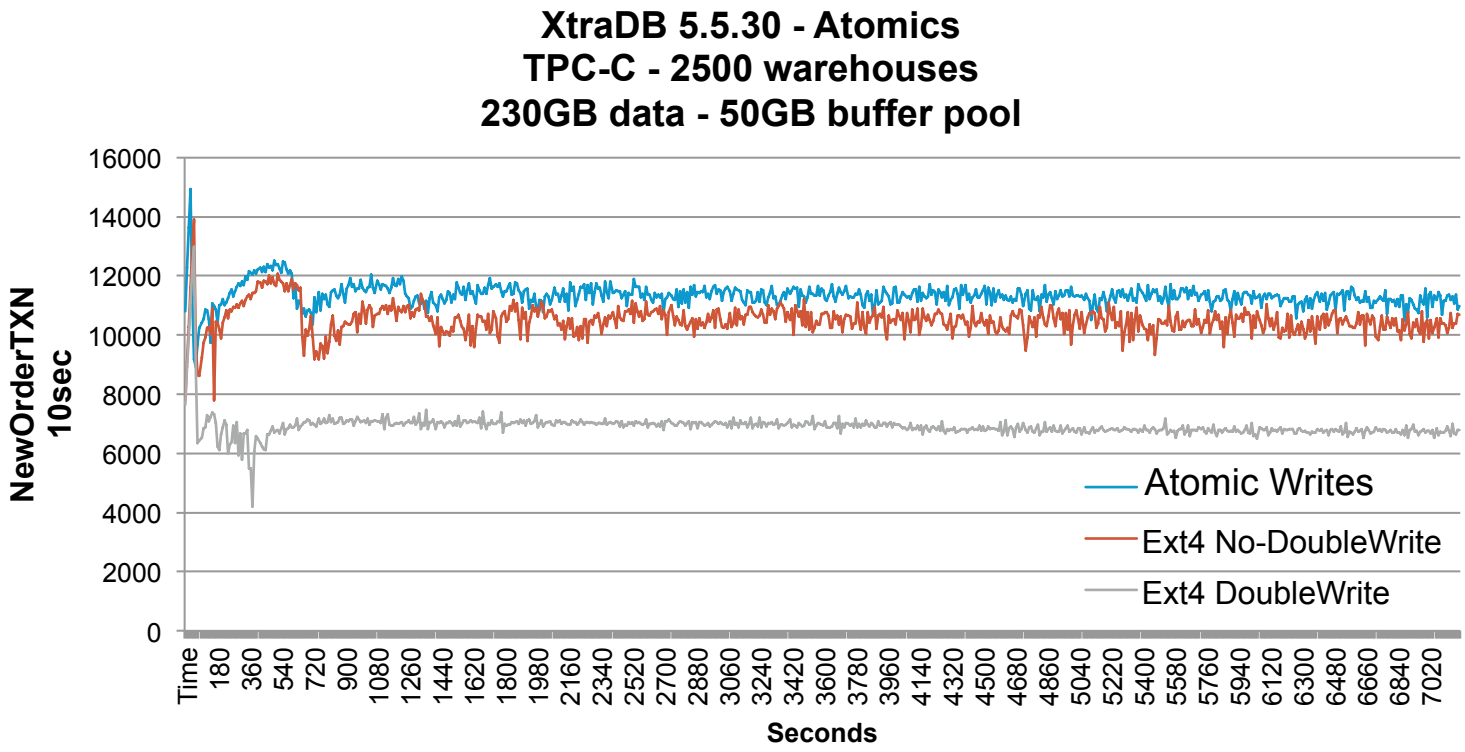
2-4x Latency Improvement on Percona Server



MYSQL EXAMPLE: THROUGHPUT IMPROVEMENT

FUSION-IO

70% Transactions/sec Improvement on MariaDB Server



KEY-VALUE INTERFACE: SAMPLE USES AND BENEFITS

FUSION-IO

NoSQL Applications

Increase performance by eliminating packing and unpacking blocks, defragmentation, and duplicate metadata at app layer.

Reduce application I/O through batched operations.

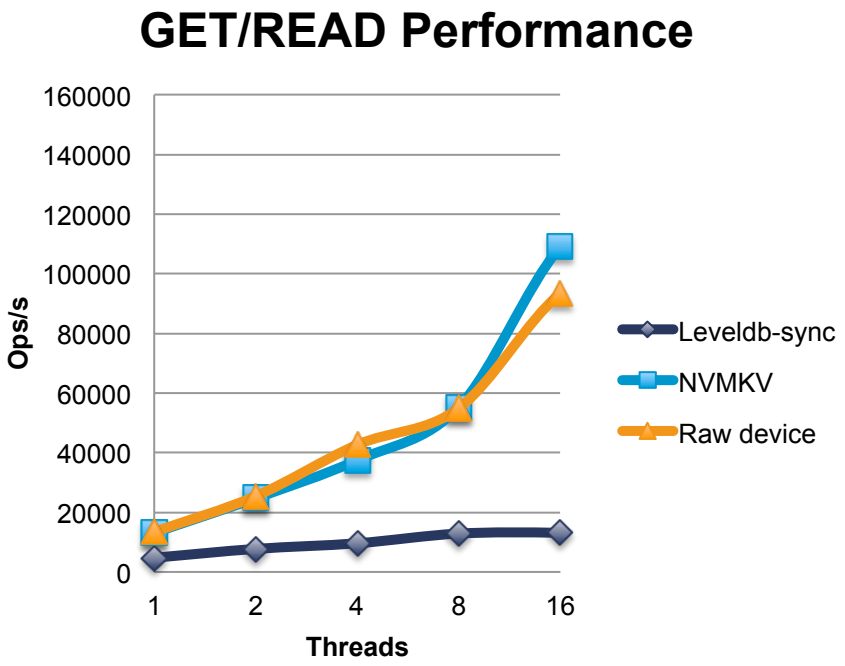
Reduce overprovisioning due to lack of coordination between two-layers of garbage collection (application-layer and flash-layer). Some top NoSQL applications recommend over-provisioning by 3x due to this.

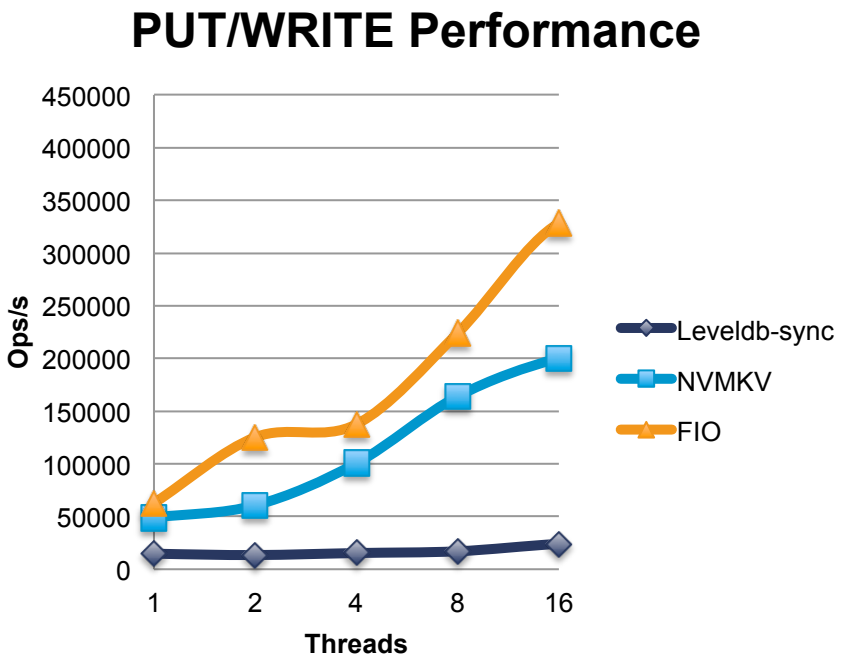
- **Near performance of raw device**
  Smarter media now natively understands a key-value I/O interface with lock-free updates, crash recovery, and no additional metadata overhead.
- **3x throughput on same SSD**
  Early benchmarks comparing against synchronous levelDB show over 3x improvement.
- **Up to 3x capacity increase**
  Dramatically reduces over-provisioning through coordinated garbage collection and automated key expiry.

KEY-VALUE INTERFACE - PERFORMANCE

FUSION-IO

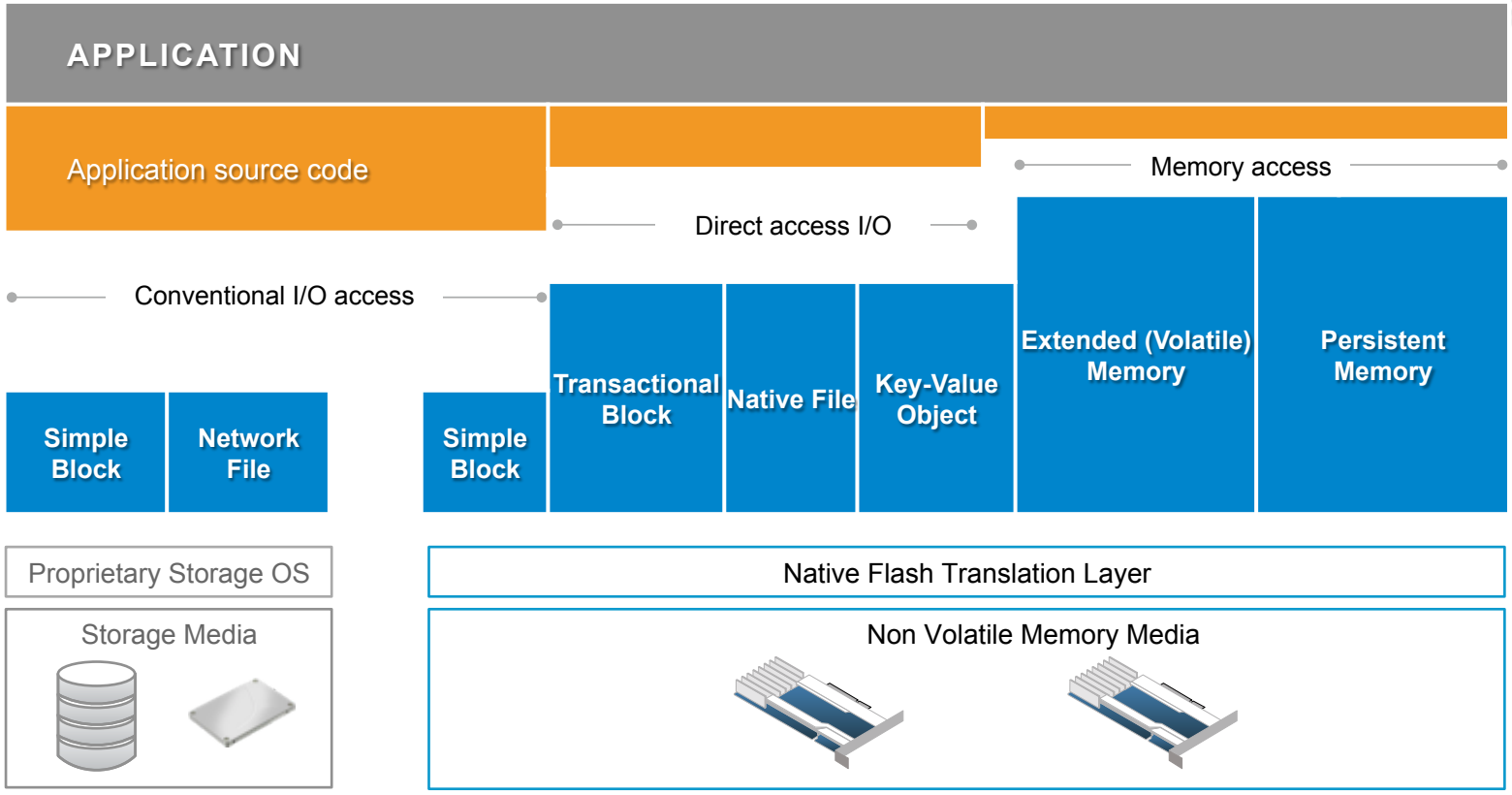
Key-Value get/put vs. Raw read/write vs. levelDB read/write





MEMORY-ACCESS THROUGH NATIVE INTERFACES

FUSION-IO



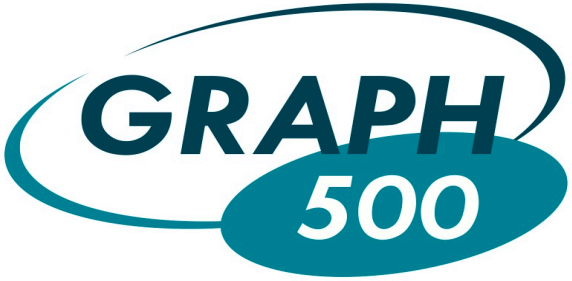
GRAPH500* AND DI-MMAP**

FUSION-IO

- Traversing massive graphs
  - "Using 2.56TB of Fusion-io NAND flash to access data using memory semantics, LLNL's new Graph500 algorithm can process graphs 8x larger than before with only a 50% performance degradation compared to an all DRAM system."
  - Results: 55.6 MTEPS (Million Traversed Edges Per Second)

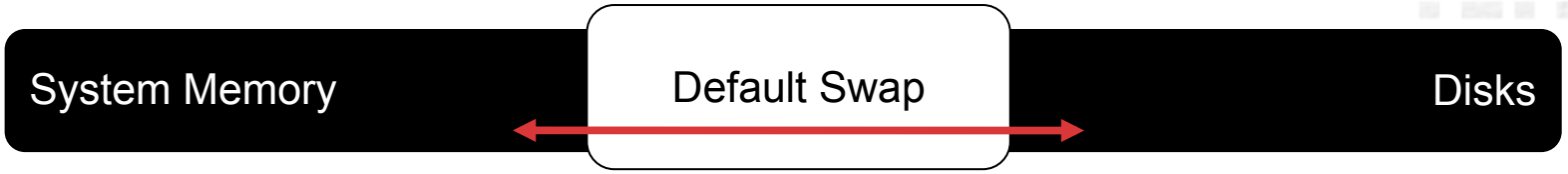
    4 x 640GB Fusion-io MLC
- DI-MMAP: Accelerated mmap for highly concurrent apps
  - 3-5x improvement in mmap performance

* Graph500: Traversing massive graphs with NAND flash; Pearce, Gokhale, & Amato (LLNL)
**DI-MMAP: A High Performance Memory Map Runtime for Data Intensive Applications; Van Essen, Hsieh, Ames, Gokhale (LLNL)



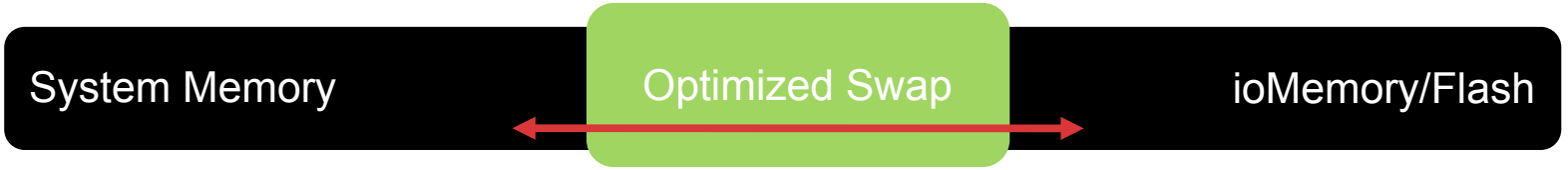
IMPROVING LINUX SWAP (DEMAND-PAGING)

FUSION-IO



Originally designed as a last resort to prevent OOM (out-of-memory) failures

- Never tuned for high-performance demand-paging
- Never tuned for multi-threaded apps
- Poor performance

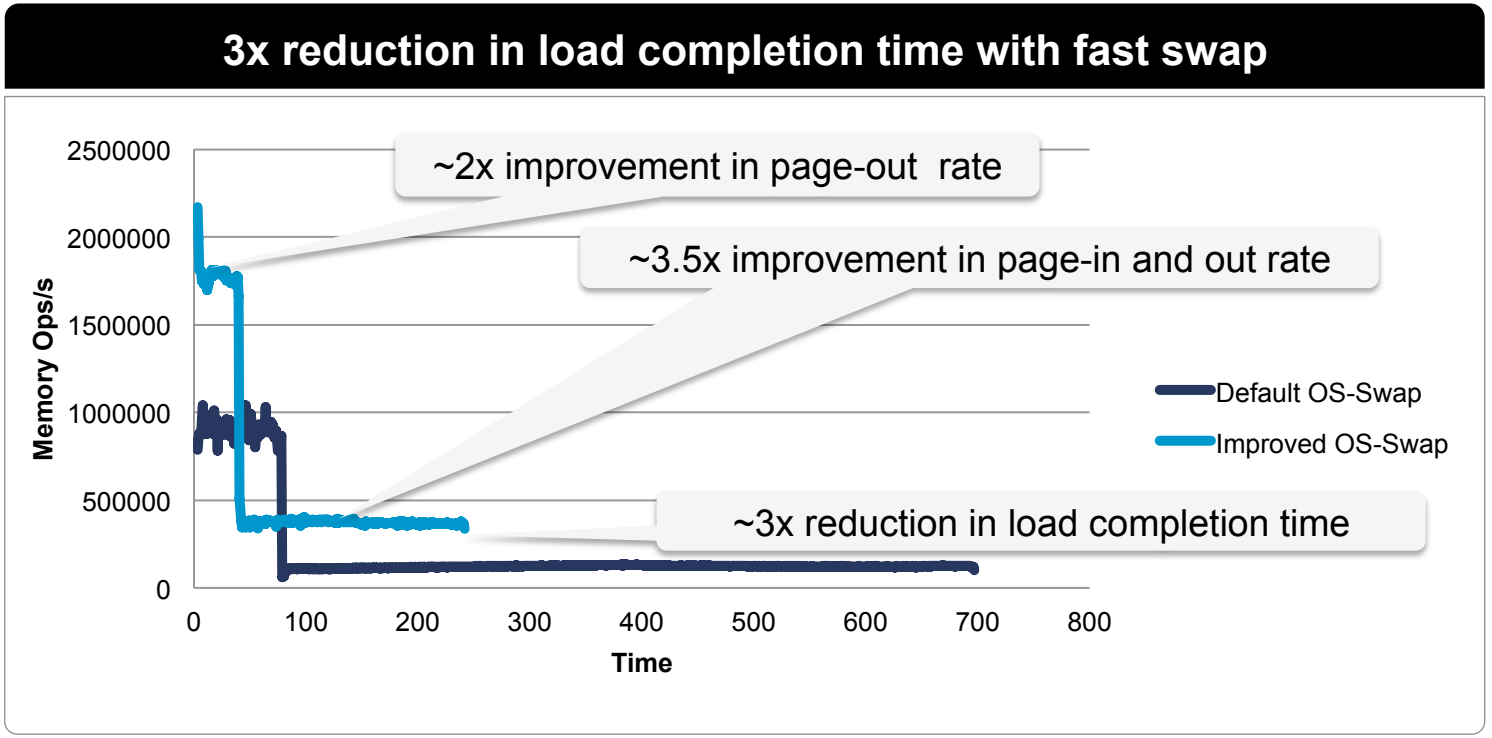


Tuned for flash (leverages native characteristics)

- O(1) algorithm for swap_out – reduce algorithm time and leverage fast random I/O
- Per CPU reclaim – greater throughput for multi-threaded environments
- Intelligent read-ahead on swap-in – cut legacy, disk-era cruft for rotational latency

FAST SWAP - PERFORMANCE

FUSION-IO



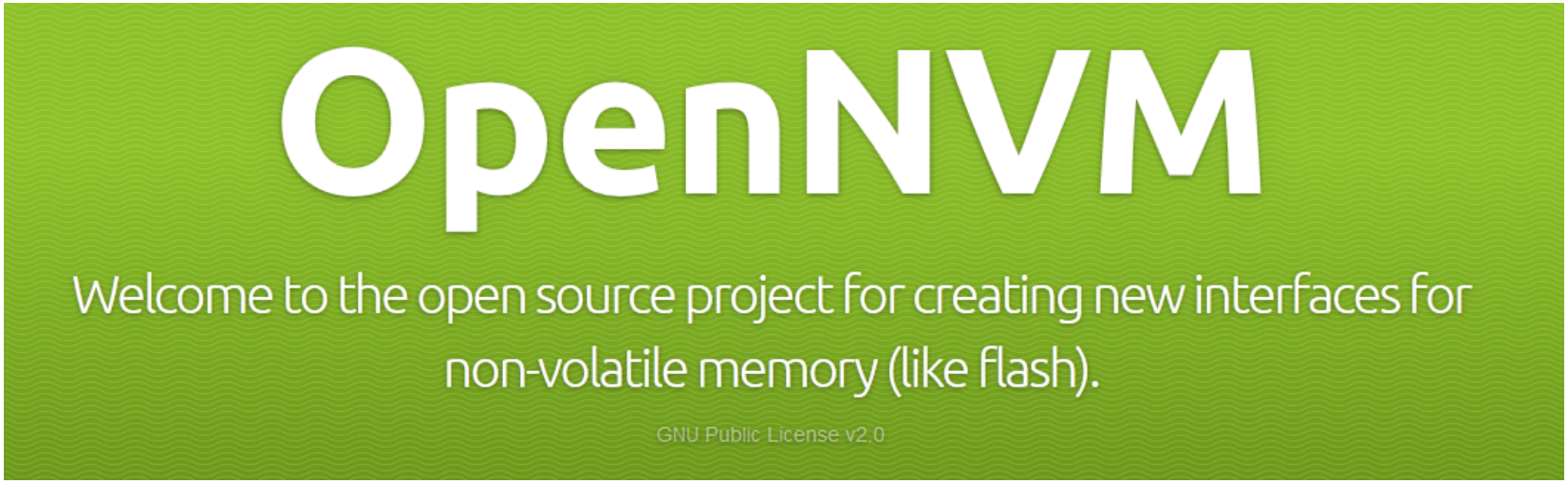
COMPARING I/O AND MEMORY ACCESS SEMANTICS

FUSION-IO

I/O	I/O semantics examples: • Open file descriptor – open(), read(), write(), seek(), close() • (New) Write multiple data blocks atomically, nvm_vectored_write() • (New) Open key-value store – nvm_kv_open(), kv_put(), kv_get(), kv_batch_*()
Memory Access **(Volatile)**	Volatile memory semantics example: • Allocate virtual memory, e.g. malloc() • memcpy/pointer dereference writes (or reads) to memory address • (Improved) Page-faulting transparently loads data from NVM into memory
Memory Access **(Non-Volatile)**	Non-volatile memory semantics example: • (New) Allocate and manage persistent memory

FUSION-IO

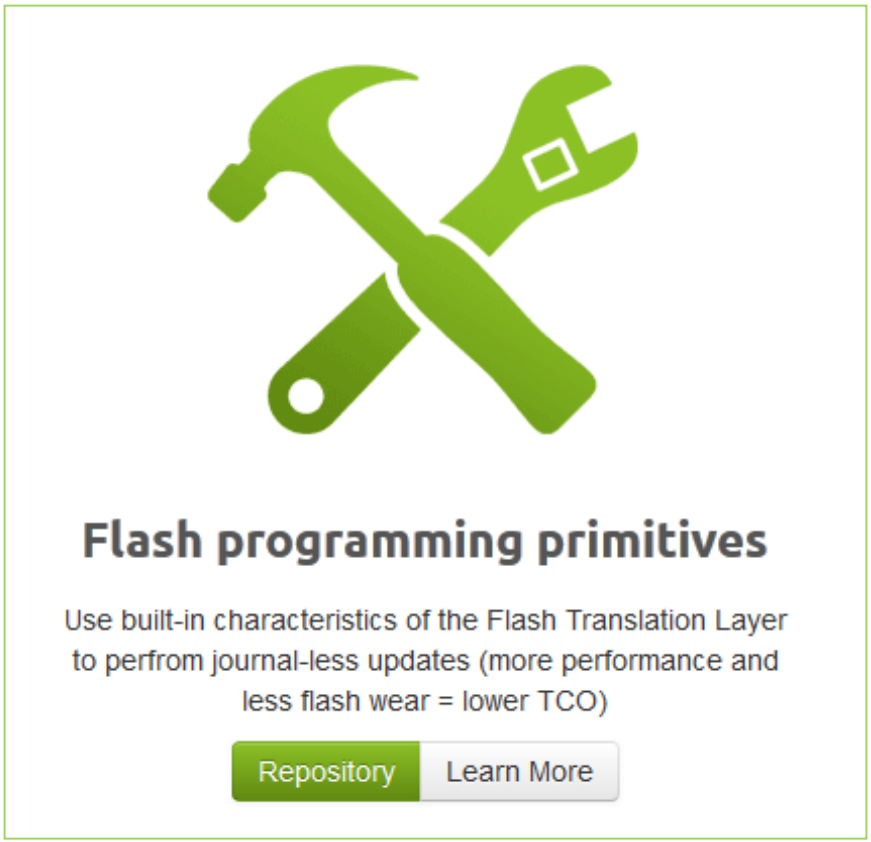
https://opennvm.github.io



http://www.opencompute.org/projects/storage/

1ST CONTRIBUTION: FLASH PRIMITIVES

FUSION-IO



https://opennvm.github.io

On GitHub:

- API specifications, such as:
  - *nvm_atomic_write()*
  - *nvm_batch_atomic_operations()*
  - *nvm_atomic_trim()*
- Sample program code

2ND CONTRIBUTION: LINUX FAST-SWAP

FUSION-IO

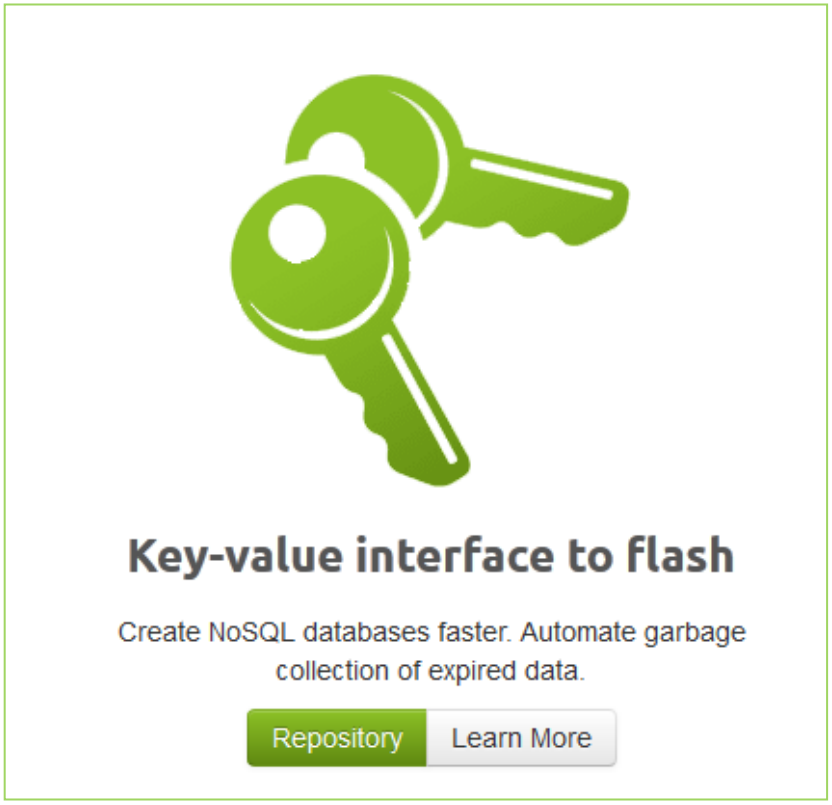
https://opennvm.github.io

On GitHub:

- Documentation
- Experimental Linux kernel with virtual memory swap patch (3.6 kernel)
- Benchmarking utility

3RD CONTRIBUTION: KEY-VALUE INTERFACE

FUSION-IO



https://opennvm.github.io

On GitHub:

- API specifications, such as:
  nvm_kv_put()
  - *nvm_kv_get()*
  - *nvm_kev_batch_put()*
  - *nvm_kv_set_global_expiry()*
- Sample program code
- Benchmarking utility
- Source code for flash optimized key value store

APPS USING OPENNVM TECHNOLOGY

FUSION-IO

Learn More »

PERCONA SERVER

Learn More »

https://opennvm.github.io

OPENNVM, STANDARDS, AND CONSORTIUMS

FUSION-IO

- opennvm.github.io
  - Primitives API specifications, sample code
  - Linux swap kernel patch and benchmarking tools
  - key-value interface API library and code, sample usage code, benchmark tools
- INCITS SCSI (T10) active standards proposals:
  - SBC-4 SPC-5 Atomic-Write
    http://www.t10.org/cgi-bin/ac.pl?t=d&f=11-229r6.pdf
  - SBC-4 SPC-5 Scattered writes, optionally atomic
    http://www.t10.org/cgi-bin/ac.pl?t=d&f=12-086r3.pdf
  - SBC-4 SPC-5 Gathered reads, optionally atomic
    http://www.t10.org/cgi-bin/ac.pl?t=d&f=12-087r3.pdf
- SNIA NVM-Programming TWG draft guide: http://snia.org/forums/sssi/nvmp

JOIN US AT OPENNVM.GITHUB.IO

FUSION-IO

Current OpenNVM Repositories

Flash-aware Linux swap

When working set size exceeds the capacity of DRAM, demand page from a flash-aware virtual memory subsystem.

Repository | Learn More

Key-value interface to flash

Create NoSQL databases faster. Automate garbage collection of expired data.

Repository | Learn More

Flash programming primitives

Use built-in characteristics of the Flash Translation Layer to perfrom journal-less updates (more performance and less flash wear = lower TCO)

Repository | Learn More


THANK YOU