Creating Flash-Aware Applications

Nisha Talagala

NVM (Flash, other) is different from Disk

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

I/O and Memory Access for Flash Aware Applications

FUSION-IO

I/O	I/O semantics examples: • Open file descriptor – open(), read(), write(), seek(), close() • (New – presented today) NVM Primitives • (New – presented today) NVM KV Store
Memory	Volatile memory semantics example: • Allocate virtual memory, e.g. malloc() • memcpy/pointer dereference writes (or reads) to memory address • (Improved – presented today) Page-faulting transparently loads data from NVM into memory

https://opennvm.github.io

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

3 Contributions to the Community

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

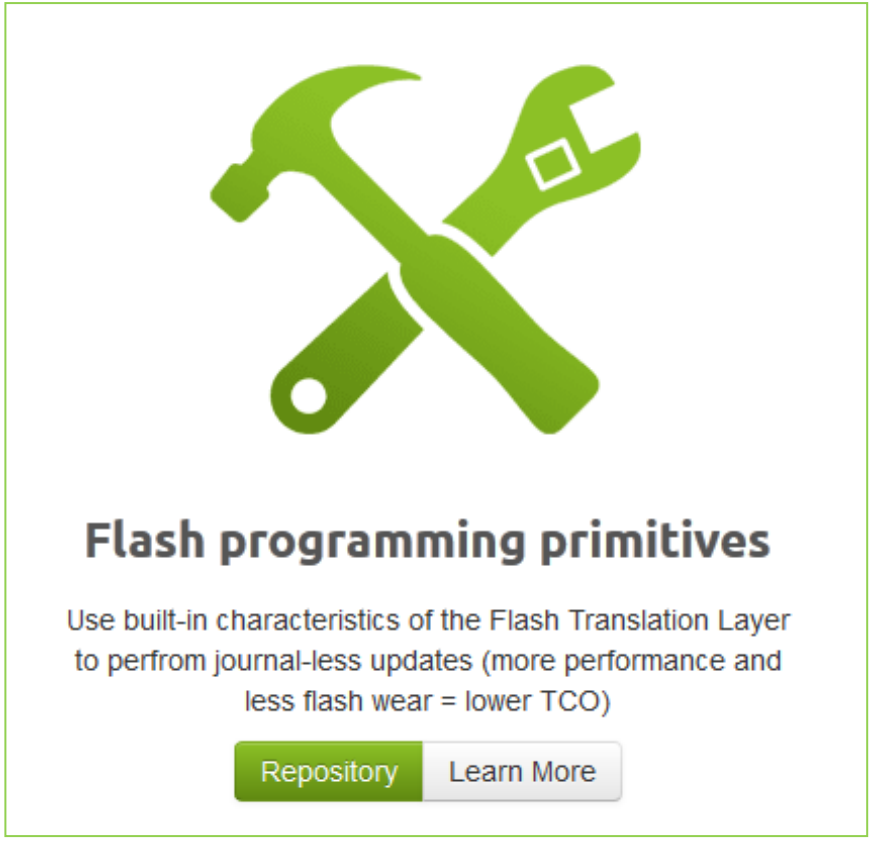
Use built-in characteristics of the Flash Translation Layer to perfrom journal-less updates (more performance and less flash wear = lower TCO)

Repository | Learn More

https://opennvm.github.io

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

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.

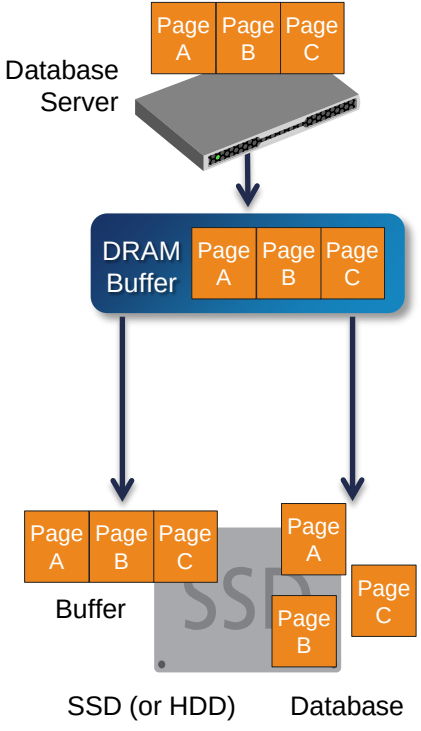
- **2x longer flash media life**
  Atomic Writes can increase the life of flash media up to 2x due to reduction in write-ahead-logging and double-write buffering.

- **50% less code in key modules**
  Atomic operations dramatically reduce application logic, such as journaling, built as work-arounds.

Atomic Writes – MySQL Example

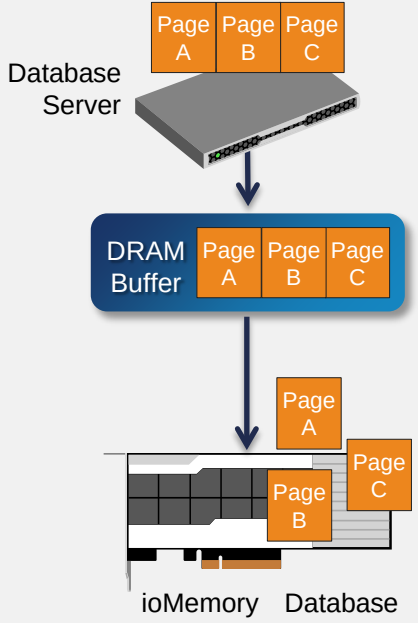
FUSION-IO

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

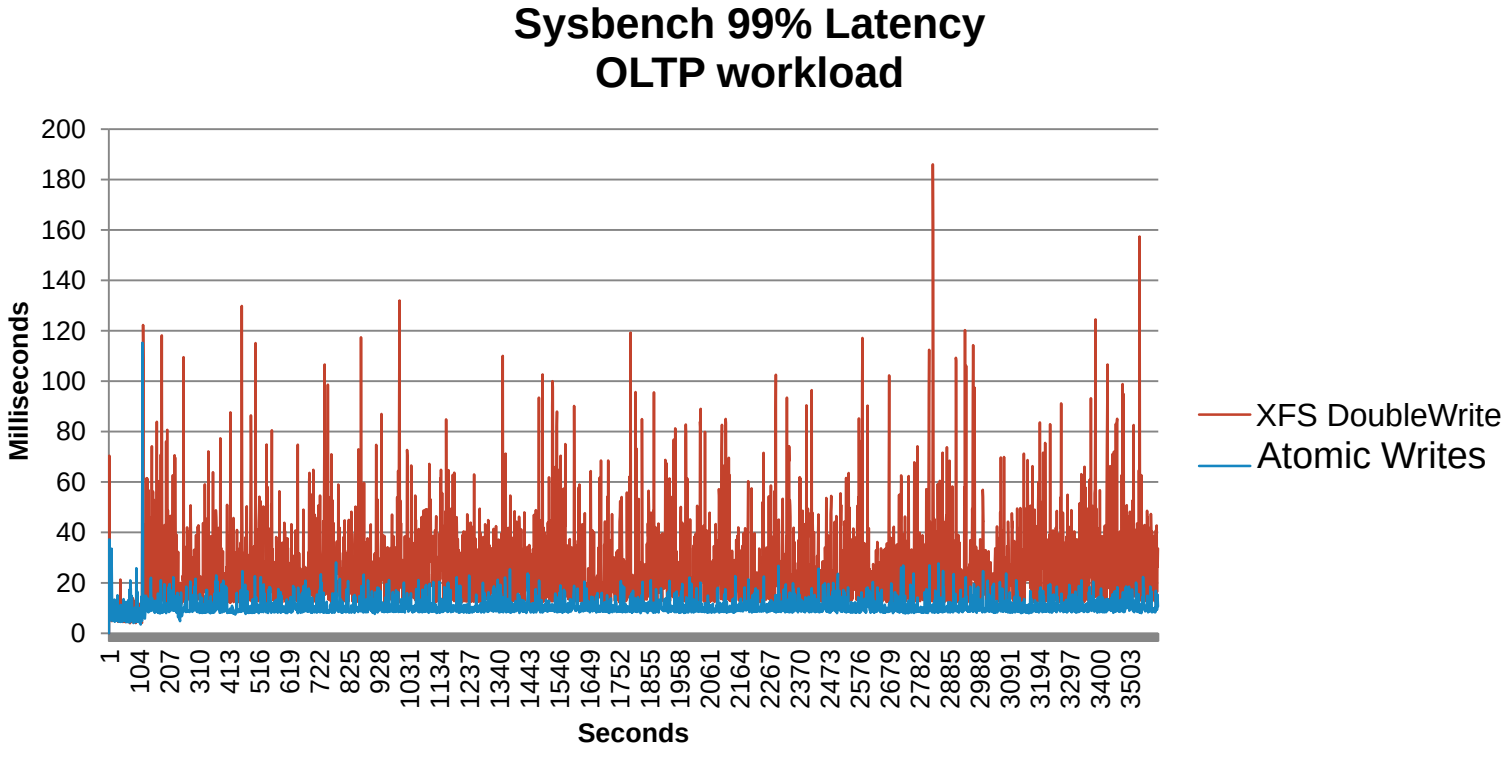


1. Application initiates updates to pages A, B, and C.
2. MySQL copies updated pages to memory buffer.
3. MySQL writes to actual tablespace, bypassing the double-write buffer step due to inherent atomicity guaranteed by the (intelligent) device.

MySQL Example: Latency Improvement

FUSION-IO

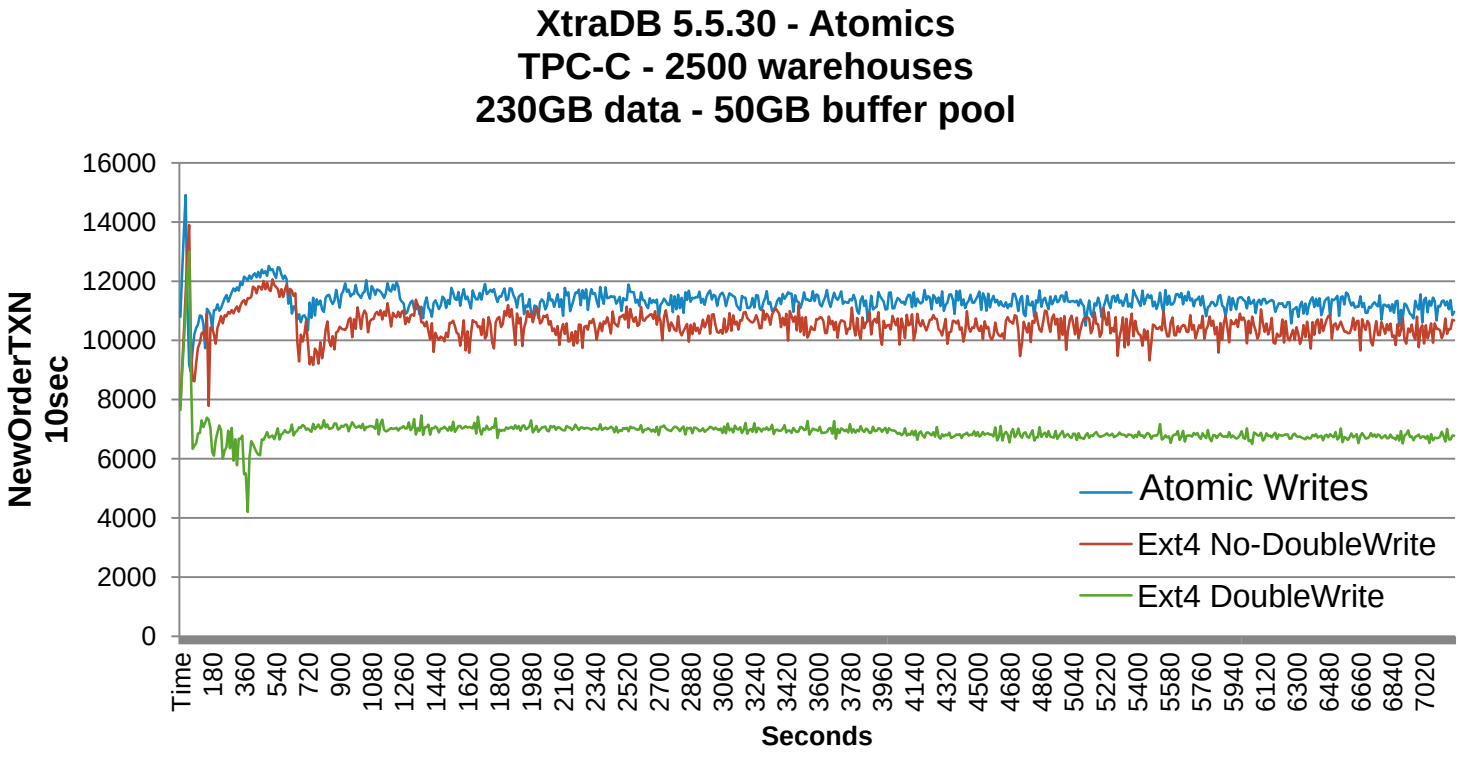
2-4x Latency Improvement on Percona Server



MySQL Example: Throughput Improvement

FUSION-IO

70% Transactions/sec Improvement on MariaDB Server



2nd Contribution: Linux Fast-Swap

FUSION-IO

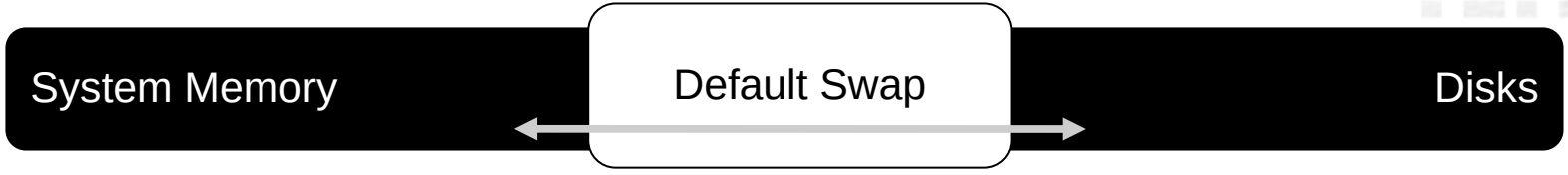
On GitHub:

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

https://opennvm.github.io

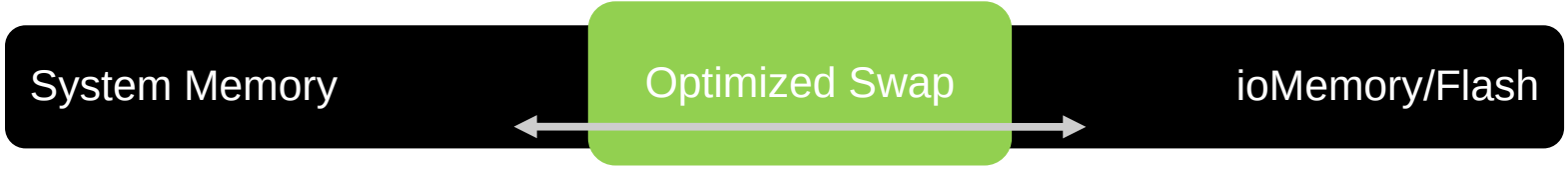
Improving Linux Swap (Demand-paging)

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



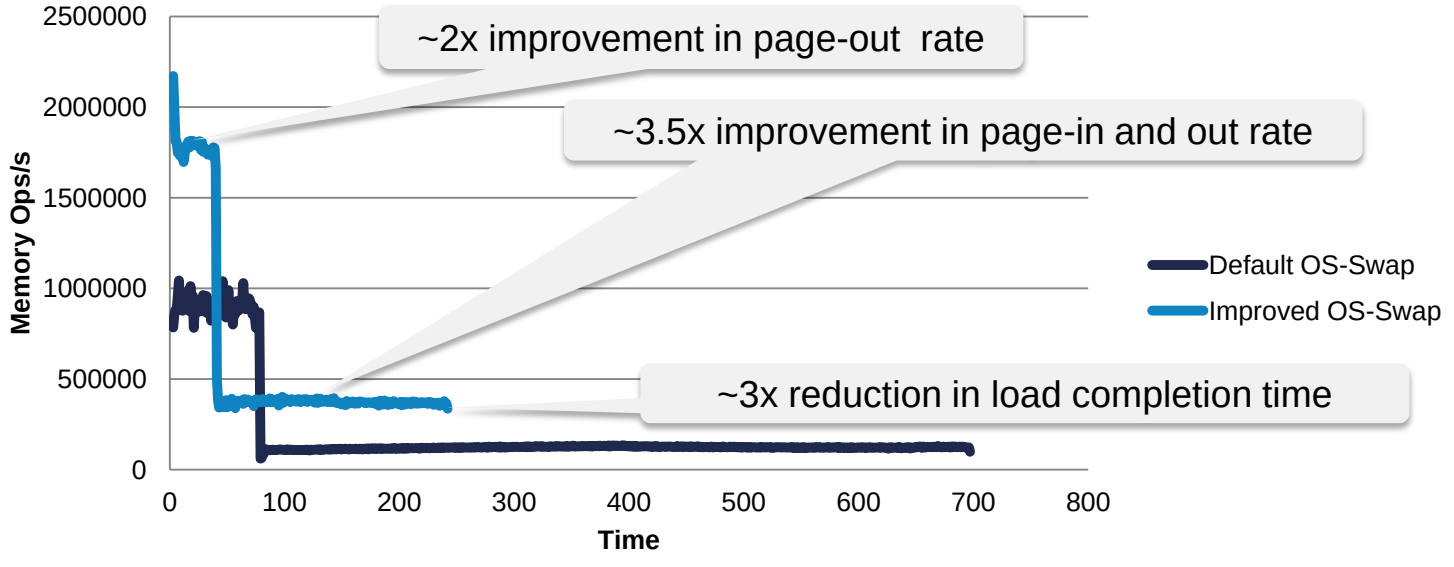
Tuned for flash (leverages native characteristics)

- 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

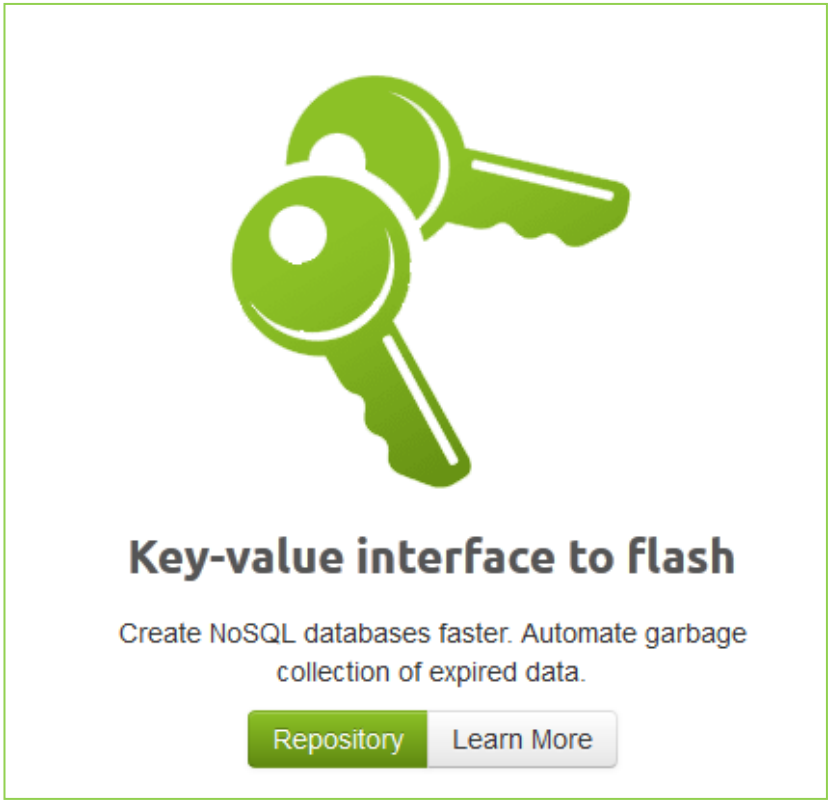
FUSION-IO

3x reduction in load completion time with fast swap



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 (30 Aug)

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.

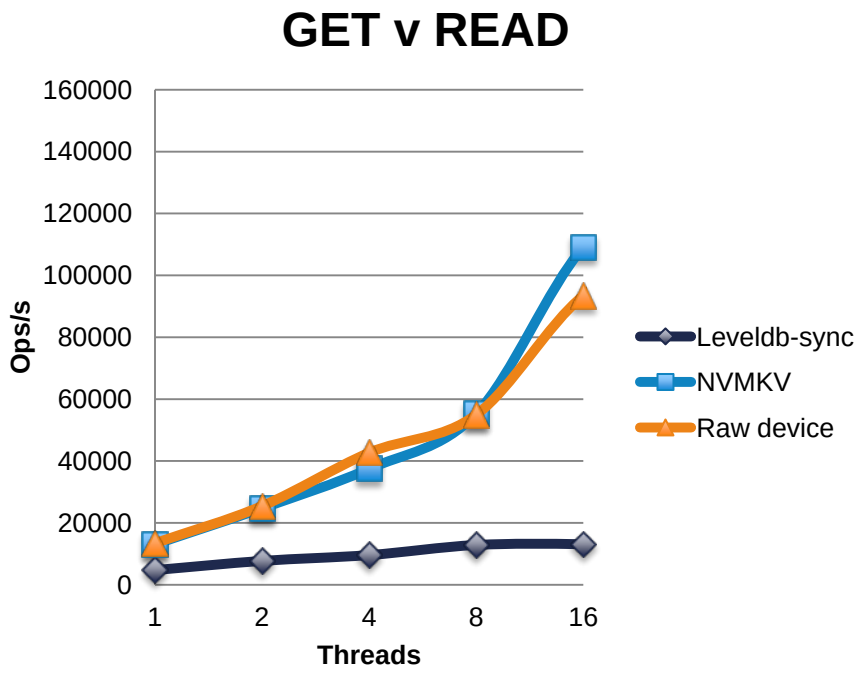
Reduce overprovisioning due to lack of coordination between two-layers of garbage collection (application-layer and flash-layer). Some top NoSQL applications recommend over-provisioning by 3x due to this.

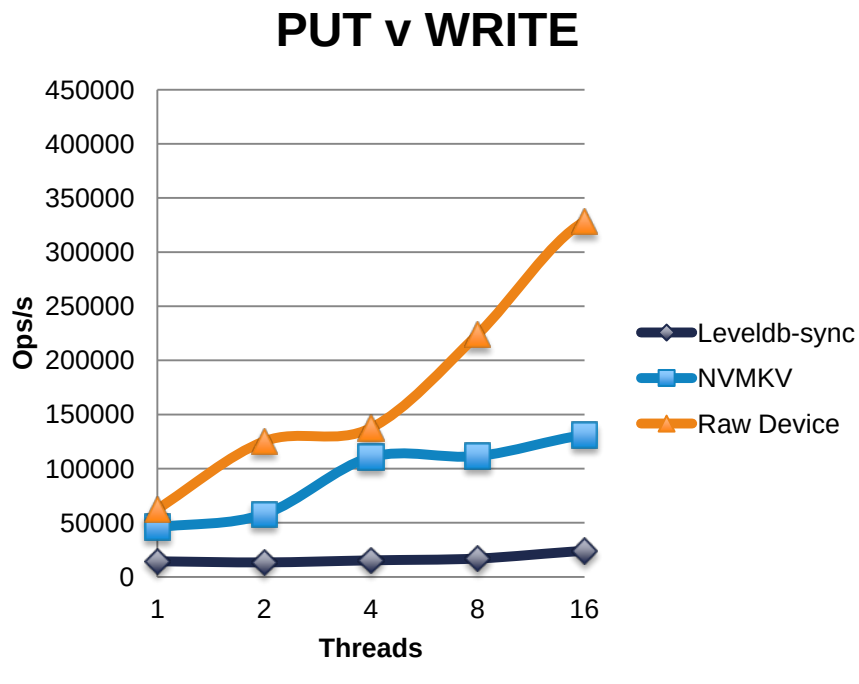
- **Near performance of raw device**
  Smarter media now natively understands a key-value I/O interface with lock-free updates, crash recovery, and no additional metadata overhead.
- **3x throughput on same SSD**
  Early benchmarks comparing against synchronous levelDB show over 3x improvement.
- **Up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





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, sample 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

Apps Using OpenNVM technology

FUSION-IO

Learn More »

Learn More »

https://opennvm.github.io

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
fusionio.com | REDEFINE WHAT'S POSSIBLE