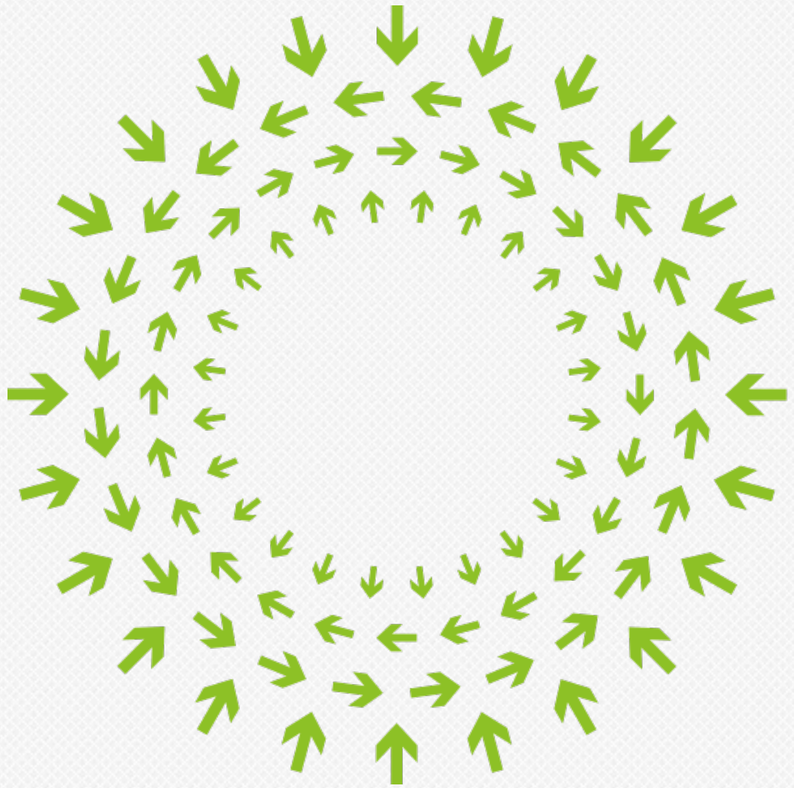


OPEN Compute Summit

January 28-29, 2014

San Jose

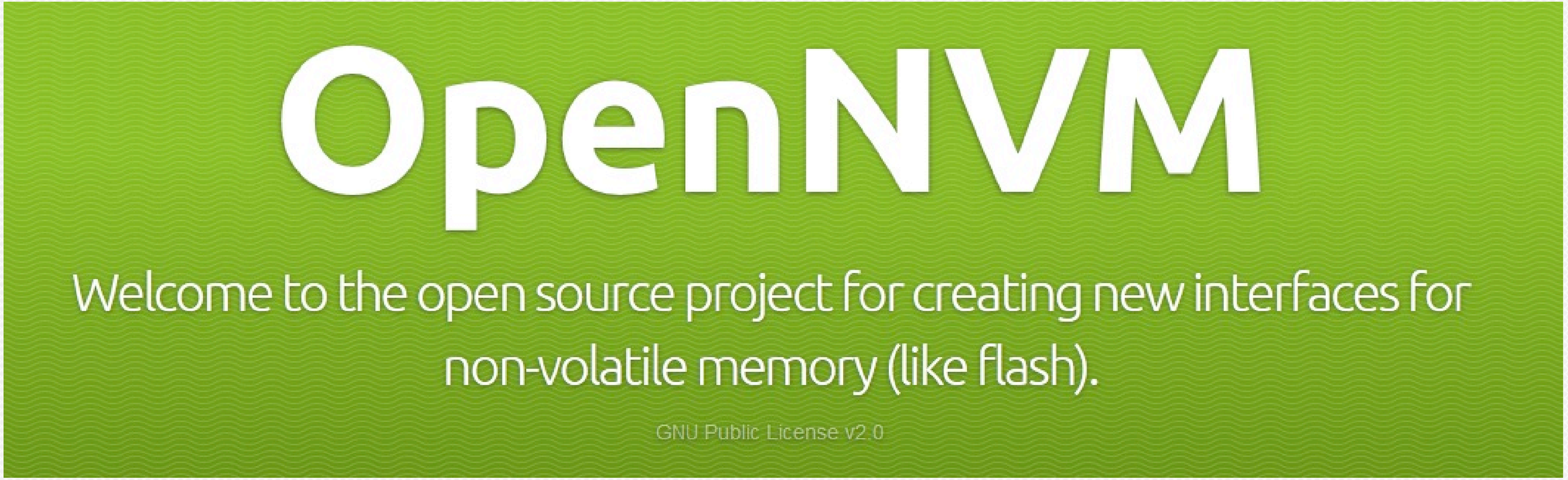
Flash for the Future

Software Optimizations for Non Volatile Memory

Nisha Talagala, Lead Architect, Fusion-io

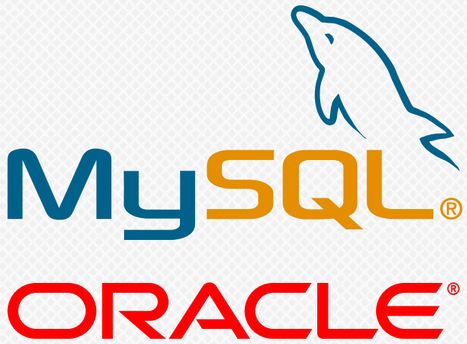
Gary Orenstein, Chief Marketing Officer, Fusion-io @garyorenstein

https://opennvm.github.io



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

Community Participation



Creating Flash-Aware Apps

I/O source code
written for disk

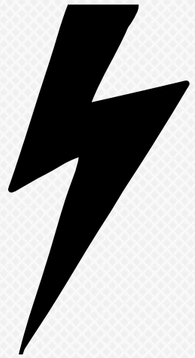
Creating Flash-Aware Apps

I/O source code
written for disk

(Flash disguised
to look like a disk)

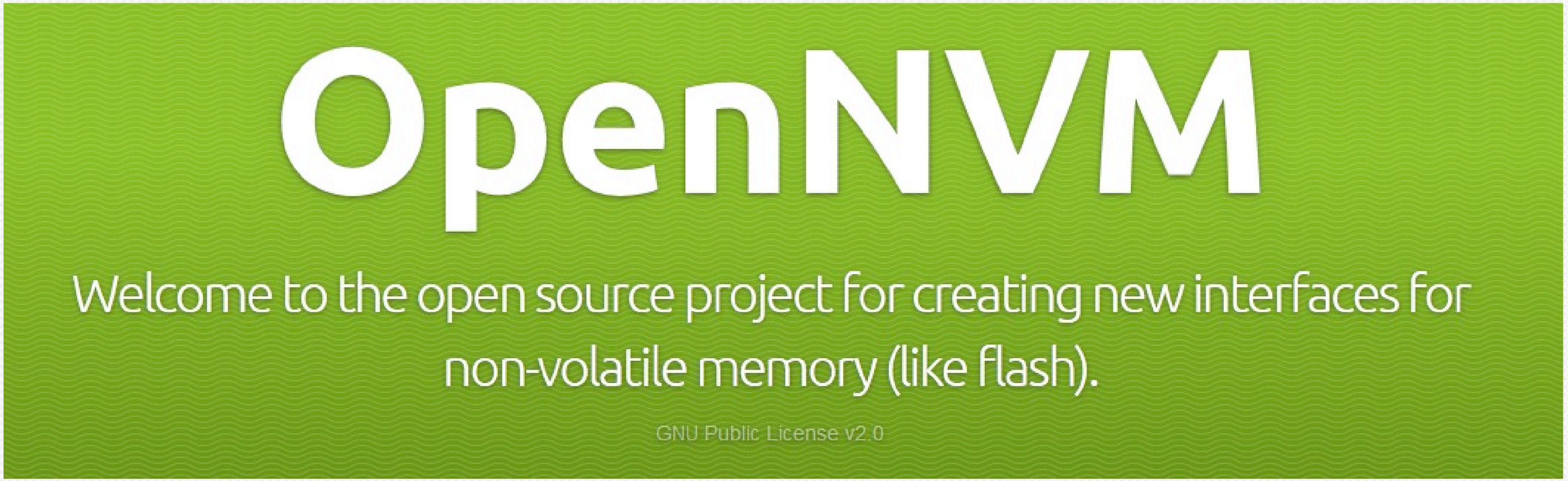
Creating Flash-Aware Apps

I/O source code written for flash



Creating Flash-Aware Apps

Leveraging the Community

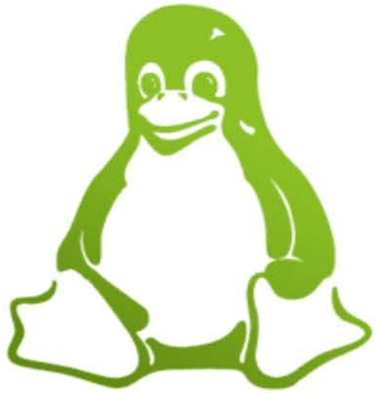


https://opennvm.github.io

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

3 Contributions to the Community

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 perform journal-less updates (more performance and less flash wear = lower TCO)

Repository | Learn More

https://opennvm.github.io

1st Contribution: Flash Primitives

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

https://opennvm.github.io

Flash Primitives: Sample Uses and Benefits

- Databases

  Transactional Atomicity:
  Replace various workarounds implemented in database code to provide write atomicity

  example: MySQL double-buffered writes

- Filesystems

  File Update Atomicity:
  Replace various workarounds implemented in filesystem code to provide file/directory update atomicity

  example: journaling

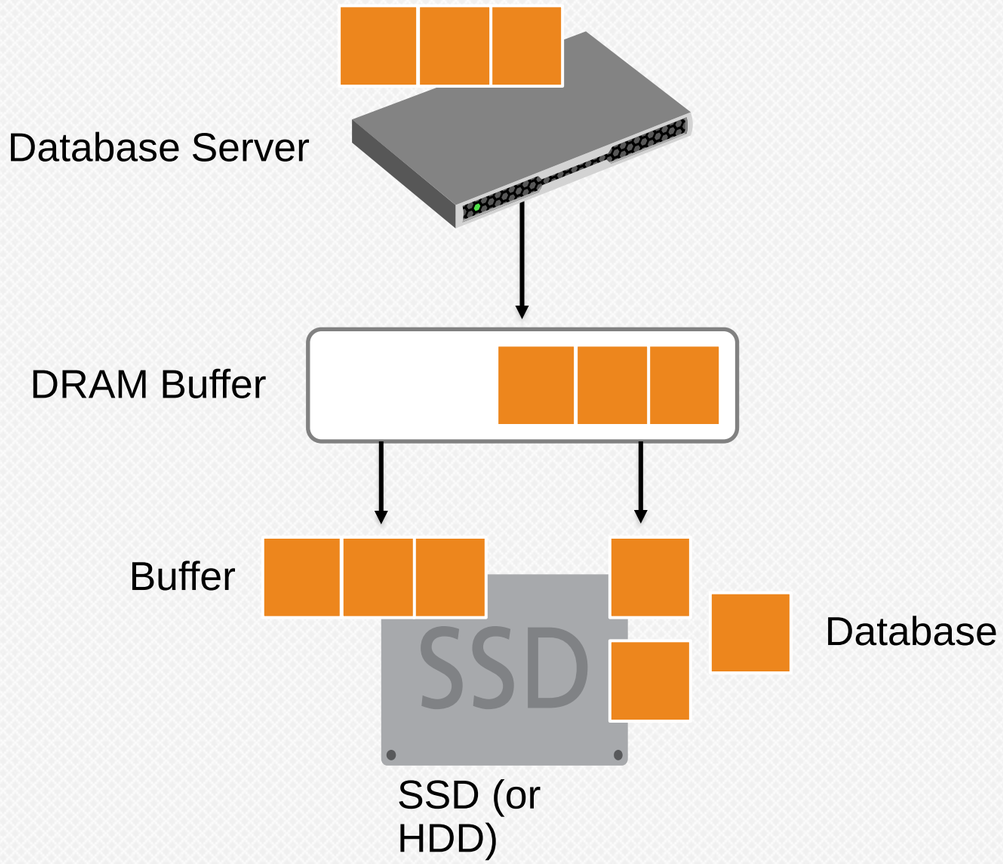
- **98% performance of raw writes**
  Smarter media now natively understands atomic updates, with no additional metadata overhead.

- **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.

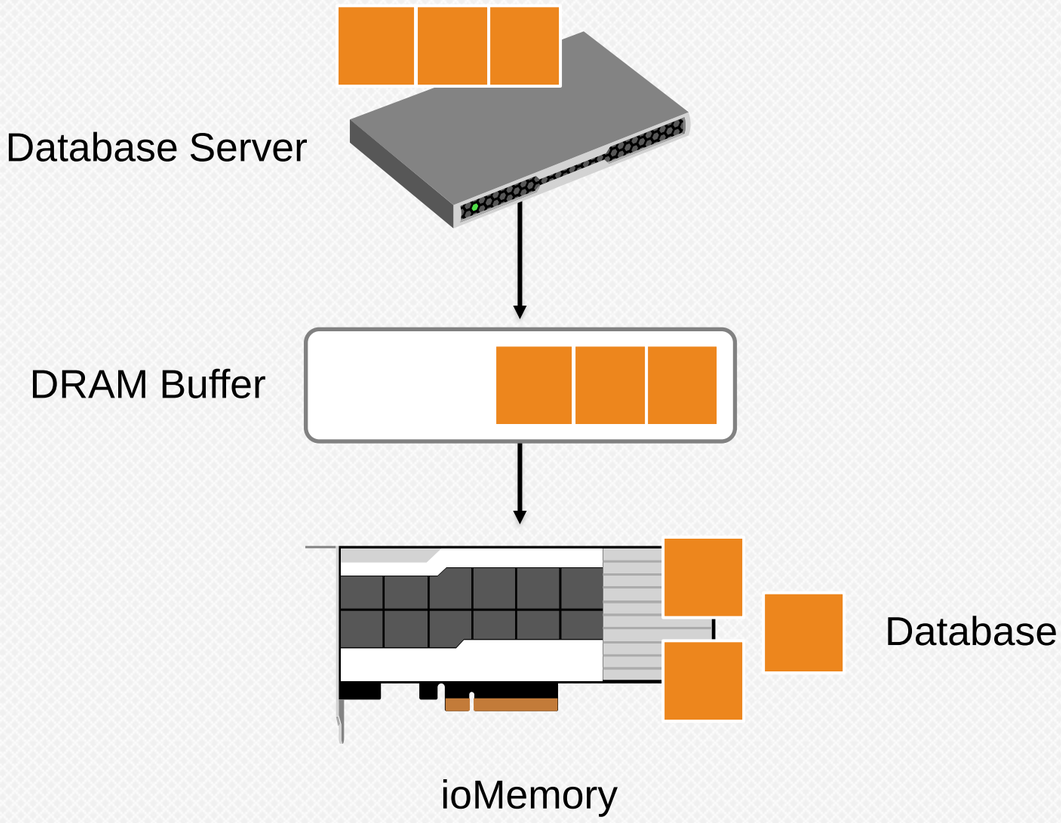
- **50% less code in key modules**
  Atomic operations dramatically reduce application logic, such as journaling, built as work-arounds.

Atomic Writes: MySQL Example

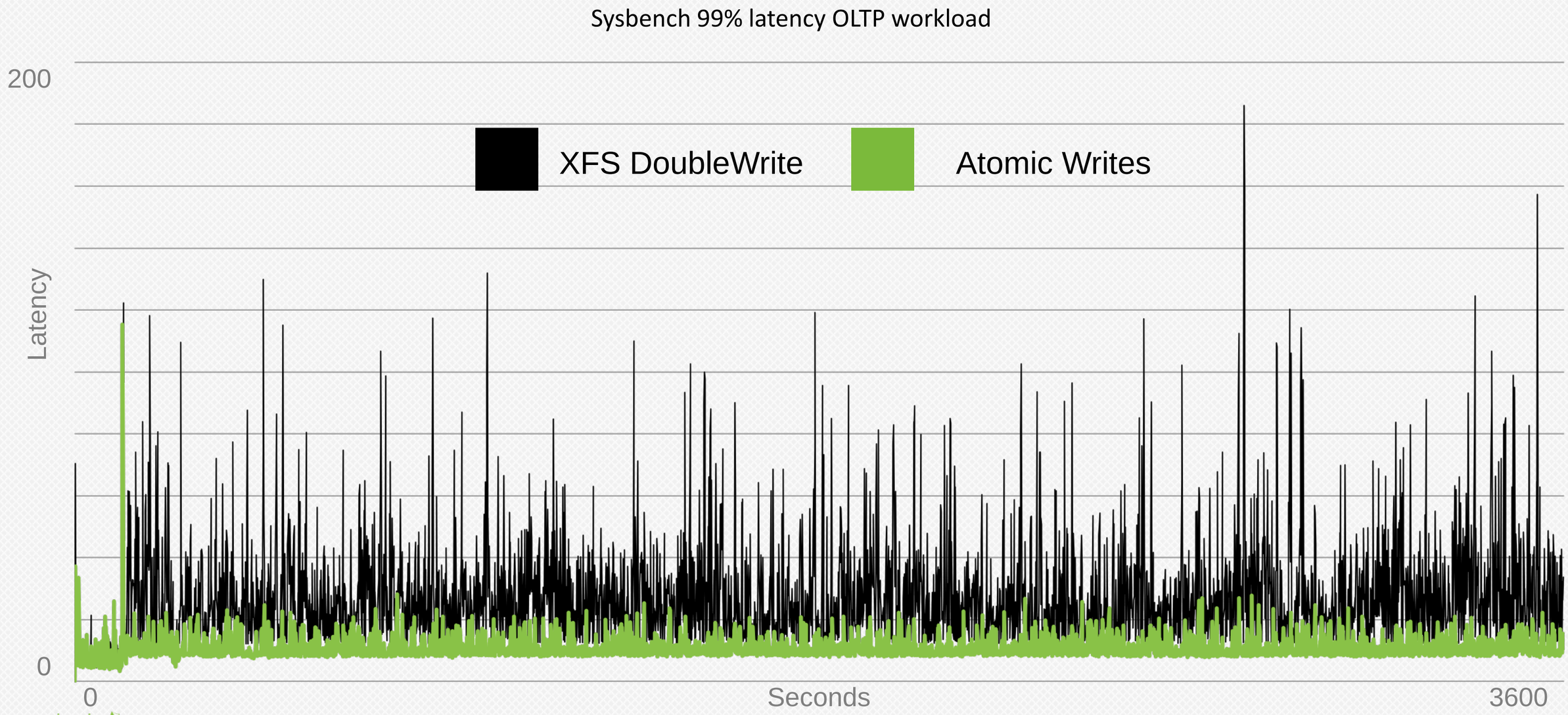
Traditional MySQL Writes



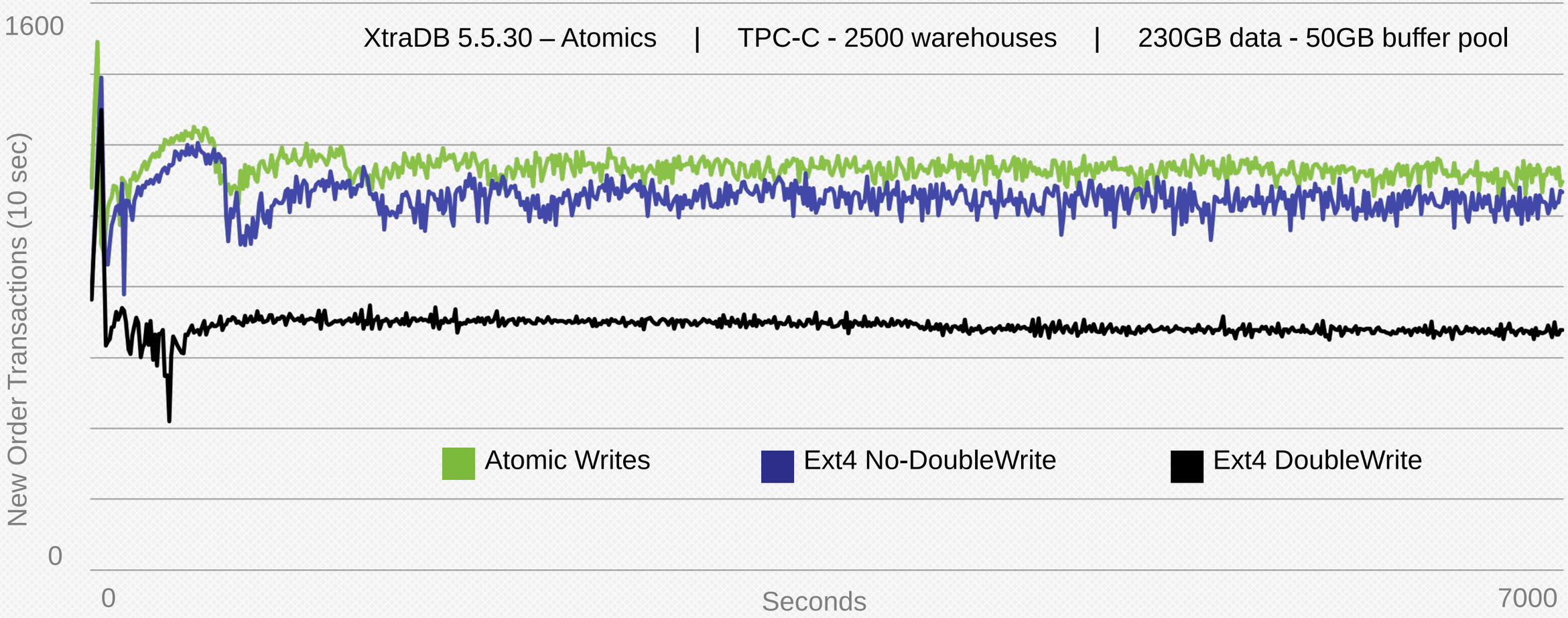
MySQL with Atomic Writes



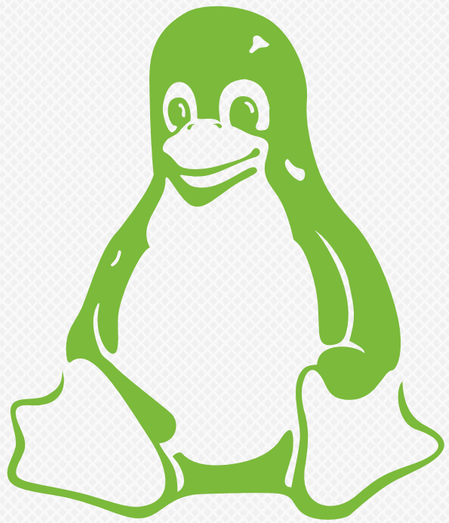
2-4x Latency Improvement on Percona Server



70% Transactions/sec Improvement on MariaDB Server



2nd Contribution: Linux Fast-Swap

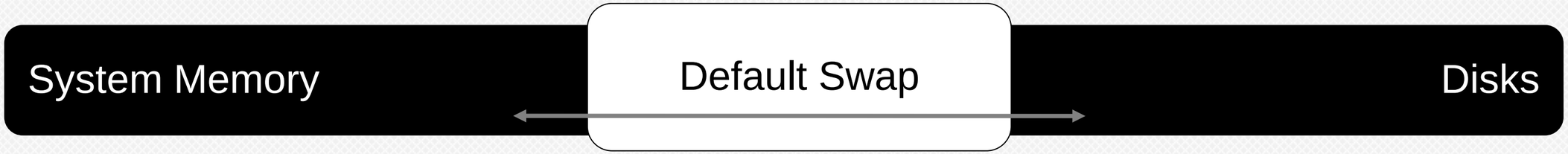


On GitHub

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

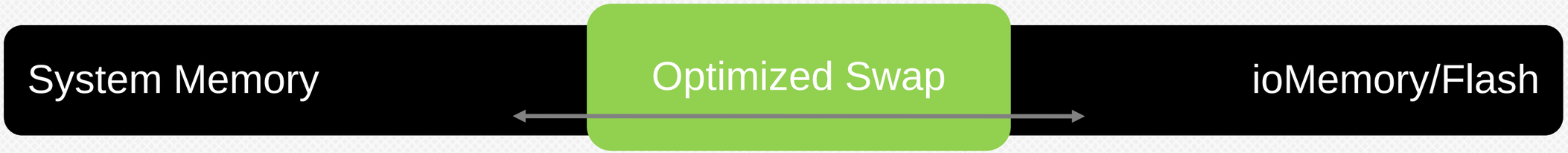
https://opennvm.github.io

Improving Linux Swap (Demand-paging)



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

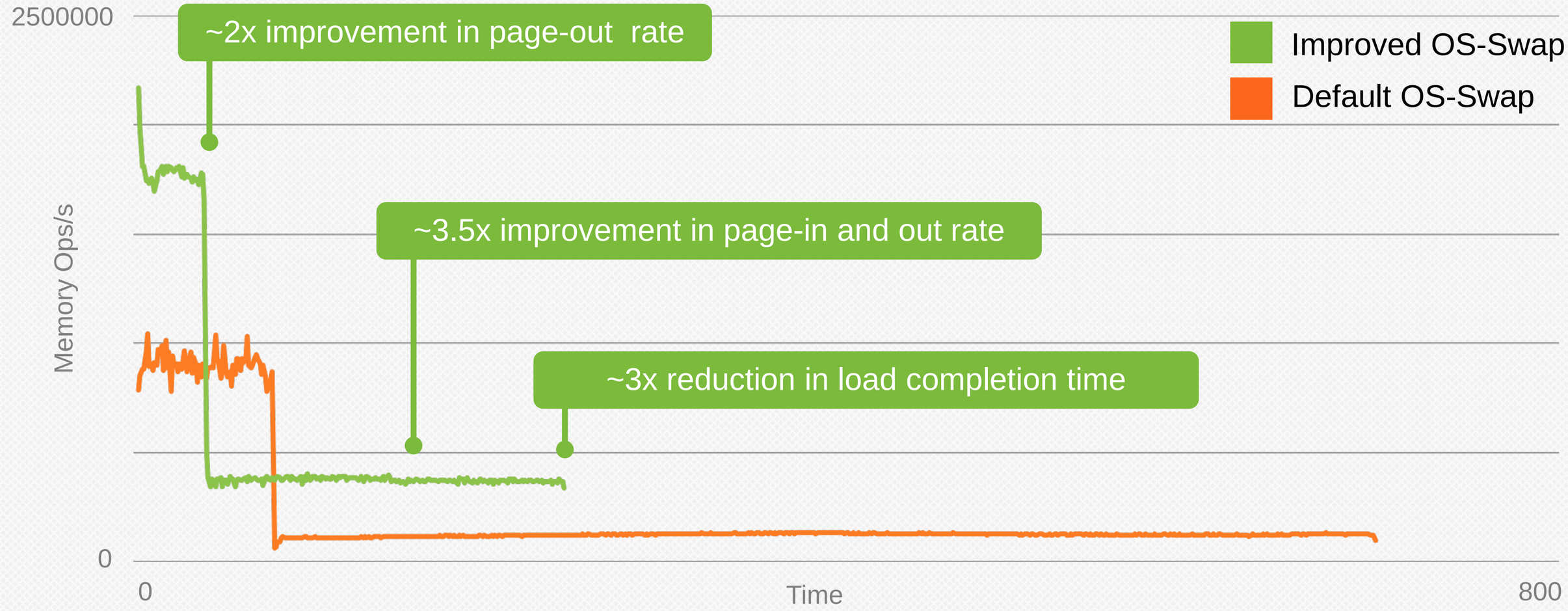
- Never tuned for high-performance demand-paging
- Never tuned for multi-threaded apps
- Poor performance



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

3x Performance with Fast Swap



3rd Contribution: Key-Value Interface

On GitHub:

- API specifications, such as:
  - *nvm_kv_put()*
  - *nvm_kv_get()*
  - *nvm_kev_batch_put()*
  - *nvm_kv_set_global_expiry()*
- KV library source code
- Sample program code
- Benchmarking utility
- Community contributions – Java bindings

https://opennvm.github.io

Key-Value Interface: Sample Uses and Benefits

- NoSQL Applications

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

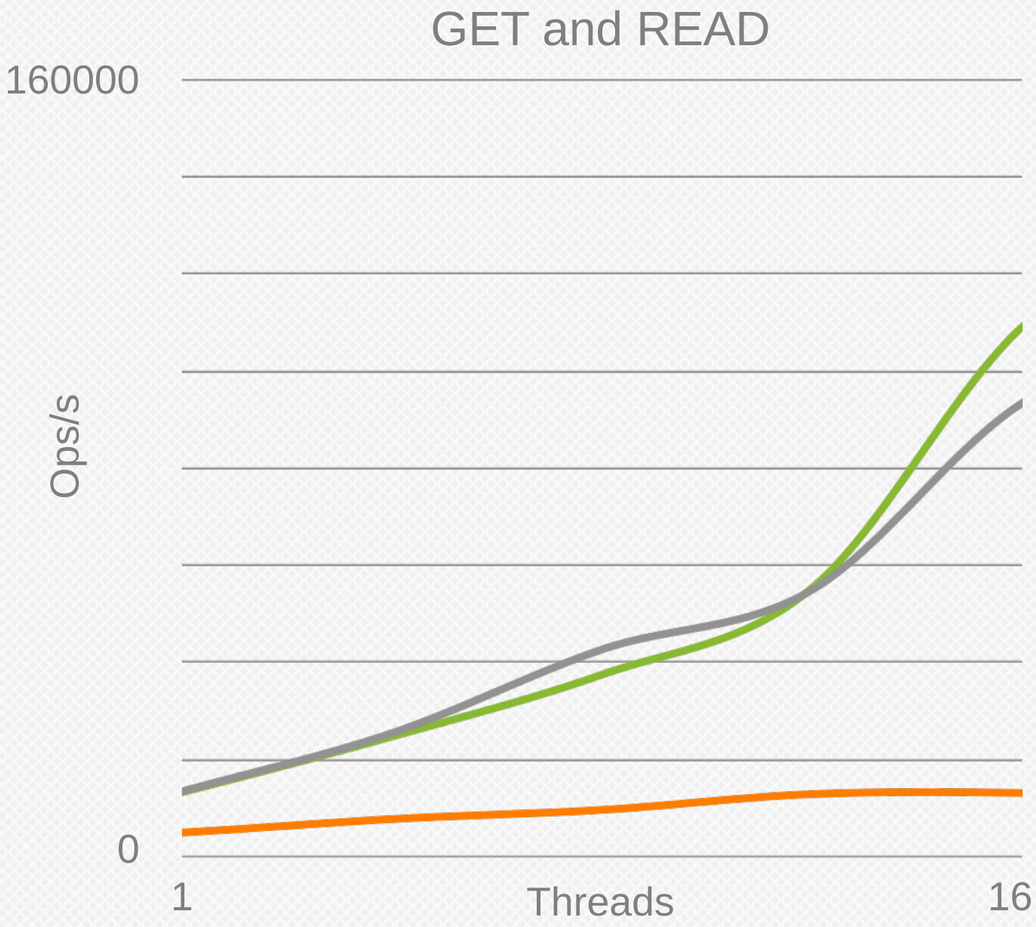
- **Near performance of raw device**
  Smarter media now natively understands a key-value I/O interface with lock-free updates, crash recovery, and no additional metadata overhead.

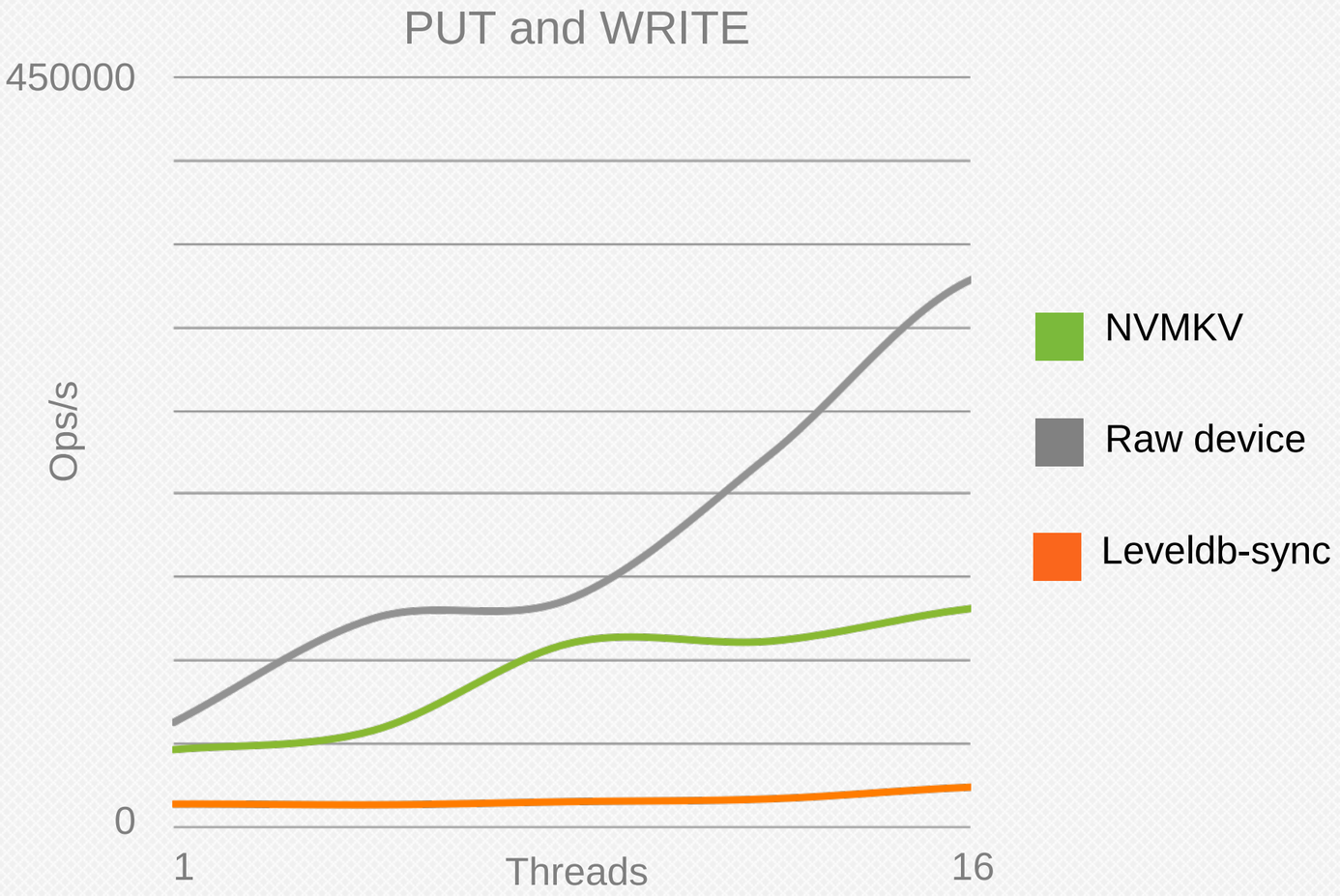
- **3x throughput on same SSD**
  Early benchmarks comparing against synchronous levelDB show over 3x improvement.

- **Up to 3x capacity increase**
  Dramatically reduces over-provisioning through coordinated garbage collection and automated key expiry.

Key-Value Interface for Performance

Key-Value get/put, Raw read/write, levelDB read/write





OpenNVM, Standards, and Consortiums

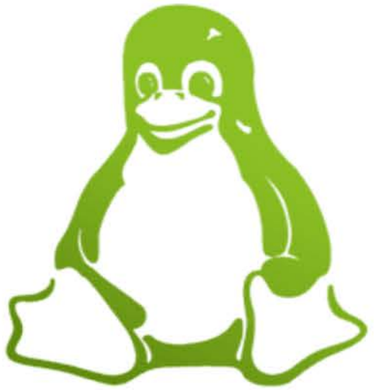
- opennvm.github.io
  - Primitives API specifications, sample code
  - Linux swap kernel patch and benchmarking tools
  - key-value interface API library, sample code, benchmark tools
- INCITS SCSI (T10) active standards proposals:
  - SBC-4 SPC-5 Atomic-Write
    http://www.t10.org/cgi-bin/ac.pl?t=d&f=11-229r6.pdf
  - 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 v1.0
  http://snia.org/tech_activities/standards/curr_standards/npm

Apps Using OpenNVM technology

https://opennvm.github.io

Join us at opennvm.github.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 perform journal-less updates (more performance and less flash wear = lower TCO)

Repository | Learn More