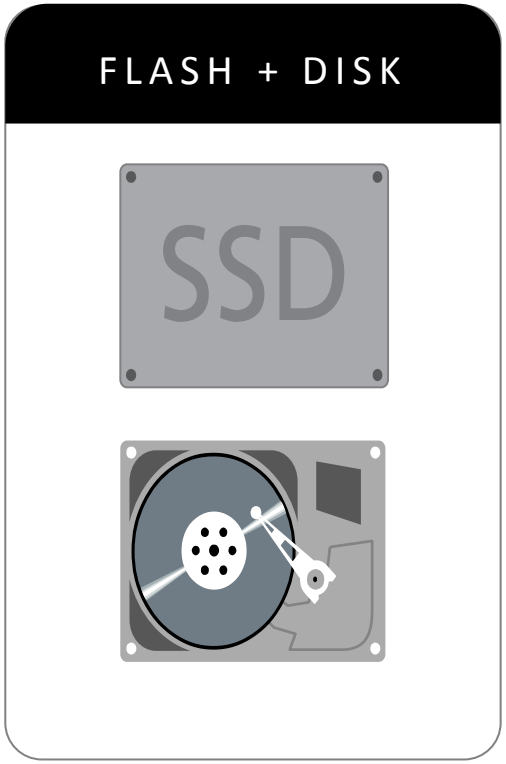
OPENFABRICS ALLIANCE

2013 OFA Developer Workshop

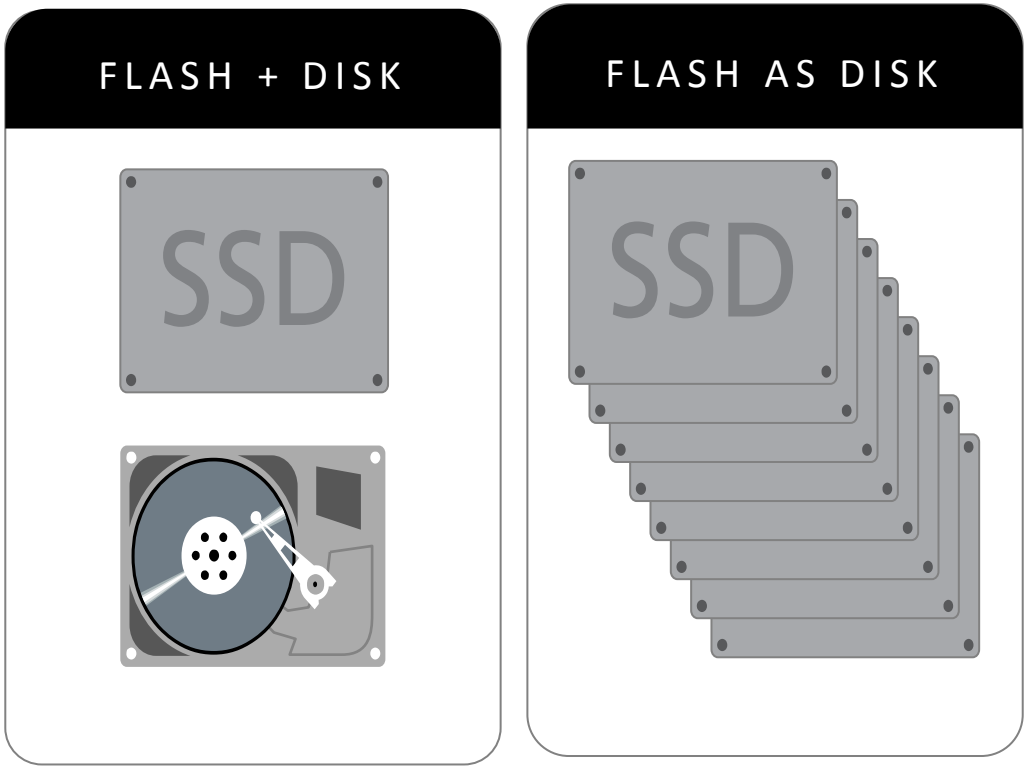
Standardizing New NVM Software Architectures and Architectures

Walt Hubis, Storage Standards Architect, Fusion-io

Evolution of Flash Adoption

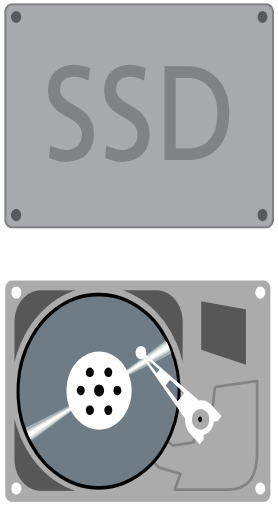
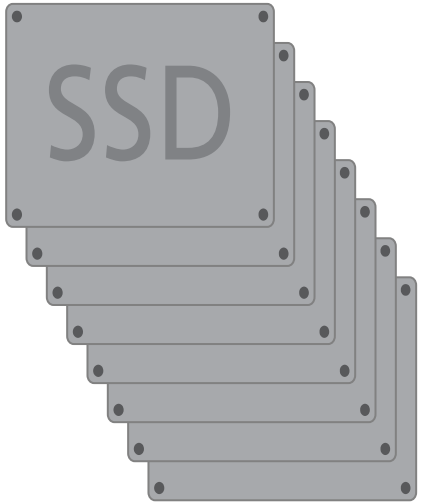
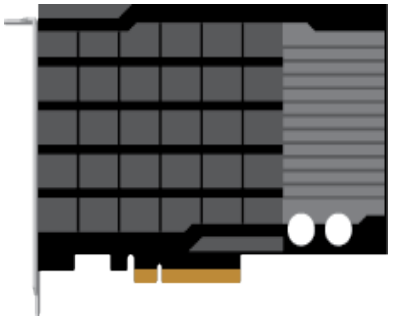


Evolution of Flash Adoption

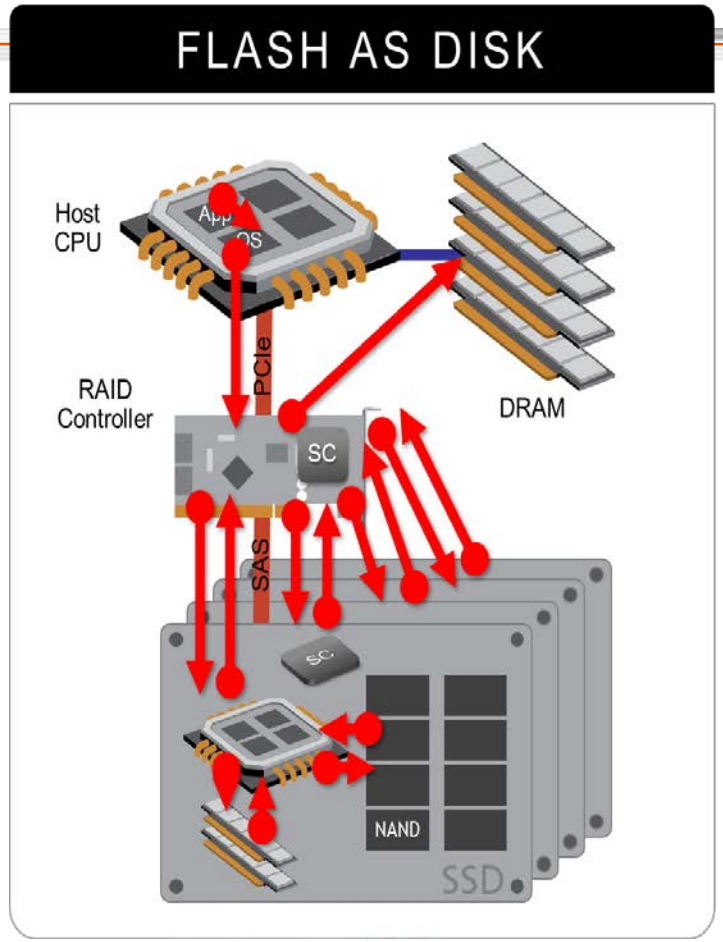


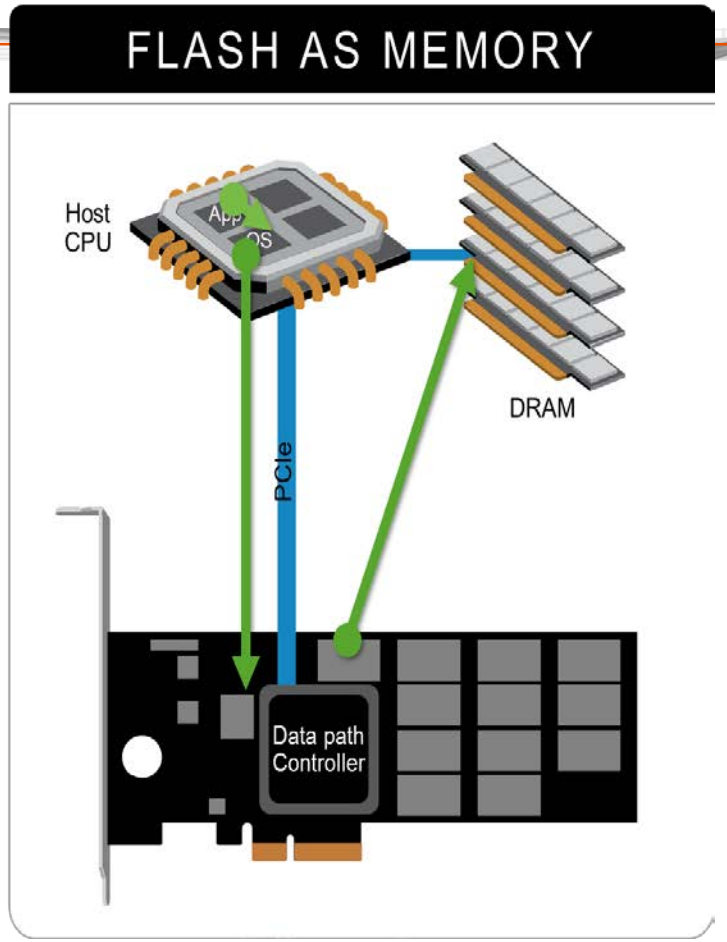
Evolution of Flash Adoption

OPENFABRICS ALLIANCE

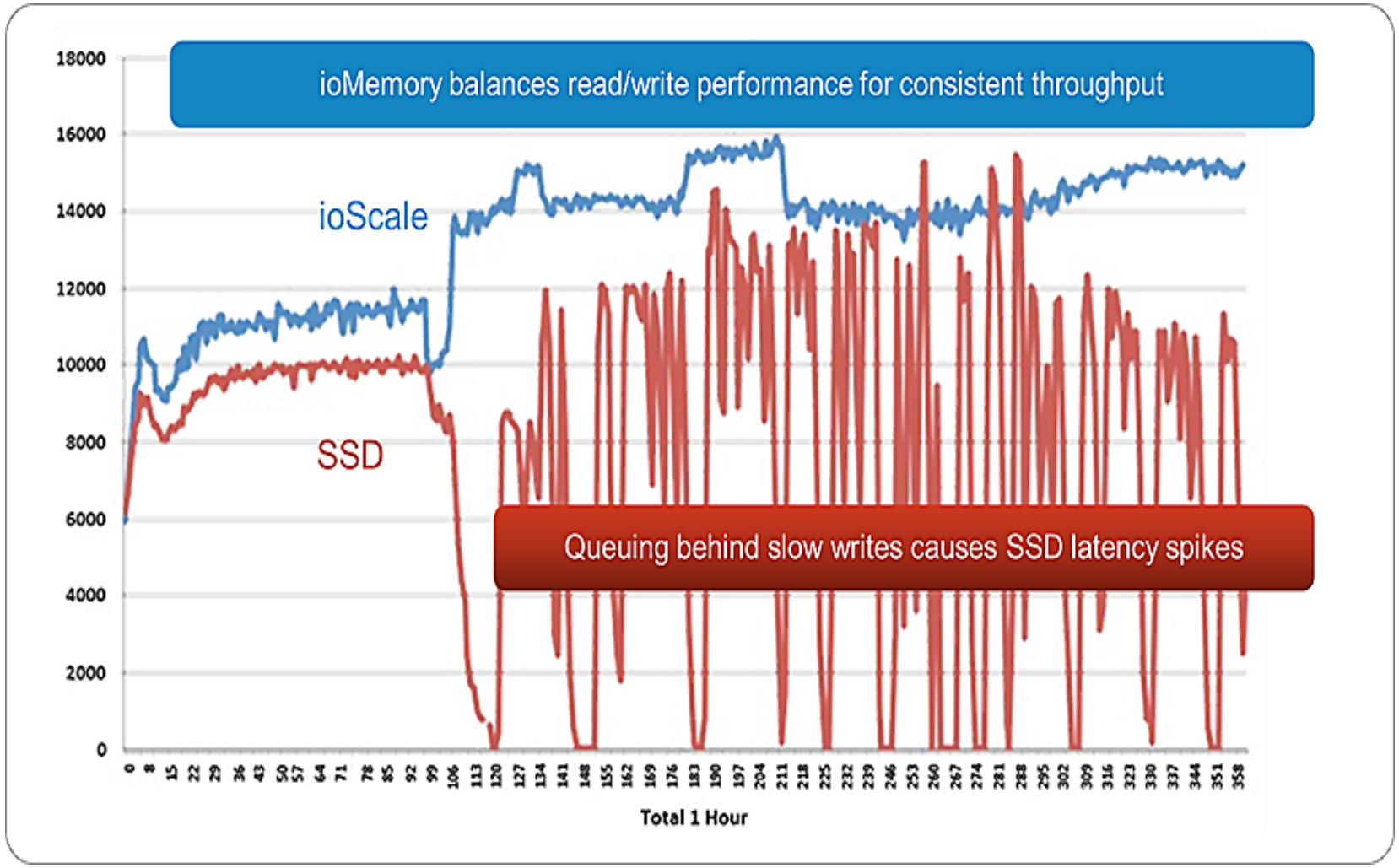
FLASH + DISK	FLASH AS DISK	FLASH AS MEMORY
		

Flash Architectures

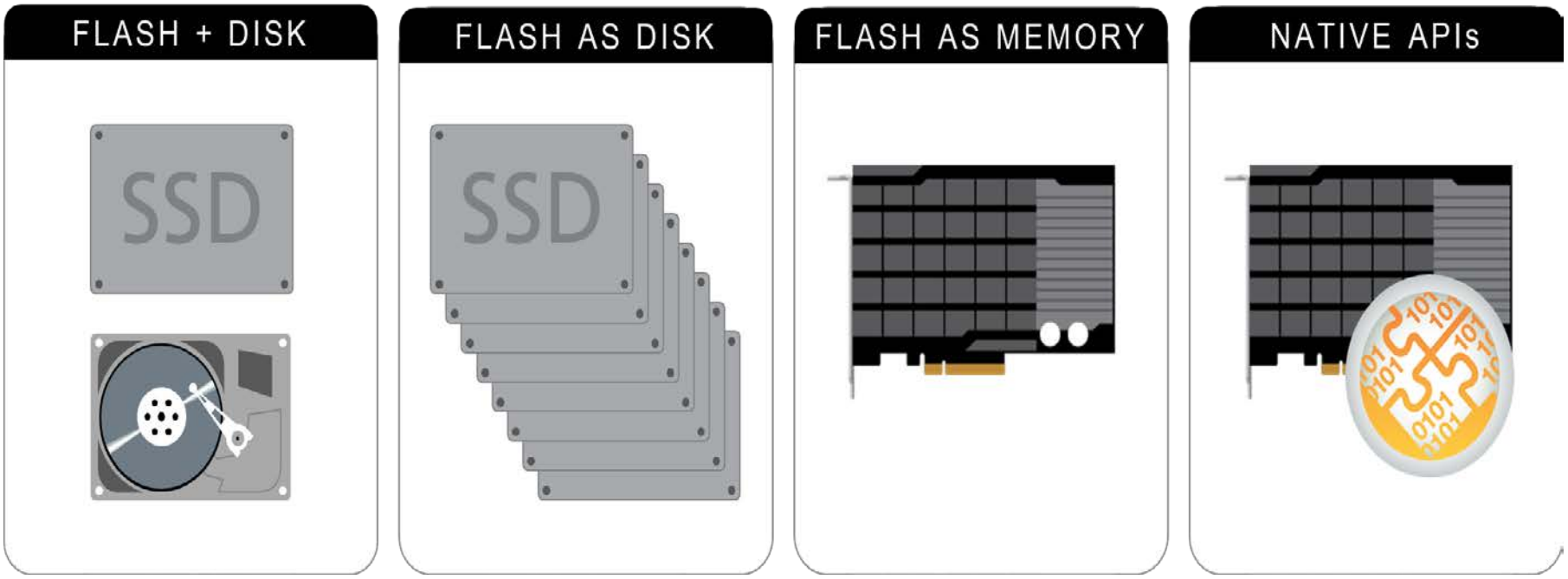




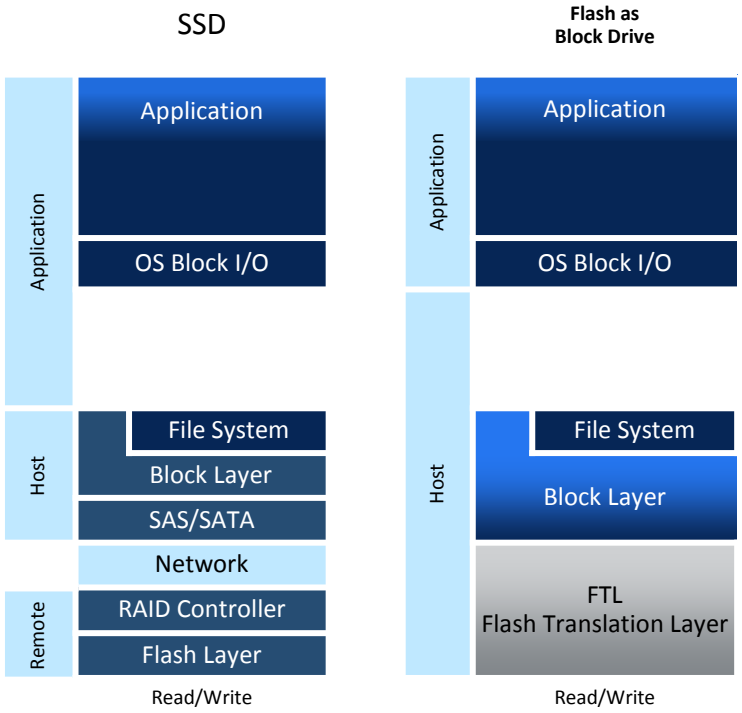
Balanced Performance Affects Throughput



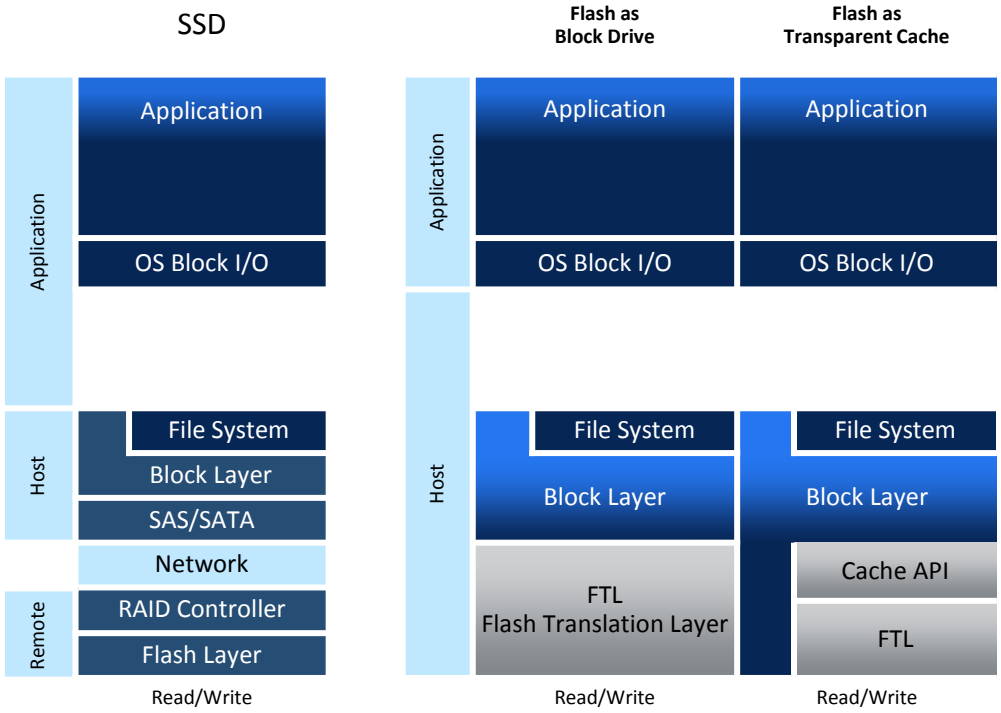
Evolution of Flash Adoption



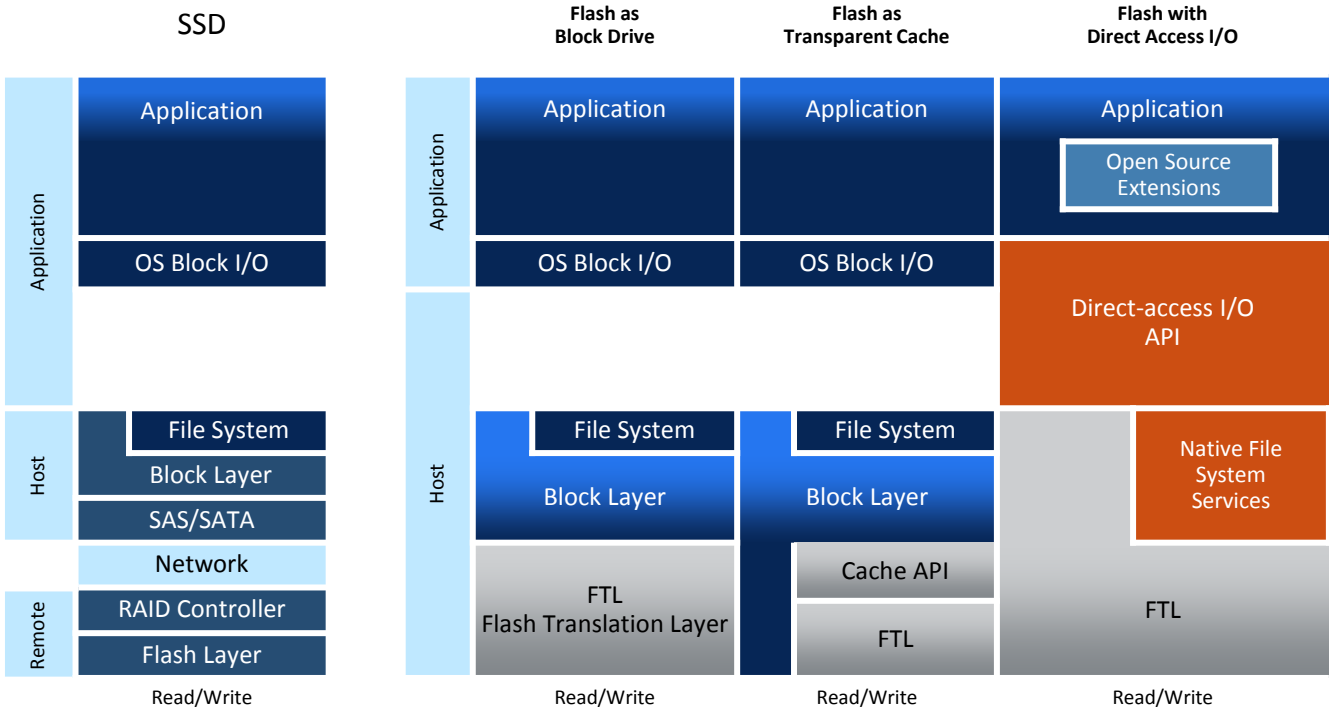
Non-Volatile Memory Evolution



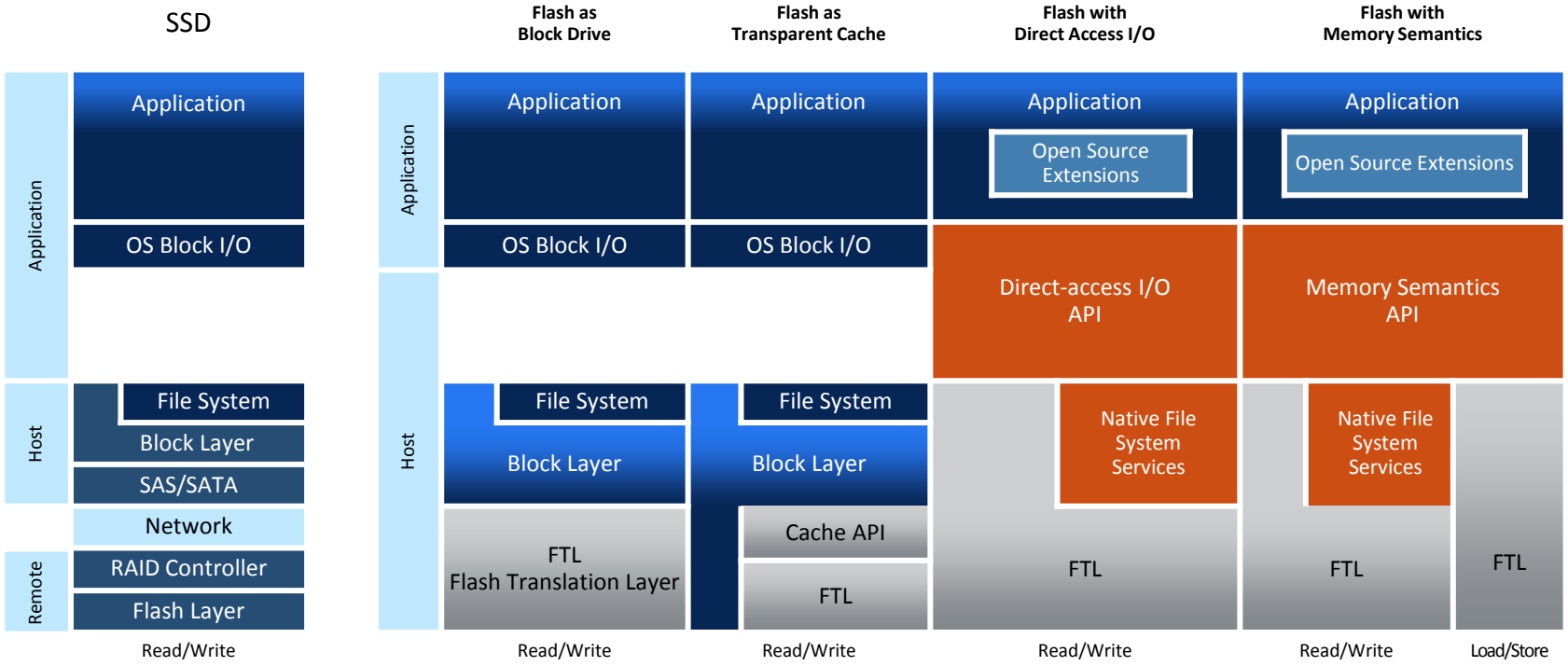
Non-Volatile Memory Evolution



Non-Volatile Memory Evolution



Non-Volatile Memory Evolution



Comparing I/O and Memory Access Semantics

OPENFABRICS ALLIANCE

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 map Auto-Commit Memory™ (ACM) virtual memory pages • memcpy/pointer dereference writes (or reads) to memory address • (New) Call checkpoint() to create application-consistent ACM page snapshots • (New) After system failure, remap ACM snapshot pages to recover memory state • (New) De-stage completed ACM pages to NVM namespace • (New) Remap and access ACM pages from NVM namespace at any time

New Primitives for a New Type of Media

OPENFABRICS ALLIANCE

Tape	Open, read, write, rewind, close.
Disk	Open, read, write, seek, close.
SSD	Open, read, write, seek, close.
ioMemory NVM	Open, read, write, seek, close. *Plus, new primitives to exploit characteristics of non-volatile memory* Basic write + atomic write, conditional write. Basic write + TTL expiry for auto-deletion. Basic mmap + crash-safety, versioning.

ATOMIC I/O Primitives: Sample Uses and Benefits

OPENFABRICS ALLIANCE

Databases

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

Filesystems

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

- 99% performance of raw writes
  Smarter media now natively understands atomic updates, with no additional metadata overhead.
- 2x longer flash media life Atomic Writes 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.

SNIA Non-Volatile Memory (NVM) Program Problem Statement

- NVM features and performance are outgrowing the existing storage model
- Sending block reads/writes down the traditional IO stack is insufficient and becoming inefficient
  - OK if NVM to be represented as a traditional disk
  - Not OK for higher order NVM operations
- NVM technology is evolving less as storage, more as memory
  - Need a programming model for storage memory usage
- Critical need to collaborate cross-industry to define and implement this new programming model
- SNIA creates NVM Technical Working Group June 2012

SNIA NVM Programming TWG Formation

- Charter: Develop specifications for new software programming models for use of NVM
  - Scope:
    - Programming models for applications and OS components
    - Each model covers NVM extensions for block storage, file access, and memory access models
- Operating System (OS) Specific APIs
  - SNIA defines the programming model specification
  - Each OS Vendor codes the programming models to the specific OS
    - Discussion with Linux community underway

NVM Accessed as Memory

- Samples of behavior to be covered in specification
  - Discover available NVM devices
  - Discover their characteristics and support for optional features:
    - Examples : Atomic operations, provisioning, etc…
  - Assign a region of NVM to a process memory address
    - Same region has to map the same way across reboots
  - How to read/write to NVM
    - How to commit changes to NVM
    - Use of behavior to assure durability and consistency (flush, …)

NVM Programming Model Exclusions

- The programming model is tied to other kernel behavior
  - Access control and ownership
  - Device discovery and naming
    - Frameworks related to storage
    - Events
    - SW install/upgrade
    - Device management
- Vendor Unique Behaviors
  - Flash maintenance and grooming
  - Implementation of FTL and associated services
  - Certain types of error conditions

TWG Status

- Weekly calls
  - Tuesdays at 4:00PM Pacific
- Two Day Face to Face Meetings
  - Quarterly at SNIA Symposia
- See me if you are interested in attending
- Current work
  - Use Cases
  - Actions
  - Glossary
- Deliverable Schedule TBD

Open Interfaces and Open Source

- Primitives: Open Interface
- API Libraries: Open Source, Open Interface
- 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

Thank You