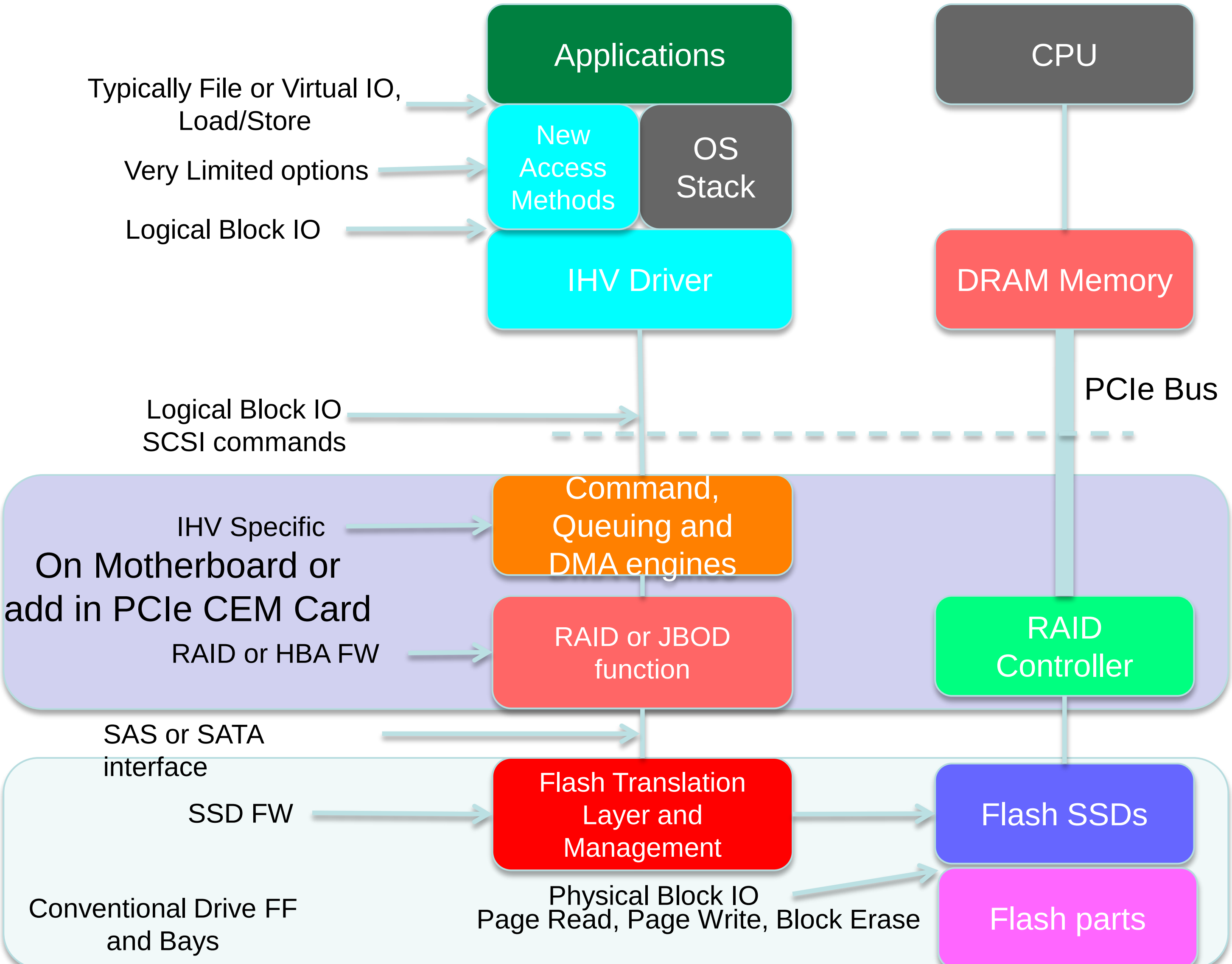
How Next Generation NV Technology Affects Storage Stacks and Architectures

Marty Czekalski, Interface and Emerging Architecture Program Manager, Seagate Technology

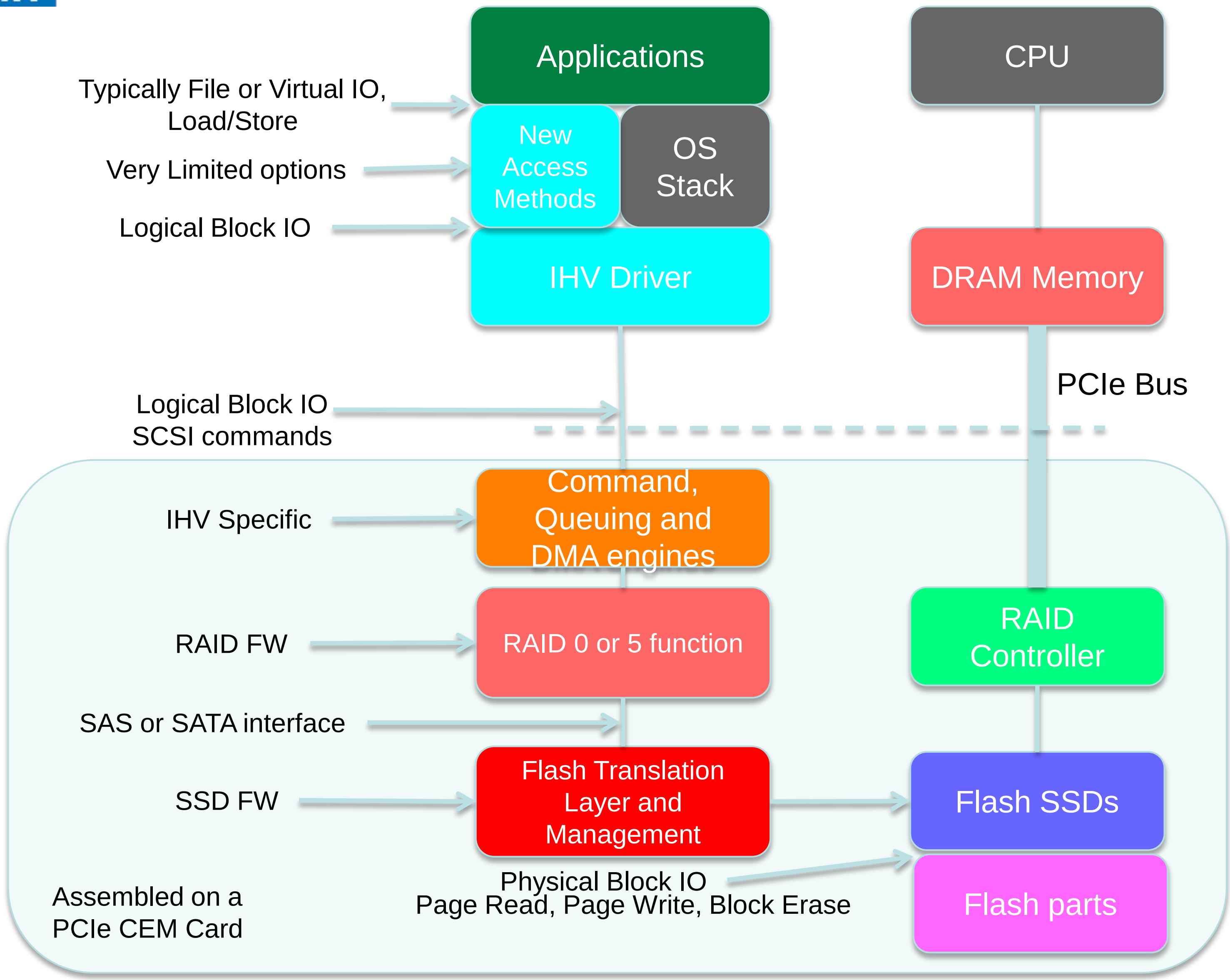
Outline

- Overview of existing SW stacks and interfaces
- New NVM devices
- NVM Programming Models
  - Block, File, Persistent Memory (PM)
- Technical and Ecosystem Challenges

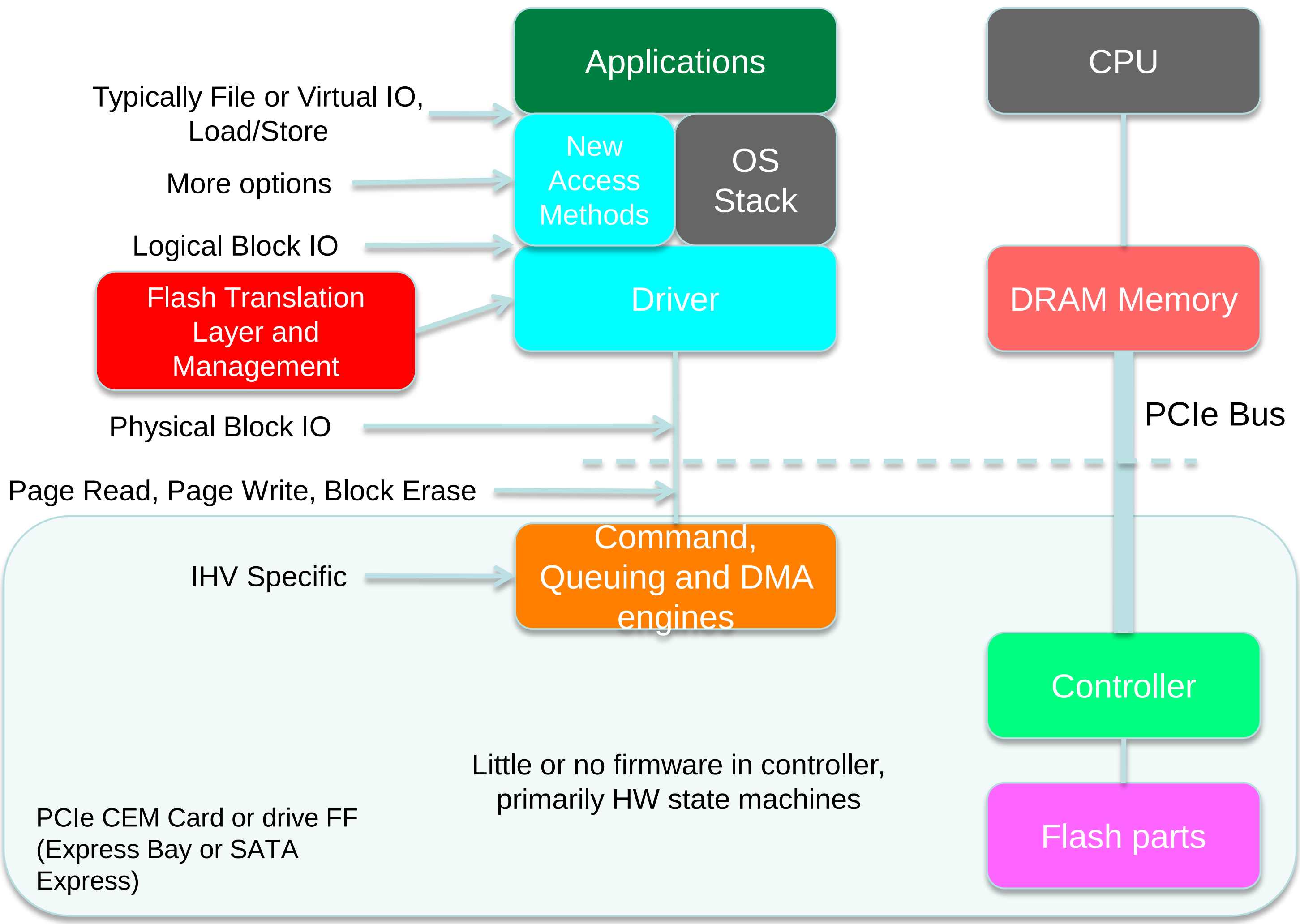
Conventional Storage Architecture



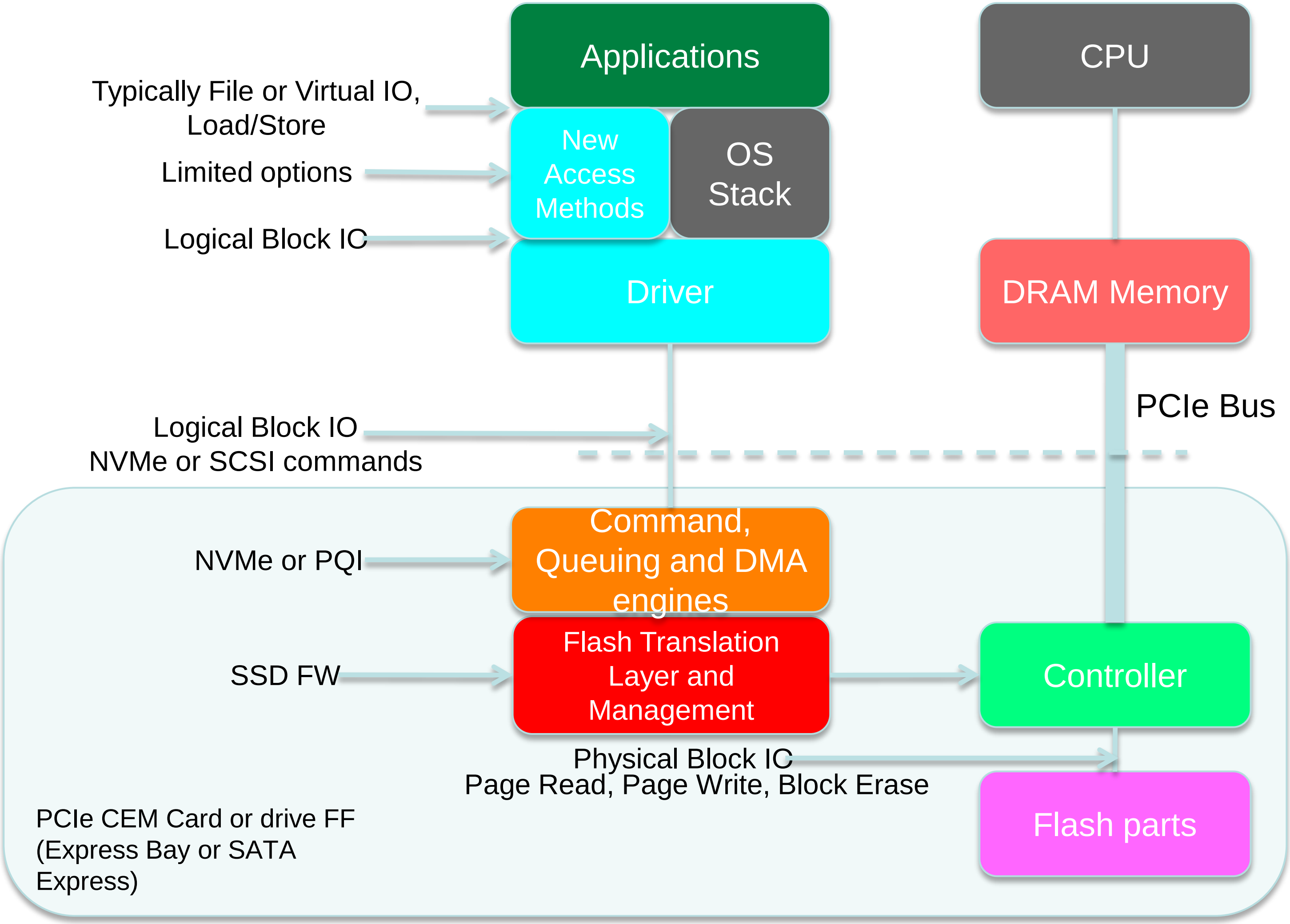
PCIe RAID Aggregation Architecture



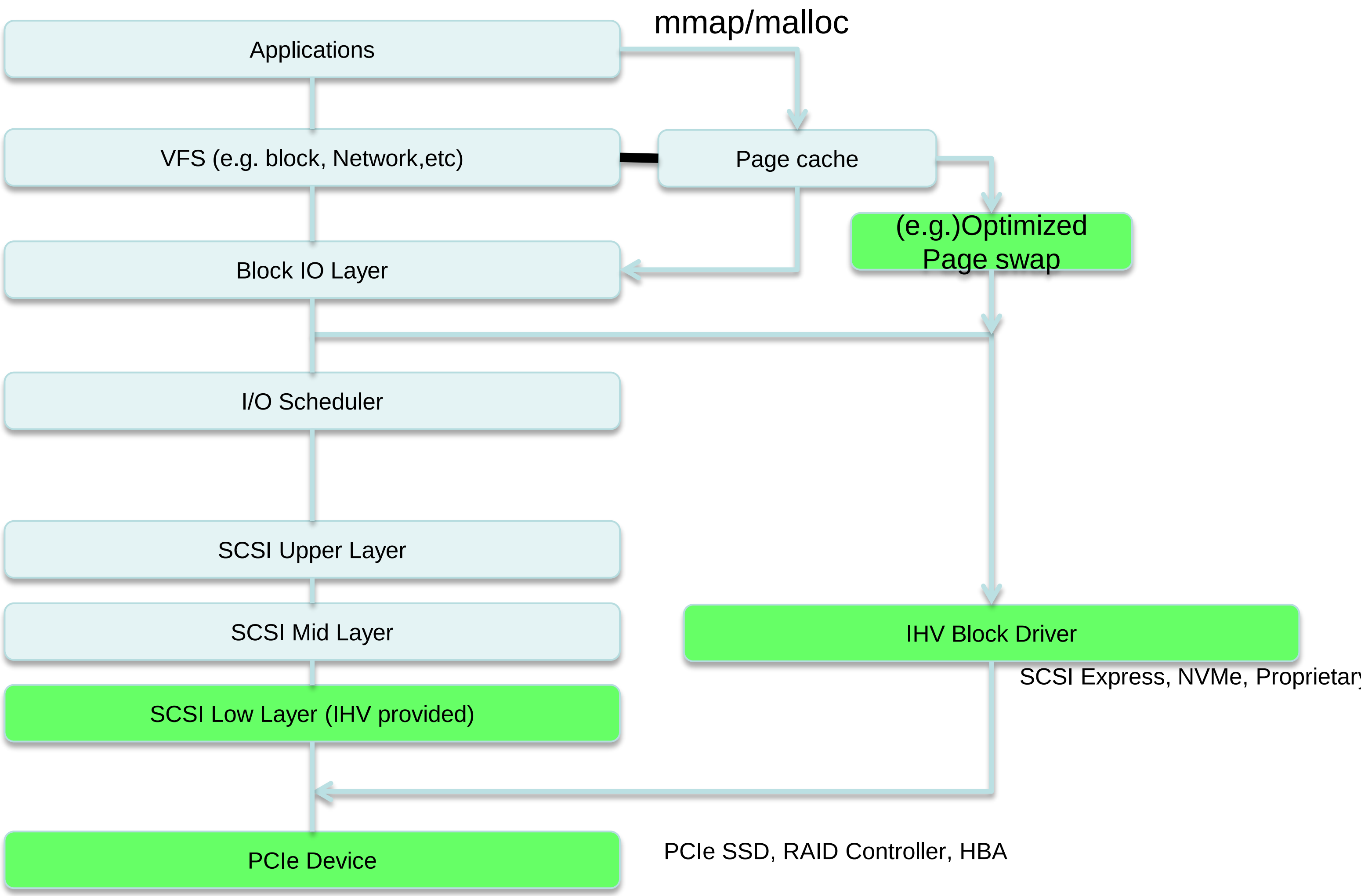
On-load SSD Architectures



NVMe and SCSIe Architectures



Linux IO Stack



Windows Storage Stack

Applications	
IO Subsystem	
File System (e.g.NTFS)	
Volume Snapshot	
Volume Manager	
Partition Manager	
Storage Spaces	SW RAID
SCSI Class Driver	Used for Virtually all Storage Interfaces
MPIO Driver	Multi-path
Storport	Storage Driver API
IHV Storport Driver	SCSI Express, NVMe, Proprietary
PCIe Device	PCIe SSD, RAID Controller, HBA

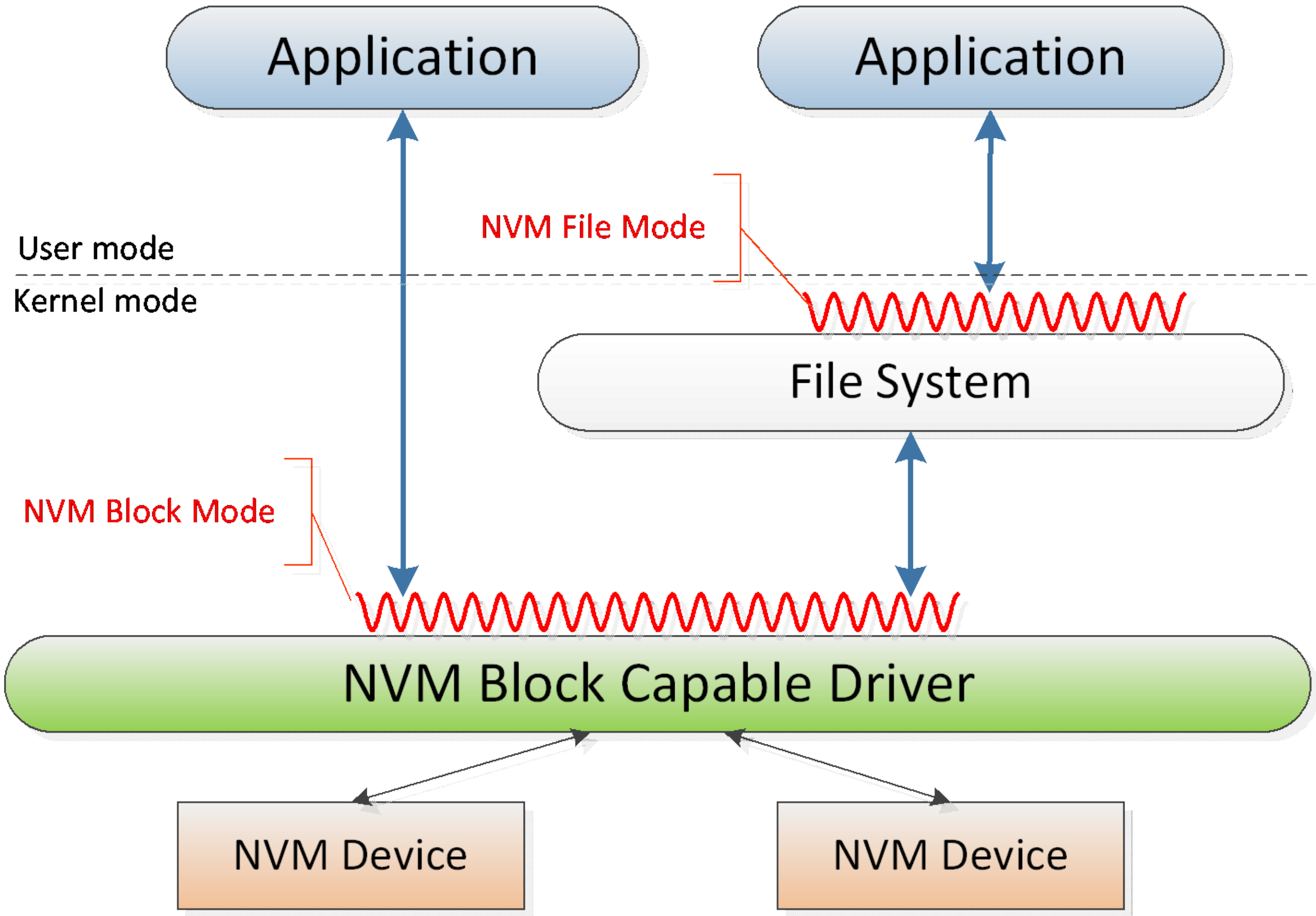
Promises of new NVM Technologies vs Flash

- High performance and low cost
  - e.g. Phase Change, Memristor, RRAM
  - Near DRAM performance
- Dramatically improved endurance
- Better scalability than current NVM solutions
  - Replacement for flash (when cost/bit is close to parity)

NVM Programming Models

- SNIA TWG (NVMP) preliminary specification, V1.0.0 Revision 5, available on SNIA.org
- NVM Block Modes
  - Consistent with current block based storage stack architectures, interfaces and protocols
    - Can use flash or new NVM technologies
- NVM PM Modes
  - Utilizes memory which can be addressed using a load/store model
    - New NVM technologies

NVM Block Interfaces

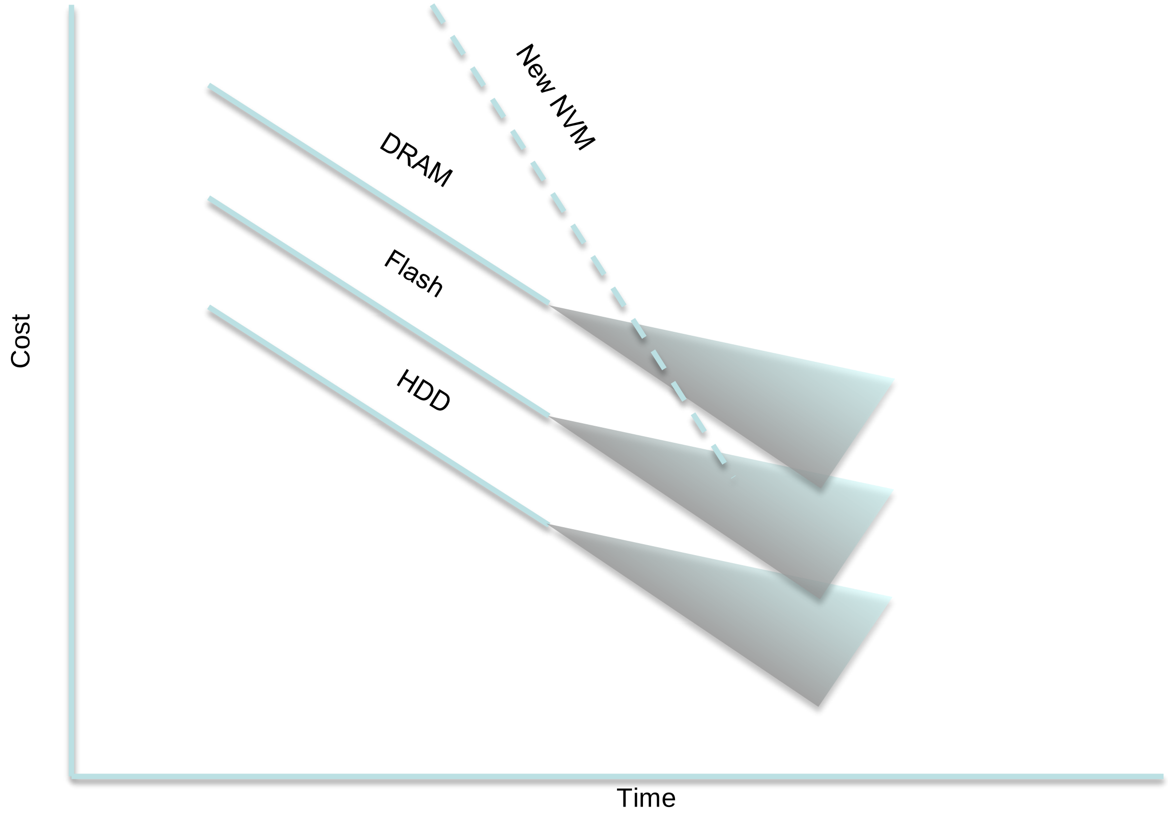


Source: SNIA NVM Programming Model V1.0.0 Revision 5

NVM Extensions

- Leverage existing OS system constructs
- NVM Block Mode and File Mode Extensions
  - Discovery and use of atomic write/read and discard features
  - The discovery of granularities (length or alignment characteristics)
  - Discovery and use of per-block metadata used for verifying integrity
  - Discovery and use of ability for applications or kernel components to mark blocks as unreadable
  - Potential use of memory mapped files

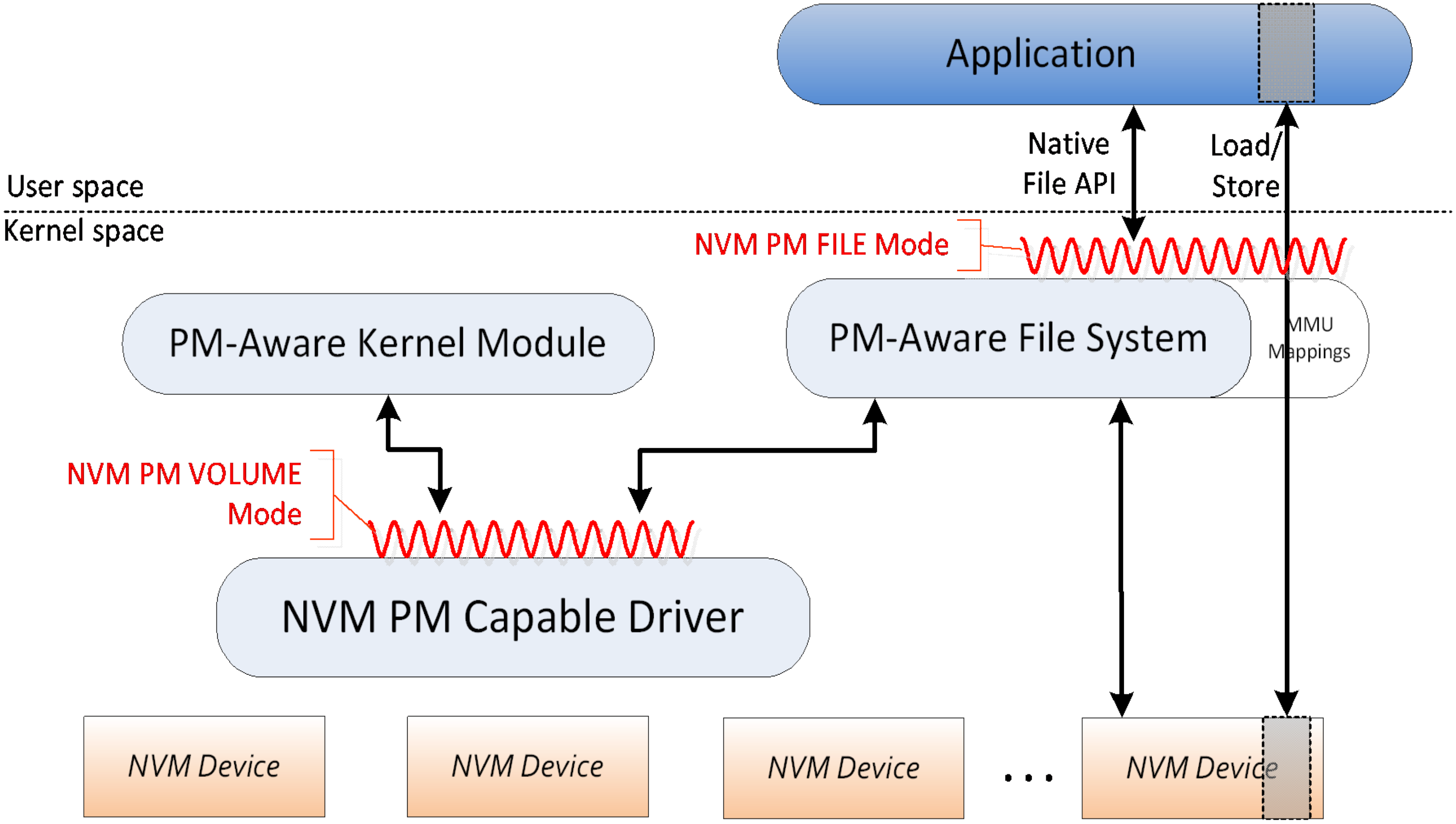
Relative Cost vs Time Chart



NVM Persistent Memory (PM)

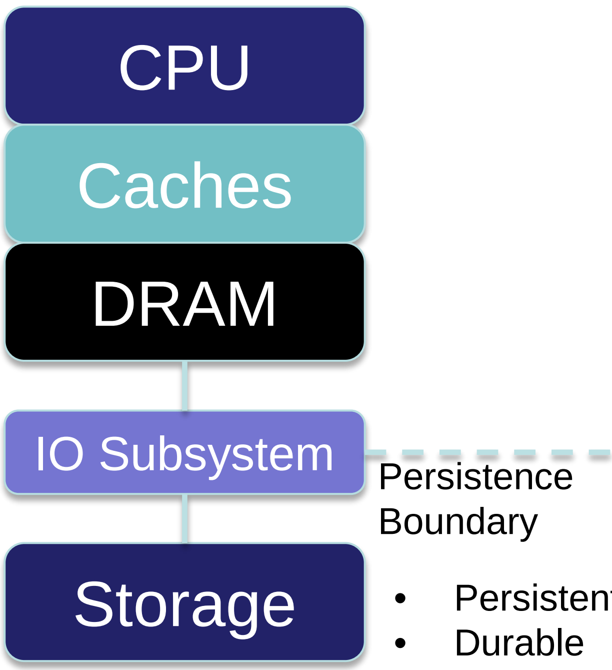
- Memory capable of Load/Store operations
- Does not cause context switching
- NV DIMMS today – New NVM devices in the future
- How to utilize PM in systems

NVM Persistent Memory (PM)Interfaces



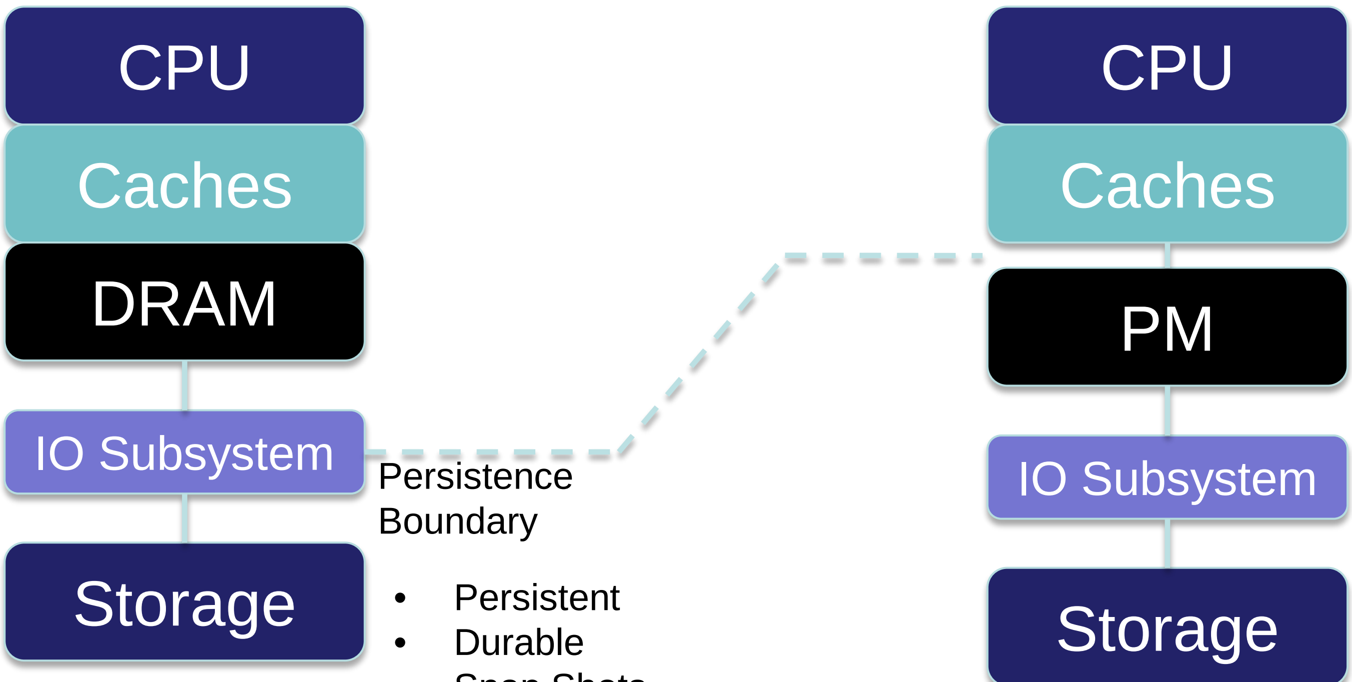
Source: SNIA NVM Programming Model V1.0.0 Revision 5

Shifting the Persistence Boundary



- Persistent
- Durable
- Snap Shots
- Multi Access
- Access Control
- Error handling
- Management
- Security

Shifting the Persistence Boundary



- Persistent
- Durable
- Snap Shots
- Multi Access
- Access Control
- Error handling
- Management
- Security

NVM PM Extensions

- Again, build upon existing OS interfaces
- NVM PM Volume and PM File extensions
  - Discovery and getting attributes; address ranges, connection channel, etc
  - Memory map (w/options), sync, and discard functions
  - Error handling

What’s the Right Physical Interface?

- PCIe
  - Memory mapped IO latency is long compared to the device access time
- DDR4
  - Excellent speed but limited configurations and channels
  - Existing virtual memory management assumptions may not map well into PM requirement
- New Bus
  - Needs to be more expandable
  - Possibility of adding a management layer to better deal with errors and media issue

Device Management

- Endurance – Promised to be much better than Flash, but is it good enough?
  - Perhaps for some applications, but not all
  - Will need a high performance virtualization layer
    - Integrate into processor virtual memory architecture
    - Even though devices capable of fine granularity writes, management likely to be memory page granularity or larger
      - Should handle grown defects without rebooting
    - Less frequent and simpler wear leveling approaches needed
    - SW flexibility vs hardware performance
- ECC – Needs to be very low latency, similar to hamming codes
  - May need periodic scrubbing

Application adoption

- Use of these new methods is not transparent to an application
  - E.g. Memory mapped files are not commonly used today
  - Applications need to worry about consistency
- Cost and multi platform portability will continue to be the key factors in architectural choices
  - Initial adoption by lead users where return is worth the extra effort/competitive edge
- What features of storage are applications willing to compromise
- Security concerns

Summary

- Conventional architectures can benefit greatly in performance with new NVM technologies, but will only see adoption when cost is close to Flash
- New NVM technologies will likely see early adoption as PM, requiring changes to applications and OSs to fully take advantage of performance capabilities