Design Considerations When Implementing NVM

Jim Pinkerton

Architect

Microsoft Windows Server

1/29/2012

Why is NVM Interesting to Microsoft?

- New levels of performance for applications & OS
- Lower storage costs
  - Extreme IOPs of NVM, when combined with the capacity of HDD through storage tiering gives excellent $/IOP while maintaining great $/GB
  - Power reduction
- New compelling distributed applications
  - Example: Distributed “in-memory” databases that are now non-volatile
  - Example: Remote storage with SMB3 RDMA has the same performance as local
    - **Windows Server 2012 SMB3: 16.8 Gbytes/s, 560K 8 KB IOPs from a single client to SSD storage**
    - But as NVM technology is deployed, the bar is raised.
- Portability and new form factors for devices

Microsoft is joining the NVM Programming TWG

We’re more than just an OS

- Device, Client OS, Server OS, Cloud OS, Applications -

Microsoft will be represented by:

Lee Prewitt - Senior PM
Windows Core - Storage & File Systems
20+ years in storage, driver & standards experience.

Spencer Shepler – Architect
Windows Server
20+ years in file sharing (NFS), storage, networking, and IETF NFSv4 co-chair.

NVM Programming TWG engagement

- My background relevant to NVM
  - Co-Chair of the InfiniBand Software Working Group of the IBTA
    - WG defined InfiniBand Verbs – the semantics of the interface to the InfiniBand “NIC”
  - Co-Chair of the RDMAConsortium
    - WG defined the iWARP Verbs – the semantics of the interface to the Ethernet/TCP “NIC”
- Verbs approach worked
  - Created a common semantic language for the interface definition
  - Allowed many different OS’s/Applications/Hardware Vendors/Systems Vendors to converge on the semantics, without the distraction of mapping an API to a specific operating environment
  - Solved both kernel mode and user mode access

The Rest of the Talk…

Focus is a broad set technologies

- NVDIMM
- NVME
- PCIE Attached NVM
- Proprietary
- Next generation work
  - Memristor, phase change RAM, Millipede, STRAM, ….

My personal turmoil

- Systems view of the problem
  - Only some portions of it are in scope for the SNIA NVM Programming TWG
- I'm a server guy
  - Thus initial thoughts are from a server perspective
- Microsoft defines "storage" as block and file…

Some Characteristics of Existing Non-Volatile Storage (from an App & Ops Perspective) 1/2

- Device Life Cycle
  - Failures
  - Replacement
  - Disposal
- Data Life Cycle
  - Backup
  - Replication
  - Security (privacy, etc..)
  - Data Integrity

- Application Semantics – which mode?
  - **Block mode** enables 100% app compat, unleashing compelling new capabilities
  - **Byte mode** (memory mapped files) has the potential for much higher IOPs, but what is the semantic of the interface?

What to support for NVM?

What to not support?

Some Characteristics of Existing Non-Volatile Storage (from an App & Ops perspective) – 2/2

- Errant program does not destroy all data
  - Most do not memory map a file
- There is “infinite” memory
  - Do we map paging file semantics? How does the OS tie into the infrastructure?

What to support for NVM?

What to not support?

Some Characteristics of Existing Non-Volatile Storage (from an OS perspective) - 1/2

- Write error atomicity
  - Traditionally 512 byte or 4 KB, and … – some apps/file systems will break with smaller atomicity
- Transactional Semantics
  - Traditional approach:
    - Flush/barrier semantic that encompasses full path to non-volatile storage
  - New approaches?
    - Is there a need for native atomics?
  - Are transactional semantics (i.e. flush with logging) required to interface to NVM?
    - This is extremely complex to get right.
- Defragmentation for optimized access – tension between
  - Page translation tables, caching, prefetch optimizations, …
  - Bin-packing of application data in NVM as files are deleted, added, extended…

Some Characteristics of Existing Non-Volatile Storage (from an OS perspective) - 2/2

- Virtualized Resources
  - Traditional Hyper Visors virtualize everything and then live migrate.
  - What is the model for NVM?
- Data Availability
  - Traditional semantics are either synchronous (i.e. every write is mirrored) or asynchronous (often done after file close). If we don't have synchronous, and the file is open indefinitely….
- Data Reduction
  - Data deduplication, data compression, …
- Security
  - How to map data encryption capabilities? Seems required for NVM?

Filter Drivers provide a ton of value

Filter Driver Examples on Windows

- **Data Modifying Examples**
  - Encryption
  - Data Checksums
  - Deduplication
  - Compression
- **Data Monitoring in I/O Path Examples**
  - Anti-Virus
  - Replication
  - Continuous Backup
  - Activity Monitor
  - Content Screener
- **Prior Versions of Files Examples**
  - System Recovery
  - Hierarchical Storage Management
  - Open File (catch all category)
- **Other Examples**
  - Undelete
  - Logical Quota Management
  - Physical Quota Management
  - Cluster File System
  - Imaging (containers)
  - Virtualization of file paths
  - Security Enhancer (beyond ACL)
  - Copy Protection

Which Path to Take?

- Do we define a new device model for the OS or leverage existing?
  - For **Block Mode**:
    - NVME provides an optimized block storage model – but is incompatible with other technologies like NVDIMM
  - For **Byte Mode**:
    - There is no common device model in general purpose OSs today
  - Do we need to do both?
- Do we define a new semantic for applications or reuse existing?
  - For **Block Mode**:
    - No need – should be 100% compatible
  - For **Byte Mode**:
    - Seems required to unleash full performance

In Closing…

- This was a “systems view” – will work within the NVM Programming TWG to map this to achievable deliverables in the scope of the TWG

- NVM technology can be both evolutionary and disruptive…
  - We should do both
    - this could be fun…