Ensuring Data Availability for NVMe-based Storage

Jeff Plank
Trends in Primary Enterprise Storage

- RAID remains important
  - OS drives (typically RAID 1)
  - Primary storage in enterprise deployments (RAID 0/1/10/5/50/6/60)
  - Caching for HDDs (RAID 0)

- NVMe enterprise SSD volumes are fast becoming the preferred storage media for capacities <4 TB*
  - Ecosystem barriers for NVMe primary storage in enterprise servers are coming down
    - Enterprise reliability (write endurance)
    - Serviceability (hot plug, surprise plug)
    - Standardization (including UBM and universal bay)
    - Technology for ubiquitous connectivity of SAS/SATA/NVMe (tri-mode)

**RAID for NVMe SSDs will be a mandatory portfolio offering for enterprise servers by the end of the decade**

*Source: IDC*
Options to Deliver RAID for NVMe

- There are two system-level options to deliver NVMe RAID
  - Host CPU through software RAID implementation
    - Using CPU/chipset resources to provide parity and redundancy
  - Controller-based hardware RAID
    - Using offload capabilities of an ASIC to generate parity and redundancy
- There are multiple implementations of controller-based hardware NVMe RAID
Evolving RAID for NVMe Architectures

- Lowest latency path
- Consumes compute and memory BW on the host CPU

- Adding a controller to the path impacts latency for read/write operations
- Offloads compute and memory bandwidth from the host CPU

- Traditional software and hardware data protection each have their advantages

- Data resiliency with high IOPs, low latency, and the benefits of hardware offload is possible for PCIe-attached drives with the appropriate silicon and firmware architectures
NVMe RAID Nirvana

Combining a multi-path driver with an embedded switch in the controller unlocks the best of both worlds for PCIe attached drives

- Full hardware offload of rich data services
- Low latency
Multi-Path Driver

- The multi-path driver
  - Opportunistically issues standard NVMe to the drives
  - Forwards commands to the rich data services engine when required

- Able to build native NVMe messages just as efficiently as a native NVMe driver (instructions/cycles)

- Path selection is simple
  - Array in good state?
  - Single column IO?
  - Read or non-parity write?

Yes to all? → Use the direct path to the NVMe drive
Consider a multi-path NVMe driver that issues commands
• Directly to NVMe drives through the PCIe switch when offload is not required
• Directly to a storage controller that presents itself as a logical NVMe drive attached to the switch
Thank You