SDC 19

September 23-26, 2019
Santa Clara, CA

NVMe based Video and Storage Solutions for Edge based Computational Storage

John Plasterer
NETINT Technologies

Introduction to Video Encoding at Scale

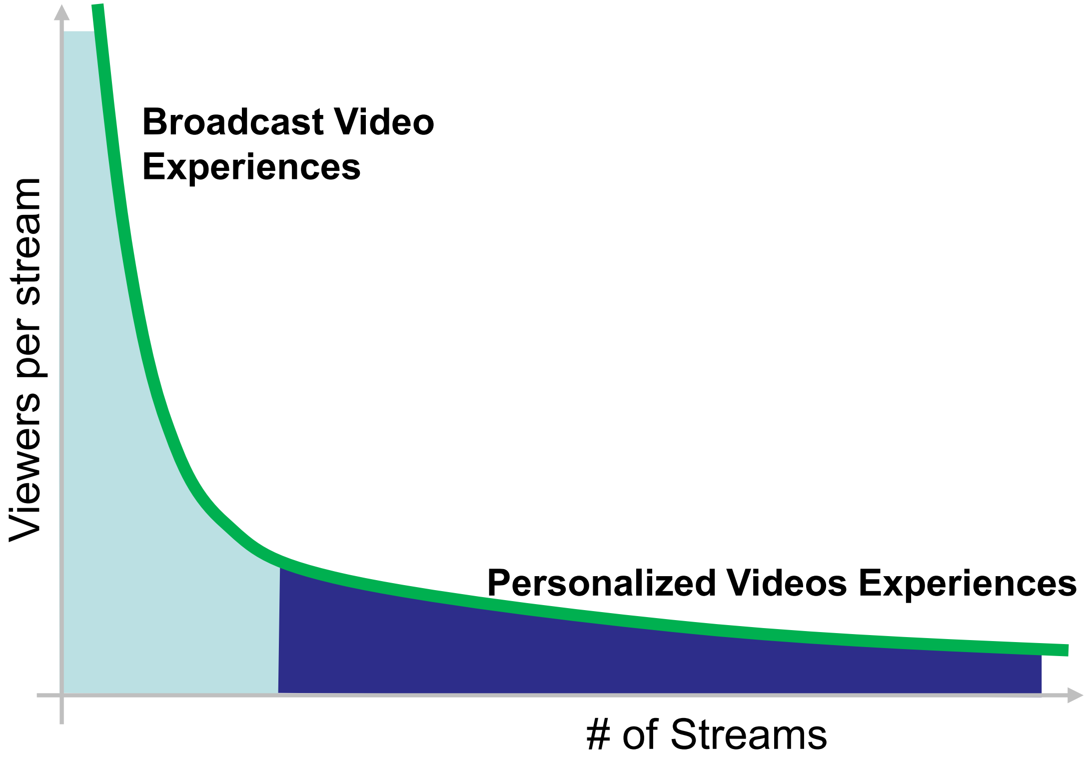
Video Distribution in the 60s

Video Distribution in the 90s / 2000s

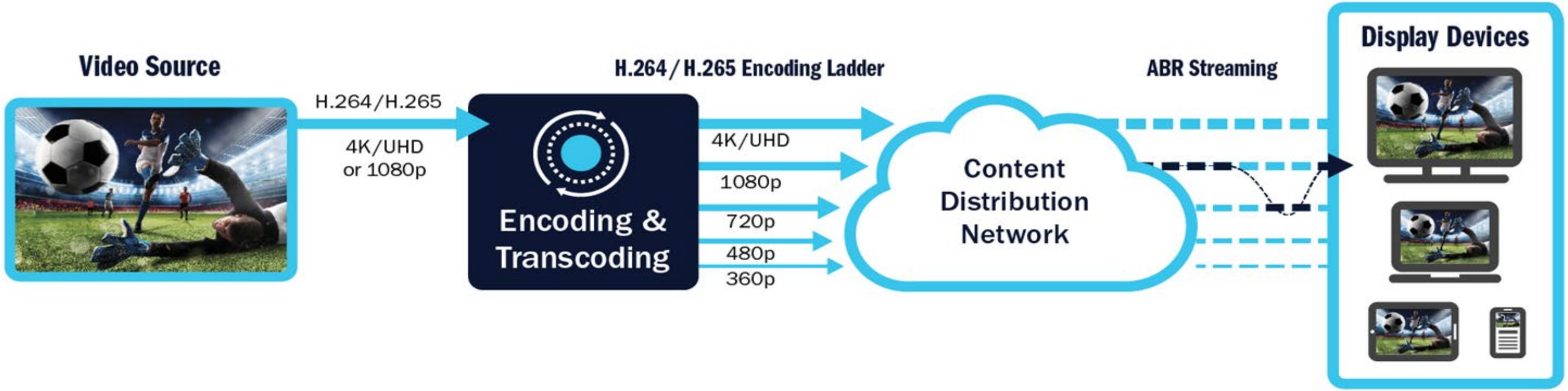
Video Distribution in 2010s

Video Distribution Now

Video Experience Distribution



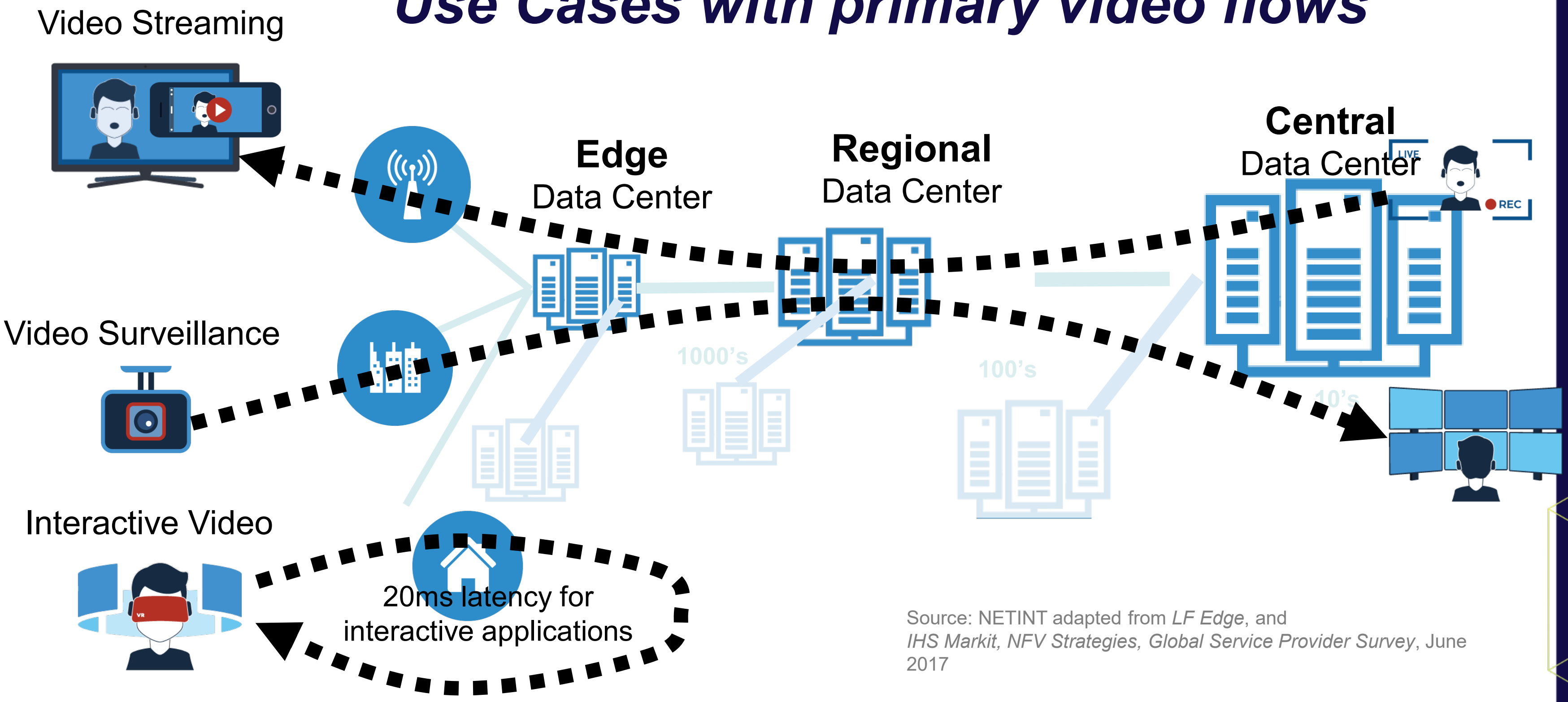
Video Transcoding for end application



- Video needs to be distributed in many formats
  - Instantaneous viewing at multiple resolutions

Video Edge Encoding and Storage in the Video Cloud

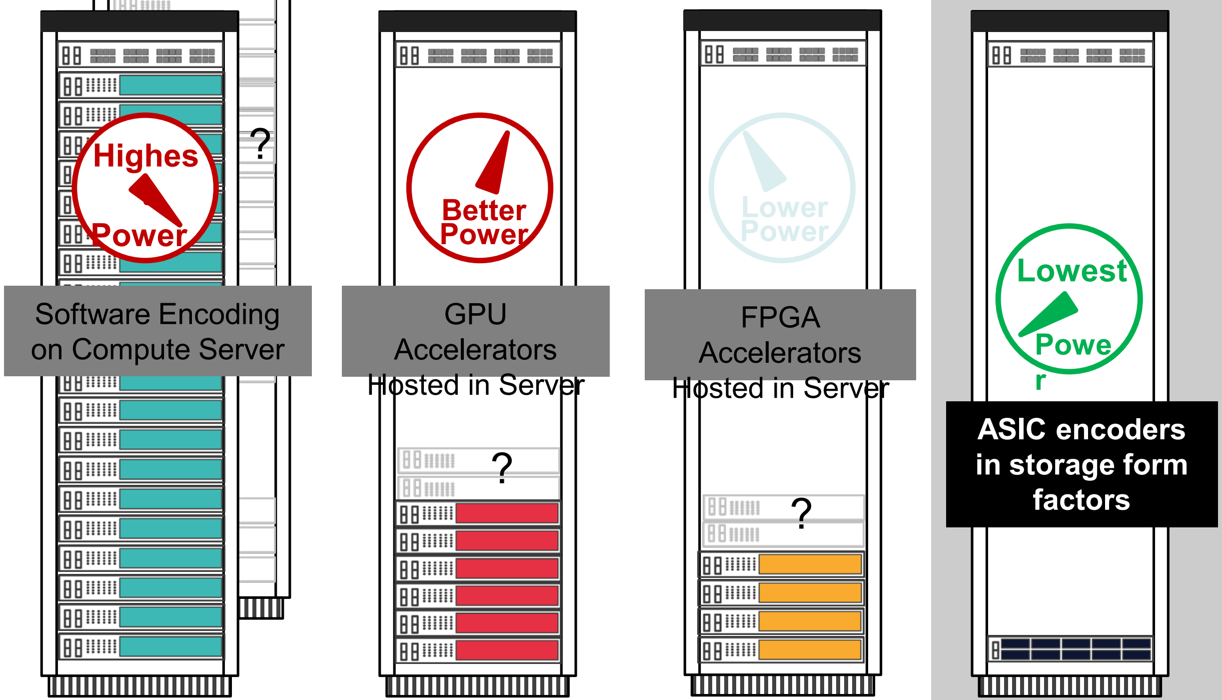
Use Cases with primary video flows



Source: NETINT adapted from *LF Edge*, and *IHS Markit, NFV Strategies, Global Service Provider Survey*, June 2017

Video Encoding Alternatives Compared: Density and Power

Approximated infrastructure required for
80x 1080p30 Encoding Streams, or
40x Typical Encoding Ladders.



- NVMe Interface
- ASIC-based Transcoding
- Highest Density
- Least Rack Space
- Lowest Power

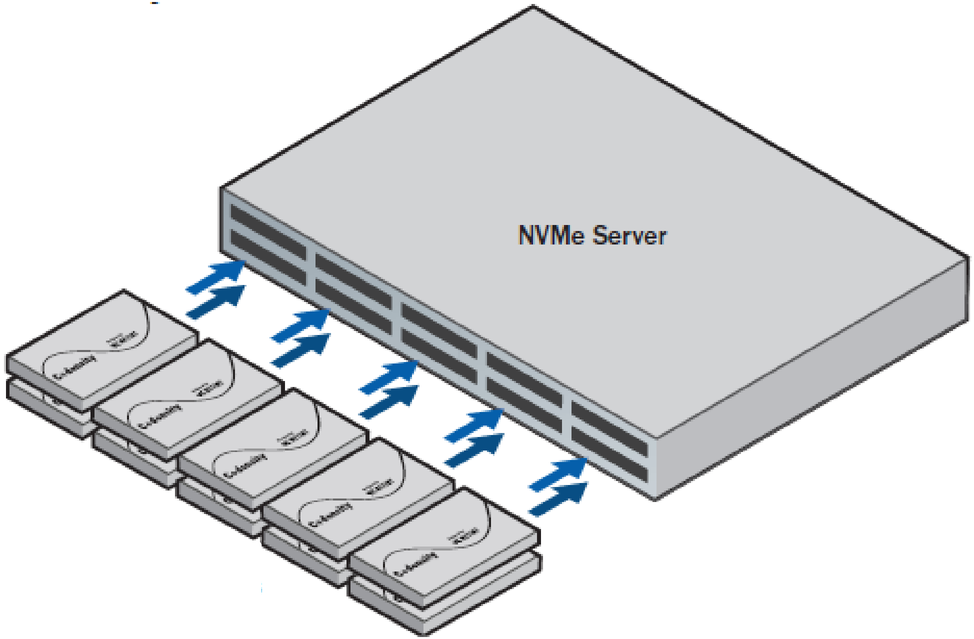
Implementation of Video Encoding using NVMe

Solution Requirements

- Fast time to market to capture fast moving live video market
- Needs to use robust, highly tested infrastructure as much as possible
- Needs to be deployable quickly by customers

Why use a storage form factor?

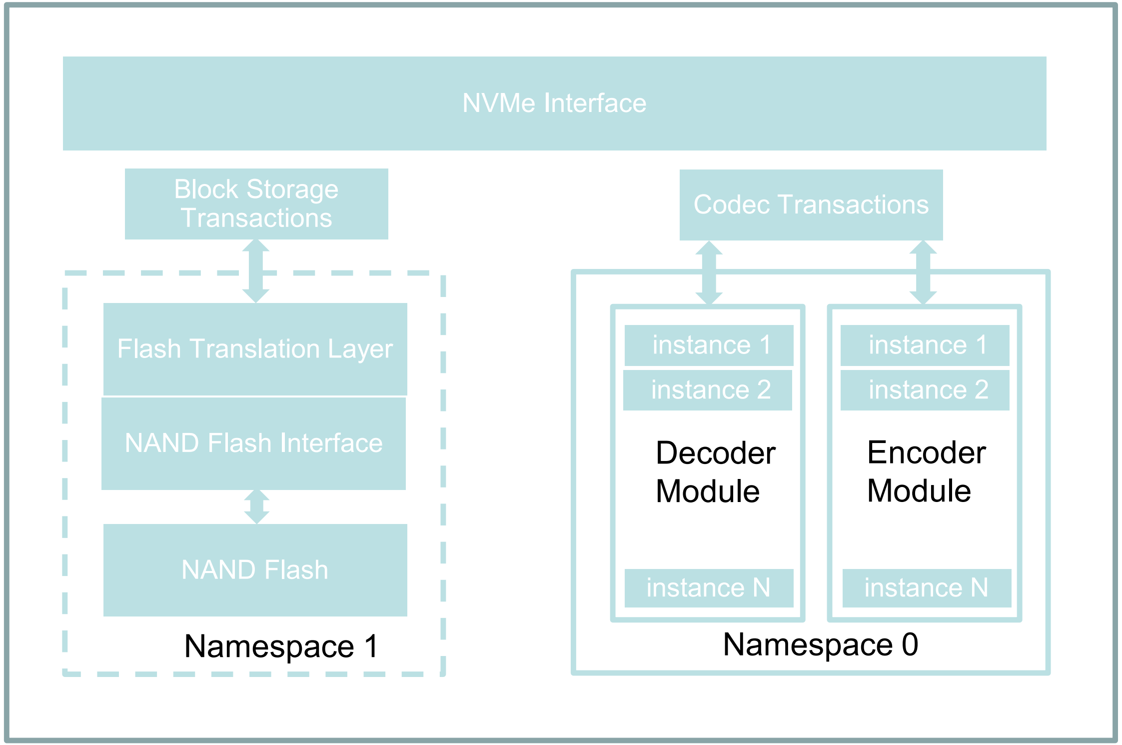
- Using storage interface allows scaling using standard server infrastructure
- Transcoding U.2 modules plug into SSD slots of NVMe Server



Why use NVMe?

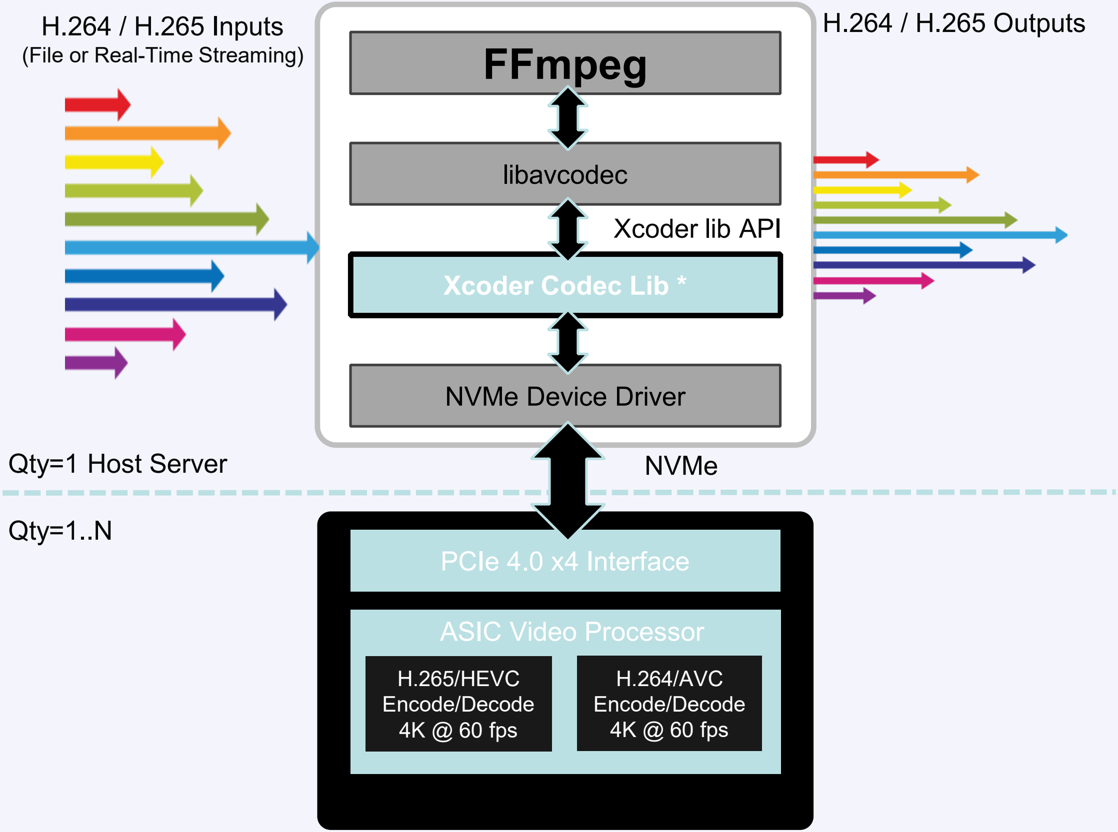
- Easily combine storage and video into the same PCI-Express Interface
- Leverage significant amount of industry investment in NVMe
  - Kernel
  - Drivers
  - Hardware

Application of NVMe to control SSD and video processing



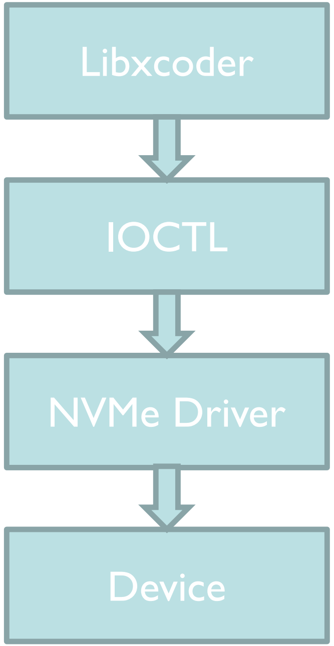
Video Transcoder – Software Integration

- FFmpeg integration achieved by installing FFmpeg Codec Lib and SDK into host server
  - Seamlessly abstracts FFmpeg video transcoding functions from 1 or more transcoder modules
- Video transcode functions controlled through standard NVMe protocol



Vendor Specific Commands

- Vendor specific commands allow a "simpler" implementation
  - Advantages:
    - Simple to architect
    - Simple to implement
  - Challenges:
    - IOCTL path in kernel/driver is not optimized for performance
    - Requires administrative privileges
    - Windows only recently supported vendor specific commands and behavior does not match Linux
    - Is not currently supported by NVMe over Fabrics



Types of commands

Command	Encoder
Xcoder Open	Open a decoder/encoder instance
Xcoder Close	Close a decoder instance
Xcoder Query	Query xcoder for current status
Xcoder Write	Transfers data from host to codec for decode/encode
Xcoder Read	Transfers data from codec to host for decode/encode

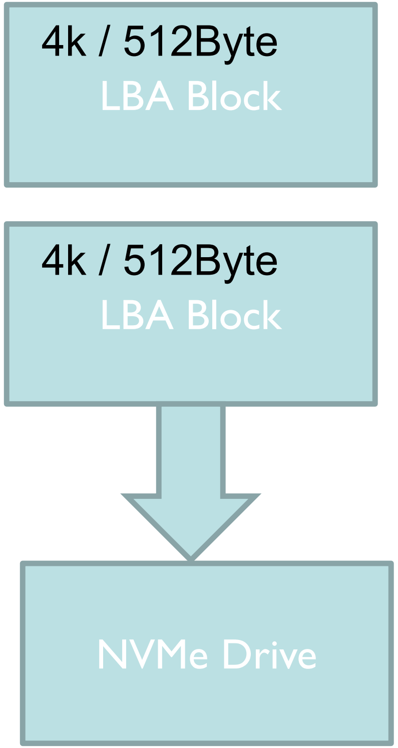
Example Command Structure

Command:	Opcode	CDW10	CDW...	CDW15
Response:	DW0			

- Xcoder Open – Opcode 0xC1
  - CDW10: Xcoder ID, configuration data
  - Completion: Xcoder instance
- Xcoder Write – Opcode -0x83
  - CDW10: Decoder id, instance, format and stream
  - CDW11: Size of data

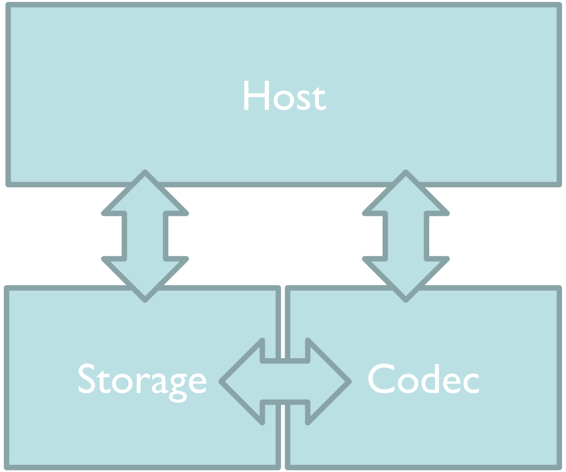
IO Commands for video

- IO commands (block level read write) allow high speed access
  - Advantages:
    - Kernel is highly optimized for block level access
    - Very low latency, high priority
  - LBA structure does not align with the structure of our data
    - Need to "hack" the usage for our device, create new definitions of LBA regions, and access patterns
    - No ability within the command to send configuration information



Codec Directly Interacting with Storage

- Challenges with direct interaction:
  - SSD is LBA based but applications are file based
  - How will the internal SSD know the file system of the OS above?
- Without significant changes at the application layer / OS layer direct storage is not practical
- Requires standardization and changes to kernel for optimal solution



Challenges with Memory Management

- Memory movement is the largest contributor of CPU cycles with this solution
- IOCTL Challenges with Memory Management:
  - IOCTL will perform a memory copy if data is not 512 Byte aligned
  - Memory copy consumes significant CPU usage
- Need to optimize overall memory movement from library to host systems

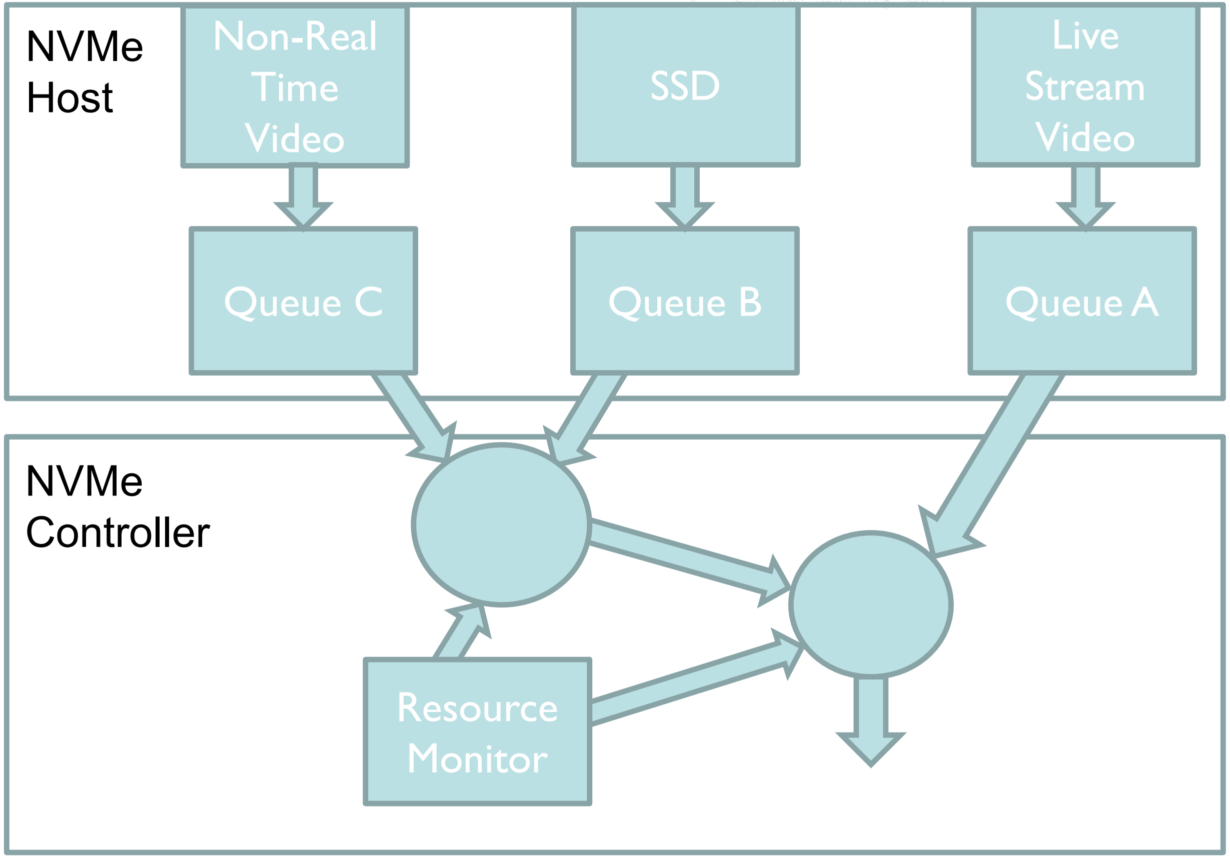
Managing SSD and Video Together

- Video codec and SSD compete for same resources
- Need to guarantee quality of service for both SSD and transcoder
- How to guarantee QOS?

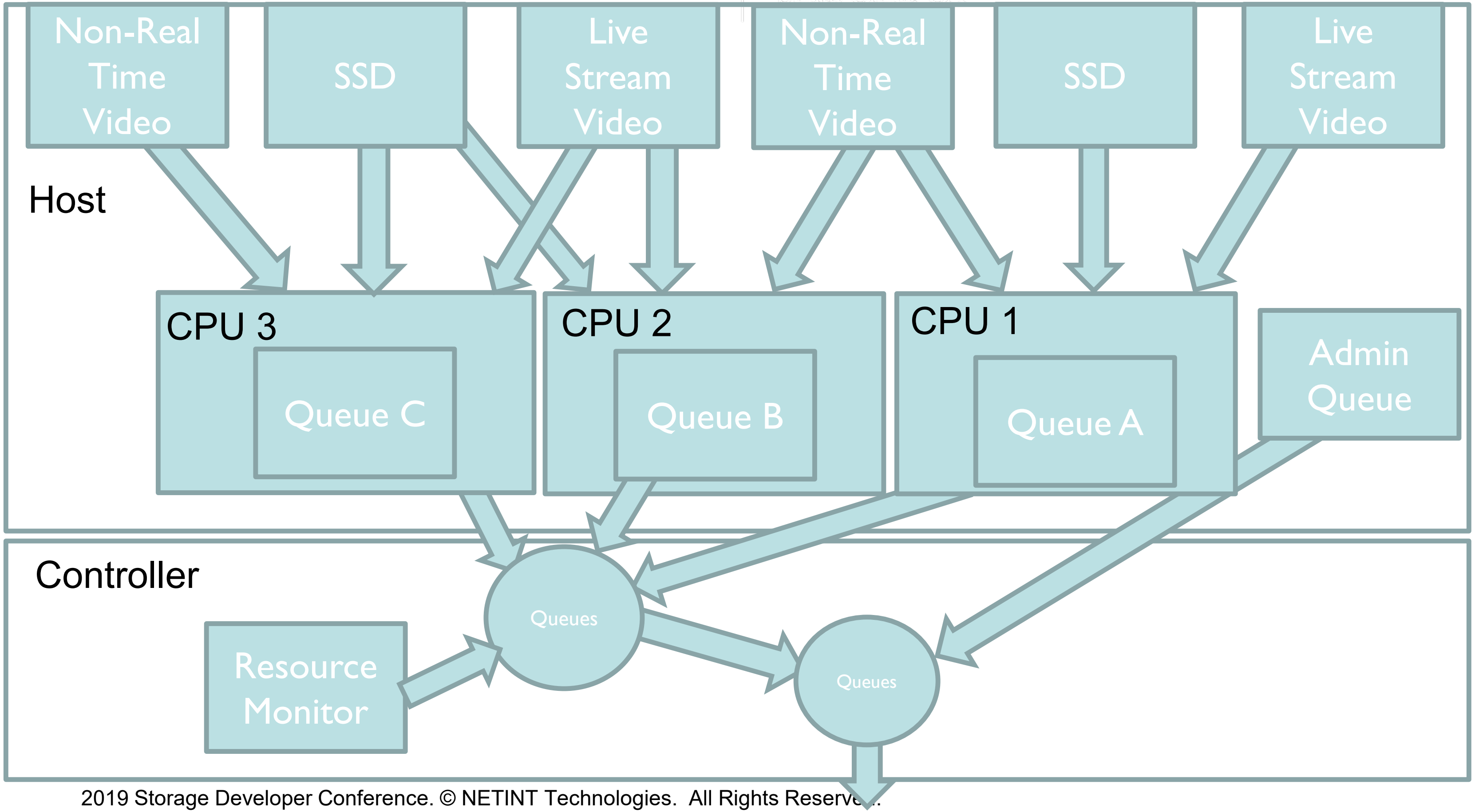
QOS Criteria / Prioritization

- Live stream / real time video
  - Requires uninterrupted service and guaranteed frame rate (i.e. 30fps)
- SSD
  - Requires predictable performance
  - Requires QOS (including 99.99% latency)
- Best effort encoding (Non-Real-time)

Queues for Priority Management

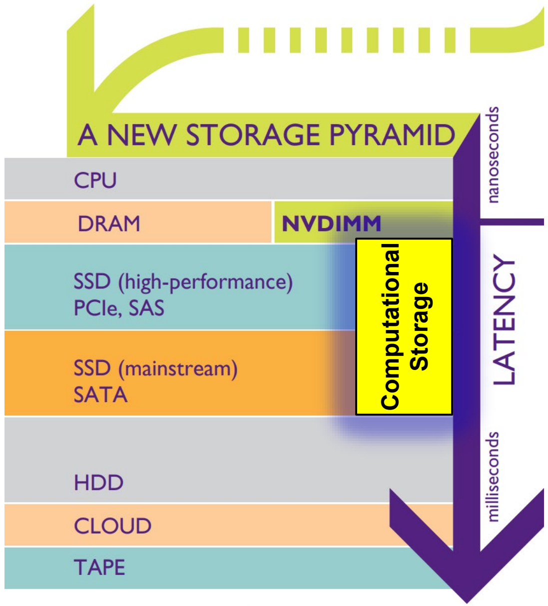


Priority Management Internal



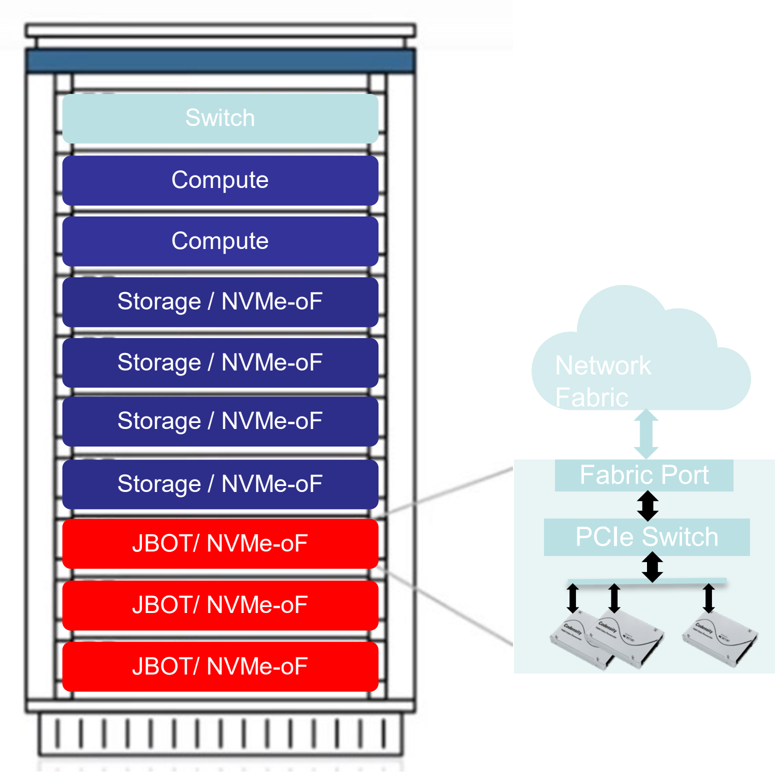
Why we need to Standardize Computational Storage?

- Both vendor specific and block command approach with current NVMe is sub-optimal
  - Is a better approach possible?
- Should rethink OS queues for computational storage.
  - Should computational storage elements get a different queue?
- Items like identification, classification provide host system more information
  - Look like a formal device to host with exposed functionality
- Can we build the hooks to allow file based interactions without host interactions?



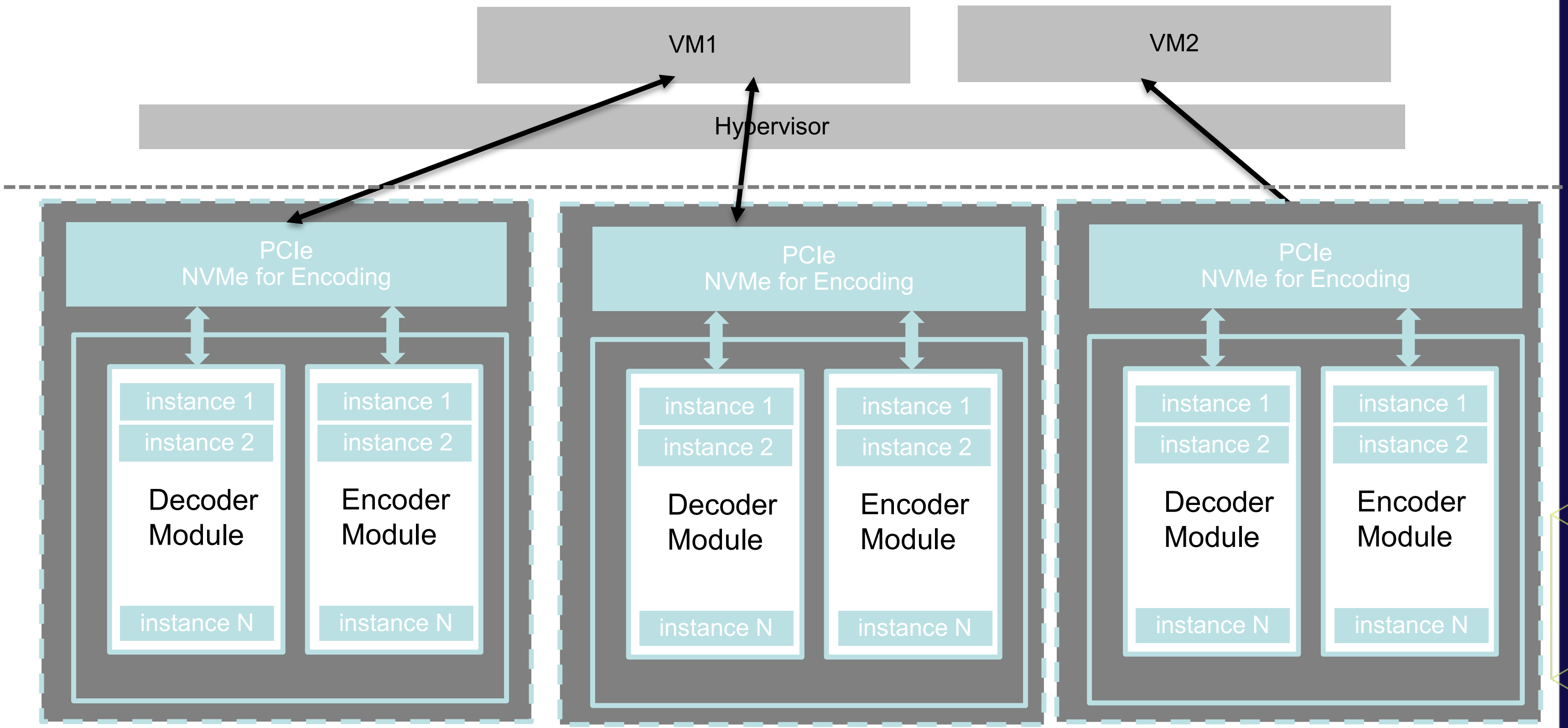
Scaling Video Encoding in the Cloud

Scaling-out Video Transcoding with NVMe-Over-Fabrics

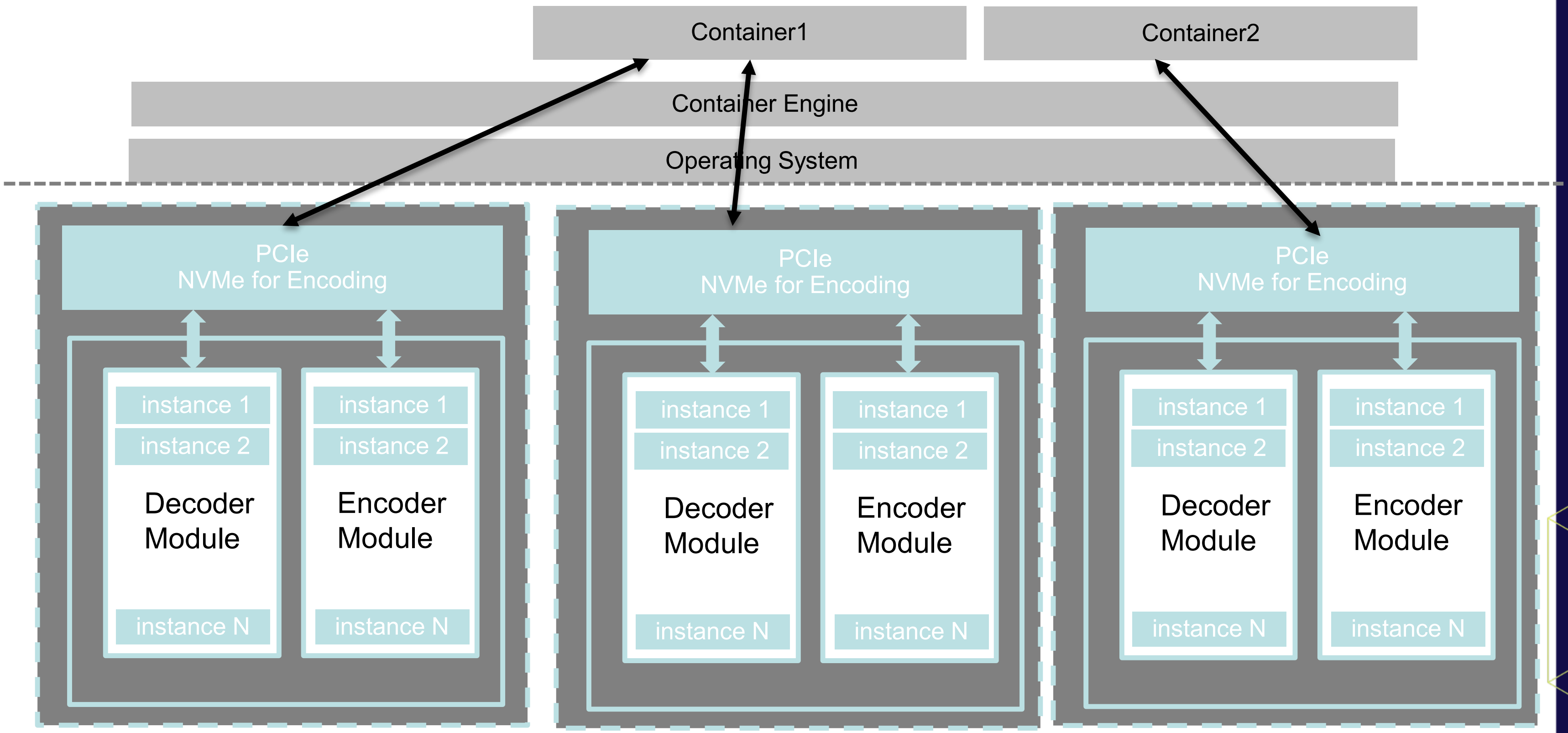


- Work with proven NVMe and NVMe-oF device drivers
- Composable infrastructure
- Just a Bunch of Transcoders (JBOT)
- Sharing video transcoding resources among servers

Virtualization for Cloud – Hypervisor

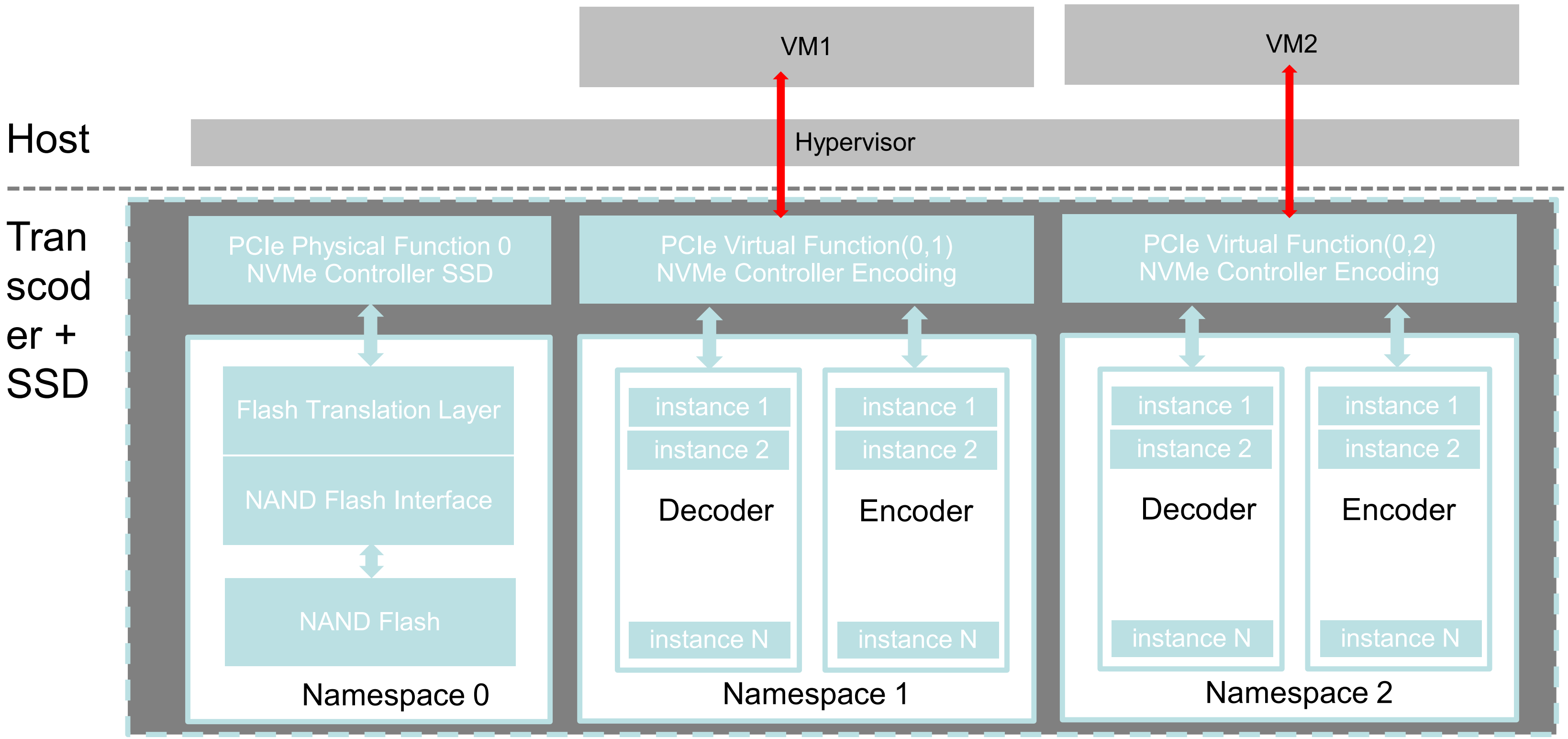


Virtualization for Cloud – Containers



31

Virtualization for Edge with SR-IOV Share One among Virtual Machines



Questions?