SNIA Tutorial: PCIe Shared I/O

Jeff Dodson / Avago Technologies

SNIA Legal Notice

- The material contained in this tutorial is copyrighted by the SNIA unless otherwise noted.
- Member companies and individual members may use this material in presentations and literature under the following conditions:
  - Any slide or slides used must be reproduced in their entirety without modification
  - The SNIA must be acknowledged as the source of any material used in the body of any document containing material from these presentations.
- This presentation is a project of the SNIA Education Committee.
- Neither the author nor the presenter is an attorney and nothing in this presentation is intended to be, or should be construed as legal advice or an opinion of counsel. If you need legal advice or a legal opinion please contact your attorney.
- The information presented herein represents the author's personal opinion and current understanding of the relevant issues involved. The author, the presenter, and the SNIA do not assume any responsibility or liability for damages arising out of any reliance on or use of this information.

NO WARRANTIES, EXPRESS OR IMPLIED. USE AT YOUR OWN RISK.

Abstract

PCIe Shared IO

- This session will appeal to System Architects, Data Center Managers, and those interested in sharing PCI Express (PCIe) endpoints with multiple hosts. PCIe is the fundamental connection between a CPU's Root Complex and nearly any IO endpoint. By inserting a switch between a Root Complex (RC) and its endpoints, compute and IO can be disaggregated. In addition, with a management RC, multiple server RCs can connect to the same switch and share the IO. This presentation gives an overview of PCIe, then goes into detail about switch features required to share IO, harden a system, and provide high availability.

Agenda

- PCIe overview
  - PCIe enumeration
- Synthetic PCIe configuration space
  - Configuration redirection
  - Reservation and event isolation
- PCIe surprise events
  - Hot add, remove
- Challenges to Scalable Shared IO

PCIe Overview

- Link flexibility
  - Can be x1, x2, x4, x8, x12, or x16
  - Can be 2.5 GT/s, 5.0 GT/s, or 8 GT/s (gen 1, 2, 3)
  - Gen4 16 GT/s coming soon
- Credit based lossless link
- Good background on PCIe covered at Flash Memory Summit 2014

Check out SNIA Tutorial:

PCI Express and Its Interfaces to Flash Storage Architectures

PCIe terminology

- RC: Root Complex
- EP: Endpoint
- Upstream: to RC
  - Subtractive decode
- Downstream: to EP
  - Active decode
- Configuration space
  - Bus numbers
- Memory Space
  - 32b and 64b address

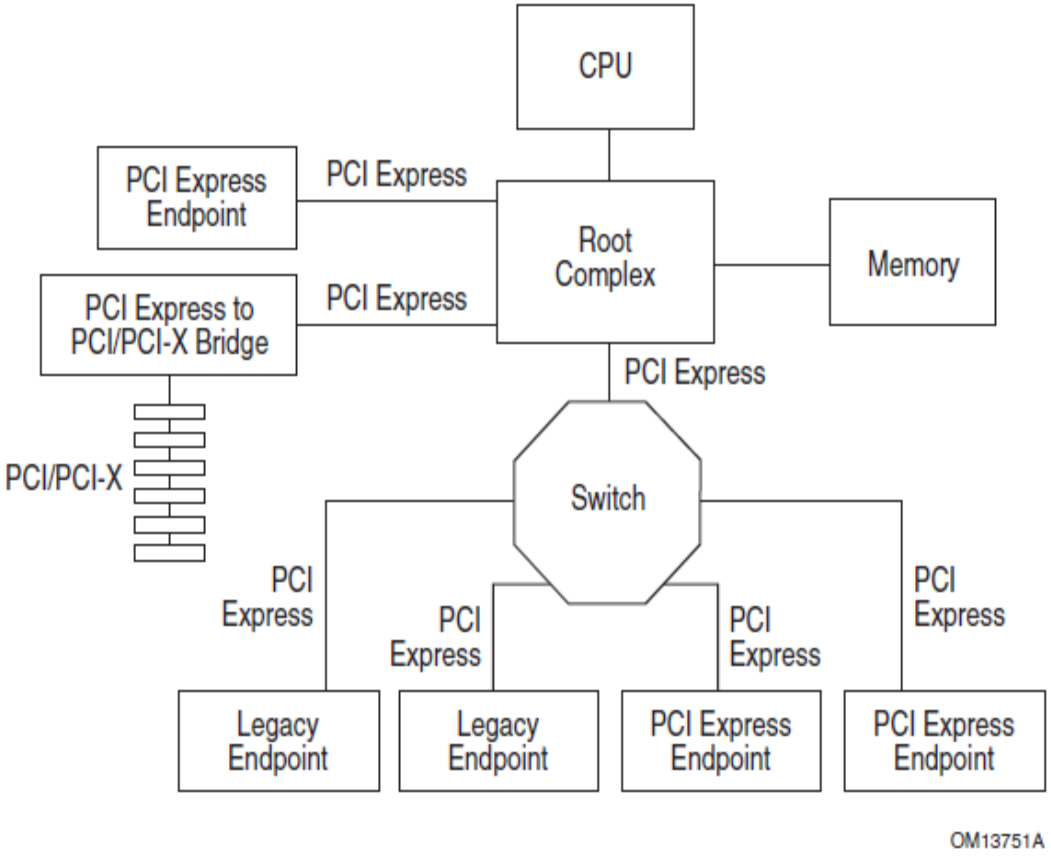
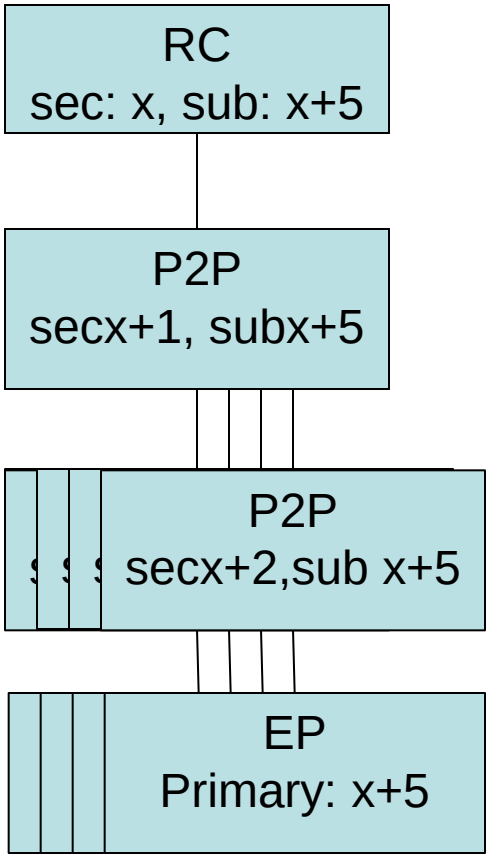


Figure 1-2: Example Topology

PCI Enumeration Steps

- Depth first enumeration
- Use PCIe hierarchy prior slide: switch + 4 EPs
- Set secondary bus to primary + 1
- Set subordinate bus to all 1s (0xff)
- Switch is PCI-PCI bridge hierarchy
- After bottom reached, set sec-sub in each P2P bridge above it
- After devices all found, assign memory map
  - set BAR in EP, base and limit in P2P

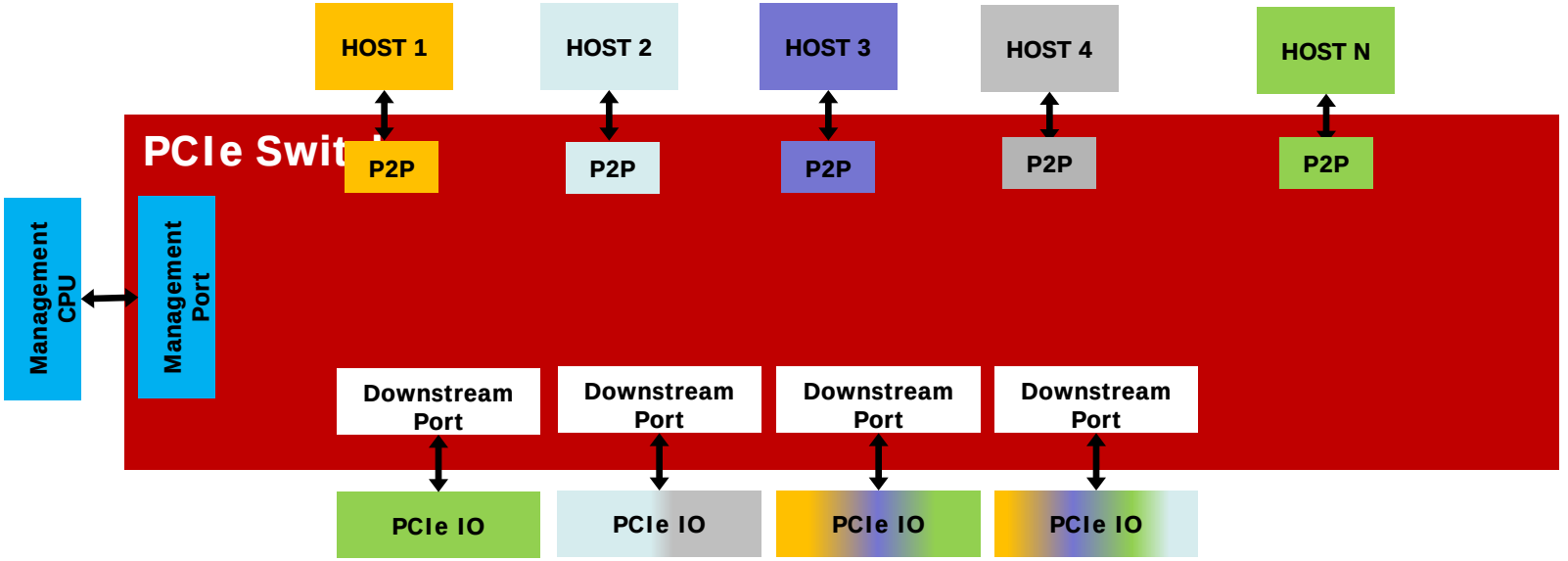


How to support multiple roots?

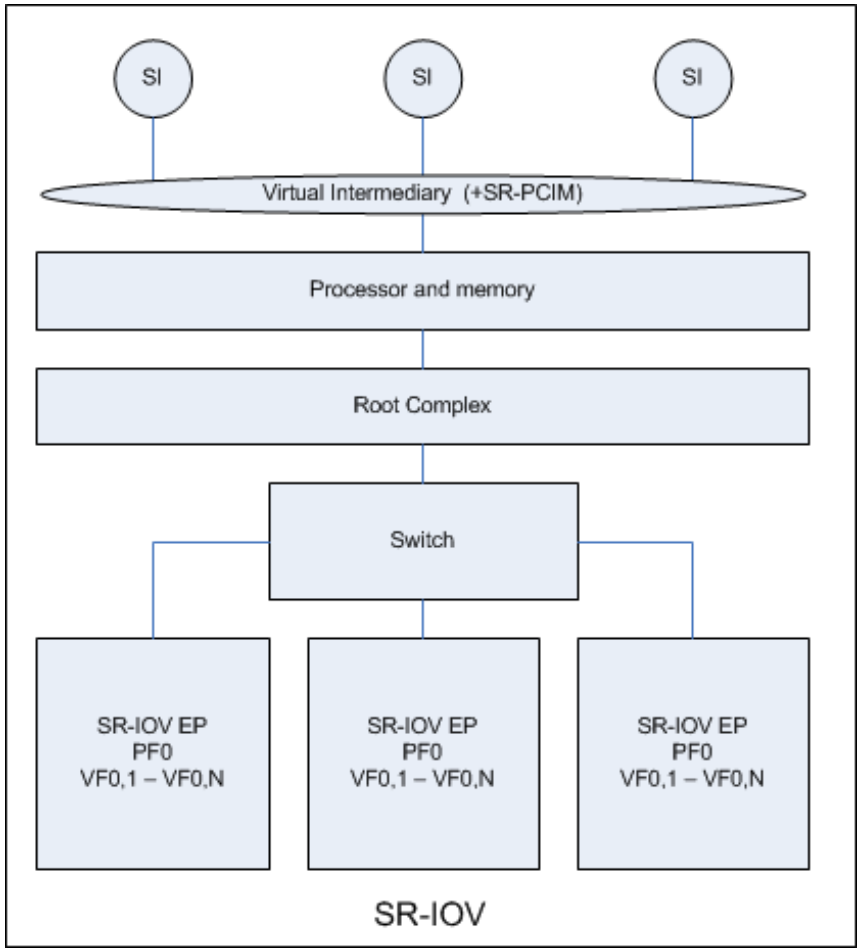
- MR-IOV spec is one way
  - But silicon doesn't exist
  - Burden on endpoint and switch silicon
- Managed PCIe switch is another way
  - Uses (existing) SR-IOV silicon
  - Synthetic configuration for each host

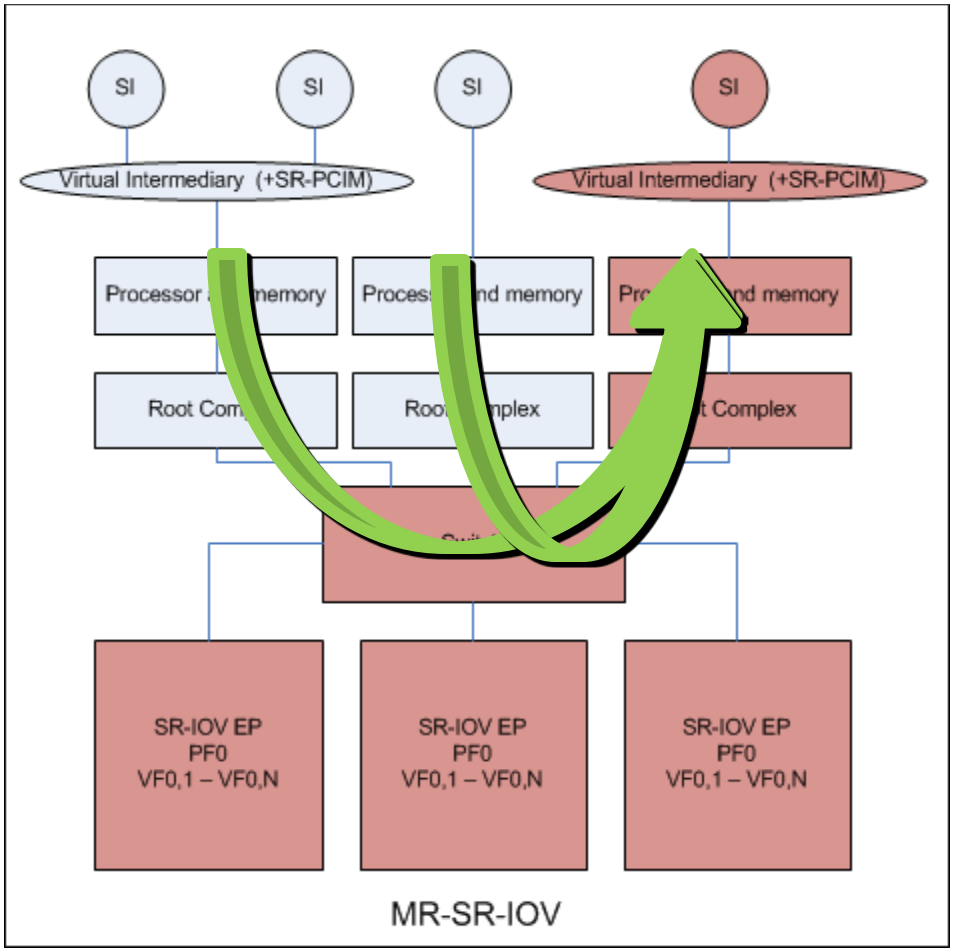
Multi-Host Shared IO

- Transparent sharing of SR-IOV endpoints by multiple hosts
- Management CPU owns the IO, synthesizes hierarchy for hosts



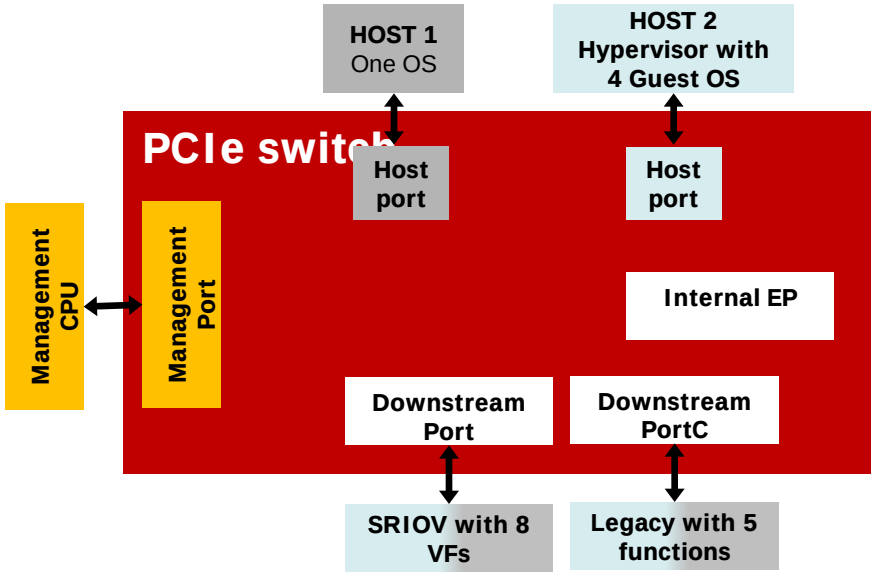
Synthetic Configuration Space





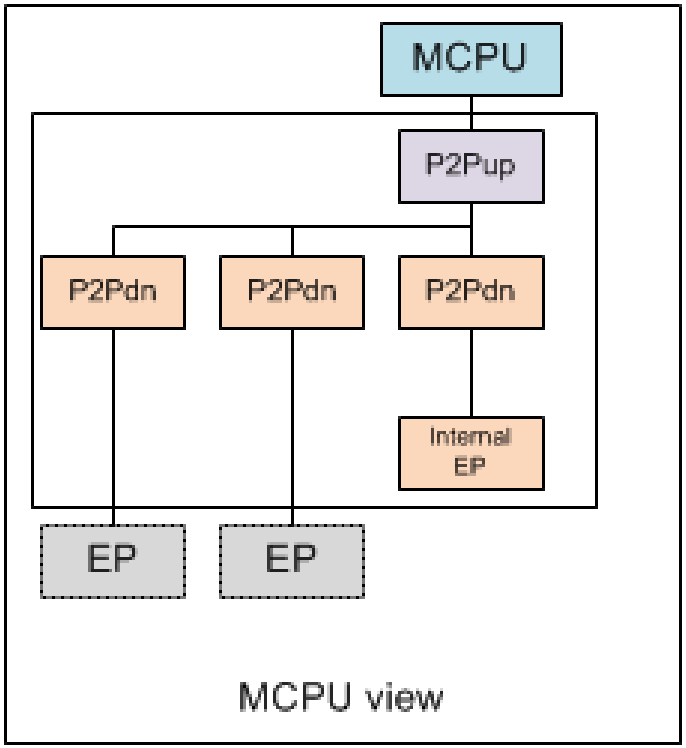
Simple Shared IO System

- Need Management CPU
  - Has a policy to assign functions
  - Endpoints may not even be attached
- Switch has port attributes assigned in internal EP
  - Host, management, downstream



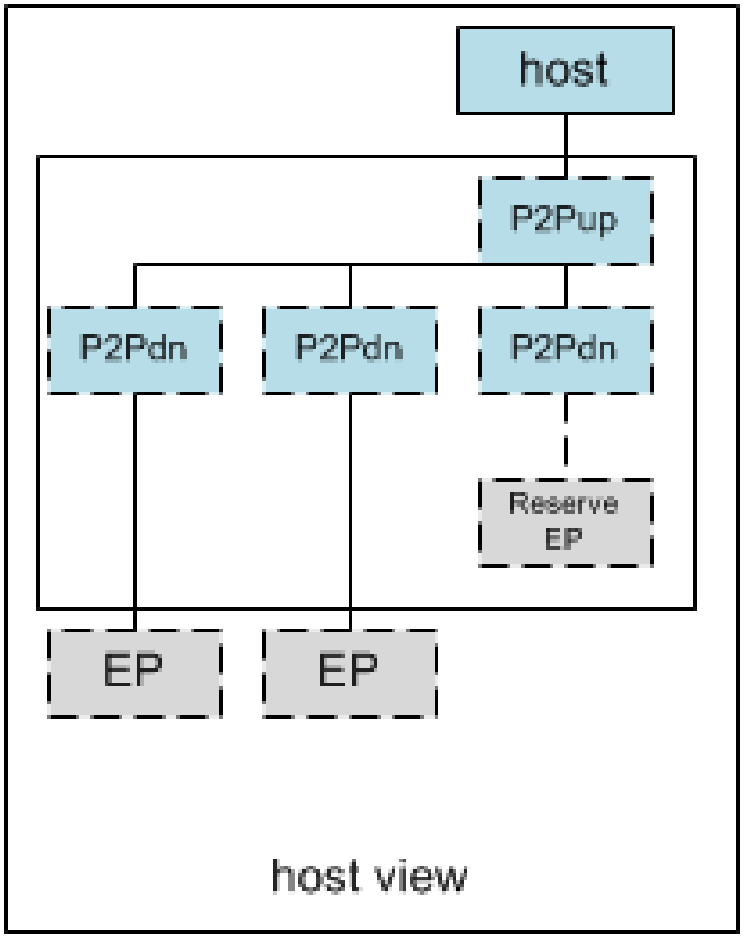
Management CPU View

- MCPU sees a standard PCIe switch with endpoints
- MCPU does not 'see' host port in its PCIe hierarchy
- Internal Endpoint tells about host ports



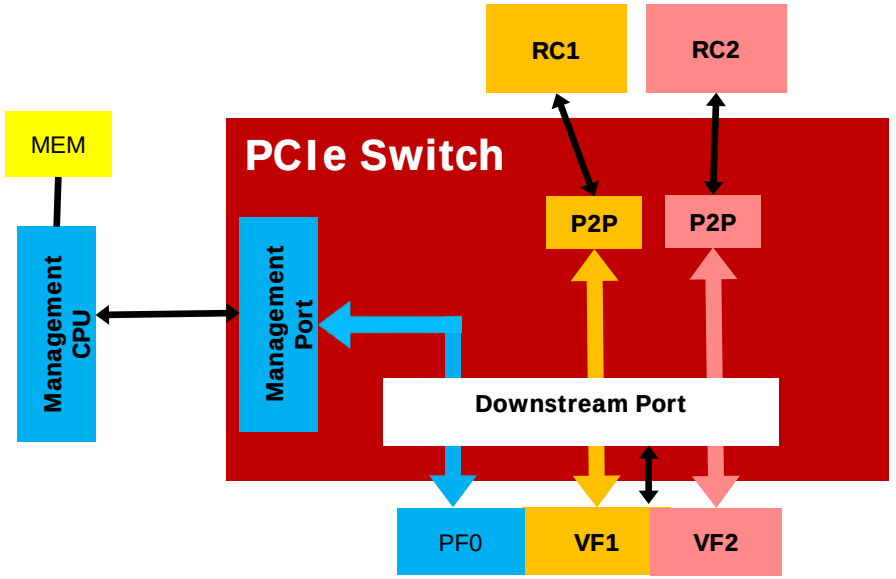
Host View

- Host port finds standard PCIe switch hierarchy
- Dashed lines show software synthesized devices
- Reserve EP(s) used to reserve resources for later hot add
- Host transparently finds endpoints
- Host does not know MCPU synthesized this hierarchy!



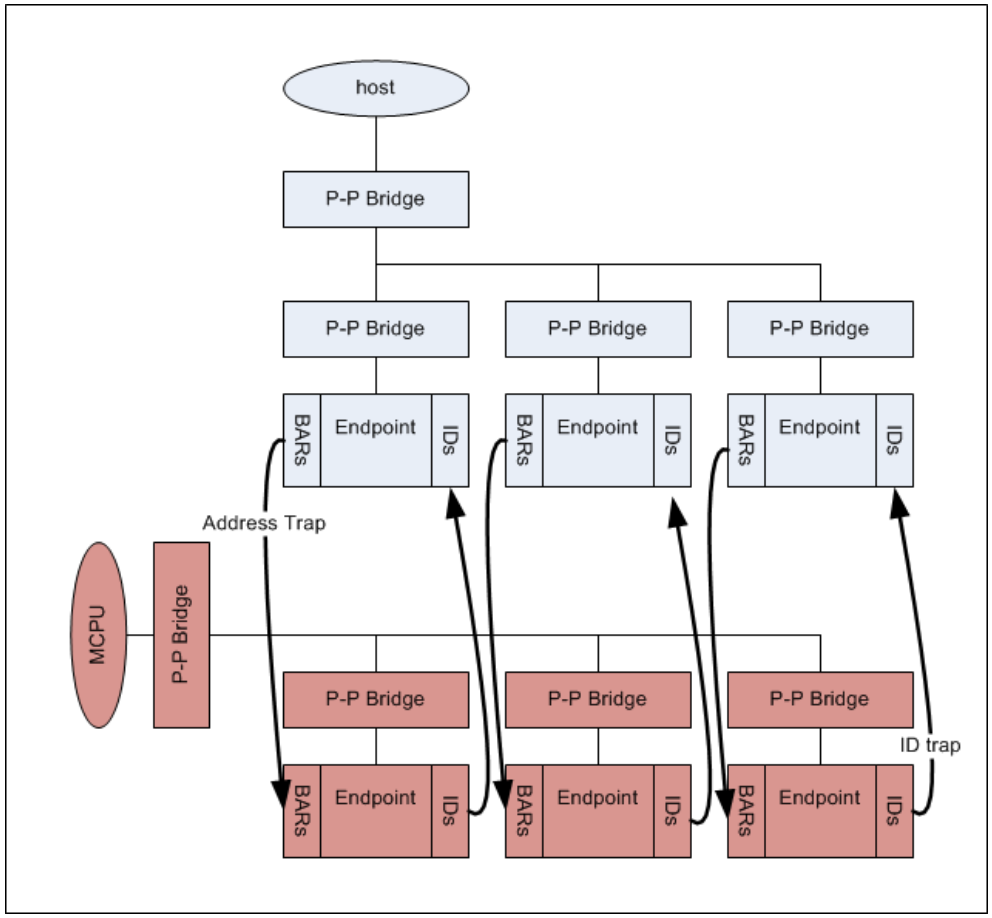
Operational Data Flow

- Data flow direct between the VFs and its host
- Memory Requests routed downstream via translated addresses configured by MCPU
- Upstream selected by RID:
  - Memory requests
    - DMA read and write
    - MSI/MSI-X Interrupts
  - Completions
  - PCIe messages
    - INTx Interrupts
    - Error messages



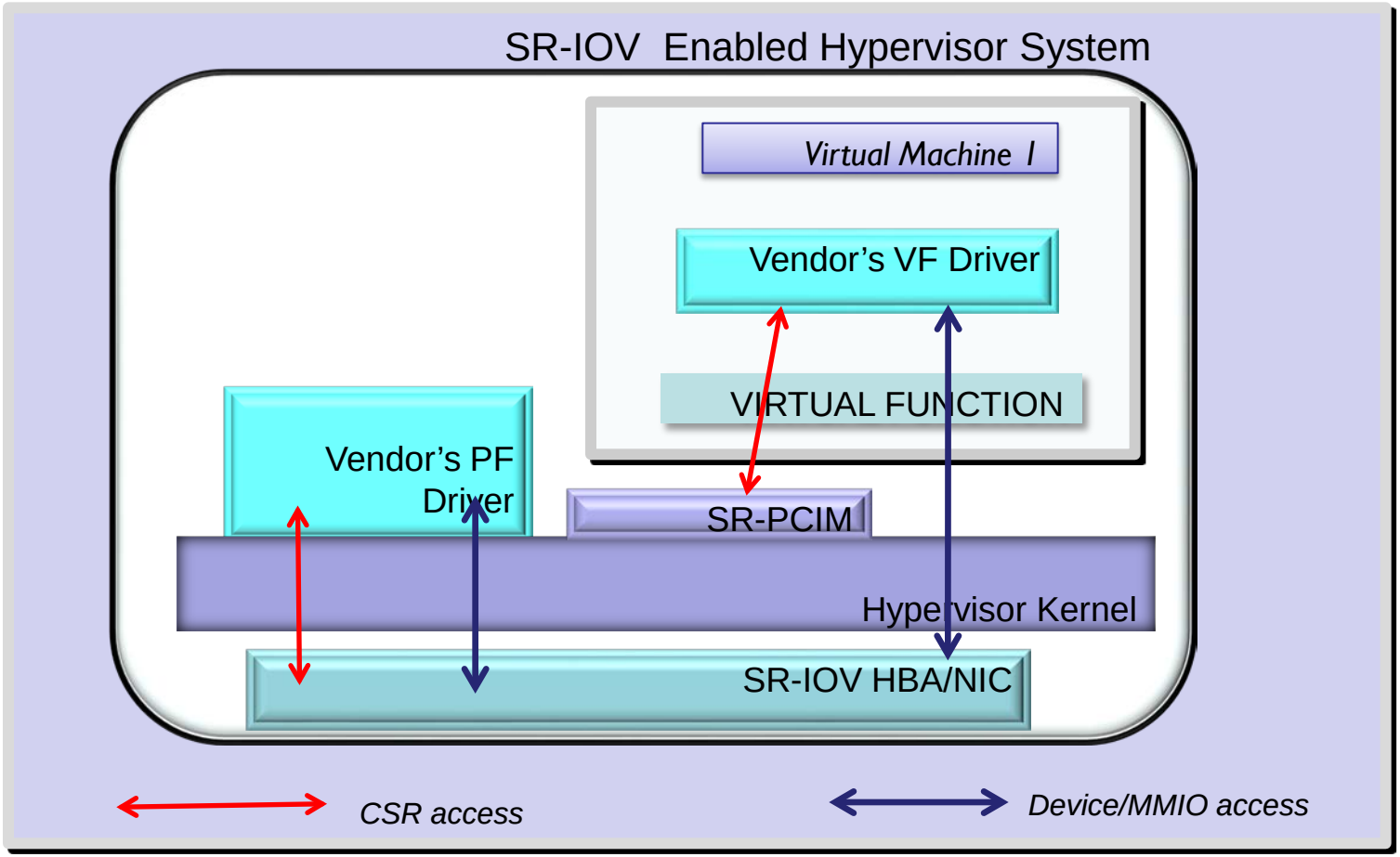
Virtual vs Physical Hierarchy

- A host port connects different address and bus number domains
- Requests going downstream are Address Trapped
- Requests going upstream are ID Trapped
- Completions have translation at host port for both directions

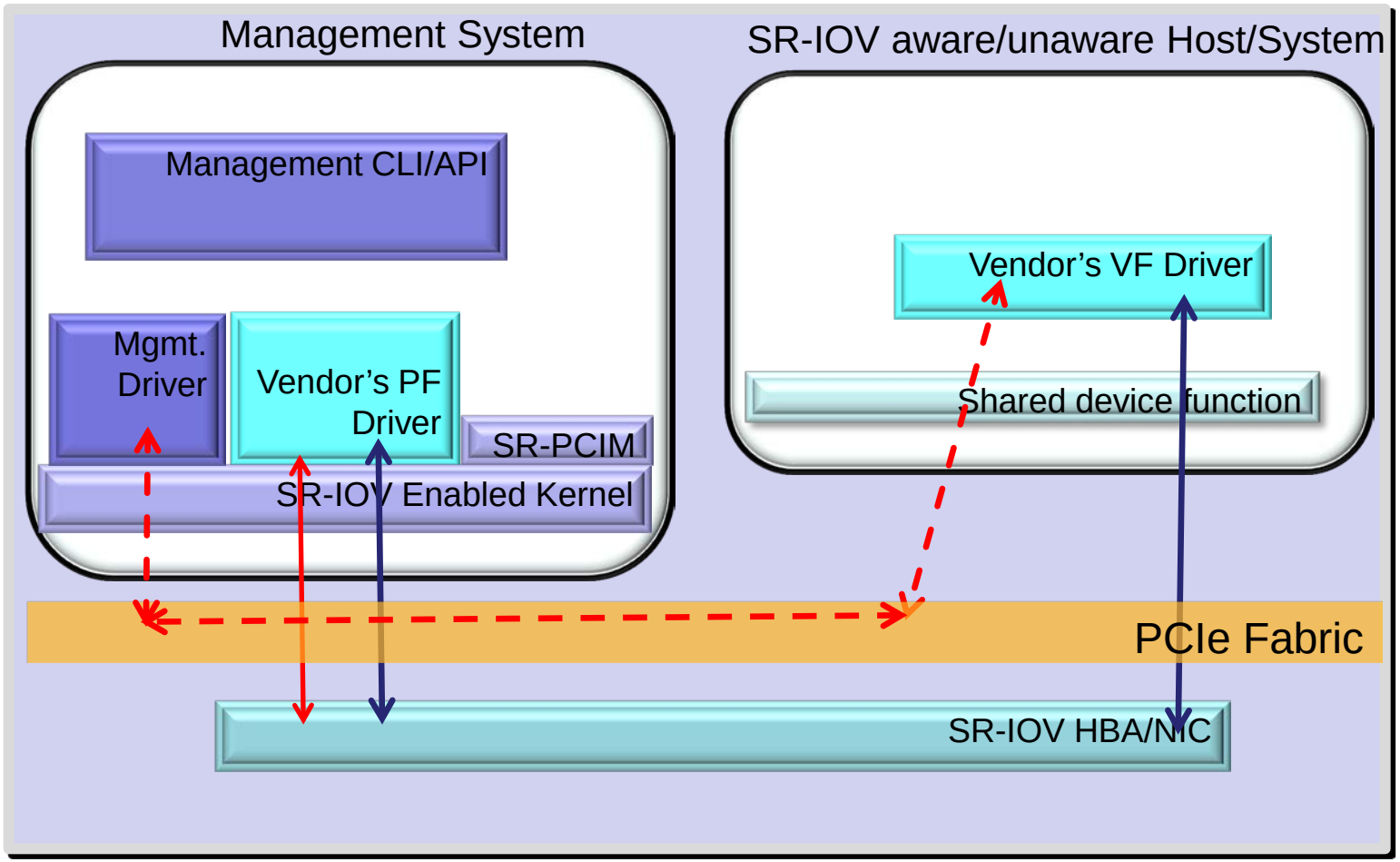


Physical Hierarchy is pink
Virtual Hierarchy seen by host is blue

SR-IOV VF – Current System Model



Sharing SR-IOV VF with Management Software



Management software Data structures

- Switch database
  - All host ports, fabric ports and devices organized under a switch data structure
- Device database
  - Lists all available devices and functions for shared IO allocation
  - Functions can be black listed (reserved)
- Configuration space database (for real and virtual devices)
- Port database
  - Port configurations for each host port
    - Shared IO allocations/virtual slot
    - Host port features
    - Options – hot plug enable/reserve, remote boot

Device inventory

- All devices under a switch added to database
  - No drivers loaded in MCPU for non-SR-IOV Single/Multi function devices
    - Built-in drivers can also be ‘black-listed’ in the OS so they won’t load
  - PF drivers for SR-IOV devices loaded – to enable VF
    - No change in vendor supplied PF drivers
  - VF devices black listed in the MCPU OS
    - If VF driver loaded, VF won’t be available for allocation!

Handling Configuration access

- MCPU initializes a ring buffer for configuration requests
  - Per (host-)edge switch in the fabric
  - Any host BIOS enumerations is captured, encapsulated and spooled to MCPU
  - This spooled data includes a pointer to get back to requesting host
- MCPU looks up Port DB and Device DB for each redirected configuration request
- MCPU typically uses the local OS API/system calls and returns the results for configuration reads/writes (also takes care of SR-PCIM)

Setting up Shared IO allocations

- For Shared IO to work, need:
  - Local/host RID and the global/MCPU RID for the device EP
  - BAR address ranges of MCPU view of device EP
  - Ordered route set in a fabric between host and device
- First access from host (e.g. BIOS) gives us the host RID
  - MCPU sets up RID mapping table on this access
  - Host's PCIe reset makes MCPU 'forget' this setup
- Host Assigning BARS triggers MCPU to set up address traps – one trap per BAR
  - In case of SR-IOV device allocations:
    - BAR addresses are contiguous in the global address space
    - 1 address trap per contiguous BAR range!

Surprise Events

- Hot Add
  - Reserve EP creates bus and memory reservations in host
  - Hot remove reserve EP
  - Hot add new EP into reserved bus and memory space
    - No re-enumeration needed
- Surprise Remove
  - Issue outstanding non-posted requests: CTO = blue screen
  - eDPC for new RC (coming)
  - Switch with ‘read tracking’ to remember outstanding non-posted
- Either host or downstream port can have event

Shared IO removal

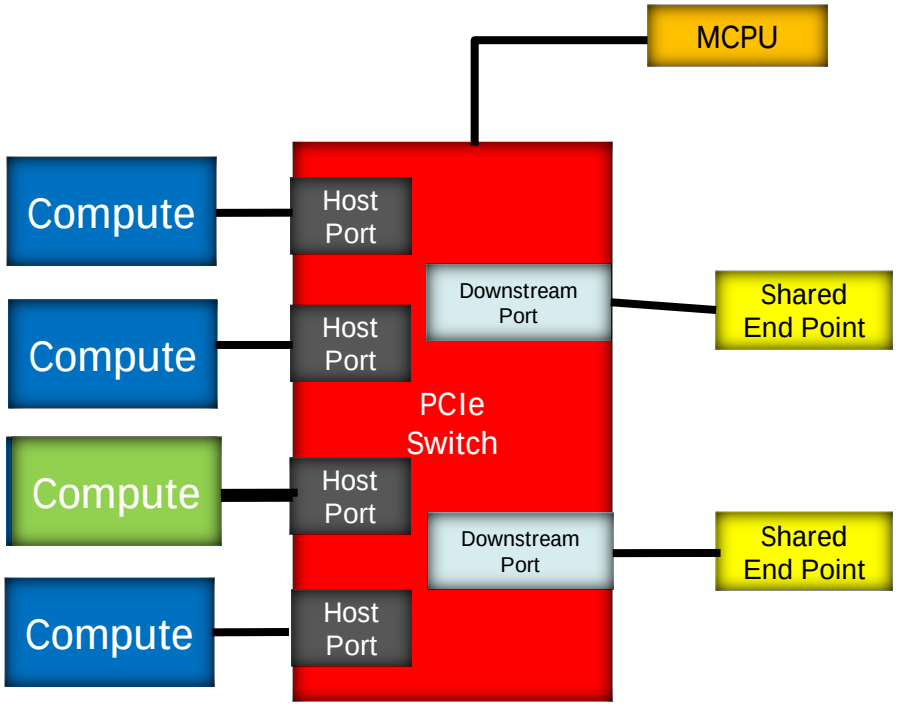
- Shared IO removal happens when:
  - Endpoint is removed
    - Triggers DPC event reported to MCPU
  - Host system reboots/shuts down
    - Temporary; host system booting will trigger allocation again
  - Admin removes an allocation (with hot-unplug enabled)
    - Permanent; configuration DB updated
- For a host reboot/shutdown, MCPU
  - Removes the address traps and RID tables for the port
  - Does a FLR to the affected physical end point (where supported)

Hot Plug Cases

- Hot plug/unplug
  - Of allocated/shared virtual function
    - Admin tools supported
  - Of an entire device
    - MCPU gets notification, tells each host to take it out
  - Of a host port
    - MCPU gets notification, FLR to VFs owned by that host
- PCIe hot-plug/link state change/surprise remove messages generated by MCPU as needed
- Optional capability:
  - A host hot-plug driver to facilitate dynamic hot-plug capability (on request by MCPU)
  - Resource/Bus reservation on host booting to support dynamic hot-plug capability

System Use Case

- Compute disaggregated from IO
- Removal of host doesn't affect other hosts or IO
- Replace with new host
- Add IO at any time
- Remove IO at any time

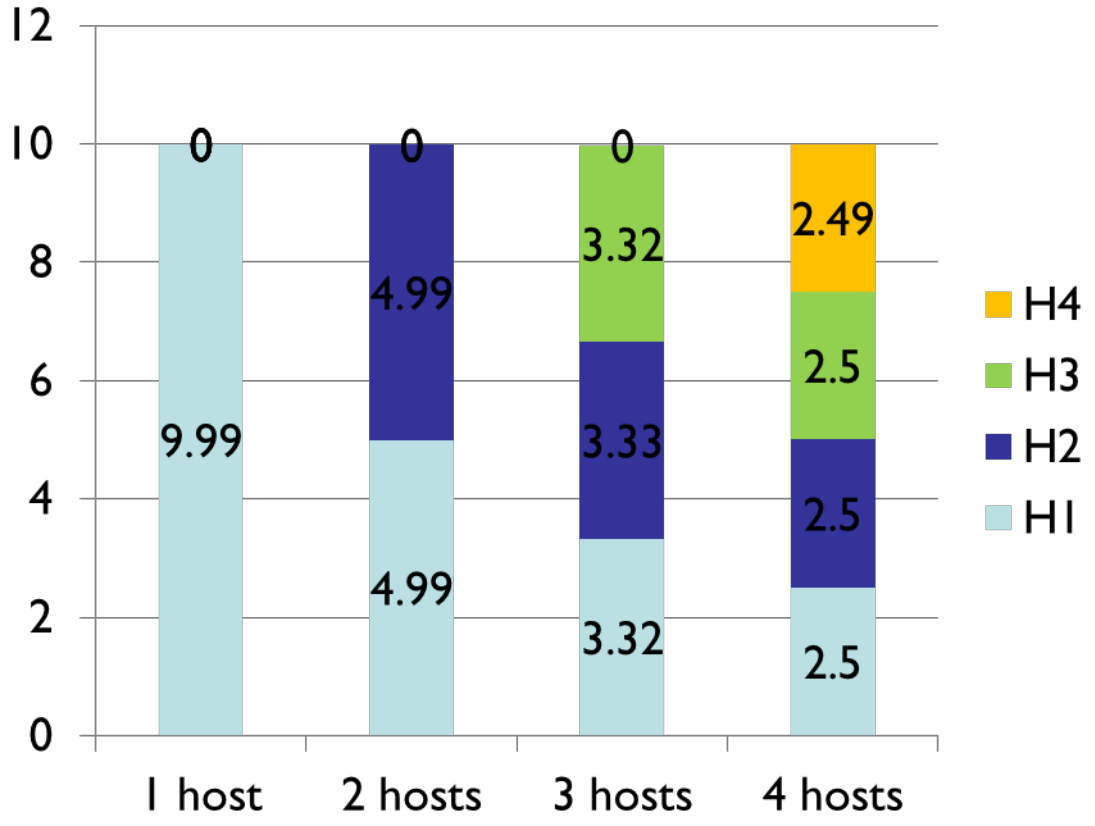


Sharing Network Endpoints

- Network endpoints do not save state
  - N functions operate independently
  - Easy to assign any function (or group) to a host
  - Performance scales based on hardware rate policies
- Virtual Ethernet Bridging (VEB)
  - Allows hosts to talk to each other without leaving PCIe
  - Have seen higher than 10 GE performance on 10 GE NIC

Shared NIC Performance

- SR-IOV NIC with 64 VFs
- 4 Hosts, each get 1 VF
- Iperf test for bandwidth
- Very fair breakdown of bandwidth

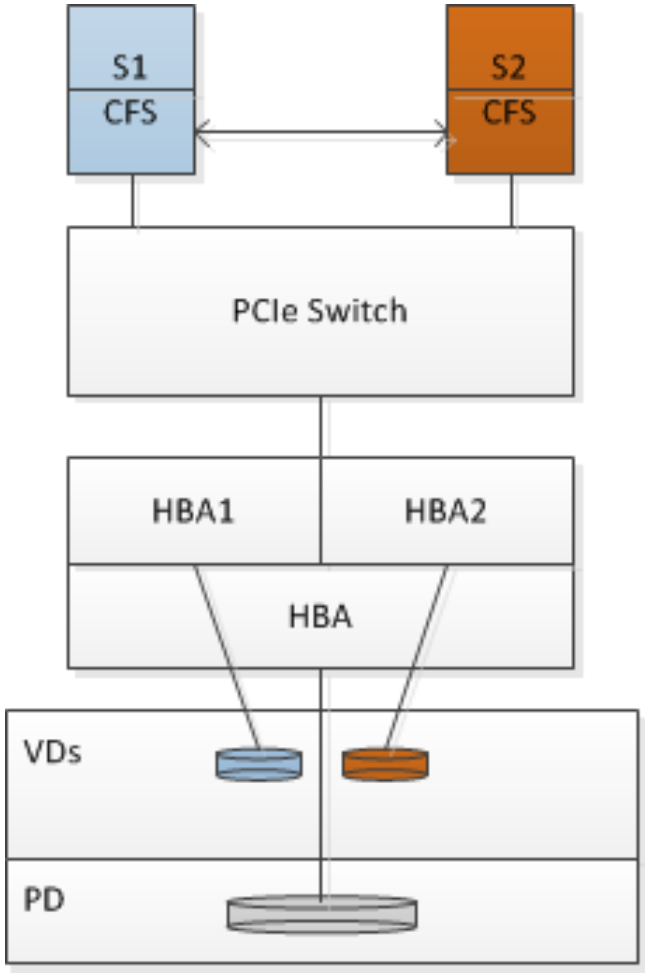


Sharing Storage Endpoints

- Complications to sharing storage endpoints
  - Multi-function controllers exist but
  - Require Clustered File System to coordinate access to LUNs
  - Or require processor in controller to track LUN access

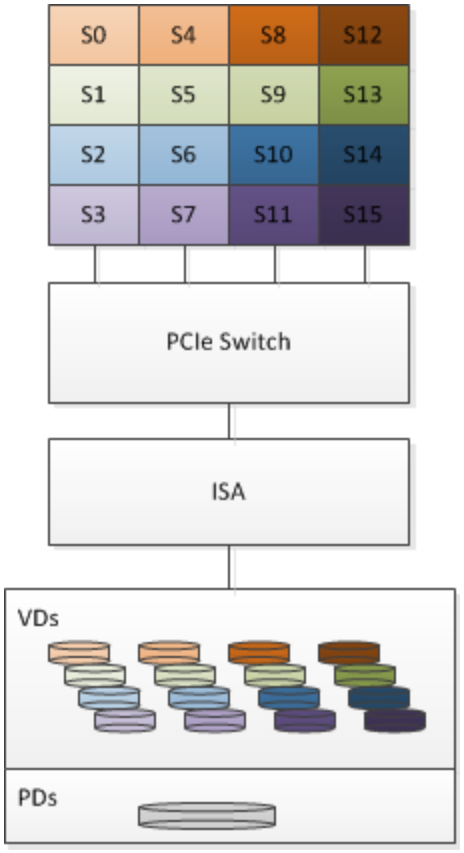
SR-IOV HBA and Clustered File System

- Clustered File System running on each server
- CFS presents one PD as n VDs
- Managed switch connects multiple hosts to SR-IOV HBA
- Each server finds one HBA
  - HBA is virtualized
  - One Physical Disk is used as multiple Virtual Disks
- Boot remains a challenge



Shared Storage with ISA

- Multiple Servers
  - No knowledge of other servers
  - Ideally cannot tell ISA apart from HBA
- Managed PCIe switch supports multiple hosts and shared IO
- IO is intelligent storage adapter (ISA) to handle storage requests
  - ISA creates VDs out of single PD
  - Flexibly assign storage
  - Shared boot VD
  - Shared PD via reservation



Sharing NVMe Endpoints

- NVMe endpoints typically single function today
  - Expensive: want to mitigate costs and get high attach rate
  - How to flexibly use?
  - How to get dual ported endpoint for reliability?
  - → Managed PCIe switch provides some options
- Simplest would be multi-function NVMe device
- Share single function via MCPU command relay
  - All admin, submission queue, completion queue via MCPU
  - Adds latency; preserves NVMe driver
- Share single function via NVMe driver change
  - Prefix host access DMA addresses to steer to correct host
  - Record non-0 start offset for submission/completion queues

Review and Next steps

- Review
  - Covered PCIe enumeration for single host system
  - Expanded to synthetic hierarchy for multiple hosts
  - Showed robustness to hot add, remove of PCIe hosts and endpoints
  - Use cases for networking and storage
  - More possibilities for NVMe and managed switch
- Next steps
  - Integrate MCPU with switch
  - QoS performance planning: TC for read and write

Attribution & Feedback

The SNIA Education Committee thanks the following Individuals for their contributions to this Tutorial.

Authorship History

Jeff Dodson / March 4, 2015

Updates:

Jeff Dodson / April 1, 2015

Additional Contributors

Please send any questions or comments regarding this SNIA Tutorial to ***tracktutorials@snia.org***

Appendix

High Availability System