Multi-Host Sharing of NVMe Drives and GPUs Using PCIe Fabrics

Vincent Haché

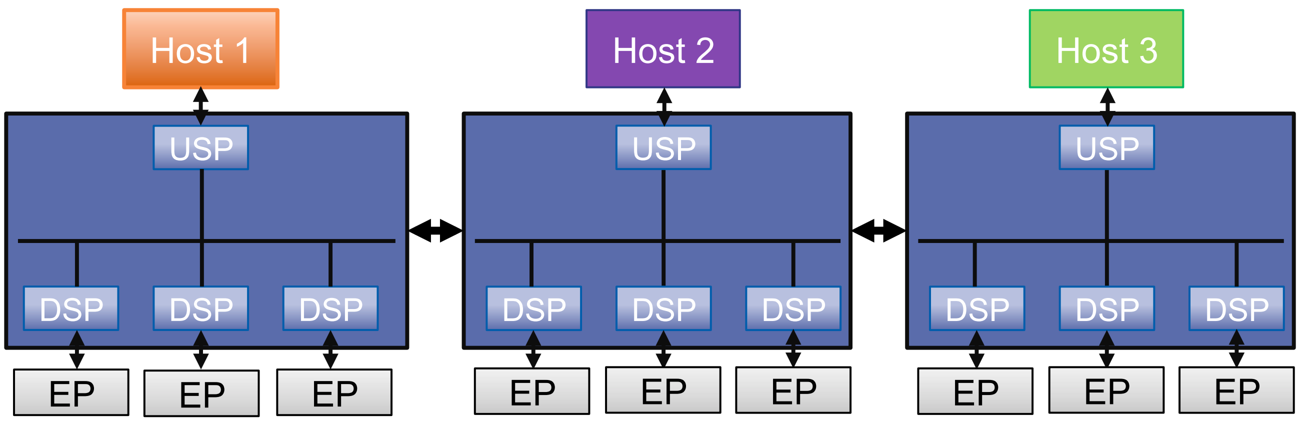
Principal Applications Engineer,
Microsemi Corporation

Introduction

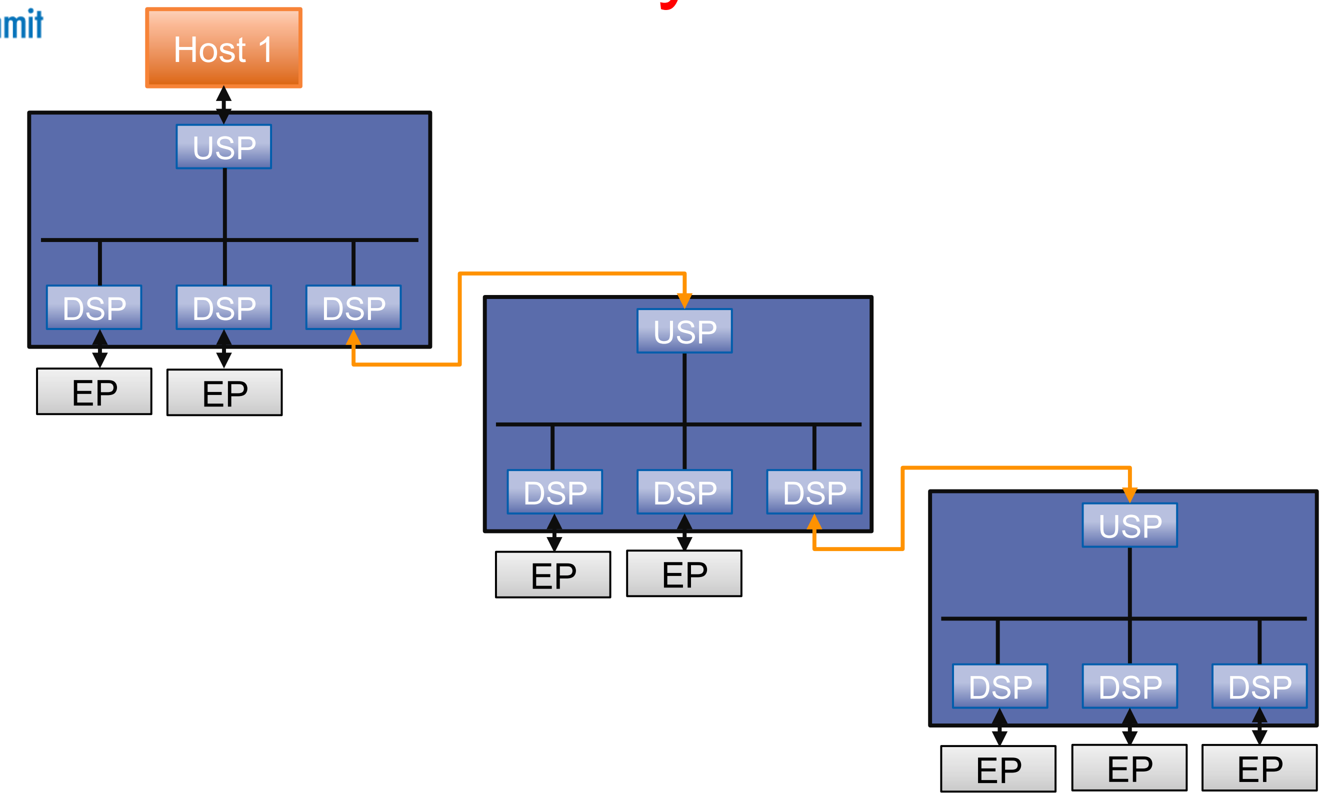
- Increase in use of GPUs and NVM in DC
- System designers need:
  - Efficient resource deployment
  - High-BW, low-latency interconnect
  - Flexible, composable architectures
- There are restrictions in standard PCIe that present challenges for system design

PCIe Hierarchy Restriction

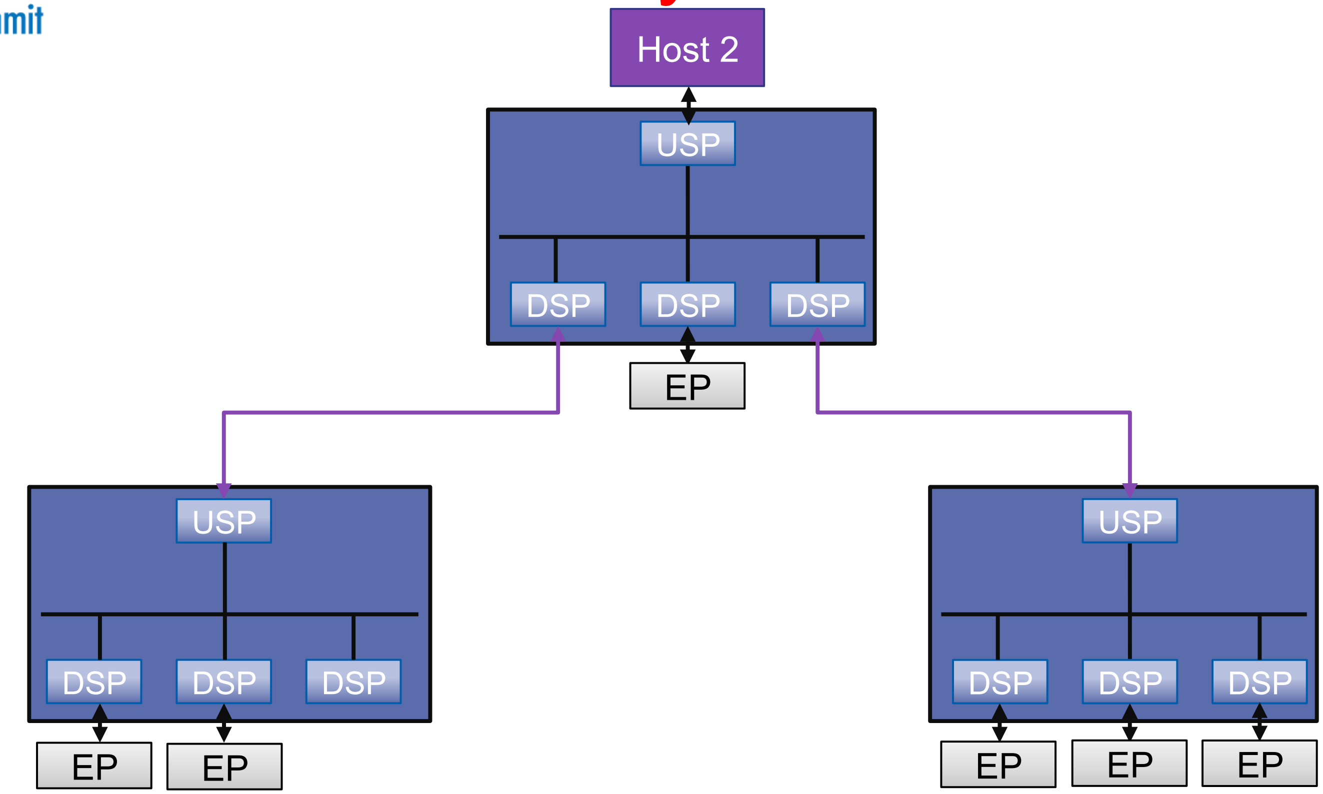
- PCIe hierarchy is restrictive, making scale out challenging



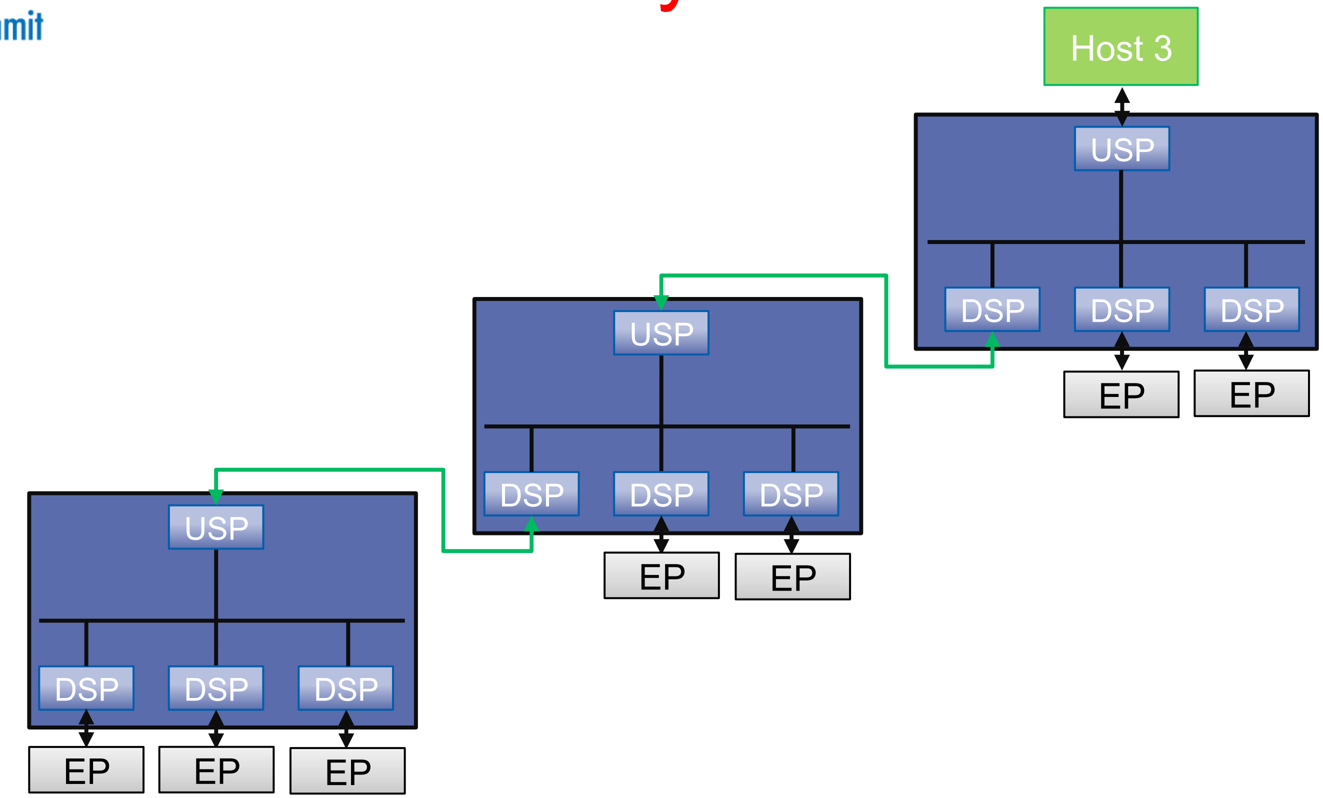
PCIe Hierarchy Restriction: Host 1



PCIe Hierarchy Restriction: Host 2

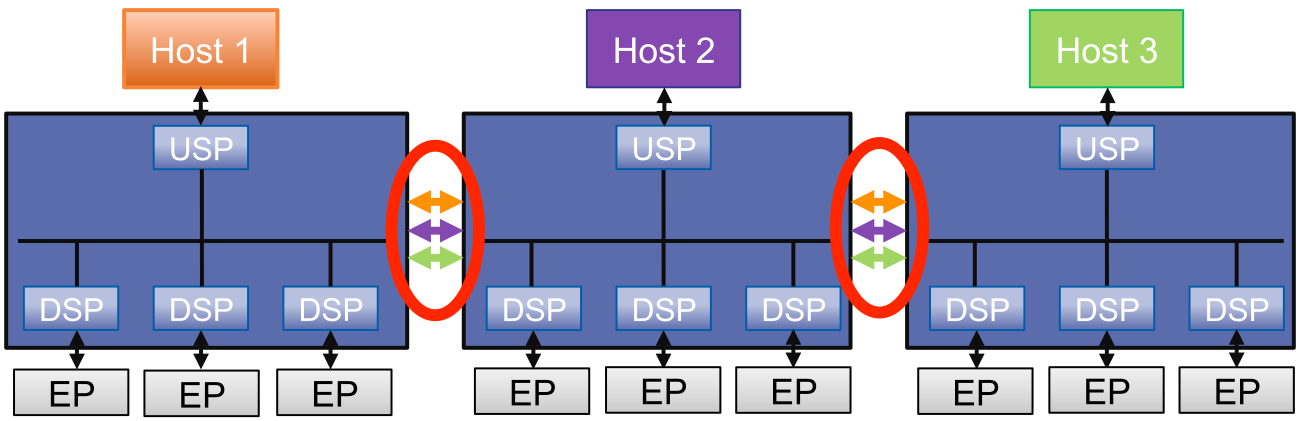


PCIe Hierarchy Restriction: Host 3



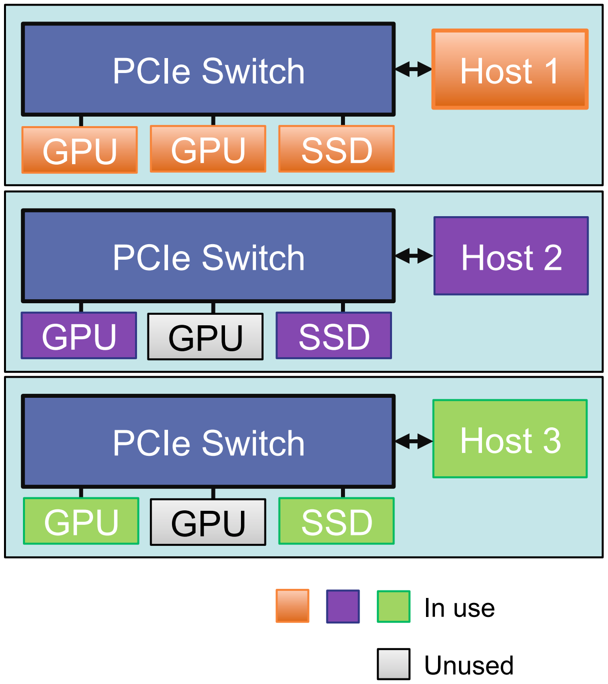
PCIe Hierarchy Restriction (continued)

- The multiple links required for transparent scale out complicates design and decreases efficiency



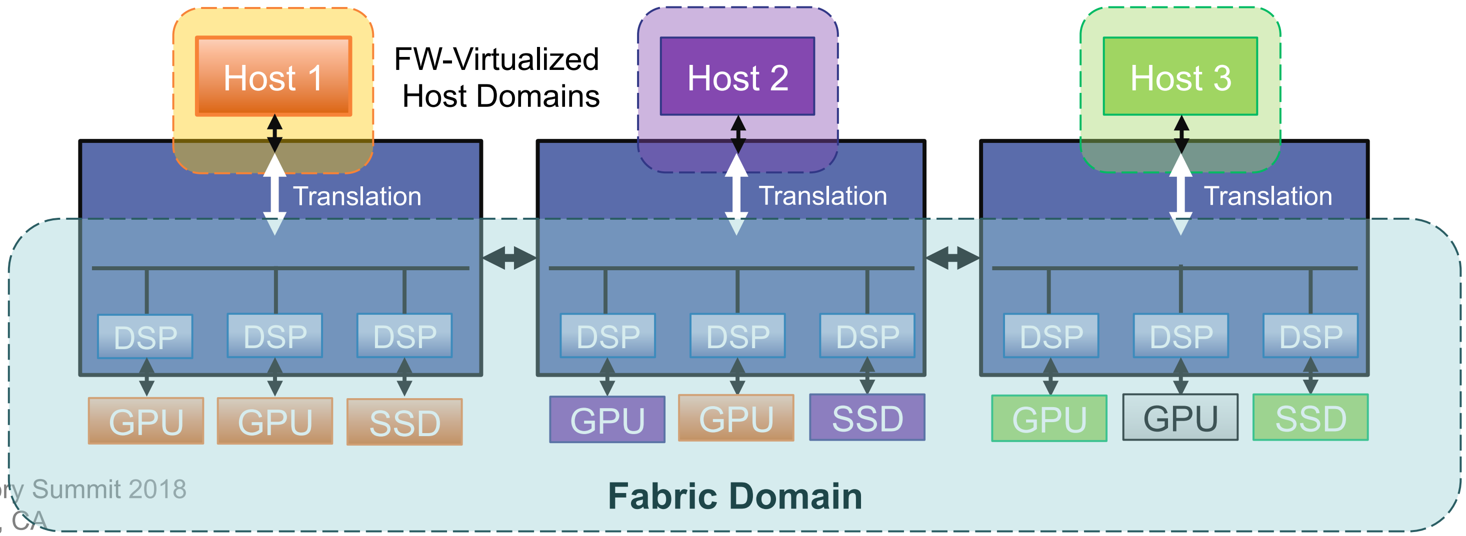
PCIe Single Domain Restriction

- PCIe is single domain
  - Unused EPs are stranded
  - Complicated, non-standard NT drivers required for sharing



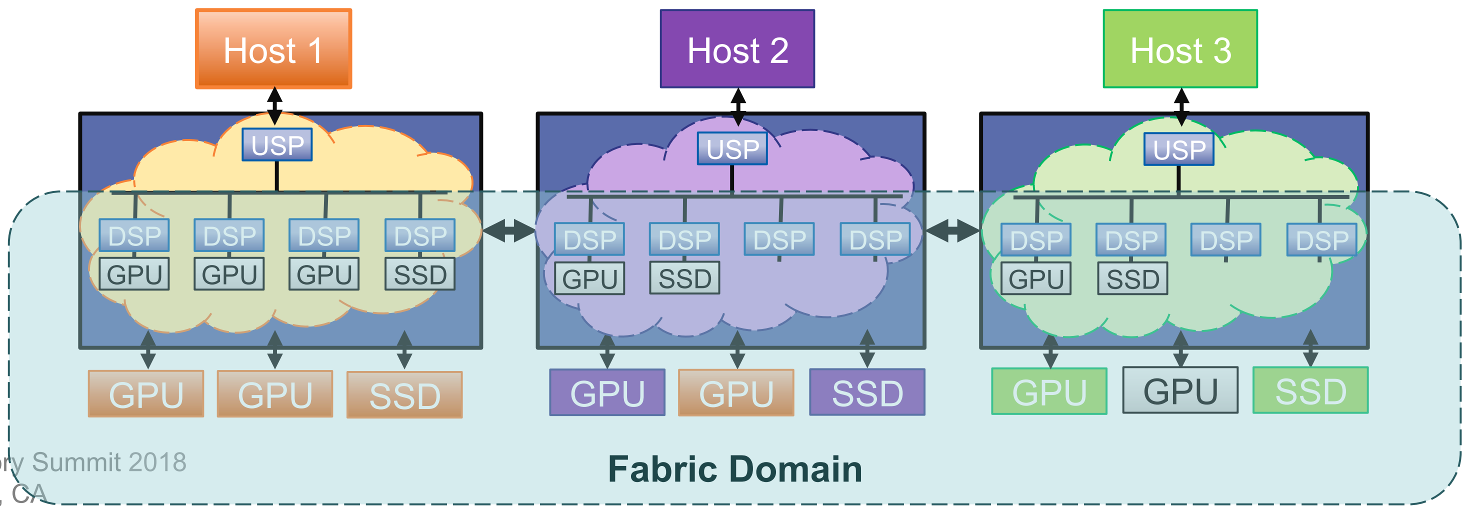
PCIe Fabrics for Scaling

- Fabric routing is proprietary, non-hierarchical
- Fabric links are shared among hosts



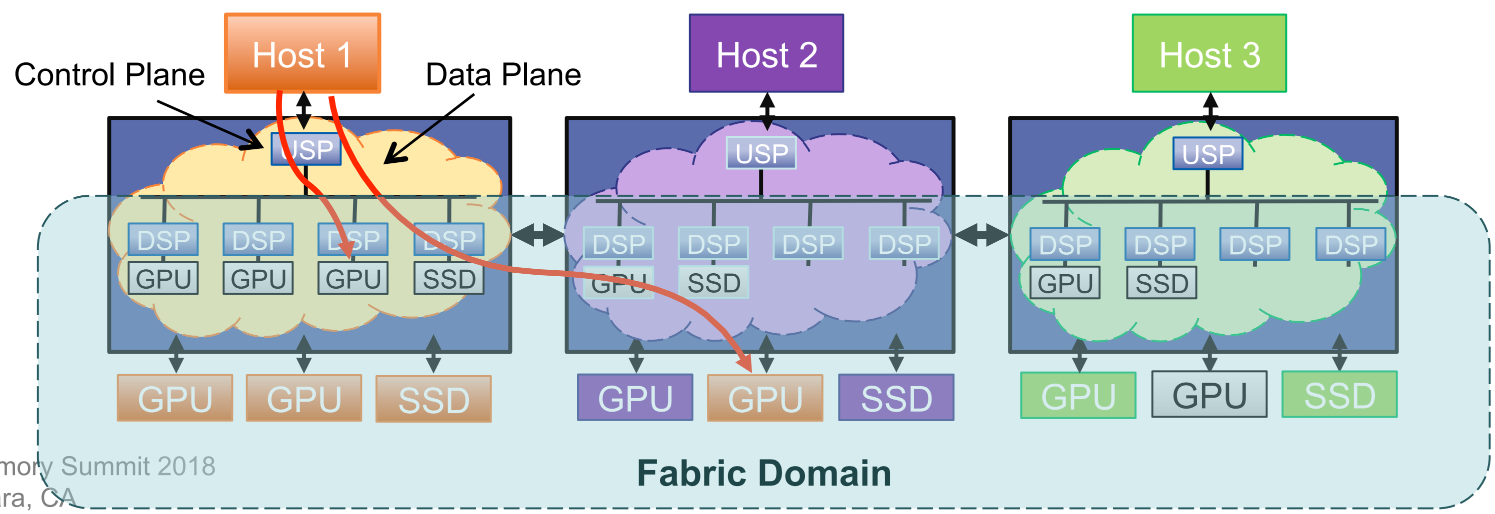
PCIe Fabrics for Scaling (continued)

- FW running on embedded CPU virtualizes a simple switch compliant with the PCIe spec



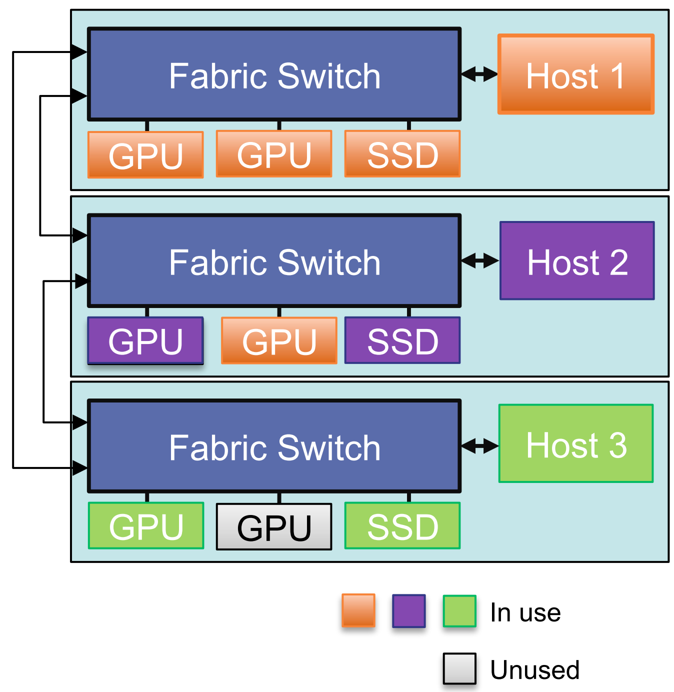
PCIe Fabrics for Scaling (continued)

- Embedded CPU handles the control plane, but data is routed directly by switch HW



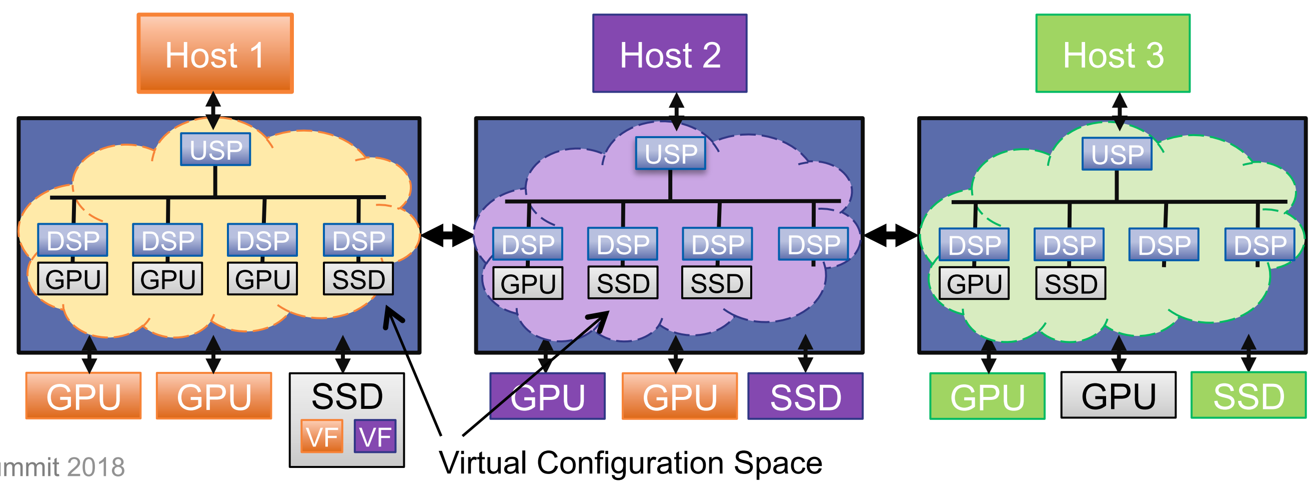
Device Sharing on a PCIe Fabric

- Unused devices can be dynamically assigned (no longer stranded)
- Low-latency, high-BW P2P within the rack
- Standard drivers to simplify system development



Multi-host Sharing of SSDs

- Fabric resources assigned by function
- SR-IOV: EP appears as multiple functions

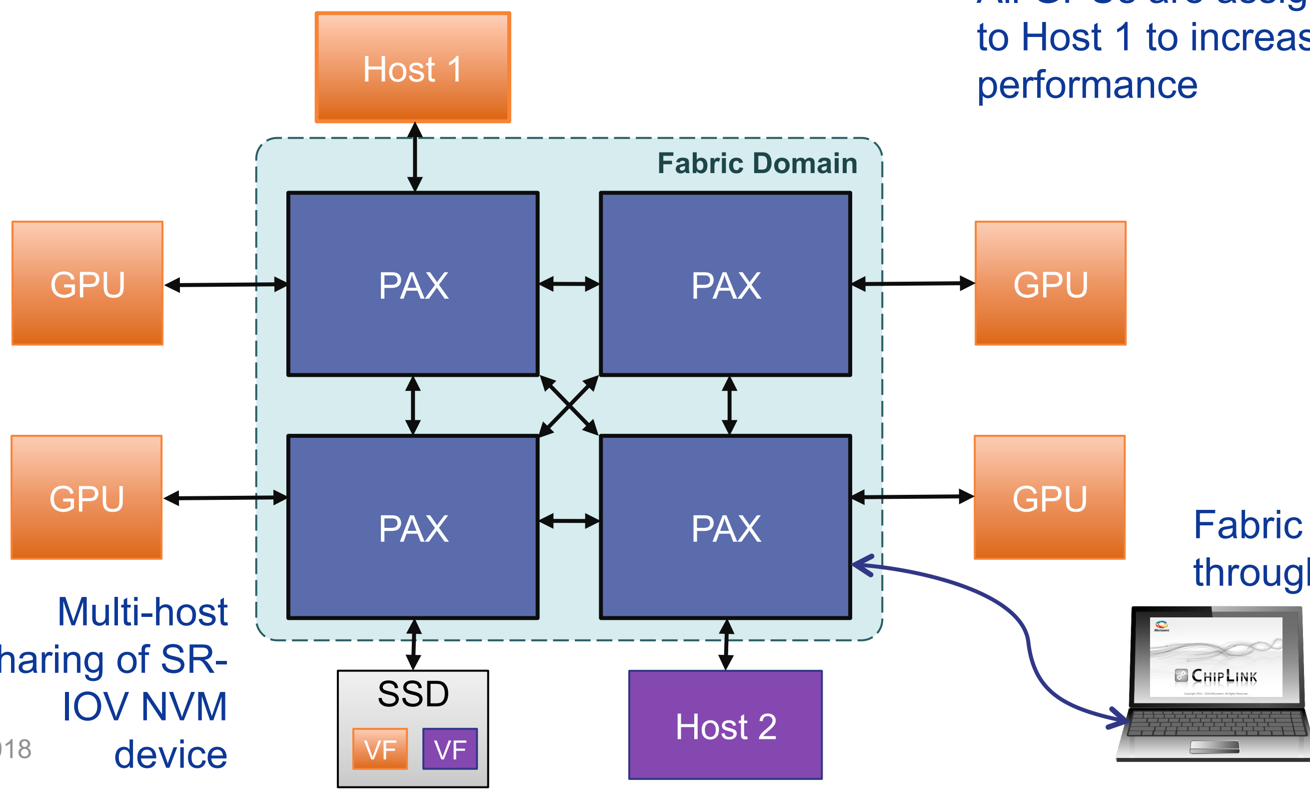


Demo: Multi-host Sharing of NVMe and GPUs

- Dynamic partitioning of GPUs and multi-host sharing of SR-IOV SSDs in real time
  - Standard host drivers in Windows Server 2016 and Ubuntu Server 16.04 LTS
- GPU P2P transfers across the fabric
  - CUDA P2P BW test and TensorFlow cifar10 image classification multi-GPU training algorithm

Demo: Multi-host Sharing of NVMe and GPUs (continued)

All GPUs are assigned to Host 1 to increase performance

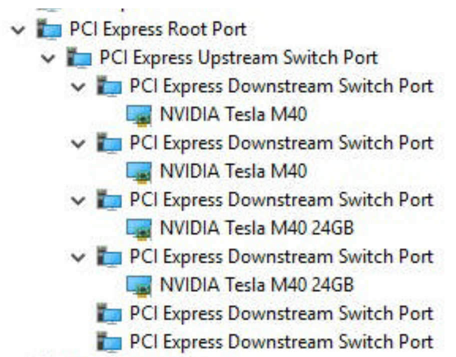


Fabric managed through UART

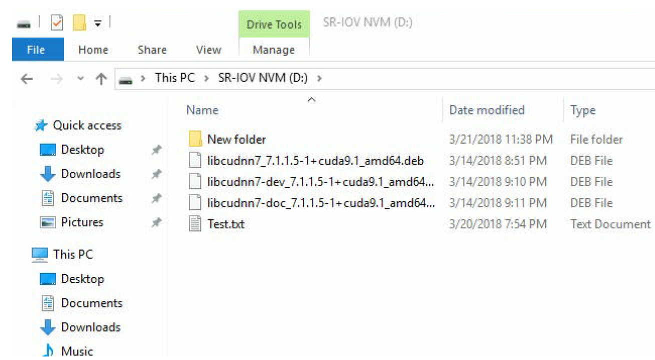
Multi-host sharing of SR-IOV NVM device

Demo: Multi-host Sharing of NVMe and GPUs (continued)

Domain Virtualized as a Spec-compliant PCIe Switch



NVM VF Appears as a Standard NVM Device



Demo: Multi-host Sharing of NVMe and GPUs (continued)

CUDA P2P Bandwidth

```
P2P Connectivity Matrix
     D\D     0     1     2     3
     0       1     1     1     1
     1       1     1     1     1
     2       1     1     1     1
     3       1     1     1     1
```

```
Unidirectional P2P=Enabled Bandwidth Matrix (GB/s)
   D\D      0      1      2      3
     0 210.61  12.96  12.54  12.53
     1  12.52 211.35  13.08  13.06
     2  12.52  12.52 212.61  13.05
     3  13.06  13.06  12.54 211.36
```

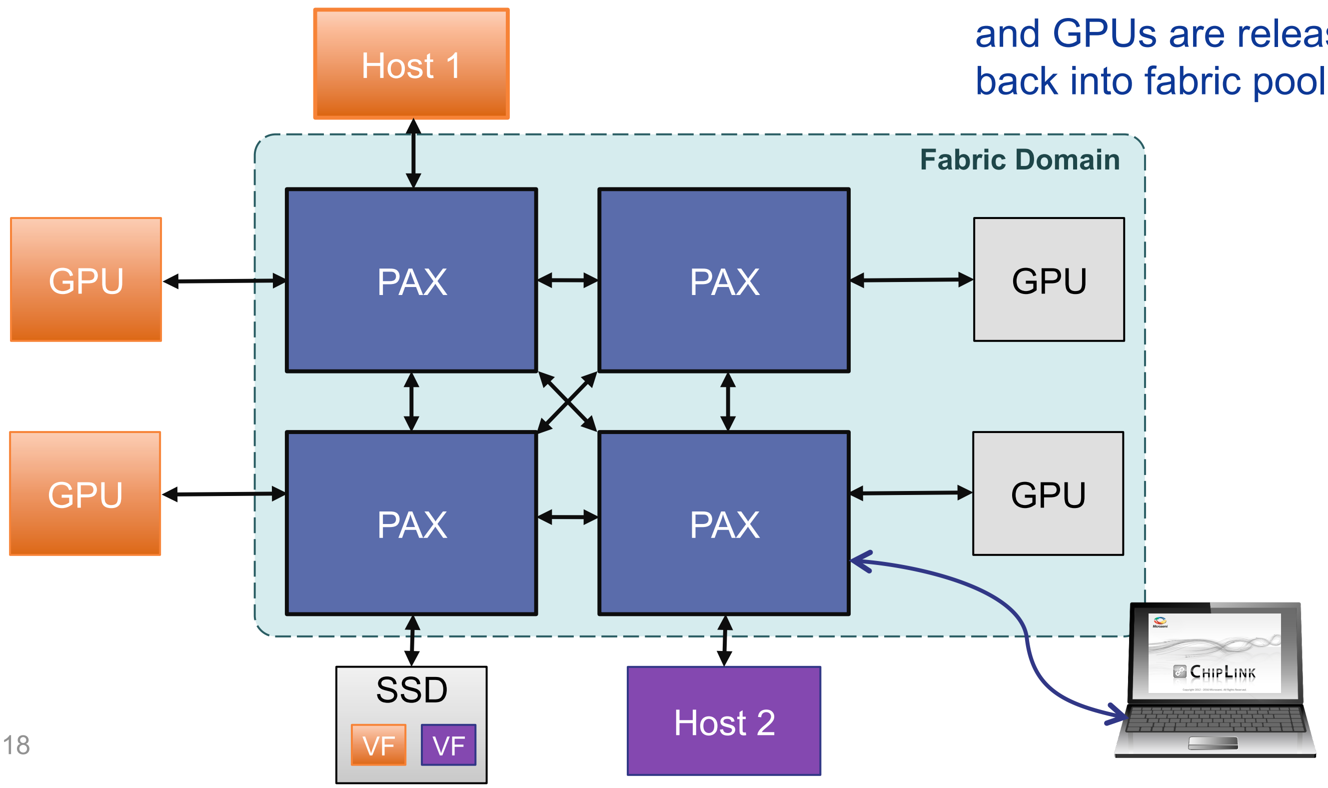
```
Bidirectional P2P=Enabled Bandwidth Matrix (GB/s)
   D\D      0      1      2      3
     0 213.51  24.81  24.77  24.72
     1  24.73 213.55  24.73  25.74
     2  24.53  24.57 214.73  24.86
     3  24.83  25.72  24.73 214.58
```

Running Tensorflow Model

```
Administrator: Command Prompt - python  cifar10_multi_gpu_train.py
2018-04-06 18:37:52.698874: I C:\tf_jenkins\home\workspace\rel-win\M\windows-gpu\PY\36\tensorflow\core\common_runtime\gp
u\gpu_device.cc:1030] Found device 3 with properties:
name: Tesla M40 24GB major: 5 minor: 2 memoryClockRate(GHz): 1.112
pciBusID: 0000:07:00.0
totalMemory: 22.43GiB freeMemory: 22.18GiB
2018-04-06 18:37:52.701446: I C:\tf_jenkins\home\workspace\rel-win\M\windows-gpu\PY\36\tensorflow\core\common_runtime\gp
u\gpu_device.cc:1045] Device peer to peer matrix
2018-04-06 18:37:52.702693: I C:\tf_jenkins\home\workspace\rel-win\M\windows-gpu\PY\36\tensorflow\core\common_runtime\gp
u\gpu_device.cc:1051] DMA: 0 1 2 3
2018-04-06 18:37:52.703145: I C:\tf_jenkins\home\workspace\rel-win\M\windows-gpu\PY\36\tensorflow\core\common_runtime\gp
u\gpu_device.cc:1061] 0:   Y Y Y Y
2018-04-06 18:37:52.704059: I C:\tf_jenkins\home\workspace\rel-win\M\windows-gpu\PY\36\tensorflow\core\common_runtime\gp
u\gpu_device.cc:1061] 1:   Y Y Y Y
2018-04-06 18:37:52.704960: I C:\tf_jenkins\home\workspace\rel-win\M\windows-gpu\PY\36\tensorflow\core\common_runtime\gp
u\gpu_device.cc:1061] 2:   Y Y Y Y
2018-04-06 18:37:52.705856: I C:\tf_jenkins\home\workspace\rel-win\M\windows-gpu\PY\36\tensorflow\core\common_runtime\gp
u\gpu_device.cc:1061] 3:   Y Y Y Y
2018-04-06 18:37:52.706817: I C:\tf_jenkins\home\workspace\rel-win\M\windows-gpu\PY\36\tensorflow\core\common_runtime\gp
u\gpu_device.cc:1120] Creating TensorFlow device (/device:GPU:0) -> (device: 0, name: Tesla M40, pci bus id: 0000:04:00.
0, compute capability: 5.2)
2018-04-06 18:37:52.707744: I C:\tf_jenkins\home\workspace\rel-win\M\windows-gpu\PY\36\tensorflow\core\common_runtime\gp
u\gpu_device.cc:1120] Creating TensorFlow device (/device:GPU:1) -> (device: 1, name: Tesla M40, pci bus id: 0000:05:00.
0, compute capability: 5.2)
2018-04-06 18:37:52.708655: I C:\tf_jenkins\home\workspace\rel-win\M\windows-gpu\PY\36\tensorflow\core\common_runtime\gp
u\gpu_device.cc:1120] Creating TensorFlow device (/device:GPU:2) -> (device: 2, name: Tesla M40 24GB, pci bus id: 0000:0
6:00.0, compute capability: 5.2)
2018-04-06 18:37:52.709626: I C:\tf_jenkins\home\workspace\rel-win\M\windows-gpu\PY\36\tensorflow\core\common_runtime\gp
u\gpu_device.cc:1120] Creating TensorFlow device (/device:GPU:3) -> (device: 3, name: Tesla M40 24GB, pci bus id: 0000:0
7:00.0, compute capability: 5.2)
```

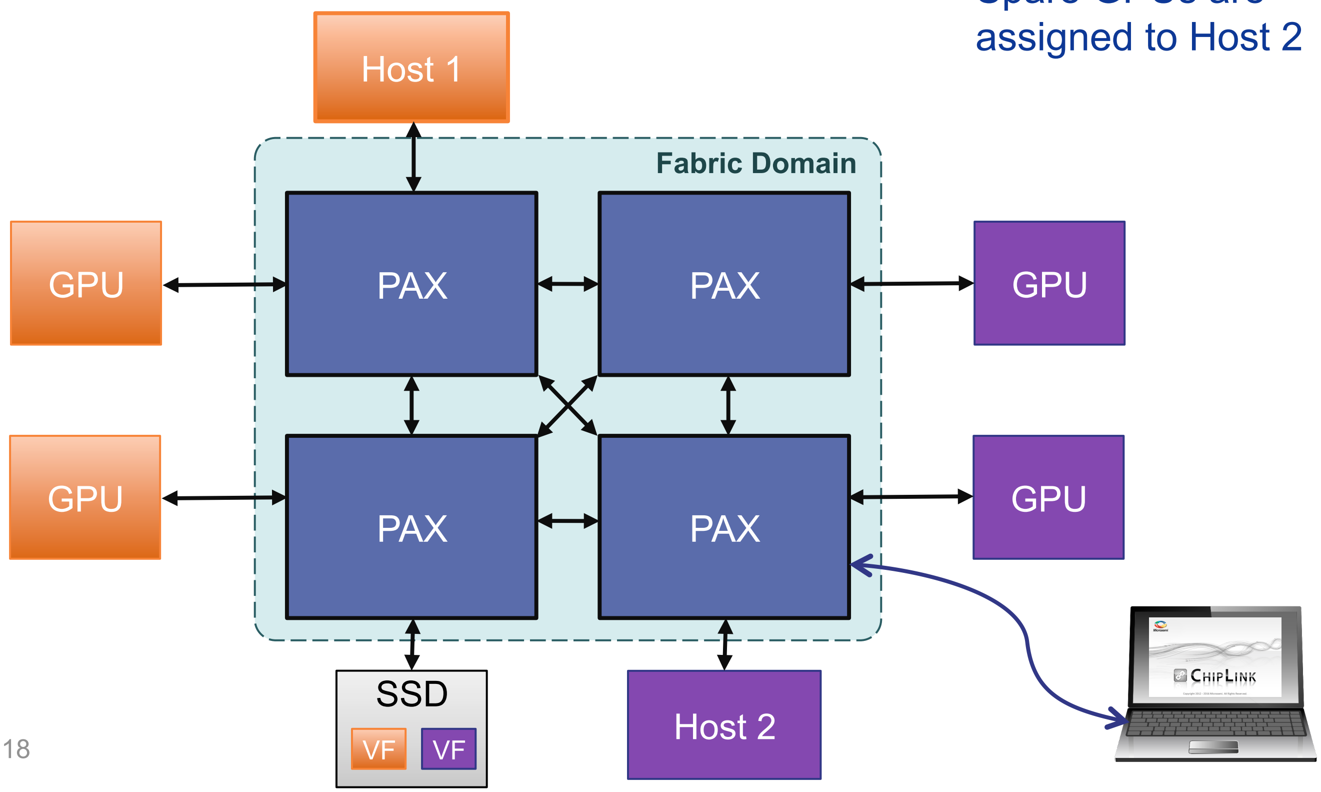
Demo: Multi-host Sharing of NVMe and GPUs (continued)

Host 1 workload completes and GPUs are released back into fabric pool



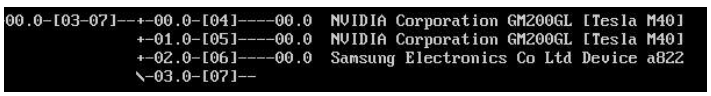
Demo: Multi-host Sharing of NVMe and GPUs (continued)

Spare GPUs are assigned to Host 2

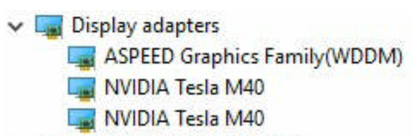


Demo: Multi-host Sharing of NVMe and GPUs (continued)

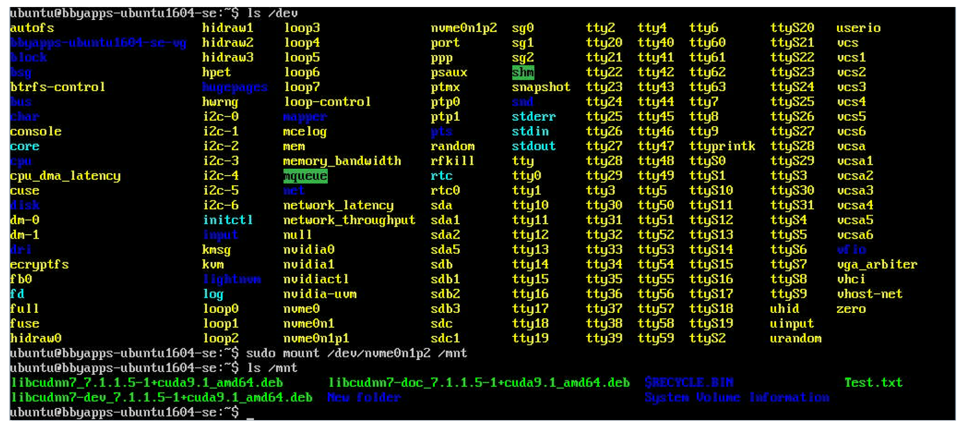
Domain Virtualized as a Spec-compliant PCIe Switch



Windows Host Still Running During Dynamic Reassignment



NVM VF Appears as a Standard NVM Device



Demo – Multi-host Sharing of NVMe and GPUs

CUDA P2P Bandwidth

```
P2P Connectivity Matrix
     D\D     0      1
     0       1      1
     1       1      1
```

```
Unidirectional P2P=Enabled Bandwidth Matrix (GB/s)
   D\D     0      1
     0 214.21  12.27
     1  13.06 212.99
```

```
Bidirectional P2P=Enabled Bandwidth Matrix (GB/s)
   D\D     0      1
     0 214.53  24.33
     1  24.83 215.55
```

- PCIe spec-compliant host domain
- Simple management
- Standard drivers
- Dynamic reassignment appears as spec-compliant surprise-plug

Summary

Microsemi's Switchtec PAX Switches enable new architectures for next-gen solutions

Benefits of PCIe fabrics with PAX:

- Scalable, low-latency, cost-effective
- Simple Management (PCIe, UART, TWI, Ethernet)
- Multi-host sharing of SR-IOV NVMe devices
- Standard host drivers

Live Demo at Booth #213