White Paper

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

Preliminary

October 2019

Contents

1 Revision History

The revision history describes the changes that were implemented in the document. The changes are listed by revision, starting with the most current publication.

1.1 Revision 2.0

Revision 2.0 was published in October 2019. It is the first publication of this document.

1.2 Revision 1.0

Revision 1.0 was an internal publication.

2 Abstract

This white paper discusses how PCIe fabrics can be used to create a flexible, low-latency, high-performance fabric interconnect to a shared pool of GPUs and NVMe SSDs while still supporting standard host operating system drivers. Using dynamic partitioning and multi-host single root I/O virtualization (SR-IOV) sharing techniques, GPU and NVMe resources can be ‘composed’ or dynamically allocated to a specific host or set of hosts, allowing real-time allocation of resources to match workload requirements. Key concepts of PCIe fabrics and multi-host sharing of SR-IOV devices are also discussed.

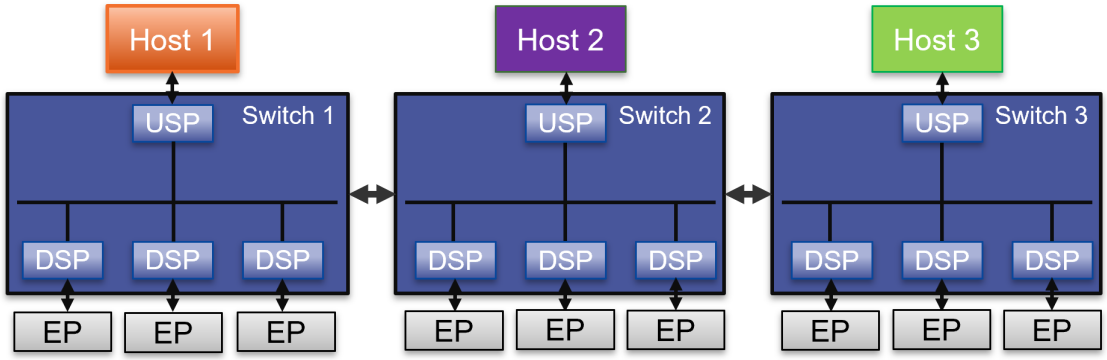
3 Introduction

As the use of GPUs for deep learning, artificial intelligence and machine learning increases, so does the demand for a more efficient deployment of GPU and NVMe resources. To effectively integrate these costly devices into systems, datacenter equipment designers require innovative technologies for efficiently sharing system resources among multiple hosts with a high-bandwidth, low-latency interconnection in disaggregated, composable architectures. PCIe is an effective, high-performance, and ubiquitous system interconnect, but there are limitations in the specification that limit its usage in such designs. This white paper discusses how PCIe fabrics are able to overcome the limitations of conventional PCIe tree-based systems to provide flexible, dynamic composition and sharing of system resources.

4 PCIe Hierarchy Restriction

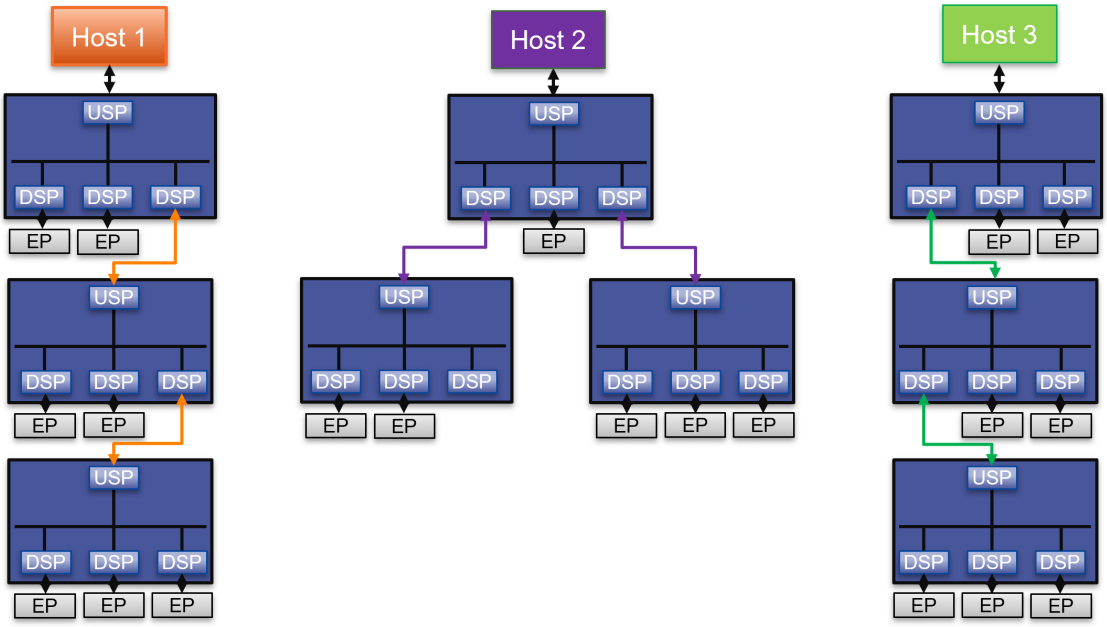
The PCIe standard defines a hierarchy domain as a rigid, hierarchical tree structure, which complicates the design of large scale systems involving multiple PCIe switches and host systems. For example, consider a system with three hosts and three switches, as shown in the following figure.

Figure 1 • Multi-Host Topology



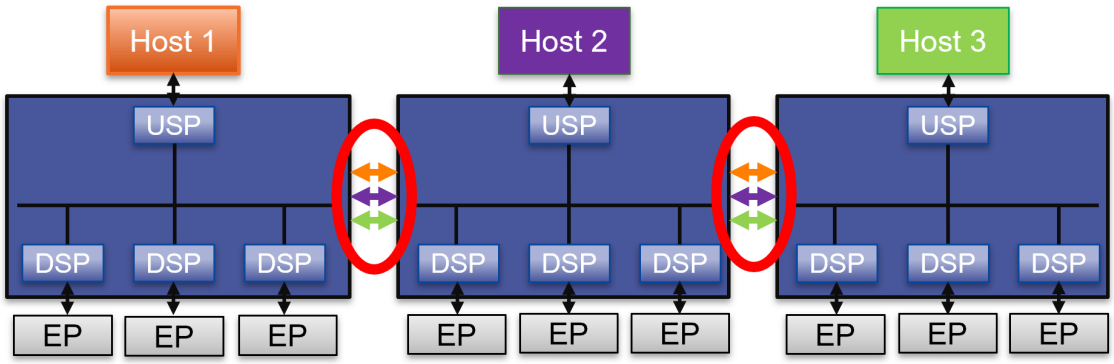
To maintain a PCIe-specification compliant hierarchy, Host 1 must have a dedicated downstream port (DSP) in Switch 1 connected to a dedicated upstream port (USP) in Switch 2 and a dedicated DSP in Switch 2 connected to a dedicated USP in Switch 3. Similar requirements exist for Host 2 and Host 3, as shown in the following figure.

Figure 2 • Hierarchy Requirements for Each Host



As a result, a simple, PCIe tree structure-based system requires three links between each switch dedicated to each host's PCIe topology. The inability to share these links between the hosts complicates the design and decreases system efficiency.

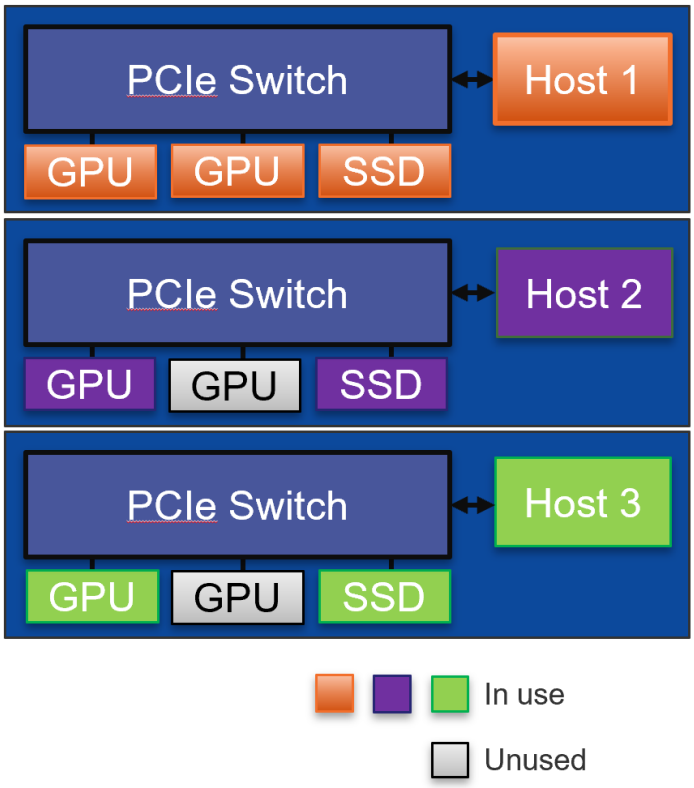
Figure 3 • PCIe Link Required for Each Host



5 PCIe Single Root Hierarchy Domain Restriction

A typical PCIe-specification compliant hierarchy domain only contains one root port. Standards have been defined for extending a hierarchy domain to support multiple roots (Multi-Root I/O Virtualization and Sharing Specification Revision 1.0), but this is a complicated capability that has not been implemented by a major, modern host CPU. The result is that unused PCIe devices are stranded in the hierarchy domain of the host that owns them. An example of such a scenario is shown in the following figure.

Figure 4 • Stranded, Unused PCIe EPs

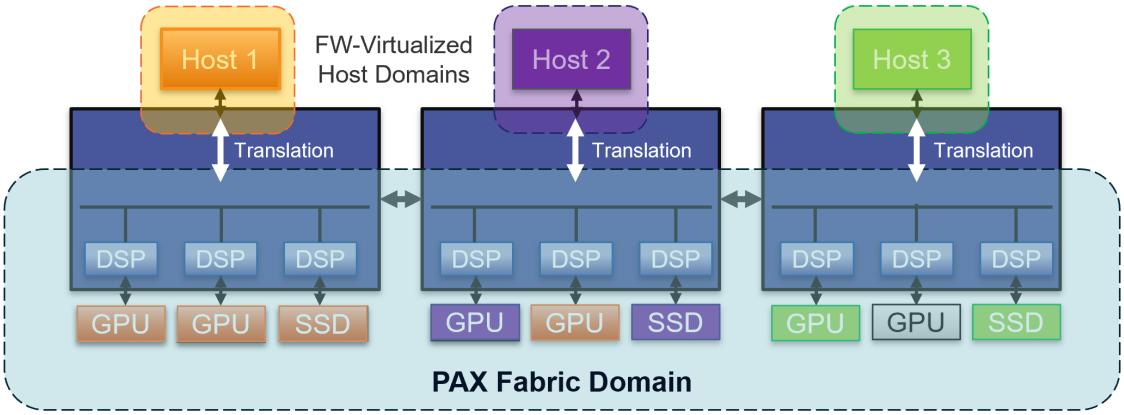


Host 1 has fully consumed its compute resources, but Hosts 2 and 3 are under-utilizing theirs. It is impossible for Host 1 to access the unused compute resources because they are outside of its hierarchy domain. Device sharing can be implemented using non-transparent bridging (NTB), but that requires the implementation of complicated, non-standard drivers and software to be developed for each type of shared PCIe device.

6 PCIe Fabrics for Scaling

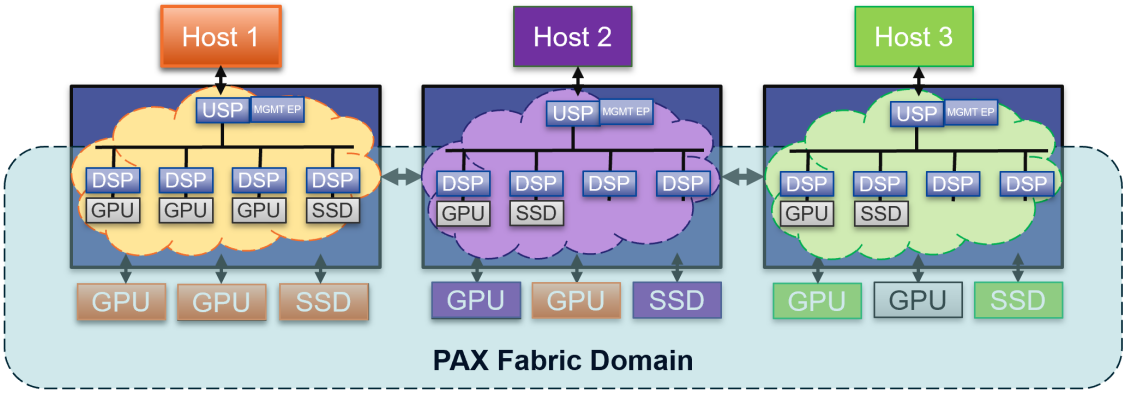
A PCIe fabric can be used to address issues with scaling a standard PCIe topology to multiple hosts and multiple switches. A PCIe fabric-based system implemented using Microchip's Switchtec PAX Advanced Fabric PCIe Switch is separated into two types of domains: a fabric domain containing all EPs and fabric links, and host domains for each connected host. Transactions from the host domains are translated to IDs and addresses in the fabric domain, and vice versa. Proprietary, non-hierarchical routing is used for traffic in the fabric domain. This allows the fabric links connecting the switches to be efficiently shared by all hosts in the system.

Figure 5 • Types of Domains



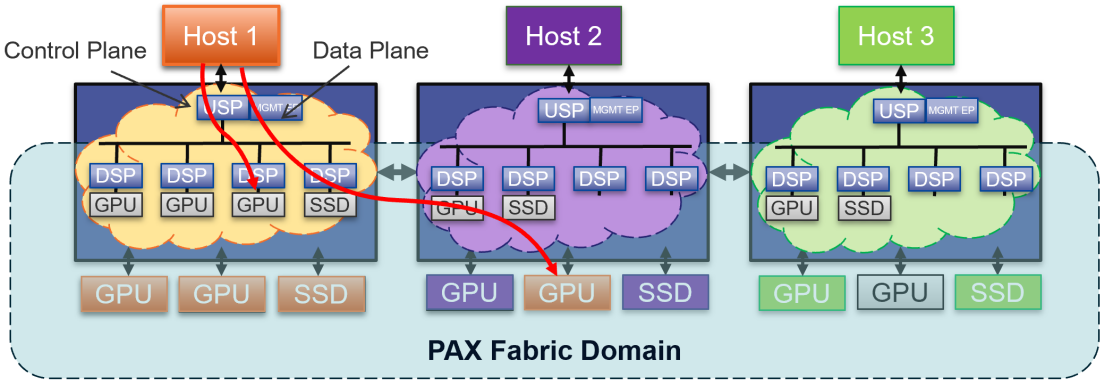
Firmware running on an embedded processor in the fabric PCIe switch intercepts all configuration plane traffic from the host, including the PCIe enumeration process. It virtualizes a simple, PCIe specification-compliant switch with a configurable number of DSPs.

Figure 6 • PCIe Switch Virtualization



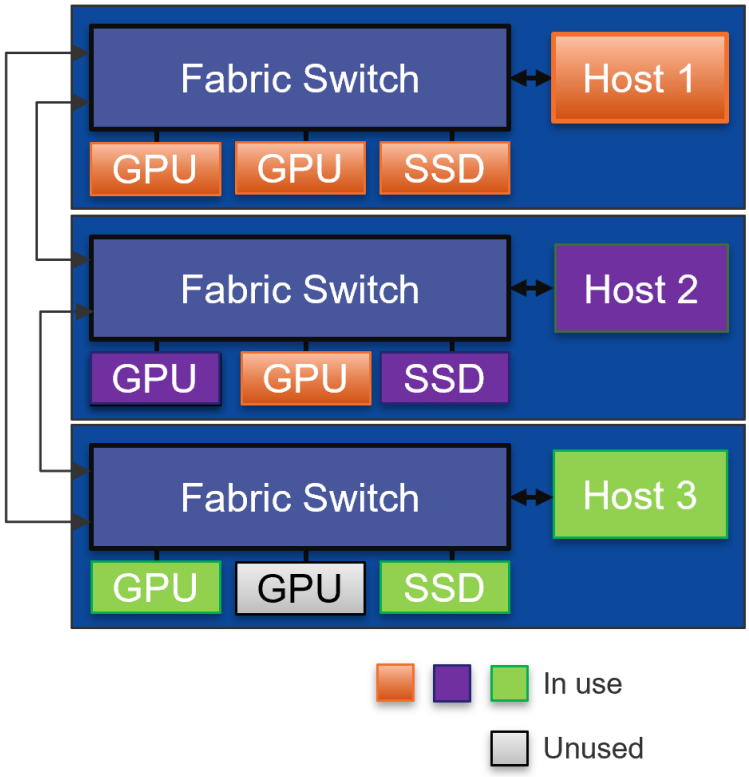
PCIe devices assigned to the host domain appear directly connected to the virtual PCIe switch. All control plane traffic will be routed to the switch firmware for processing, but data plane traffic will be routed directly to the PCIe EPs to ensure maximum system performance.

Figure 7 • Traffic Routing in a PAX PCIe Fabric



Unused devices in other host domains are no longer stranded and can be dynamically assigned based on each host's resource requirements. Peer-to-peer traffic is supported within the fabric to enable modern AI/ML applications. These system capabilities are presented to hosts in a PCIe specification-compliant manner, so standard drivers can be used.

Figure 8 • PCIe Fabric for EP Pooling

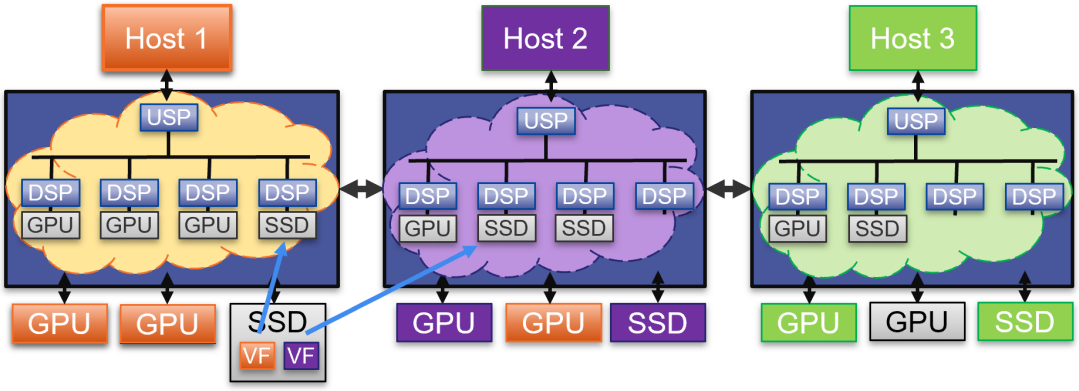


7 PCIe Fabrics for Multi-Host Sharing

Within the PCIe fabric, devices are assigned to hosts at a PCIe function-level granularity. As a result, multi-function devices can have individual functions assigned to different hosts, allowing for multi-host sharing. This capability can be used for devices with multiple functions and those that implement SR-IOV, a PCIe capability that allows a device to present multiple physical and virtual functions (VFs). Switch firmware can also be used to virtualize the configuration space of the VFs to alter their contents and capabilities. For example, the VFs of an SR-IOV capable NVM device can be modified to present the VF as a standard, single function NVM device. This allows devices to be shared among hosts using standard, in-box drivers.

Not all multi-function and SR-IOV capable devices are designed to support this model for multi-host sharing. Each PCIe function must be designed to support stand-alone operation with minimal requirements for coordinating with other functions. PAX evaluation platforms are available to test EP compatibility with multi-host sharing across a fabric.

Figure 9 • Multi-Host Sharing of SR-IOV VFs

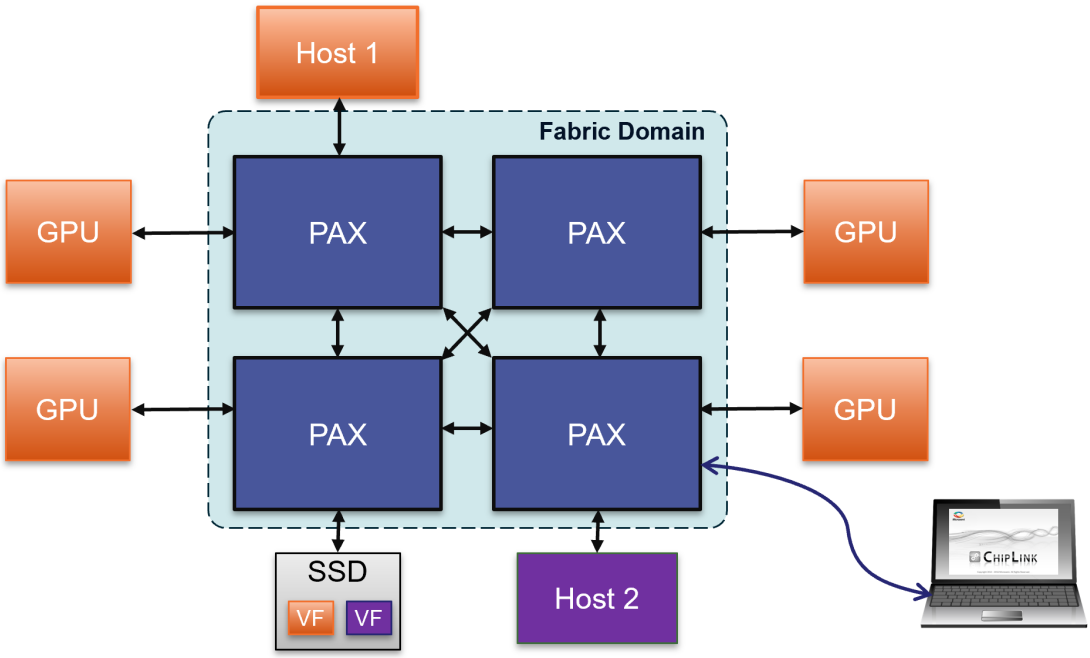


8 Demonstration

Microsemi has successfully proven the concepts presented in this paper with a real-world example. We have implemented a proof-of-concept system that demonstrates dynamic assignment of GPUs and multi-host sharing of SR-IOV SSDs. The hosts tested are running Windows Server 2016 and Ubuntu Server 16.04 LTS, neither of which require any custom drivers or software to facilitate the multi-host sharing. The hosts run traffic representative of actual AI/ML workloads, including Nvidia's CUDA peer-to-peer traffic benchmarking utility, p2pBandwidthLatencyTest, and training the cifar10 image classification Tensorflow model.

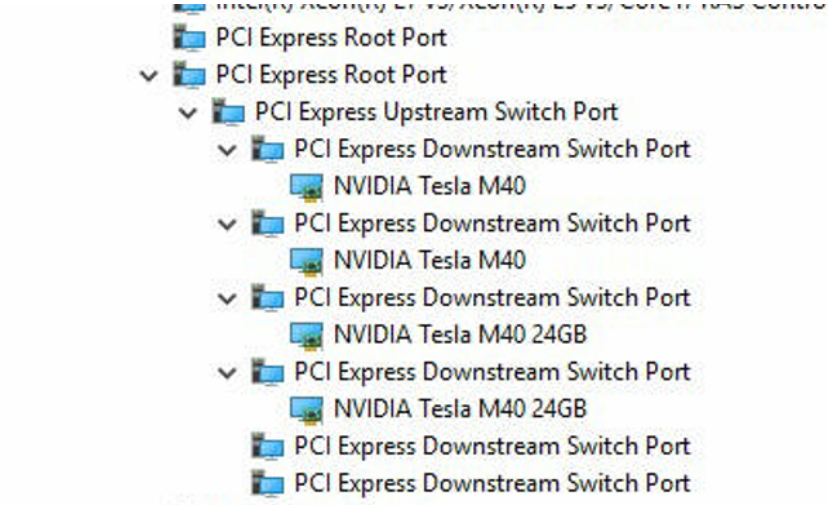
The demo system comprises four PAX fabric PCIe switches, four Nvidia Tesla GPGPUs, one Samsung PM1725a NVM device with SR-IOV support, and two Supermicro servers interconnected as illustrated below. The embedded switch firmware handles the low-level configuration and management of the switch hardware, so the demo system is managed from Microsemi's debug and diagnostics utility, ChipLink, connected over a simple UART management interface.

Figure 10 • Demo System Block Diagram



Initially, all GPUs are assigned to Host 1, which is running Windows Server 2016, to increase the performance of the AI training. The virtual switch presented to the host can be seen in the Windows Device Manager tool. All GPUs appear as though they are directly connected to the virtual switch; the fabric links and complexities of the physical topology are obscured from the host.

Figure 11 • Virtual Switch Topology in Device Manager



The p2pBandwidthLatencyTest utility is used to measure the bandwidth of GPU-to-GPU transfers across the PCIe fabric.

Figure 12 • p2pBandwidthLatencyTest Performance Excerpt from Host 1

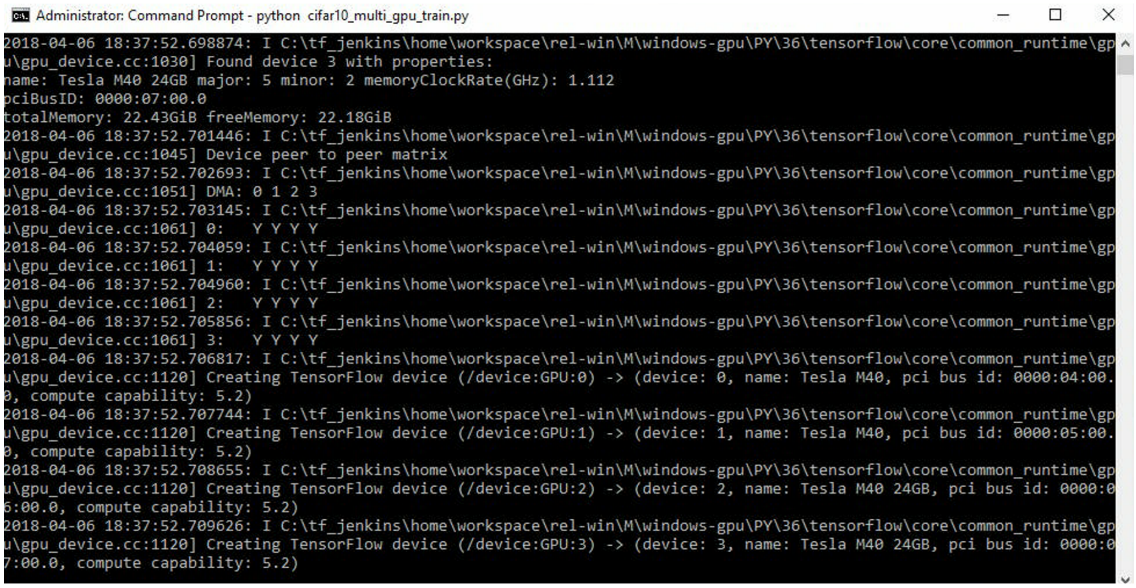
```
P2P Connectivity Matrix
     D\D     0     1     2     3
     0       1     1     1     1
     1       1     1     1     1
     2       1     1     1     1
     3       1     1     1     1
```

```
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
```

With the GPUs discovered and the standard Nvidia drivers loaded, an AI workload is started. In this example, the user is training the cifar10 image classification algorithm. The training algorithm will be distributed across all four GPUs.

Figure 13 • cifar10 Multi-GPU Training Output Excerpt



When the workload completes, the user can release two of the GPUs back into the fabric pool. Now, we'll assign one of the SR-IOV drive's VFs to each host. On Host 1, the drive appears as a "Standard NVM Express Controller" using the standard, in-box Windows driver for NVM devices.

Figure 14 • SR-IOV NVM VF Virtualized as Standard NVM Controller

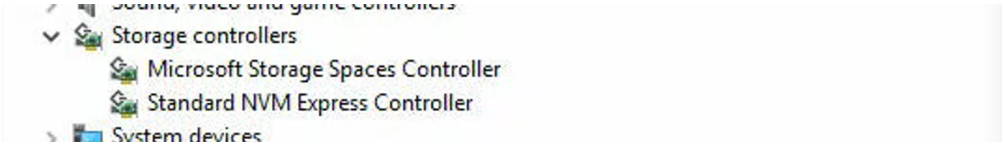
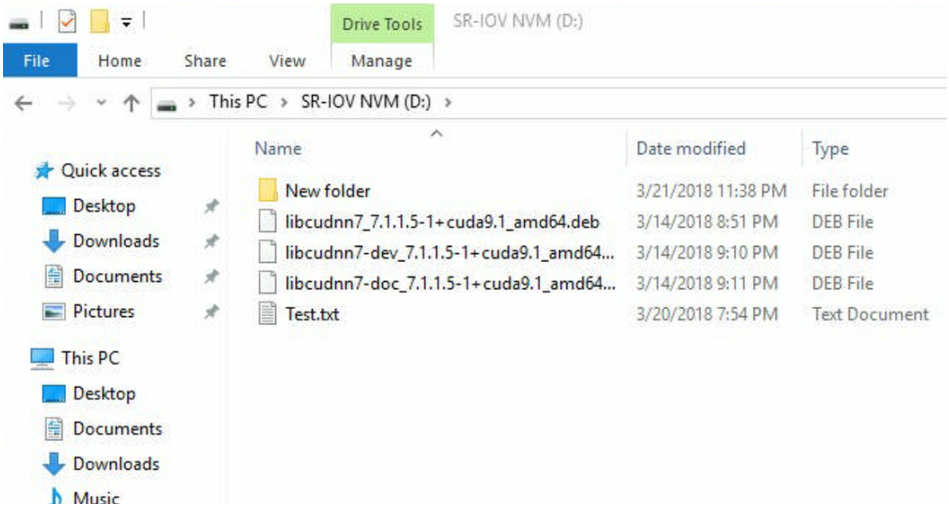


Figure 15 • SR-IOV NVM VF Partition

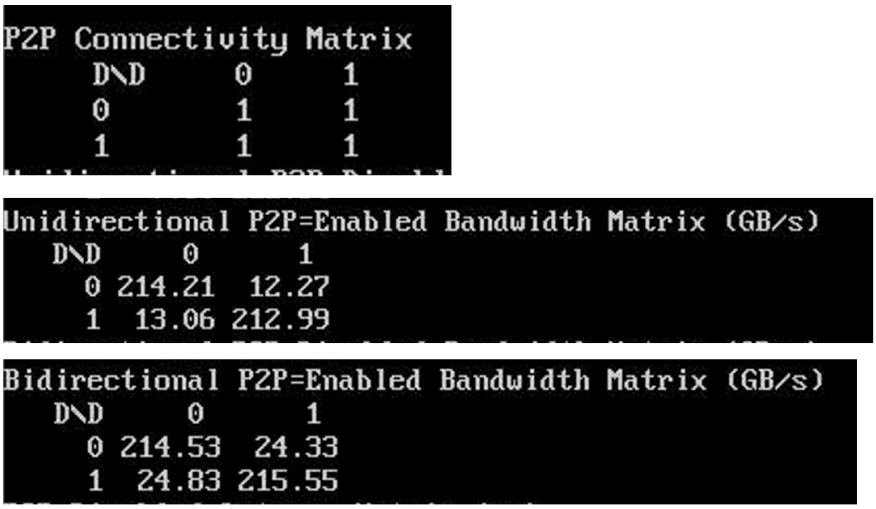


Host 2 is running Ubuntu Server 16.04 LTS, and its PCIe topology is also a simple switch with locally attached GPUs and a standard NVM device.

Figure 16 • Virtual Switch Topology in Linux

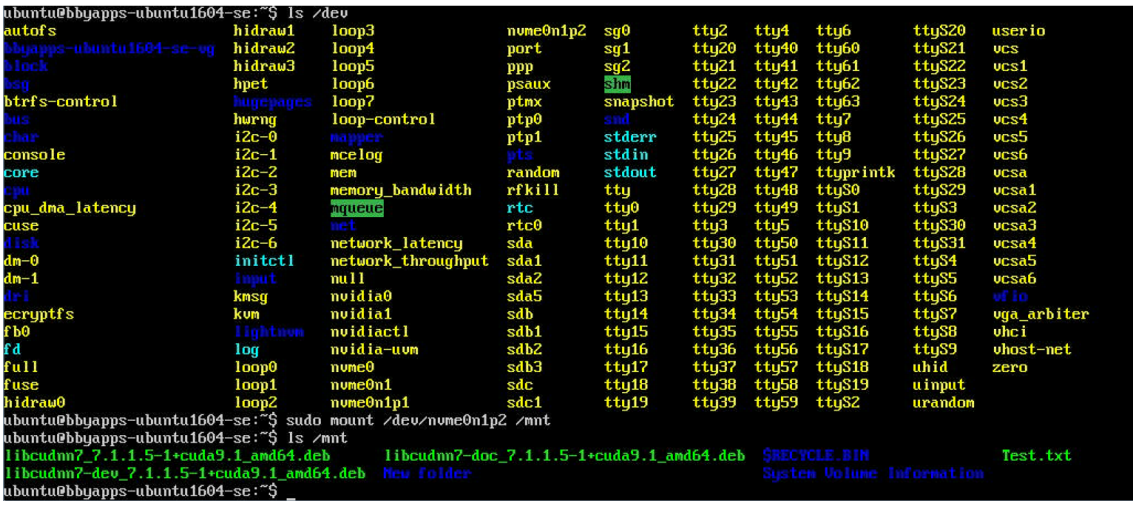
The p2pBandwidthLatencyTest utility produces similar performance results on Host 2.

Figure 17 • p2pBandwidthLatencyTest Performance Excerpt from Host 2



The NVM VF is presented to Host 2 as a standard NVM device, so its standard, in-box drivers can be used.

Figure 18 • SR-IOV NVM VF in Device Listing and Mounted Volume Contents



As demonstrated above, the virtual PCIe switch and all dynamic assignment operations are presented to the host as fully PCIe specification-compliant, enabling the hosts to use standard, in-box drivers. The embedded switch firmware provides a simple management interface so the PCIe fabric can be configured and managed by a light-weight external processor. Device peer-to-peer transactions are enabled by default and require no additional configuration or management from an external fabric manager.

9 Conclusion

PCIe fabrics provide the high-bandwidth, low-latency interconnections required to implement the next generation of disaggregated, composable architectures. System designers can provision or share system resources in scalable, efficient ways using standard in-box drivers.

Microchip's Switchtec PAX Advanced Fabric PCIe switches enable new architectures for next-generation solutions. They provide a scalable, low-latency, high bandwidth, and cost-effective option for system design. With its virtualization of switch topologies and multi-host sharing of PCIe devices using standard, in-box drivers, PAX significantly reduces time to market and storage costs and simplifies system development. For more information, visit Switchtec PAX Advanced Fabric PCIe Switches or contact sales.support@microsemi.com.

Microsemi Headquarters
One Enterprise, Aliso Viejo,
CA 92656 USA
Within the USA: +1 (800) 713-4113
Outside the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996
Email: sales.support@microsemi.com
www.microsemi.com

ESC-2190732