Best Practices for Deploying and Managing GPU Clusters

Dale Southard, NVIDIA

dsouthard@nvidia.com

NVIDIA

About the Speaker and You

[Dale] is a senior solution architect with NVIDIA (I fix things). I primarily cover HPC in Gov/Edu/Research and cloud computing. In the past I was a HW architect in the LLNL systems group designing the vis/post-processing solutions.

[You] are here because you are interested in designing, deploying, and managing clusters that use GPUs for High Performance Computing.

Outline

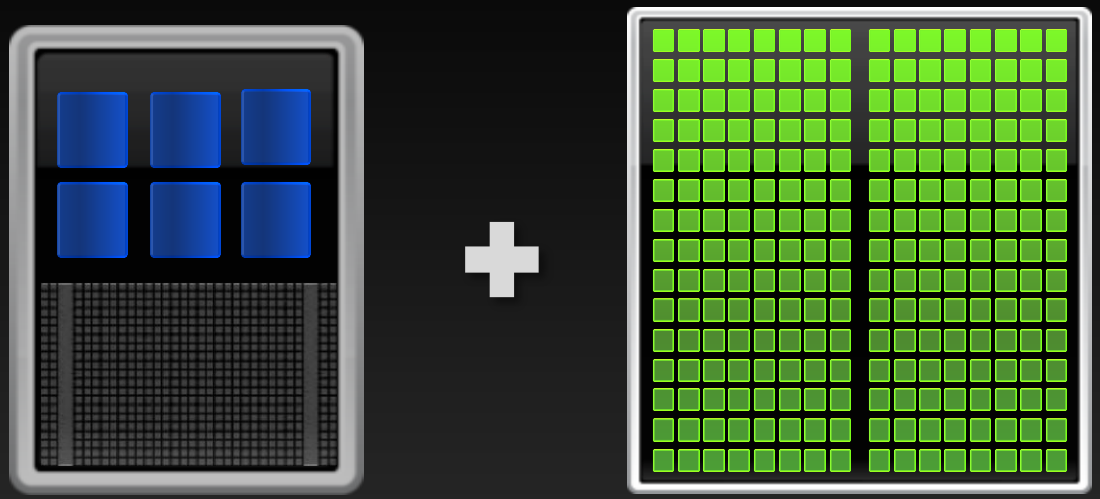
- Nomenclature
- Selecting the Correct Hardware
- Driver Installation and System Configuration
- Monitoring

Nomenclature


GeForce
Quadro
NVIDIA
Tegra
Tesla

CUDA Accelerates Computing

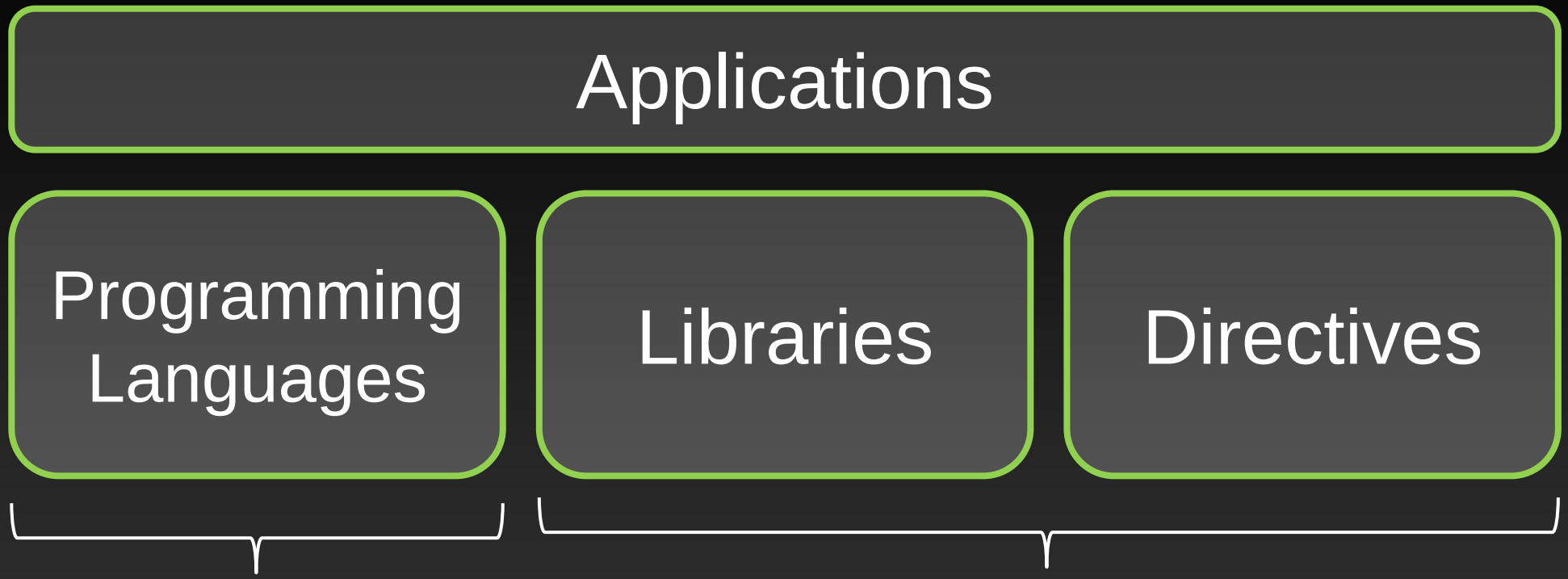
Choose the right processor for the right task



Several sequential cores

Hundreds of parallel cores

3 Ways to Accelerate Your Apps



GPUDirect

GPUDirect is a long term effort to eliminate the CPU bottleneck

- Version 1: GPU and NIC to shared pinned memory
- Version 2: Peer-to-peer memory access between GPUs
- RDMA: Direct access to GPU memory from other devices.

Hardware Selection

Choosing the Correct GPU

- Passively Cooled
- Out-of-Band Monitoring
- Chassis/BMC Integration
- More performance

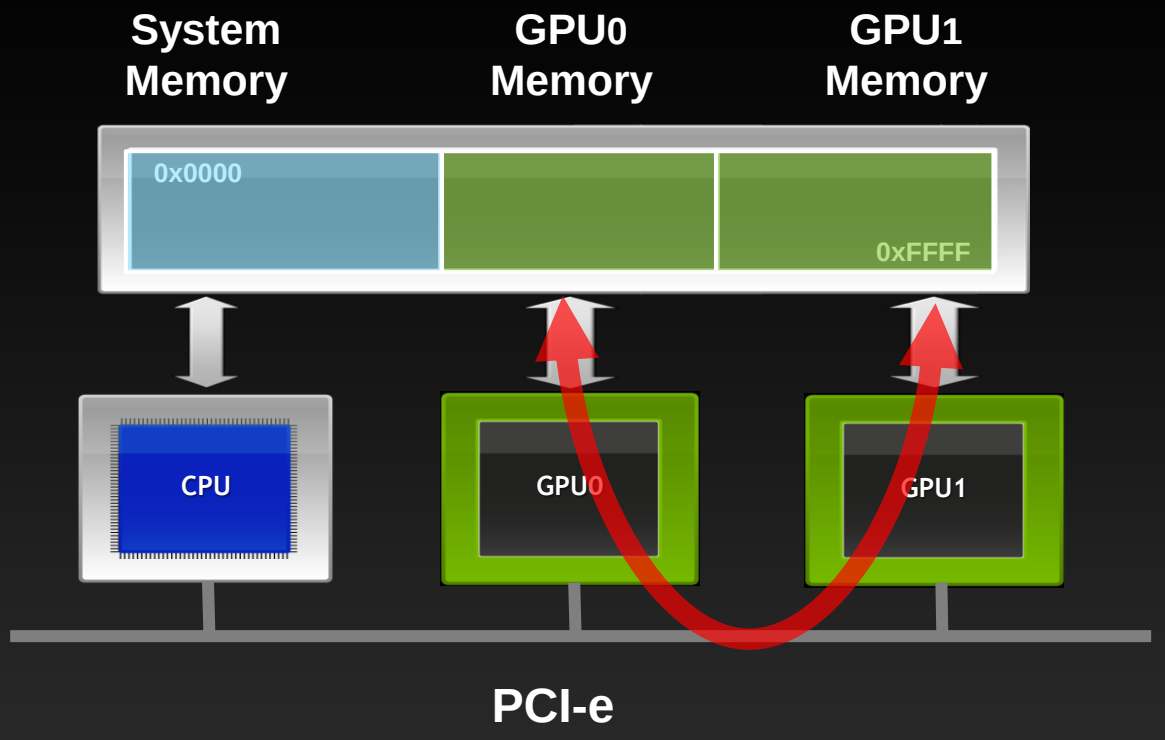
Tesla M-series is Designed for Servers

(C-series GPUs will work, but those target workstation environments, not servers)

Doing it Wrong

QPI @3.2	12.8 GB/s
HT 3.0 (16b)	10.4 GB/s
PCIe gen2	8 GB/s
PCIe gen3	16 GB/s

UVA is Driving Server Design

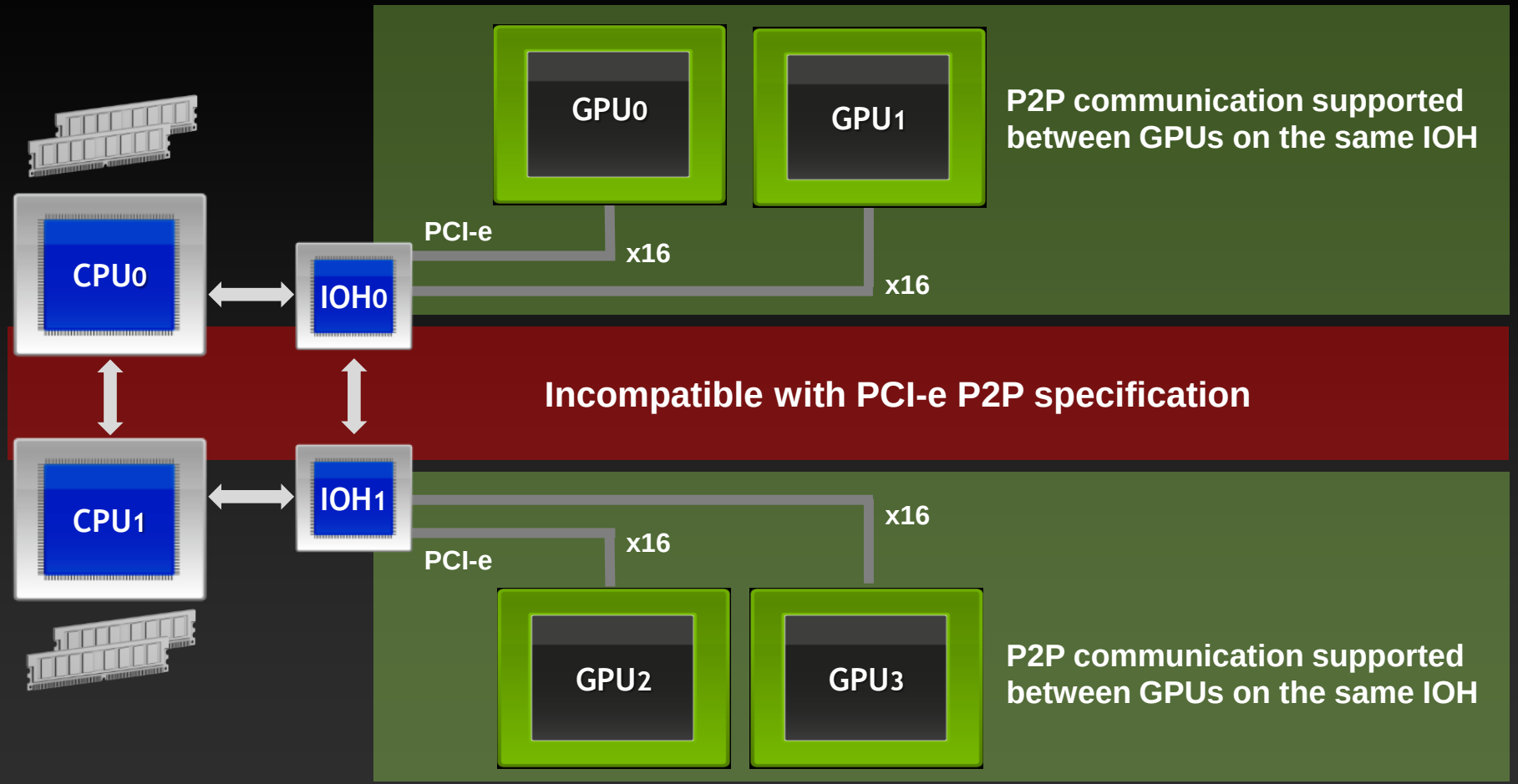


Direct Transfer & Direct Access between GPUs

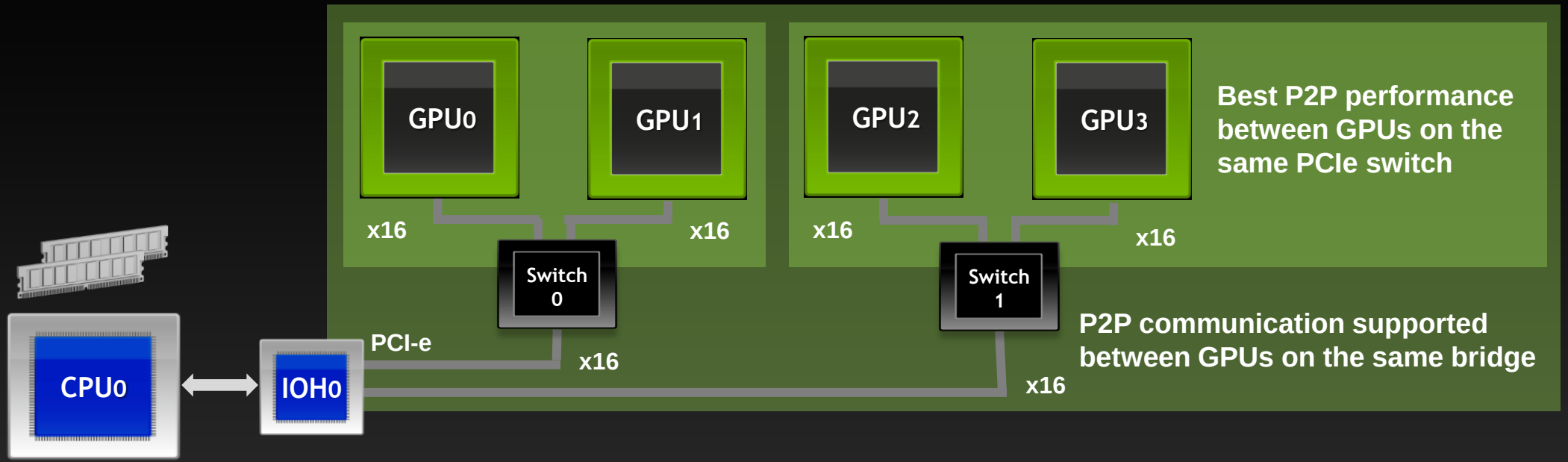
without going through host memory

Topology Matters for P2P Communication

P2P Communication is Not Supported Between Bridges

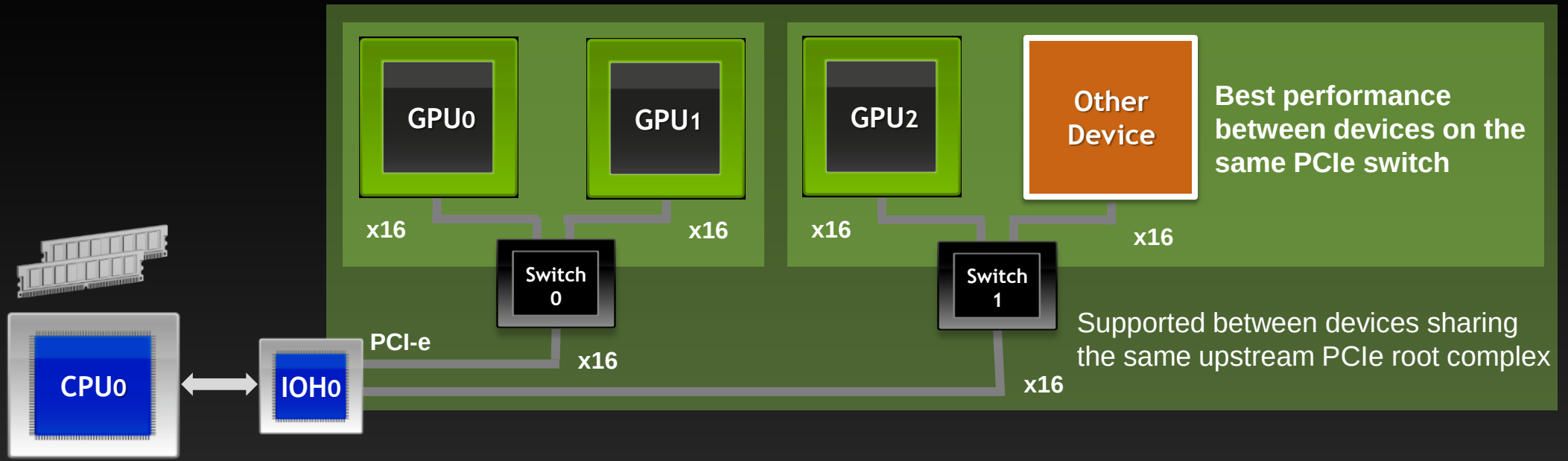


Topology Matters, PCIe Switches



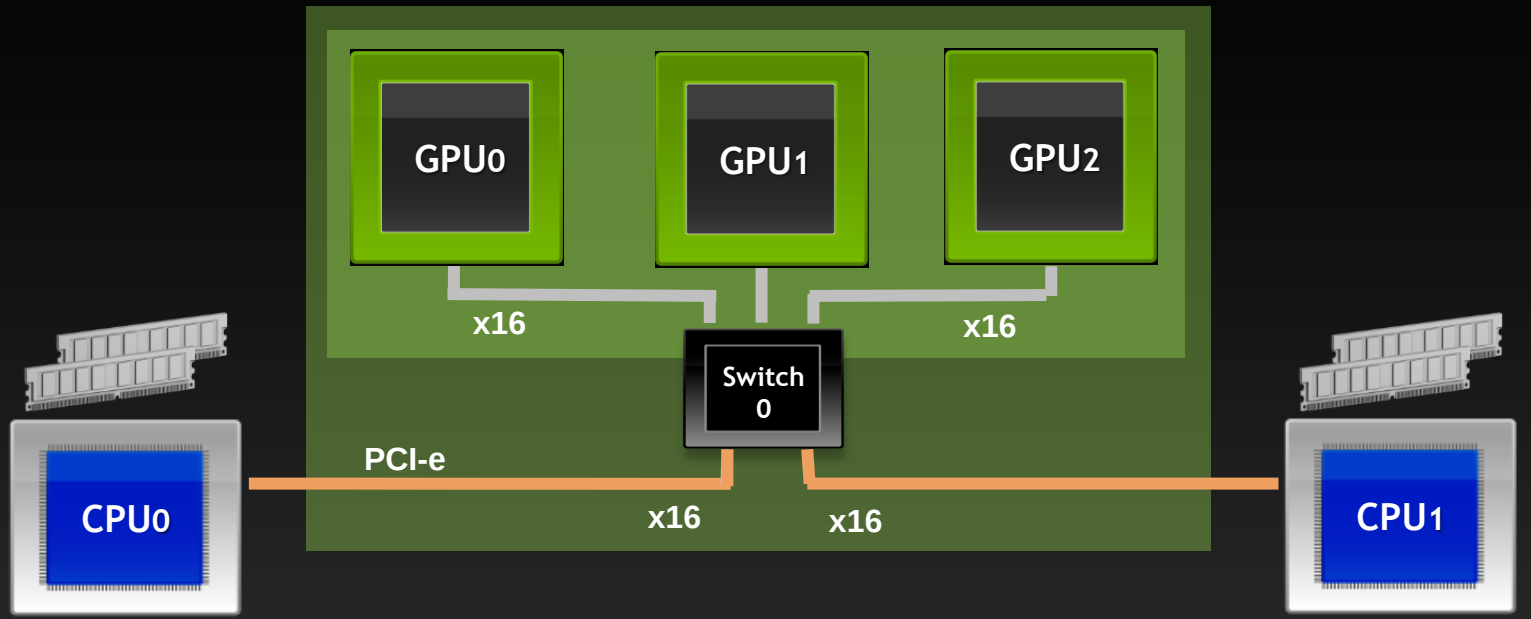
PCI-e Switches: Fully Supported

Topology Matters, GPUDirect RDMA



PCI-e Switches: Fully Supported

Non-transparent Switches



Non-transparent PCI-e Switches: Future Flexibility

Driver Installation and System Configuration

Driver Installation

- NVIDIA provides drivers with a runnable shell script installer
- Most cluster managers repackage the driver into a format that works with their provisioning system
  - Install into a tree
  - Extract and unpack the tarfile
- If you are also using OpenGL
  - UseDisplayDevice none and a virtual screen
  - Be wary of dual ownership of files with Mesa

Dealing with Runlevel 3

Most clusters operate at runlevel 3 (no xdm), so best practice is to configure the GPUs from an init.d script:

- `modprobe nvidia`
- `mknod` devices
- Assert that ECC is set correctly
- Set compute-exclusive mode
- Set persistence

NVIDIA provides both command -line (`nvidia-smi`) & API (NVML)

Compute Mode

The Compute Mode setting controls simultaneous use

- `DEFAULT` allow multiple simultaneous processes
- `EXCLUSIVE_THREAD` allows only one context
- `EXCLUSIVE_PROCESS` one process, but multiple threads
- `PROHIBITED`

Can be set by command -line (`nvidia-smi`) & API (NVML)

Persistence Mode

Controls driver unloading

- Persistence mode set
  - Driver does not unload when GPU is idle
  - Slightly lower idle power
  - Faster job startup
- Persistence mode not set
  - If ecc is on, memory is cleared between jobs

Can be set by command -line (`nvidia-smi`) & API (NVML)

Job Scheduling

For time-sharing a node use $CUDA_VISIBLE_DEVICES:

```
$ ./deviceQuery -noprompt | egrep "^Device"
 Device 0: "Tesla C2050"
 Device 1: "Tesla C1060"
 Device 2: "Quadro FX 3800"

$ export CUDA_VISIBLE_DEVICES="0,2"

$ ./deviceQuery -noprompt | egrep "^Device"
 Device 0: "Tesla C2050"
 Device 1: "Quadro FX 3800"
```

Several batch systems and resource mangers support GPUs as independent consumables using $CUDA_VISIBLE_DEVICES

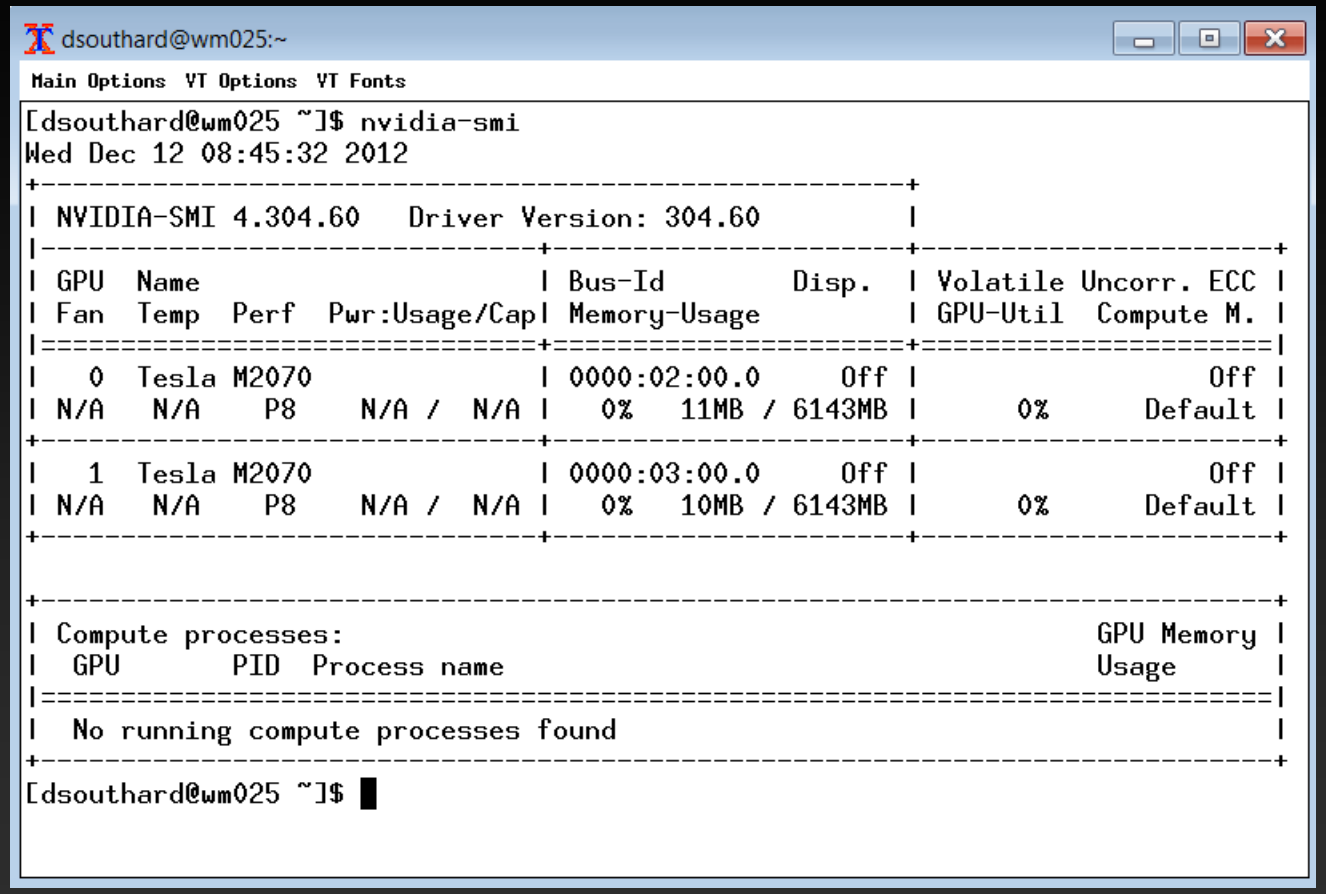
Resource Limits

- UVA depends on allocating virtual address space
- Virtual address space != physical ram consumption
- Use cgroups, not ulimit

Several batch systems and resource mangers support cgroups directly or via plugins.

Monitoring

Monitoring with nvidia-smi



Long form nvidia-smi

- nvidia-smi also provides more extensive output with -q
  - Information on PCI, GPU, VBIOS, etc
  - PCI link info
  - Thermals
  - Performance, Clocks, Throttling/Slowdown
  - Memory usage
  - ECC errors (including location)

But not in a format that’s easy to parse

NVML

NVML is a monitoring Library

- Environmental and utilization metrics are available
- NVML can be used from C, Python, or Perl
- NMVL has been integrated into Ganglia gmond.

http://developer.nvidia.com/nvidia-management-library-nvml

Healthmon

Healthmon is a quick running diagnotic

- Designed to be used between batch jobs
- Uses the existing driver (not a specialized diag)
- Provides simplified output

http://developer.nvidia.com/tesla-deployment-kit

Finding Problems

- Monitoring though nvidia-smi or NVML
- Watching for Xid errors in syslog
- PCIe parity via EDAC

```
modprobe    edac_mc
echo 1 > /sys/devices/system/edac/pci/check_pci_parity
```

- CUDA-gdb, user feedback, and testing

Handling Bad Devices

- Three different enumeration systems:
  - PCIe
  - CUDA runtime
  - nvidia-smi
- Do not assume that the three enumerations are consistent!
- PCIe device ID, serial number, and UUID are consistent

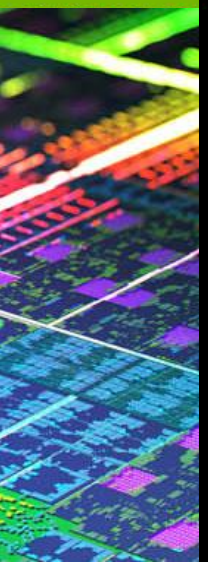
Always have operators check serial number of pulled HW

Work with your OEM for Triage and RMA

When in Doubt…

- Developer forums at nvida.com
- Forums at stackoverflow.com
- Contact your OEM
- Reach out to NVIDIA

Questions



GTC 2013:The Premier Event in Accelerated Computing

Registration is Open!

- Four days – March 18-21, San Jose, CA
- Three keynotes
- 300+ sessions
- One day of pre-conference developer tutorials
- 100+ research posters
- Lots of networking events and opportunities

Visit www.gputechconf.com for more info.