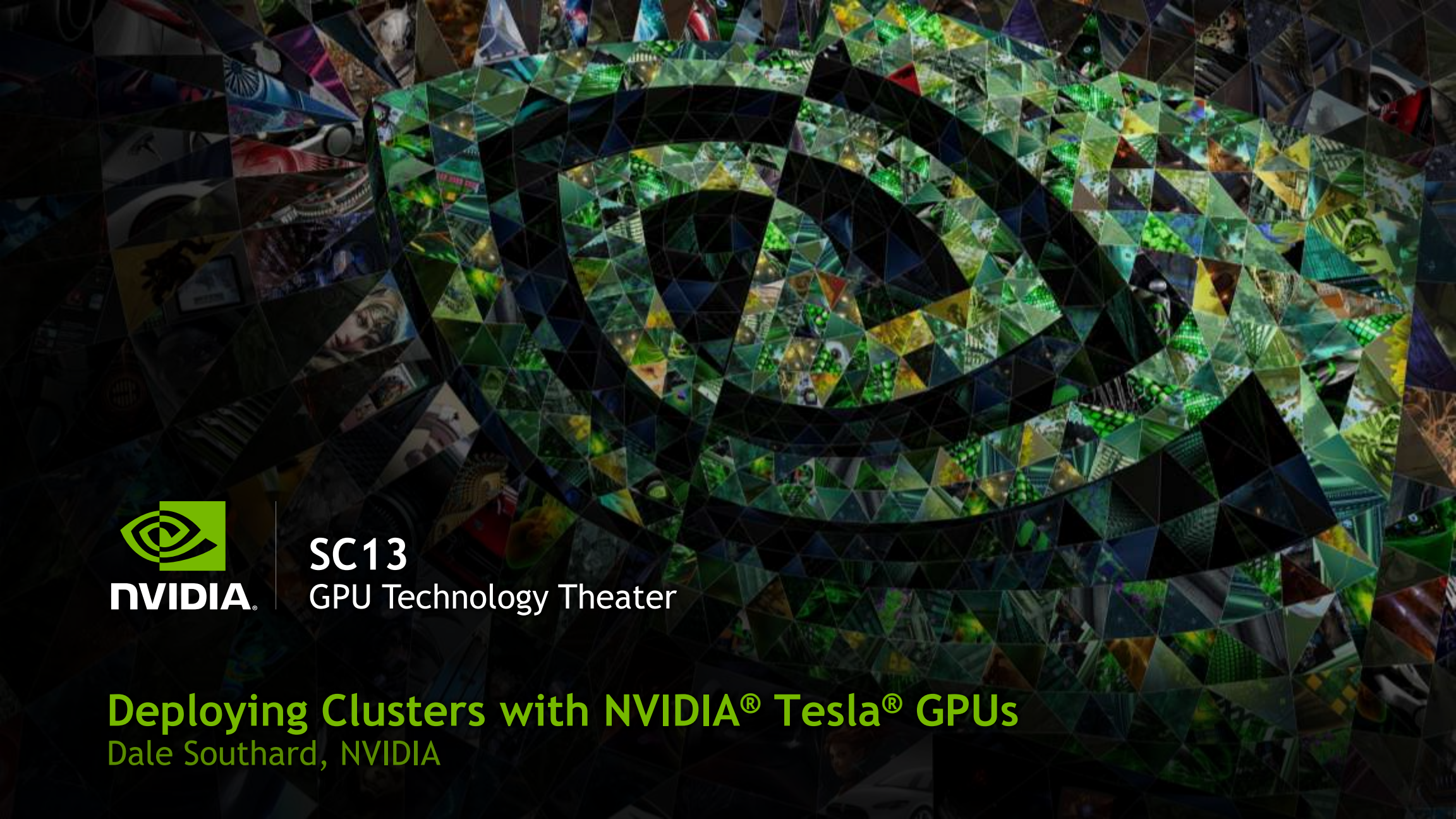


NVIDIA

SC13
GPU Technology Theater

Deploying Clusters with NVIDIA® Tesla® GPUs

Dale Southard, NVIDIA

About the Speaker

[Dale] is a senior solution architect with NVIDIA (I fix things).

I primarily cover HPC in and cloud computing. In the past I was a HW architect in the LLNL systems group designing the vis/post-processing solutions.

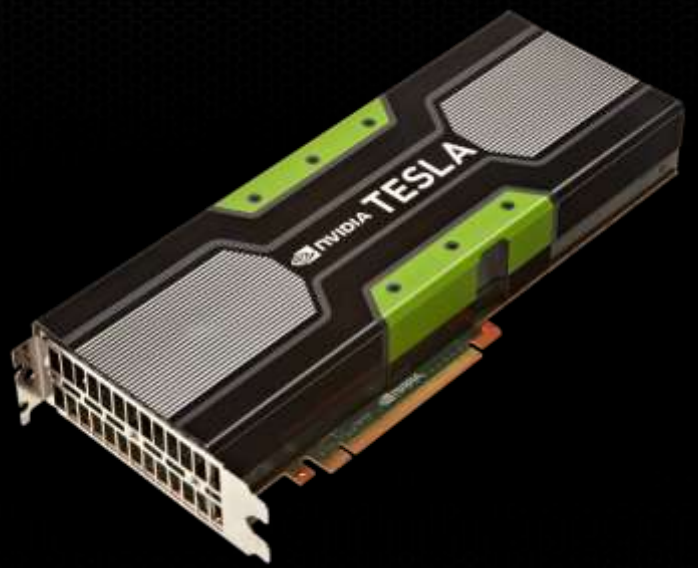
Outline

- Hardware Selection
- Running the Cluster
- Monitoring the Cluster

Hardware Selection

Start with the Correct GPU

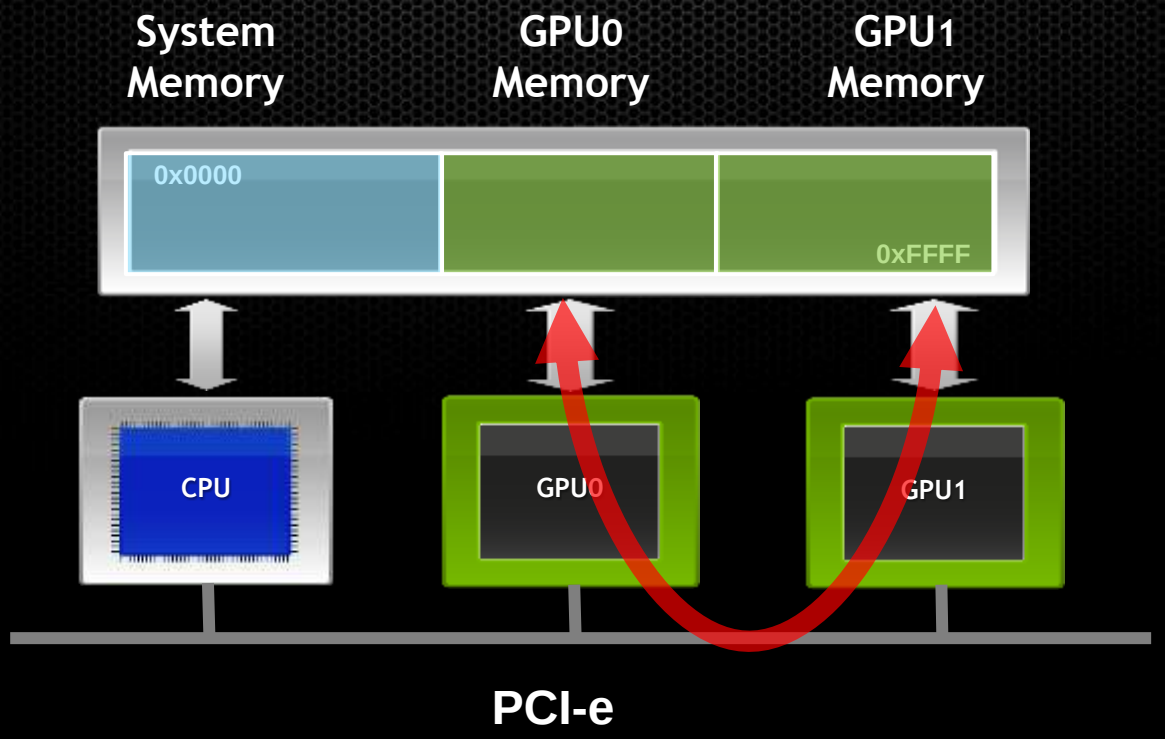
- Passively Cooled
- Higher Performance
- Chassis/BMC Integration
- Out-of-Band



Tesla M-series is Designed for Servers

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

UVA and UM Drive Server Design



Direct Transfer & Direct Access between GPUs

works best between GPUs on same PCI switch or root complex

SBIOS Configuration - Cooling Matters

- Tesla M (and X) modules have passive heatsinks
  - Depend on chassis fans for airflow
  - Communicate thermals to BMC
- Some systems require specific fan settings
- Some systems require baffles or blanks or other airflow directors
- During burn-in/acceptance
  - Verify that fans are correctly set
  - Verify that your workload performance is consistent on all nodes

Running the Cluster

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)

CUDA MPS

Cuda Multi Process Service

- Allows multiple processes to share a single context to the GPU
- Allows more efficient kernel scheduling
- Useful for MPI

Start and stop with `nvidia-cuda-mps-control`

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:

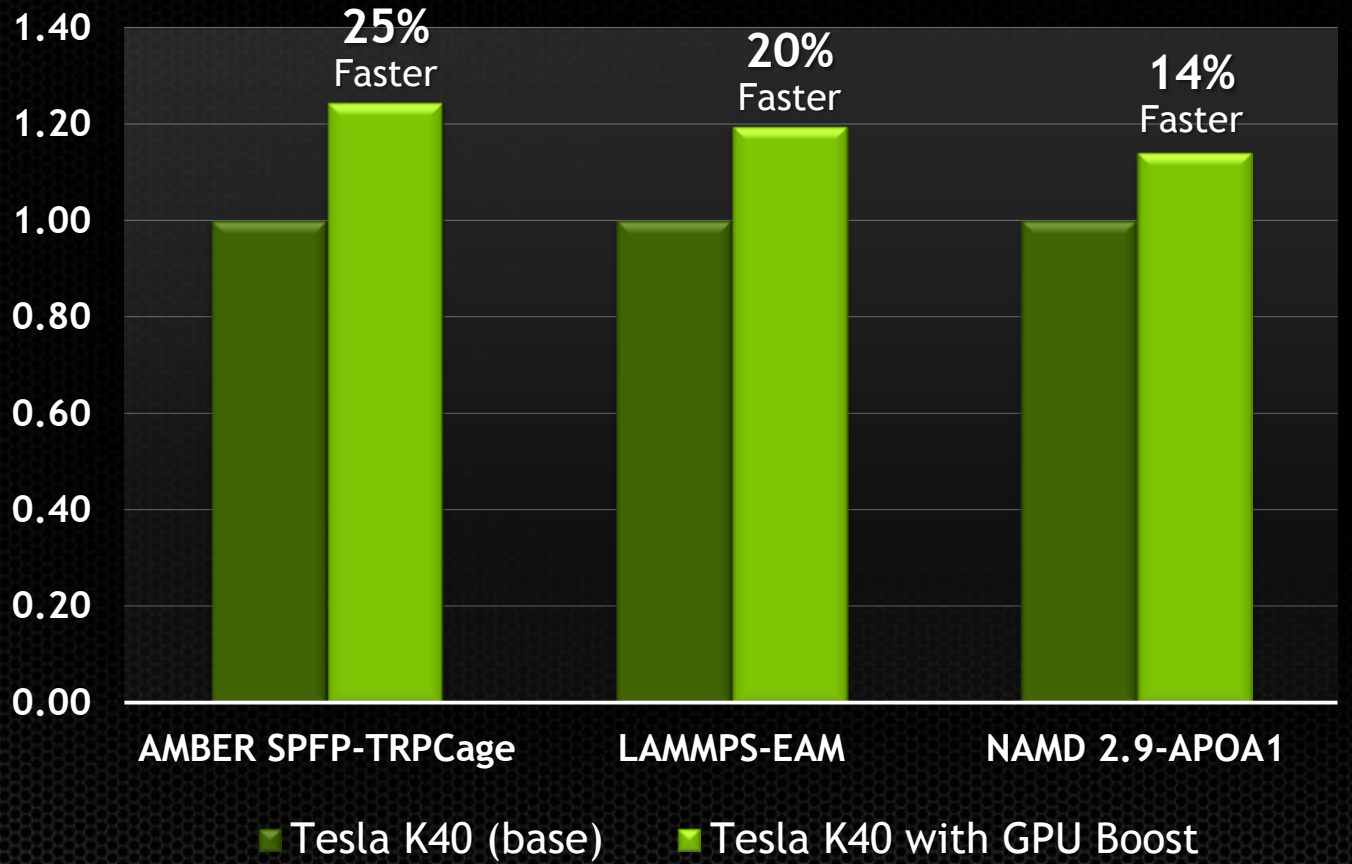
```
$ ./deviceQuery -noprompt | egrep "^Device"
 Device 0: "Tesla C2050"
 Device 1: "Tesla C1060"
 Device 2: "Quadro FX 3800"

$ 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

K40 GPU Boost

Up to 25% Extra Performance on Applications

Use Power Headroom to Run at Higher Clocks



Paradigm 17% Faster

OpenEye 13% Faster

ANSYS 11% Faster

K40 GPU Boost

- GPU boost can be controlled by `nvidia-smi`
  - `nvidia-smi -q -d SUPPORTED_CLOCKS`
  - `nvidia-smi -ac <MEM clock, Graphics clock>`
  - `nvidia-smi -q -d CLOCK` shows current mode
  - `nvidia-smi -rac` resets all clocks
  - `nvidia-smi -acp 0` allows non-root to change clocks
- Changing clocks does not affect power cap
  - But power cap can be adjusted with `nvidia-smi -pl`

Boost is not persistent – clocks reset when driver unloads

Does NVIDIA Still Do Graphics?

Yes

xorg.conf Tips and Tricks

- Xorg usually relies on automatic detection of displays
  - But tesla has no display, so no EDID
  - No display, X gets angry
  - Solution is to setup xorg with a virtual screen and no display
  - Nvidia-xconfig can automate that:

```
nvidia-xconfig --virtual=1280x1024 --use-display-device=none
```

- If mixing CUDA and graphics, the X watchdog timer can cause issues
  - Add `Option "Interactive" "true"` to xorg.conf

GPU Operation Mode (GOM)

- Disabling graphics features to optimize for compute
  - Support on Kepler GK110 based K20/K20X/K40 (not on C-Class K20)
- Modes:
  - All On – All features are on (including graphics)
  - Compute – Graphics feature off, only compute tasks can run
  - Low Double Precision – Graphics are on, but reduced FP64 features

Persistent, but currently requires reboot to change modes

X11 and Batch Jobs

- Most sites don’t want X11 servers running persistently
  - Perf impact is minimal, but dealing with Xauth and DISPLAY is painful
- Have users use Xinit to launch them on demand:

```
xinit /your/program/here
```

- Trick I use: `xinit /bin/sleep 2m`

Monitoring the Cluster

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 (volatile and persistent counts)

But not in a format that’s easy to parse

NVML

NVML is a monitoring and management 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

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


Thank You
NVIDIA
SC13
GPU Technology Theater
Dale Southard, <dsouthard@nvidia.com>