GPU TECHNOLOGY CONFERENCE

Cluster Monitoring and Management Tools

RAJAT PHULL, *NVIDIA SOFTWARE ENGINEER*

ROB TODD, *NVIDIA SOFTWARE ENGINEER*

MANAGE GPUS IN THE CLUSTER

...

...

Administrators,
End users
Middleware Engineers

NVIDIA

Monitoring/Management Tools

- NVML
- Nvidia-smi
- Health Tools

NVIDIA MANAGEMENT PRIMER

- NVIDIA Management Library
  - Provides a low-level C API for application developers to monitor, manage, and analyze specific characteristics of a GPU.

```
nvmlDeviceGetTemperature(device, NVML_TEMPERATURE_GPU, &temp);
```

- NVIDIA System Management Interface
  - A command line tool that uses NVML to provide information in a more readable or parse-ready format
  - Exposes most of the NVML API at the command line

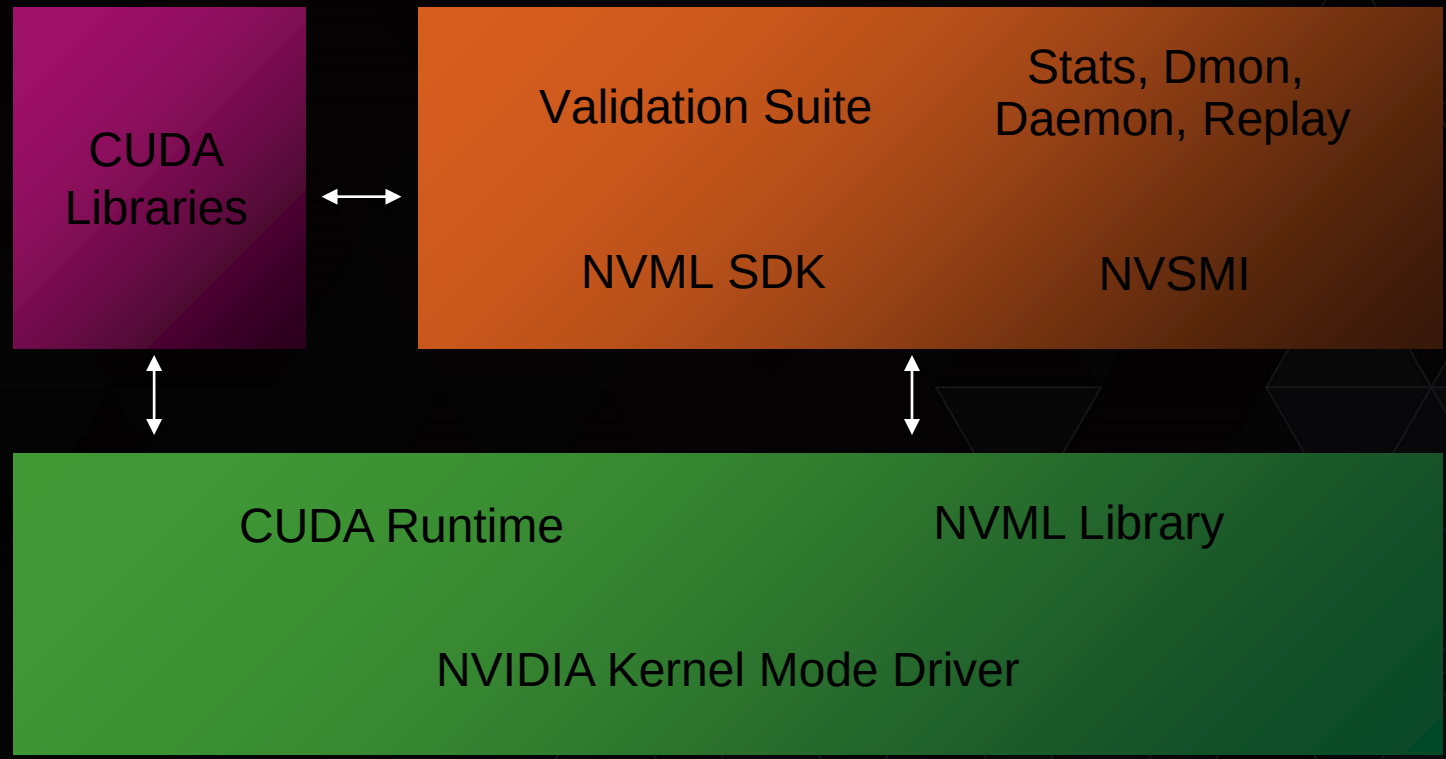
```
# nvidia-smi --query-gpu=temperature.gpu --format=csv
```

- Health Tools

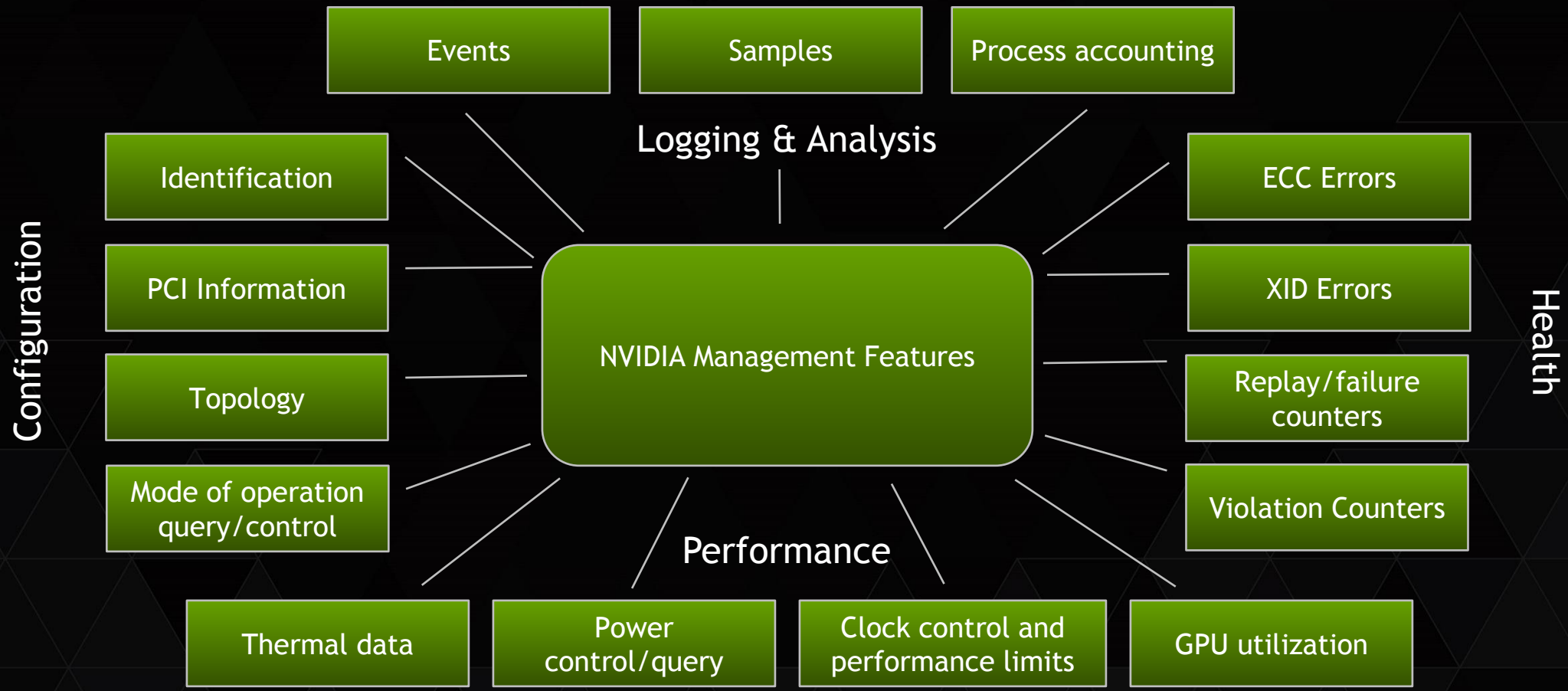
SOFTWARE RELATIONSHIPS

Key SW Components

- CUDA Toolkit
- NV Driver
- GPU Deployment Kit



MANAGEMENT CAPABILITIES



NVML EXAMPLE WITH C

```
#include “nvml.h”

int main()
{
    nvmlReturn_t result;
    nvmlPciInfo_t pci;
    nvmlDevice_t device;

    // First initialize NVML library
    result = nvmlInit();
    if (NVML_SUCCESS != result) {
        printf("Failed to initialize NVML: %s\n", nvmlErrorString(result));
        return 1;
    }
    result = nvmlDeviceGetHandleByIndex(0, &device);
    (check for error...)

    result = nvmlDeviceGetPciInfo(device, &pci);
    (check for error...)

    printf("%d. %s [%s]\n", i, name, pci.busId);

    result = nvmlShutdown();
    (check for error...)
}
```

NVML EXAMPLE WITH PYTHON BINDINGS

Errors are handled by a “raise NVMLError(returncode)”

https://pypi.python.org/pypi/nvidia-ml-py/

```
import pynvml

pynvml.nvmlInit()
device = nvmlDeviceGetHandleByIndex(0);
pci = pynvml.nvmlDeviceGetPciInfo(device);

print pci.busId

pynvml.nvmlShutdown();
```

CONFIGURATION

- Identification
  - Device handles: ByIndex, ByUUID, ByPCIBusID, BySerial
  - Basic info: serial, UUID, brand, name, index
- PCI Information
  - Current and max link/gen, domain/bus/device
- Topology
  - Get/set CPU affinity (uses *sched_affinity* calls)
- Mode of operation

ECC SETTINGS

- Tesla and Quadro GPUs support ECC memory
  - Correctable errors are logged but not scrubbed
  - Uncorrectable errors cause error at user and system level
  - GPU rejects new work after uncorrectable error, until reboot
- ECC can be turned off – makes more GPU memory available at cost of error correction/detection
  - Configured using NVML or nvidia-smi

```
# nvidia-smi -e 0
```

  - Requires reboot to take effect

P2P AND RDMA

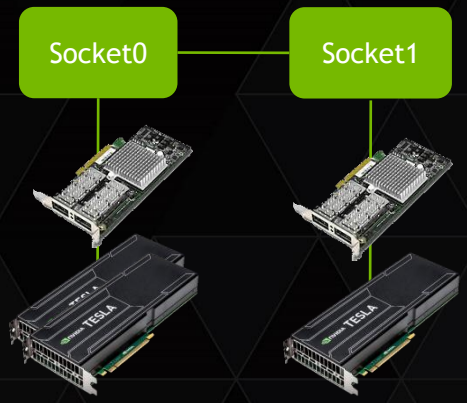
- Shows traversal expectations and potential bandwidth bottleneck via NVSMI
- Cgroups friendly

For NUMA binding

```
GPUDirect Comm Matrix
               GPU0    GPU1    GPU2    mlx5_0  mlx5_1  CPU Affinity
     GPU0       X       PIX     SOC     PHB     SOC     0-9
     GPU1      PIX       X      SOC     PHB     SOC     0-9
     GPU2      SOC     SOC       X      SOC     PHB     10-19
     mlx5_0    PHB     PHB     SOC       X      SOC
     mlx5_1    SOC     SOC     PHB     SOC       X

     Legend:

       X   = Self
       SOC = Path traverses a socket-level link (e.g. QPI)
       PHB = Path traverses a PCIe host bridge
       PXB = Path traverses multiple PCIe internal switches
       PIX = Path traverses a PCIe internal switch
       CPU Affinity = The cores that are most ideal for NUMA
```



HEALTH

- Both APIs and tools to monitor/manage health of a GPU
- ECC error detection
  - Both SBE and DBE
- XID errors
- PCIe throughput and errors
  - Gen/width
  - Errors
  - Throughput
- Violation counters
  - Thermal and power violations of maximum thresholds

PERFORMANCE

- Driver Persistence
- Power and Thermal Management
- Clock Management

DRIVER PERSISTENCE

By default, driver unloads when GPU is idle

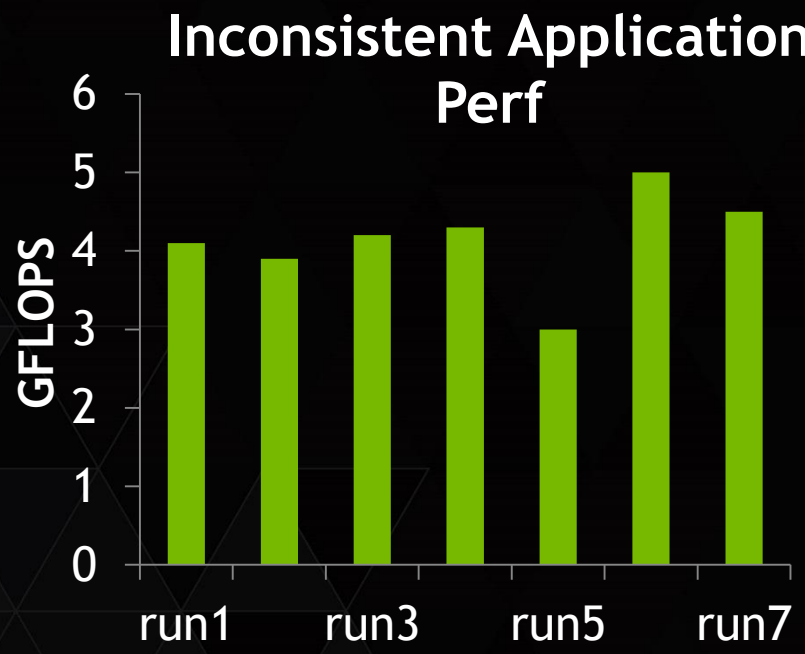
- Driver must re-load when job starts, slowing startup
- If ECC is on, memory is cleared between jobs

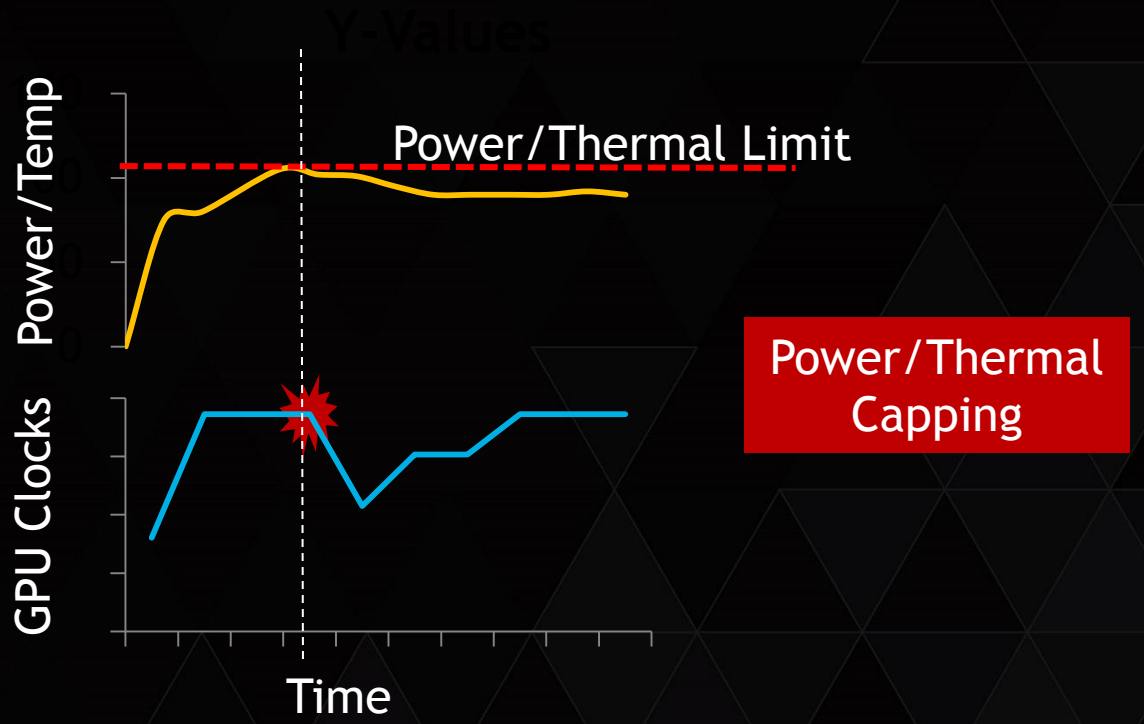
Persistence keeps driver loaded when GPUs idle:

```
# nvidia-smi –i <device#> –pm 1
```

- Faster job startup time

POWER AND THERMAL DATA





Clocks lowered as a preventive measure

POWER AND THERMAL DATA

List Temperature Margins

Query Power Cap Settings

Set Power Cap

nvidia-smi -q -d temperature

```
Temperature
Current Temp            : 90 C
GPU Slowdown Temp       : 92 C
GPU Shutdown Temp       : 97 C
```

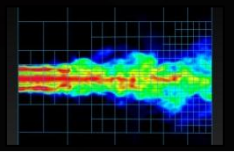
nvidia-smi -q -d power

```
Power Readings
Power Limit             : 95   W
Default Power Limit     : 100  W
Enforced Power Limit    : 95   W
Min Power Limit         : 70   W
Max Power Limit         : 10 W
```

nvidia-smi --power-limit=150

```
Power limit for GPU
0000:0X:00.0 was set to
150.00W from 95.00W
```

CLOCK MANAGEMENT



List Supported Clocks → List Current Clocks → Set Application clocks → Launch CUDA Application → Reset Application Clocks

nvidia-smi –q –d supported_clocks

```
Example Supported Clocks
Memory      : 3004 MHz
  Graphics  : 875 MHz
  Graphics  : 810 MHz
  Graphics  : 745 MHz
  Graphics  : 666 MHz
Memory      : 324 MHz
  Graphics  : 324 MHz
```

nvidia-smi –q –d clocks

```
Current Clocks
Clocks
  Graphics    : 324 MHz
  SM          : 324 MHz
  Memory      : 324 MHz
Applications Clocks
  Graphics    : 745 MHz
  Memory      : 3004 MHz
Default Applications Clocks
  Graphics    : 745 MHz
  Memory      : 3004 MHz
```

nvidia-smi –ac 3004,810

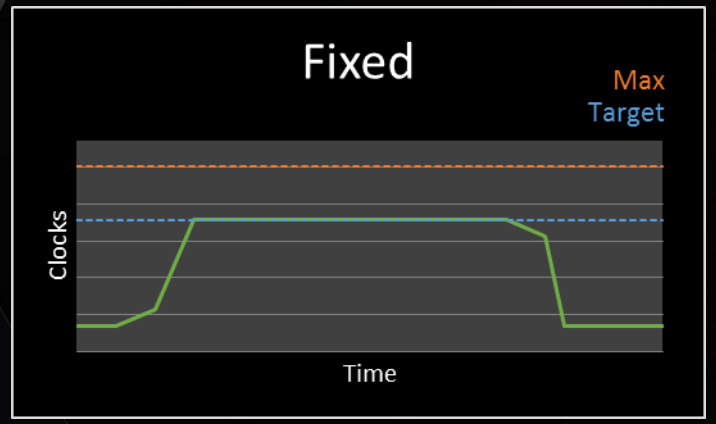
```
Applications Clocks
  Graphics    : 810 MHz
  Memory      : 3004 MHz
```

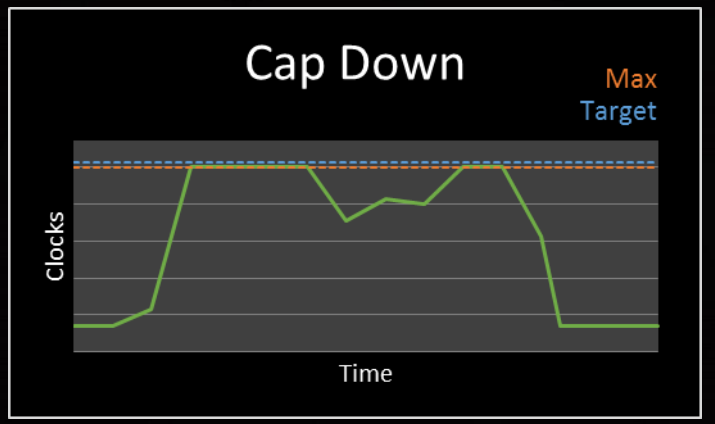
```
Clocks
  Graphics    : 810 MHz
  SM          : 810 MHz
  Memory      : 3004 MHz
```

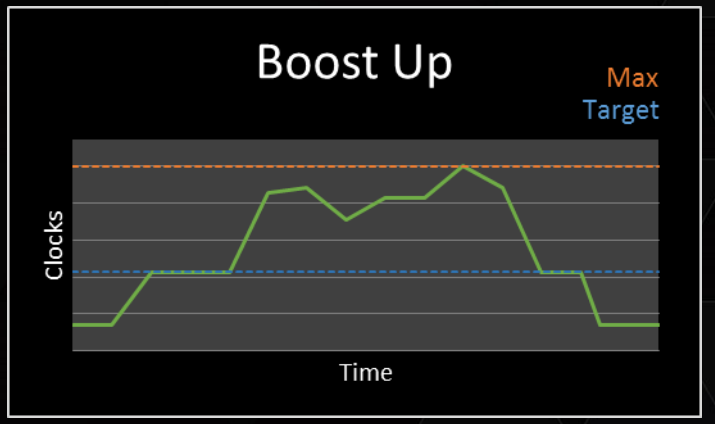
nvidia-smi –rac

```
Applications Clocks
  Graphics    : 745 MHz
  Memory      : 3004 MHz
```

CLOCK BEHAVIOR (K80)







- Fixed Clocks best for consistent perf
- Autoboost (boost up) generally best for max perf

MONITORING & ANALYSIS

- Events
- Samples
- Background Monitoring

HIGH FREQUENCY MONITORING

Events	Samples	Counters
• Clock Changes • XID/ECC Errors	• Power Draw • FB/GPU Utilization	• Power Caps • Thermal Caps

- Provide higher quality data for perf limiters, error events and sensors. Includes xids, power, clocks, utilization and throttle events

HIGH FREQUENCY MONITORING

- nvidia-smi stats
- Visualize monitored data using 3rd party custom UI

Timeline

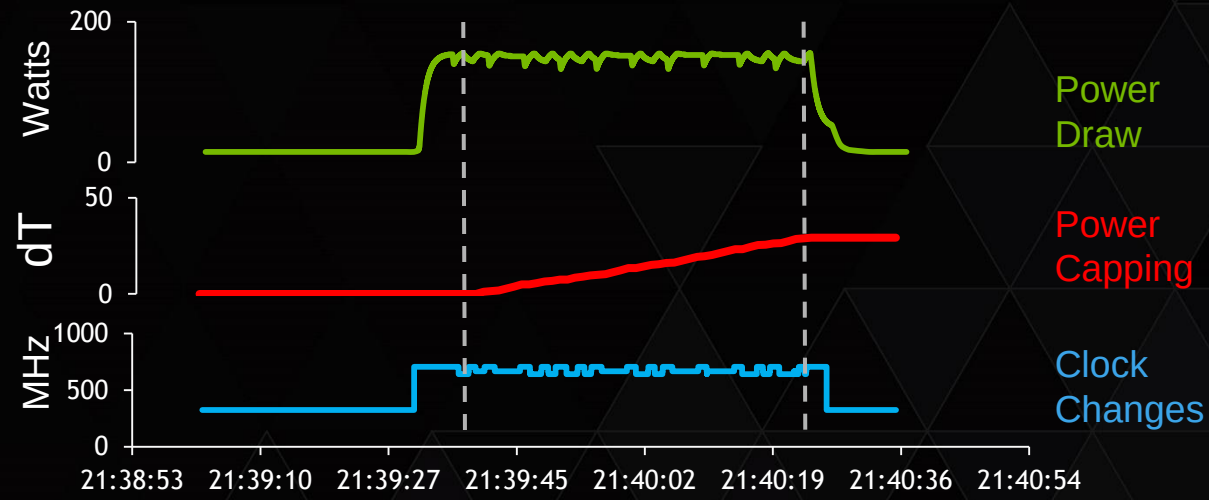
```
procClk   , 1395544840748857,   324
memClk    , 1395544840748857,   324
pwrDraw   , 1395544841083867,   20
pwrDraw   , 1395544841251269,   20
gpuUtil   , 1395544840912983,   0
violPwr   , 1395544841708089,   0
procClk   , 1395544841798380,   705
memClk    , 1395544841798380,   2600
pwrDraw   , 1395544841843620,   133
xid       , 1395544841918978,   31
pwrDraw   , 1395544841948860,   250
violPwr   , 1395544842708054,   345
```

Clocks Idle

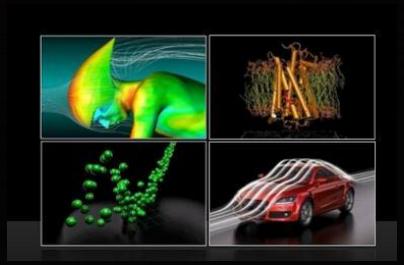
Clocks boost

XID error

Power cap



BRIEF FORMAT



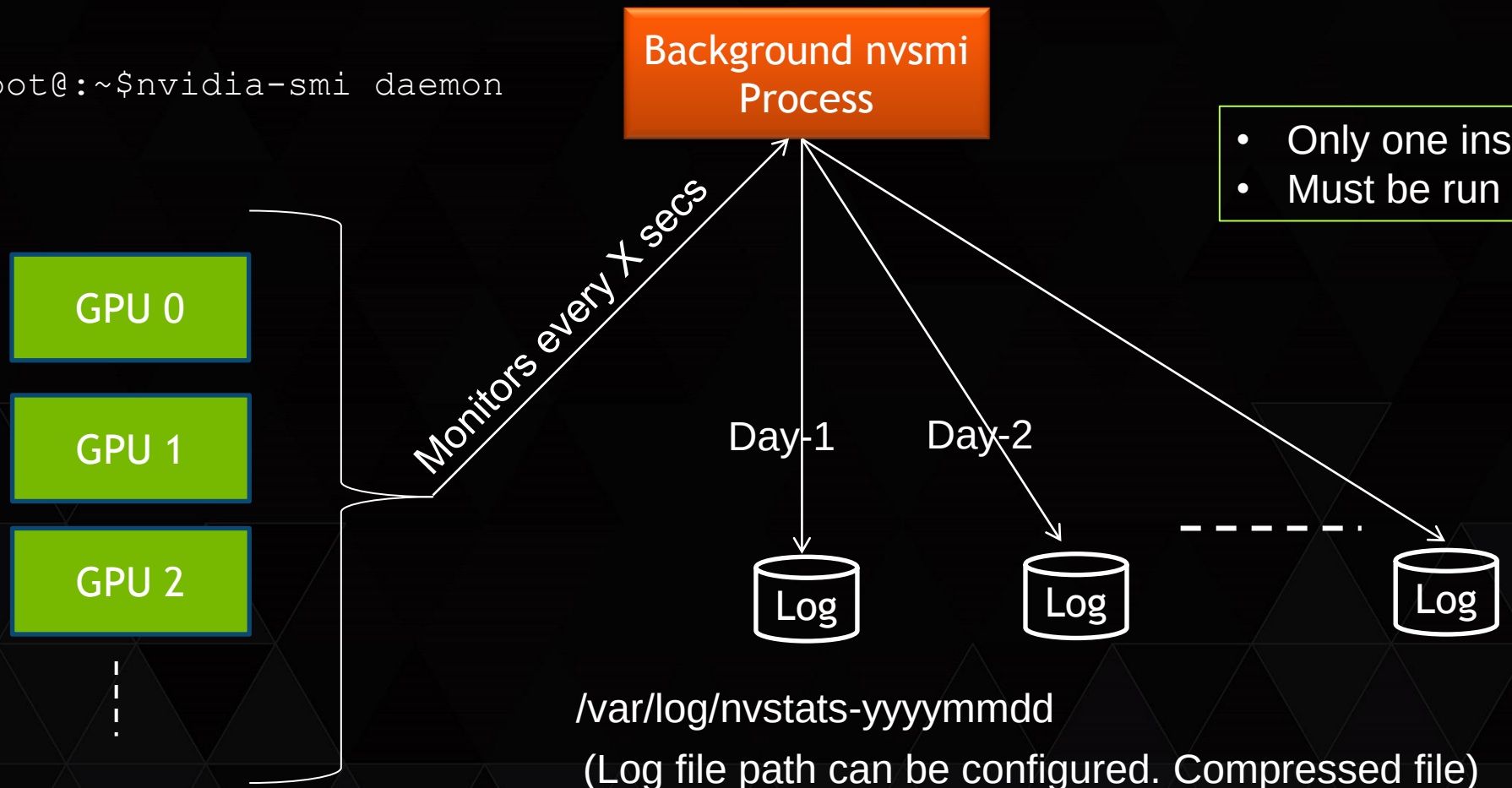
CUDA APP
Power Limit = 160 W
Slowdown Temp = 90 C

- Scrolling single-line interface
- Metrics/Devices to be displayed can be configured

```
nvidia-smi dmon -i <device#>
#Date       Time        gpu    pwr  temp    sm   mem   enc   dec   mclk   pclk   pviol  tviol     fb   bar1
#YYYYMMDD   HH:MM:SS    Idx      W     C     %     %     %     %    MHz    MHz       %   bool     MB     MB
 20150127   11:28:36      2    155    59    99     5     0     0   2600    705       0      0    273      4
 20150127   11:28:37      2    156    61    99     4     0     0   2600    705       0      0    273      4
 20150127   11:28:38      2    160    67    99     5     0     0   2600    705       0      0    273      4
 20150127   11:28:39      2    160    74    99     5     0     0   2600    705       0      0    273      4
 20150127   11:28:40      2    151    79    99     5     0     0   2600    666      56      0    273      4
 20150127   11:28:41      2    153    79    99     5     0     0   2600    666     100      0    273      4
 20150127   11:28:42      2    152    84    99     5     0     0   2600    666     100      0    273      4
 20150127   11:28:43      2    155    88    99     4     0     0   2600    666     100      0    273      4
 20150127   11:28:44      2    152    90    99     4     0     0   2600    666     100      0    273      4
 20150127   11:28:45      2    152    90    99     5     0     0   2600    666     100      0    273      4
 20150127   11:28:46      2    152    89    99     5     0     0   2600    614     100      1    273      4
 20150127   11:28:47      2    151    88    99     4     0     0   2600    614     100      1    273      4
 20150127   11:28:48      2    152    89    99     4     0     0   2600    614     100      1    273      4
```

BACKGROUND MONITORING

```
root@:~$nvidia-smi daemon
```



- Only one instance allowed
- Must be run as a root

/var/log/nvstats-yyyymmdd

(Log file path can be configured. Compressed file)

PLAYBACK/EXTRACT LOGS

- Extract/Replay the complete or parts of log file generated by the daemon
- Useful to isolate GPU problems happened in the past

```
nvidia-smi replay -f <replay file> -b 9:00:00 -e 9:00:05
#Date      Time        gpu   pwr  temp    sm   mem   enc   dec   mclk   pclk  sbecc  dbecc    fb  bar1
#YYYYMMDD  HH:MM:SS    Idx     W     C     %     %     %     %    MHz    MHz   Errs   Errs    MB    MB
20150127   09:00:00      1   120    59    99     5     0     0   2600    705      0      0   124     4
20150127   09:00:01      1   120    61    99     4     0     0   2600    705      0      0   124     4
20150127   09:00:02      1   121    67    99     5     0     0   2600    705      1      0   124     4
20150127   09:00:03      1   123    74    99     5     0     0   2600    705      3      0   124     4
20150127   09:00:04      1   151    79    99     5     0     0   2600    705      5      0   124     4
20150127   09:00:05      1   153    79    99     5     0     0   2600    705      5      0   124     4
```

LOOKING AHEAD

- NVIDIA Diagnostic Tool Suite
- Cluster Management APIs

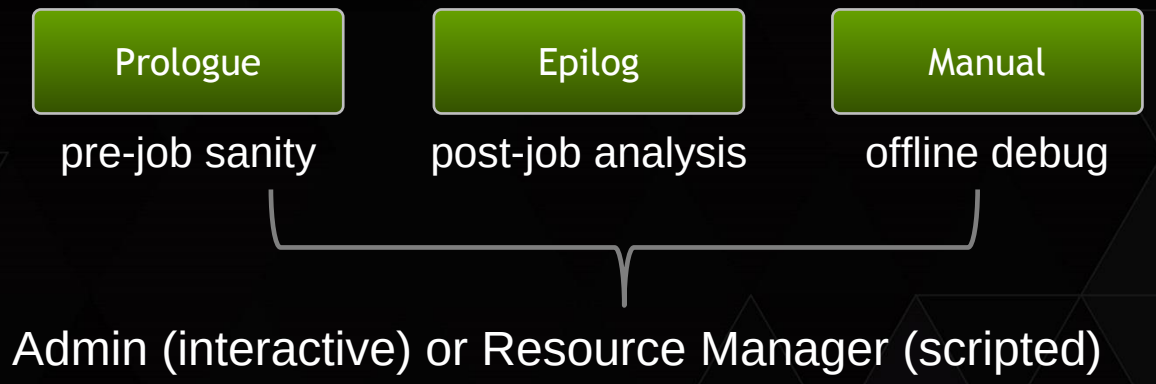
NVIDIA DIAGNOSTIC TOOL SUITE

User runnable, user actionable health and diagnostic tool

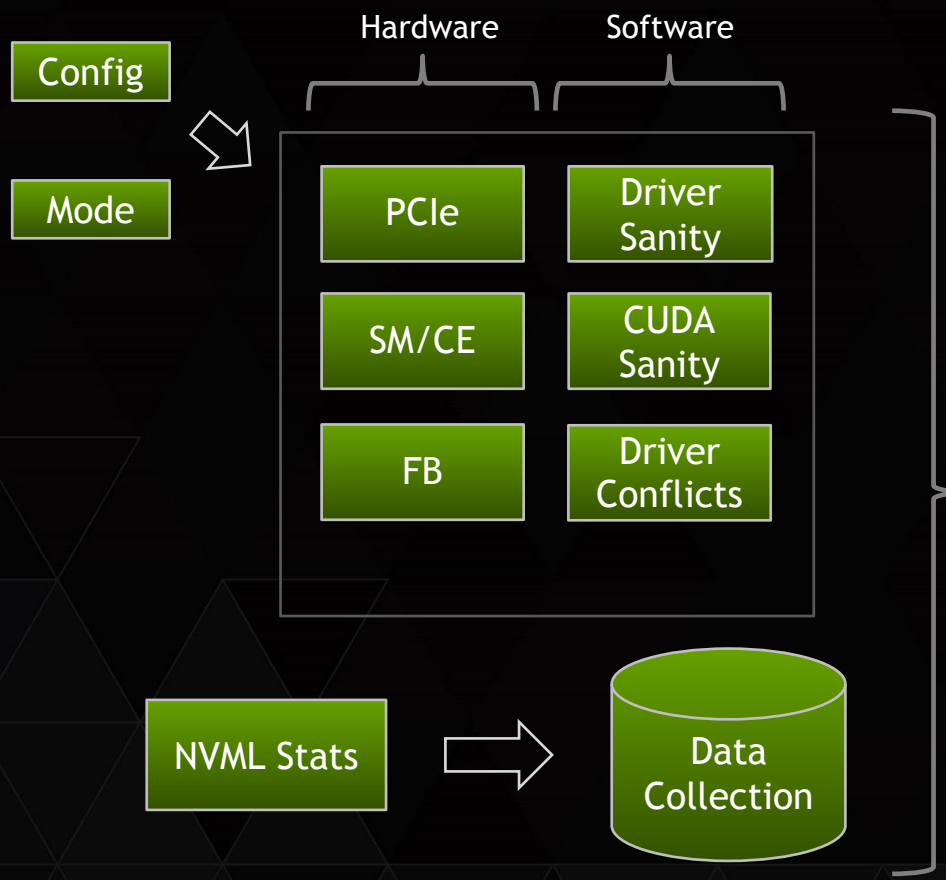
SW, HW, perf and system integration coverage

Command line, pass/fail, configurable

Goal is to consolidate key needs around one tool

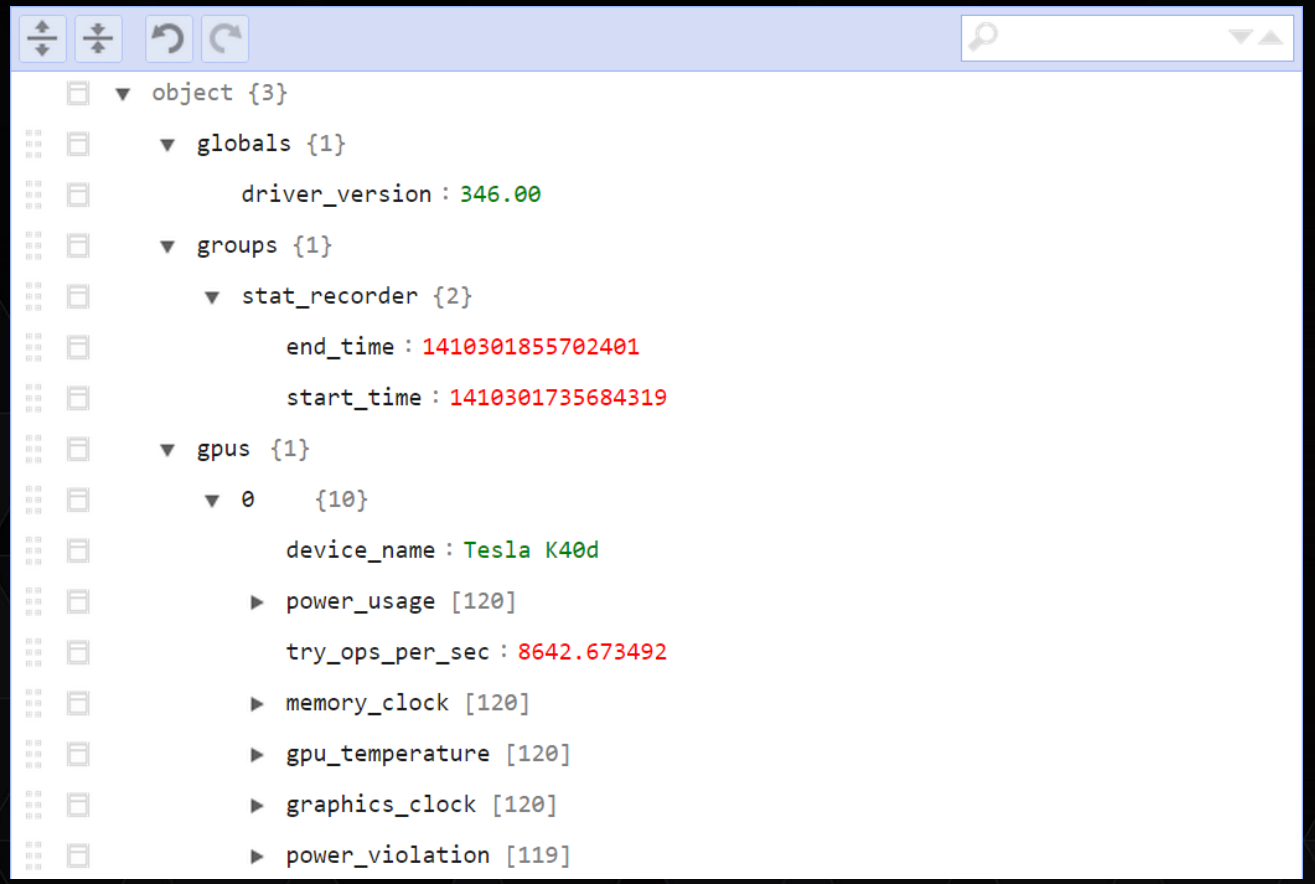


NVIDIA DIAGNOSTIC TOOL SUITE



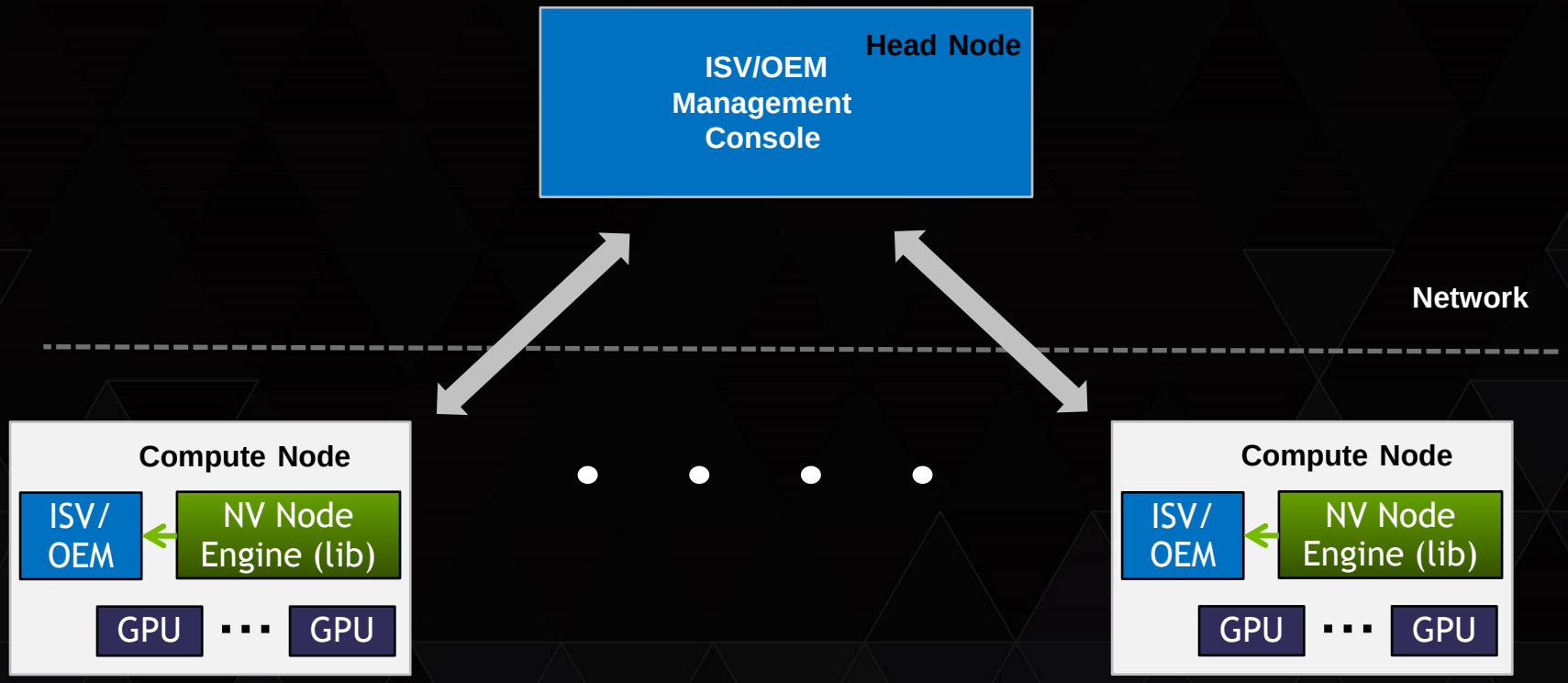
- Extensible diagnostic tool
  - Healthmon will be deprecated
- Determine if a system is *ready* for a job

NVIDIA DIAGNOSTIC TOOL SUITE

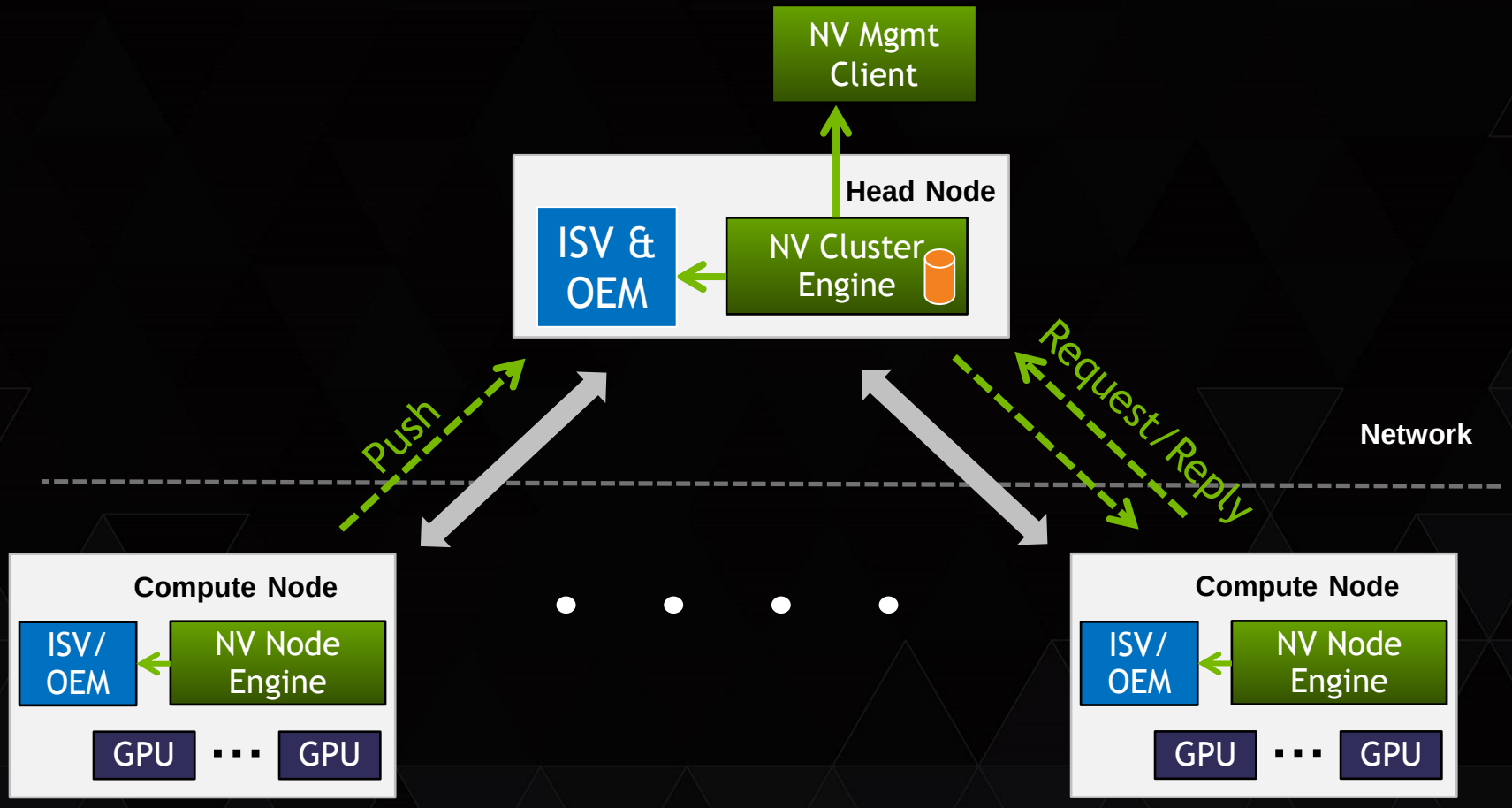


- JSON format
- Binary and text logging options
- Metrics vary by plugin
- Various existing tools to parse, analyze and display data

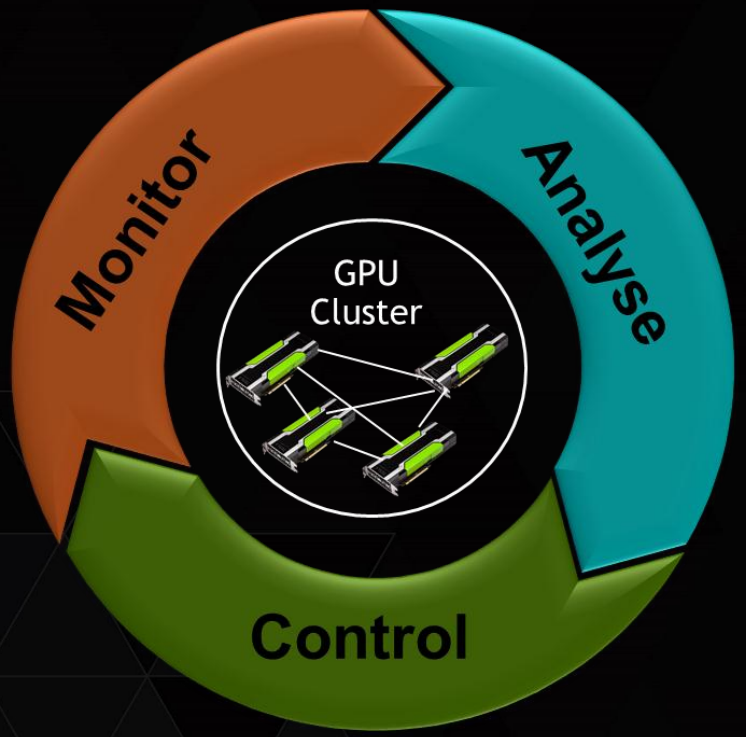
NVIDIA CLUSTER MANAGEMENT



NVIDIA CLUSTER MANAGEMENT



NVIDIA CLUSTER MANAGEMENT



- Stateful
- Proactive Monitoring with Actionable Insights
- Comprehensive Health Diagnostics
- Policy Management
- Configuration Management

NVIDIA REGISTERED DEVELOPER PROGRAMS

- Everything you need to develop with NVIDIA products
- Membership is your first step in establishing a working relationship with NVIDIA Engineering
  - Exclusive access to pre-releases
  - Submit bugs and features requests
  - Stay informed about latest releases and training opportunities
  - Access to exclusive downloads
  - Exclusive activities and special offers
  - Interact with other developers in the NVIDIA Developer Forums

REGISTER FOR FREE AT: developer.nvidia.com

S5894 - Hangout: GPU Cluster Management & Monitoring
Thursday, 03/19, 5pm – 6pm, Location: Pod A

http://docs.nvidia.com/deploy/index.html

contact: cudatools@nvidia.com

THANK YOU

JOIN THE CONVERSATION
#GTC15

APPENDIX

SUPPORTED PLATFORMS/PRODUCTS

Supported platforms:

- Windows (64-bits) / Linux (32-bit and 64-bit)

Supported products:

- Full Support
  - All Tesla products, starting with the Fermi architecture
  - All Quadro products, starting with the Fermi architecture
  - All GRID products, starting with the Kepler architecture
  - Selected GeForce Titan products
- Limited Support
  - All Geforce products, starting with the Fermi architecture

CURRENT TESLA GPUS

GPUs	Single Precision Peak *(SGEMM)*	Double Precision Peak *(DGEMM)*	Memory Size	Memory Bandwidth (ECC off)	PCIe Gen	System Solution
K80	*5.6 TF*	*1.8 TF*	2 x 12GB	480 GB/s	Gen3	Server
K40	4.29 TF *(3.22TF)*	1.43 TF *(1.33 TF)*	12 GB	288 GB/s	Gen 3	Server + Workstation
K20X	3.95 TF *(2.90 TF)*	1.32 TF *(1.22 TF)*	6 GB	250 GB/s	Gen 2	Server only
K20	3.52 TF *(2.61 TF)*	1.17 TF *(1.10 TF)*	5 GB	208 GB/s	Gen 2	Server + Workstation
K10	4.58 TF	0.19 TF	8 GB	320 GB/s	Gen 3	Server only

AUTO BOOST

- User-specified settings for automated clocking changes.
- Persistence Mode
- nvidia-smi --auto-boost-default=0/1
- Enabled by default
- Tesla K80

GPU PROCESS ACCOUNTING

- Provides per-process accounting of GPU usage using Linux PID
- Accessible via NVML or nvidia-smi (in comma-separated format)
- Requires driver be continuously loaded (i.e. persistence mode)
- No RM integration yet, use site scripts i.e. prologue/epilogue

Enable accounting mode:

```
$ sudo nvidia-smi –am 1
```

Human-readable accounting output:

```
$ nvidia-smi –q –d ACCOUNTING
```

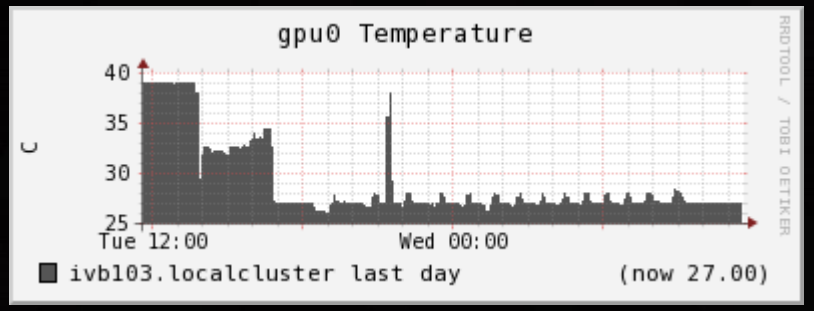
Output comma-separated fields:

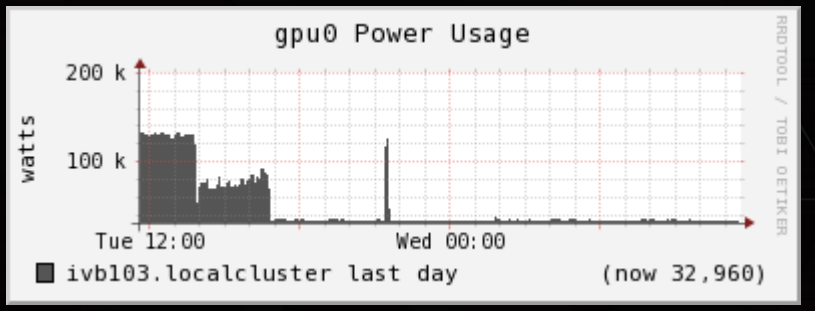
```
$ nvidia-smi --query-accounted-apps=gpu_name,gpu_util –format=csv
```

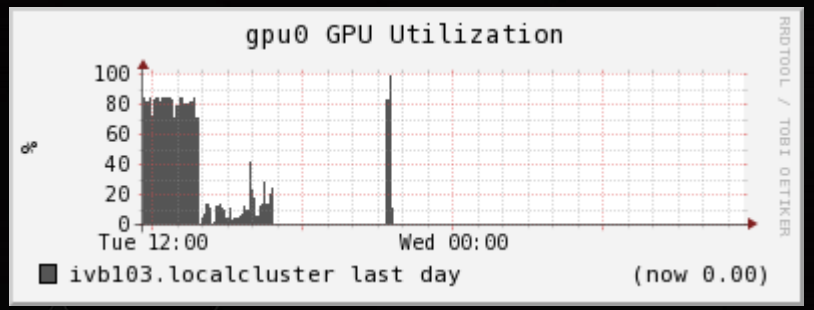
Clear current accounting logs:

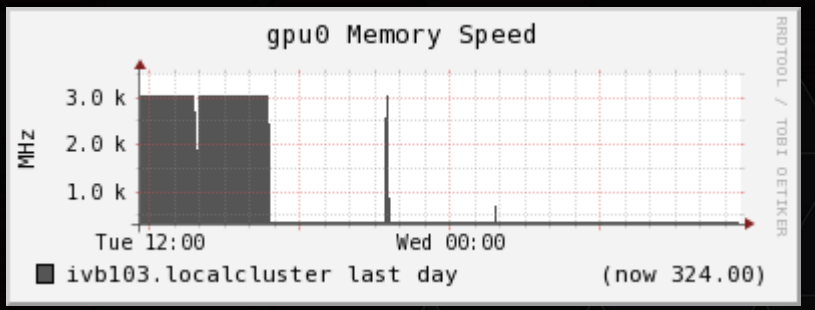
```
$ sudo nvidia-smi -caa
```

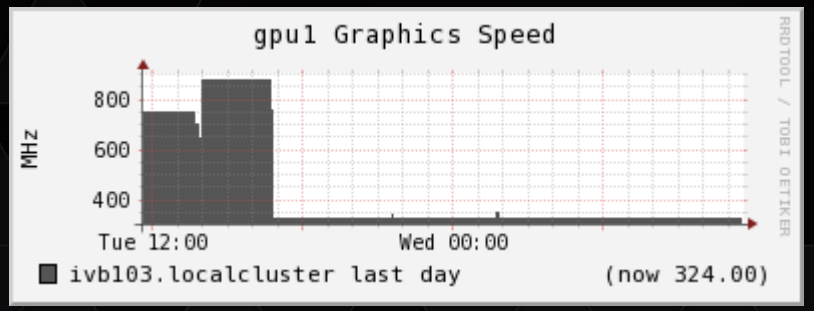
MONITORING SYSTEM WITH NVML SUPPORT











Examples: Ganglia, Nagios, Bright Cluster Manager, Platform HPC

Or write your own plugins using NVML

TURN OFF ECC

- ECC can be turned off – makes more GPU memory available at cost of error correction/detection
  - Configured using NVML or nvidia-smi

```
# nvidia-smi -e 0
```

  - Requires reboot to take effect