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

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

- Health Tools
SOFTWARE RELATIONSHIPS

Key SW Components
- CUDA Toolkit
- NV Driver
- GPU Deployment Kit

CUDA Libraries

Validation Suite
Stats, Dmon, Daemon, Replay

NVML SDK

NVIDIA Kernel Mode Driver

NVSMI

CUDA Runtime

NVML Library

GPU TECHNOLOGY CONFERENCE
MANAGEMENT CAPABILITIES

NVIDIA Management Features

Events
Samples
Process accounting

Identification
PCI Information
Topology
Mode of operation query/control

Logging & Analysis

ECC Errors
XID Errors
Replay/failure counters
Violation Counters

Configuration

Thermal data
Power control/query
Clock control and performance limits
GPU utilization

Performance

Health
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/

```python
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

<table>
<thead>
<tr>
<th>GPUDirect Comm Matrix</th>
<th>GPU0</th>
<th>GPU1</th>
<th>GPU2</th>
<th>mlx5_0</th>
<th>mlx5_1</th>
<th>CPU Affinity</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPU0</td>
<td>X</td>
<td>FIX</td>
<td>SOC</td>
<td>PHB</td>
<td>SOC</td>
<td>0-9</td>
</tr>
<tr>
<td>GPU1</td>
<td>FIX</td>
<td>X</td>
<td>SOC</td>
<td>PHB</td>
<td>SOC</td>
<td>0-9</td>
</tr>
<tr>
<td>GPU2</td>
<td>SOC</td>
<td>SOC</td>
<td>X</td>
<td>SOC</td>
<td>PHB</td>
<td>10-19</td>
</tr>
<tr>
<td>mlx5_0</td>
<td>PHB</td>
<td>PHB</td>
<td>SOC</td>
<td>X</td>
<td>SOC</td>
<td></td>
</tr>
<tr>
<td>mlx5_1</td>
<td>SOC</td>
<td>SOC</td>
<td>PHB</td>
<td>SOC</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

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

For NUMA binding

Socket0

Socket1
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:

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

- Faster job startup time
POWER AND THERMAL DATA

Inconsistent Application Perf

GFLOPS

run1 run3 run5 run7

Power/Thermal Limit

Clocks lowered as a preventive measure

Y-Values

Power/Thermal Capping

GPU Clocks

Time
POWER AND THERMAL DATA

**List Temperature Margins**

- **Current Temp**: 90 °C
- **GPU Slowdown Temp**: 92 °C
- **GPU Shutdown Temp**: 97 °C

**Query Power Cap Settings**

- **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

**Set Power Cap**

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

---

- **nvidia-smi -q -d temperature**
- **nvidia-smi -q -d power**
- **nvidia-smi --power-limit=150**
CLOCK MANAGEMENT

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

**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

**Launch CUDA Application**
- nvidia-smi -ac 3004,810

**Reset Application Clocks**
- nvidia-smi -rac

**Supported Clocks**
- nvidia-smi -q -d supported_clocks

**Current Clocks**
- nvidia-smi -q -d clocks

**Applications Clocks**
- nvidia-smi -ac

**Default Applications Clocks**
- nvidia-smi -rac
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**
  - Clock Changes
  - XID/ECC Errors

- **Samples**
  - Power Draw
  - FB/GPU Utilization

- **Counters**
  - 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
### BRIEF FORMAT

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

```
nvidia-smi dmon -i <device#>
```

<table>
<thead>
<tr>
<th>#Date</th>
<th>Time</th>
<th>gpu</th>
<th>pwr</th>
<th>temp</th>
<th>sm</th>
<th>mem</th>
<th>enc</th>
<th>dec</th>
<th>mclk</th>
<th>pclk</th>
<th>pvio</th>
<th>tvio</th>
<th>fb</th>
<th>barl</th>
</tr>
</thead>
<tbody>
<tr>
<td>YYYYMMDD</td>
<td>HH:MM:SS</td>
<td>Idx</td>
<td>W</td>
<td>C</td>
<td>%</td>
<td>%</td>
<td>%</td>
<td>%</td>
<td>MHz</td>
<td>MHz</td>
<td>%</td>
<td>bool</td>
<td>MB</td>
<td>MB</td>
</tr>
<tr>
<td>20150127</td>
<td>11:28:36</td>
<td>2</td>
<td>155</td>
<td>59</td>
<td>99</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>2600</td>
<td>705</td>
<td>0</td>
<td>0</td>
<td>273</td>
<td>4</td>
</tr>
<tr>
<td>20150127</td>
<td>11:28:37</td>
<td>2</td>
<td>156</td>
<td>61</td>
<td>99</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>2600</td>
<td>705</td>
<td>0</td>
<td>0</td>
<td>273</td>
<td>4</td>
</tr>
<tr>
<td>20150127</td>
<td>11:28:38</td>
<td>2</td>
<td>160</td>
<td>67</td>
<td>99</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>2600</td>
<td>705</td>
<td>0</td>
<td>0</td>
<td>273</td>
<td>4</td>
</tr>
<tr>
<td>20150127</td>
<td>11:28:39</td>
<td>2</td>
<td>160</td>
<td>74</td>
<td>99</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>2600</td>
<td>705</td>
<td>0</td>
<td>0</td>
<td>273</td>
<td>4</td>
</tr>
<tr>
<td>20150127</td>
<td>11:28:40</td>
<td>2</td>
<td>151</td>
<td>79</td>
<td>99</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>2600</td>
<td>666</td>
<td>56</td>
<td>100</td>
<td>0</td>
<td>273</td>
</tr>
<tr>
<td>20150127</td>
<td>11:28:41</td>
<td>2</td>
<td>153</td>
<td>79</td>
<td>99</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>2600</td>
<td>666</td>
<td>100</td>
<td>0</td>
<td>273</td>
<td>4</td>
</tr>
<tr>
<td>20150127</td>
<td>11:28:42</td>
<td>2</td>
<td>152</td>
<td>84</td>
<td>99</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>2600</td>
<td>666</td>
<td>100</td>
<td>0</td>
<td>273</td>
<td>4</td>
</tr>
<tr>
<td>20150127</td>
<td>11:28:43</td>
<td>2</td>
<td>155</td>
<td>88</td>
<td>99</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>2600</td>
<td>666</td>
<td>100</td>
<td>0</td>
<td>273</td>
<td>4</td>
</tr>
<tr>
<td>20150127</td>
<td>11:28:44</td>
<td>2</td>
<td>152</td>
<td>90</td>
<td>99</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>2600</td>
<td>666</td>
<td>100</td>
<td>0</td>
<td>273</td>
<td>4</td>
</tr>
<tr>
<td>20150127</td>
<td>11:28:45</td>
<td>2</td>
<td>152</td>
<td>89</td>
<td>99</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>2600</td>
<td>614</td>
<td>100</td>
<td>1</td>
<td>273</td>
<td>4</td>
</tr>
<tr>
<td>20150127</td>
<td>11:28:46</td>
<td>2</td>
<td>152</td>
<td>88</td>
<td>99</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>2600</td>
<td>614</td>
<td>100</td>
<td>1</td>
<td>273</td>
<td>4</td>
</tr>
<tr>
<td>20150127</td>
<td>11:28:47</td>
<td>2</td>
<td>151</td>
<td>88</td>
<td>99</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>2600</td>
<td>614</td>
<td>100</td>
<td>1</td>
<td>273</td>
<td>4</td>
</tr>
<tr>
<td>20150127</td>
<td>11:28:48</td>
<td>2</td>
<td>152</td>
<td>89</td>
<td>99</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>2600</td>
<td>614</td>
<td>100</td>
<td>1</td>
<td>273</td>
<td>4</td>
</tr>
</tbody>
</table>

CUDA APP
Power Limit = 160 W
Slowdown Temp = 90°C
BACKGROUND MONITORING

root@:~$nvidia-smi daemon

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

/var/log/nvstats-yyyyymmdd
(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

```bash
code
nvidia-smi replay -f <replay file> -b 9:00:00 -e 9:00:05
```

<table>
<thead>
<tr>
<th>#Date</th>
<th>Time</th>
<th>gpu</th>
<th>pwr</th>
<th>temp</th>
<th>sm</th>
<th>mem</th>
<th>enc</th>
<th>dec</th>
<th>mclk</th>
<th>pclk</th>
<th>sbecc</th>
<th>dbecc</th>
<th>fb</th>
<th>bar1</th>
</tr>
</thead>
<tbody>
<tr>
<td>#YYYYMMDD HH:MM:SS</td>
<td>Idx</td>
<td>W</td>
<td>C</td>
<td>%</td>
<td>%</td>
<td>%</td>
<td>MHz</td>
<td>MHz</td>
<td>Errs</td>
<td>Errs</td>
<td>MB</td>
<td>MB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20150127 09:00:00</td>
<td>1</td>
<td>120</td>
<td>59</td>
<td>99</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>2600</td>
<td>705</td>
<td>0</td>
<td>0</td>
<td>124</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>20150127 09:00:01</td>
<td>1</td>
<td>120</td>
<td>61</td>
<td>99</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>2600</td>
<td>705</td>
<td>0</td>
<td>0</td>
<td>124</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>20150127 09:00:02</td>
<td>1</td>
<td>121</td>
<td>67</td>
<td>99</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>2600</td>
<td>705</td>
<td>1</td>
<td>0</td>
<td>124</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>20150127 09:00:03</td>
<td>1</td>
<td>123</td>
<td>74</td>
<td>99</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>2600</td>
<td>705</td>
<td>3</td>
<td>0</td>
<td>124</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>20150127 09:00:04</td>
<td>1</td>
<td>151</td>
<td>79</td>
<td>99</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>2600</td>
<td>705</td>
<td>5</td>
<td>0</td>
<td>124</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>20150127 09:00:05</td>
<td>1</td>
<td>153</td>
<td>79</td>
<td>99</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>2600</td>
<td>705</td>
<td>5</td>
<td>0</td>
<td>124</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>
```
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

Prologue
pre-job sanity

Epilog
post-job analysis

Manual
offline debug

Admin (interactive) or Resource Manager (scripted)
NVIDIA DIAGNOSTIC TOOL SUITE

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

Config

Mode

Hardware

- PCIe
- SM/CE
- FB

Software

- Driver Sanity
- CUDA Sanity
- Driver Conflicts

Analysis

Data Collection

logs

stdout

NVML Stats
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

- 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
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

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