NVIDIA GPU Debug Guidelines

Application Note

DA-10231-001_vR450 | November 2020

Table of Contents

Chapter 1. Overview

This document provides a process flow and associated details on how to start debugging general issues on GPU servers. It is intended to cover the most common issues one may see in the operation of GPUs in the data center but is not meant to be comprehensive. The information below is a summary of several different documents and best practices for debugging GPU system issues.

This debug process is intended to be generic and may not align with your system vendors specific triage guidelines. Please engage with your system vendor earlier, rather than later, to ensure your system is restored to full health as fast as possible. However, by using this process, infrastructure teams should be able to work through the first steps of the debug process to gather as much data as possible or avoid the need to submit a help request entirely.

Chapter 2. Initial Incident Report

When managing GPU system incidents, as in all system incidents, having a well-designed process can help so incidents can be diagnosed more rapidly, and systems can be returned to production faster. At the beginning of any incident, whether detected by the system or reported from a user, try to document the following questions

- What was observed about the incident?
- When was the incident observed?
- How was it observed?
- Is this behavior observed on multiple systems or components?
- If so, how many, and how frequent?
- Has anything changed with the system, driver, or application behavior recently?

Collecting information regarding these questions to kick-off the debug process is important as it provides the best understanding of the problem and by recording this information can be correlated to other events to gain a better understanding of overall system behavior and health.

Chapter 3. GPU Node Triage

There are several ways that system support teams are notified about potential issues on GPU based systems. These can come from error reports, monitoring system events, and diagnostic metrics. While these events can impact the operation of a GPU system, not all require intervention of the system vendor for resolution. In addition, there are common tools which can be used to gather data after a system issue which are useful to both the local system support team and the vendor for node triage.

Figure 1 to Figure 3 are flowcharts of how node triage should start based on the information provided indicating there may be a node issue.

Figure 1. GPU Triage Flowchart

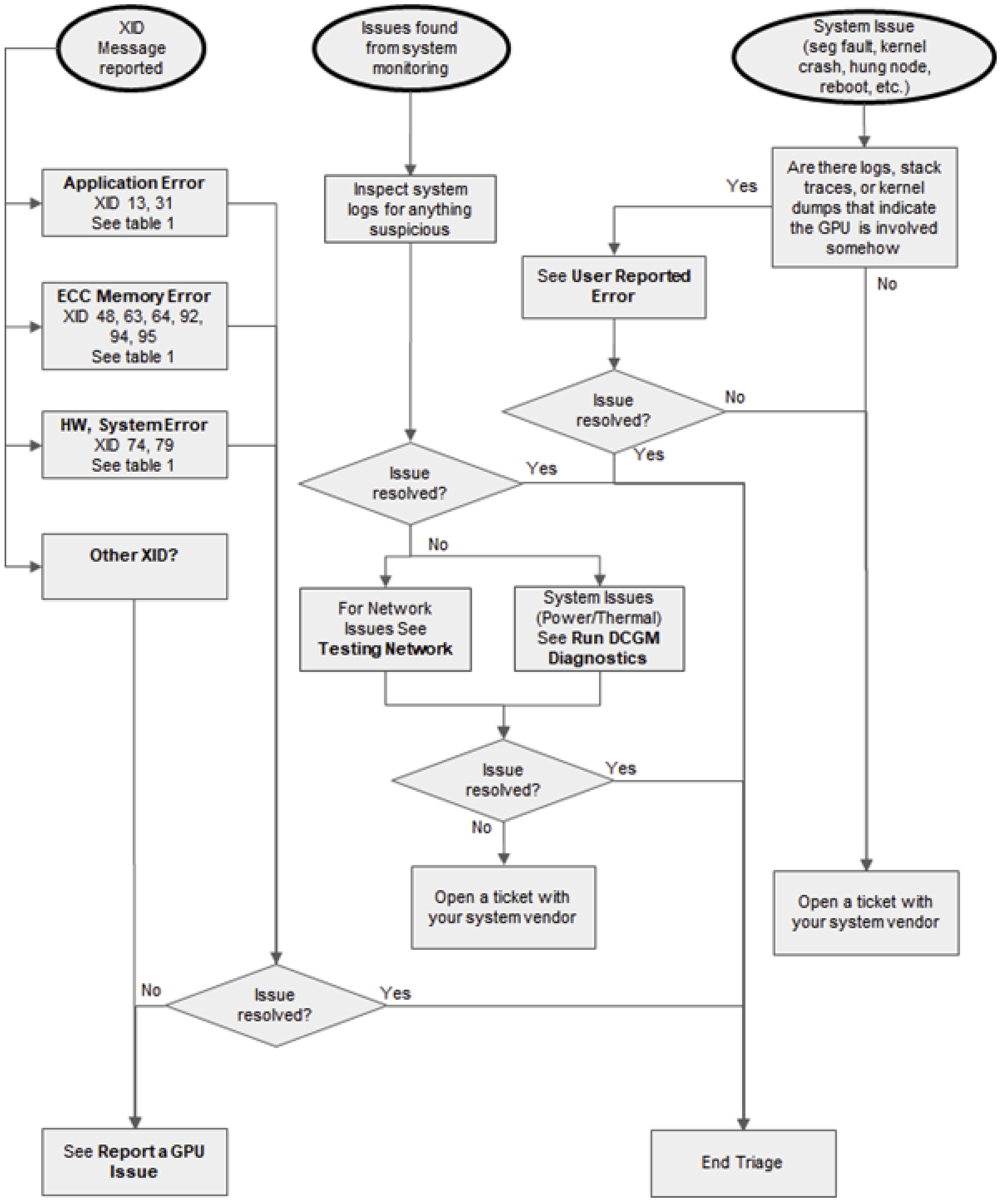

XID Message reported
Issues found from system monitoring
System Issue (seg fault, kernel crash, hung node, reboot, etc.)
Application Error XID 13, 31 See table 1
ECC Memory Error XID 48, 63, 64, 92, 94, 95 See table 1
HW, System Error XID 74, 79 See table 1
Other XID?
Inspect system logs for anything suspicious
Are there logs, stack traces, or kernel dumps that indicate the GPU is involved somehow
Yes
No
See User Reported Error
Issue resolved?
No
Yes
Issue resolved?
Yes
No
For Network Issues See Testing Network
System Issues (Power/Thermal) See Run DCGM Diagnostics
Issue resolved?
Yes
No
Open a ticket with your system vendor
Open a ticket with your system vendor
No
Issue resolved?
Yes
See Report a GPU Issue
End Triage

Figure 2. GPU Triage Flowchart (User Reported Error)

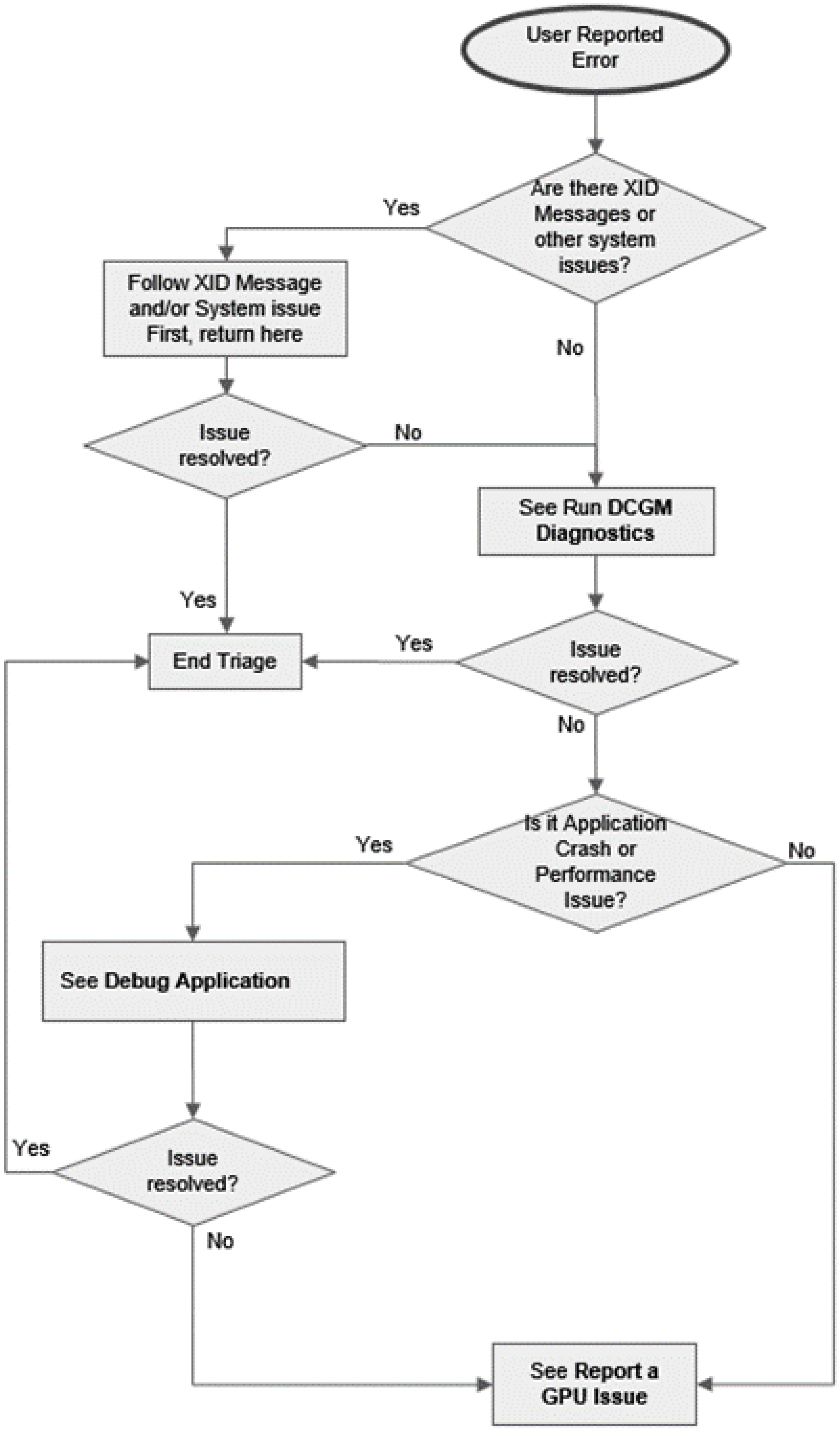
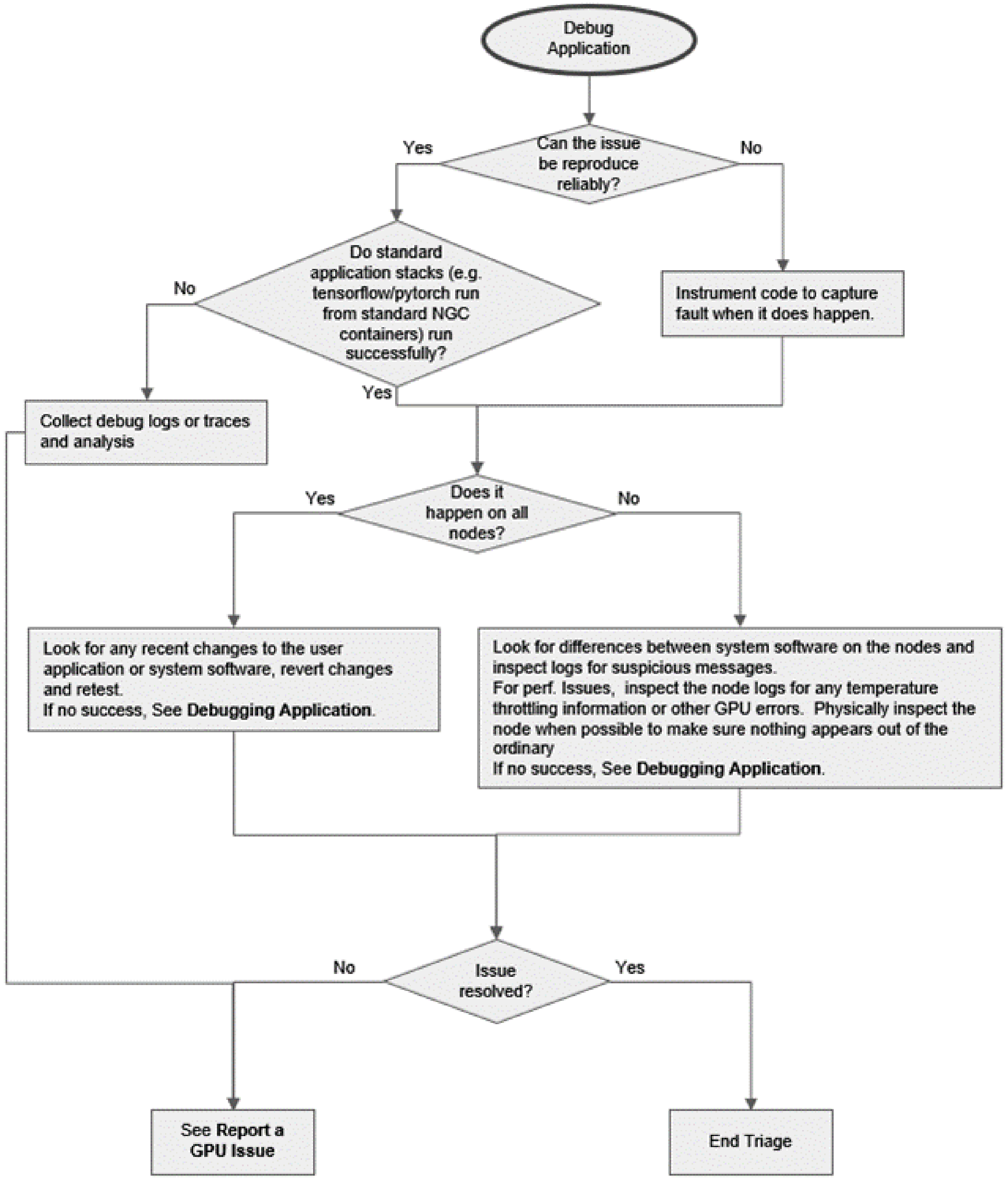


Figure 3. GPU Triage Flowchart (Debug Application)


Debug Application
Can the issue be reproduce reliably?
Yes
No
Do standard application stacks (e.g. tensorflow/pytorch run from standard NGC containers) run successfully?
Instrument code to capture fault when it does happen.
No
Yes
Collect debug logs or traces and analysis
Does it happen on all nodes?
Yes
No
Look for any recent changes to the user application or system software, revert changes and retest. If no success, See Debugging Application.
Look for differences between system software on the nodes and inspect logs for suspicious messages. For perf. Issues, inspect the node logs for any temperature throttling information or other GPU errors. Physically inspect the node when possible to make sure nothing appears out of the ordinary If no success, See Debugging Application.
Issue resolved?
No
Yes
See Report a GPU Issue
End Triage

3.1. Reporting a GPU Issue

When gathering data for your system vendor, you should include the following:

1. Basic system configuration such as OS and driver info
2. A clear description of the issue, including any key log messages describing the problem.
3. List of debug steps taken.
4. A full listing of the log used for the key messages above
5. Output of *nvidia-bug-report.sh*
6. Fabric manager log files for HGX systems
7. DCGM Diagnostics logs
8. If associated with a user application, any details you can provide to the nature of the application (ISV code, framework, version numbers, etc) and links to source code (if possible).
9. Submit a ticket to your system vendor

3.2. Understanding XID Messages

The XID error document gives a listing of XID errors and potential causes. At the end of the document it also provides more information on some common XID errors and recommended actions. For details see https://docs.nvidia.com/deploy/xid-errors/index.html.

The most common XID messages in Data Center deployments and recommended steps are listed in Table 1.

Table 1. Common XID Messages and Recommended Steps

XID	Description	Action
13	Graphics Engine Exception	Run DCGM and Field diagnostics to confirm if the issue is related to hardware. If not, debug the user application using guidance fromhttps://docs.nvidia.com/deploy/xid-errors/index.html. If the latter, see Reporting a GPU Issue.
31	GPU memory page fault	Debug the user application unless the issue is new and there have been no changes to the application but there has been changes to GPU driver or other GPU system software. If the latter, see Reporting a GPU Issue.
48	Double Bit ECC Error	If Xid 48 is followed by Xid 63 or 64: Drain/cordon the node, wait for all work to complete, reboot the node.

		If Xid 48 is not followed by Xid 63 or 64: see Running Field Diagnosticto collect additional debug information. See below for guidelines on when to RMA GPUs based on excessive errors.
63	**Legacy GPU:** ECC page retirement recording event	If associated with XID 48, drain/cordon the node, wait for all work to complete, and reboot the node. If not, it is from a single bit error and the system can keep running as is until there is a convenient time to reboot it. See below for guidelines on when to RMA GPUs based on excessive errors.
	A100: Row-remapping recording event	If associated with XID 94, the application that encountered the error needs to be restarted. All other applications on the system can keep running as is until there is a convenient time to reboot for row remapping to activate. See below for guidelines on when to RMA GPUs based on row remapping failures.
64	**Legacy GPU:** ECC page retirement recording failure	See above, however the node should be monitored closely. If there is no associated XID 48 error, then these are related to single bit-errors. The node should be rebooted immediately since there is a recording failure. If the errors continue, drain, triage, and see Reporting a GPU Issue. See below for guidelines on when to RMA GPUs based on excessive errors.
	A100: Row-remapping recording failure	The node should be rebooted immediately since there is a recording failure. If the errors continue, drain, triage, and see Reporting a GPU Issue.

		See below for guidelines on when to RMA GPUs based on row remapping failures.
74	NVLink Error	Drain and see Reporting a GPU Issue.
79	GPU has fallen off the bus	Drain and see Reporting a GPU Issue.
92	High single-bit ECC error rate	See Running Field Diagnosticto collect additional debug information. See below for guidelines on when to RMA GPUs based on excessive errors.
94	Contained ECC error occurred (**A100 only**)	The application that encountered the error needs to be restarted. All other applications on the system can keep running as is until there is a convenient time to reboot for row remapping to activate. See below for guidelines on when to RMA GPUs based on row remapping failures
95	Uncontained ECC error occurred (**A100 only**)	If MIG is enabled, drain any work on the other GPU instances, wait for all work to complete, and reset the node. If MIG is disabled, the node should be rebooted immediately since there is an uncorrectable uncontained ECC error. If the errors continue, drain, triage, and see Reporting a GPU Issue. See below for guidelines on when to RMA GPUs based on row remapping failures.

XID messages 48, 63, 64, 92, 94 and 95 are related to GPU memory errors. NVIDIA GPUs prior to A100 support dynamic page retirement. For details on dynamic page retirement, see https://docs.nvidia.com/deploy/dynamic-page-retirement/index.html

NVIDIA A100 GPUs introduce new memory error recovery features that improve resilience and avoid impacting unaffected applications. For details on A100 GPU Memory Error Management, see https://docs.nvidia.com/deploy/a100-gpu-mem-error-mgmt/index.html

In addition, there are SXID messages for issues with the NVSwitch. Depending on the severity (fatal vs non- fatal) and the impacted port, the errors may abort existing CUDA jobs and prevent new CUDA job launches. For details see https://docs.nvidia.com/datacenter/tesla/pdf/fabric-manager-user-guide.pdf

3.3. Running DCGM Diagnostics

DCGM is a system level tool that provides a diagnostic test for production environments to assess node health and cluster readiness. For details see https://docs.nvidia.com/datacenter/dcgm/latest/dcgm-user-guide/dcgm-diagnostics.html

There are many options available for running the test, and a few default configurations that should work on most systems, specified with the `--run (-r)` option. The available test suites are 1 (short), 2 (medium), and 3 (long). For fully stressing the system, use the long (3) option.

```
# dcgmi diag -r 3
```

The long test should take approximately 30 minutes. The other options can be used as pre-flight checks or in the system prolog to validate the node prior to starting jobs.

When DCGM diagnostics finds an issue, attempt to resolve it. Issues with the configuration can be handled with IT commands and DCGM may provide suggestions as to where to start. If the diagnostic tests find an issue with the operation of the GPU or NVSwitch fabric (when present), inspect the node and node configuration for anything out of normal.

3.4. Running Field Diagnostic

Field diagnostic is the authoritative and comprehensive NVIDIA tool for determining the health of GPUs. It is usually required before an RMA can be started. Please contact your system vendor for instructions on when, if, and how you should run this tool.

3.5. Network Testing

Network performance and latency can be tested using the NCCL performance tests (https://github.com/NVIDIA/nccl-tests). In particular, all_reduce_perf test is a great test for establishing network performance between groups of nodes. Pairwise tests of nodes should result in the same performance. When a slow pair of nodes is identified, retest with those nodes and different nodes to isolate the issue. In multi-rail network topologies, the tests can also be isolated to specific rails to determine which network interface is of concern. More information can be found at https://developer.nvidia.com/nccl.

It is advisable to run the network tests prior to putting systems into production and record the performance achieved. This will help understand the performance of your network and provide a baseline for future comparisons. The performance of groups of nodes can vary depending on the system network topology. Coordinate with your network vendor/architect to help understand what performance should be achievable.

3.6. Debugging Applications

Debugging an application typically requires attaching a debugger to the user application and gathering as much data as possible up until the application crashes. The issue causing the application crash could be related to the system software, but debugging the process is the fastest way to understand where the crash is and if it can be resolved from the application side. The data gathered from the debugger can be used by NVIDIA to start debugging system software issues.

CUDA-GDB is the NVIDIA tool for debugging CUDA applications running on Linux and QNX. This enables developers to debug applications without the potential variations introduced by simulation and emulation environments. For details see https://developer.nvidia.com/cuda-gdb and https://docs.nvidia.com/cuda/cuda-gdb/index.html.

If the issue is related to performance, one can use a performance profiler, such as Nsight Systems or Nsight Compute, to understand the performance and bottlenecks in an application. Changes in system software can cause changes in application performance so even if the application did not change, profiling the application is an important step. Profiler data can highlight performance issues in the CPU, storage, or GPUs which help understand where the performance issue may be originating.

Chapter 4. Best Practices

Debugging an application typically requires attaching a debugger to the user application and gathering as much data as possible up until the application crashes. The issue causing the application crash could be related to the system software, but debugging the process is the fastest way to understand where the crash is and if it can be resolved from the application side. The data gathered from the debugger can be used by NVIDIA to start debugging system software issues.

CUDA-GDB is the NVIDIA tool for debugging CUDA applications running on Linux and QNX. This enables developers to debug applications without the potential variations introduced by simulation and emulation environments. For details see https://developer.nvidia.com/cuda-gdb and https://docs.nvidia.com/cuda/cuda-gdb/index.html.

If the issue is related to performance, one can use a performance profiler, such as Nsight Systems or Nsight Compute, to understand the performance and bottlenecks in an application. Changes in system software can cause changes in application performance so even if the application did not change, profiling the application is an important step. Profiler data can highlight performance issues in the CPU, storage, or GPUs which help understand where the performance issue may be originating.

4.1. Collecting Node Metrics

Node performance and operation can be deduced from system metrics. Gathering time series data of system metrics can allow administrators to detect when nodes are starting to operate incorrectly or be used for proactive system maintenance.

Metrics can be gathered both in-band with tools like the DCGM Prometheus plugin and out-of-band with IPMI. For details on DCGM Prometheus plugin, see https://docs.nvidia.com/datacenter/dcgm/latest/dcgm-user-guide/integrating-with-dcgm.html

A containerized version of DCGM is provided for Kubernetes deployments. For details see:

https://developer.nvidia.com/blog/monitoring-gpus-in-kubernetes-with-dcgm/

Key metrics that should be collected on GPU based systems include:

- Power draw of each power supply
- Power draw at the PDU
- PDU Power factor

- CPU Power, and CPU Temperature
- GPU Power, GPU Temperature, GPU Memory Temperature and GPU clocks
- NVLink metrics (A100 only)
- Node Fan Speed
- Inlet Temperature

Other metrics may be useful depending on specifics of the datacenter.

The data can be stored in a time-series or other NoSQL database and be recalled as needed. In addition, alerts can be created to detect conditions that may negatively affect system and application performance. These can include:

- A rise in inlet temperature could be an indicator of an issue with datacenter cooling or other debris blocking airflow
- High GPU and GPU Memory Temperature will lead to GPU throttling and poor performance

4.2. Catching Errors Before They Occur

The goal of this document is to help administrators find the most common system issues and provide understanding or a quick path to resolution. However, it is best to catch system issues before they affect user jobs. The best time to do this is at the beginning of jobs via init-containers or prolog scripts or at the end of jobs with epilog scripts. Here are some suggestions on tests to include to help catch system errors before they affect user jobs.

GPU Specific Checks:

- Confirm key pieces of software are loaded
  - GPU driver
  - nv-peer-mem (for GPUDirect RDMA, https://developer.nvidia.com/gpudirect)
  - Gdrdrv (for GDRCopy, https://github.com/NVIDIA/gdrcopy)
  - Fabric manager for HGX systems
- Confirm specific GPU features of the nodes are correct (GPU Count, PCIe link speed, VBIOS, etc)
- Confirm driver persistence setting is correct. Driver persistence should be controlled through Persistence Daemon. (https://docs.nvidia.com/deploy/driver-persistence/index.html)
- For NVSwitch based Systems
  - Confirm the fabric manager is running
  - Confirm the NVLink fabric topology is correct
- Check for any recent XID errors
- Run `dcgmi diag -r 2` for a quick check of system software and GPU functionality

Other non-GPU checks to include:

- Check all network links are active
  - This could be a simple check to see it is active, or a more robust check that validates expected link speed, firmware, PCIe link width, etc.
- Check all expected filesystem mounts are present
- Check to ensure nodes are set up correctly by running some CUDA workloads. For details see https://docs.nvidia.com/datacenter/tesla/hgx-software-guide/index.html and https://docs.nvidia.com/datacenter/tesla/tesla-installation-notes/index.html

ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, "MATERIALS") ARE BEING PROVIDED "AS IS." NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE.

Information furnished is believed to be accurate and reliable. However, NVIDIA Corporation assumes no responsibility for the consequences of use of such information or for any infringement of patents or other rights of third parties that may result from its use. No license is granted by implication of otherwise under any patent rights of NVIDIA Corporation. Specifications mentioned in this publication are subject to change without notice. This publication supersedes and replaces all other information previously supplied. NVIDIA Corporation products are not authorized as critical components in life support devices or systems without express written approval of NVIDIA Corporation.

NVIDIA and the NVIDIA logo are trademarks or registered trademarks of NVIDIA Corporation in the U.S. and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

© 2013-2020 NVIDIA Corporation. All rights reserved.

NVIDIA Corporation | 2788 San Tomas Expressway, Santa Clara, CA 95051
www.nvidia.com