

NVIDIA GPU Debug Guidelines

Release 620

NVIDIA Corporation

Nov 11, 2026

Contents

1 Overview **3**

2 Initial Incident Report **5**

3 GPU Node Triage **7**
3.1 Reporting a GPU Issue 7
3.2 Understanding Xid Messages 7
3.3 Running DCGM Diagnostics 12
3.4 Running Field Diagnostics 12
3.5 Network Testing 12
3.6 Debugging Applications 13

4 Best Practices **15**
4.1 Collecting Node Metrics 15
4.2 Catching Errors Before They Occur 16

5 Notices **17**
5.1 Trademarks 18

This document provides GPU error debug and diagnosis guidelines, and is intended to assist system administrators, developers and FAEs get servers back up and running as quickly as possible.

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.

3.1. Reporting a GPU Issue

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

- Basic system configuration such as OS and driver info.
- A clear description of the issue, including any key log messages describing the problem.
- List of debug steps taken.
- A full listing of the log used for the key messages above.
- Output of `nvidia-bug-report.sh`.
- Fabric manager log files for HGX systems.
- DCGM Diagnostics logs.
- If associated with a user application, any details you can provide to the nature of the application (ISV code, framework, version numbers, and so on) and links to source code (if possible).
- 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, refer to 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.

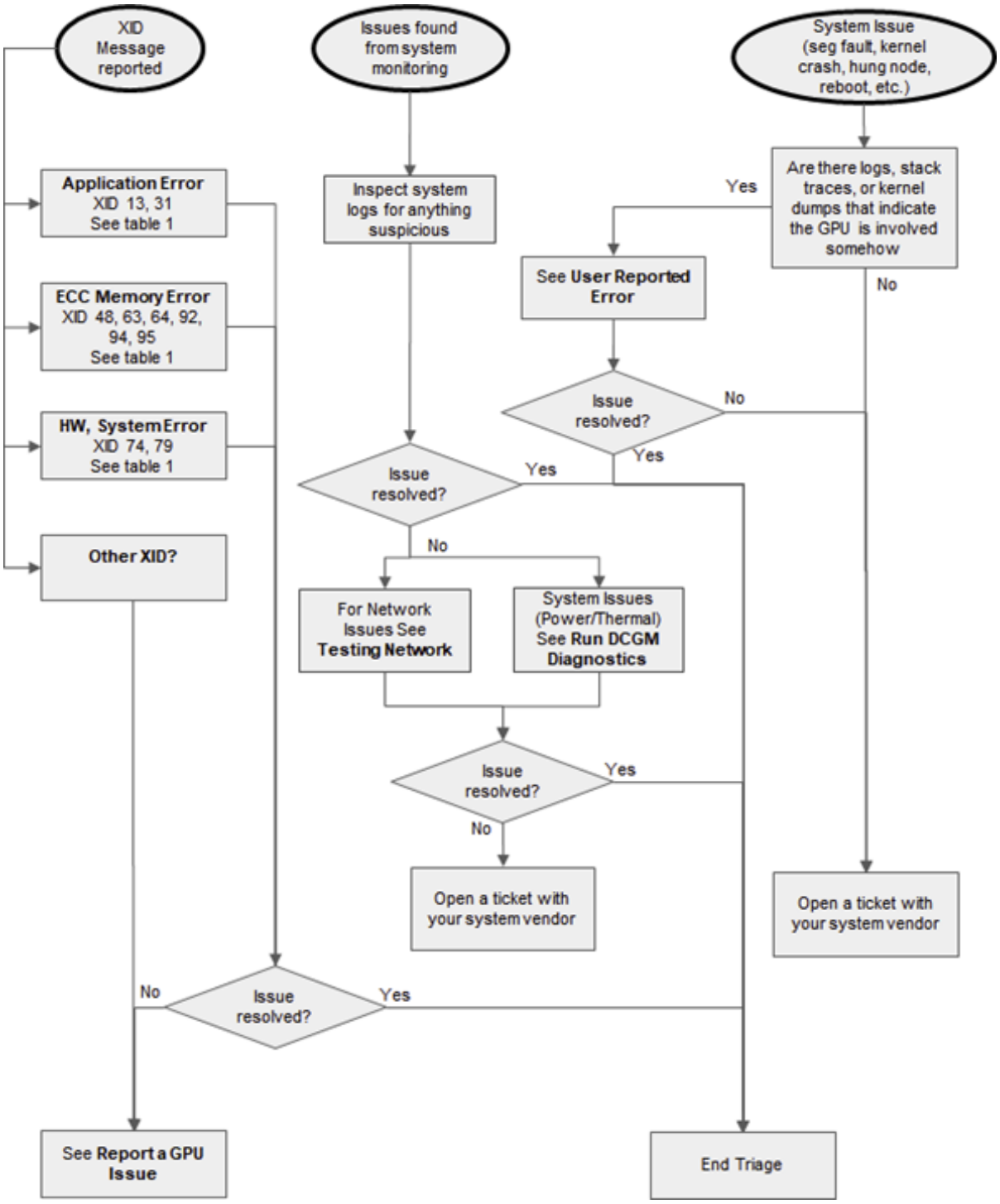


Figure 1: GPU Triage Flowchart

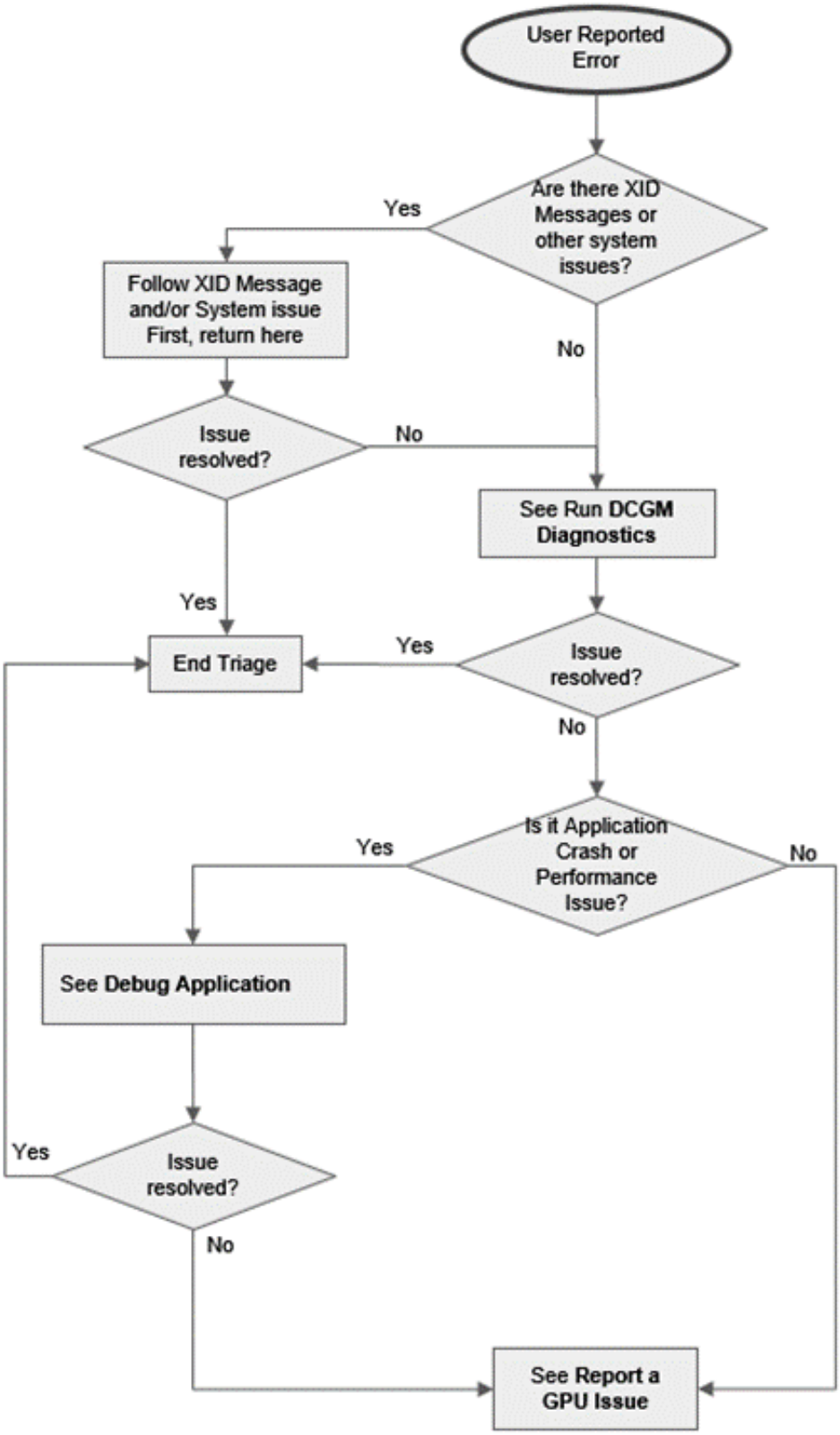


Figure 2: GPU Triage Flowchart (User Reported Error)

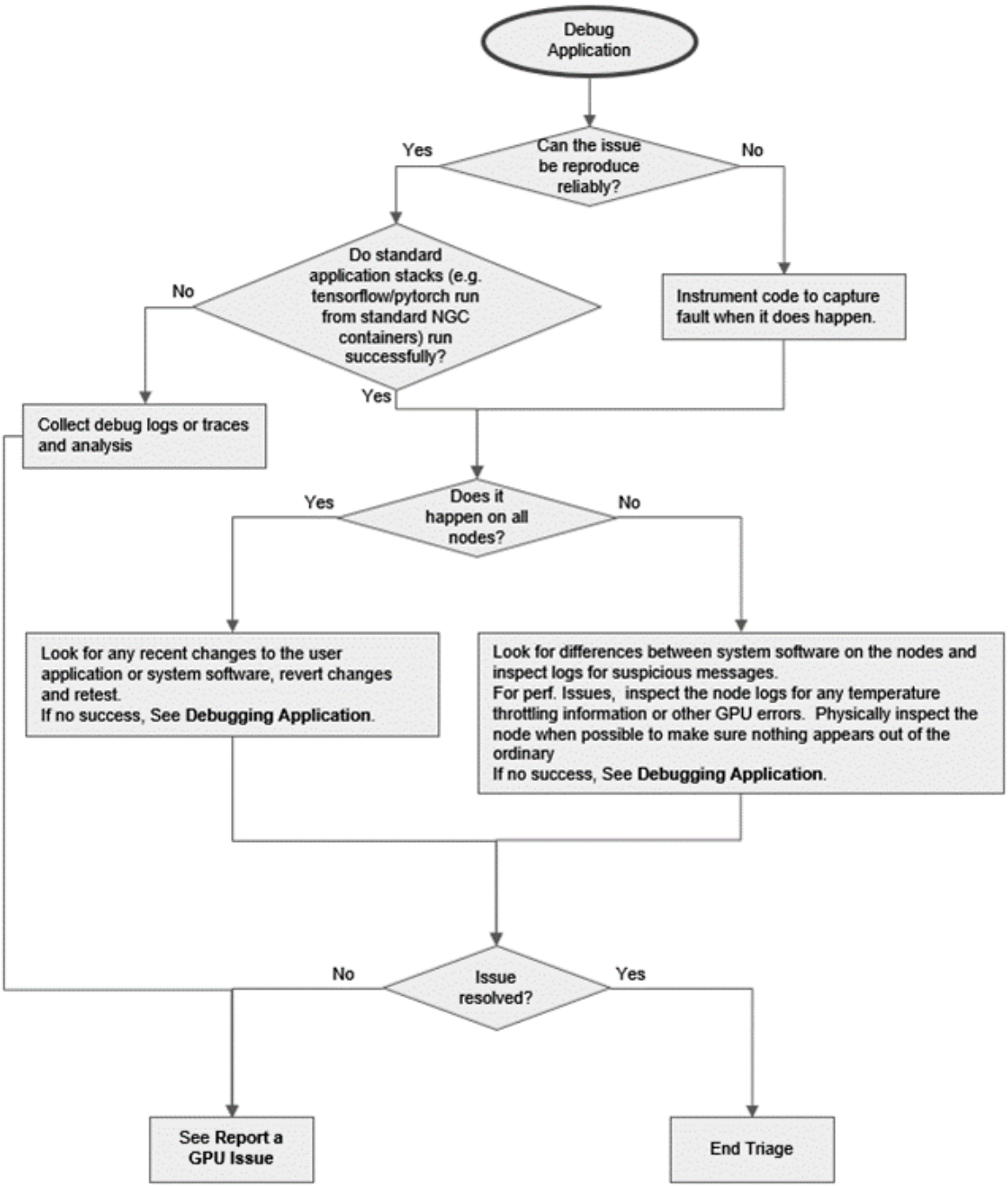


Figure 3: GPU Triage Flowchart (Debug Application)

Xid	Description	Action
13	Graphics Engine Exception	Run DCGM and Field diagnostics to confirm if the issue i
31	Suspected Hardware Problems	Contact the hardware vendor. They can run through the
45	Robust Channel Preemptive Removal	No action, informative only. Indicates channels affected
48	Double Bit ECC Error	If Xid 48 is followed by Xid 63 or 64: Drain/cordon the no
61	PMU Breakpoint	Report a GPU Issue and Reset GPU(s) reporting the XID
62	PMU Halt Error	Report a GPU Issue and Reset GPU(s) reporting the XID
63	**Legacy GPU:** ECC page retirement recording event	If associated with XID 48, drain/cordon the node, wait fo
	A100: Row-remapping recording event	If associated with XID 94, the application that encounte
64	**Legacy GPU:** ECC page retirement recording failure	See above, however the node should be monitored close
	A100: Row-remapping recording failure	The node should be rebooted immediately since there is
74	NVLink Error	Extract the hex strings from the XID error message. e.g.
79	GPU has fallen off the bus	Drain and see *Reporting a GPU Issue*.
92	High single-bit ECC error rate	See *Running Field Diagnostics* to collect additional debug
94	Contained ECC error occurred **(A100 only)**	The application that encountered the error needs to be r
95	Uncontained ECC error occurred **(A100 only)**	If MIG is enabled, drain any work on the other GPU instar
123	SPI PMU RPC write fail	Report a GPU issue and reset GPU(s) reporting the XID (r

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, refer to 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, refer to 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, refer to https://docs.nvidia.com/datacenter/tesla/fabric-manager-user-guide/index.html.

GPU/VM/System Reset Capabilities/Limitations:

Refer to the following excerpt from the `nvidia-smi` man page:

Trigger a reset of one or more GPUs. Can be used to clear GPU HW and SW state in situations that would otherwise require a machine reboot. Typically useful if a double bit ECC error has occurred. Optional `-i` switch can be used to target one or more specific devices. Without this option, all GPUs are reset. Requires root. There can't be any applications using these devices (for example, CUDA application, graphics application like X server, monitoring application like other instance of `nvidia-smi`).

	FM State	Bare Metal
NVIDIA Ampere architecture and later with direct NVLink connect	N/A	A GPU can be reset individually.
NVIDIA Ampere architecture + NVSwitch	Running	A GPU can be reset individually.
	NOT Running	Individual GPU reset is not suppo
Hopper and later + NVSwitch	N/A	GPU(s) can be reset individually

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, refer to
https://docs.nvidia.com/datacenter/dcgm/latest/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 Diagnostics

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, the `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 you 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, refer to https://docs.nvidia.com/datacenter/cloud-native/gpu-telemetry/latest/kube-prometheus.html.

A containerized version of DCGM is provided for Kubernetes deployments. For details, refer to 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, refer to see https://docs.nvidia.com/datacenter/tesla/hgx-software-guide/index.html and https://docs.nvidia.com/datacenter/tesla/driver-installation-guide/index.html

Chapter 5. Notices

This document is provided for information purposes only and shall not be regarded as a warranty of a certain functionality, condition, or quality of a product. NVIDIA Corporation (“NVIDIA”) makes no representations or warranties, expressed or implied, as to the accuracy or completeness of the information contained in this document and assumes no responsibility for any errors contained herein. NVIDIA shall have no liability for the consequences or use of such information or for any infringement of patents or other rights of third parties that may result from its use. This document is not a commitment to develop, release, or deliver any Material (defined below), code, or functionality.

NVIDIA reserves the right to make corrections, modifications, enhancements, improvements, and any other changes to this document, at any time without notice.

Customer should obtain the latest relevant information before placing orders and should verify that such information is current and complete.

NVIDIA products are sold subject to the NVIDIA standard terms and conditions of sale supplied at the time of order acknowledgement, unless otherwise agreed in an individual sales agreement signed by authorized representatives of NVIDIA and customer (“Terms of Sale”). NVIDIA hereby expressly objects to applying any customer general terms and conditions with regards to the purchase of the NVIDIA product referenced in this document. No contractual obligations are formed either directly or indirectly by this document.

NVIDIA products are not designed, authorized, or warranted to be suitable for use in medical, military, aircraft, space, or life support equipment, nor in applications where failure or malfunction of the NVIDIA product can reasonably be expected to result in personal injury, death, or property or environmental damage. NVIDIA accepts no liability for inclusion and/or use of NVIDIA products in such equipment or applications and therefore such inclusion and/or use is at customer’s own risk.

NVIDIA makes no representation or warranty that products based on this document will be suitable for any specified use. Testing of all parameters of each product is not necessarily performed by NVIDIA. It is customer’s sole responsibility to evaluate and determine the applicability of any information contained in this document, ensure the product is suitable and fit for the application planned by customer, and perform the necessary testing for the application in order to avoid a default of the application or the product. Weaknesses in customer’s product designs may affect the quality and reliability of the NVIDIA product and may result in additional or different conditions and/or requirements beyond those contained in this document. NVIDIA accepts no liability related to any default, damage, costs, or problem which may be based on or attributable to: (i) the use of the NVIDIA product in any manner that is contrary to this document or (ii) customer product designs.

No license, either expressed or implied, is granted under any NVIDIA patent right, copyright, or other NVIDIA intellectual property right under this document. Information published by NVIDIA regarding third-party products or services does not constitute a license from NVIDIA to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property rights of the third party, or a license from NVIDIA under the patents or other intellectual property rights of NVIDIA.

Reproduction of information in this document is permissible only if approved in advance by NVIDIA in writing, reproduced without alteration and in full compliance with all applicable export laws and regulations, and accompanied by all associated conditions, limitations, and notices.

THIS DOCUMENT AND 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. TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL NVIDIA BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Notwithstanding any damages that customer might incur for any reason whatsoever, NVIDIA’s aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms of Sale for the product.

5.1. Trademarks

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.

Copyright

©2013-2026, NVIDIA Corporation & affiliates. All rights reserved