Detecting and Managing GPU Failures

Nicholas P. Cardo
HPC Operations, Systems Group Lead
Swiss National Supercomputer Centre (CSCS)
Lugano, Switzerland
Nicholas.Cardo@cscs.ch

Abstract— **GPUs have been found to have a variety of failure modes. The easiest to detect and correct is a clear hardware failure of the device. However, there are a number of not so obvious failures that can be more difficult to detect. With the objective to provide a stable and reliable GPU computing platform, it is imperative to identify issues with the GPUs and remove them from service. At the Swiss National Supercomputing Centre (CSCS), a significant amount of effort has been invested in the detection and isolation of suspect GPUs. Techniques have been developed to identify suspect GPUs and automated testing put into practice, resulting in a more stable and reliable GPU computing platform. This paper will discuss these GPU failures and the techniques used identify suspect nodes**.

Keywords-component; formatting; style; styling; insert (key words)

I. About CSCS

"Founded in 1991, CSCS, the Swiss National Supercomputing Centre, develops and provides the key supercomputing capabilities required to solve important problems to science and/or society. The Centre enables world-class research with a scientific user lab that is available to domestic and international researchers through a transparent, peer-reviewed allocation process. CSCS's resources are open to academia, and are available as well to users from industry and the business sector. The Centre is operated by ETH Zurich and is located in Lugano."

"CSCS is a unit of the Swiss Federal Institute of Technology in Zurich (ETH Zurich). Since October 1st, 2008, CSCS has been part of the division headed by the Vice president for personnel, resources and infrastructures of ETH Zurich." [1]

II. Introduction

Traditionally, High Performance Computing (HPC) has been focused on Time to Solution (TtS). However, with the scalability of today's systems, the power required to run such systems would be enormous, putting the total cost of ownership, which includes operational costs, out of reach. In recent years there has been a new focus, Energy to Solution (EtS). Not only is it important to provide systems that can make the unobtainable calculations, obtainable, but also do so in a power efficient design. The combination of TtS and EtS in a system design is expanding the possibilities of scalability by allowing systems to grow in scale to new heights. [2]

GPU based systems have grown in HPC due to the fact of their ability to achieve high performance with low energy consumption. This has expanded the realm of scientific computing by allowing larger computationally capable systems to be deployed.

While traditional processor systems have enjoyed a long and successful life in HPC, allowing them to mature through the generations, GPUs on the other hand are relative newcomers to the playing field. While their success is growing, there are still areas for improvement. Only through a combined effort between GPU providers, computer system manufacturers and customers, can the success of the GPU continue to grow.

CSCS is committed to advancing GPU processing and to sharing gained knowledge for managing such a system. One area is focused on identifying problematic GPUs and removing them from service before a user application finds it. A significant time investment has been made to improve the diagnostic capabilities of GPUs in order to provide a rock solid computational system

III. Piz Daint Configuration

The flagship system for the Swiss national High Performance Computing (HPC) service is Piz Daint, a Cray® XC™ 30 name for a prominent mountain peak in the Grisons that overlooks the Fuorn pass[1].

Piz Daint has 5,272 compute nodes and 52 service nodes. Each compute node contains an 8-core Intel® Xeon® E5-2670 host processor running at 2.6 GHz with 32 GB of DDR3 memory and an NVIDIA® Tesla K20x Graphics Processing Unit (GPU) with 6 GB of GDDR5 memory. This yields for each compute node approximately 1.5 TFlops of computing power resulting in an impressive total computational capability of about 7.8 PFlops of theoretical peak performance for the entire system.

As of November 2014, Piz Daint sits at position #6 on the Top500 List. However, this only shows that the system is delivering on the TtS. The Green500 List focuses on the second component, EtS. At position #9, Piz Daint is showing that it possible for a state of art HPC system to be both computationally and efficient in the power consumption required to achieve solutions.

IV. GPU Failures

Production computing is supported by the fundamental principle of providing a stable and reliable compute platform. Minimizing interruptions and improving the overall success rate of completing applications will enhance the user experience on the system. The end result is increased scientific productivity.

It has been found that the system is very capable of identify node related issues but is lacking in the identification of GPU problems. Having a stable node with a problematic GPU presents itself as a problem to the user. It is a key objective to remove problems from service before detection by the user community.

With this in mind, it is important to detect and correct problems before returning the system back to production service. CSCS has gone to great lengths to develop a process to locate potential problems before exposing the system to the production workload. Each test performed has its roots in a problem that was encountered. Over time, this has grown into a suite of system and application checks designed to detect system problems, hardware and software.

A. Technical Error Identification[3]

GPUs report device errors through `xid` codes. Each `xid` code is intended to provide insight into the root cause of the problem, which in turn suggests a corrective action. The complication is that errors may multiple root causes. Each error identifies the following:

- XID Code
- Failure Description
- Hardware Error
- Driver Error
- User Application Error
- System Memory Corruption
- Bus Error
- Thermal Issue
- Frame Buffer Corruption

For example, XID 56 is a Display Engine Error that could be a Hardware Error or a Driver Error. Hopefully, in cases like this, there are sufficient error messages surrounding this error to provide the necessary clues to resolve the error.

B. Common Hardware Failures

Compute nodes can fail for a wide variety of hardware related reasons. In most cases, these are relatively simple to identify by performing fault isolation procedures or even by the errors called out in the logs. Even GPUs can fail in such a manner to indicate that the hardware has failed. The recovery procedures for these types of errors is to replace the failing component once identified. If only all failures were this simple to identify and correct…

C. GPU Has Fallen Off the Bus

One of the more annoying errors is the report that a “GPU Has Fallen Off the Bus”. The symptom to the user is that their application has failed due to an unexpected error on one of their nodes. When the node is investigated, it appears to be completely healthy. When searching system logs, the error can be found.

However, this error can be reported as a result of a wide variety of issues, as well as for an unknown error. Because of this, it has been incredibly difficult to diagnose a root cause. Since there are no hardware errors reported, it is difficult to simply replace parts. Furthermore, this is a software reported hardware error for which the hardware has not produced an error. As these nodes are reported or identified they are removed for service. The process is to remove and clean the GPU and then return the node back to service.

The annoyance factor is high for this error, as it requires a user to complain that their application has failed in an unexpected way. A number of bugs have been filed against this problem: 789858, 790527, 804390, 808113, 808114, 814107, and 818374.

It is possible that improvements in the GPU driver may help to further identify the root cause of this problem or even correct the problem. Plans are in place to upgrade CUDA® to version 6.5, which includes the latest driver.

D. GPU Killed by Applications

Applications have been found to hang or even crash the compute node. Sometimes only the GPU would crash but it is not possible to recover just the GPU. When any of these situations occurs, it is necessary to reboot the system, as a warmboot is not sufficient to recover the GPU. Relying on user applications to crash a compute node due to a GPU not functioning correctly is not a diagnostic. The challenge is to remove these nodes from service before they are found by a user application.

At CSCS, it was found that a GPU optimized version of HPCG is sufficient to crash the compute nodes that would have been crashed by user applications. A short 10-minute run will remove the suspect nodes from the system by crashing them. Because of the success rate of this test to find nodes with issues by taking them down, it has been nicknamed the “GPU Killer”.

The failure mode of each of these compute nodes is analyzed to determine the proper course of action. In some cases, the problem is clearly hardware, requiring the failing components to be replaced. However, in most cases, no component is called out and the error condition reported is simply a generic catchall error with no clear indication of the problem. This is the majority of failures encountered. It has been found that by pulling the GPU, cleaning all contacts and reseating the GPU, the problem goes away. There is no clear explanation as to why this corrects the problem. In many cases, simply rebooting the system clears the problem. The Standard Operating Procedure for this type of failure where no failed hardware can be identified is to reseat and clean the GPU.

This test has also been found to remove nodes from service that would have produced “GPU Has Fallen Off the Bus” errors. Since implementing this test, the occurrence of user applications killing nodes or encountering "GPU Has Fallen Off the Bus" have been almost eliminated.

E. *Slow GPUs*

Obtaining consistent performance levels across the entire system is important for all HPC systems. This is complicated on Piz Daint by the fact there is a host node and accelerator in the same node. Performance issues can often be difficult to identify as well as identify which node is affecting the applications overall performance. The problem is how to respond to a complaint that an application is running slower than normal.

CSCS developed a test code that utilizes a GPU optimized version of DGEMM to identify slow performing GPUs. In just a few minutes a base performance level of the GPU can be ascertained. By analyzing the results from all nodes against a base performance level, slow performing nodes can be identified. Nodes failing to meet the baseline are flagged for further investigation. The typical failure mode of this test is a drop in performance of over 40%, easy to detect. By performing this test prior to returning the system back to production CSCS has significantly reduced the occurrences of reports of degraded performance.

However, relying on an application to detect problems is not a sustainable solution and the root of the problem needed to be identified. A device showing signs of such performance degradation should also be showing signs of a problem. After careful investigation, the problem was clear and could be detected with `nvidia-smi -q`. The root cause of this problem was that the PCI link width dropped from 16x down to 8x.

```
GPU Link Info
    Link Width
        Max          : 16x
        Current      : 8x
```

This excerpt from `nvidia-smi -q` shows that the maximum link width is 16x and that currently it is only performing at 8x. The symptom is a significant drop in application performance.

Now, a simple check after each system boot can detect if nodes are exhibiting this problem. The DGEMM test is still performed to help catch other failure modes where the symptom is performance degradation.

Ideally, this check should be performed as part of the Node Health Check as nodes can degrade after system boot. Bug 822929 has been filed to request that this test be included in order to catch these failures. However, Cray support has decided to treat this as a Request For Enhancement. Currently, there is nothing supplied by Cray to catch that the PCI link width is running at less than the maximum.

The corrective action for this failure is to replace the GPU. There is no known way to restore the PCI Link Width back to its maximum rate.

V. System and Regression Test Suites

Building tools and test cases is only part of the solution. It is crucial to have a reliable, yet simple, interface to consistently run the tests in an automated method. Furthermore, the regression suite needs to be quick and clearly identify failures. [4]

After each reboot, it takes approximately 30 minutes to test and validate the system is ready for production service. This is done through a series of system checks followed by the regression test suite.

The first set of tests focus on the system itself. Here, checks for correct device settings, error conditions, and operational ability are all performed. Most of these checks are fairly standard and performed on HPC systems in general. However, a few checks specifically for GPUs have been added. Piz Daint runs with the GPU Operation Mode (GOM) set. A check is performed to ensure that the setting is not lost. Another is for the PCI Link Width as described earlier in this paper. The objective of these tests is to provide a stable compute platform to be validated by the regression test suite. Ideally, all error conditions would have been detected by the completion of the system checkout tests.

Following a successful system checkout, the regression test suite is run. These are an automated set of tests designed to validate the user environment as well the capability of the system to run the production workload. Each test has been specifically designed based on reported errors and the ability to quickly detect any problems that may have been missed by the system checkout.

In the end, the combination of both sets of tests results in a system that has been stressed, checked, and is ready for the standard production workload. This has proven itself to improve the reliability of the system as well as the user's ability to get their work done efficiently and effectively.

CSCS continues to refine the existing checks and add new checks as failure modes are identified. These tests have proven to be highly effective in identifying problems before putting the system into full production operation.

VI. Metric of Success

There are many ways to measure success. For HPC systems, the true measure of success is the amount of science that can be accomplished on the system. Providing the maximum possible hours on the system with the least amount of interruptions results in the maximizing the time spent on science.

For Piz Daint, the benefits were realized in two areas: the time necessary to return the system back to production and minimizing user found suspect nodes.

Before returning the system to production after a system boot, both the system checks and regression tests would be run to validate the state of the system. Several runs would be required in order to complete the regression tests. This process would take approximately 3 hours to complete. Now, after implementing improved tests and checks, the process completes in less than 1 hour. This represents an additional 10,544 node hours that are now available for scientific computations.

The user experience on the system is an important consideration. The checks now in place are so successful, that user interrupts due to GPU related issues have virtually been eliminated. These types of failures are no longer routine, but rather have become exceptions.

The improvements made in time to return to service and user interrupts have returned a significant amount of compute cycles back to the users while improving their application success rate.

VII. CONCLUSION

Maintaining any HPC system can be a challenge. Add in the complexity of a GPU and the combination of challenges can mount. With careful planning and management, these challenges can be overcome to provide a stable and reliable compute platform creating a positive end-user experience. The benefits enjoyed by a stable compute platform far exceed the overall investment in the creation of the tests used.

There is still much work to complete. It is possible that software improvements in NVIDIA® CUDA® Toolkit 6.5 and the latest driver improve error handling and recoverability. Plans are in place to complete this upgrade along with an upgrade to the latest version of the Cray Linux Environment (CLE) and Programming Environment (PE).

Work will continue on enhancing and refining the system and regression tests in order to optimally find suspect nodes and remove them from service before detection by a production application.

The end result of all the time invested has been an overall improvement in the reliability of the system. Most problems are now caught before exposing the system to the production workload. However, there are still areas where improvements can be made. All these tests can identify issues, but they are run only after each system boot. The regression test suite has the ability to focus on individual nodes, once a suspect node is identify, usually from a produce application failure. Ideally, improvements in the Node Health Checker could better capture error conditions and remove nodes from service as they fail.

Work continues on understanding each GPU issue in hopes of finding a solution to mitigate the impact of that type of failure. Additionally, work continues on identifying ways to perform as much of the system software updates without the requirement of removing the system from production operation. The ultimate goal is to maximize the available computational hours for the user community.

Next steps include tracking the various errors that could be produced by the GPU. There are 69 documented error codes (XID) by NVIDIA®. For each error code, the possible root cause is documented. By tracking these error codes, it will be possible to better quantify the root causes and improve the time to diagnosis and return to service. [66]

ACKNOWLEDGMENT

The work reported within this paper is the result of a large concentrated effort by a large number of staff at CSCS. I would like to recognize Timothy Robinson for his work on the regression test suite. I would also like to recognize Vincenzo Annaloro, Nicola Bianchi, Marco Induni, and Fabio Verzelloni for their work on implementing system level checks.

REFERENCES

[1] CSCS Web Pages, http://www.cscs.ch/about/index.html.

[2] "Application centric energy-efficiency study of distributed multi-core and hybrid CPU-GPU systems", Cumming, Fourestey, Fuhrer, Gysi, Fatica, Schultess, SC14.

[3] "Understanding XID Errors", NVIDIA®,
http://docs.nvidia.com/deploy/xid-errors.

[4] "Systems-level Configuration and Customisation of Hybrid Cray XC30, Aielli, Alam, Annaloro, Bianchi, Benini, McMurtrie, Robinson, Verzelloni", CUG 2014.