On Testing GPU Memory for Hard and Soft Errors

Guochun Shi, Jeremy Enos, Michael Showerman, Volodymyr Kindratenko
National Center for Supercomputing Applications
University of Illinois at Urbana-Champaign
Urbana, IL, USA
e-mail: {gshi|jenos|mshow|kindr}@ncsa.illinois.edu

***Abstract*—NVIDIA GPUs are becoming increasingly popular in scientific computation as a way to accelerate the execution of computationally demanding codes. The graphics memory used in GPUs is not protected against soft errors that may be caused by cosmic radiation and thus is a source of concern for the scientific computing community. In this short paper we report on an attempt to test GPU memory for both permanent memory errors due to manufacturing defects and prolonged use and soft errors due to single radiation events. We present a new GPU memory test methodology and show results of error measurements on two large GPU clusters.**

Keywords-GPU, memory test

I. INTRODUCTION

Modern high-performance computers (HPCs) are built from highly reliable components that are generally guaranteed not to corrupt data during normal operation. Prior to production deployment, HPC systems are rigorously tested by the manufacturers to eliminate defective components. In addition, various error detection and correction schemas are implemented directly in hardware to ensure data integrity during the normal operation of the equipment. Users of HPC systems expect that the computed results are correct and are not subject to errors due to hardware faults. Consequently, the HPC community is very reluctant to accept any new technology that does not guarantee data integrity.

Not surprisingly, one of the arguments *against* using graphics processing units (GPUs) in scientific computing is based on the fact that the Graphics Double Data Rate 3 (GDDR3) memory chips used in GPU cards are not protected against "soft errors." Even though Compute Unified Device Architecture (CUDA) GPUs from NVIDIA deliver performance levels far exceeding those attainable on modern multi-core CPUs, the scientific computing community remains skeptical and cautious when it comes to using GPUs in production runs.

Since NCSA deployed two large GPU clusters for use by the scientific computing community, we thought to investigate how reliable the GPU memory is in practice. In our GPU memory tests we consider both manufacturing defects and susceptibility to "soft errors." This is still an ongoing work and in this paper we report on the methodology used to test GPU memory and our initial findings based on testing over 500 NVIDIA Tesla GPUs.

II. NVIDIA GPU MEMORY SYSTEM

This work is based on the NVIDIA Tesla GPU product line that is specifically designed for use in general purpose computing. C1060 Computing Processor and S1070 1U GPU Computing Server are the latest products based on NVIDIA's T10 GPU chip that are widely used in HPC systems. Both products use 32 pieces 32M × 32 GDDR3 136-pin ball grid array (BGA) mounted synchronous dynamic random access memory (SDRAM), or 4 GB of memory per GPU. The memory operates at 800 MHz and is connected to the GPU via a 512-bit memory interface that provides over 102 GB/s memory bandwidth.

III. GPU MEMORY TEST METHODOLOGY

We consider two classes of errors in GPU memory chips: i) permanent hardware errors due to manufacturing defects and as a result of prolonged use, and ii) transient, or *soft* errors that are induced by cosmic radiation.

A. Detecting permanent errors

Permanent memory errors can manifest themselves in a number of ways and under varying conditions [1], and thus require a comprehensive test suite. We adapted test methodology used in Memtest86 utility [5] modified for GPU device memory. The main idea is to write a test pattern to memory, read it back and verify if it is the same as what was written. To realize this idea, we implemented separate GPU kernels in CUDA C that execute the basic functions listed above and can be used to assemble tests consisting of sequences of basic operations using different test patterns.

The GPU kernels are executed on a grid of Nx1x1 thread blocks, with only 1 thread per each thread block. When launched, each kernel works with N MBs of GPU memory, one MB per thread. To test the entire M MB of GPU memory, we run M/N kernels with different initial memory addresses. In our tests M=4,037 MB and N=128 MB.

The GPU memory test program starts by allocating all allocatable GPU memory (M MB) and launching a sequence of individual tests. Within the kernels that verify values stored in the GPU memory, when the expected values are not the same as the values stored in memory, the error counter is incremented and the error address, current and expected values, and other useful information are saved. When the

GPU kernel exits, the error information is pulled out to the CPU memory and is logged in an error file.

The ten memory tests are similar to those used in the Memtest86 utility, with adjustments made for the GPU execution. Table I provides information about these tests. Figure 1 gives an example of the *moving inversion with 8-bit pattern* test (test #3 in Table I), and Figure 2 shows the GPU implementation of one of the three kernels used in the test.

```
void move_inv_test(char* ptr, unsigned int tot_num_blocks, unsigned int
p1, unsigned p2) {
   unsigned int i;
   char* end_ptr = ptr + tot_num_blocks* BLOCKSIZE;
   for (i= 0;i < tot_num_blocks; i+= GRIDSIZE) {
      dim3 grid;  grid.x= GRIDSIZE;
      _move_inv_write<<<grid, 1>>>(ptr + i*BLOCKSIZE, end_ptr, p1);
   }
   for (i=0;i < tot_num_blocks; i+= GRIDSIZE) {
      dim3 grid;  grid.x= GRIDSIZE;
      _move_inv_readwrite<<<grid, 1>>>(ptr + i*BLOCKSIZE, end_ptr,
p1, p2, err_count, err_addr, err_expect, err_current);
      error_checking("move_inv_readwrite",  i);
   }
   for (i=0;i < tot_num_blocks; i+= GRIDSIZE) {
      dim3 grid;  grid.x= GRIDSIZE;
      _move_inv_read<<<grid, 1>>>(ptr + i*BLOCKSIZE, end_ptr, p2,
err_count, err_addr, err_expect, err_current);
      error_checking("move_inv_read",  i);
   }
}
```

Figure 1. Host side of the moving inversion with 8-bit pattern test.

```
__global__ void _move_inv_readwrite(char* _ptr, char* end_ptr, unsigned
int p1, unsigned int p2, unsigned int* err, unsigned long* err_addr, unsigned
long* err_expect, unsigned long* err_current) {
   unsigned int i;
   unsigned int* ptr = (unsigned int*) (_ptr + blockIdx.x*BLOCKSIZE);
   if (ptr >= (unsigned int*) end_ptr) return;
   for (i = 0;i < BLOCKSIZE/sizeof(unsigned int); i++) {
      if (ptr[i] != p1) { RECORD_ERR(err, i, p1, ptr[i]); }
      ptr[i] = p2;
   }
   return;
}
```

Figure 2. Read-test-write GPU kernel used in the moving inversion test.

B. Detecting soft errors

A single radiation event can cause a data bit stored in memory to be corrupted until new data is written to the memory, manifesting itself as a *soft error*. The predominant source of such single radiation events is cosmic rays [3]. The rate at which soft errors occur is measured in the *number of failures in time* (FIT); 1 FIT equals 1 soft error per billion hours of device operation. This rate is also sometimes reported as FIT per Mbit.

Soft memory errors occur mostly during memory access, so the error rate rises with the intensity of memory use [4]. Thus, some of the tests described in the previous section, if run long enough, can be used to detect soft errors as well. In particular, the moving inversion with shifted pattern test (#6) is a good candidate, as it writes to memory and reads from it continuously for a prolonged period of time. However, the tests designed for hardware error detection do not stress the memory system enough, thus not maximizing the chances of catching many soft errors. Therefore, we implemented a GPU kernel (test #10 in Table I) that launches a large number of threads that access the memory simultaneously. It reads the test pattern previously stored in the GPU memory, checks it against the expected value, and writes its complement back to the memory. If an error is found, it reads from the affected memory one more time and reports the two read values. The test patterns are random numbers. The kernel is launched in an infinite loop and uses the same test pattern 1,000 times before moving to a new test pattern.

TABLE I. IMPLEMENTED GPU MEMORY TESTS

Test	Test pattern	Purpose
0: walking ones	walking ones	address bus test
1: own address	own physical address	address bus test
2: moving inversion	moving inversions (ones and zeros)	stuck-at faults errors
3: moving inversion with 8-bit pattern	moving inversions (8-bit pattern)	stuck-at faults errors
4: moving inversion with random pattern	moving inversions (random pattern)	data sensitive errors
5: block move test	block move, 64 moves	data sensitive errors
6: moving inversion with shifted pattern	moving inversions, 32-bit pat	data sensitive errors
7: random numbers sequence	random values and their compliments	data sensitive errors
8: modulo 20 random pattern	modulo 20, ones and zeros	data sensitive errors
9: bit fade test	0xFFFFFFFF	data retention test
10: soft errors test	random values	soft errors

IV. EXPERIMENTAL RESULTS

For our experiments, we used two clusters with NVIDIA Tesla S1070 GPU accelerators attached. *AC*, which is an extension of *QP* cluster [8], is an AMD based research cluster with 32 nodes and four GPUs per node. *Lincoln* is an Intel-based production cluster with 192 nodes and two GPUs per node (96 Teslas, each shared between two nodes).

We envision the GPU memory test as a utility that is used continuously for monitoring the system for the appearance of new hardware faults and for collecting statistics about the rate of soft errors. To accomplish this goal, on AC we ran the GPU memory test during all the time available on the system that would otherwise be idle. To facilitate this, the batch system and scheduler (Torque and Moab, respectively) are configured to allow for preemptible jobs. By default, any job submitted would be a *preemptor*. The GPU memory test jobs are submitted with the *preemptee* flag, allowing regular job submission to cause them to be canceled and restarted when resources were available. A separate memtest job is created for each node, specifically targeted to that node. In the event of error detection, the job emails notification and continues testing.

On Lincoln, we run the GPU memory test as a regular job for a period of time sufficient to run through all the tests for detecting hardware faults. We have not yet attempted to collect statistics about soft errors on Lincoln.

A. Soft errors

Modern memory devices are believed to have a 1,000 to 5,000 FIT per Mbit error rate, or about one bit error per month per gigabyte of memory [2], and thus we should expect to see error rates at the order of tens of millions FIT per Mbit, or 4 bit errors per GPU card per month of continuous operation. We conducted two independent runs of the soft error memory test (test #10): i) combined 7,222 hours of operation per ~4 GB of memory while running the test on AC cluster nodes when not in use during a 15-day period, and ii) combined 10,080 hours of operation per ~4 GB of memory running the test on 15 AC cluster nodes solely dedicated to this test. Neither of these tests, however, detected any soft errors. This indicates that either our test methodology is not correct, or the soft errors are much less frequent than other studies have indicated.

B. Permanent errors

A single pass through the GPU memory tests for detecting permanent hardware faults (tests #0-9) requires almost three and a half hours to execute all 10 tests. The bit fade test (#9) is the most time-consuming at three hours. Since it is only designed to catch errors due to the data retention issues over long periods of time, we do not run it as part of the routine test. All other tests combined are responsible for 27.8 minutes; the moving inversion with shifted pattern test (#6) is responsible for the majority of the time at about 23.5 minutes. After tests #0-8 are done, we also run test #10, which turned out to be instrumental in finding some sudden hardware errors, as described later. Once a node with memory error is found, we reboot the node and perform the same tests to confirm our findings.

We were able to identify 4 GPU cards out of 128 on AC cluster and 5 GPU cards out of 378 tested on Lincoln cluster with faulty memory that we were able to consistently reproduce using our test methodology. Here we list errors found on the AC cluster:

- Node 3, GPU 3: Tests 1, 5-8, and 10 fail within minutes of starting the tests, reporting many errors for different memory addresses.
- Node 5, GPU 1: Tests 6 and 10 fail a few times per a 24-hour period. Errors are detected on different memory addresses. Test 10 reports single bit flips that would be characteristic of soft errors. However, in light of no other GPU nodes on the same Tesla unit reporting similar errors, we do not think these are soft errors.
- Node 25, GPU 1: Tests 6 and 10 fail within few hours. Errors are detected on different memory addresses. In contrast to the previous node, test 10 reports errors that are not characteristic of soft errors. Test 10 writes a random value to memory and then reads it back twice. The value returned on the first read has one random bit flipped, but the value returned on the second read is correct.
- Node 28, GPU 1: Tests 1-4, 6, and 10 report a single-bit flip error within a few hours of running, coming from the same memory address (0xe7fbc990). The error appears to be that a single bit (bit # 6) is stuck at '1.'

While working on this paper, we discovered similar GPU memory test software under development at Stanford, called MemtestG80 [7]. Using the Stanford code, we were able to confirm memory errors on 3 out of 4 nodes; we were not able to confirm errors on Node 5, GPU 1.

NVIDIA confirmed that some Tesla units that were in our clusters were not properly tested with the manufacturing tests and thus could have had bad memory.

Since some of the tests use randomly generated test patterns, there is a danger that a test pattern that can trigger an erroneous response may not occur in a given test run. We noticed, however, that when an error is detected it is repeatable at a frequency between every three minutes up to 20 hours at most. It is noteworthy that different GPUs had errors show up with different tests, but any given GPU failed consistently within the same test. Also, no test would necessarily fail on its first pass.

V. Conclusions and Future Work

We have implemented a comprehensive set of tests for checking the GPU memory for permanent hardware errors and are refining our methodology and software tools for monitoring the rate of transient soft errors. In two large NVIDIA GPU installations at NCSA we have yet to see any soft errors, but 1.8% of tested GPUs turned out to have permanent memory errors. Our tests found the failures on systems that should not have been shipped to customers if the manufacturing tests were run properly. NVIDIA has since corrected the problem. But even with 1.8% of systems confirmed to have memory issues, the memory failure rate is below of what has been reported on non-GPU clusters, e.g., Li et al. [6] discovered hardware memory faults on 9 out of 212 (or 4.5%) Ask.com servers.

The GPU memory test code is available for download from http://sourceforge.net/projects/cudagpumemtest/.

Acknowledgment

This work is in part sponsored by the Institute for Advanced Computing Applications and Technologies. Some of the GPUs used in this study are available due to a generous donation of hardware from NVIDIA. Special thanks to NCSA's Jim Long for running tests on *Lincoln* cluster and Trish Barker for help in preparing this article.

References

[1] R. Dean Adams, High Performace Memory Testing: Design Principles, Fault Modeling and Self-Test, Kluwer: Boston, 2003.

[2] Tezzaron Semiconductor, Soft errors in electronic memory. White paper, 2004. http://www.tezzaron.com/about /papers/papers.html.

[3] R. Baumann, Soft errors in advanced computer systems, IEEE Design & Test of Computers, vol. 22, no. 3, pp. 258–266, 2005.

[4] A. Johnston, Scaling and Technology Issues for Soft Error Rates, In Proc. Fourth Ann. Research Conference on Reliability, 2000.

[5] Memtest86 Memory Diagnostic Utility, http://www.memtest86.com

[6] X. Li, K. Shen, M. Huang, A Memory Soft Error Measurement on Production Systems, In Proc. USENIX Annual Technical Conference, 2007.

[7] MemtestG80: A Memory and Logic Tester for NVIDIA CUDA-enabled GPUs, https://simtk.org/home/memtest/

[8] M. Showerman, J. Enos, A. Pant, V. Kindratenko, C. Steffen, R. Pennington, W. Hwu, QP: A Heterogeneous Multi-Accelerator Cluster, In Proc. 10th LCI International Conference on High-Performance Clustered Computing, 2009