Failing In Place for Low-Serviceability Infrastructure Using High-Parity GPU-Based RAID

Matthew L. Curry, the University of Alabama at Birmingham and Sandia National Laboratories[1], mlcurry@sandia.gov
Lee Ward, Sandia National Laboratories[1], lee@sandia.gov
Anthony Skjellum, the University of Alabama at Birmingham, tony@cis.uab.edu

Motivation

In order to provide large quantities of high-reliability disk-based storage, it has become necessary to aggregate disks into fault-tolerant groups based on the RAID methodology [6]. Most RAID levels do provide some fault tolerance, but there are certain classes of applications that require increased levels of fault tolerance within an array. Some of these applications include embedded systems in harsh environments that have a low level of serviceability, or uninhabited data centers servicing cloud computing.

When describing RAID reliability, the Mean Time To Data Loss (MTTDL) calculations will often assume that the time to replace a failed disk is relatively low, or even negligible compared to rebuild time. For platforms that are in remote areas collecting and processing data, it may be impossible to access the system to perform system maintenance for long periods. A disk may fail early in a platform's life, but not be replaceable for much longer than typical for RAID arrays. Service periods may be scheduled at intervals on the order of months, or the platform may not be serviced until the end of a mission in progress. Further, this platform may be subject to extreme conditions that can accelerate wear and tear on a disk, requiring even more protection from failures.

We have created a high parity RAID implementation that uses a Graphics Processing Unit (GPU) to compute more than two blocks of parity information per stripe [4], allowing extra parity to eliminate or reduce the requirement for rebuilding data between service periods. While this type of controller is highly effective for RAID 6 systems, an important benefit is the ability to incorporate more parity into a RAID storage system. Such RAID levels, as yet unnamed, can tolerate the failure of three or more disks (depending on configuration) without data loss.

While this RAID system certainly has applications in embedded systems running applications in the field, similar benefits can be obtained for servers that are engineered for storage density, with less regard for serviceability or maintainability. A storage brick can be designed to have a MTTDL that extends well beyond the useful lifetime of the hardware used, allowing the disk subsystem to require less service throughout the lifetime of a compute resource. This approach is similar to the Xiotech ISE [8]. Such a design can be deliberately placed remotely (without frequent support) in order to provide colocation, or meet cost goals.

[1]Sandia National Laboratories is a multi-program laboratory operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin company, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Extra Parity or Hot Spares?

While it is possible to include hot spares, intentionally idle disks that automatically replace failed disks, there is an increasing source of errors that hot spares cannot address. Unrecoverable read errors (UREs), encountered when a disk cannot read data that was previously written, are a well-known problem [5]. Disks are becoming much larger, but no significant improvements have been made to bit error rates (BERs), which range from one sector per 10^{14} bits read to one sector per 10^{15} bits read for SATA disks [5].

For a single two-terabyte drive with a BER of one sector in 10^{15} bytes, reading all of its contents has a 1.58% probability of encountering a lost sector. However, in an array composed of sixteen of these disks, a single pass has a 22.6% chance of losing data. When an array is left with no available parity during a disk rebuild, bit errors are uncorrectable, and data are lost. Figure 1 shows the relationship between UREs and lost data. While a Bit Error Rate (BER) of 10^{-15} provides much more protection than a BER of 10^{-14}, the chance of data loss is still high if no parity is present to support a rebuild, depending on the size of the array. As disk sizes increase, the likelihood of encountering a URE becomes unacceptably high when no parity is available. Even disks with a BER of 10^{-16} cannot read 13 TB with a 99% probability of success.

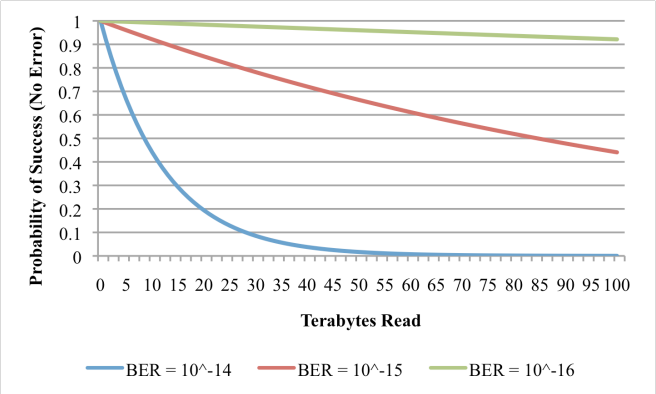

Figure 1: Probability of Avoiding URE

A further disadvantage of the hot spare is the unavoidable window of vulnerability between when a disk has failed and when a hot spare has been populated with reconstructed data. When performing a rebuild operation, disks can be stressed to the point of further failures, leaving the array in a precarious state, lacking further fault tolerance. For example, Sandia systems have experienced double-disk failures in ten-disk RAID 6 arrays, requiring extensive system maintenance to recover data due to subsequent UREs.

A GPU-Based RAID Controller

RAID controllers do not typically have the capability of supporting single volumes that can tolerate more than two arbitrary failures, a requirement of RAID 6. Popular hierarchical RAID levels [1], while capable of supporting more than two failures, do not offer complete protection: RAID 60, for example, can tolerate more than two failures in some cases, but can lose data with as few as three failures.

In order to support more fault tolerance, we have defined, implemented, and optimized a high-performance software RAID based on graphics processing units [4]. It incorporates a Reed-Solomon coding library that uses NVIDIA GPUs to provide a user-selectable number of parity blocks per data stripe [2, 3].

A technical challenge in implementing a RAID system with GPUs is the lack of kernel interfaces for accessing GPU compute resources. The GPU-based RAID software is implemented entirely in user space, interfacing with clients via an iSCSI target. This arrangement allows for network clients to mount the volume, or the local host to mount the volume via a loopback interface.

Performance

Performance is evaluated by benchmarking three GPU-based RAID levels, ranging from a two disk fault tolerant configuration (denoted as GPU-RAID +2) to a four disk fault tolerant configuration (denoted as GPU-RAID +4). The GPU used is an NVIDIA GeForce GTX 285. The tests for reads and writes were streaming, contiguous one megabyte operations through the first ten gigabytes of the volume. The results can be found in Figure 2.

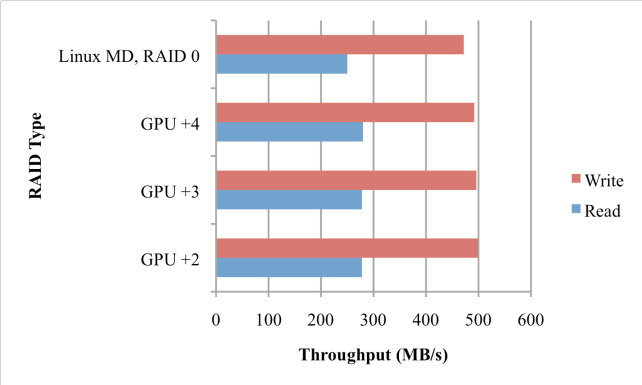

Figure 2: Performance Comparison of RAID Levels through iSCSI Target, Mounted Locally

While the GPU-based implementation performs read-verification, a feature that is missing from Linux, performance remains comparable to accessing a Linux MD RAID 0 device through the iSCSI target. This points to the iSCSI target implementation constituting the bottleneck in the tests, as the disk bandwidth consumed by reading parity does not affect the throughput attained. If the storage must be accessed through iSCSI, the performance observed indicates that there is no penalty for performing a streaming workload, or using a file system that has a streaming access pattern (like a log-structured file system [7]).

While no performance differences between different levels are apparent from Figure 2, there is a computational cost for implementing higher parity. The speed of the iSCSI target hides the computation in this case. In other cases, the computation may be hidden by the speeds of the disk subsystem, with the exception of the bandwidth cost of read verification [4].

Conclusions

For workloads where reliability is key, but conditions are sub-optimal for routine serviceability, a high-parity RAID can provide extra reliability in extraordinary situations. For example, for installations requiring very high Mean Time To Repair, the extra parity can eliminate certain problems with maintaining hot spares, increasing overall reliability. Furthermore, in situations where disk reliability is reduced because of harsh conditions, extra parity can guard against early data loss due to lowered Mean Time To Failure. If used through an iSCSI interface with a streaming workload, it is possible to gain all of these benefits without impacting performance.

References

[1] S. Baek, B. Kim, E. Joung, and C. Park, "Reliability and Performance of Hierarchical RAID with Multiple Controllers," *Proceedings of PODC 2001: ACM Symposium on Principles of Distributed Computing*.

[2] M. Curry, A. Skjellum, H. L. Ward, and R. Brightwell, "Arbitrary Dimension Reed-Solomon Coding and Decoding for Extended RAID on GPUs," *Petascale Data Workshop 2008*.

[3] M. Curry, A. Skjellum, H. L. Ward, and R. Brightwell, "Gibraltar: A Library for RAID-Like Reed-Solomon Coding on Programmable Graphics Processors," submitted for publication.

[4] M. Curry, H. L. Ward, A. Skjellum, and R. Brightwell, "A Lightweight, GPU-Based Software RAID System," to appear in *Proceedings of ICPP 2010: International Conference on Parallel Processing*.

[5] J. Gray and C. van Ingen, "Empirical Measurements of Disk Failure Rates and Error Rates," technical report, December 2005.

[6] D. Patterson, G. Gibson, and R. Katz, "A Case for Redundant Arrays of Inexpensive Disks," *Proceedings of ACM SIGMOD 1988*.

[7] M. Rosenblum and J. Ousterhout, "The Design and Implementation of a Log-Structured File System," *ACM Transactions on Computer Systems*, Vol. 10, No. 1, Feb. 1992.

[8] Xiotech, "ISE - The New Foundation of Data Storage," http://www.xiotech.com/ise-technology.php, accessed on May 25, 2010.