Using GPUs to Enable Highly Reliable Embedded Storage

Matthew L. Curry, University of Alabama at Birmingham, curryml@cis.uab.edu
Anthony Skjellum, University of Alabama at Birmingham, tony@cis.uab.edu
H. Lee Ward, Sandia National Laboratories, lee@sandia.gov
Ron Brightwell, Sandia National Laboratories, rbbrigh@sandia.gov

Introduction

Embedded systems are often exposed to harsh environments. Such conditions often result in increased failure rates of the various hardware components in the system, including storage devices. This lowered reliability exposes a need for more highly reliable storage than current RAID solutions can provide (while using a minimum of hardware). Furthermore, it would be desirable to have a tunable amount of reliability based on the anticipated conditions and failure rates without sacrificing I/O performance.

Current high-performance RAID solutions are limited in the amount of parity that can be created from a single set of data blocks. These RAID solutions are typically ASIC technologies that are created to support efficient implementations of standard RAID levels [1], supporting up to two parity blocks per stripe of data. While more reliable RAID levels have been invented [2], these are not tunable technologies.

Reed-Solomon coding is a widely used optimal erasure coding algorithm which can be used to provide an arbitrary amount of parity information [3]. However, it is not well-suited to most CPU architectures, resulting in low performance in practice. Accelerating Reed-Solomon coding is an important step in creating a high-performance storage system supporting high reliablity.

Approach

The primary reason that CPUs suffer from lower performance in Reed-Solomon coding is the lack of parallel table lookup functionality. We have identified graphics processing units (GPUs) as supporting this feature, along with very wide vector functionality for performing the necessary arithmetic for parity generation. We have implemented a parity generation component of a RAID-like system which utilizes a CUDA-enabled GPU from NVIDIA.

This component could be integrated into a RAID-like system via pipelining of read and write requests, with three stages: A user-buffer stage, which serves as a buffer containing the user data before a write or after a read; a parity generation stage, where data is transferred back and forth between main memory and the GPU memory for parity generation or data regeneration; and a disk stage, which interfaces directly with the disks of the array. The throughput of such a system would be limited by the slowest component, so a goal of the system would be to have useful configurations where the GPU stage is faster than the disk stage for most read and write sizes.

Experimental Results

One system tested was a parity generator for a 3+3 system of disks; that is, three blocks in a RAID stripe are dedicated to holding data, while the remaining three are dedicated to holding parity information. This configuration would allow up to half of the disks to fail without incurring any data loss. This system was benchmarked against a similar CPU system for calculating parity in the same manner.

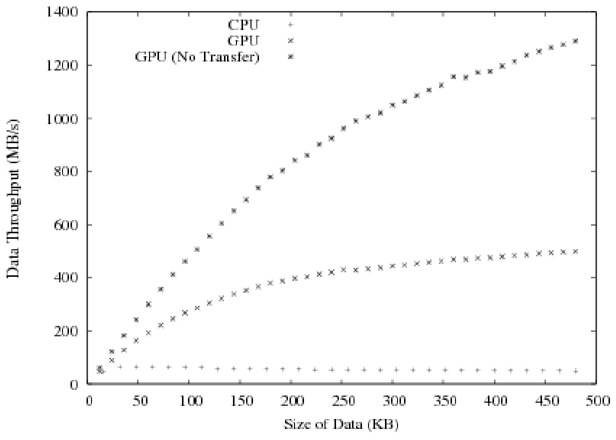

Figure 1: CPU vs. GPU Performance

Figure 1 shows the tests as performed. The first test utilized the CPU (Opteron 246) for doing all work. The next test measured the full time required for contents of main memory to be checksummed with a GPU (GeForce 8800GTX), including the transfer to and from the graphics card over the PCI Express 1.1 buss. The final test discounted this transfer cost, allowing a view into many scenarios: For example, the GPU can someday have direct access to a fast main memory, or transfer latencies can be hidden via asynchronous DMA to the graphics card as part of the pipeline.

These results are clear: The GPU has a tenfold performance advantage for large writes when using PCI-Express transfers. If the cost of these transfers were to be hidden or eliminated, a 30-fold performance increase can be obtained.

Another experiment performed compares the GPU checksum component to the speed of disks for same-sized writes. As mentioned previously, a goal is to have the GPU portion of the pipeline be slower than the disk stage, ensuring throughput is maintained for a steady stream of writes. This experiment's timings include the transfer cost over the PCI Express bus. The experiment compares the time for generating the checksums with the time required to write the data and its checksum information to disks (10,000 RPM Western Digital Raptors).

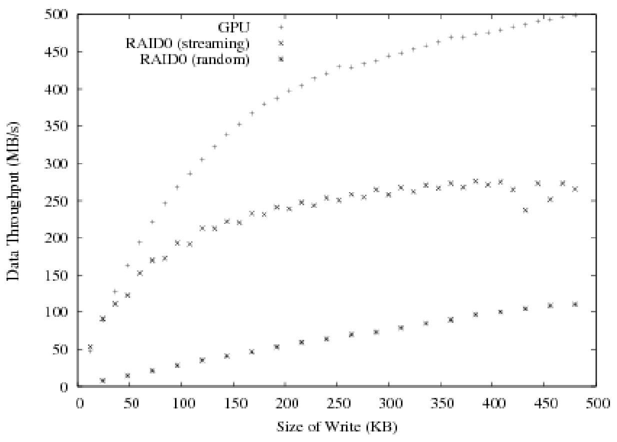

Figure 2: CPU vs. GPU Performance

The results, shown in Figure 2, show the results. The GPU outperforms the disks by a factor of two for very large writes, and is slower than the disk set only for small write sizes for streaming writes to the disk array.

Conclusion

For modestly-sized arrays that need high reliability, which can be found in an embedded environment, GPUs have shown themselves to be both faster than CPUs and the disks which they utilize. This is due to the unique memory architecture and highly parallel design. While this alone is sufficient, there are other conclusions to draw from this work.

RAID solutions, due to their design, are usually only applicable to the problem of managing disks. Embedded systems often see a need to reduce the number of components in the system in order to increase the overall mean time to failure of the system. Using GPUs for parity information allows for the reuse of a component in the system if the I/O workload is sporadic; when not performing checksum tasks, the GPU can be used for performing other tasks. Using a GPU in this context increases the performance of a system without necessarily increasing the number of components in the system, as GPUs are already being used in embedded systems.

References

[1] P. M. Chen, E. K. Lee, G. A. Gibson, R. H. Katz, and D. A. Patterson, “RAID: High-performance, reliable secondary storage,” ACM Computing Surveys, 26(2): 145-185, 1994.

[2] Accusys, “Accusys demonstrates triple parity RAID technology on the New Generation of SATA II RAID controllers at Computex Show,” 2005.

[3] I. S. Reed and G. Solomon, “Polynomial codes over certain finite fields,” *Journal of the Society for Industrial and Applied Mathematics*, 8(2):300-304, 1960.