EGE: A New Energy-Aware GPU based Erasure Coding Scheduler for Cloud Storage Systems

Mehdi Pirahandeh
Department of Electronic Engineering, Inha University
Incheon, South Korea
mehdi@iesl.inha.ac.kr

Deok-Hwan Kim*
Department of Electronic Engineering, Inha University
Incheon, South Korea
deokhwan@inha.ac.kr

***Abstract*—** Redundant array of inexpensive disks (RAID) based storage systems performance is limited to the sequential nature of the central processing unit (CPU), and they consume high amounts of energy because they need erasure coding for striping data and parity into storage devices. This paper proposed an energy-aware GPU based scheduling. The proposed scheduler differs from existing RAID in that it can reduce the number of CPU cycles and coding time. The proposed system generates parity by using a GPU, stripes parity at the initiator server and stripes data at the target server. We also apply an energy-aware scheduling scheme based on solid state disk–based data storage and hard disk drive–based parity storage. Experimental results show that the energy consumption by GPU-RAID is 45% less than Linux-RAID.

Keywords— cloud storage system, energy-aware scheduling, hybrid storage, solid state disk (SSD)

I. INTRODUCTION

Recent storage trends report the necessity for researchers and solid state disk (SSD) manufacturers to promise breakthroughs in terms of energy consumption, reliability, and performance in cloud storage systems [1–10]. Many methods have been proposed to decrease energy consumption for hard disk drives (HDDs), SSDs and hybrid cloud storage systems [3-6]. These methods include profiling energy cost [6], the graphics processor unit (GPU)-based file system (GPUfs) [4], energy-saving redundant array of inexpensive disks (RAID) configuration [5], and power mode scheduling [8]. In a cloud storage system, RAID has high energy consumption and needs to apply erasure coding for striping data and parity into storage. This property empowers the storage system to resist disk failure. However, performance of conventional RAID systems is limited to the sequential nature of the central processing unit (CPU) in that their input/output (IO) operations are accompanied with coding and scheduling using the CPU.

This paper proposes energy-aware GPU-RAID scheduling in a hybrid cloud storage system for big-data processing. The proposed hybrid cloud storage system consists of an initiator server, a target storage server and 10G high-speed storage networks. The proposed system stripes data at the target server, generates parity by using the GPU and stripes parity at the initiator server. We call this GPU-RAID. The proposed GPU-RAID differs from traditional RAID in that it can reduce the number of CPU cycles and coding time. It also resolves the parity and data striping bottleneck due to the sequential nature of the CPU, and then will increase IO performance. In addition, a new energy-aware scheduling scheme is applied at initiator and target servers. Section 2 presents the background and motivation. Section 3 describes the proposed techniques in detail. The experimental environment and results are described in Section 4. Section 5 reports conclusions.

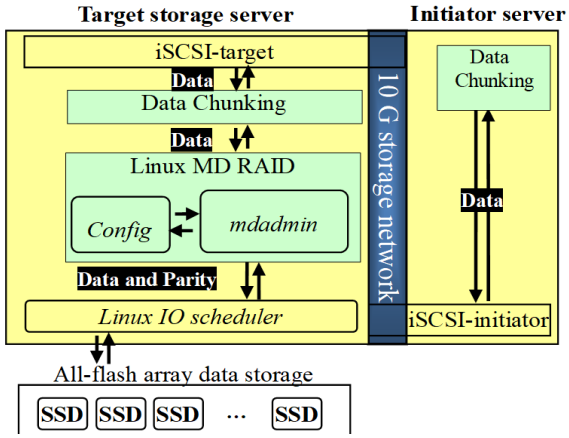

Fig. 1. The architecture of traditional cloud storage system

II. BACKGROUND AND MOTIVATION

In terms of energy consumption, IO performance in a cloud storage system is related to the number of CPU cycles for generating parity and the number of IO operations per second. Erasure coding techniques provide resistance against data failure in cloud storage. However, traditional erasure coding performance is degraded due to the sequential nature of the CPU and inflexible power scheduling schemes. Consequentially, it costs a lot of energy and time to encode and decode data.

In terms of energy efficiency, various IO scheduling schemes [3, 5, 8] have been presented, since SSDs have various power modes, such as *ON*, *OFF*, *sleep*, *active* and *idle*, whereas HDDs have different power modes, such as *sleep* (the drive is shut down), *standby* (low power mode, the drive is spun down), *active* and *idle* (normal operations).

The total energy cost of CPU-based encoding, $E_{encoding}$, is calculated based on Eq. 1 and Eq. 2, where t_{idle} , t_{coding} , and

t_{active} are denoted as the *idle* time, the time to generate parity, and *read/write* time, respectively. The power consumptions for *coding*, *active* and *idle* modes are denoted as e_{coding}, e_{active}, and e_{idle}. So, the power consumption for each CPU cycle is denoted as $e_{CPU\ cycle}$. The number of disks, threads and CPU cores are denoted as N_{disk}, N_{thread}, and N_{core}.

$$e_{coding} = e_{CPU\ cycle} \times N_{core} \quad (1)$$

$$E_{encoding} = (e_{active}\ t_{active} + e_{idle}\ t_{idle} + e_{coding}\ t_{coding}) \times N_{thread} \times N_{disk} \quad (2)$$

2.1 The traditional cloud storage system

Encoding requests occur via the write operation from the initiator server [9],[10]. The traditional architecture of flash array cloud storage is shown in Fig. 1. At the initiator server, a data chunking method assorts files into chunks. At the target server, Linux multiple devices (MD) RAID stripes data and parity chunks into the flash array storage system. The Linux MD RAID stripes data D into k data elements D $=\{d_{1,1},..., d_{n,\beta}\}$, and we assume that n is the number of disks and β $(= k/n)$ is the number of stripes. A sequence of exclusive or (XOR) operations for generating a parity element is shown as follows:

$$p_1 = d_{1,1} \oplus d_{1,2} \oplus \ldots \oplus d_{1,n}$$

XOR operations are executed among data elements in a stripe. All parity data is generated by repeating $\beta\times(n\text{-}1)$ times the XORing of the data vector, and each XOR operation must end before executing the next XOR operation. As a result, a total of $\beta\times(n\text{-}1)$ delays occur due to the sequential nature of the CPU.

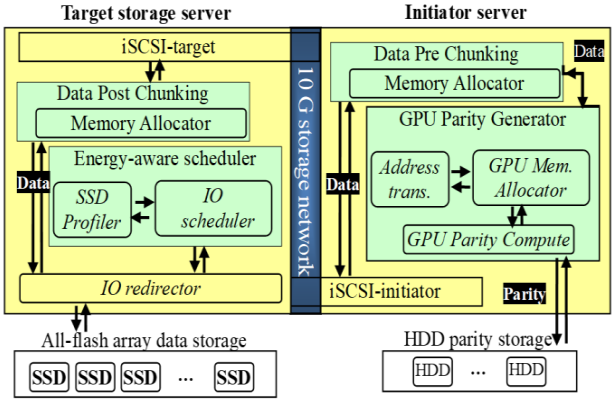

Fig. 2. The architecture of the proposed storage system

III. The Proposed Storage System

Fig. 2 shows the proposed architecture of the energy-aware GPU-based flash array cloud storage. The proposed hybrid cloud storage architecture is composed of an initiator server, a target storage server and 10G high-speed storage networks. Encoding is the procedure of creating parity data from the actual data and then striping data and parity onto data disks [11]-[14]. The proposed system provides energy-aware GPU-RAID scheduling at the target and initiator servers. The proposed method uses a GPU to generate parity and stripe them at the initiator server and only stripes data at the target server. This reduces coding time and energy consumption and also reduces the storage network burden. For both data and parity storage, we apply an energy-aware scheduling scheme. At the initiator server, a data pre-chunking module assorts a file into chunks, and then a GPU parity generator creates parity data by using a GPU parity computing module, a memory allocator and an address translation function. The proposed GPU parity generator needs address translation from main memory to GPU memory. Finally, parity is written to HDD storage at the initiator server. To reduce power consumption, the proposed scheduler switches HDD power modes between active and standby. In the target storage server, a data post-chunking module assorts chunks with the same size as defined by the pre-chunking module. The proposed scheduler switches power modes of flash array–based data storage by using SSD profiles and an IO scheduler function. The energy-aware scheduler module is used to reduce power consumption by switching SSD power modes between active and idle. Finally, an IO redirector uses an energy-aware scheduler scheme to write data to flash array storage. The pseudo-code of the encoding scheduler at initiator and target server is shown in Table 1. In line (1), data block is loaded into the CPU memory. In line (2), encoding scheduler at initiator server invokes GPU_parity_compute function to generate parity block from data block. In line (3), to write parity block into HDD storages, it needs to switch HDDs from standby to active power mode. In line (4), initiator server scheduler stripes parity blocks into parity HDDs. Finally, to reduce the power consumption, the scheduler switches HDDs from active to standby power mode in line (5). In line (6), encoding scheduler at target server generates SSD profiler consisting of n SSDs. In line (7), data block is loaded from iSCSI target to the CPU memory at target server. In line (8), to stripes data blocks data block into SSDs, it is required to change SSDs power mode from idle to active mode. In line (9), encoding scheduler at target server stripes data blocks into n data SSDs. Finally, in the last line, to reduce the power consumption, the scheduler change SSDs power modes from active to idle.

TABLE 1. THE PSEUDO-CODE OF THE ENCODING SCHEDULER

Encoding scheduler (d[*nw*], *n, w, m*)
Input: d [*nw*]: data blocks. *n:* the number of data disks. *w:* word size. *m:* the number of parity disks.
*/ Data initialization at initiator server */ **1.** **read** data blocks from host and **allocate** it into *d*[*nw*] *array*. */ Parity creation using GPU based parity scheduler at initiator */ **2.** parity = **GPU_*parity_compute*** (*d*[*nw*], *n, w,m*). **3.** **switch** HDD power mode from "*standby*" to "active" mode. **4.** **write** parity blocks from CPU memory into Parity HDDs. **5.** **switch** HDD power mode from "*active*" to "*standby*" mode. */ Data stripping at target server */ **6.** **generate** SSD profiler consisting of *n* SSDs. **7.** **read** data blocks from *iSCSI target* and **allocate** it into CPU memory *d*[*nw*]. **8.** **switch** SSDs power mode from "*idle*" to "*active*" mode. **9.** **write** data blocks from CPU memory into *n* SSDs. **10.** **switch** SSDs power mode from "*active*" to "*idle*" mode.

TABLE 3. SSD AND HDD SPECIFICATIONS

	Samsung SSD 830	Samsung SSD 840	WD Caviar HDD WD10EZEX
e_{active}	0.15 watts	0.07 watts	9 watts
e_{idle}	0.08 watts	0.05 watts	-
$e_{standby}$	-	-	0.09 watts
Capacity	64 GB	128 GB	1 TB
Max read	520 MB/s	530 MB/s	150 MB/s

IV. THE PRIMARLY RESULTS

The proposed architecture for the energy-aware GPU-based flash array cloud storage is implemented under the CentOS operating system using open source Jerasure software [15], [16]. SSD and HDD specifications for hybrid cloud storage are shown in Table 3. To enable the proposed energy-aware features, the hdparm tool was used from the CentOS 6.2 operation system. GPU-RAID 6 is used as the proposed RAID 6 configuration, whereas Linux-RAID 6 is used as the traditional RAID 6 configuration.

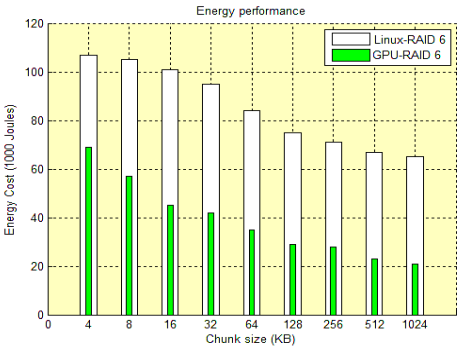

Fig. 3.The energy consumption results

Fig. 3 shows the energy consumption of Linux-RAID 6 and GPU-RAID 6. The average energy consumption of GPU-RAID 6 is improved by 45% compared to Linux-RAID 6. This is because t_{active} and t_{coding} under GPU-RAID 6 decreases more sharply than Linux–RAID 6 as chunk size increases.

V. CONCLUSION

The proposed system provides energy-aware scheduling for GPU-RAID at the target and initiator servers with higher input/output performance. GPU-RAID differs from existing RAID in that it reduces the number of CPU cycles and coding time by generating parity using the GPU, striping them at the initiator server, and then striping data at the target server. This will increase parallelism and resolve any CPU bottleneck from parity generation.

ACKNOWLEDGMENT

This work was supported by the Industrial Technology Innovation Program funded By the Ministry of Trade, industry & Energy (MI, Korea) [10073154, Development of human-friendly human-robot interaction technologies using human internal emotional states] and in part by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education(2015R1D1A1A01061112) and in part by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education(2010-0020163).

REFERENCES

[1]. J.S. Plank, “XOR’s Lower Bounds and MDS Codes for Storage,” IEEE Information Theory Workshop, Brazil, pp. 529-551, October 2011.

[2]. L.M. Grupp, J.D, Davis and S. Swanson, “The Bleak Future of NAND Flash Memory, ” FAST Conference in Storage systems. San Jose, USA, February 2012.

[3]. Y. Won, et al, “Energy-aware disk scheduling for soft real-time I/O requests,” Springer. Multimedia System. vol. 13, pp.409–428, 2007.

[4]. Silberstein, Mark and et al , “GPUfs: integrating file systems with GPUs,” Proc. of the 18th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '13), ACM, New York, NY, USA, pp. 485-498, 2013.

[5]. T. Xie, "SEA: A Striping-Based Energy-Aware Strategy for Data Placement in RAID-Structured Storage Systems," IEEE Transactions on Computers, vol.57, no.6, pp.748-761,June,2008.

[6]. Li, Yan, et al. "Energy-Aware Storage,” USENIX, FAST Conference in Storage, Feb. 2013.

[7]. M. Pirahandeh, D.H. Kim, “Adopted erasure code for SSD based RAID-6 System,” ITC-CSCC. Sapporo, Japan, pp. 81–85, July 2012.

[8]. M. Pirahandeh and D.H. Kim, “Reliable Energy-Aware SSD based RAID-6 System,” FAST Conference in Storage systems. San Jose, USA, February 2012.

[9]. K.M.Greenan, X. Li, J.J.Wylie, "Flat XOR-based erasure codes in storage systems: Constructions, efficient recovery, and tradeoffs," IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST), pp.1-14, May 2010.

[10]. M. Blaum, J.Brady, J. Bruck and J. Menon, “EVENODD: An Efficient Scheme for Tolerating Double Disk Failure in RAID Architectures,” IEEE Trans. on computer, vol. 44, no.2, February 1995.

[11]. A.Kadav, M. Balakrishnan, V. Prabhakaran, D. Malkhi,” Differential RAID: rethinking RAID for SSD reliability,” ACM Trans. Operating Systems Review, pp. 55-59. 2010

[12]. H. J. Lee, K.H. Lee and S.H. Noh, “Augmenting RAID with an SSD for Energy Relief,” FAST Conference in Storage systems, San Jose, USA, February 2012.

[13]. D. Kenchammana, D. He and J. L. Hafer, “REO: A generic RAID Engine and Optimizer,” Proc. 4th USENIX Conference on File and Storage Technologies(FAST ’07). San Francisco, USA, pp. 261-276, December 2005.

[14]. J.L. Hafner, V. Deenadhayalan, K. K. Rao, and A. Tomlin, “Matrix methods for lost data reconstruction in erasure codes,” Proc. of the 4th USENIX Conference on File and Storage Technologies (FAST'05), San Francisco, USA, pp. 183-196, December 2005.

[15]. J. Luo; L.Xu and J.S Plank, "An efficient XOR-scheduling algorithm for erasure codes encoding," IEEE/IFIP International Conference on Dependable Systems & Networks (DSN '09), pp.504-513, June 2009.

[16]. J. S. Plank, “The RAID-6 Liberation Codes,” Proc. of the 6th USENIX Conference on File and Storage Technologies (FAST ’08), San Jose, USA, pp. 97-110, February 2008.