Cluster Computing (2019) 22:445–454
https://doi.org/10.1007/s10586-018-2842-3

Energy-aware RAID scheduling methods in distributed storage applications

Mehdi Pirahandeh[1] · Deok-Hwan Kim[1]

Received: 11 December 2017 / Accepted: 20 August 2018 / Published online: 25 August 2018

Abstract
A dynamic power management (DPM) method makes power-mode-related decisions based on the information available at runtime (online) or before (offline). This paper proposes energy-aware RAID scheduling methods to reduce energy consumption for distributed storage applications using both online and offline DPM strategies. The proposed energy-aware redundant array of inexpensive disk (RAID) scheduling method differs from the existing RAIDs method in that it separately strips data and parity blocks and switches the power mode from active to idle or standby while storage applications are not performing any physical I/O operations. Traditional DPM schedulers are applied for a single logical I/O operation or multiple of the logical I/O operations (workload), whereas the proposed DPM scheduler is applied for physical I/O operations. Experimental results show that the proposed storage application reduces average energy consumption by 26–36% compared to existing storage applications.

Keywords Storage application · Energy consumption · RAID · Fault tolerance · Erasure codes

1 Introduction

Recent trends in storage application development show the necessity for researchers to produce breakthroughs in terms of energy consumption in distributed storage systems [1–7]. In distributed storage applications, a redundant array of inexpensive disks (RAID) has high energy consumption for striping data and parity blocks into storage nodes[1, 3–6]. RAID empowers the storage application to resist node failures. First, RAID splits data chunks into data blocks at a file-system level. Then, RAID generates redundant data blocks from the total number of data blocks. Finally, RAID performs simultaneous physical writes at a kernel level. In Table 1, the specifications of the proposed storage application are compared to those of existing storage applications in terms of applied erasure codes, storage network, disk array, and type of I/O level and process for dynamic power management (DPM). A DPM scheduler enables efficient usage of storage devices by switching the power mode to a lower power-mode while a storage system does not perform any I/O operation. In the works of Son et al. [9–11], Zhu et al. [12], Pirahandeh et al. [13], and Yin et al. [14], DPM schedulers were applied for single logical I/O operation or multiple logical I/O operations (workload), whereas the proposed DPM scheduler is applied for physical I/O operations. Irani et al. [1] described two types of DPM: off-line and on-line. The off-line DPM knows the length of the idle period of the storage nodes in advance, whereas the online DPM does not. In terms of the DPM process type, the DPM schedulers proposed by Son et al. [9–11], Yin et al. [14], and Pirahandeh et al. [13] used on-line processes, and the DPM scheduler proposed by Zhu et al. [12] used an on-line or off-line process, whereas the proposed DPM scheduler uses both on-line and off-line processes. This hybrid storage application is constructed through a combination of SSDs and HDDs in a node array. In the proposed distributed storage application, the encoder pipelines the physical data blocks using an on-line DPM process to exploit the high transfer rate, while parity blocks will be pipelined later using an off-line DPM process. This will enable the proposed

✉ Deok-Hwan Kim
deokhwan@inha.ac.kr

Mehdi Pirahandeh
mehdi@inha.ac.kr

[1] Department of Electronic Engineering, Inha University, 253 Yonghyun-dong, Nam-gu, Incheon 402-751, South Korea

Table 1 Comparison of the developed scheduling methods for distributed storage applications

Storage applications (SA)	Erasure codes	Storage network	Disk array	DPM[a] I/O level	DPM process
SA1: Son et al. [9–11]	RS, SPC	SAN[b]	ALL HDD	Logical	On-line
SA2: Zhu et al. [12]	RS, SPC	SAN	ALL HDD	Logical	On-line or Off-line
SA3: this research	MDS[c]	SAN	Hybrid	Physical	On-line and Off-line
SA4: Pirahandeh et al. [4]	SPC	DAS[d]	ALL SSD	Physical	On-line
SA5: Yin et al. [14]	Mirroring	DAS	Hybrid[e]	Workload	On-line

[a]DPM denotes dynamic power management

[b]SAN denotes storage area network

[c]MDS denotes minimum-distance separable erasure codes such as Reed–Solomon (RS), EVENODD [8], and Single parity check (SPC) codes

[d]DAS denotes directed attached storage

[e]DPFS denotes the distrusted parallel file system

distributed storage application to perform separate DPM method for data nodes using SSDs and parity nodes using HDDs.

In this paper, we propose power mode scheduling to reduce energy consumption for distributed storage applications. The platform of the proposed distributed storage application consists of an initiator server, a target storage server, and iSCSI high-speed storage networks. The proposed application generates parity by using the CPU and it stripes parity and data blocks at the target server. An energy-aware encoding scheduler allows storage applications to switch the power mode of the storage nodes from *active* to *idle* or *standby* while storage applications are not performing any I/O operations. As a result, the average encoding energy consumption of the proposed storage application (SA3) is lower that of SA1, SA2, SA4, and SA5 traditional storage systems, respectively. Moreover, the proposed storage application (SA3) have a higher encoding and decoding performance that of SA1, SA2, SA4, and SA5 traditional storage systems, respectively. Section 2 presents the background and motivation. Section 3 describes the proposed techniques in detail. The experimental environment and results are described in Sect. 4. Section 5 reports the conclusions.

2 Background and motivation

Distributed RAID storage applications designed to ensure reliability generate coding data nodes called "parity nodes" using various erasure coding schemes to provide node fault tolerance. There are various types of erasure codes, such as Reed–Solomon (RS) codes and single parity check (SPC) codes, defined by their coding scheme. In distributed RAID storage applications, the authors assume that the workload D consists k data chunks, $D = \{D_0, \ldots, D_{k-1}\}$. The ith data chunk consists of nw data blocks, $D_i = \{d_0, \ldots, d_{nw-1}\}$, and the ith parity chunk consists of parity blocks $P_i = \{p_0, \ldots, p_{mw-1}\}$, where mw denotes the number of stripes. Table 2 shows the pseudo-code of the parity generator using Cauchy RS codes.

In erasure codes, we assume that $(Data|Coding) = (D|P) = D \times G = D \times (I|X)$, where the generator matrix $G = (I|X)$ consists of the identity matrix I with $wn \times wn$ code words and coding matrix $X = \{x_0, \ldots, x_{nw-1}, \ldots, x_{mw-1,nw-1}\}$ with $m \times n$ coding blocks. XOR operations are executed between data code words in a stripe. In storage applications using CPU cores (SA1, SA2, SA4, and SA5), the sequences of XOR operations for generating a parity code word P_i at the ith stripe as follows:

$$P_i = \sum_{j=0}^{nw-1} P_i \oplus (d_j \times x_{i,j}) = (d_0 \times x_{i,0}) \oplus, \ldots, \oplus \left(d_{nw-1} \times x_{i,nw-1}\right). \tag{1}$$

2.1 Traditional energy-aware I/O scheduling method in distributed storage applications

Figure 1 shows a sequence diagram of the traditional energy-aware I/O scheduling method for distributed storage applications [15, 16] in traditional RAID 5 storage applications using two slave data nodes (two SSDs) and one slave parity node (one HDD). The logical write operation delay in the traditional RAID level 5, LW, is as follows:

$$\begin{aligned} LW_{Online|Off-line} &= 3 \times Spinup + 2 \times Ch + 6 \times ABC + 4 \times ABR \\ &\quad + 2 \times XOR + 6 \times SR + 6 \times PW + 3 \times Spindown. \end{aligned} \tag{2}$$

As noted in Table 3, *Spin up*, *Ch*, *ABC*, *ABR*, *XOR*, *SR*, *PW*, and *Spin down* denote the delay due to switching the disk power mode from *standby*/*idle* to *active*, splitting data to chunks, allocating a block to main

Table 2 The pseudo-code of the parity generator using Cauchy RS codes [17]

Parity compute$(d[nw], n, w, m)$
Input
$d[nw]$: a data chunk
n: the number of data disks
w: the number of code words in a data block
m: the number of parity disks
Output
$p[mw]$: m parity block

```
1.  for (j = 0; j < nw − 1; j++) do {
2.    for (i = 0; i < mw − 1; i++) {
3.        p_i = p_i ⊕ (d_j × x_{i,j})
4.    } endfor
5.  } endfor
6.  return P as m parity blocks;
```

memory, allocating a block from/to the register, performing the XOR operation using a parity computing scheduler, reading a block from the main memory by a system kernel, physically writing a block to the disk using a RAID device driver , and switching the disk power mode from *active* to *standby/idle*, respectively. The *LW* latency can be defined as the elapsed time (t_{wc}) between the command of an iSCSI initiator and the response to completion of the command because the next command can be transmitted after the write completion (*WC*) response of a target storage server. Finally, a logical write can be processed *on-line* or *off-line* regardless of the type of storage device in node array.

3 Proposed energy-aware distributed storage application

The proposed storage application(SA3) is composed of an initiator server, a target storage server, and iSCSI high-speed storage networks. The aim is to reduce energy consumption for the distributed storage system. At the initiator server, a data chunking module assorts a file into multiple

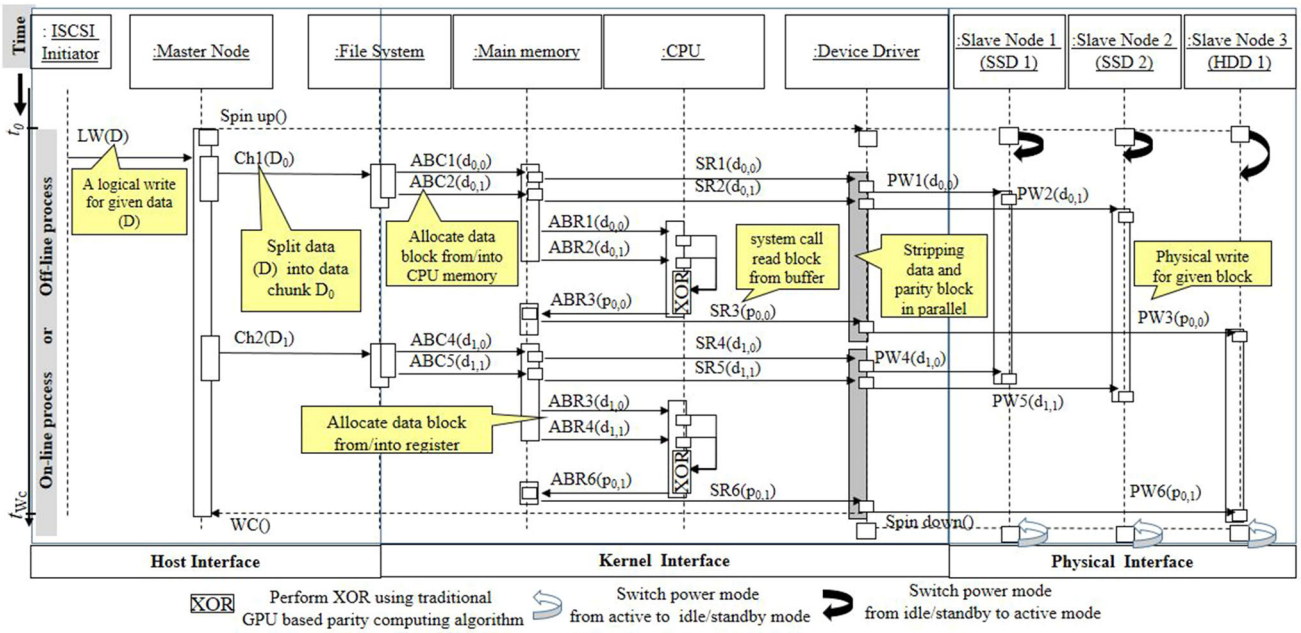


Fig. 1 DPM-based logical write scheduling method in a traditional distributed storage application ($k = 2, n = 2, m = 1, w = 2$)

Table 3 Notation list

Symbol	Explanation
LW	The logical write operation delay in the traditional RAID level 5
Spin up	The delay due to switching the disk power mode from *standby/idle* to *active*
Spin down	The delay due to switching the disk power mode from *active* to *standby/idle*
ABC	The delay due to allocating a block to main memory
ABR	The delay due to allocating a block from/to the register
XOR	The delay due to performing the XOR operation using a parity computing scheduler
SR	The delay due to reading a block from the main memory by a system kernel
PW	The delay due to physically writing a block to the disk using a RAID device driver

data chunks, and these data chunks are transferred to the target server using the iSCSI protocol via the storage network. At the target storage server, a data chunk is assorted into multiple data blocks which are loaded into the CPU main memory. The parity generator creates parity blocks using a parity compute function and a memory allocator function. By allocating the parity generator in the target server, we can fairly compare the I/O performance of the proposed storage application with that of traditional storage applications. This is because, in all traditional storage applications (SA1, SA2, SA4, SA5), a parity generator is allocated in the same storage server. The proposed energy-aware scheduler switches the power modes of the data disks using a power-mode-switching function and a disk (SSD and HDD) energy profiler. The energy-aware scheduler module is used to reduce power consumption by switching SSD power modes between active and idle modes or by switching the HDD power mode between active and standby modes. Finally, an I/O redirector uses an energy-aware scheduler scheme to write data and parity blocks to hybrid storage.

3.1 Energy-aware write scheduler at the initiator and the target server

Figure 2 shows the proposed sequence diagram of the DPM-based logical write RAID 5 scheduling method in the proposed storage application(SA3) using two data nodes (two SSDs) and one parity node (one HDD). The logical write operation delay in the proposed distributed application, LW, is defined as:

$$LW_{On-line} = 2 \times Spinup + 2 \times Ch + 4 \times ABC + 4 \times SR + 4 \times PW + 2 \times Spindown. \quad (3)$$

$$LW_{Off-line} = Spinup + 4 \times ABR + 2 \times XOR + 2 \times ABC + 2 \times SR + 2 \times PW + Spindown. \quad (4)$$

$LW_{On-line}$ latency can be defined as the elapsed time (t_{DWC}) between a command of the iSCSI initiator and the data completion response. $LW_{Off-line}$ latency can be defined as the elapsed time ($t_{PWC} - t_{DWC}$) between a command of a data completion response and the parity completion response, because the next command can be transmitted after the data write completion (DWC) response of a target storage server.

3.2 Energy-aware distributed RAID decoder

Table 4 shows the distributed RAID decoder, which can recover m disk failures. In lines (1–3), the distributed RAID decoder spins up data disks located in data nodes, reads a data chunk from n data disks, and finally spins down the data disks. In line (4), when the number of data disk failures $n - \Phi(G)$ exceed m, it is a disaster failure that cannot be recovered from. In lines (5–13), when $n - \Phi(G)$ disk failures occur, the algorithm recovers up to mw failed data code words. In line (6), the distributed RAID decoder spins up parity disks located in parity nodes. In line (7), the distributed RAID decoder allocates the corresponding survival data and parity code words to ds. In line (8), the distributed RAID decoder spins down parity disks located in the parity nodes. The RAID decoder loads the inverted coding code words B^{-1}, where columns corresponding to failed data code words are deleted in lines (9–10). By using the parity computation algorithm, the distributed RAID decoder recovers the failed data code words by performing matrix multiplication between inverted coding code words B^{-1} and survival data and parity code words ds in line (11). The survival data code words and recovered data code words d' are merged to recover $d[nw]$ in line (12). Finally, recovered data chunks $d[nw]$ are read from main memory into the host in line (13).

3.3 Evaluation of energy consumption

In RAID storage systems, data should be archived along with parity data across many storage devices, so that if a storage device fails, here is still sufficient data to repair the disks [2]. It is important to develop an energy-saving storage system based on the RAID structure. Irani et al. [1] elucidated that disk power management aims to save energy by switching disks to lower- power modes whenever possible without adversely affecting performance. As soon as I/O operations are completed, DPM decides if the disk should stay in the idle mode. However, the idle period needs to justify the cost of spinning the disk up and down. If the encoding time is greater than the time needed to spinn the disk up or down, then it is more beneficial to spin down the disk to a lower-power standby mode. Therefore, the disk is spun down immediately after the current request is serviced and spun up to an active mode just in time for the next request. Otherwise it is better to stay in the idle mode after the current request completes. Irani et al. [1] described that the DPM strategy must make decisions with only partial information. An online power management method must make decisions about the expenditure of resources before all the input to the system is available. Therefore, an online DPM does not know the length of an idle period until the moment that it ends. However, an off-line DPM knows the length of the idle period in advance.

Disk power management We proposed a new DPM for encoding and decoding physical data blocks. The proposed

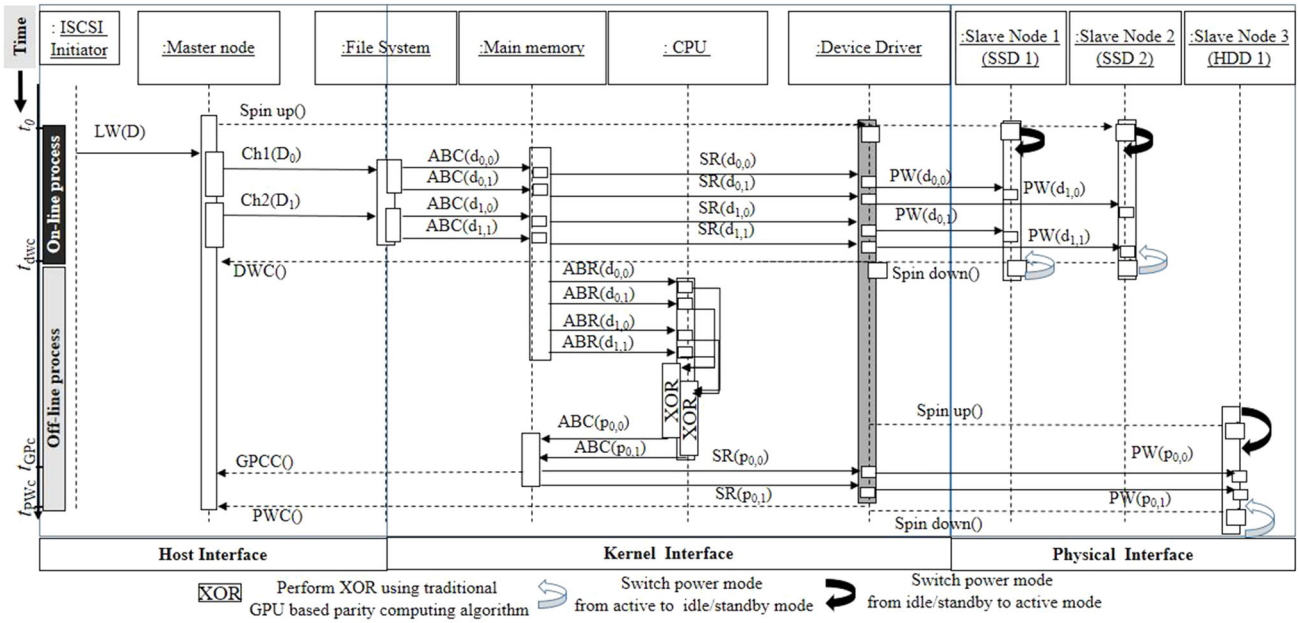


Fig. 2 Sequence diagram of the DPM-based logical write scheduling method in the proposed storage application ($k = 2, n = 2, m = 1, w = 2$)

Table 4 Distributed RAID decoder

Distributed RAID decoder (^1G,$^2\Phi(G)$,^3n,^4m).
1. Spinup data disks located in data nodes.
2. Read a data chunk from n data disks and transfer it into host.
3. Spindown data disks located in data nodes.
4. if $(m < (n - \Phi(G))$ return; // it is called disaster failure and it can not be recovered.
5. if $(1 \leq \Phi(G) \leq m)$ { */ Recovering $n - \Phi(G)$ failed disks.*/
6. Spinup parity disks located in parity nodes.
7. Allocate the corresponding survival data and parity code words to ds.
8. Spindown parity disks located in parity nodes.
9. Coding code words B is generated by deleting the columns of original coding code words X corresponding to failed data code words.
10. load B^{-1} inverted coding code words.
11. $d' =$ parity compute$(ds[nw], \Phi(G), w, n - \Phi(G))$.
12. Merge survival data code words and d' into a recovered data chunk $d[nw]$.
13. Read a data block from main memory and transfer it into the host. }

^{1}G contains the group of survival data disks. $^2\Phi(G)$ denotes the number of survival data disks.
^3n is the number of data disks, ^4m is the number of parity disks.

DPM enables on-line DPM processing for data SSDs and off-line DPM processing for parity HDDs. The total energy cost of CPU-based encoding, $E(t)$, is calculated based on:

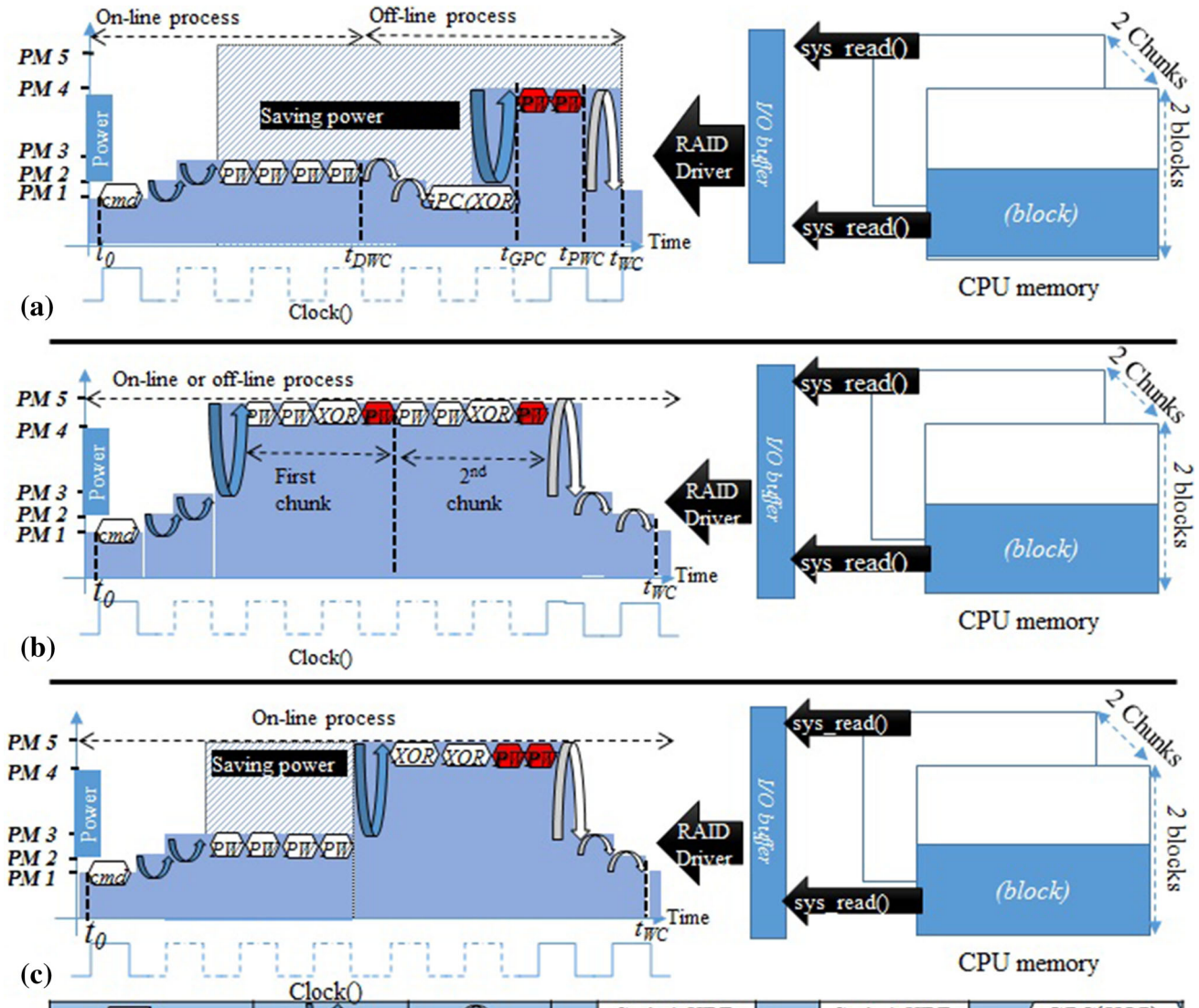
$$e_{coding} = e_{CPUcycle} \times N_{core}. \tag{5}$$

$$E(t) = \sum_{i=1}^{m+n} e_{active-Disk_i}(t_{active-Disk_i} - T_{active-Disk_i}) + C_{active-Disk_i} + e_{idle|standby-Disk_i} t_{idle|standby-Disk_i} + e_{coding} t_{coding}. \tag{6}$$

$$Thr = \frac{D_{size}}{t_{SSD-active} + t_{HDD-active} + t_{coding} + t_{HDD-standby} + t_{SSD-idle}}. \tag{7}$$

where $t_{idle|standby}$, t_{coding} and t_{active} are denoted as the idle/standby time, the time to generate parity, and read/write time, respectively. The power consumptions for coding, active, and idle/standby modes are denoted as e_{coding}, e_{active}, and $e_{idle|standby}$, respectively, and the power consumption for each CPU cycle is denoted as $e_{CPUcycle}$. The numbers of data disks, coding disks, and CPU cores are denoted as n, m, and N_{core}, respectively.

In addition, T_{active} is the time required to spin down a disk from active to idle/standby mode and C_{active} is the energy required to spin up a disk from idle/standby to active mode. The throughput of a storage system Thr is a ratio of the data size D_{size} to the encoding time at the target



cmd			Switch HDD power mode from standby to active (Spin up)	Switch HDD power mode from active to standby (Spin down)	GPC(XOR)
Logical read/write Command	Switch SSD power mode from idle to active	Switch SSD power mode from active to idle			Perform XOR operations among multiple blocks using proposed parity computing
PW	PR	PW	PR		XOR
A physical data block write operation	A physical data block read operation	A physical parity block write operation	A physical parity block read operation	A failed physical data block read operation	Perform block based XOR operations in CPU

RAID Power Mode (PM 1)	RAID Power Mode (PM 2)	RAID Power Mode (PM 3)	RAID Power Mode (PM 4)	RAID Power Mode (PM 5)
All disks in idle|standby mode	*SSD 1 in active mode*	*SSD 1 and SSD 2 in active mode*	*HDD 1 in active mode*	*All disks in active mode*

Fig. 3 Examples of three RAID 5 logical write (encoding) power management models. **a** The proposed model, **b** the naive model, and **c** Pirahandeh et al.'s model ($k = 2$, $w = 2$, $n = 2$, m=1)

server, as shown in Eq. (7). Therefore, the encoding time is calculated from a summation of the *idle* time, the *standby* time, the *coding* time and the *active* time.

Energy-aware model analysis Three RAID power management models are used to measure the energy flow, as shown in Figs. 3 and 4. The SSDs have the power modes *ON*, *OFF*, *active* and *idle*, whereas the HDDs have the power modes *sleep* (the drive is shut down), *standby*, (a low-power mode where the drive is spun down,) and *active* (normal operations). The RAID power-management equipments enables power mode switching when the encoder or decoder reads data code words from storage devices, t_{read}, parity is generated using a CPU, t_{PG}; and code words are written into the storage devices, t_{write}.

(a) Figures 3a and 4a show the energy flow of the encoding and decoding processes, respectively, using the proposed RAID power management model (SA3). To reduce energy consumption, the proposed model switches disk power from the *idle*/*standby* mode to an *active* mode (spin up) before each read/write operation occurs. The proposed model switches disk power from the *active* mode to the *idle*/*standby* mode (spin down) after each read/write operation is done.

(b) Figures 3b and 4b show the naive RAID power management models proposed by Son et al. [9–11] (SA1) and Zhu et al. [12] (SA2), respectively. The naive model is designed based on the the CPU-based parity generation algorithm. To reduce energy consumption, the naive model switches disk power from the *idle*/*standby* mode to the *active* mode (spin up) before the encoding or decoding process is started. The naive model switches disk power from the *active* mode to the *idle*/*standby* mode (spin down) after the encoding or decoding process is completed.

(c) Figures 3c and 4c show the RAID power management model proposed by Pirahandeh et al. [13]. This model is designed from the the CPU-based parity generation algorithm. To reduce energy consumption, Pirahandeh et al.'s model switches SSD power from the *idle* mode to the *active* mode (spin up) before the encoding process is started. This model switches SSD power from the *active* mode to the *idle* mode (spin down) after the encoding process is

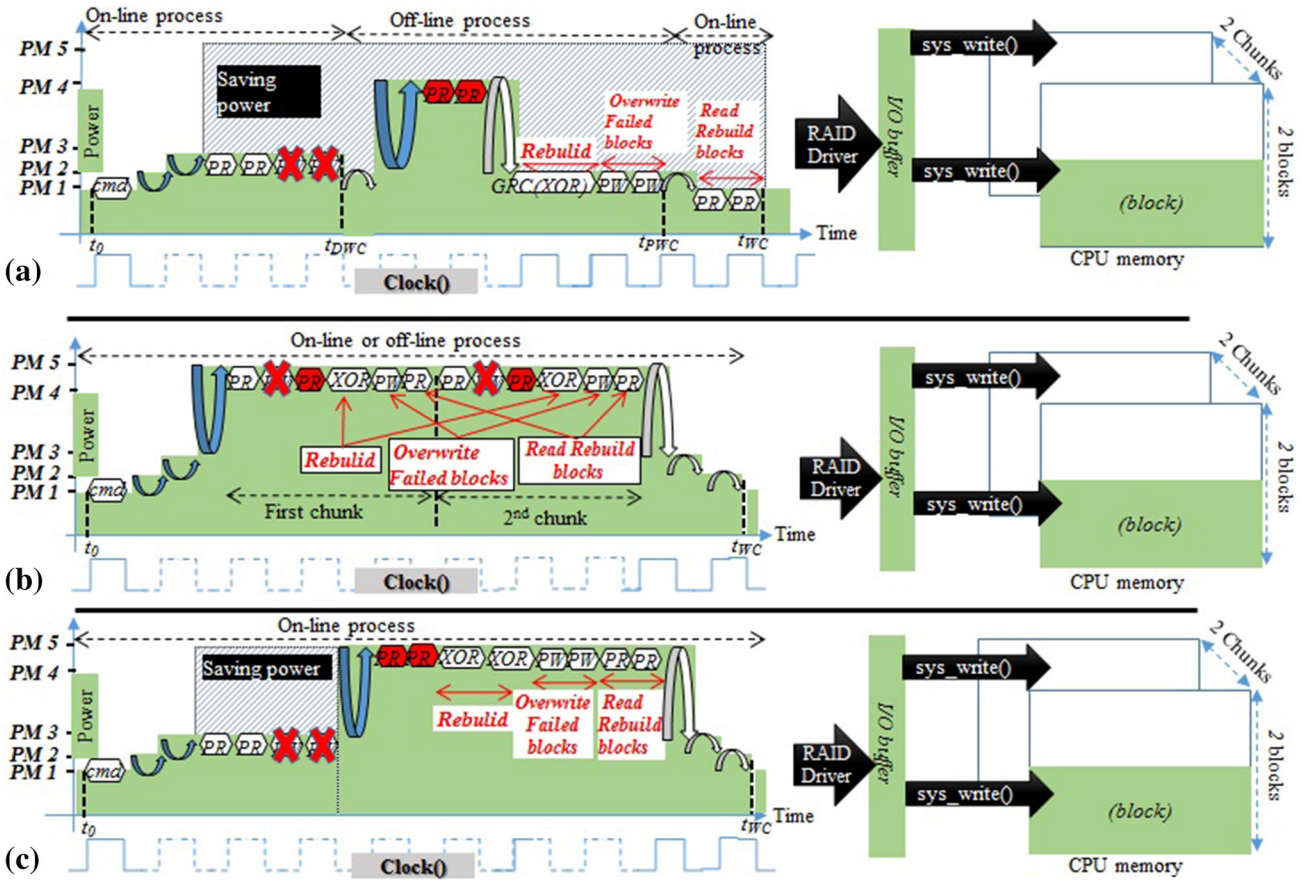


Fig. 4 Examples of three RAID logical degraded read (decoding) power management models when SSD 2 has failed **a** the proposed model, **b** the naive model, and **c** Pirahandeh et al.'s model ($k = 2$, $w = 2$, $n = 2$, m=1)

completed. HDD power will be in the *active* mode only when the encoder perform a parity write operation. During the decoding process, when SSD 2 fails, survival SSD power will be in the *active* mode, and when failed data is recovered, the failed SSD will be in the *active* mode. However, this model switches the HDD power to the *active* mode when the decoder recovers failed data by accessing survival SSD and parity HDD.

The energy-aware scheduling system consists of a disk energy profiler and power-mode-switching functions. The disk energy profiler initializes the power mode profile of the storage devices. The power-mode-switching function will also switch storage device power to the *idle*/*standby* power mode based the proposed RAID power management model, as soon as the I/O operation is completed.

4 Experimental results

The proposed architecture of the energy-aware RAID storage system is implemented. Figure 5a–c show the target/initiator server and the workload specifications. Figure 5d shows the SSD and HDD specifications for the hybrid storage application.

4.1 Energy-aware degraded read performance

Figure 6 shows the decoding performance of various storage applications (SA1, SA2, SA3, SA4 , and SA5) using the SPC, RS, and EVENODD erasure codes for given chunk sizes (4 KB, 16 KB, 64 KB, 256 KB, and 1MB). The average throughput of the one-disk failure recovery using the proposed storage application (SA3) is improved by 67%, 63%, 27%, 49%, 41%, and 18% compared to that of SA1, SA2, SA4, and SA5, respectively. To be more specific, the average decoding performance of the storage applications using the EVENODD code is lightly lower than the performance of the storage applications using RS and SPC code because the encoding performance of the

(a) Initiator server specification

OS	Microsoft Windows 8
CPU MEMORY	DDR3 2 GB
CPU	2.0 GHz

(b) Target server specifications

OS	Cent OS 6.2
CPU MEMORY	DDR3 32 GB
CPU	2.0 GHz

(c) Workloads specifications

TP1	Transaction processing files	5 hours	2 GB
VG1	Video and Game files	1 hour	5 GB

(d)

	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
Max write	320 MB/s	390 MB/s	150 MB/s

Fig. 5 Specifications of the experimental environment

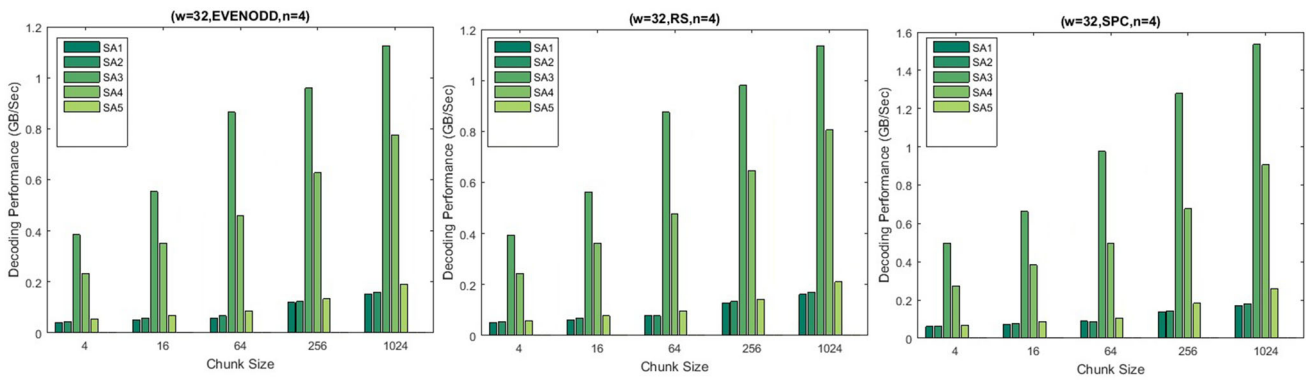


Fig. 6 One-disk failure recovery (decoding) performance of the SPC, RS, and EVENODD erasure codes for the given chunk sizes and storage applications

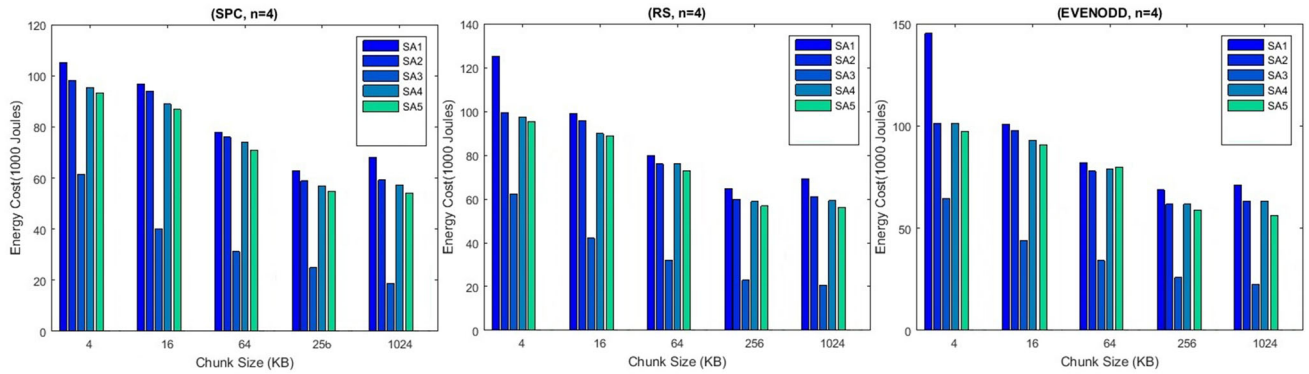


Fig. 7 Energy consumption of the SPC, RS, and EVENODD erasure codes for the given chunk sizes and storage applications

Table 5 Specifications of the experimental DPM method for the proposed storage application compared to those of traditional storage applications

SA	Disk array	$t_{SpinUp-HDDs}$	$t_{SpinUp-SSDs}$	C_{SSDs}	C_{HDDs}
SA1	4 HDDs	64 ms	–	–	2.4 J
SA2	4 HDDs	64 ms	–	–	2.4 J
SA3	3 SSDs + 1 HDD	16.1 ms	0.6 ms	0.3 J	0.6 J
SA4	4 SSDs	–	0.8 ms	0.4 J	-
SA5	3 HDDs + 1 SSD	48.3 ms	0.2 ms	0.1 J	1.8 J

storage applications using EVENODD erasure code has more one bit code words in the coding code words, which affects the number of required XOR operations.

4.2 Energy consumption performance

Figure 7 and Table 5 show the encoding energy consumption of various storage applications (SA1, SA2, SA3, SA4 and SA5) using the SPC, RS, and EVENODD erasure codes for given chunk sizes (4 KB, 16 KB, 64 KB, 256 KB, and 1MB). Equations (5) and (6) are used to measure the encoding energy consumption in these storage applications. The average encoding energy consumption of the proposed storage application (SA3) is improved by 36%, 28%, 26%, and 27% compared to that of SA1, SA2, SA4, and SA5, respectively. In SA3, t_{coding} decreases in proportion as the number of disks for coding increases. This is because t_{active} and t_{coding} under SA3 decrease more sharply than others as chunk size increases. Table 5 shows the specifications of the experimental DPM method for the proposed storage application compared to those of the traditional storage applications.

5 Conclusion

The proposed system provided energy-aware scheduling for distributed RAID storage applications at the target server with higher I/O performance. The proposed distributed RAID scheduling method differs from existing RAID methods in that it reduces the energy consumption by stripping and switching the power modes of data and parity blocks in storage devices in a separated manner. The proposed power management model is applied to both SSD-based data storage and HDD-based parity storage. Experimental results from the proposed storage application (SA3) exhibit decoding performance with throughput that is 67%, 63%, 27%, and 41% higher than that of SA1, SA2, SA4, and SA5, respectively. The proposed storage exhibits 89%, 75%, 71% and 65% faster parity computation than SA1, SA2, SA4, and SA5, respectively.

Acknowledgements This work was supported by Inha University Research Grant.

References

1. Irani, S., Shukla, S., Gupta, R.: Online strategies for dynamic power management in systems with multiple power-saving states. ACM Trans. Embed. Comput. Syst. **2**(3), 325–346 (2003)
2. Plank, JS: Lower bounds and MDS codes for storage. In: IEEE Information Theory Workshop, pp. 503–507 (2011)
3. Xie, T.: SEA: a striping-based energy-aware strategy for data placement in RAID-structured storage systems. IEEE Trans. Comput. **57**(6), 748–761 (2008)
4. Won, Y., et al.: Energy-aware disk scheduling for soft real-time I/O requests. Springer Multimed. Syst. **13**(5–6), 409–428 (2007)
5. Silberstein, M., et al.: GPUfs: integrating file systems with GPUs. In: Proceedings of the 18th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '13), pp. 485–498 (2013)
6. Yan, L., et al.: Energy-aware storage. In: Proceedings of the FAST Conference in Storage (2013)
7. Pirahandeh, M., Kim, D.H.: Adopted erasure code for SSD based RAID-6 System. In: Proceedings of the ITC-CSCC Conference, Japan, pp. 81–85 (2012)
8. Blaum, M., Brady, J., Bruck, J., Menon, J.: EVENODD: an optimal scheme for tolerating double disk failures in RAID architectures. In: Proceedings of the 21st Annual International Symposium on Computer Architecture, pp. 245–254 (1994)
9. Son, S.W., Kandemir ,M., Choudhary, A.: Software-directed disk power management for scientific applications. In: Proceedings of

the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS-05) (2005)
10. Garg, R., Woo, S.S., Kandemir, M., Raghavan, P., Prabhakar, R.: Markov model based disk power management for data intensive workloads. In: Processing of the Cluster Computing and the Grid, pp. 76-83 (2009)
11. Woo, S.S., Kandemir, M.: Runtime system support for software-guided disk power management. In: Processing of the Cluster Computing, pp. 139–148 (2007)
12. Zhu, Q., David, F.M., Devaraj, C.F., Li, Z., Zhou, Y., Cao, P.: Reducing energy consumption of disk storage using power-aware cache management. In: Processing of the IEEE on Software (2004)
13. Pirahandeh, M., Kim, H.: Reliable energy-aware SSD based RAID-6 system. In: Proceedings of the FAST Conference in Storage Systems, San Jose, USA (2012)
14. Yin, S., Li, X., Li, K., Huang, J., Ruan, X., Zhu, X., Cao, W., Qin, X.: REED: a reliable energy-efficient RAID. In: Processing of the 44th International Conference on Parallel Processing (ICPP), pp. 649–658 (2015)
15. Anvin, PH.: The mathematics of RAID-6. http://kernel.org/pub/linux/kernel/people/hpa/raid6.pdf. Accessed 6 Jan 2010
16. Pirahandeh, M., Kim, D.H.: Energy-aware GPU-RAID scheduling for reducing energy consumption in cloud storage systems. Lect. Notes Electr. Eng. **330**(1), 705–711 (2015)
17. Pirahandeh, M., Kim, D.H.: Energy-aware and intelligent storage features for multimedia devices in smart classroom. Multim. Tools Appl. **76**(1), 1139–1157 (2017). https://doi.org/10.1007/s11042-015-3019-1

Mehdi Pirahandeh received an M.S. and Ph.D. from the Inha University in South Korea. His research interests include embedded systems, cloud storage systems, parallel computing and IoT. He is research professor and lecturer at Inha University in Korea.

Deok-Hwan Kim received a M.S. and Ph.D. from the Korea Advanced Institute of Science and Technology. He is a professor at Inha University in Korea. His research interests include embedded systems, storage systems, cloud systems, multimedia systems and brain computer interfaces.