Multimed Tools Appl (2018) 77:4563–4583
DOI 10.1007/s11042-017-4540-1

An extra-parity energy saving data layout for video surveillance

Yu Xiao[1,2] · Zhang Changyou[3] · Xue Yuan[1] · Zhu Hongfei[1] · Li Yuanzhang[1] · Tan Yu-An[1,4]

Received: 31 December 2016 / Revised: 9 February 2017 / Accepted: 20 February 2017 /
Published online: 8 March 2017

Abstract The advent of big data age has brought about a growing performance and scale of the storage system, as well as huge energy consumption. Based on the sequential data storage featured workload as video surveillance, etc., we proposed EPS-RAID, that is to add a solid state disk(SSD) and a parity disk on S-RAID, and to optimize the random reads and writes in storage system by means of random write logs and Reed-Solomon(RS) code. Preserving the grouping strategy and the local parallel strategy of S-RAID, EPS-RAID has achieved good effect for random reads and writes on standby grouping, especially for the repeated reads and writes on the same logical address. The experiments show that, in the 3 file systems of EXT4, NTFS, NILFS, the energy consumption of EPS-RAID which consists of 2 disks of each grouping has effectively improved under the premise of ensuring performance and reliability.

Key words Storage system · Disk array · RS code · S-RAID · Energy efficiency

1 Introduction

Recent years have seen increased application of big data in various fields and the popularization of the idea of seeking value in data. For instance, Facebook now processes 350 million pictures, 4.5 billion “likes” and 10 billion messages every single day from its 1.4 billion users, and the hard drive space on its Hadoop (HDFS) cluster exceeds 100 PB, which generates the

✉ Tan Yu-An
tan2008@bit.edu.cn

[1] School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China

[2] Department of Computer Science and Technology, Shandong University of Technology, Zibo, Shandong 255022, China

[3] Institute of Software, Chinese Academy of Science, Beijing 100190, China

[4] Research Center of Massive Language Information Processing and Cloud Computing Application, Beijing 100081, China

advertising revenue of 12 billion US dollars each year. How to cost-effectively store and process the big data at PB and EB magnitudes has become a huge challenge.

Hard disk has become a major storage medium for the large-scale storage system, while redundant array of independent disks (RAID) has realized the parallel working of multiple disks through disk striping to improve the performance, volume and reliability of the storage system [4, 5, 12, 18]. However, the parallel working of large quantities of disks also bring with it the problem of high energy costs. Previous studies show that the energy consumed by disk storage system accounts for over 27% of the total energy consumption of data center [19]. The high energy costs not only add to energy consumption costs, but also pose great danger to reliability.

In fact, some storage systems (video surveillance, etc.) with workload based on sequential data storage require massive storage space, but the demand for performance is not high [24]. Under these circumstances, all disks of RAID still remain active and run at their full capacity, bringing the high energy consumption to the storage system [7, 33].

On the other hand, though well known for its low energy consumption and excellent read and write performance, SSD has stayed out of the order list of storage vendors due to its high price/volume ratio for a long time [11]. With the improvement of the manufacturing technologies, SSD has seen its unit prices falling continuously and its gradual adoption in large-scale storage systems. The storage plan of the data centers has gradually developed in the direction of SSD + HDD.

Our contributions are as follows. In consideration of the workload characteristics of sequential data storage, we present EPS-RAID (Extra-Parity S-RAID) architecture to process the random read/write operations by adding a parity disk and an SSD to S-RAID. Using SSD to cache the parity data with RS code [14, 20, 21] and delaying writing random write into disk arrays, so greater disk idle time will be achieved. By testing several main file systems including EXT4, NTFS and NILFS, it has been found that the energy consumption generally drop by over 20% under the premise of ensuring write performance.

2 Related works

Surveillance video grows exponentially and becomes real big data. The research scope of surveillance video is very broad, including surveillance video processing, video transmission and video storage, etc. [17, 22, 25]. Saeid et al. presents temporal profile method that samples critical locations in the video frames to search long surveillance quickly [3]. Alsmirat M A. et al. designs evaluation frameworks for automated wireless video surveillance system [1, 23]. [6] points out that high bandwidth demand and the large storage requirements are the main challenges in video surveillance system. Based on this, Alsmirat M A. et al. designs a reliable IoT-based wireless video surveillance system that provides an optimal bandwidth distribution and allocation to minimize the overall surveillance video distortion [2]. In addition, with the scale enlargement of surveillance video storage, multiview video coding, feature analysis and conservation of surveillance video have also become focuses [16, 30].

The expansion of surveillance video requires large capacity storage devices and that bring high energy consumption. So, energy saving research of surveillance video storage system has remained a hotspot of research and certain key results have been achieved. The current researches are mainly centered on two levels: physical device level and system level. The physical device level is not of much help for large scale data storage system as it fails to consider the energy consumption and data security of the whole storage system. The system level energy saving research on the storage system is the major direction of research, and its

major way to lower energy consumption is to reduce the number of working disks consuming energy in the storage system, and realized the maximum energy saving under the condition of guaranteeing the performance and reliability.

Weddle et al. propose a Gear-Shifting Power-Aware RAID (PARAID) which constructs sub-RAIDs containing different number of disks within RAID5, dynamically adjusts the number of working disks according to the different loads of the system to achieve the maximal energy saving while guaranteeing performance and reliability [31]. PARAID is typically applied for Web applications. Its defect lies in its insufficient utilization of the disk space as PARAID has to store multiple mirrors of the same data in the sub-RAIDs of different levels at the cost of large amount of storage space. In addition, PARAID requires data synchronization when switching between different sub-RAIDs, and brings negative impact on the performance of the storage system.

Pergamum [26], targets at the archive system, adds a NVRAM and a low performance CPU in each storage node to form a tome unit. Storing data items, data signature and timestamp etc. as the metadata in NVRAM, Pergamum enables some random I/Os to be implemented when the disks are in standby mode. However, the complex structure of Pergamum increases the overall cost of the storage system.

Mao et al. propose a Green RAID (GRAID) to expand the RAID10 of data mirroring and achieve the energy saving by adding a log disk, periodically updating the data on the mirrored disks and shutting down all mirrored disks during the interval between two data updating [13].

Wang et al. propose an eRAID model which uses the redundancy trait of RAID1 to redirect I/O requests, and reduce the energy consumption by shutting down all or partial redundant grouping of disks [29].

Li et al. propose S-RAID [7, 8] which is mainly adopted for sequential write-dominated application system, and reach energy conservation effects by shutting down some idle disks under the premise of meeting the performance requirements of the storage system. Under the ideal state, there is only one grouping of disks remaining working and other groupings of disks remaining in shutdown or in standby mode. S-RAID has a good energy efficient effect compared with the traditional RAID layout. However in reality, the random I/Os, e.g. superblocks and metadata, has resulted in frequent startup of standby disk groupings, leading to degraded performance and extra energy consumption.

Liu et al. propose HS-RAID [10], which made certain energy efficient optimization on S-RAID with NILFS. However, limited by the small volume and erased times of SSD, this scheme is not apply to mainstream file system, as EXT4, NTFS, etc.

DPPDL [28] dynamically allocates the storage space with appropriate degree (number of disks) of partial parallelism according to performance requirement. DPPDL achieves 19% and 56% of the energy saving effect compared with S-RAID and eRAID, but the disk control strategy is too complicated.

3 Brief introduction to S-RAID and data analysis

3.1 Data layout of S-RAID

The S-RAID contains two redundancy strategies: S-RAID4 redundancy strategy and S-RAID5 redundancy strategy, which are similar to the redundancy strategies of RAID4 and RAID5 respectively, and the difference lies in whether the parity data is fixed in one disk or decentralized in different disks. Only the S-RAID5 architecture consisting of 5 disks ($4D + 1P$) is introduced here.

Figure 1 shows data layout of S-RAID5 consisting of 4 data disks (divided into 2 groupings: G_0 and G_1) and 1 parity disk. S-RAID5 has 10 stripes ranging from $Stripe_0$ to $Stripe_9$ with the parity data decentralized evenly on 5 disks. Each stripe grouping (VGroup) contains 2 stripes with the P parity disk fixed, equal to an S-RAID4 architecture. Stripe grouping strategy can ensure the good parallelism and lower the startup and shutdown frequency of the disks. In S-RAID5, parity data are evenly distributed and the startup and shutdown frequency of the disks is higher than S-RAID4, but it effectively avoid the bottleneck of parity disk of S-RAID4, leading to levels of higher performance and reliability over S-RAID4.

The logical block address (LBA) are arranged along the direction of the arrows and next to each other, as in Fig. 1. Therefore, when the system I/O is dominated by sequential data write, the I/O requests of the storage system will concentrate on the same grouping of disks within a certain period of time and other disks will remain free, thus lowering the energy consumption of the whole storage system. In addition, the write operations in S-RAID are all small-write. When the write request comes, the data block and parity block on the same stripes of the disks will be read, S-RAID will recalculated the parity block data and write the new data and calculated parity data into corresponding disk. The recalculation of the parity block is as formula (1).

$$P' = P \underset{D_i \in G_k}{\oplus} \left(D_i \oplus D_i'\right) = P \oplus D_q \oplus D_q' \oplus \cdots \oplus D_{q+N_G-1} \oplus D_{q+N_G-1}' \tag{1}$$

Where $\oplus$ indicates exclusive OR operation, D_i indicates the old data on data disk D_i, D'_i indicates the new data on data disk D_i, N_G is the number of disks in the grouping G_k, D_q is the first disk in the grouping G_k, i.e. $q = k^* N_G$. It can be represented in formula (2) in the stripe form:

$$SP_s' = SP_s \underset{S_{i,s} \in G_k}{\oplus} \left(S_{i,s} \oplus S_{i,s}'\right) = SP_s \oplus S_{q,s} \oplus S_{q,s}' \oplus \cdots \oplus S_{q+N_G-1,s} \oplus S_{q+N_G-1,s}' \tag{2}$$

Where $S_{i,s}$ indicates the old data and $S'_{i,s}$ indicates the new data.

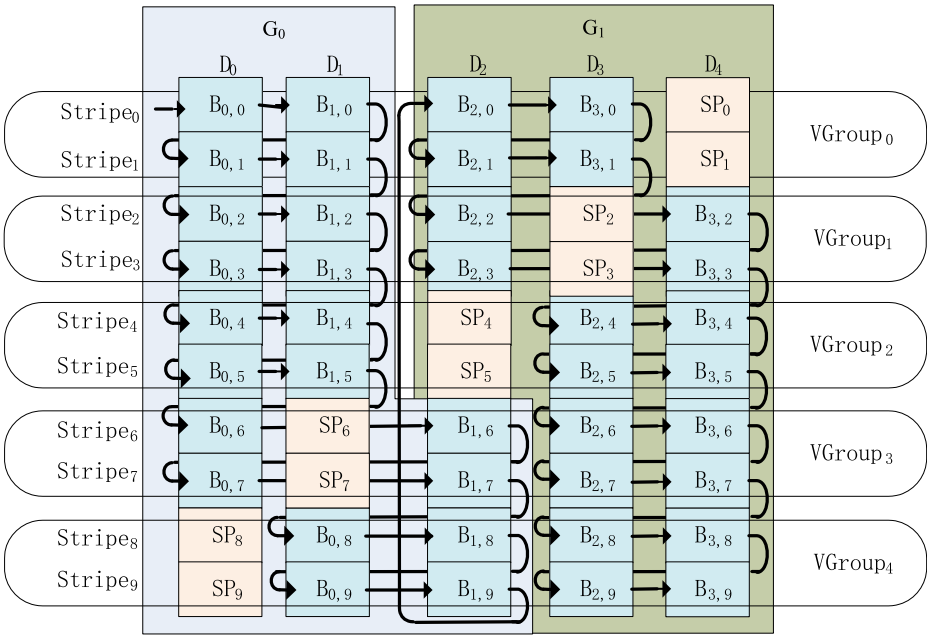


Fig. 1 Data layout of S-RAID5

3.2 Data characteristics of read/write operations of S-RAID

3.2.1 NTFS-based read/write operation characteristics

NTFS is the standard file system launched by Microsoft after Windows NT and the following Windows file systems. In this experiment, NTFS is arranged for S-RAID5 and data is sequentially written into disk groupings by the analog data generator [7]. Figure 2(a), (b) and (c) are the characteristics at the initial 10 min, 1 h and 5 h respectively triggered by the first time to write.

Figure 2(b) and (c) show that the spatial continuity of allocated LBAs is good and basically escalate linearly with the time. In addition, there are also some random reads and writes exist. Figure 2(c) shows that there are a number of random read and write near LBAs $4.0*10^7$, $1.0*10^8$, $1.6*10^8$ and $2.0*10^8$, especially near $4.0*10^7$ where show frequent reads and writes. We suppose that these areas may contain logs or some important data structure. Though in an overall sense the spatial continuity of NTFS is strong, it also shows that locally, the data write fails to strictly following the linear continuity and there is overlapping between LBAs.

3.2.2 EXT4-based read/write operation characteristics

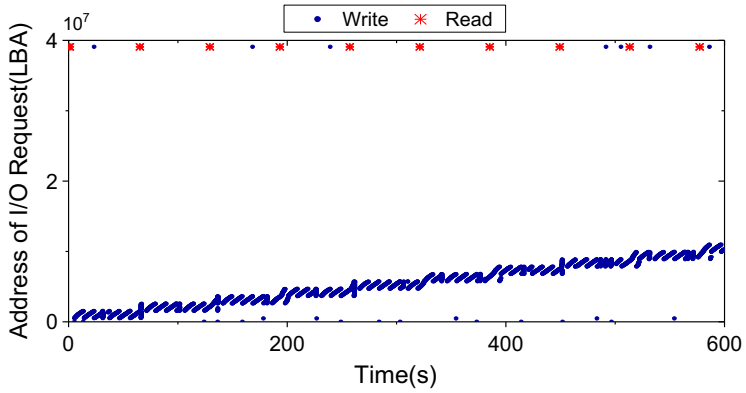
EXT4 is a log-type file system and has been applied extensively in the Linux operating system. We did the sequential write tests for 10 min, 1 h and 5 h on EXT4 just as we did with NTFS.

Unlike NTFS, EXT4 does not record write requests from the start position of the logic address when the data are written in. Figure 3(a) is the data characteristics at the first 10 min. The start LBA of writes is near 1.05×10^8, and there are always read and write operations near LBAs 1.05×10^8 and 1.57×10^8. Figure 3(b) is the data characteristics at the first hour, and includes 6 locally continuous sub areas which are decentralized due to the arrangement of 10 min as a recording unit. EXT4 will choose different block groupings to store the data when each recording unit starts, i.e. the data is written into new areas separated from the previous areas for the purpose of I/O parallelism and balanced allocation of the block groupings. There are random read/write operations and multiple accesses at the initial LBAs of each sub area and near LBA 1.57×10^8, and it is believed that some block grouping and log information of EXT4 are stored in these areas. Figure 3(c) is the data characteristics in 5 h. Overall, the data distribution is in disorder and the continuity is weaker than NTFS.

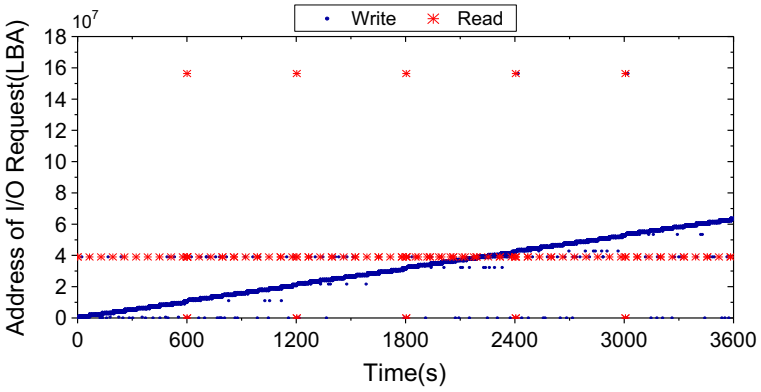
3.2.3 NILFS-based read/write operation characteristics

NILFS is a typical log structured file system and is targeted at restoring the overwritten or damaged files by consecutive snapshots.

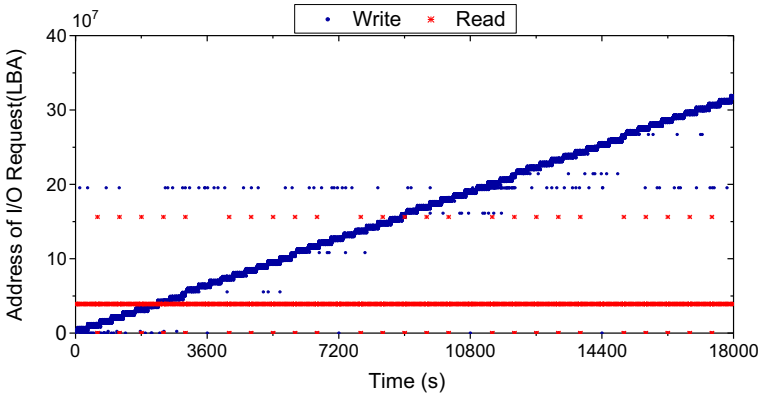
Figure 4(a), (b) and (c) give the I/O distribution at the initial 10 min, 1 h and 5 h respectively. In Fig. 4, I/O requests are mainly distributed in 3 linear areas, where the two lines near LBAs 0 and $9.8*10^8$ are horizontal, indicating that these areas have been accessed for multiple times, and the third line is a standard incremental slash, indicating that these LBAs have been accessed at least once basically. The data characteristics of NILFS are more regular than EXT4 and NTFS, but there are frequent I/Os near LBAs 0 and $9.8*10^8$, which results in the failure to shut down the disk grouping and negatively influences the energy saving effects of the system



(a) 10 mins

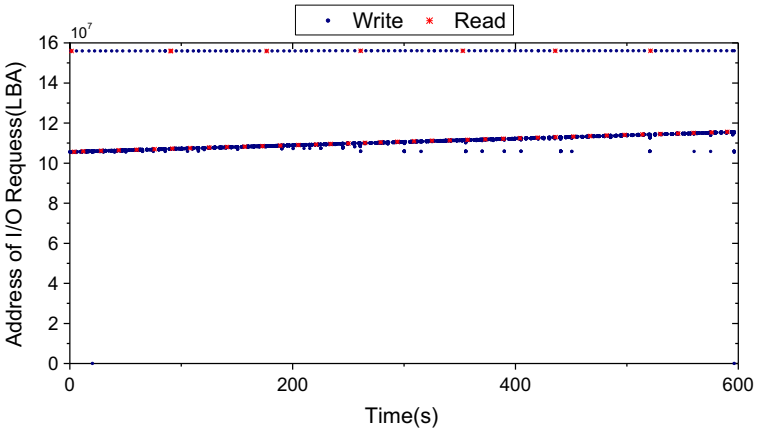


(b) 1 hour

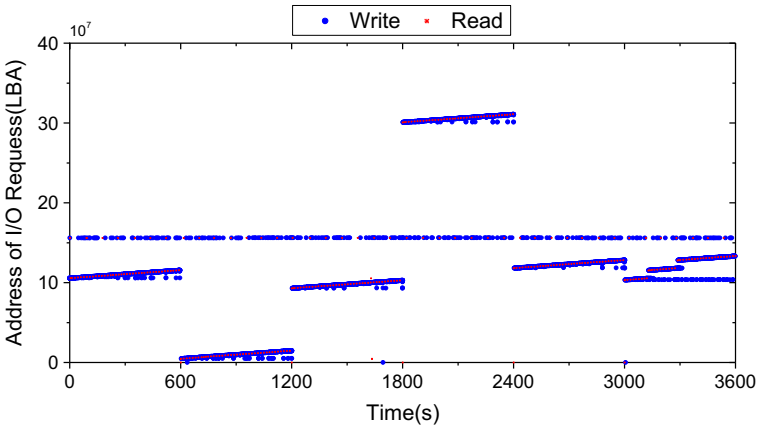


(c) 5 hour

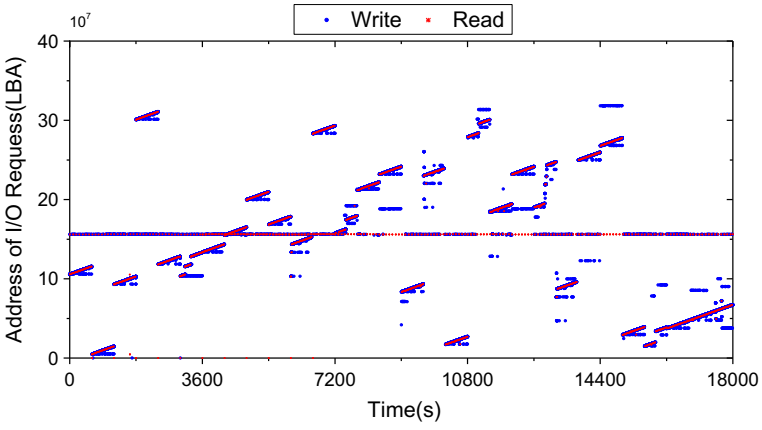
Fig. 2 Data characteristics of NTFS file system



(a) 10 mins

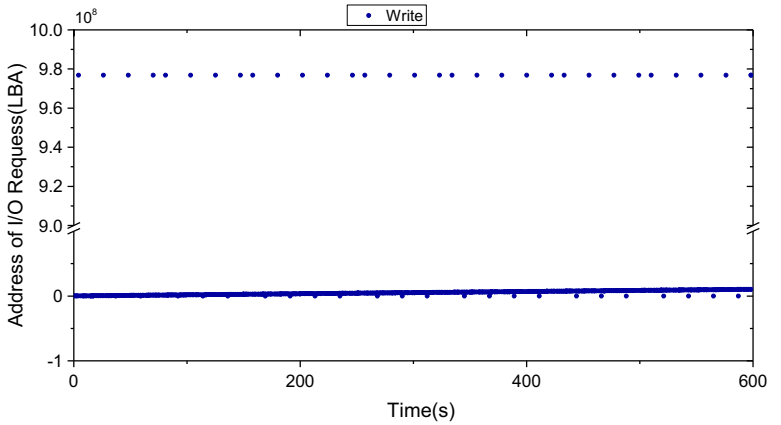


(b) 1 hour

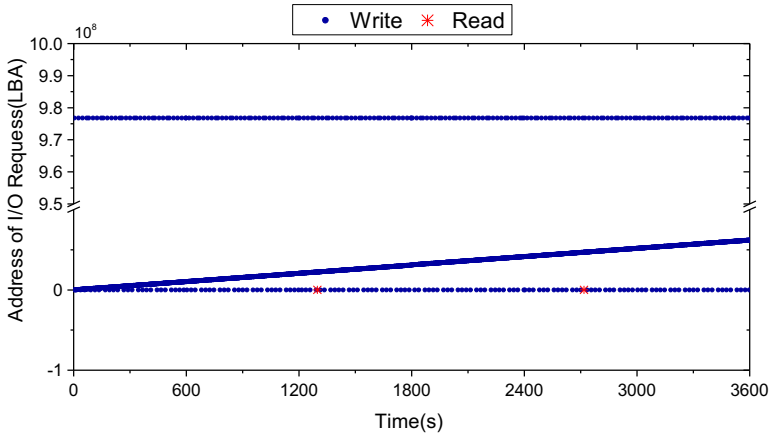


(c) 5 hours

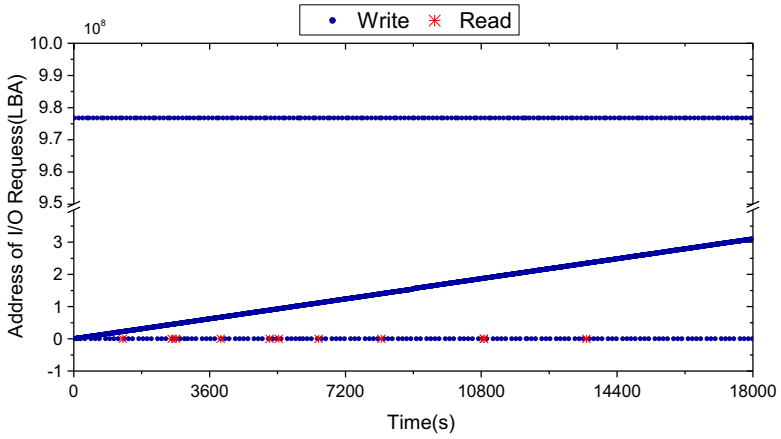
Fig. 3 Data characteristics of EXT4



(a) 10 mins



(b) 1 hour



(c) 5 hours

Fig. 4 Data characteristics of NILFS file system

The above data characteristics provide the basic references for our application of EXT4, NTFS, and NILFS file systems and guide us to conduct energy saving optimization on S-RAID5.

4 Architecture of EPS-RAID

S-RAID is mainly used in applications dominated by sequential writes. By analyzing the data characteristics of EXT4, NTFS, and NILFS as in section 2.2, it is found that even when the system is dominated by sequential I/Os, a few random reads and writes still exist. The existence of such random I/Os make the idle time of some disks shorten, inability to shut down or require frequent startup, which seriously impacts on the performance and energy saving effects of the storage system. This paper proposes an S-RAID-based extra parity method, EPS-RAID, to deal with the random reads and writes of the standby data disk groupings to extend the standby time of the disks. The main characteristic of EPS-RAID is as follows.

(1) Adding 1 parity disk to S-RAID, there will be two parity disks: P and U. Parity disk P shall be calculated by exclusive OR operation with every data disk of each grouping, i.e. the former S-RAID parity; parity disk U shall be calculated according to RS codes, i.e., every data disk of each grouping multiplied by a non-vanishing element g_i within the finite group GF(256) and the results do exclusive OR operation.
(2) An SSD is added to S-RAID as the cache memory to record the new random data and intermediate U parity data(U_t) while U parity disk shall be used to record the latest U parity data.
(3) A data processing algorithm is added to maintain logs in SSD. Keep records of updated data and temporary parity data in the face of write operations and retrieve log list in the face of read operations.

The EPS-RAID architecture is shown in Fig. 5.

The storage system consists of two parts: one part is the SSD used as cache, and the other part is a RAID consisting of several disk groupings and performing quadratic parity (P&U).

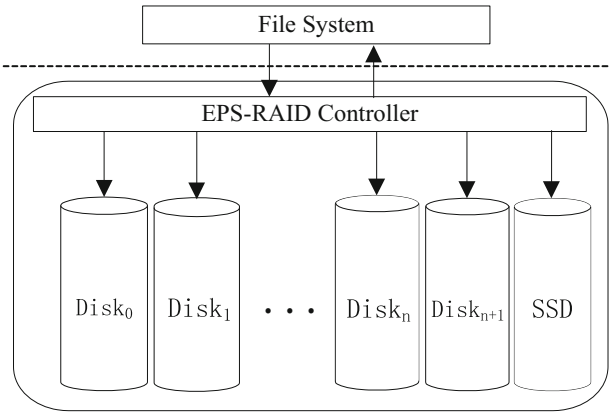


Fig. 5 Architecture diagram of EPS-RAID

EPS-RAID contains two parity disks *P&U* as well as two redundancy strategies, which are similar to that of S-RAID4 and S-RAID5 respectively. We will first discuss EPS-RAID4 and then proceed to EPS-RAID5. Figure 6 is the data layout of EPS-RAID4 (3 disk groupings and 2 disks for each grouping) strategy consisting of 8 disks ($6D + P + U$).

We can see there are 8 disks in Fig. 6: D_0 to D_5 are data disks and *P, U* are parity disks. There are 3 groupings of disks: G_0, G_1 and G_2, and each grouping contain 2 disks: $G_0 = \{D_0, D_1\}$, $G_1 = \{D_2, D_3\}$ and $G_2 = \{D_4, D_5\}$. Therefore we have $N_{array\_disk} = 8$ and $N_G = 3$. Blocks in data disks are written as $B_{d,s}$ where d denotes the disk number and s denotes the stripe number. Unlike S-RAID4, such data layout has a parity disk U beside the original parity disk P.

To generalize the idea to an arbitrary EPS-RAID4, we use a matrix to represent an EPS-RAID4 with n data disks and m stripes:

$$\begin{pmatrix} S_{0,0} & S_{1,0} & \cdots & S_{n-1,0} & SP_0 & SU_0 \\ S_{0,1} & S_{1,1} & \cdots & S_{n-1,1} & SP_1 & SU_1 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ S_{0,m-1} & S_{1,m-1} & \cdots & S_{n-1,m-1} & SP_{m-1} & SU_{m-1} \end{pmatrix} \quad (3)$$

In the matrix, each row corresponds to a stripe and each column corresponds to a disk. Element $S_{d,s}$ in the matrix corresponds to the block of disk D_d which lies in $Stripe_s$. The first n disks can be appropriately divided into groupings to meet the performance requirement and energy saving. For example, suppose a single disk of EPS-RAID can provide 30 MB/s write bandwidth, every grouping with 2 disks may be constructed facing an application at sequential write speed of 60 MB/s; every grouping with 4 disks may be constructed facing an application at sequential write speed of 120 MB/s.

As discussed above, EPS-RAID4 uses fixed parity disks like traditional RAID4, and the parity disks cannot be shut down or put into standby mode, so the parity disks may become a bottleneck. To ease the bottleneck of parity disks, we introduce the EPS-RAID5 that decentralizes parity blocks among the disks, as shown in Fig. 7.

Figure 7 shows the scenario of $N_{array\_disk} = 8$ and $N_G = 3$. If each disk grouping requires $r (r > 2)$ disks, we can label the LBAs of remaining r-2 disks(s) as NULL. In this case, the EPS-

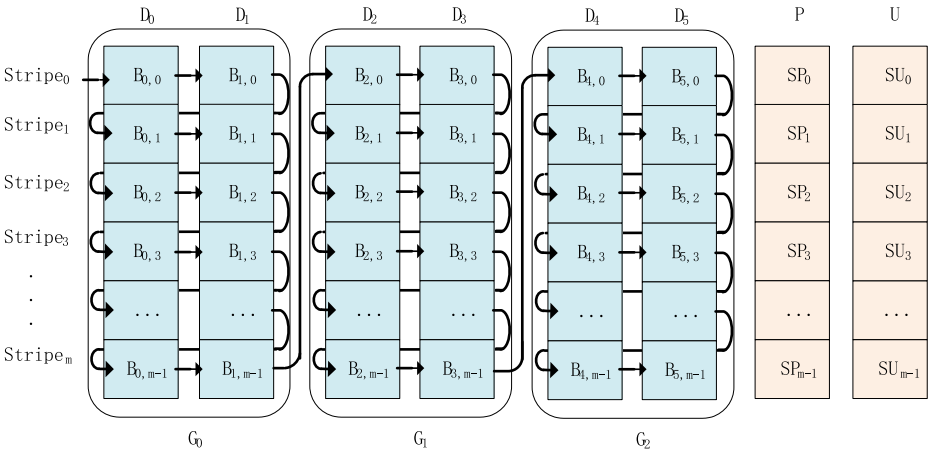


Fig. 6 Data layout of EPS-RAID4

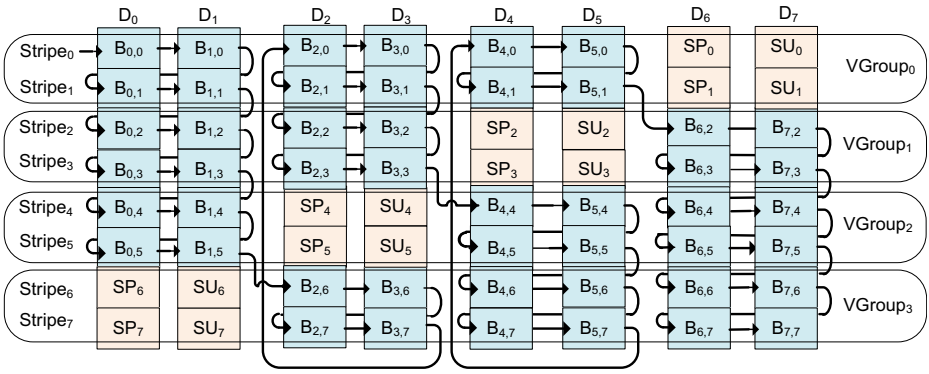


Fig. 7 Data layout of EPS-RAID5

RAID5 will be able to work normally and obtain good energy saving effects, but the labeled disk space will be wasted.

Similar to the general form of EPS-RAID4, we represent EPS-RAID5 with n data disks and m stripes:

$$\begin{pmatrix} S_{0,0} & S_{1,0} & \cdots & S_{n-2,0} & S_{n-1,0} & SP_0 & SU_0 \\ S_{0,1} & S_{1,1} & \cdots & S_{n-2,1} & S_{n-1,1} & SP_1 & SU_1 \\ S_{0,2} & S_{1,2} & \cdots & SP_2 & SU_2 & S_{n-2,2} & S_{n-1,2} \\ S_{0,3} & S_{1,3} & \cdots & SP_3 & SU_3 & S_{n-2,3} & S_{n-1,3} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ SP_{m-2} & SU_{m-2} & \cdots & S_{n-4,m-2} & S_{n-3,m-2} & S_{n-2,m-2} & S_{n-1,m-2} \\ SP_{m-1} & SU_{m-1} & \cdots & S_{n-4,m-1} & S_{n-3,m-1} & S_{n-2,m-1} & S_{n-1,m-1} \end{pmatrix} \quad (4)$$

5 Write operations in EPS-RAID

Read operations of EPS–RAID are always sequential read operations which are composed of replaying previous write operations, such as replaying in video surveillance system. Write operations take sequential writes as a main form, but a small amount of random writes throughout the process. Here we mainly discuss the write operations of EPS-RAID.

5.1 Write operations of working disk

Working disks are mainly written in the form of sequential writes. By the analysis of characteristics in section 2, we can see that most of the writes concentrate in one main linear area which lies in working disk grouping. At this moment, whole stripe writing strategy can be used, i.e., write operation is performed when the cached data is enough to fill with a whole stripe. The whole stripe writing strategy greatly improves write efficiency.

As mentioned above, the difference between EPS-RAID and S-RAID is that the EPS-RAID adopts quadratic redundancy model where a parity disk U is added to the original parity disk P. Parity disk P will retain the original parity on S-RAID, as indicated in formula (1)(2), while parity disk U adopt the RS code parity methods similar to RAID6, i.e. every data disk of each

grouping will be multiplied by a non-vanishing element $g_k \in GF(256)$ $(0 \le k \le 254)$ before the exclusive OR calculation. The calculation of parity disk U is as follows:

$$U' = U \underset{D_i \in G_k}{\oplus} \left(g_k D_i \oplus g_k D_i'\right) = U \oplus g_k D_q \oplus g_k D_q' \oplus \cdots \oplus g_k D_{q+N_G-1} \oplus g_k D_{q+N_G-1}' \quad (5)$$

Where g_k is a non-vanishing element in GF(256). We can express it in stripe form as follows:

$$SU_s' = SU_s \underset{S_{i,s} \in G_k}{\oplus} \left(g_k S_{i,s} \oplus g_k S_{i,s}'\right) = SU_s \oplus g_k S_{q,s} \oplus g_k S_{q,s}' \oplus \cdots \oplus g_k S_{q+N_G-1,s} \oplus g_k S_{q+N_G-1,s}' \quad (6)$$

5.2 Write operations of non-working disk

Writes of non-working disk is mainly random writes, e.g. near $4*10^7$ and $1.6*10^8$ in Fig. 2(c), near $1.57*10^8$ in Fig. 3(c) and near 0 and 9.7×10^8 in Fig. 4(c), etc. Though such random writes are not many in the whole write aggregate compared with sequential writes, it has great influence on the energy consumption of S-RAID. EPS-RAID focuses on optimizing these random writes using SSD and parity disk by absorbing the random writes of the non-working disks to extend the idle time of the disk groupings. Thus, frequent switch of the data disks between standby mode and working mode is avoided.

5.2.1 Writes of non-sequential data will also result in the recalculation of the parity data.

Let's assume that EPS-RAID contains n disk groupings $\{G_0, G_1, \ldots\ldots, G_{n-1}\}$, i.e., $N_G = n$, random writes are sent to *No.s* stripe of data disk grouping $G_k (0 \le k \le n-1)$, and the new data is G'_k. We can first of all calculate a new temporary parity U_t:

$$\begin{aligned} U_t &= U \oplus g_k P \\ &= (g_0 G_0 \oplus g_1 G_1 \oplus \cdots \oplus g_k G_k \oplus \cdots \oplus g_{n-1} G_{n-1}) \oplus (g_k G_0 \oplus g_k G_1 \oplus \cdots \oplus g_k G_k \oplus \cdots \oplus g_k G_{n-1}) \\ &= (g_0 \oplus g_k) G_0 \oplus (g_1 \oplus g_k) G_1 \oplus \cdots \oplus (g_{k-1} \oplus g_k) G_{k-1} \oplus \left(g_{k+1} \oplus g_k\right) G_{k+1} \oplus \ldots (g_{n-1} \oplus g_k) G_{n-1} \end{aligned} \quad (7)$$

where $G_k = \{D_{NG*k}, D_{NG*k+1}, \cdots, D_{NG*(k+1)-1}\}$. P and U are the parity data corresponding to *No.s* stripe, and can be directly obtained as the disks remain in working mode. Note there is no old information of G_k in U_t, therefore it is not necessary to startup G_k to read the old data. Adding in G'_k and the newest U parity data: U', can be expressed as:

$$\begin{aligned} U' &= U_t \oplus g_k G_k' \\ &= (g_0 \oplus g_k) G_0 \oplus (g_1 \oplus g_k) G_1 \oplus \cdots \oplus (g_{k-1} \oplus g_k) G_{k-1} \oplus g_k G_k' \oplus \left(g_{k+1} \oplus g_k\right) G_{k+1} \oplus \cdots \oplus (g_{n-1} \oplus g_k) G_{n-1} \end{aligned} \quad (8)$$

where $G'_k = \{D'_{NG*k}, D'_{NG*k+1}, \cdots, D'_{NG*(k+1)-1}\}$. It can also be expressed in stripe form as follows:

$$SU_s' = SU_{ts} \oplus g_k S G_s' = SU_{ts} \oplus g_k S_{N_G*k}' \oplus g_k S_{N_G*k+1}' \oplus \cdots \oplus g_k S_{N_G*(k+1)-1}' \quad (9)$$

Writing U' into parity disk U, delaying the write of new data G'_k into G_k grouping and writing G'_k and U_t temporarily into SSD. The largest advantage of the above operations lies in the extension of the idle time of data disk grouping G_k, making it possible to remain standby

Table 1 Examples of committed stripe coefficients in EPS-RAID

Examples	Example1				Example 2			
	G_0	G_1	G_2	G_3	G_0	G_1	G_2	G_3
Coefficients	g_{0x00}	g_{0x01}	g_{0x02}	g_{0x03}	g_{0x11}	g_{0x21}	g_{0x31}	g_{0x41}

Hexadecimal number 0×00 and 0×01 are the offset in non-vanishing element table of GF(256)

instead of re-start. In addition, as U_t is temporarily stored in SSD, it can be used for quick calculation of U' when new data write occurs to G_k.

In Formula (7)(8) where it is required that $(g_0 \oplus g_k)$,$(g_1 \oplus g_k)$ $(g_{n-1} \oplus g_k)$ is not a vanishing element, it is necessary that the coefficients for each of disk groupings shall be different. GF(256) consists of 255 non-vanishing elements: (g_0, g_1, g_{254}), and $g_k (0 \le k \le 254)$ can be any one of them, but it must be guaranteed that it will not be repeated in the calculation of the same stripe. A good solution is to pre-commit the g_k. Table 1 is two examples of committed stripe coefficients in EPS-RAID consisting of 4 disk groupings.

The above method requires the calculation in advance of the multiplication table and the inverse element table on GF(256). Each of the tables occupies 65 k bits. In addition, the coefficient of each stripe on each data disk grouping shall be recorded, and as each coefficient needs one byte, the space required shall be very small. After the optimization of the random writes, the parity disk P can no longer be used as a parity disk due to the data it stores are not the latest parity data. But the parity disk U is still be effective and can be used to restore data. We can re-implement above small write optimization on any group again as long as we re-read $G_0 \sim G_{n-1}$ and restore the parity disks P and U. Such restriction is acceptable in sequential data storage system. Therefore, this method has the best optimization effects on the multiple repeated reads and writes near $4*10^7$ and $1.6*10^8$ in Fig. 2(c), near 1.57×10^8 in Fig. 3(c) and near 0, $9.7*10^8$ in Fig. 4(c).

5.2.2 Data structure in SSD and record algorithm of random writes

G'_k and U_t are kept in SSD, managed in logs temporarily, as indicated in Fig. 8. Each LogID corresponds to a write block [15, 32].

EPS-RAID controls the log record of blocks, deciding when to redirect blocks to SSD and when to write them in disks directly. A write request may also be split into several pieces with different logIDs respectively while it is redirected to SSD. The main log variables are as follows:

- LogID: log manager identity;
- LBA: LBA of random write in EPS-RAID5;

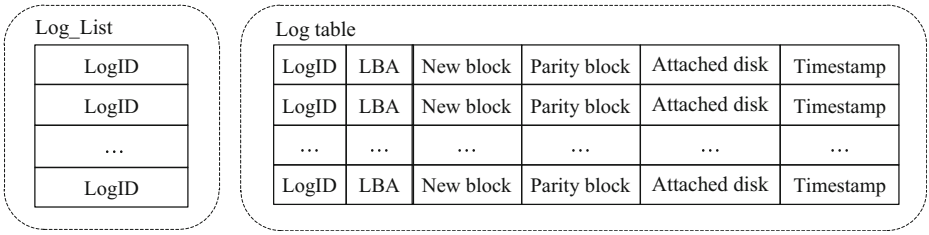


Fig. 8 Structure of log list in SSD

- New block: new random write block in EPS-RAID5;
- Parity block: temporary parity block;
- Attached disk: disk to which the random write block is belonged;
- Timestamp: timestamp of the random write request;

Redirected random writes will cause conformity issues of recorded data, therefore it is necessary to execute a special write request process that records every random write and its corresponding parity data. Record algorithm of random writes is as follows.

Algorithm 1 Record algorithm of random writes

W_{random} : Random write data; *WB*: Write block； *Log_List*: Log list of write block； G_{active}: active disk grouping

LBA(*WB*): Logic block address of write block; *Log table*: log table;

Input: W_{random}, *Log_List*

Output: Updated *Log_List*

Recordwrite (W_{random}， *Log_List*)

Foreach *WB* in W_{random}

If LBA(*WB*) $\in G_{active}$ **then**

// Write block is located in active disk grouping. Save *WB*, *P*&*U* directly

Save *WB* in G_{active}

Recalculate updated parity *P'*,*U'*

Save *P'*,*U'* in *P*, *U*

Else

// Write block is located in inactive disk groupings. Delay writing in disks and save *WB*, U_t in SSD

//temporarily

AppendNode(*Log_List,LogID*)

Calculate parity U_t

AppendRecord(*Log table, LogID,WB,* U_t,…)

5.2.3 Read operations of SSD

The controller needs to read from SSD and switch the standby disks to working condition first when writes back the logs. There are two situations. One is incidental-write: Every disk grouping has more than one time in active mode and EPS-RAID can reconstruct logs and write them back into active disks during these periods. For example, in Fig. 3(c), LBAs of 8400 s ~ 9000 s and 9600 s ~ 10,200 s are sequential respectively and located on the same disk grouping. The logs of random writes on these disk groupings can be rewritten within any one of the two periods. The other is to set a threshold t_{flush} to control the disks.

According to the algorithm 1, the controller need to search logs to obtain the latest version at time of writing back logs because multiple logs that have different version numbers (due to different time of updating) may have been recorded [15]. In addition, a small amount of read operations exist irregularly in logs of random writes. Read-hit in SSD is most likely to occur and LBAs with more intensive I/Os has a higher hit-rate. For example, near $15.7*10^7$ in

Table 2 ST1000NM0033 parameter

Parameter	Specifications
Interface type	SATA 3.0
Speed	7200 rpm
Disk capacity	1000G
Active mode energy Consumption	15.7 W
Stand-by mode energy consumption	1.3 W

Fig. 3(c) and near 0, 9.7×10^8 in Fig. 4(c). The subsequent experiments also prove that our scheme contributes most to improving the degradation from these LBAs.

6 Energy consumption test

We constructed a EPS-RAID5 consisting of 12 hard disks and 1 SSD under Linux kernel 3.2.81, and the default size of data block within the stripe is 64 KB. The disks are ST1000NM0033; the SSD is Samsung840Pro and the specific parameters can be found in Table 2 and Table 3.

EPS-RAID5 uses *MD* array management functions and *diskpm* tool under Linux to dispatch the disks, and will put the disk in stand-by mode when the disk has remained idle for a certain period of time without data I/Os [7]. As far as the EPS-RAID is concerned, the optimal situation is to have one grouping of data disks, 2 parity disks, and 1 SSD remaining in working conditions.

In order to test the energy consumption effects, we established an experiment environment indicated as Fig. 9 [7, 8, 27]: A_0 to A_{n+1} are high accuracy ammeter in series with the disk and the power. The energy consumption of the disk array includes the energy consumption of the SSD and all disks (working status and stand-by status). By measuring the current value of the ammeter periodically, the energy consumption of the whole storage system can be calculated using the formula $W = \sum_{i=0}^{n+1} V_i * I_i$, where V_i indicates the voltage of the storage medium and I_i indicates the current value.

An analog data generator was programmed with C language to simulate 256-line D1 video surveillance data. The video data shall be written into video file as append and the interval of time specified in the experiment is 10 min. 12-h energy consumption test was carried out. Though EPS-RAID consisting of 2 disks in each grouping could provide larger bandwidth, we did not challenge it in order to ensure the steady state of the system. The right part of Fig. 10 is the test results of S-RAID and EPS-RAID with 2 disks in each grouping.

The left part of Fig. 10 illustrates the energy consumption of RAID5/PARAID/eRAID/CacheRAID/DPPDL as reference. The above five data layout is configured to store 256-line

Table 3 Samsung840Pro parameter

Parameter	Specifications
Interface type	SATA 3.0
Disk capacity	128G
Read/ write mode energy consumption	7.5 W

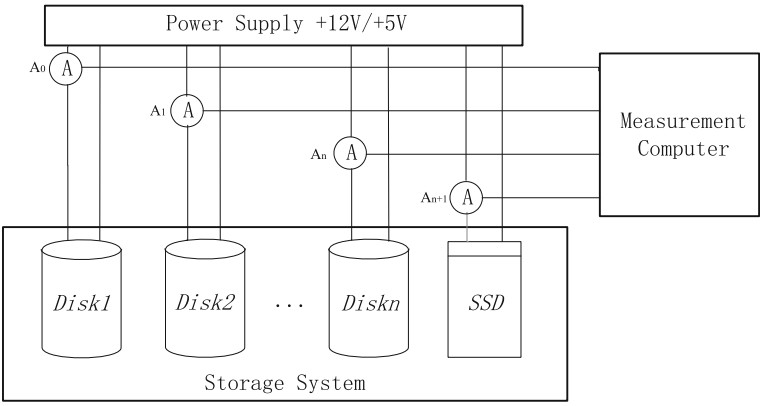


Fig. 9 Structure diagram of energy consumption test

D1 video surveillance data. It can be seen from Fig. 10 that the RAID5 has the largest energy consumption, reaching $9.90*10^6$ J, mainly because the work mode of RAID5 is that workloads are balanced to use all disks even for light workload. PARAID adopts special striping method and uses different quantities of disks to match different workload, and eRAID reduces the energy consumption by reducing the speed of partial or entire mirror group disks to the standby mode, achieving the effect of energy consumption cut by 18.7% and 27.9% compared with that of RAID5. In fact, they are less-targeted as in the background of random data storage.

CacheRAID and DPPDL are two kinds of effective energy saving solutions and consume 37.6% and 35.7% power compared with that of RAID5. CacheRAID reduce the energy consumption by large-capacity and low-power SSD cache. On the other hand, the high cost per GB of SSD is not suitable for wide arrangement in the video surveillance system. We also notice that DPPDL dispatches 4 working disks on average to record the 256-lined D1 video surveillance data and is the most energy efficient layout in reference solutions. In contrast, to afford 256-lined D1 surveillance video, EPS-RAID only needs to employ 2 working

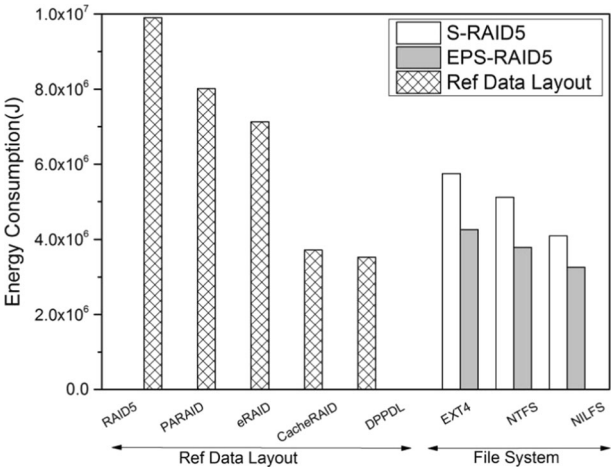


Fig. 10 A comparison on energy consumption among PARAID/eRAID/CacheRAID/DPPDL/S-RAID/EPS-RAID

disks. The main reason is that random read and write operations in workload lead to write performance degradation of DPPDL and more working disks needed to match the required bandwidth.

We also test the S-RAID in EXT4, NTFS and NILFS file systems with sequential data I/O dominated workload and find that they are all energy efficient, saving 41.9%, 48.3%, and 58.7% than RAID5 respectively. S-RAID becomes less energy efficient while the intensity of random small reads and writes increases and EXT4 performs the worst. After the optimization strategy is adopted, with 2 working disks, the energy consumption of the EPS-RAID drops by $1.49*10^6$J (25.91%), $1.33*10^6$J (25.98%) and $0.83*10^6$J (20.29%) respectively compared with that of S-RAID. NTFS and EXT4 improve energy saving effect obviously but NILFS still achieves the best. In summary, EPS-RAID has remarkable energy efficiency in EXT4, NTFS and NILFS.

7 Conclusion

Surveillance video has become the largest source of the big data and consumed large amounts of energy. This paper presents an energy saving data layout by adding a solid-state disk and parity disk to deal with random data reading and writing. The scheme is suitable for sequential data storage featured workload such as video surveillance system, etc. We have designed and implemented EPS-RAID in the Linux kernel. The experiments have been carried out to show that disks can be more effectively controlled between working mode and idle mode. The major advantage of EPS-RAID is that the energy saving is greatly improved compared with S-RAID. So, EPS-RAID is a good solution to energy saving. This scheme can also be implemented at commercial RAID controllers with embedded processors, which is one of our future research works [9, 34–38].

Acknowledgments This research was supported by the National Natural Science Foundation of China (No.61370063, 61379048, No. U1636213).

References

1. Alsmirat M, Jararweh Y, Obaidat I, Gupta B (2016a) Automated wireless video surveillance: an evaluation framework. J Real-Time Image Proc 1–20
2. Alsmirat M, Jararweh Y, Obaidat I, Gupta BB (2016b) Internet of surveillance: a cloud supported large-scale wireless surveillance system. J Supercomput 1–20
3. Bagheri S, Zheng J, Sinha S (2016) Temporal mapping of surveillance video for indexing and summarization. Comp Vision Image Underst 144(C):237–257
4. Chen P, Lee E, Gibson G, Katz R, Patterson D (1997) Raid: high-performance, reliable secondary storage. ACM Comput Surv 26(2):145–185
5. Fang Y, Tan Y, Zhang Q, Fei W, Cheng Z, Zheng J (2016) An effective RAID data layout for object-based de-duplication backup system. Chin J Electron 25(5):832–840
6. Gupta B, Agrawal D, Yamaguchi S (2016) Handbook of research on modern cryptographic solutions for computer and cyber security
7. Li X, Tan Y, Sun Z (2011) Semi-RAID: a reliable energy-aware RAID data layout for sequential data access. In: Proceedings of IEEE 27th Symposium on Mass Storage Systems & Technologies(MSST), pp. 1–11
8. Li Y, Sun Z, Ma Z, Tan Y (2014) S-raid 5:an energy-saving raid for sequential access based applications. Chinese J Comput 36(6):1290–1302

9. Li J, Yan H, Liu Z, Chen X (2015) Location-sharing systems with enhanced privacy in mobile online social networks. IEEE Syst J:1–10. doi:10.1109/JSYST.2015.2415835
10. Liu J, Zheng J, Li Y, Sun Z, Wang W, Tan Y (2013) Hybrid s-raid: an energy-efficient data layout for sequential data storage. Journal of Computer Research and Development 50(1):37–48
11. Lu Y, Shu J (2013) Survey on FTL-based storage systems. Journal of Computer Research and Development 01:49–59
12. Ma A, Douglis F, Lu G, Sawyer D, Chandra S, Hsu W (2015) RAIDShield: characterizing, monitoring, and proactively protecting against disk failures. ACM Transactions on Storage 11(4):1–28
13. Mao B, Feng D, Jiang H, Wu S (2008) GRAID: a green RAID storage architecture with improved energy efficiency and reliability. In: Proceedings of International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, pp. 113–120
14. Mu Y, Liu J, Zhang J, Mei R (2007) Application of reed-Solomon algorithm in RAID6. Chin J Electron 35(s2):90–94
15. Narayanan D, Donnelly A, Rowstron A (2008) Write off-loading:practical power management for enterprise storage. ACM Transactions on Storage 4(3):256–267
16. Pan Z, Zhang Y, Kwong S (2015) Efficient motion and disparity estimation optimization for low complexity multiview video coding. IEEE Trans Broadcast 61(2):166–176
17. Pan Z, Lei J, Zhang Y, Sun X (2016) Fast motion estimation based on content property for low-complexity h.265/hevc encoder. IEEE Trans Broadcast 62:1–10
18. Patterson D, Gibson G, Katz R (1988) A case for redundant arrays of inexpensive disks (RAID). In: Proceedings of ACM international conference on management of data, pp 109–116
19. Pinheiro E, Bianchini R, Dubnicki C (2006) Exploiting redundancy to conserve energy in storage systems. ACM Sigmetrics Performance Evaluation Review 34(1):15–26
20. Plank J (2010) A tutorial on reed-Solomon coding for faulttolerance in raid-like systems. Softw Pract Exp 27(9):995–1012
21. Plank J, Ding Y (2005) Note: correction to the 1997 tutorial on reed–Solomon coding. Softw Pract Exp 35(2):189–194
22. Psannis K (2009) Efficient redundant frames encoding algorithm for streaming video over error prone wireless channels. IEICE Electronics Express 6(21):1497–1502
23. Psannis K (2016) Hevc in wireless environments. J Real-Time Image Proc 12(2):509–516
24. Psannis K, Ishibashi Y (2008) Efficient flexible macroblock ordering technique. IEICE Trans Commun E91B(8):2692–2701
25. Psannis K, Ishibashi Y (2009) Efficient error resilient algorithm for h.264/avc: mobility management in wireless video streaming. Telecommun Syst 41(2):65–76
26. Storer M, Greenan K, Miller E, Voruganti K (2008) Pergamum: replacing tape with energy efficient, reliable, disk-based archival storage. In: Proceedings of Usenix Conference on File and Storage Technologies(FAST), pp.1–16
27. Sun Z, Tan Y, Li Y (2014) An energy-efficient storage for video surveillance. Multimed Tools Appl 73(1): 151–167
28. Sun Z, Zhang Q, Li Y, Tan Y (2016) DPPDL: a dynamic partial-parallel data layout for green video surveillance storage. IEEE Trans Circuits Syst Video Technol. doi:10.1109/TCSVT.2016.2605045
29. Wang J, Zhu H, Li D (2007) Eraid: conserving energy in conventional disk-based raid system. IEEE Trans Comput 57(3):359–374
30. Wang J, Li T, Shi Y, Lian S, Ye J (2016) Forensics feature analysis in quaternion wavelet domain for distinguishing photographic images and computer graphics. Multimed Tools Appl 1–17
31. Weddle C, Oldham M, Qian J, Wang A, Reiher P, Kuenning G (2007) Paraid: a gear-shifting power-aware raid. ACM Transactions on Storage 3(3):1–13
32. Wu S, Jiang H, Feng D, Tian L, Mao B (2009) WorkOut: I/O workload outsourcing for boosting RAID reconstruction performance. In: Proceedings of Usenix Conference on File and Storage Technologies(FAST), pp 239–252
33. Yuan C, Sun X, Lv R (2016) Fingerprint liveness detection based on multi-scale lpq and pca. China Commun 13(7):60–65
34. Zhang X, Tan Y, Xue Y, Zhang Q, Li Y, Zhang C, Zheng J (2016) Cryptographic key protection against FROST for mobile devices. Clust Comput. doi:10.1007/s10586-016-0721-3
35. Zhu R, Tan Y, Zhang Q, Wu F, Zheng J, Xue Y (2016a) Determining image base of firmware files for ARM devices. IEICE Trans Inf Syst E99D(2):351–359

36. Zhu R, Tan Y, Zhang Q, Li Y, Zheng J (2016b) Determining image base of firmware for ARM devices by matching literal pools. Digit Investig 16:19–28
37. Zhu R, Zhang B, Mao J, Zhang Q, Tan Y (2016c) A methodology for determining the image base of arm-based industrial control system firmware. Int J Crit Infrastruct Prot. doi:10.1016/j.ijcip.2016.12.002
38. Zhu H, Tan Y, Zhang X, Zhu L, Zhang C, Zheng J (2017) A round-optimal lattice-based blind signature scheme for cloud services. Futur Gener Comput Syst. doi:10.1016/j.future.2017.01.031

Yu Xiao : Ph.D. candidate at the School of Computer Science, Beijing Institute of Technology. His current research interests include network storage and embedded system.

Zhang Changyou , received his Ph.D degree in Beijing Institute of Technology, Beijing, China. He is a professor in Institute of Software, Chinese Academy of Sciences. His current research interests include machine learning, distributed computing and system, parallel algorithm.

Xue Yuan : received the MS degree in software engineering in 2010 from Beijing Institute of Technology, now he is a PhD candidate in Beijing Institute of Technology. His main research interests include information security, vulnerability discovery and malicious code detection.

Zhu Hongfei : born in 1985. Ph.D. candidate in Beijing Institute of Technology, Beijing, China. His research areas are cryptography and privacy protection, 5 south zhongguancun street, Haidian District, Beijing 100081, China.

Li Yuanzhang : Ph.D. and master supervisor in Beijing Institute of Technology, China. His main research interests focus on network storage, storage security and embedded system.

Tan Yu-An : Ph.D. and master supervisor in Beijing Institute of Technology, China. His main research interests focus on network storage, storage security and embedded system.