Out-of-channel data placement for balancing wear-out and I/O workloads in RAID-enabled SSDs

Fan Yang, Chengqi Xiao, Jun Li, Zhibing Sha, Zhigang Cai, Jianwei Liao
College of Computer and Information Science, Southwest Univer sity, Chongqing, China
Corresponding author: J. Liao (*liaotoad@gmail.com*)

***Abstract*—Channel-level RAID implementation SSDs can fight against channel failures inside SSDs, but greatly suffer from imbalanced wear-out (i.e. erase) and I/O workloads across all SSD channels, due to the nature of in-channel updates on data/parity chunks of data stripes. This paper proposes exchanging channel locations of data/parity chunks belonging to the same stripe when satisfying update (write) requests, termed as out-of-channel data placement. Consequently, it can smooth wear-out and I/O workloads across SSD channels, thus reducing I/O response time. Through a series of emulation experiments on several realistic disk traces, we show that our proposal can greatly improve I/O performance, as well as noticeably balance the wear-out and I/O workloads, in contrast to related methods.**

***Index Terms*—RAID-enabled SSDs, Imbalanced workloads, Out-of-channel placement, Modeling, I/O performance.**

I. Introduction

Thanks to the advantages of small size, high performance, random-access and low energy consumption, NAND flash-based Solid-State Drives (SSDs) are becoming the mainstream storage in a wide range of embedded systems, personal computers, and high performance platforms [1], [2]. On the other hand, modern high density SSD devices are prone to errors caused by read/write disturb and data retention [3], [4]. Error correction codes (ECCs) (e.g., low density parity code) have been applied to SSDs for correcting read errors and thus preventing uncorrectable bit errors [5].

However, ECCs cannot recover the corrupted data from device-level failures [3]. To enhance SSD reliability, RAID (Redundant Array of Independent Disks) has been applied inside SSD, as parallelism between chips or channels[1] can offer supports to implement RAID into a single SSD [4]. Moreover, some vendors have applied the RAID technology in their SSD products [6], and several researches have studied optimization on RAID-enabled SSDs [7], [8].

As a level of RAID, specially, RAID-4 organizes the data as stripes, where each stripe has N data chunks and 1 parity chunk (termed as the N+1 structure), and the parity chunk is XORed with the corresponding data chunks. Because of the nature of in-channel update, all updates on data/parity chunks must be completed on the same channels for maintaining the stripe structure. Moreover, all data chunk updates must renew the corresponding parity chunk of stripe, which results in bottleneck on the parity channel of SSD. Therefore, RAID-5 has been introduced, which divides the parity chunks to each channel, to address the issue of balancing parity distribution among all RAID components [9].

[1]We use channel-level RAID as the illustration in the paper, and a (data/parity) chunk is normally referred to a page in RAID-enabled SSDs.

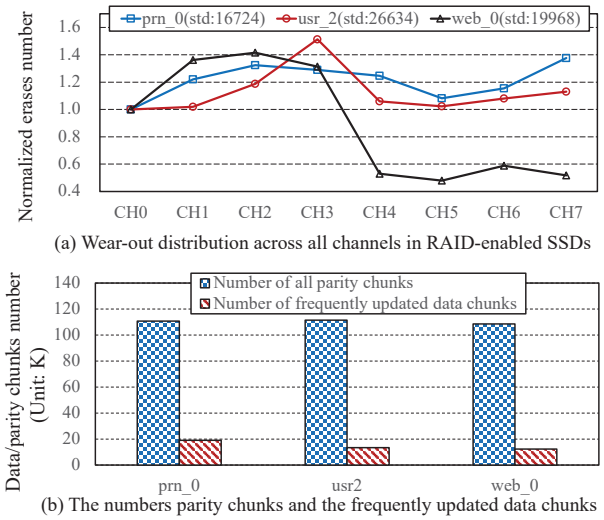


Fig. 1: Observations in the RAID-5 implementation of 7+1.(a) the wear-out distribution in term of erase among SSD channels (normalized to *CH0*), with standard deviations. (b) the comparison of the number of parity chunks and the number of hot data chunks that have more updates than the average updates on parity.

However, the RAID-5 implementation ignores that different stripes have varied access hotness on their data, so that the channels allocated with more hot data/parity chunks must endure more I/O workloads and even wear out faster [10]. In order to balance wear-out (in term of erase) and I/O workloads over all channels of RAID-5 enabled SSDs, we propose out-of-channel data placement to satisfy an update (write) request, by considering the wear-out status and I/O intensity of SSD channels, as well as the access hotness of data/parity chunks. In summary, it makes the following three contributions:

- We introduce out-of-channel data placement to reallocate the data/parity chunk(s) when updating the relevant data stripes, for balancing both workloads of wear-out and I/O across all RAID components (i.e. channels) in SSDs.
- We build a mathematical model to separately measure the balance levels of wear-out and I/O workloads over all RAID components. After that, we exchange channel locations of parity/data chunks in the same stripe, if it can better balance the workloads in RAID-enabled SSDs.

978-3-9819263-7-8/DATE23/© 2023 EDAA

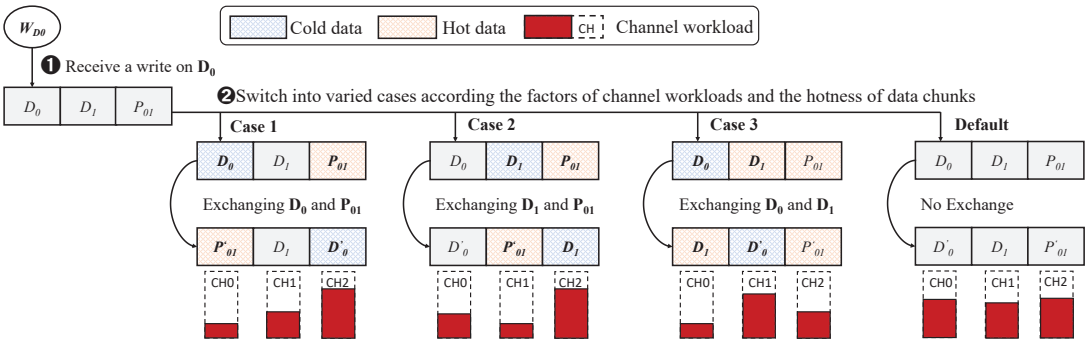


Fig. 2: The high level overview of out-of-channel data placement. For illustration simplicity, we assume the RAID-enabled SSD consists of 3 channels, with the 2+1 stripe structure.

- We evaluate our proposal by replaying several disk traces of real-world applications on the simulated RAID-enabled SSDs. As measurements indicate, our method reduces I/O response time by 29.9%, and cuts down the coefficient of variation for the number of channel-level wear-outs and I/Os by between 15.0% and 90.4%, compared to state-of-the-art methods.

The rest of paper is organized as follows. Section II introduces related work and our motivations. Section III designs the proposed method of out-of-channel data placement. The evaluation methodology and experimental results are presented Section IV. Finally, the paper is concluded in Section V.

II. Background and Motivation

A. Background and Related Work

ECCs have been commonly used in modern high density SSDs to recover the bit errors, for example, Low Density Parity Check Code (LDPC) can easily cover RBERs at the cost of additional latency through read retries [11]. However, advanced ECCs cannot correct chip/channel-level failures inside SSDs [7], [12]. Thus, parity-based RAID schemes have been introduced to address the problem. Though enabling *RAID-5* seems to increase SSD reliability, it doubles write operations as every write operation on the data chunk leads to another write on the parity chunk (termed as *write penalty*) [7].

On the other side, the memory updates follow the spatial locally of reference, so that a majority of writes are destined to only a small portion of application data, creating an extremely unbalanced write distribution [13]. Consequently, the blocks holding the frequently updated data/parity chunks will wear out much sooner than the rest of blocks, leading to less available capacity of SSDs and shorter lifetime.

With respect to this issue of wear-out unevenness, Change et al. [14] have proposed to migrated static or infrequently updated data so as to spread out the wear-leveling actions over the entire physical address space. Considering the parity chunks are frequently updated and the RAID components may have varied wear-out states in RAID systems, Du et al. [10] proposed enhancing the endurance and performance of SSD RAID, by allocating more parity to younger RAID components and less parity to older ones, We cannot ignore, but, hot stripes have more updates on their data chunks, in contrast to that on the parity chunks of cold stripes.

B. Motivation

Because all updates data/parity chunks have to be completed on their original channels (i.e. in-channel updates), which cause imbalanced I/O workloads across all RAID components [15]. Figure 1(a) shows wear-out differences among channels in a conventional RAID5-enabled SSD after running some benchmarks (see Section IV-A for details), and we see significant wear-out dissimilarity among channels.

Furthermore, in order to verify that certain hot data chunks may endure more updates than the parity chunks of cold stripes, we recorded the update frequency of data chunks and parity chunks separately. Figure 1(b) presents the distribution results of data chunk updates, when comparing to the average updates of all parity chunks. As seen, 11.4%-17.1% of data chunks endured more update operations, in contrast to the average updates on all parity chunks.

Such observations motivate us to study on balancing workloads of wear-outs and I/O requests across all RAID components, through adaptively exchanging channel locations of the parity chunk and data chunks belonging to the same stripe, according to their access frequency and SSD use states.

III. Design and Implementation of *Out-of-channel*

A. System Overview

The basic idea of our approach, called as *Out-of-channel*, is to swap channel locations of the data/parity chunks associating with the same data stripe in accordance with certain conditions, when servicing an update request. To this end, we first build an assessment model to measure balance levels of all channels of RAID-enabled SSDs, including the wear-out balance and the I/O balance. Then, it supports exchanging channel locations of data/parity chunks of the same stripe, according to the different balance scenarios. Consequently, our method can achieve a balanced state over all channels.

Figure 2 shows a high-level overview of servicing an update request of W_{D0}, with the support of our proposed scheme of *Out-of-channel*. As seen, it classifies the balance state of stripe-involved channels into 4 cases. Then, we can decide triggering a location exchange or not in the data stripe in three

TABLE I: Notation Descriptions

Symbol	Explanation
N	The number of channel in SSDs
P	The number of stripes in SSDs
α	The coefficient of wear-out workload
β	The coefficient of I/O workload
CH_i	The ith channel of SSD
a_{rs}	Write counts of s_{th} page in r_{th} stripe
b_{rs}	Read counts of s_{th} page in r_{th} stripe
W_r	Wear-out (erase) workload on CH_r
L_r	I/O workload on CH_r
$\bar{W}$	Average wear-out workload of all CH
$\bar{L}$	Average I/O workload of all CH
U_t	The imbalance degree of wear-out in SSDs
V_t	The imbalance degree of I/O in SSDs

of cases, according to the current balance state of wear-out and I/O workloads over all relevant channels.

B. Balance Assessment Model

To precisely guide out-of-channel data placement, we construct a mathematical model for assessing the balance degree of wear-out and I/O workloads over stripe-involved channels. Table I summarizes the symbols and their definitions used in the model.

Assuming the SSD device consists of N channel, labelling as CH_0, CH_1, ..., CH_{N-1}, the stripe structure spans K channels, and there are P stripes in total. The number of occurred write and read requests on the s_{th} page of r_{th} stripe, are represented as a_{rs} and b_{rs}.

Equations 1 and 2 define the level of wear-outworkload and I/O workload on CH_r, respectively.

$$W_r = \sum_{S=0}^{P-1} a_{rs} \tag{1}$$

$$L_r = \sum_{S=0}^{P-1} (a_{rs} + \delta b_{rs}) \tag{2}$$

where δ is hardware dependent, and means the ratio of write latency to read latency with the page granularity.

Next, we can get the average wear-out and I/O workloads of all channel in SSDs, with Equations 3 and 4:

$$\bar{W} = \frac{1}{N} \sum_{r=0}^{N-1} W_r \tag{3}$$

$$\bar{L} = \frac{1}{N} \sum_{r=0}^{N-1} L_r \tag{4}$$

After that, we can use two indicators of U_t and V_t to assess the wear-out and I/O balance level of all channels in the RAID-enabled SSDs, at the time point of t.

$$U_t = \begin{cases} 0, & |W_i - W_j| \le \alpha \bar{W} \quad \forall(i,j) \\ \frac{1}{N} \sum_{i=0}^{N-1} |W_i - \bar{W}|, & other\ cases \end{cases} \tag{5}$$

where the coefficient of α represents the endurable degree of imbalanced wear-out workload.

$$V_t = \begin{cases} 0, & |L_i - L_j| \le \beta \bar{L} \quad \forall(i,j) \\ \frac{1}{N} \sum_{i=0}^{N-1} |L_i - \bar{L}|, & other\ cases \end{cases} \tag{6}$$

where the coefficient of β means the endurable degree of imbalanced I/O workload.

We argue that the SSD device is in an even state of channel-level wear-out, if U_t is equal to 0. Otherwise, we regard the wear-out distribution is imbalanced, and a larger value of U_t indicates a greater difference of wear-out among SSD channels. Similarly, the value of V_t addresses the balance level of I/O workload across all channels of SSD.

C. Out-of-channel Placement with Data Exchanges

When servicing an update request on the stripe, our proposal considers whether exchanging channel locations of data/parity chunks or not, according to the balance state of SSD and the access hotness of data/parity chunks. Specifically, it compares the reduction of balance indicators of U_t and V_t after location exchange. Note that we use the historical access frequency to reflect the future access frequency when estimating the balance indicators after exchange. Then, it triggers an exchange operation if the exchange can yield the largest reduction of two balance indicators.

By referring back to Figure 2, we depict **four** scenarios of out-of-channel data placement, by considering the workload balance of SSD channels and the access frequency of the involved data/parity chunks:

- **Case 1 of swapping the update chunk and the parity chunk.** The updated data chunk is not frequently accessed and located in a light wear-out workload channel, meanwhile the parity chunk of stripe is frequently updated and located in a heavy workload channel.
  Since every update request must lead to a parity update, there is *no extra cost of read/write* caused by swapping the locations of the updated data chunk and the parity chunk. Then, we only refer to the factor of wear-out balance, when deciding whether triggering location exchange or not. That is, if the current value of U_t becomes smaller after the exchange operation, we conduct an exchange operation, by inserting W_{D_0} to the waiting queue of CH_2, and creating a write request on CH_0 to update the parity chunk.
- **Case 2 of swapping other data chunk and the parity chunk**. The data update channel of CH_0 endures more wear-out and I/O workloads than CH_1 that holds another data chunk of the stripe, while the parity chunk is located in the heaviest workload channel of CH_2.
  We prefer to consider the exchange operation on the data chunk of D_1 and the parity chunk of P_{01}, if the balance indicators will become less than before. Since this kind of location exchange will lead to one more write request, we apply stronger constraints on triggering such exchanges, see Algorithm 1 for details.

- **Case 3 of swapping the update chunk and other data chunk**. Although the parity chunk of P_{01} is the hottest data in the stripe, the relevant channel (i.e. CH_2) has a light workload. On the other side, two data chunks of D_0 and D_1 has distinct access frequency and located in the channels having varied balance level of workloads.
  Then, we will perform an location exchange on the update chunk of D_0 and another data chunk of D_1 in the stripe, if it can contribute to reductions of balance indicators.
- **Default**: There are no obvious differences in wear-out and I/O workload balance across all channels, or the access hotness of data/parity chunks of stripe is not noticeably distinct. In the default situation, we do not carry out exchange of channel locations, when responding an update request.

D. Implementation Specifications

Algorithm 1 presents the implementation specifications on the newly proposed *Out-of-channel* method, which intends to yield a balanced workload across all channels in RAID-enabled SSDs. As seen, Lines 2-10 identify the process of obtaining the values of balance indicators of U_t or V_t, with relevant inputs. Then, Lines 15 and 16 compute the results (i.e. U_{t_0} and V_{t_0}) of wear-out balance indicator before exchange. Corresponding to **Case 1**, Lines 18-21 show computing the difference between U_{t_0} and U_{t_1} (i.e. the future wear-out balance indicator assuming the exchange was done.), to decide whether a location exchange of the update data chunk and the parity data chunk should be triggered or not. Corresponding to **Cases 2** and **3**, it calculates the values of U_{t_1} and V_{t_1}, and carries out relevant data exchanges indeed, if the gap is beyond the threshold, as illustrated in Lines 23-33. Otherwise, it chooses the common way to update the stripe.

Algorithm 1: Out-of-channel data placement

Input: args of Req, W_Ar, L_Ar, α, β
Output: null;

```
1  /*Compute U_t or V_t with inputs*/
2  Function get_balance(WL_Ar, α_β_val)
3      threshold = Avg(WL_Ar) × α_β_val;
4      /*Traverse all channel to get max/min value*/
5      W_L_MAX=find_max(WL_Ar);
6      W_L_MIN=find_min(WL_Ar);
7      if |W_L_MAX − W_L_MIN| ≤ threshold then
8          return 0; // Balance
9      end
10     return 1/N Σ_{i=1}^{N−1} |WL_Ar[i] − Avg(WL_Ar)|;
11
12 /*Main function starts*/
13 if Req is write on D_i then
14     /*Compute U_{t_0} and V_{t_0} (current value)*/
15     U_{t_0} = get_balance(W_Ar_now, α);
16     V_{t_0} = get_balance(L_Ar_now, β);
17     /*Compute U_{t_1}(the future value after exchange of
         update chunk and parity chunk) */
18     U_{t_1} = get_balance(W_Ar_after1, α);
19     if U_{t_0} · (U_{t_0} − U_{t_1}) > 0 then
20         SwapCase1(D_i, P, Req); return;
21     end
22     /*Compute U_{t_1} and V_{t_1}(future vlaues after
         exchange of other data and parity chunk) */
23     U_{t_1} = get_balance(W_Ar_after2, α);
24     V_{t_1} = get_balance(L_Ar_after2, β);
25     if U_{t_0} · (U_{t_0} − U_{t_1}) > 0 and V_{t_0} · (V_{t_0} − V_{t_1}) > 0
         then
26         SwapCase2(D_j, P, Req); return;
27     end
28     /*Compute U_{t_1} and V_{t_1}(future values after
         exchange of two data chunks) */
29     U_{t_1} = get_balance(W_Ar_after3, α);
30     V_{t_1} = get_balance(L_Ar_after3, β);
31     if U_{t_0} · (U_{t_0} − U_{t_1}) > 0 and V_{t_0} · (V_{t_0} − V_{t_1}) > 0
         then
32         SwapCase3(D_i, D_j, Req); return;
33     end
34     /*Default routine to update stripe*/
35     CommonUpdate(Req);
36 end
```

IV. Experiments and Evaluation

A. Experiment Settings

Since the *SSDSim* simulator has a diverse set of configurations and its validation accuracy against a real hardware platform [16], we employed it replaying the selected disk traces of real-world applications, for evaluating our proposed scheme. We applied our method as a part of *SSDsim*, for supporting location exchanges of data/parity chunk on SSD channels, by considering both wear-out and I/O workload balance, and the access frequency of chunks. Table II presents the settings of SSDsim in our experiments.

We employed 6 widely used block I/O traces in the tests that cover a wide range of write ratios. Among them, three traces of *usr_2*, *web_0* and *prn_0* are from the block I/O trace collection of Microsoft Research Cambridge [17]. Another three recently block I/O traces are collected from a part of an enterprise virtual desktop infrastructure (VDI) [18]. Specifically, they are additional-01-2016021620-LUN6 (*lun0)*, additional-03-2016021618-LUN3 (*lun1*), and additional-03-2016021614-LUN0 (*lun2*). The specifications on the selected traces are reported in Table III, and the metric of **Freq Wr** means the ratio of data pages that have been updated not less than 4 times, to all data pages. We set α and β as the outcomes of the pre-defined thresholds of T_α and T_β divided by the average request size in the previous time window. After sensitive analysis, we suggest configuring the time window size as 8,192, T_α as 500, and T_β as 2,000.

Besides, we used the following comparison counterparts for measuring the performance of our proposed mechanism:

- *Baseline*: which is the conventional RAID-5 implementation. It distributes parity onto all SSD channels evenly, to ensure wearing out balance across of RAID components [19].

TABLE II: Experimental settings of *SSDsim*

Parameters	Values	Parameters	Values
Channel size	8	*Read latency*	0.045ms
Chip size	4	*Write latency*	0.7ms
Plane size	1	*Erase latency*	3.5ms
Block per plane	512	*XOR latency*	0.019ms
Page per block	64	*GC threshold*	10%
Page size	8KB	*RAID level*	5
FTL scheme	Page level	*Stripe struct.*	7+1

TABLE III: Specifications on traces (ordered by write ratio)

Traces	Req #	Wr Ratio	Wr Size	Freq Wr
usr_2	10,570,046	18.9%	43.8KB	35.7%
lun0	944,526	31.5%	21.1KB	6.1%
lun1	1,289,238	49.3%	22.1KB	8.5%
lun2	949,064	52.8%	20.1KB	10.7%
web_0	2,029,945	70.1%	15.0KB	60.7%
prn_0	5,585,886	89.2%	11.1KB	32.6%

- *CSWL*: which adopts age-driven parity distribution, to make all channels of RAID-enabled SSD achieving an even wear-out distribution [10]. The main idea is to let the younger channels undertaking more parity chunks.
  We argue *CSWL* is the most related work, aiming at yielding a wear-out balance across all RAID components. But it ignores that many data chunks are more frequently updated than the parity chunks of cold stripes.
- *Out-of-channel*: which is the proposed data placement scheme. More specially, the native design of *Out-of-channel* needs recording the read/write counts with the fine granularity of data page, which must result in non-negligible space overhead. Thus, we offer an empirical implementation, labeled as *Out-of-channel**, that records the read/write counts with the granularity of block, by regarding all data pages in the block have a similar read/write frequency.

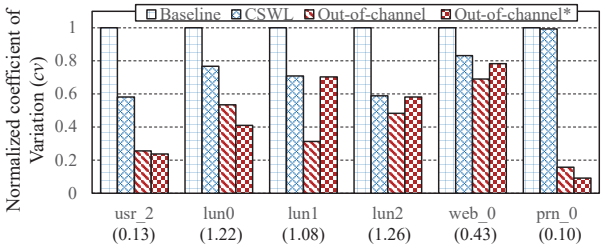
B. Results and Discussions

To measure validity of the proposed mechanism that supports reallocating the parity/data chunks, we use the following three metrics in our tests: (a) *Wear-out and I/O workload balance*, (b) *I/O latency*, and (c) *Exchange case distribution*.

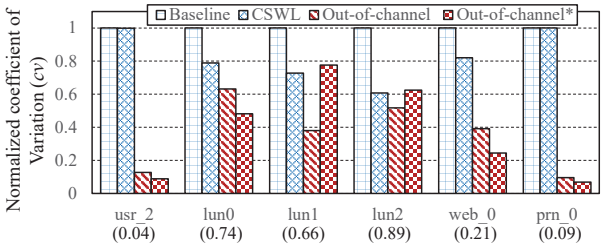
1) Wear-out and I/O workloads balance: We compute the metric of coefficient of variation (*cv*) [20] for wear-outs and I/Os in all channels, to quantify the degree of imbalance level over all SSD channels. A larger value of *cv* implies that all channels have varied workloads, while a smaller value of *cv* indicates that all channels have a similar workload level.

Figures 3(a) and 3(b) respectively show the *cv* results of wear-out and I/O workloads over all channels. As seen, *Out-of-channel* performs the best, and reduces *cv* values of wear-out by 59.4% and 43.5% on average, compared to *RAID-5* and *CSWL*. Besides, our proposal reduces *cv*s of I/Os by 17.0%-84.1% in contrast to *CSWL*. This is because *Out-of-channel* supports location exchanges, which can better improve the wear-out and I/O workload balance.

Another clue is that our empirical implementation of *Out-of-channel** yields a comparable improvement on workload



(a) Coefficient of variation with wear-outs



(b) Coefficient of variation with I/Os

Fig. 3: Comparison of balance across all channels with respect to wear-out workload (a) and I/O workload (b).

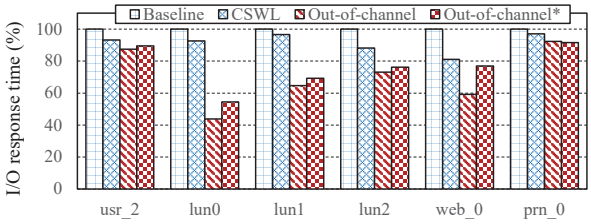


Fig. 4: Comparison of I/O response time.

balance, to *Out-of-channel* after replaying the traces, verifying it is practicable to take the count of workload on the block to reflect the access hotness of its pages.

2) I/O performance: I/O response time is the most critical indicator to reflect SSD performance, and Figure 4 reports the normalized results. As shown, *CSWL*, *Out-of-channel* , and *Out-of-channel** , can cut down the I/O time by 8.5%, 29.9%, and 23.6% respectively. We consider more balanced workloads can make full use of the inside parallelism of SSDs, benefiting to faster responses for I/O requests.

Moreover, to reveal the reason for our methods can yield better I/O performance than *CSWL*, we recorded the number of extra writes caused by location exchanges, as the exchange may require creating an extra write and more extra writes must impact normal I/O processing. Figure 5 presents the results. Compared with *CSWL*, *Out-of-channel* and *Out-of-channel** reduce the extra writes by 98.8% and 89.1%, respectively.

3) Model verification with exchange cases distribution: We have analyzed the distribution of exchange cases according to the assistance of the balance assessment model, and Figure 6 presents the results. As seen, *Out-of-channel* occupies 96.9%, 1.3%, and 1.7% of total location exchanges in **Cases** 1-3 respectively. The most interesting clue is that our empirical implementation of *Out-of-channel** greatly increases the number of **Case 2** by 11 times, comparing to *Out-of-channel*. We argue this is because block-level workload counters weaken

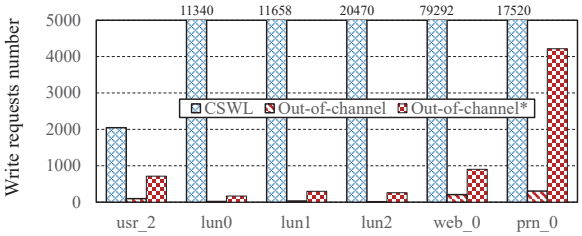


Fig. 5: Comparison of extra write reduction caused by location exchanges of data/parity chunks.

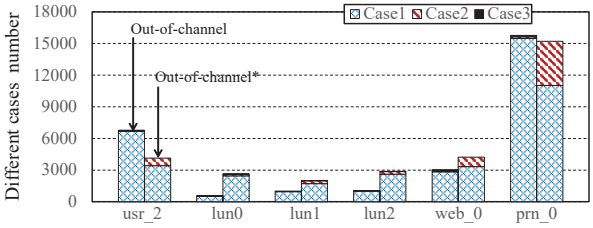


Fig. 6: Exchange cases distribution of our proposals.

access difference between pages in the same block, which transfers more exchanges of **Case 1** into **Case 2**.

4) Overhead: The main memory overhead of our proposals is due to the additional storage required for the parameters used by the balance assessment model. Figure 7 presents the results of space overhead of three optimization schemes. As seen, *CSWL* requires the least space overhead, as it only records erase numbers of each channel,consuming no more than `0.3`KB memory space. The important clue is that our empirical implementation of *Out-of-channel** can greatly reduce the space overhead by `98.4%`, in contrast to *Out-of-channel*, corresponding to less than `54.4`KB memory overhead.

With respect to time overhead, the proposed approaches only require to additionally compute the values of balance indicators, which does not introduce noticeable time overhead and remains negligible.

V. Conclusion

This paper proposes an out-of-channel data placement scheme, for achieving wear-out and I/O workload balance among all RAID components in channel-level RAID SSDs. To this end, we build an assessment model, for measuring the workload balance level across all SSD channels. Thus, it can trigger a location exchange of data/parity chunks in the same data stripe, if the exchange operation can benefit to workload balance. The experimental results show our proposal of *Out-of-channel* substantially decreases the I/O latency by `29.9%` on average, and smooths workload balance by more than `51.5%`, in contrast to state-of-the-art methods.

Moreover, considering the balance assessment model needs recording parameters on the granularity of data page, which results in non-negligible space overhead, we also present an empirical implementation of *Out-of-channel**, that only records the parameters on the block granularity. The tests illustrate that it is comparable to the model-based implementation of *Out-of-channel* when running the most benchmarks.

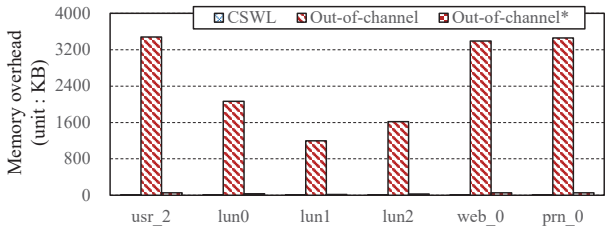


Fig. 7: Comparison of memory overhead.

Acknowledgment

This work was partially supported by "National Natural Science Foundation of China (No. 61872299, No. 62032019)", and "the Natural Science Foundation Project of CQ CSTC (No. cstc2021ycjh-bgzxm0199)".

References

[1] Kishani M, Ahmadian S, Asadi H. A modeling framework for reliability of erasure codes in ssd arrays. In ***TC***, 2019.
[2] Cui J, Liu J, Huang J, et al. SmartHeating: On the performance and lifetime improvement of self-healing SSDs. In ***TCAD***, 2020.
[3] Balakrishnan M, Kadav A, Prabhakaran V, and Malkhi D. Differential raid: Rethinking raid for ssd reliability. In ***TOS***, 2010.
[4] Bianca Schroeder, Raghav Lagisetty, and Arif Merchant. Flash reliability in production: The expected and the unexpected. In ***FAST***, 2016.
[5] Yu Cai, Saugata Ghose, Erich F. Haratsch, Yixin Luo, and Onur Mutlu. Error characterization, mitigation, and recovery in flash-memory-based solid-state drives. In ***PIEEE***, 2017.
[6] P320h 2.5-Inch PCIe NAND SSD Features. https://www.micron.com/-/media/client/global/documents/products/data-sheet/ssd/p320h_2_5.pdf
[7] Jun Li, Zhibing Sha, Zhigang Cai, François Trahay, and Jianwei Liao. Patch-Based Data Management for Dual-Copy Buffers in RAID-Enabled SSDs. In ***TCAD***, 2020.
[8] Sha Z, Li J, Cai Z, et al. Degraded mode-benefited I/O scheduling to ensure I/O responsiveness in RAID-enabled SSDs. In ***TODAES***, 2022.
[9] Zhou Y, Wu F, Huang W, et al. LiveSSD: A low-interference RAID scheme for hardware virtualized SSDs. In ***TCAD***, 2020.
[10] Du Y, et al. CSWL: Cross-ssd wear-leveling method in ssd-based raid systems for system endurance and performance. In ***JCST***, 2013.
[11] Tanakamaru, S., Yanagihara, Y., & Takeuchi, K. Error-prediction LDPC and error-recovery schemes for highly reliable solid-state drives (SSDs). In *IEEE journal of solid-state circuits (**IJSSC**)*, 2013.
[12] Schroeder B, Merchant A, Lagisetty R. Reliability of NAND-based SSDs: What field studies tell us. In ***PIEEE***, 2017.
[13] Zhou, P., Zhao, B., Yang, J., and Zhang, Y. A durable and energy efficient main memory using phase change memory technology. In *ACM SIGARCH computer architecture news*, 2019.
[14] Chang Y, Hsieh J, and Kuo T. Improving flash wear-leveling by proactively moving static data. In ***TC***, 2009.
[15] Pan, W. and Xie, T. A mirroring-assisted channel-RAID5 SSD for mobile applications. In ***TECS***, 2018.
[16] Hu Y, Jiang H, Feng D, et al. Performance impact and interplay of SSD parallelism through advanced commands, allocation strategy and data granularity. In ***ICS***, 2011.
[17] Narayanan D, Donnelly A, Rowstron A. Write off-loading: Practical power management for enterprise storage. In ***TOS***, 2008.
[18] Lee C, and Kumano T, et al. Understanding storage traffic characteristics on enterprise virtual desktop infrastructure. In ***Systor***, 2017.
[19] Li H, Putra M, and Shi R, et al. lODA: A Host/Device Co-Design for Strong Predictability Contract on Modern Flash Storage. In ***SOSP***, 2021.
[20] Kim Y, Lee J, and Oral S, et al. Coordinating garbage collectionfor arrays of solid-state drives. In ***TC***, 2012.