



Combining Cache and Refresh to Optimize SSD Read Performance Scheme

Jinli Chen^{1,2,3,4}, Peixuan Li^{1,2,3,4}, and Ping Xie^{1,2,3,4}(✉)

¹ College of Computer of Qinghai Normal University, Xining 810016,
People's Republic of China
xieping@qhnu.edu.cn

² The State Key Laboratory of Tibetan Intelligent Information Processing and Application,
Xining 810016, People's Republic of China

³ The Key Laboratory of Internet of Things of Qinghai Province, Xining 810016,
People's Republic of China

⁴ Academy of Plateau Science and Sustainability, Xining 810016, People's Republic of China

Abstract. In the era of continuous advances in flash technology, the storage density of NAND flash memory is increasing, but the availability of data is declining. In order to improve data availability, low-density parity check codes (LDPC), which have been highly corrected in recent years, are used in flash memory. However, although LDPC can solve the problem of low data availability, it also brings the problem of long decoding time. Moreover, the LDPC decoding delay time is related to its decoding level, and the higher the LDPC decoding level, the longer the delay time. Long decoding delays can have an impact on the read performance of the flash. Therefore, in order to improve the speed of reading data in flash memory, this paper proposes a scheme combining cache and flushing, the main idea of which is to use the cache to reduce LDPC decoding time, and at the same time refresh the pages with high latency replaced by the cache, so that the pages with high latency can be restored to the state of low latency pages. Experimental results show that this scheme can significantly reduce LDPC decoding delay and improve data availability with less overhead, and optimize the read performance of flash memory. The experimental results show that compared with the original strategy, the average response time is reduced by 24%, and the average IOPS value is increased by 32%.

Keywords: Cache · Refresh · LDPC · Flash memory

1 Introduction

Solid-state drives (SSDs) based on NAND flash memory are non-volatile and have slowly developed into dominant secondary storage due to their small size, high performance, and low energy consumption [1, 2]. In order to increase the storage capacity of SSD, the memory cell density of flash memory has slowly increased from single-level cell flash memory (SLC) to multi-level cell (MLC) or three-level cell (TLC). As the memory density of flash memory increases, the gap between flash memory cells becomes smaller and smaller [3, 4], and the voltage of the memory cell is more likely to shift. As a result, the endurance of flash memory will be reduced, and the bit error rate will increase.

With the increasing density of flash memory cells leading to the continuous increase of bit error rate, users have higher and higher requirements for erasure coding in flash memory, and BCH erasure coding (Bose, Ray Choudhury, and Huo Quinheng) can no longer meet people's requirements for flash memory reliability [5]. In recent years, LDPC (Low-Density Parity-Check Codes), as a high-cost, error-correcting ECC, has become the default ECC scheme for the most advanced flash-based storage devices [6]. However, although the error correction ability of LDPC coding is strong, it also has disadvantages. LDPC code is highly complex to compile and requires up to 7 read retries to take full advantage of its error correction capabilities. Larger read-retry operations contain more fine-grained read voltages. In order to successfully decode, this data may waste more time to decode, and the long decoding delay reduces the read performance of the flash memory. Therefore, in order to improve the read performance of flash memory, this paper starts from reducing the number of read retries in the LDPC decoding process of the LDPC decoding process, and proposes a scheme to apply cache and flushing to the LDPC decoding process of flash memory, and experiments show that this scheme improves the read performance and prolongs the life of flash memory to a certain extent.

2 Research Background and Motivation

2.1 Introduction to the Basics of Flash Memory

Flash memory is a non-volatile memory device, also known as solid-state memory [7], which is widely used in various devices. Flash memory consists of many memory cells, each of which can store one or more bits. A storage unit is usually composed of a transistor and a capacitor, and the transistor is used to control the charging and discharging of the capacitor to realize data storage. Flash memory can be divided into various types according to the structure and working principle of the storage unit. NAND flash memory is the most common type of flash memory, and it is usually used in large-capacity storage media, such as solid-state drives (SSD). NAND flash memory connects multiple storage units into a block, multiple blocks form a plane, multiple planes form a core, and multiple cores form a chip. Read and write operations for NAND flash are performed on a page-by-page basis, while erase operations are performed on a per-block basis [8]. Therefore, when a storage unit needs to be updated or deleted, the block where the storage unit is located needs to be erased, and the data in other data pages is copied to the new block. Compared to traditional hard drives, flash memory offers faster read and write speeds, lower power consumption, smaller size, and higher reliability.

2.2 LDPC in Flash Memory

In order to solve the reliability problem of flash memory, BCH code with high performance and error correction capability is applied to flash memory [9]. However, as the reliability of flash memory decreases, the error correction capability of BCH can no longer meet the reliability requirements. To meet the reliability requirements of state-of-the-art flash systems, LDPC codes are recommended [10]. LDPC codes, also known as low-density parity codes, were proposed in the 1960s and have been widely used in storage in recent years [11]. LDPC code has high error correction ability, can achieve high

error correction performance, and high coding efficiency, so it is used in flash memory devices. In the following, we will discuss the decoding process of LDPC.

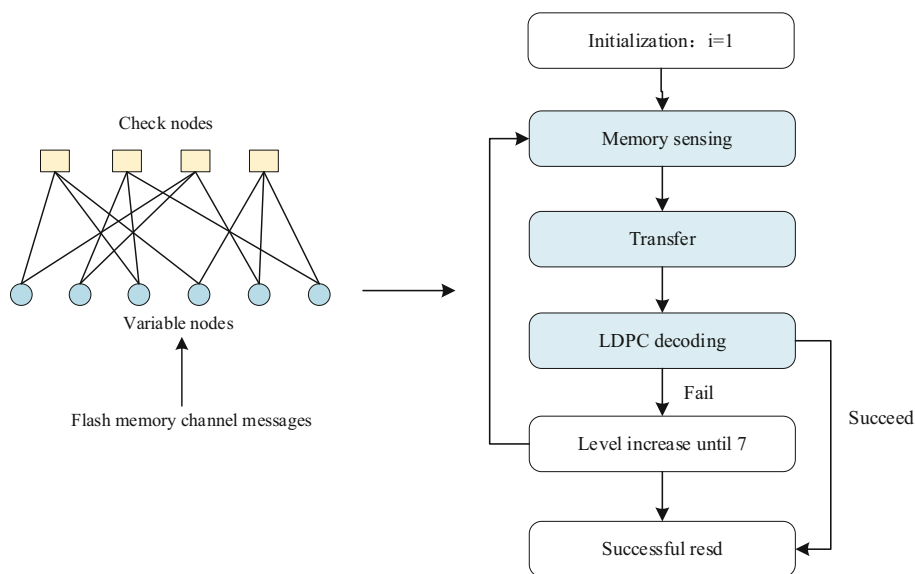


Fig. 1. Tanner diagram of LDPC

LDPC codes generally use an iterative belief propagation decoding algorithm to complete the input of decoding information. The basic idea is to XOR the received code word and code book to obtain the constraints involved in each check node, and then the algorithm converts these constraints into a Tanner graph, where nodes represent the variable nodes and check nodes of the code word, and edges represent the relationships between them, as shown in Fig. 1. When decoding fails, multiple read retries are required, as shown in Fig. 1. Although increasing the number of read retries can increase the probability of successful decoding. However, the decoding delay also increases significantly.

2.3 Research Motivation

LDPC code is an error-correcting code that has been commonly used in storage systems in recent years to correct data errors. The decoding level of LDPC code refers to the trade-off between error correction performance and computational complexity in the process of LDPC code decoding, and the decoding level is mainly adjusted to achieve the optimal balance between decoding performance and computational complexity. The decoding level of LDPC code can be divided into seven levels, among these seven levels, the higher the decoding level, the higher the success rate, but the longer the data decoding delay time, as shown in Fig. 2.

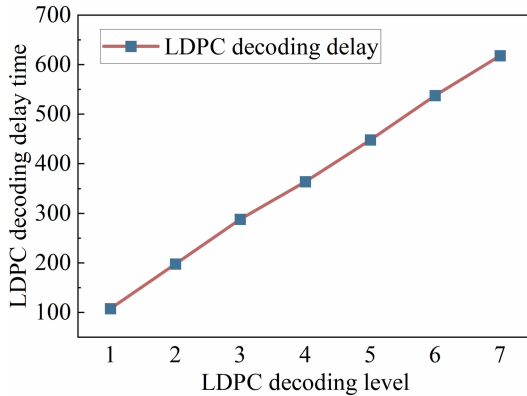


Fig. 2. LDPC different decoding delays.

The LDPC decoder can handle data with a high bit error rate, but if the data has a high bit error rate, the decoding time overhead of LDPC will be large. There are two reasons, first, to handle more data errors, more accurate data entry information is required. This process requires multiple inputs of different read reference voltages. Improving the accuracy of input information requires additional awareness, which results in additional flash read latency and data transfer delays. Secondly, the iterative decoding algorithm of LDPC will iterate for each decoding, and the decoding time of each decoding will iteratively accumulate until all errors are corrected. When the bit error rate of data is high, the time overhead of decoding will be very large. For these two reasons, this paper proposes a new method combining caching and refreshing. The main idea is to utilize caching to reduce LDPC decoding time for pages with high latency, thereby reducing the performance overhead of LDPC.

3 Related Work

In recent years, as the storage density of flash memory cells increases, the size of storage devices and the gap between flash memory cells gradually decrease [3, 4], and the possibility of data errors gradually increases. The disadvantages of LDPC codes in flash memory are more prominent. To this end, in order to reduce the decoding time of LDPC, many optimization methods have been proposed. There are many of these methods, which are briefly introduced below.

The first approach is to improve the reliability of flash memory by using refresh [12–14]. For the first method, most of them suggest to refresh data by in-place charging [5] or non-in-place updating [12, 15]. In the literature [12], a lightweight data refreshing method is utilized, which primarily relies on soft sensing to reduce read latency. By leveraging the characteristics of read-hot data, only a small portion of the data is refreshed, requiring only a few PE cycles. Furthermore, the refresh granularity of this solution is performed at the page level. By only moving one or a few hot pages during each refresh operation, it can reduce the cost of heavy block migration in the current refresh method.

This literature borrows the idea of refreshing to reset the long LDPC read latency and soft sensing back to the cheapest latency and hard sensing, in order to reduce RBER (Raw Bit Error Rate) and improve flash memory read performance.

The second method is to use caching [16] to reduce the process of LDPC read retries, thereby improving the read performance of flash memory. The main idea of this scheme is to correct most of the errors that occur in the requested Flash page before the LDPC decoding process begins, by caching the detected errors. This method improves the read performance of flash memory by speeding up the decoding process of LDPC.

Both solutions reduce LDPC decoding time, but they have their own respective drawbacks. Although refresh operation can reduce the decoding latency of LDPC, it also incurs additional migration costs, especially in the case of frequent refreshes. Using caching can indeed reduce the decoding time of LDPC. However, in the early stages of flash memory, the bit error rate is low, and using LDPC hard decision decoding can recover most of the data. Therefore, in the early stages of flash memory, using a larger cache would result in wasted space. Therefore, this article proposes an optimized LDPC read-retry scheme that combines refresh and caching. In this proposed scheme, refresh operations are employed in the early stages of flash memory to transform high-latency pages into low-latency states. Due to the low bit error rate in the early stages of flash memory, there are not many pages with high latency. As a result, the number of refresh operations required is minimal, significantly reducing the cost associated with refreshing. In the later stages of flash memory, cache techniques can be employed to store pages with high latency in a cache. When the same read request occurs again, the data can be directly retrieved from the cache, reducing LDPC decoding time and improving space utilization.

4 Design of the Scheme

Due to the increase in flash memory density, the reliability of flash memory data has decreased. To enhance data reliability, LDPC (Low-Density Parity-Check) codes are applied in flash memory. But using LDPC decoding directly can introduce significant latency. The increased decoding latency is primarily caused by the multiple read retry steps in LDPC decoding. To address this issue, this paper proposes a hybrid approach combining caching and refreshing to reduce LDPC read retries. By leveraging high-latency caching pages, LDPC decoding process can be reduced, thereby improving data reading performance and reliability.

This chapter focuses on the combination of caching and flushing to optimize the performance of flash reads, the CR solution. First, the first step is to distinguish between hot and cold reading data, and apply different recovery methods to the data according to the hot and cold nature of the data. This paper distinguishes hot read data and cold read data according to the number of data reads. When the number of data reads exceeds 1, it is hot read data, otherwise it is cold read data. The second step is to migrate high-latency pages with LDPC decoding level greater than or equal to 3 to the cache for hot-read data. The third step is to refresh the replaced high-latency pages. Due to the limited size of the cache, high-latency data cannot be stored infinitely. Therefore, when the cache space is exhausted, pages in the cache that have not been accessed for a long time are replaced

according to the cache replacement algorithm, and then these pages are refreshed. This scheme reduces the LDPC decoding time of high-latency data and improves the speed of reading data from flash memory to a certain extent. The implementation of this scheme includes three parts: the determination of high-latency pages, the selection of cache size, and the selection of cache replacement algorithm. Below, I will briefly explain the choice of these three parts.

4.1 Determining the Latency Threshold T

The choice of the T threshold is critical for performance improvement. First of all, if T is set too small, a large amount of data will be put into the cache, due to the limited cache space, a large amount of data into the limited cache will cause high-latency pages to be frequently replaced, which will cause frequent refresh operations. Frequent refreshing not only brings additional read and write operations to the device, but also introduces wear and tear, which will further reduce reliability. If T is set too large, data with long access latency will significantly affect performance, especially for frequently accessed data. As stated in the review, the T-threshold should not be set too large or too small. In order to select the appropriate T, we performed experiments and analyzed the normalized delay time comparison of each load at different LDPC levels. From Fig. 3, it can be seen that the latency cost of each load is higher after LDPC decoding level 3, so setting the threshold for high latency to 3 is reasonable.

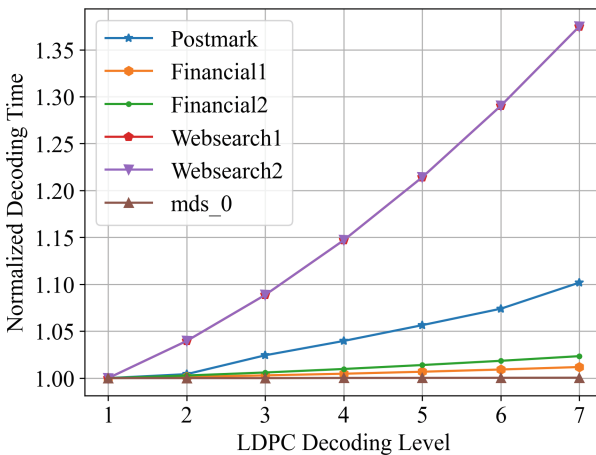


Fig. 3. Delay of each load under different LDPC decoding levels

4.2 Choice of Cache Size M

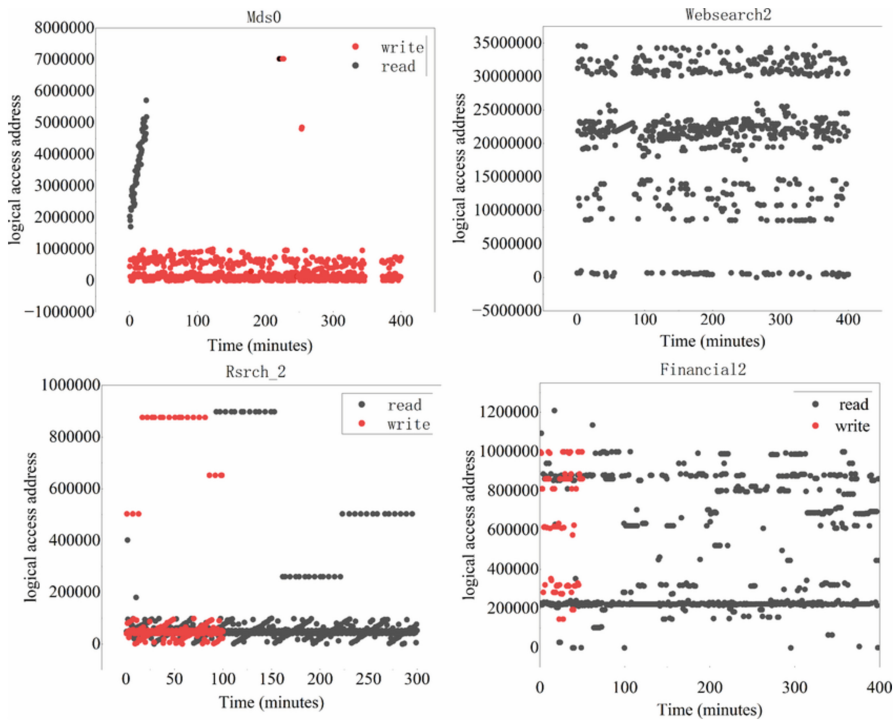


Fig. 4. Data access under different Workload

Another threshold cache size M setting of the scheme is also very important, M is not the bigger the better [17]. In this article, the size of M affects the number of refreshes, so it must be set within a reasonable range. Figure 4 intercepts access to a part of the payload Mds0, Rsrch_2, Financial2 and Websearch2. From the Fig. 4, we can see that the user's access to the data is local to the group, and through the analysis and calculation of the relevant load, it is found that only about 8% of the data is hot read data, and most of the remaining data is cold data. We set the cache size based on this.

The bit error rate of flash memory increases with the number of P/E times. As shown in Fig. 5, in the early stage of flash memory, due to the small number of P/E times of blocks, there are fewer pages with high latency. Therefore, if a fixed cache size is used all the time, it will cause a large waste of space in the early stage of the flash memory. In order to improve space utilization, a dynamic cache size adjustment scheme is proposed. The main idea is to set a P/E threshold C , if the P/E period of the block exceeds the set threshold, the block will be recorded as a late block, and count the number of blocks that exceed the P/E period threshold, when the late block is greater than or equal to half of the total number of blocks T , the size of the cache will be increased, otherwise the size of the cache will not change.

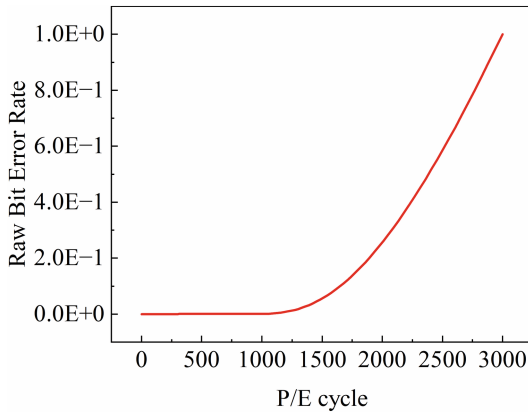


Fig. 5. RBER change graph with P/E cycle

The proposed key algorithm for dynamically adjusting the cache size is as follows: How to combine flushing and caching as shown in Algorithm 1.

Algorithm 1 Dynamic cache size algorithm

Input: P/E cycle of the block

Output: Appropriate cache size

if blk_numerases > Threshold_numerases

target_blocks++

if target_blocks \geq 1/2 total_blocks

cache size changed from 4MB to 8MB

else

end if

end if

4.3 Choice of Cache Replacement Algorithm

The function of the cache replacement algorithm is how to better manage the cache, whether it is caching in Memory or caching in SSD controllers, its main purpose is to make the cache play the maximum role as much as possible, so as to better improve the performance of the storage system. Different application scenarios and data access patterns may require different cache replacement strategies [18], and appropriate replacement strategies can be selected according to different application scenarios and requirements to achieve the best performance and efficiency [19]. Currently common cache replacement algorithms include least recently used (SLRU) algorithm, least frequently used (LFU) algorithm [20], random replacement (RR) algorithm, first-in-first-out (FIFO) algorithm, etc., each algorithm has its own advantages. Disadvantages and applicable scenarios, as shown in Table 1. In this experimental environment, due to the large read ratio of the load used, the data read access frequency is high, and the LFU algorithm

mainly selects the cache replacement according to the access frequency of the data, in order to reduce unnecessary space overhead, the LFU cache replacement algorithm is selected.

Table 1. Comparison of different cache replacement algorithms

	LFU	LRU	FIFO	Random
Main basis	access frequency	time	time	random
advantage	effective identify of hot data	high space utilization	easy to implement	easy to implement
disadvantage	low space utilization	not flexible	unable to identify hot data	low hit rate
applicable Scenario	high read frequency, low write frequency	read and write frequency is high	stable access frequency	irregular access time

4.4 CR Program

In the traditional LDPC usage process in flash memory, the user sends a write request to the SSD controller. The ECC (Error Correction Code) module in the controller temporarily stores the write request data in an I/O buffer and then encodes it using LDPC before sending it to the flash memory cells. When there is a read request from the user, the data is retrieved from the flash memory cells. The LDPC decoder is then used to recover the correct data, which is then transferred to the I/O buffer. Finally, the data is returned to the user. In the traditional approach, reading data requires LDPC decoding, and when the read data has a high error rate, the LDPC decoding time will also be longer. To reduce read latency, this solution stores high-latency pages in the cache. When the data being read is hit in the cache, it is directly accessed from the cache without going through LDPC decoding. This reduces the decoding latency of LDPC and improves the read performance of flash memory. If there is a cache miss, the data will be read from the flash memory chip unit. After that, it will go through LDPC code decoding, and once successfully decoded, the data will be returned to the user. Figure 6 illustrates the comparison of data retrieval before and after optimization.

In this scheme, the first step is to differentiate between hot and cold read data, and adopt different recovery methods for each. For hot read data, the first step is to check if there is enough space in the cache. If there is sufficient space, the hot read data is placed into the cache. If there is not enough space, the Least Frequently Used (LFU) replacement algorithm is used. The page with the least number of accesses is evicted from the cache and refreshed. Then, the newly identified hot read data is placed into the cache. When a new read request comes in, the first step is to check if the data is present in the cache. If it is a cache hit, the data is directly read and returned to the user after the read operation is completed. In this process, LDPC code decoding is not required, reducing the decoding latency of LDPC code. If there is a cache miss, the data is read from the flash memory chip unit. After that, it goes through LDPC code decoding. Upon successful decoding, the data is returned to the user. The structure of this scheme is illustrated in Fig. 7.

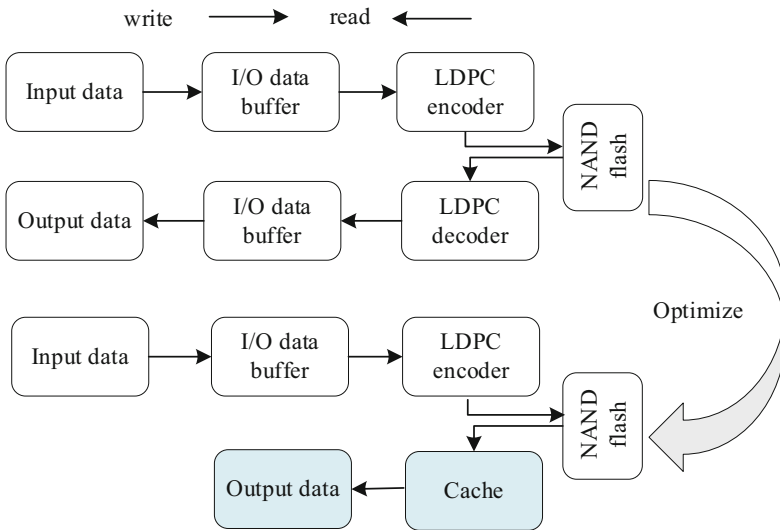


Fig. 6. Flash memory read and write process before and after optimization

The key algorithm of the caching and refreshing combined optimization scheme proposed in this paper is as follows: how to combine flushing and caching as shown in Algorithm 2.

Algorithm 2 Combination of cache and refresh

```

Input: Quantization level for LDPC soft decoding
Output: suitable decoding method
if data is hot data and high latency data
    if data_size < cache_size
        cache_add(new_blkno, new_data)
    else
        cache_replace_lfu(int blkno, void* data)
        cache_add(new_blkno, new_data)
        cache_refresh(blkno, data)
    end if
else
    direct use of LDPC decoding
end if
    
```

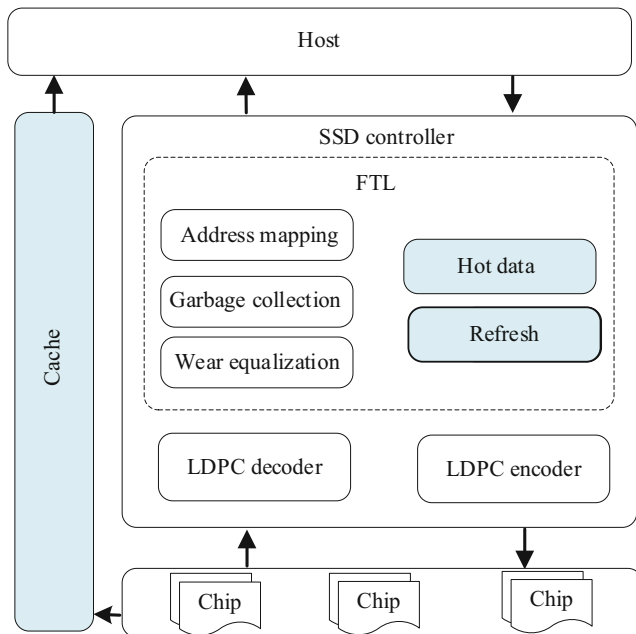


Fig. 7. Flash memory read and write process before and after optimization

The detailed reading process of the scheme is as follows:

- 1) Distinguish hot and cold data, and classify the hot and cold data according to the number of data reads. If the number of reads exceeds 1, it is hot read data, otherwise it is cold data. For hot-read data, determine the LDPC decoding method when the second read is successful, direct decoding if it is LDPC hard decoding, and record it if it is LDPC soft judgment decoding.
- 2) For hot-read data, determine the LDPC decoding method when the second read is successful, direct decoding if it is LDPC hard decoding, and record it if it is LDPC soft judgment decoding.
- 3) Determine whether the LDPC level of the hot read data is greater than or equal to 3. If it is greater than or equal to 3, check whether the cache space is sufficient. If it is sufficient, put it into the cache. If it is insufficient, use the LFU replacement algorithm to access it in the cache. The page with the least number of times is replaced to make room for newly identified hot read data.
- 4) Refresh the replaced page
- 5) When there is a new read request, it is first looked up in the cache, if found in the cache, it is read directly, and vice versa, the data is read from the flash memory chip cell.
- 6) Detects whether the data has completed error correction, and if it is complete, returns the data to the user, and vice versa.

5 Experiment

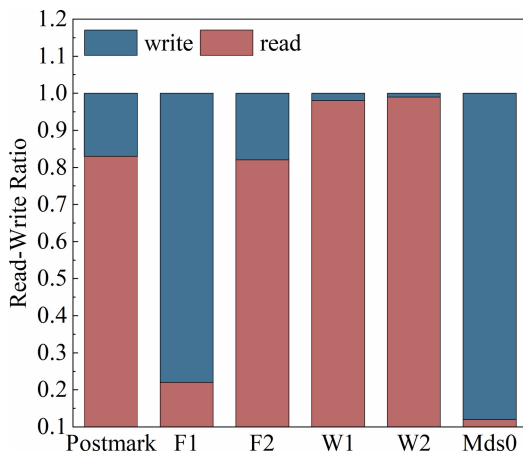
5.1 Experimental Environment

The solutions in this article are all implemented in the SSDModel module in disksim-4.0. Disksim-4.0 is a widely used disk drive performance simulator that can be used to evaluate the performance and behavior of disk drives. SSDModel is a module added to disksim, mainly used to simulate the performance of solid-state drive (SSD) simulators. SSDModel can simulate various SSD configurations, including different flash chip types, read/write cache sizes, block sizes, and more. This article's experiment was implemented in SSDModel, mainly modifying address mapping and cache size. The experimental parameters are shown in Table 2.

This experiment collects data for postmark, financial 1, financial 2, web search 1, web search 2, network search 3, mds0, rsrch_2 loads for workloads. Figure 8 shows the read/write ratios for various loads. In the following text, F1, F2, W1, W2, W3 are used to represent the Finance 1, Finance 2, Web Search 1, Web Search 2, and Web Search 3 workloads.

Table 2. Flash memory parameter configuration

parameter configuration	value
Planes per pack	8
Block per planes	2048
Page per block	64
Page size	16
Page read transfer time	0.0002384
Page write transfer time	0.0002384
chip transmission delay	0.000025
page read delay	0.025
page write latency	0.200
block erase delay	1.5

**Fig. 8.** Reading and Writing Ratios for Different Workload

5.2 Experimental Analysis

In our experiments, we evaluated three strategies.

- 1) Original strategy(NS): In the original strategy, only the garbage collection module, ECC module, and wear leveling module inherent in flash memory could be relied on to ensure data reliability.
- 2) Traditional refresh strategy(TRS): In the traditional refresh strategy, only when the quantization level of LDPC reaches 7 will the refresh condition be triggered and the data will be refreshed.
- 3) Strategy combining cache and refresh(CR): In this scheme, mainly for hot read data, when the read request arrives, first check whether the data exists in the cache, if it

exists, read directly, if it does not exist, check whether the cache has free space, if there is, put the hot read data into the cache, if not, use the LRU algorithm to put the recently infrequently accessed data into the cache, and refresh the replaced page.

Figure 9 shows the cache hit ratio under different loads. It can be seen from the Fig. 9 that the load with a higher read ratio has a higher hit rate, but not all loads are higher read ratio, and the higher the hit rate. For example, the read ratio of load Financial2 is 82%, and the hit rate is indeed 0.25. This is because although the read ratio of Financial2 is high, it can be seen from Fig. 4 that the access logical address of Financial2 is relatively scattered, and there are not many data that belong to hot reading, so the hit rate of Financial2 is low.

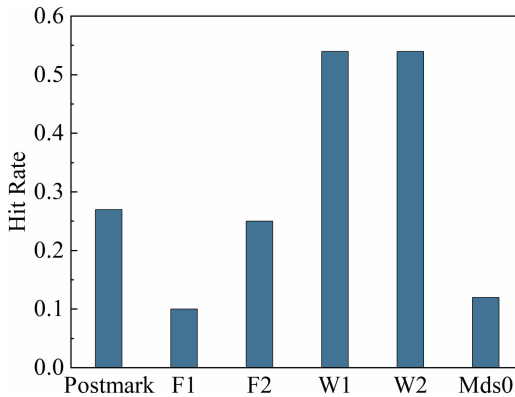


Fig. 9. Cache hit ratio under different Workload

Figure 10 shows the average response time of the caching and refresh combination scheme with the normalized average response time of the original and traditional refresh strategies. As can be seen in Fig a, the average response time of the CR scheme is reduced under all loads, but by different proportions. This is because this solution mainly optimizes the read operation of the flash SSD, and the optimization effect is more obvious for the load with more read operations. Such as Websearch1, Websearch2, Postmark, Financial2 These loads have a large read ratio, and their fluctuations are larger than the original scheme. However, the change of Financial1 is relatively stable, which is because the load of Financial1 has more write requests. As you can see from Fig b, the total average response time of Websearch1 and Financial2 has been reduced by 28% and 29%, respectively, compared to the traditional refresh strategy, and by 26% and 2%, respectively, compared to the traditional refresh strategy. It can be seen from this that the solution combining caching and refreshing has a better optimization effect under read-intensive loads.

Figure 11 shows the comparison of the normalized IOPS of the combined cache and refresh scheme with the original strategy and the traditional refresh strategy, and the comparison with the growth rate of the original strategy and the traditional refresh scheme. IOPS refers to the number of input/output operations that can be performed per second and is one of the important indicators to measure the performance of storage

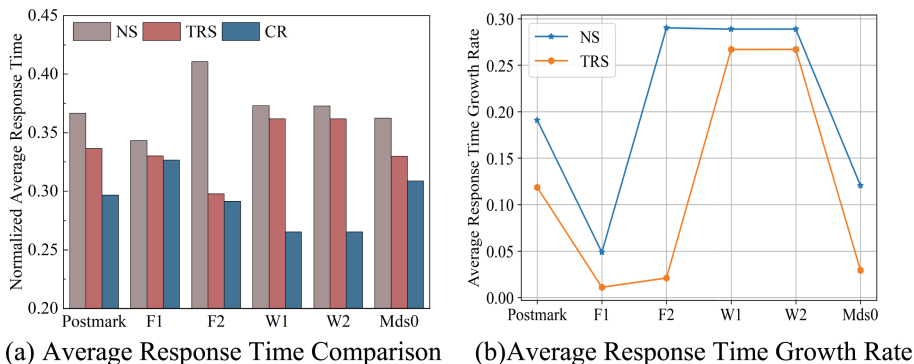


Fig. 10. Average Response Time and Reduction Rate under Different Workloads

devices. The higher the IOPS, the stronger the read and write capability of the storage device and the faster the data transmission speed. From Figure a, we can see that the IOPS of the CR solution performs higher than the IOPS of the other two solutions in these workloads. And the load with a small read ratio of the load changes more smoothly in IOPS, and the load with a larger read ratio has a greater increase in IOPS, such as Websearch1 and Financial2 for loads with a large read ratio. It can be seen from Figure B that compared with the original strategy, the I/O performance of Websearch1 and Financial2 has been improved by 43% and 46%.

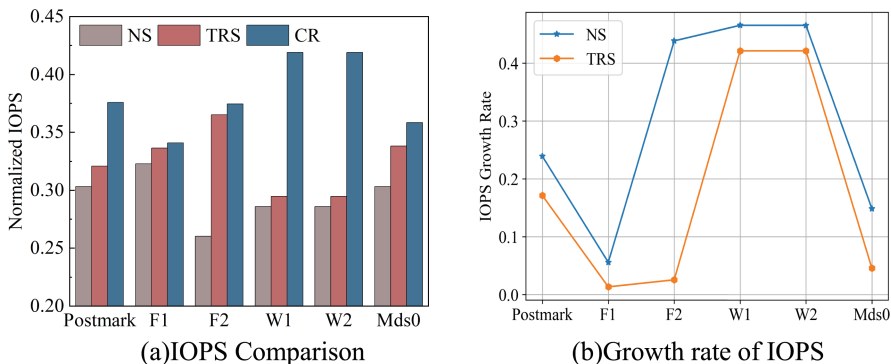


Fig. 11. IOPS vs. growth rate under different workloads

6 Conclusion

LDPC, as a powerful error correction ECC, is the default use of erasure coding by state-of-the-art flash memory. However, the large time overhead caused by LDPC decoding delay cannot be ignored. In order to solve this problem, this paper proposes a scheme combining cache and flushing, which aims to use cache and refresh technology to reduce

the decoding process of LDPC, further reduce the decoding delay of LDPC, and improve the read performance of flash memory. Combined load-based testing proves that this scheme is feasible, but using flushes will bring additional read and write operations, and we will continue to consider whether other ways to improve the read performance of flash memory in future further research.

Acknowledgments. This work is supported by The National Natural Science Foundation of China under Grant No. 61762075. It is also supported in part by the Provincial Natural Science Foundation Team Project of Qinghai under Grant 2020-ZJ-903. Ping Xie is the corresponding author of this paper.

References

1. Bjrling, M., Gonzalez, J., Bonnet, P.: LightNVM: the Linux open-channel SSD subsystem. In: 15th USENIX Conference on File and Storage Technologies, FAST 2017, USA 27 February–2 March 2017, Santa Clara, CA, pp. 359–374 (2017)
2. Zuolo, L., Zambelli, C., Micheloni, R., et al.: Solid-state drives: memory driven design methodologies for optimal performance. *Proc. IEEE* **105**(9), 1589–1608 (2017)
3. Lee, S., et al.: A 128 Gb 2b/cell NAND flash memory in 14 nm technology with tprog = 640 μ s and 800 MB/s I/O Rate. In: 2016 IEEE International Solid-State Circuits Conference (ISSCC), Digest of Technical Papers, San Francisco, USA, pp. 138–139 (2016)
4. Jeong, W., et al.: A 128 Gb 3b/cell V-NAND flash memory with 1 Gb/s I/O rate. *IEEE J. Solid-State Circuits* **51**(1), 204–212 (2016)
5. Cai, Y., et al.: Flash correct-and-refresh: Retention-aware error management for increased flash memory lifetime. In: 2012 IEEE 30th International Conference on Computer Design (ICCD), Montreal, QC, Canada, pp. 94–101 (2012)
6. Tanakamaru, S., Yanagihara, Y., Takeuchi, K.: Error-prediction LDPC and error-recovery schemes for highly reliable solid-state drives (SSDs). *IEEE J. Solid-State Circuits* **48**(11), 2920–2933 (2013)
7. Tseng, Y.-F., Shieh, M.-D., Kuo, C.-H.: Low latency design of polar decoder for flash memory. In: 2021 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW), Penghu, Taiwan, pp. 1–2 (2021)
8. Shi, L., Lv, Y., Luo, L., et al.: Read latency variation aware performance optimization on high-density NAND flash based storage systems. *CCF Trans. High Perform. Comput.* **4**(3), 265–280 (2022)
9. Cai, Y., Haratsch, E.F., Mutlu, O., Mai, K.: Error patterns in MLC NAND flash memory: measurement, characterization, and analysis. In: 2012 Design, Automation & Test in Europe Conference & Exhibition (DATE), Dresden, Germany, pp. 521–526 (2012)
10. Yu, C., Haratsch, E.F., Mutlu, O., et al.: Threshold voltage distribution in MLC NAND flash memory: characterization, analysis, and modelling. In: Design, Automation & Test in Europe Conference & Exhibition, pp.1285–1290. IEEE (2013)
11. Kou, Y., Lin, S., Fossorier, M.P.C.: Low-density parity-check codes based on finite geometries: a rediscovery and new results. *IEEE Trans. Inf. Theory* **47**(7), 2711–2736 (2001)
12. Du, Y., Li, Q., Shi, L., Zou, D., Jin, H., Xue, C.J.: Reducing LDPC soft sensing latency by lightweight data refresh for flash read performance improvement. In: 2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC), Austin, TX, USA, pp. 1–6 (2017)
13. Li, Q., Shi, L., Xue, C.J., et al.: Access characteristic guided read and write cost regulation for performance improvement on flash memory. In: 14th FAST 2016, Santa Clara, CA, USA, pp.125–132 (2016)

14. Du, Y., Zou, D., Qiao, L., et al.: LaLDPC: latency-aware LDPC for read performance improvement of solid state drives. In: 33rd International Conference on Massive Storage Systems and Technology (MSST 2017) (2017)
15. Shi, L., Wu, K., Zhao, M., et al.: Retention trimming for lifetime improvement of flash memory storage systems. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **35**(1), 58–71 (2015)
16. Liu, R.-S., Chuang, M.-Y., Yang, C.-L., Li, C.-H., Ho, K.-C., Li, H.-P.: EC-Cache: exploiting error locality to optimize LDPC in NAND flash-based SSDs. In: 2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC), San Francisco, CA, USA, pp. 1–6 (2014)
17. Gu, B., Luo, L., Lv, Y., et al.: Dynamic file cache optimization for hybrid SSDs with high-density and low-cost flash memory. In: 2021 IEEE 39th International Conference on Computer Design (ICCD), pp. 170–173 (2021)
18. Yoon, J., Ro, W.W.: Access characteristic-based cache replacement policy in an SSD. In: ICCV Workshops, pp. 1–4 (2019)
19. Tripathy, S., Satpathy, M.: SSD internal cache management policies: a survey. *J. Syst. Archit.* (122), 122 (2022)
20. International Conference on Networking, Architecture and Storage (NAS), pp. 1–10. EnShi, China (2019)