applied sciences

MDPI

Article

Exploiting Data Similarity to Improve SSD Read Performance

Shiqiang Nie [1], Jie Niu [1], Zeyu Zhang [1], Yingmeng Hu [2], Chenguang Shi [3] and Weiguo Wu [1,*]

[1] School of Computer Science and Technology, Xi'an Jiaotong University, Xi'an 710049, China; nsqiang@gmail.com (S.N.)
[2] China Satellite Network Innovation Co., Ltd., Beijing 100190, China
[3] Shanghai Key Laboratory of Satellite Network, Shanghai Satellite Network Research Institute Co., Ltd., Shanghai 201210, China
* Correspondence: wgwu@xjtu.edu.cn

Abstract: Although NAND (Not And) flash-based Solid-State Drive (SSD) has recently demonstrated a significant performance advantage against hard disk, it still suffers from non-negligible performance under-utilization issues as the access conflict often occurs during servicing IO requests due to the share mechanism (e.g., several chips share one channel bus, several planes share one data register inside the die). Many research works have been devoted to minimizing access conflict by redesigning IO scheduling, cache replacement, and so on. These works have achieved reasonable results; however, the potential data similarity characterization is not utilized fully in prior works to alleviate access conflict. The basic idea is that, as data duplication is common in many workloads where data with the same content from different requests could be distributed to the address with minimized access conflict (i.e., the address does not share the same channel or chip), the logic address is mapped to more than one physical address. Therefore, the data can be read out from candidate pages when the channel or chip of its original address is busy. Motivated by this idea, we propose Data Similarity aware Flash Translation Layer (DS-FTL), which mainly includes a content-aware page allocation scheme and a multi-path read scheme. The DS-FTL enables maximization of the channel-level and chip-level parallelism and avoids the read stall induced by bus-shared mechanisms. We also conducted a series of experiments on SSDsim, with the subsequent results depicting the effectiveness of our scheme. Compared with the state-of-art, our scheme reduces read latency by 35.3% on average in our workloads.

Keywords: NAND flash; data duplication; page allocation; read redirection

Citation: Nie, S.; Niu, J.; Zhang, Z.; Hu, Y.; Shi, C.; Wu, W. Exploiting Data Similarity to Improve SSD Read Performance. *Appl. Sci.* **2023**, *13*, 13017. https://doi.org/10.3390/app132413017

Academic Editors: Antonio Fernández-Caballero and Christos Bouras

Received: 24 October 2023
Revised: 21 November 2023
Accepted: 3 December 2023
Published: 6 December 2023

1. Introduction

Data-dominated workloads, such as artificial intelligence (AI) and high-performance computing (HPC), have become the dominant applications in data centers. To close the performance gap between computation and storage, NAND flash-based solid-state drives (SSDs) instead of hard disks have been extensively deployed in data centers. SSDs utilize multi-parallel channels that connect NAND flash chips, enabling parallel read/write operations and advanced command designs to simultaneously operate multiple planes. This architecture provides enhanced parallelism for serving the host. However, when multiple requests from the host access the same channel or chips simultaneously, these requests must wait for each other and be serviced sequentially. This access conflict among requests hampers the rich parallelism offered by multiple channels and chips per channel, leading to a decrease in overall performance.

Reducing access conflict has become a hot topic in the research field of NAND flash-based SSDs. On the host side, some studies propose grouping I/O requests without conflicts into the same batch to avoid access conflicts [1], while others suggest replicating hot data at different parallel units (e.g., channel, chip, and plane) as potential solutions [2]. Additionally, some researchers focus on redesigning the connection topology between the

channel and NAND flash chips to alleviate the limited bus resource issue [3]. These works successfully mitigate access conflicts in NAND flash-based SSDs and achieve considerable performance improvements. However, it is important to note that these approaches entail modifications to the hardware/software stack or result in storage capacity overhead.

This paper is motivated by the observation of data duplication being a common occurrence in various workloads. The main idea behind this research is to allocate sub-requests with identical content but different logical addresses to different parallel units (i.e., channels, chips, or planes) in the SSD. The SSD records this 1-to-n mapping information, allowing it to read data with the same content from any physical address, thereby avoiding access conflicts. Unlike previous approaches, such as replicating and spreading hot data across the SSD [2], our method does not incur any additional storage cost. Furthermore, unlike many studies that focus on data deduplication schemes within the SSD [4–6], we utilize data duplication characteristics to improve performance while maintaining data reliability. Overall, this paper offers the following contributions.

- We conduct preliminary experiments that reveal a low and skewed utilization issue among channels, indicating insufficient bus utilization within the SSD. Additionally, we analyze the data redundancy ratio in various workloads, which highlight the potential opportunity in data placement. These findings motivate us to leverage the data duplication characteristics for performance improvement.
- We reveal the limitations of the traditional static page allocation scheme and dynamic page allocation scheme. To address these limitations, we propose a new approach called Data Similarity aware Flash Translation Layer (DS-FTL). DS-FTL consists of a content-aware page allocation scheme and a multi-path read scheme. This approach maximizes channel-level and chip-level parallelism while preventing read stalls caused by bus-sharing mechanisms.
- We conduct a series of experiments and validate the effectiveness of our scheme. The results demonstrate its superiority compared to state-of-the-art approaches. On average, DS-FTL improves the channel utilization ratio and reduces read latency by 35.3%.

The remainder of this paper is structured as follows. Section 2 discusses the background of SSDs and motivations behind our design. Section 3 presents the proposed scheme in detail. Section 4 describes the experimental methodology and analyzes the results. Section 5 describes the related work. Lastly, the conclusion of this paper is given in Section 6.

2. Background and Research Motivation

2.1. SSD Architecture

Figure 1 shows the internal organization of an SSD. The components of an SSD are divided into two groups :the front-end and the back-end. The front-end is responsible for managing the back-end resources and issuing I/O transactions to the back-end channels [7]. The Host Interface Logic receives an I/O request from the host, splits it into page-sized sub-requests with specific Logical Page Number (LPN) [8], and inserts them into device queues for internal processing. The Flash Translation Layer (FTL) maintains a mapping table that tracks the current physical location, known as the Physical Page Number (PPN), of each LPN. When handling read sub-requests, the FTL searches the mapping table to find the corresponding PPN based on the LPN. For write requests, the FTL employs a page allocation scheme to assign free pages for data storage. The page allocation determines the target channel ID, chip ID, die ID, and plane ID of a page-sized transaction, following a specific priority order of the parallelism levels [9].

In the subsequent steps, a block ID is assigned to the target plane, and pages are programmed in a predetermined order within the block. Subsequently, the PPN of the block is calculated. After the completion of address translation, the page-sized transactions are delivered to the transaction scheduling unit (TSU) for scheduling their execution on the target die [10]. The front-end manages the back-end resources and issues I/O requests to the back-end channels via the I/O scheduler. The SSD back-end contains multiple channels.

Each channel is connected to one or more chips. Each chip is made up of one or more chips, each of which has one or more planes. For processing I/O requests, such an organization offers four layers of parallelism (channel, chip, die, and plane).

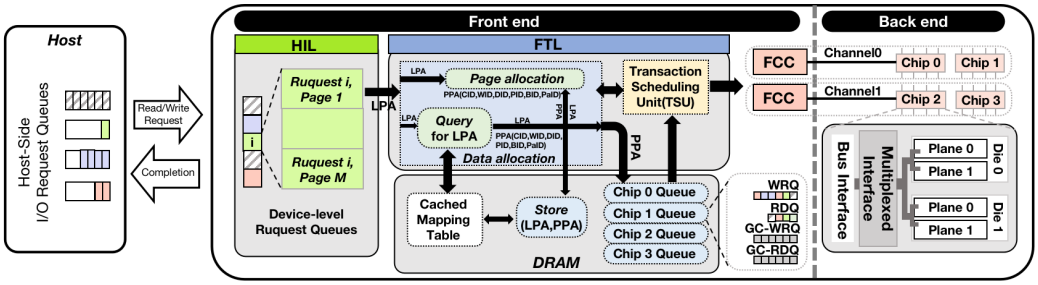


Figure 1. An illustration of SSD architecture.

2.2. Page Allocation Scheme

The page allocation scheme plays a crucial role in determining how free physical pages are allocated to accommodate the data of write requests, which can significantly impact the read performance of the SSD as it determines the degree of access parallelism [11].

Many studies have been conducted on page allocation schemes, and two primary types are widely used: static allocation and dynamic allocation. Static allocation is based on fixed striping, where logical pages are initially allocated to predefined channels, chips, dies, and planes, followed by allocation to the first available page in the active block of the target plane [12]. The target channel, chip, die, and plane addresses for a page-sized flash transaction are determined in a specific order of priority for these four levels of parallelism, usually calculated using specific formulas.

On the other hand, dynamic allocation allocates physical pages dynamically, allowing logical pages to be assigned at runtime to any available physical page across the SSD. When a write request arrives, the dynamic allocation scheme selects free physical pages based on several factors, such as the idle/busy status of the channel and chip, the erase counts of the block, and the priority of the parallel level. It also takes into consideration the utilization of multiple parallel levels [13].

While dynamic allocation is more flexible and adaptive in exploiting the parallelism of SSDs and generally provides better performance in most cases, static allocation is simple to implement and can be highly effective in certain workloads. Static allocation performs best for read operations in all workloads, while dynamic allocation tends to exhibit superior overall performance and durability under most workloads in aging SSDs [13].

2.3. Motivation

The FTL employs various page allocation schemes to address the challenge of data placement for write requests. These allocation schemes determine the placement of data, which subsequently impacts the read and write performance of SSDs. This section provides a detailed discussion of the drawbacks associated with the current page allocation scheme, which serve as the motivation for presenting the proposed Data Similarity aware FTL (DS-FTL).

2.3.1. Uneven Utilization of Channels and Chips

The allocation schemes used in SSDs can result in an uneven distribution of workload across different channels and chips [3,14], leading to a phenomenon known as skewed requests. This phenomenon leads to significant variations in the utilization of channels and chips, ultimately impacting the overall performance of the SSD. Figure 2 presents the observed channel and chip utilization, with the specific configuration outlined in Section 4. The figure clearly illustrates the substantial variation in utilization, with a difference of approximately 34% between chips. In the case of read requests, the distribution of data across different channels, chips, dies, planes, and blocks is determined by the employed

page allocation scheme and the workload characteristics. Similarly, for write requests, the data distribution is also influenced by the page allocation scheme. Although dynamic page allocation schemes can effectively address the issue of load imbalance, they can lead to further disparities in utilization, subsequently impacting read performance.

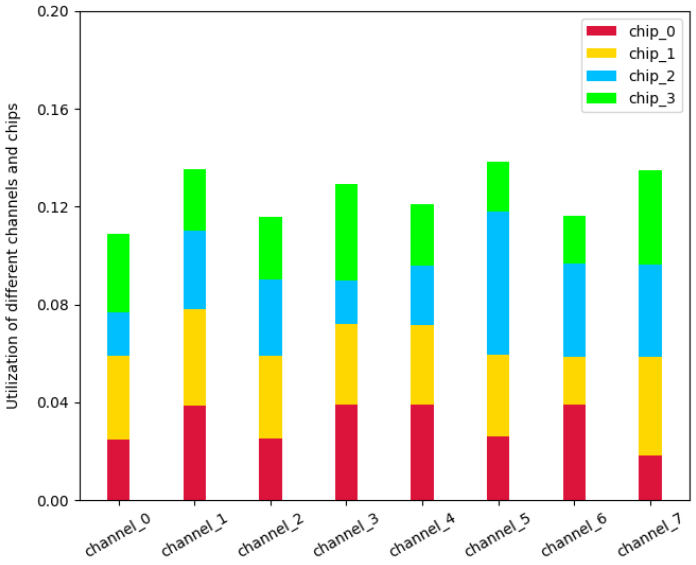


Figure 2. Utilization of different channels and chips.

2.3.2. Data Similarity in the Workload

In Figure 3, we present our findings after conducting an initial analysis to determine the percentage of duplicate data in eight real traces. The results reveal that data duplication is a prevalent characteristic in these traces, with some traces even exhibiting a duplication rate as high as 70.4%. This high data redundancy rate is caused by the intrinsic user pattern, such as write-ahead logs and copy operations within the application. Additionally, we assess the data duplication rate across different planes following data writing, as depicted in Figure 4. These results demonstrate that the percentage of duplicate data ranges from 5.1% to 44.5% across different planes, reaffirming the commonality of data duplication in actual SSD data storage [4,5,15].

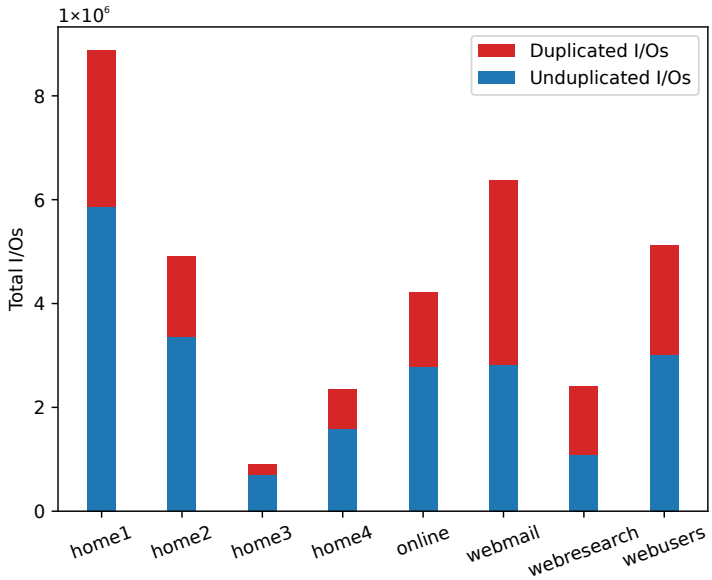


Figure 3. The data redundancy ratio of different traces.

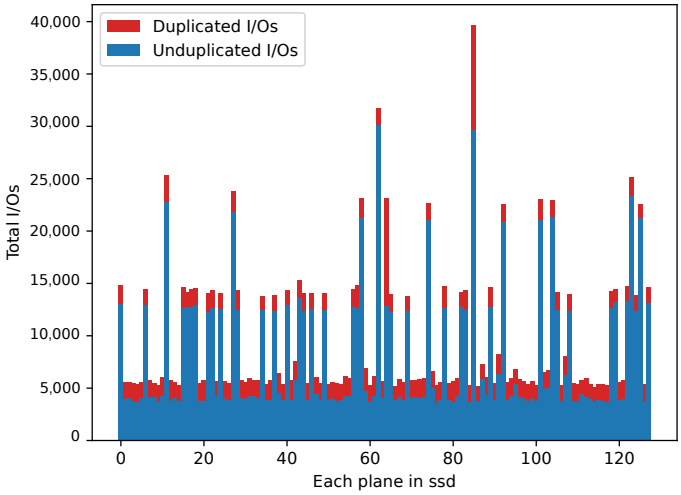


Figure 4. The proportion of duplicate data on each plane.

2.3.3. Key Idea of the Proposed Solution

To address the aforementioned issues, designing an effective page allocation scheme emerged as the most direct and efficient approach to enhance channel and chip utilization, harness data similarity in workloads, and thus improve SSD performance. However, mainstream allocation schemes, including static and dynamic schemes, possess limitations. Static allocation schemes fail to evenly distribute the read and write loads, whereas dynamic allocation schemes address the bus resource underutilization and imbalance during data writes but impede the exploitation of rich parallelism during data reads from flash chips, thereby impacting read performance. As a result, the channel and chip utilization imbalance problem remains unsolved [16]. Consequently, based on the observed high data similarity in workloads, we propose a novel page allocation scheme that combines the strengths of static and dynamic allocation while accounting for the need for simplicity and lightweightness. This scheme aims to improve the channel and chip balance for read operations by leveraging the feature of data content similarity. In this paper, we present DS-FTL, which facilitates the allocation of page-size sub-requests to different parallel units to maximize channel and chip utilization and enhance overall write performance. Additionally, we adopt a multi-path read scheme based on data similarity to boost overall read performance.

3. Related Work

In Table 1, we present a comprehensive summary of the body of literature related to page allocation schemes, data deduplication in SSD, and path conflict resolution in SSD.

3.1. Page Allocation Schemes

Shin et al. [12] proposed 16 different levels of static page allocation schemes according to different priorities of flash memory parallel cells, such as channel-level first and chip-level second parallelism. Hu et al. [13] experimentally analyzed the impact of static and dynamic page allocation schemes based on SSD performance, and experimentally showed that dynamic allocation outperform static allocation in terms of write amplification and wear levels, the static allocation is usually used for read-intensive workloads, while the dynamic allocation is suitable for write-intensive workloads. On this basis, Jung et al. [11] simulated a cycle-accurate SSD platform with 24 page allocation schemes and found that flash-level parallelism is disturbed by system-level concurrency mechanisms. However, internally in SSDs, there are limitations to static and dynamic allocation schemes. Chang et al. [17] proposed an adaptive striping allocation scheme for handling load imbalance between flash units. In addition, Reddy et al. [18] also proposed a hybrid data allocation scheme by dynamically monitoring and analyzing the access load characteristics of flash memory to establish an access model for data, through which the relevant parameters of the hybrid

data allocation scheme can be adjusted. Paik et al. [19] proposed a read-aware dynamic allocation mechanism for multi-channel solid-state devices, this read-aware mechanism reduces read latency by avoiding read operation conflicts when trying to access resources already occupied by pre-issued write operations. Wu et al. [20] proposed OSPADA, a one-shot programming aware data allocation policy, which reorders the written data and allocates logically ordered data pages to different flash parallel cells based on a distance-cycling-aware scheme to optimize flash read performance by exploiting the multi-level parallelism of flash channels. Sun et al. [21] proposed a hybrid page allocation scheme called HIPA, which uses a load prediction mechanism to monitor chip-level load in real time, thus determining the page allocation scheme based on the state of chip-level load and improving SSD read and write parallelism.

Table 1. Related work.

Problem	Literature
Adopting diverse page allocation strategies to enhance the parallelism of SSD read and write operations	Shin et al. [12], Hu et al. [13], Jung et al. [11], Wu et al. [20], Chang et al. [17], Reddy et al. [18], Paik et al. [19], Sun et al. [21]
Implementing data deduplication in SSD to reduce the amount of data being written	Chen et al. [4], Gupta et al. [15], Kim et al. [5], Wu et al. [22], Chen et al. [23], Ni et al. [6]
Employing path conflict resolution in SSD to improve the parallelism of read and write operations	Schuetz et al. [24], Gillingham et al. [25], Kim et al. [3], Tavakkol et al. [26], Kim et al. [14], Nadig et al. [27]

3.2. Data Deduplication in SSD

Chen et al. [4] introduced a content-aware flash memory translation layer (CAFTL) to enhance SSD endurance at the device level. It employed a combination of online and offline data deduplication techniques to eliminate redundant data and effectively reduce write traffic to flash memory. However, CAFTL utilized a pre-hashing technique that did not detect duplicate data pages in subsequent arrivals, resulting in a loss of deduplication rate. Gupta et al. [15] proposed CA-SSD, a flash SSD that employed content addressable storage (CAS) for internal data management. CA-SSD leveraged value locality, which exploits the likelihood of data being accessed again in the near future, based on the content of the data. Unlike CAFTL, CA-SSD fully implemented online data deduplication and did not rely on offline deduplication mechanisms. However, CA-SSD utilized a dedicated hardware processor for hash computation, thereby increasing the production cost of SSDs and limiting its universal applicability. Kim et al. [5] further examined the impact of data deduplication on SSD performance. They proposed managing fingerprints based on temporal locality, retaining only the most recently generated fingerprints instead of all generated fingerprints. This method maximized SSD deduplication rate while reducing memory space consumption. Unfortunately, this technique also required a dedicated hardware processor similar to CA-SSD, resulting in increased SSD costs. Wu et al. [22] presented ΔFTL (Delta Flash Translation Layer), a data deduplication method that minimized redundant data in SSDs using a compression increment strategy. ΔFTL achieved a small compression ratio by leveraging content locality between written data and its corresponding old version stored in the SSD. Instead of writing the entire new data, only the compression increment was stored, ultimately reducing the amount of data written to the SSD. However, during reads, ΔFTL required the SSD to be read twice and the data to be decompressed and merged, which negatively impacted the read performance. To mitigate the overhead of fingerprint computation, Chen et al. [23] introduced NF-Dedupe. Unlike traditional deduplication methods, NF-Dedupe avoided using the time-consuming SHA-1 algorithm for fingerprint computation. Instead, it employed the weaker CRC32 fingerprint to identify potential duplicate data pages. Non-duplicate data were filtered, and only potentially duplicated data were read from the SSD. Byte-by-byte comparison was then used to confirm duplication,

thus achieving the desired deduplication effect. Ni et al. [6] presented WOJ (Write-Once data Journaling), a data journal-based deduplication system. WOJ utilized data deduplication techniques to eliminate redundant data from the journal records, thus reducing the amount of data written to the SSD and enhancing the device's lifespan.

3.3. Path Conflict Resolution in SSD

Previous research has suggested the implementation of an interconnection network within SSDs to enhance parallelism in flash arrays. Effectively managing path conflicts is crucial for improving parallelism and performance in SSDs. The HyperLink NAND flash architecture (HLNAND) utilizes either a daisy-ring or hierarchical ring topology to connect flash chips [24,25]. In terms of mitigating path conflicts by increasing flash channel bandwidth, Kim et al. [3] proposed the utilization of packetized communication to enhance the effective flash memory interconnect bandwidth. They presented the packetized SSD (pSSD) system architecture, which enables higher effective flash channel bandwidth by utilizing "packets" instead of dedicated signals. However, this technique necessitates significant modifications to the NAND flash chip, resulting in substantial overhead and only partially addressing the path conflict problem. Alternatively, to mitigate path conflicts by increasing path diversity, Kim et al. [3] proposed the Packet Network Solid State Drive (pnSSD). This technology incorporates an interconnection network resembling a 2D mesh topology, providing two paths to access each flash chip and thus reducing the performance overhead caused by path conflicts. Similarly, Tavakkol et al. [26] suggested replacing shared multi-channel bus wiring with an interconnection network. They introduced the Network-on-SSD (NoSSD), which enables pipelined multi-router access to flash memory, resulting in improved performance and greater bandwidth compared to traditional SSD architectures. However, Tavakkol's study lacks implementation details of network routers and practical models of NoSSD for realistic messaging protocol-related considerations. Another proposed solution is the Decoupled SSD system, as proposed by Kim et al. [14]. This system decouples the front-end (i.e., cores, system bus) from the back-end (i.e., flash memory) and incorporates an on-chip network to interconnect the controllers. However, none of these approaches effectively mitigate the path conflict problem due to their lack of providing sufficient path diversity between the flash controller and the flash chip. In contrast, Nadig et al. [27] proposed Venice, a novel mechanism that introduces a cost-effective interconnection network of flash chips and efficiently utilizes path diversity to fundamentally address the path conflict problem in SSDs. Venice significantly enhances SSD performance for various real-world data-intensive workloads.

4. Design

4.1. Design Overview

In order to address the issue of performance degradation caused by read access conflicts, we have redesigned the existing components of the Flash Translation Layer and propose a new approach called Data Similarity aware FTL (DS-FTL). The main idea behind DS-FTL is to evenly distribute the data with the same content across all channels, chips, and planes as quickly as possible, while also keeping track of the address mapping information. This allows multiple PPNs to be mapped to a single LPN, enabling data targeting a single LPN to be read from multiple PPNs. This approach helps avoid stalling due to busy channel or chip buses during read operations. To achieve this objective, DS-FTL introduces two novel techniques: the content-aware page allocation scheme and the multi-path read scheme, as illustrated in Figure 5. The content-aware page allocation scheme determines the physical address allocation by comparing the new coming data's content with the existing data resident inside the SSD. This scheme builds upon state-of-art page allocation schemes. The multi-path read scheme allows for data to be read from any idle channel or chip by looking up the mapping table if a LPN has multiple target PPNs.

Based on the above principle, some design challenges are discussed in detail. The main issue includes:

1. How to integrate the content similarity comparison into the existing write procedure inside SSD with less or negligible overhead.
2. How to utilize the physical page with the same content at different parallel units for read performance improvement, that is how to redesign the mapping management and IO scheduler inside SSD for multi-path read.

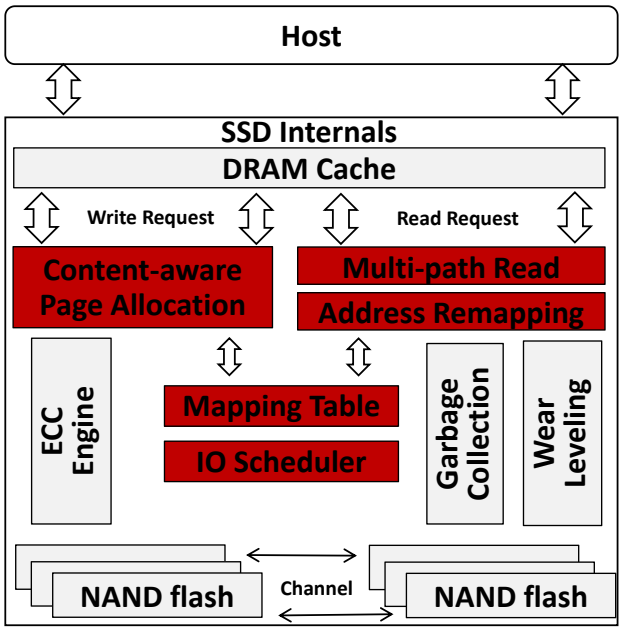


Figure 5. The Design of Data Similarity aware FTL.

4.2. Content-Aware Page Allocation Scheme

To alleviate the read access conflict issue, it is crucial to ensure that data with the same content is evenly written across all parallel units. In DS-FTL, we propose a content-aware allocation scheme that leverages existing page allocation schemes to redirect data to the parallel unit with the least likelihood of data duplication.

4.2.1. Write Operation Procedure

Figure 6 illustrates the process of writing data into SSD in detail. (1) When a write request is received from the host, the SSD divides each data into page-sized segments (e.g., 4 KB) and caches them in the DRAM. The first phase of content comparison involves calculating the fingerprint (i.e., SHA-1 [5]) of each data segment. This computation time overhead can be hidden by performing these steps concurrently.

(2) The cached page-sized data segments are evicted from the DRAM and written into the NAND flash back-end. DS-FTL utilizes existing static or dynamic page allocation schemes or other widely used schemes [9,10,13,17], to calculate the target PPNs for the data segments. Then, it looks up the fingerprint table of the corresponding channel, which records the fingerprints of all resident pages in that channel. It is worth noting that we maintain the fingerprint table at the channel level, as it imposes the least parallelism restriction compared to the chip or plane level. In the figure, the symbol "#" denotes the respective label number.

(3) If the fingerprint of the write request matches one in the channel's fingerprint table, indicating that the same data segment is already stored in this channel, a round-robin approach is used to select the next channel or chip as a candidate until the write sub-request is serviced.

(4) If the fingerprint does not match, we then update the mapping table and schedule the write sub-request to be written into this PPN as normal.

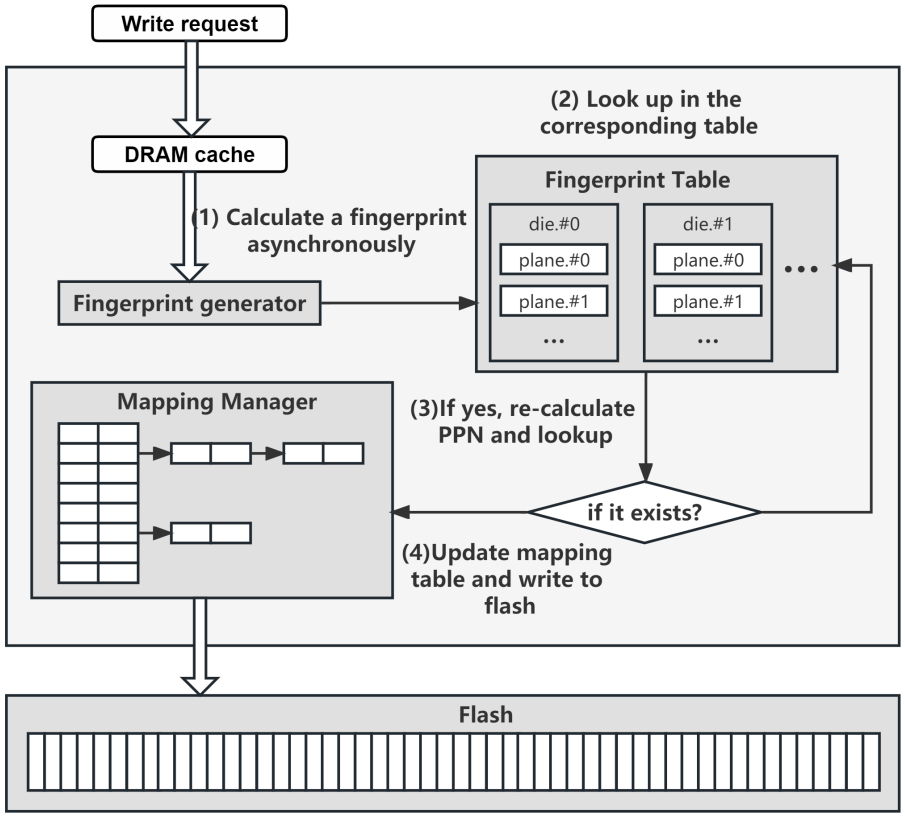


Figure 6. Write operation process.

4.2.2. Data Content Comparison

Due to the limited DRAM space in SSDs, performing byte-to-byte comparisons results in significant time consumption and computational overhead. In order to mitigate this, we propose dividing the data content comparison process into three parts. In the first phase, we employ the bloom filter, which is a hash-function-based algorithm used widely in data deduplication systems [28], to store written data information. During the page allocation procedure, our DS-FTL verifies whether the written data contains the same content on the target channel by employing the bloom filter. Bloom filter is a space-efficient probabilistic data structure characterized by an m-bit array and k-independent hash functions $h_1, h_2, ..., h_k$, where the value domain of the hash function is $1, 2, ..., m$. Suppose the input data set $S = X_1, X_2, ..., X_n$ has n elements, each element is mapped to a different position of the array by each hash function, and the value of the corresponding position is set to 1, $x \in S, Bf[h_i(x)] = 1, i \in [1, k]$. Thus, if the element y has a value of 1 at all positions mapped by all hash functions, then it is highly likely that this element has existed in the set S, as shown in Figure 7. On the contrary, if there is any position with a value of 0, the element must not be in the set S.

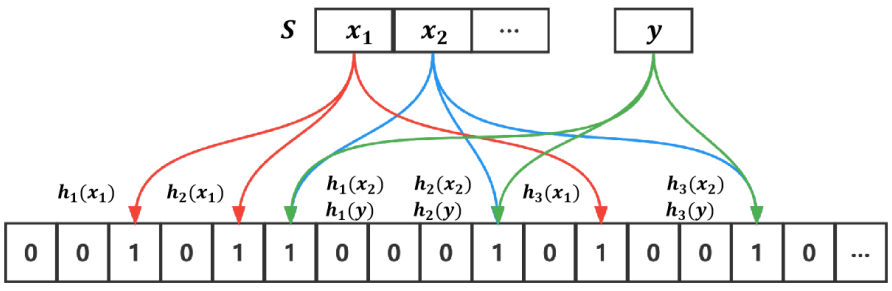


Figure 7. Bloom filter design.

In the second phase, we conduct byte-to-byte comparison further to validate the data segment having the same content in the channel, and this time-consuming operation is performed asynchronously in idle time.

After the fast approximate comparison using the bloom filter, DS-FTL compares the data of the write request with the data comparison set. The data comparison set consists of PPNs whose data are hashed in the same position in the Bloom filter but are not identical. It is worth noting that DS-FTL periodically performs the byte-to-byte comparison task in the background or during idle periods, consuming only limited CPU and memory resources.

In the third phase, after the data content comparison is completed, DS-FTL updates the mapping table, which is discussed in Section 4.3.1.

4.3. Multi-Path Read Scheme

4.3.1. Address Mapping Management

As data with the same content are distributed across different parallel units, such as different channels or chips, DS-FTL utilizes the modified mapping table to enable reading data from an idle channel/chip, even when other channels/chips are busy. This further enhances parallelism and improves system performance. The existing mapping table in SSDs typically stores a one-to-one mapping pair between logical and physical addresses. However, DS-FTL modifies this mapping to a one-to-many (1-to-n) relationship. To facilitate this, DS-FTL employs a two-level mapping table, as utilized in previous works [4,29], to record the 1-to-n mapping. The two-level mapping table consists of a preliminary mapping table and a secondary mapping table. The preliminary mapping table contains entries with LPN and PPN pairs. The highest bit in the PPN is utilized to indicate whether a lookup in the secondary mapping table is required. For unique data within the SSD, the mapping table records the mapping entry as usual. However, for redundant data within the SSD, multiple LPNs point to the same Virtual Physical Page Number (VPPN), and a lookup in the secondary mapping table is performed using the VPPN to retrieve the page addresses with the same content. The garbage collection scheme works as normal; the only difference is that DS-FTL updates the 1-to-n mapping pair in the secondary mapping table if the data in the corresponding PPN are deleted.

4.3.2. I/O Scheduling Optimization

Figure 8 illustrates the detailed process of reading data in DS-FTL. Similar to existing FTL works, DS-FTL handles read sub-requests in the following manner. When a read sub-request is received by the flash back-end, DS-FTL performs a lookup in the mapping table and the indirect mapping table to retrieve the associated PPNs. One or several PPNs may be obtained. DS-FTL then transfers this result to the I/O scheduler and selects an idle path to retrieve data from one of the resident pages. If a single PPN is obtained, this indicates that the data to be read are unique within the SSD. In this case, the read sub-request is directly inserted into the corresponding I/O request queue. However, if a set of PPNs is obtained, corresponding to multiple chip queues (e.g., chip A queue, chip B queue, and chip C queue), DS-FTL chooses the chip C queue with the lowest number of requests and more available idle slots, as depicted in Figure 9. This approach ensures maximum utilization of the SSD's parallelism and improves the efficiency of processing read requests. Furthermore, if there is a sub-request in the selected chip queue that reads similar data, DS-FTL can merge the two sub-requests to reduce duplicate read operations and unnecessary time overhead.

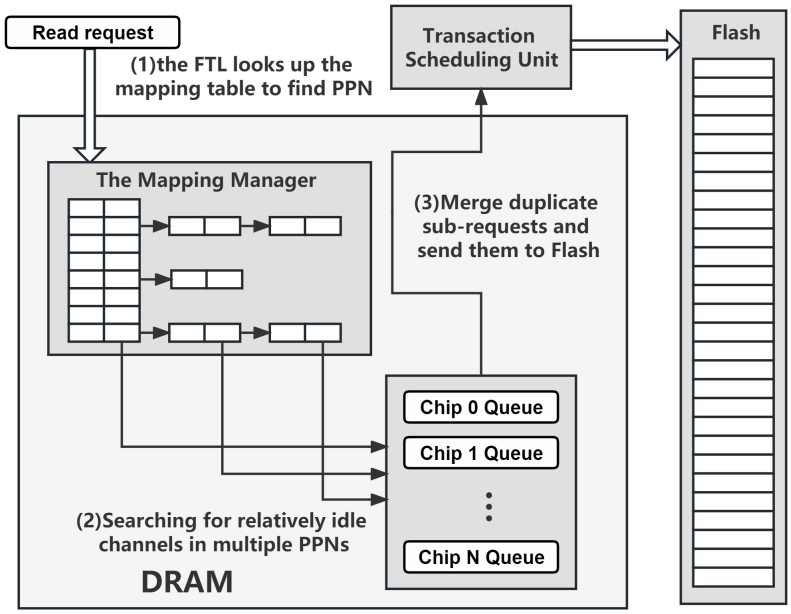


Figure 8. Read optimized design.

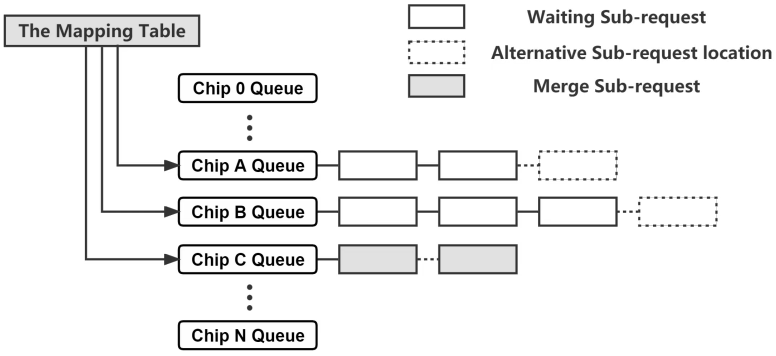


Figure 9. I/O scheduling policy optimization.

4.4. Analysis and Limitation Discussions

The SSD typically employ a scrambler module to randomize user data before writing them into a page. This randomization helps balance the distribution of 1s and 0s and improves flash reliability [30]. DS-FTL, being aware of the data content, performs data content comparison before data randomization, allowing it to coexist with the scrambler module. DS-FTL enhances read performance by distributing duplicate data from the host across different parallel units. This arrangement enables retrieving data from multiple candidate physical pages without any blocking in subsequent read operations. The performance benefit of this approach depends on the workload. For workloads with a high data duplication ratio [31], DS-FTL can effectively improve performance. However, for workloads with less duplicated data, the write procedure of DS-FTL does not introduce significant time overhead, as fingerprint computation occurs while the data are cached in DRAM. Hence, DS-FTL eliminates the overhead associated with hash computation on the critical write I/O path. Although write performance may slightly degrade due to the bloom filter, the read performance remains the same as the baseline. In cases where data with the same content undergo modification, only the PPN is modified, while other data remain in their default positions. Therefore, the utilization of read/write parallelism remains normal in the worst-case scenario.

5. Experiment

5.1. Experimental Environment Setup

We implemented and evaluated our proposed optimization scheme in SSDsim, an SSD simulator. In this section, we introduce the SSD simulator, SSD configuration and the characterization of these traces utilized in the experimental section.

5.1.1. SSD Simulator

We used an open-source simulator, SSDsim [13], to implement our proposed scheme. We modify the page allocation scheme, mapping management module and so on into SSDsim to evaluate our scheme.

5.1.2. SSD Configurations

According to the hardware information and software configuration in real commercial SSD, we configured the parameters of the SSDsim for the experiment, as shown in Table 2.

Table 2. SSD configurations.

Parameter	Value
Number of SSD channels	8
Chips per Channel	2
Dies per Chip	2
Planes per Die	2
Blocks per Plane	2048
Pages per Block	64
Flash Page Size	4 KB

5.1.3. Workload Characteristics

We utilized a set of block I/O trace data obtained from an operational system running for several days. These traces were collected from four different end-user/developer home directories, a course management system, and three network servers, making them widely applicable in related research [32,33]. Table 3 presents the three-week I/O data for these workloads and the characteristics of these traces.

5.1.4. Comparison with Other Schemes

In this subsection, we compare the following schemes.

- `static allocation [12]`. This scheme is a simple but effective page allocation used widely, it could spread the user data by calculating LPN across the parallel unit evenly.
- `dynamic allocation [11]`. This scheme improves the write performance by allocating physical pages for write requests in a round-robin way, and sacrifices the read parallelism.
- `HIPA [21]`. This scheme uses a load prediction mechanism to monitor chip-level load in real time, thus determining the page allocation scheme based on the state of chip-level load and improving SSD read and write parallelism.
- `DS-FTL`. We implement DS-FTL in the SSD simulator to optimize the read performance based on the access conflict reduction idea.

Table 3. Workload characterization.

Trace	Total I/Os	Write I/Os	Write Ratio	Average Request Size (KB)
home1	8,973,201	8,882,831	0.990	16.06
home2	5,390,703	4,901,091	0.909	23.47
home3	918,010	908,863	0.990	34.16
home4	2,491,157	2,354,060	0.945	7.99
online	5,700,499	4,211,806	0.739	7.99
webmail	7,795,815	6,381,985	0.819	7.99
webresearch	2,414,031	2,414,008	0.999	24.57
webusers	5,697,272	5,127,101	0.900	12.31

5.2. Results and Analysis

In this section, we empirically investigate the performance of DS-FTL in terms of read/write latency, data distribution, and channel utilization. Additionally, we compare the performance of DS-FTL with state-of-the-art schemes.

5.2.1. Read and Write Latency Comparison

To evaluate the performance, we employ different page allocation schemes on eight real trace files under the same SSD environment configuration. The corresponding average read and write latency results are normalized and presented in Figure 10.

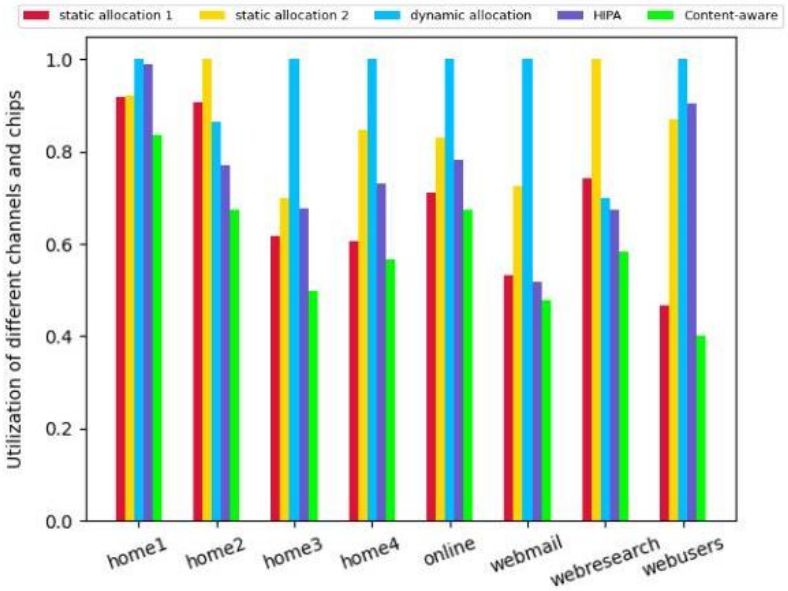
Figure 10a illustrates the write latency under different allocation schemes. In comparison to the two static allocation schemes, DS-FTL exhibits a significant improvement by reducing the average write latency by up to 54.0% and an average reduction of 24.7%. When compared to dynamic allocation, DS-FTL achieves a 36.7% decrease in average write latency, with the largest improvement observed for the webusers trace. In addition, we compare DS-FTL with the HIPA [21], DS-FTL demonstrates a more substantial reduction in write latency of 30.3% for write-intensive workloads (e.g., home3 and webusers), with an overall average write latency reduction of 19.0%.

Figure 10b illustrates the read latency under different allocation schemes. Compared to the two static allocation schemes, DS-FTL can reduce the average read latency time by up to 56.3% and 35.3% on average. Compared with dynamic allocation, the average read latency is reduced by 34.2%. HIPA exhibits better performance in reducing read latency, and DS-FTL shows a similar performance with HIPA. HIPA is even better on online workloads with more read requests, but overall, DS-FTL reduces the read latency by up to 23.0% and 9.8% on average compared to HIPA.

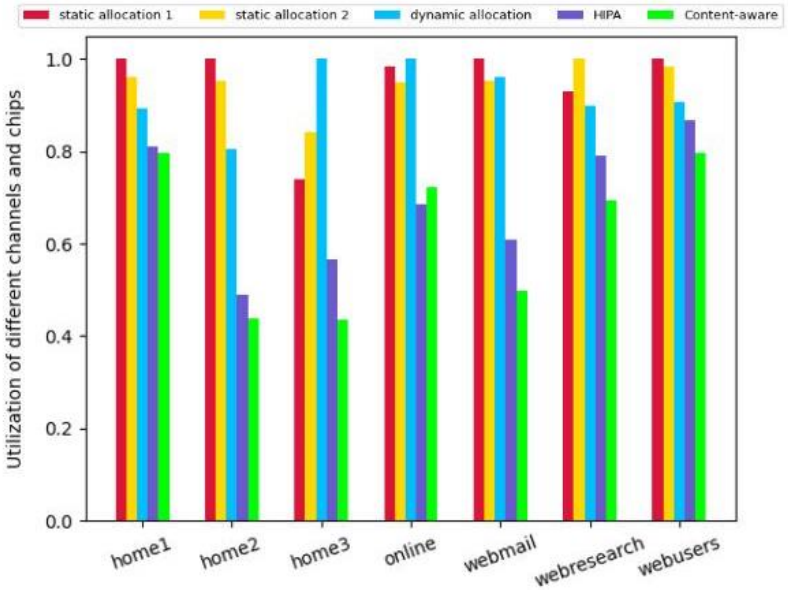
5.2.2. Distribution Degree of Redundancy Data

Figure 11 illustrates the average redundancy rate for the eight workloads under different allocation schemes. The redundancy rate for static allocation ranges from 2.3% to 46.5%, with an average of 18.5%. The high redundancy rate in the static allocation is primarily caused by the assignment of duplicate data with the same LPN to the same planes in webmail workloads. This results in an uneven data distribution, excessive data duplication, and missed opportunities for parallel read access. The dynamic allocation scheme, which employs a round-robin mapping strategy with update tokens, effectively reduces the data duplication rate. The duplication rate in the dynamic allocation ranges from 1.9% to 29.3%, with an average of 13.5%. In contrast, DS-FTL allocates duplicate data to different planes whenever possible, enhancing the efficiency of subsequent read requests. The final duplication rate for DS-FTL ranges from 1.2% to 28.3%, with an average of 12.2%. In comparison to static and dynamic allocation, DS-FTL consistently reduces data

duplication in individual planes across all workloads, resulting in a reduction ranging from 3.4% to 62.8%, with an average reduction of 23.6%.



(**a**)



(**b**)

Figure 10. Read and write latency under different allocation schemes. (**a**) Write Latency. (**b**) Read Latency.

To further investigate the impact of DS-FTL on reducing data duplication, we compare the duplication rates between static allocation and DS-FTL in different planes for the home2 workload, as demonstrated in Figure 12. DS-FTL significantly reduces the duplication rate in planes with a high density of duplicate data, enhancing uniformity in the overall data distribution.

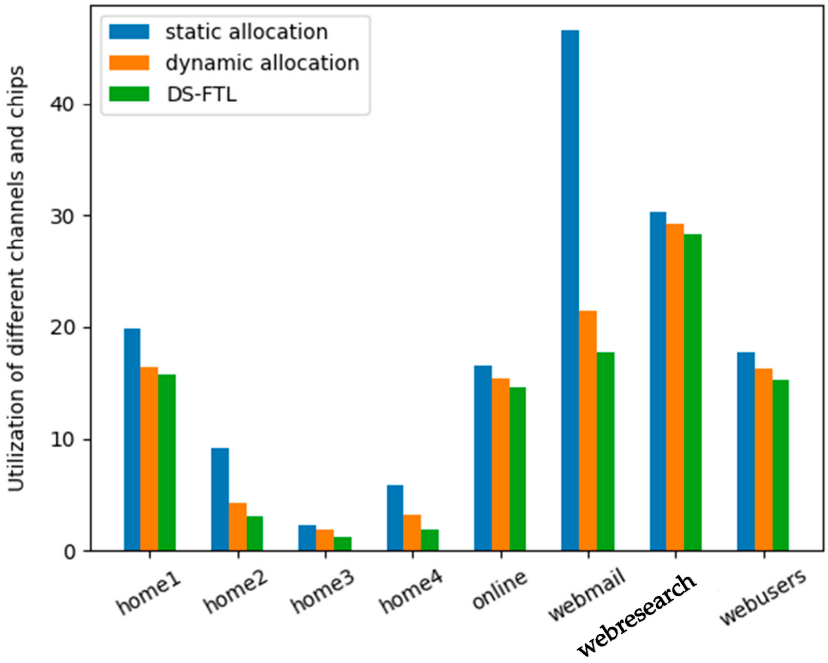


Figure 11. Data redundancy rate under different allocation schemes.

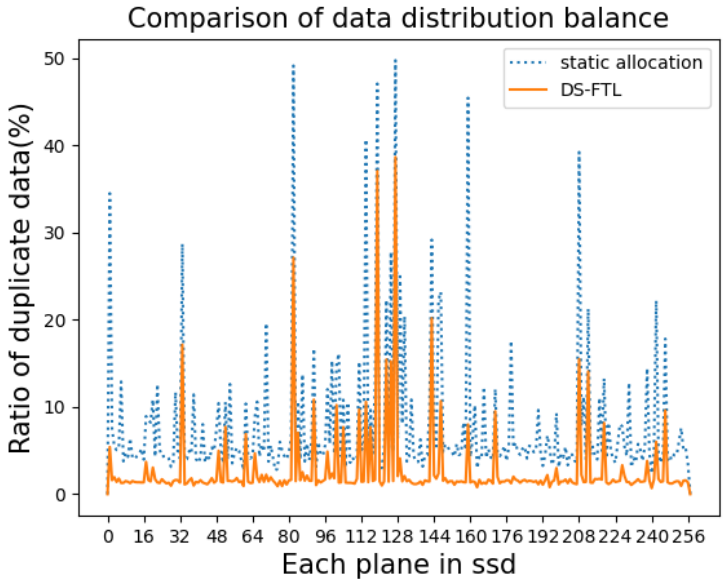


Figure 12. Duplication rate of home2 in each plane.

5.2.3. Proportion of Read and Write Requests That Are Redirected

We evaluated the percentage of redirected read and write requests, as depicted in Figure 13. Redirected requests refer to those where the target PPN is occupied, causing the system to read data from alternative candidate PPNs in the mapping table. As mentioned in the previous subsection, the webmail workload contains a significant amount of duplicate data with the same LPN. To address this, DS-FTL assigns these duplicate data to different planes and reads them from the candidate physical address, resulting in relatively higher redirected read and write request ratios of 15.6% and 29.2%, respectively. On the other hand, the home3 workload exhibits a smaller number of read and write requests, leading to lower probabilities of data duplication within the same plane. Consequently, the final redirected read and write ratios for the home3 workload are smaller, at 1.46% and 1.48%, respectively. In general, workloads with a higher occurrence of duplicate data and a greater

number of write requests tend to have a higher proportion of redirected read, while the proportion is lower for other scenarios. Overall, the average percentage of redirected read and write requests across all eight workloads is 8.21% and 10.5%, respectively.

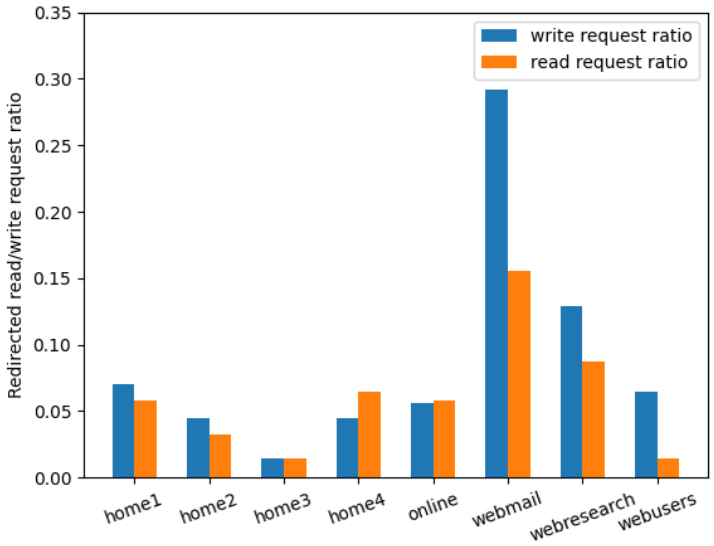


Figure 13. Redirected read/write request ratio.

5.2.4. Comparison of Channel Utilization

Figure 14 illustrates the standard deviation of channel utilization for the eight workloads under different allocation schemes. A higher standard deviation value indicates a greater disparity in channel utilization and a more uneven distribution of read and write requests. As depicted in Figure 14, the standard deviation for static allocation ranges from 0.30 to 3.00, with an average of 1.12. Similarly, the standard deviation for HIPA ranges from 0.23 to 3.52, with an average of 1.11, demonstrating little difference from static allocation. In contrast, DS-FTL effectively reduces the unevenness of channel utilization by allocating duplicate data to different planes. The standard deviation of DS-FTL ranges from 0.22 to 1.31, with an average of 0.61, representing a significant improvement compared to the previous allocation schemes. This reduction in standard deviation demonstrates the improved balance in channel utilization achieved by DS-FTL.

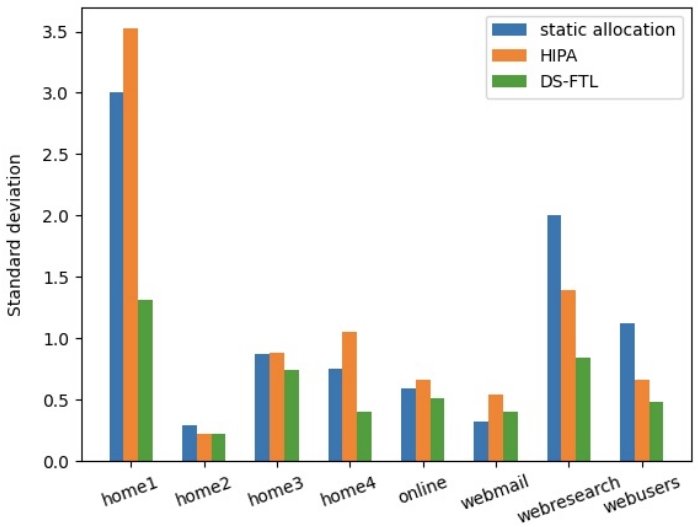


Figure 14. Standard deviation of channel utilization.

6. Conclusions

Motivated by the observation of under-utilization issues in parallel units, such as channels and NAND flash chips, as well as the phenomenon of data duplication, we have developed a data similarity Flash Translation Layer to address the read access conflict problem. Our approach is based on the principle that the FTL distributes page-sized data with identical content across different parallel units and records this information in the mapping table. In DS-FTL, we introduce two novel schemes. The first is the content-aware page allocation scheme, which optimizes storage efficiency and reduces conflicts by intelligently allocating physical addresses based on content similarity. The second is the multi-path read scheme, which improves read performance and utilizes SSD resources effectively by allowing data to be read from any available idle channel or chip through a mapping table lookup for LPNs with multiple corresponding PPNs. This allows for the alleviation of read access conflicts by enabling data to be read from multiple alternative pages in different parallel units. We conducted a series of experiments to evaluate our proposed scheme, and the results demonstrate that it achieves an average read performance improvement of up to 35.3%. Our research is dedicated to enhancing the parallelism and read performance which ignore high-density NAND flash-based SSDs. As high-density NAND flash becomes the mainstream technology, identifying a feasible method of employing DS-FTL in high-density NAND flash-based SSDs has become a key issue, where high-density NAND flash (e.g., TLC, QLC and PLC) exhibits high latency and renders byte-to-byte comparison unacceptable for inline deduplication. We intend to devote our efforts to developing more effective data deduplication schemes for high-density NAND flash-based SSDs.

Author Contributions: Software, Z.Z.; Investigation, W.W.; Experiment, Z.Z.; Data curation, Y.H.; Writing—original draft, J.N.; Writing—review & editing, S.N.; Visualization, C.S. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the No.5 Project: Research on container-based spaceborne lightweight edge cloud technology funded by the Shanghai Key Laboratory of Satellite Network, Shanghai Satellite Network Research Institute Co., Ltd. and China Satellite Network Innovation Co., Ltd., in part by the National Science Foundation of China under Grant 61972311, and in part by the Shandong Provincial Natural Science Foundation, China under Grant ZR2021LZH009.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data is contained within the article.

Acknowledgments: The authors would like to thank the reviewers for their thoughtful comments and efforts toward improving our manuscript.

Conflicts of Interest: Author Yingmeng Hu was employed by the company China Satellite Network Innovation Co., Ltd., author Chenguang Shi was employed by the company Shanghai Key Laboratory of Satellite Network, Shanghai Satellite Network Research Institute Co., Ltd. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Gao, C.; Shi, L.; Zhao, M.; Xue, C.J.; Wu, K.; Sha, E.H.M. Exploiting parallelism in I/O scheduling for access conflict minimization in flash-based solid state drives. In Proceedings of the 30th Symposium on Mass Storage Systems and Technologies (MSST), Santa Clara, CA, USA, 2–6 June 2014; pp. 1–11.
2. Wu, S.; Du, C.; Zhang, W.; Mao, B.; Jiang, H. Deduphr: Exploiting content locality to alleviate read/write interference in deduplication-based flash storage. *IEEE Trans. Comput.* **2021**, *71*, 1332–1343. [CrossRef]
3. Kim, J.; Kang, S.; Park, Y.; Kim, J. Networked SSD: Flash Memory Interconnection Network for High-Bandwidth SSD. In Proceedings of the 55th IEEE/ACM International Symposium on Microarchitecture (MICRO), Chicago, IL, USA, 1–5 October 2022; pp. 388–403.

4. Feng, C.; Tian, L.; Zhang, X. CAFTL: A Content-Aware Flash Translation Layer Enhancing the Lifespan of Flash Memory based Solid State Drives. In Proceedings of the 9th USENIX Conference on File and Stroage Technologies, San Jose, CA, USA, 15–17 February 2011.
5. Kim, J.; Lee, C.; Lee, S.; Son, I.; Choi, J.; Yoon, S.; Lee, H-u.; Kang, S.; Won, Y.; Cha, J. Deduplication in SSDs: Model and quantitative analysis. In Proceedings of the IEEE 28th Symposium on Mass Storage Systems and Technologies (MSST), Pacific Grove, CA, USA, 16–20 April 2012; pp. 1–12.
6. Ni, F.; Wu, X.; Li, W.; Wang, L.; Jiang, S. WOJ: Enabling Write-Once Full-data Journaling in SSDs by using weak-hashing-based deduplication. *Perform. Eval.* **2018**, *127–128*, 56–69. [CrossRef]
7. Tavakkol, A.; Sadrosadati, M.; Ghose, S.; Kim, J.; Mutlu, O. FLIN: Enabling Fairness and Enhancing Performance in Modern NVMe Solid State Drives. In Proceedings of the ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA), Los Angeles, CA, USA, 1–6 June 2018.
8. Arash, T.; Gómez-Luna, J.; Mohammad, S.; Saugata, G.; Onur, M. MQsim: A framework for enabling realistic studies of modern multi-queue SSD devices. In Proceedings of the FAST, Oakland, CA, USA, 12–15 February 2018.
9. Arash, T.; Mohammad, T.; Hamid, S.-A. Unleashing the potentials of dynamism for page allocation strategies in SSDs. In Proceedings of the SIGMETRICS'14: The 2014 ACM International Conference on Measurement and Modeling of Computer Systems, Austin, TX, USA, 16–20 June 2014; pp. 551–552.
10. Arash, T.; Pooyan, M.; Mohammad, A.; Hamid, S.-A. Performance evaluation of dynamic page allocation strategies in SSDs. *ACM Trans. Model. Perform. Eval. Comput. Syst.* **2016**, *1*, 1–33.
11. Jung, M.; Kandemir, M. An evaluation of different page allocation strategies on high-speed SSDs. In Proceedings of the HotStorage'12: Proceedings of the 4th USENIX Conference on Hot Topics in Storage and File Systems, Berkeley, CA, USA, 13–14 June 2012.
12. Shin, J.Y.; Xia, Z.L.; Xu, N.Y.; Gao, R.; Cai, X.F.; Maeng, S.; Hsu, F.H. FTL design exploration in reconfigurable high-performance SSD for server applications. In Proceedings of the ICS '09: Proceedings of the 23rd International Conference on Supercomputing, Yorktown Heights, NY, USA, 8–12 June 2009.
13. Yang, H.; Hong, J.; Dan, F.; Lei, T.; Shu, P.Z. Performance impact and interplay of SSD parallelism through advanced commands, allocation strategy and data granularity. In Proceedings of the ICS '11: Proceedings of the International Conference on Supercomputing, Tucson, AR, USA, 31 May–4 June 2011.
14. Kim, J.; Jung, M.; Kim, J. Decoupled SSD: Reducing Data Movement on NAND-Based Flash SSD. *IEEE Comput. Archit. Lett.* **2021**, *20*, 150–153. [CrossRef]
15. Gupta, A.; Pisolkar, R.; Urgaonkar, B.; Sivasubramaniam, A. Leveraging Value Locality in Optimizing NAND Flash-based SSDs. In Proceedings of the 9th USENIX Conference on File and Stroage Technologies, San Jose, CA, USA, 15–17 February 2011.
16. Hu, Y.; Jiang, H.; Feng, D.; Tian, L.; Luo, H.; Ren, C. Exploring and Exploiting the Multilevel Parallelism Inside SSDs for Improved Performance and Endurance. *IEEE Trans. Comput.* **2013**, *62*, 1141–1155. [CrossRef]
17. Chang, L.P.; Kuo, T.W. An adaptive striping architecture for flash memory storage systems of embedded systems. In Proceedings of the 8th IEEE Real-Time and Embedded Technology and Applications Symposium, San Jose, CA, USA, 27 September 2002.
18. Reddy, B.R.; Narendra, C.; Prakash, T.; Kavirayani, V.R.C.; Singh, P.N.P. Method and an Apparatus for Analyzing Data to Facilitate Data Allocation in a Storage Device. U.S. Patent 9,582,199, 28 February 2017.
19. Paik, J.Y.; Chung, T.S.; Cho, E.S. Dynamic Allocation Mechanism to Reduce Read Latency in Collaboration With a Device Queue in Multichannel Solid-State Devices. *IEEE Trans.-Comput.-Aided Des. Integr. Circuits Syst.* **2017**, *36*, 600–613. [CrossRef]
20. Wu, F.; Lu, Z.; Zhou, Y.; He, X.; Tan, Z.; Xie, C. OSPADA: One-shot programming aware data allocation policy to improve 3D NAND flash read performance. In Proceedings of the IEEE 36th International Conference on Computer Design (ICCD), Orlando, FL, USA, 7–10 October 2018; pp. 51–58.
21. Sun, H.; Cheng, X.; Zhang, C.; Yue, Y.; Qin, X. HIPA: A hybrid load balancing method in SSDs for improved parallelism performance. *J. Syst. Archit.* **2022**, *131*, 102705. [CrossRef]
22. Wu, G.; He, X. Delta-FTL: Improving SSD lifetime via exploiting content locality. In Proceedings of the EuroSys '12: Proceedings of the 7th ACM European Conference on Computer Systems, Bern, Switzerland, 10–13 April 2012.
23. Chen, Z.; Chen, Z.; Nong, X.; Fang, L. NF-Dedupe: A novel no-fingerprint deduplication scheme for flash-based SSDs. In Proceedings of the IEEE Symposium on Computers and Communication (ISCC), Larnaca, Cyprus, 6–9 July 2015.
24. Schuetz, R.; Oh, H.J.; Kim, J.K.; Pyeon, H.B.; Gillingham, P. HyperLink NAND Flash Architecture for Mass Storage Applications. In Proceedings of the IEEE Non-Volatile Semiconductor Memory Workshop, Monterey, CA, USA, 26–30 August 2007.
25. Gillingham, P.; Chinn, D.; Choi, E.; Kim, J.K.; Macdonald, D.; Oh, H.; Pyeon, H.B.; Schuetz, R. 800 MB/s DDR NAND Flash Memory Multi-Chip Package With Source-Synchronous Interface for Point-to-Point Ring Topology. *IEEE Access* **2013**, *1*, 811–816. [CrossRef]
26. Tavakkol, A.; Arjomand, M.; Sarbazi-Azad, H. Network-on-SSD: A Scalable and High-Performance Communication Design Paradigm for SSDs. *IEEE Comput. Archit. Lett.* **2013**, *12*, 5–8. [CrossRef]
27. Nadig, R.; Sadrosadati, M.; Mao, H.; Ghiasi, N.M.; Tavakkol, A.; Park, J.; Sarbazi-Azad, H.; Luna, J.G.; Mutlu, O. Venice: Improving Solid-State Drive Parallelism at Low Cost via Conflict-Free Accesses. *arXiv* **2023**, arXiv:2305.07768.
28. Bloom, B.H. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM* **1970**, *13*, 422–426. [CrossRef]

29. Wu, S.; Du, C.; Zhu, W.; Zhou, J.; Jiang, H.; Mao, B.; Zeng, L. EaD: ECC-Assisted Deduplication With High Performance and Low Memory Overhead for Ultra-Low Latency Flash Storage. *IEEE Trans. Comput.* **2022**, *72*, 208–221. [CrossRef]
30. Cha, J.; Kang, S. Data randomization scheme for endurance enhancement and interference mitigation of multilevel flash memory devices. *Etri J.* **2013**, *35*, 166–169. [CrossRef]
31. Zhou, Y.; Wu, Q.; Wu, F.; Jiang, H.; Zhou, J.; Xie, C. {Remap-SSD}: Safely and Efficiently Exploiting {SSD} Address Remapping to Eliminate Duplicate Writes. In Proceedings of the FAST, Santa Clara, CA, USA, 23–25 February 2021; pp. 187–202.
32. Traces from SyLab Energy Proportional Storage Systems. Available online: https://drive.google.com/drive/folders/1Ajt0UImAx-ielUKoN0QssPFAhgqTZ-Ef (accessed on 23 October 2023).
33. Koller, R.; Rangaswami, R. I/O deduplication: Utilizing content similarity to improve I/O performance. *ACM Trans. Storage (TOS)* **2010**, *6*, 1–26. [CrossRef]