WARCIP: Write Amplification Reduction by Clustering I/O Pages

Jing Yang, Shuyi Pei and Qing Yang

Dept. of Electrical, Computer, and Biomedical Engineering, University of Rhode Island

Kingston, RI, USA 02881

Shenzhen DAPU Microelectronics Co., Ltd

Shenzhen, China

{jyang,spei,qyang}@ele.uri.edu

ABSTRACT

The storage volume of SSDs has been greatly increased recently with emerging multi-layer 3D triple-level cell and quad-level cell. However, one critical overhead of any flash memory SSD is the garbage collection (GC) process that is necessary due to the inherent physical property of flash memories. GC is a time consuming process that slows down I/O performance and decreases endurance of SSD. To minimize the negative impact of GC, we introduce Write Amplification Reduction by Clustering I/O Pages (WARCIP). The idea is to use a clustering algorithm to minimize the rewrite interval variance of pages in a flash block. As a result, pages in a flash block tend to have a similar lifetime, minimizing write amplification during a garbage collection. We have implemented WARCIP on an enterprise NVMe SSD. Both simulation and measurement experiments have been carried out. Real world I/O traces and standard I/O benchmarks are used in our experiments to assess the potential benefit of WARCIP. Experiment results show that WARCIP reduces write amplification dramatically and the number of block erasures by 4.45 times on average, implying extended lifetimes of flash SSDs.

CCS CONCEPTS

• Information systems → Flash memory;

KEYWORDS

SSD, garbage collection, write amplification

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

SYSTOR '19, June 3–5, 2019, Haifa, Israel

© 2019 Association for Computing Machinery.

ACM ISBN 978-1-4503-6749-3/19/06...\$15.00

<https://doi.org/10.1145/3319647.3325840>

ACM Reference Format:

Jing Yang, Shuyi Pei and Qing Yang. 2019. WARCIP: Write Amplification Reduction by Clustering I/O Pages. In *SYSTOR '19: The 12th ACM International Systems and Storage Conference, June 3–5, 2019, Haifa, Israel*. ACM, New York, NY, USA, 12 pages. <https://doi.org/10.1145/3319647.3325840>

1 INTRODUCTION

Flash memory solid-state drives (SSDs) have fundamentally changed the landscape of data storage market and continue its broader applications due to its increased performance and reduced cost. With the emergence of 3D triple-level cell (TLC) and quad-level cell (QLC) flash technologies, modern SSDs can offer storage capacity of tens of terabytes (TB). In addition to huge volume, SSD's high throughput meets the requirements for enterprise applications by virtue of its massively parallel architecture at the back-end including multiple channels, multiple dies per channel, multiple logic units (LUNs) per die, and multiple planes per LUN. Parallel I/O operations can be performed among independent planes giving rise to a great potential for very high I/O performance.

One major concern of SSD's deployment, however, is the performance degradation and variation caused by its unavoidable internal activities, garbage collection (GC). GC consumes a significant amount of back-end bandwidth and increases the latency of regular I/O requests by a factor of 100 [9]. The key reason why GC drags down I/O performance is Write Amplification (WA), because valid pages have to be migrated from a block to be garbage collected (erased). WA not only slows down I/O performance but also decreases the endurance of SSD. Hence, there is a pressing need for minimizing WA and reducing GC overheads.

Extensive research has been reported in the literature on mitigating the impact of GC [8, 12, 16, 19, 22, 35]. In traditional SSDs that have only one appending point (open block) for SSD writes, data pages with various lifetimes are very likely mixed in the same erase unit [15, 37]. Multi-Stream technology [15] emerged recently has multiple open blocks in an SSD and allows data with different stream IDs to be stored in different erase units. Leveraging the

Multi-Stream technology, many techniques have been proposed [15, 17, 25, 37]. Multi-Stream SSD [15] exposes its streams to applications and let applications decide which stream to be used. FStream [25] automatically maps file system generated data to different streams. PCStream [17] automates the stream selection based on program counters. AutoStream [37] provides an automated streams classification strategy based on data updating frequency and recency to assign data to different streams.

While existing researches have made advances in terms of minimizing WA and GC cost, rapid changing I/O workloads in ever widening applications still impose many challenges to enterprise SSD designs. First of all, due to stream resource limitations, data with different lifetimes may be assigned to the same stream, especially in the cloud environment. While Multi-Stream SSD only supports 8 to 16 streams, dozens of processes may write simultaneously [6, 17, 28, 38]. There is a high chance that two processes with different data update intervals share the same stream ID. Furthermore, all existing automatic stream management strategies classify writing data into a limited number of streams with fixed classification rules. Each stream covers a relatively large range of data's lifetimes and cannot be quickly self-optimized for different workloads. In this situation, when workload changes, some streams can be extremely busy while others are idle wasting the stream resources.

To tackle these challenges and make SSD a viable solution for enterprises with steadily sustainable I/O performance and satisfactory endurance, we introduce Write Amplification Reduction by Clustering I/O Pages (WARCIP). The principal idea of WARCIP is minimizing rewrite interval (RWI) variance of pages in a flash block. RWI is the time interval of two consecutive write requests for the same logical block address. WARCIP calculates RWI of each write request in the I/O stream and groups it into an open block whose average RWI is closest. What is more, WARCIP learns current I/O patterns to fine-tune its clustering activities adaptively. It merges, splits, and adjusts clusters automatically to achieve the best clustering result. While WARCIP only has several clusters (open blocks) at a time point, it has no upper limit in terms of how many clusters it can support. WARCIP is able to quickly change its grouping strategy to adapt to new workload characteristics by closing several open blocks and open new ones. Therefore, WARCIP can group pages with similar RWIs to the same physical block efficiently and accurately.

In order to assess how efficient the newly proposed WARCIP is in terms of I/O performance and endurance, We have carried out extensive experiments on both simulator and real SSD. Our simulator shares the same FTL design that is used in our in-house enterprise-level SSD prototype. The simulator uses page-level address mapping. Whenever an open block is filled, the simulator chooses the block that has

the lowest erasure count as the new open block. When the number of free blocks is lower than a threshold, the block that has the least valid pages is selected by GC. To support Multi-Stream, 8 open blocks are maintained. After extensive simulation experiments, we have implemented a fully functional WARCIP on our NVMe SSD prototype. The SSD prototype aims at enterprise applications using Marvell's "Zao" NVMe SSD controller and Micron TLC NAND flash memory chips with total storage capacity of 8TB. The fully functioning prototype SSD has PCIe interface at the front end and it has 4GB on-board DDR4 SDRAM. At the back-end, it has 8 channels, 4 dies per channel, 2 luns per die and 4 planes per lun. The SSD is plugged into a PCIe Gen3x8 slot of a storage server that generates I/O activities to the SSD based on standard I/O benchmarks and real-world I/O traces. Experimental results show that write amplification is reduced dramatically and the number of block erasures drops to as low as one-tenth of the traditional SSDs. This paper makes the following contributions.

- Our analyses and measurements have discovered that clustering pages in open blocks is more effective than classifying data streams in the host. Clustering pages into an open block directly reduces rewrite interval variance of pages in the block, minimizing WA.
- A novel self-optimizing and adaptive clustering algorithm is proposed, referred to as Write Amplification Reduction by Clustering I/O Pages (WARCIP). WARCIP can minimize rewrite interval variance in a block by clustering write requests according to their rewrite intervals. It is adaptive to and self-optimized for different workloads through a reward mechanism.
- A fully functioning WARCIP has been implemented in an enterprise PCIe SSD card. Comprehensive experimental evaluations have been carried out to show WARCIP's advantages over SSDs using the state-of-the-art WA minimization techniques.

The rest of this paper is structured as follows. WARCIP's design and implementation will be given in Sections 2 and 3, respectively. Section 4 describes the experimental setup and workload characteristics. In Section 5, experiment results will be presented and discussed. Related works are discussed in Section 6. We conclude our work in Section 7.

2 WARCIP DESIGN

WARCIP seeks to intelligently minimize lifetime variance of pages in a flash block. To efficiently and accurately achieve this, WARCIP takes the following three strategies. Firstly, RWI, the write request interval of the same logical block address, is introduced to quantitatively characterize pages' lifetime based on LBA access patterns (Section 2.2). Secondly,

to meet the stringent time and resource constraints of enterprise SSDs, we propose an on-line self-optimized clustering algorithm which can quickly tune its grouping strategy to adapt to changing I/O patterns (Section 2.3). Thirdly, a rewarding feedback mechanism is used to achieve global optimization (Section 2.4). The overall structure of WARCIP is shown in Figure 1.

2.1 SSD Internal Operations

Before presenting WARCIP, we first give a brief overview of internal operations of modern SSD that are essential for understanding WARCIP design.

Log-structured Programming: NAND flash memory cells can only be changed in one direction (e.g., from 1 to 0 or 11 to 10, but not vice versa). No in-place write (program) can be executed in NAND flash based SSD. Hence, each write request requires a new page (clean page) to store new data and turns the corresponding obsolete data from valid to invalid if any. Flash translation layer (FTL) is designed to manage this one-way write manner. It maintains a mapping from logical address to physical address and tracks the status of each page. To facilitate the work of FTL, SSD organizes new writes in a log-structured way, new data is always appended at the end of an open block.

Garbage Collection and Write Amplification: When the number of clean blocks reduces to a certain amount, GC is triggered to generate clean pages [36]. The block chosen for GC often contains valid pages. Before erase operation can be started on that block, valid pages have to be moved out. Valid pages migration is known as write amplification (WA) because of the extra writes of the same content. The more valid pages there are in an erasing block, the larger WA will be. WA consumes SSD back-end bandwidth and drags down its I/O performance significantly.

2.2 Rewrite Interval Measurement

To properly predict the lifetime of a flash page for optimal clustering, intuition and experiments should be used to analyze typical I/O behaviors, especially in different environments. For example, users typically edit files and periodically

save their edited files to persistent storage. When such operation is performed, a bunch of pages will be written to the storage at the same time with the same update cycle in adjacent locations. For a server equipped with buffer cache and page cache, cache management algorithms flush a bunch of pages across a large logical address range. In online transaction processing, many small transactions submitted at the same time generate mixed pages and store them in a random way. The same situation (pages mixed) happens in the virtual machine environments such as VMWare and Xen. In all above scenarios, LBAs of these write requests may or may not be in proximity address range. Even though write requests arrive at the same time, they have different update cycles. Therefore a fine-grained page-level rewrite interval measurement is necessary to calculate update cycles accurately.

It is ideal to measure the lifetime of data at page-level, which allows maintaining the lifetime of every page in SSD. However, tracking at page-level is very costly in terms of DRAM usage. Although DRAM price is decreasing and enterprise SSDs have multiple GBs of on board DRAM, such DRAM is still precious resources that store FTL table and cache data. Analyzing 18 cloud server traces [4] from a production environment, we observed that average write request size of most servers is larger than 8KB. Therefore, we measure rewrite intervals on multiple pages to reduce DRAM consumption while maintaining sufficient accuracy. Specifically, WARCIP maintains a time stamp for 8 consecutive logical pages. These 8 consecutive pages form the basic measuring unit for our WARCIP implementation. A 4-byte time stamp is kept associated with each measuring unit and is refreshed whenever any of the 8 pages are updated. Based on this time stamp, WARCIP calculates RWI for each unit for the prediction of data life. When a write request arrives, its RWI is calculated by subtracting the recorded time stamp from current system time.

2.3 Clustering

Clustering data into groups is one of the most fundamental methods of understanding and learning. It has long been used in many scientific fields. The most popular and straightforward clustering algorithms are streaming k -means algorithms [1, 3]. They can cluster a stream into k groups in one-pass. Streaming k -means algorithms are light-weight and practical. They have been successfully used in many applications. However, direct application of streaming k -means algorithms in our situation would not work for the following reasons. First of all, clustering write requests in SSD requires that the size of a cluster is fixed and equals the size of a flash block, but streaming clustering algorithms do not restrict the size of a cluster. Secondly, the number of clusters is unknown

Figure 1: High-level overview of WARCIP.

in advance and depends on the number of flash blocks used. However, streaming clustering algorithms are either required to define cluster number in advance or have very high time complexity. Thirdly, the performance target of our SSD is 1 million IOPS since it is for enterprise applications. Such high IOPS require that the clustering algorithm make a decision in less than one microsecond on an embedded system with computing and storage resource restrictions. Fourthly, since write pattern changes over time, clustering policy needs to be tuned and adjusted continuously to capture new IO patterns correctly.

To deal with these challenges, WARCIP adopts the following two strategies: Greedy Clustering and Dynamic Split-and-Merge as explained in the following paragraphs.

2.3.1 Greedy Clustering. Traditional clustering algorithms cluster data into a limited number of clusters regardless of the size of each cluster. However, in SSD, the size of each cluster should be equal to the size of an open block. In addition, traditional clustering algorithms find solutions iteratively, while SSD has to cluster write requests as soon as they arrive in one shot. In practice, clusters are only effective before the next GC. Dated write requests should not be clustered with current one even they have the same RWI. Therefore, we propose the following greedy clustering strategy.

Assuming there are k open blocks (clusters) in the system, WARCIP calculates the average RWI of each cluster, designated as the *center* of the cluster, u_k . An incoming write request is appended to an open block whose center is closest to the RWI of the write request. The center u_k is immediately updated to the new average RWI. When an open block is full, WARCIP chooses a free block as a new open block and closes the old one. The center of the new open block is initialized equidistantly from the nearest two centers. Although WARCIP only supports k clusters at a certain time point, it can potentially support an unlimited number of clusters by continually opening new clusters with different centers. Each cluster corresponds to one open block that contains multiple pages ranging from 64, 128, to 256 or more, depending on the type of flash memories (e.g., SLC, MLC, or TLC).

2.3.2 Dynamic Split-and-Merge. Write requests arrive continuously with evolving patterns. Clustering on outdated information may not reflect IO pattern changes and result in poor clustering quality. For example, when I/O pattern changes from inactive to active, it is possible that RWIs of all bursty I/O requests are less than the minimum average RWI of all open blocks. If all bursty I/Os go into the same open block, the block becomes extremely busy, while other open blocks may be inactive or idle that wastes the limited open block resources. Moreover, situations become even worse when SSD is full. Our extensive experiments have revealed a very important phenomenon: Flash blocks that absorb bursty

I/Os contributes to a noticeable portion of WA. This is due to the fact that, when SSD is short of free blocks, the block that absorbs bursty I/Os is likely to become erase candidate because it will contain the least number of valid pages in a short time. Although the number of valid pages is small during each GC, it still causes noticeable WA due to the high erasing frequency. In addition, such blocks increase GC activities significantly, which adversely affect front-end I/O performance. If we can separate the bursty I/O requests to more open blocks based on their rewrite interval times, we can further reduce WA.

For this purpose, we introduce a dynamic split-and-merge (D-SAM) mechanism. When WARCIP detects more than half of the write requests are assigned to an open block in a time window, it splits the open block into two by adding a new grouping center and adjusting the old one. As shown in Figure 2(a), a new open block is spun off for the new RWI range, and bursty I/Os are separated to two clusters accordingly. In this scenario, both clusters have lower rewrite interval variances, thus WA is reduced. On the contrary, when I/O pattern changes, some open blocks may become inactive containing pages partially filling the blocks. To better utilize the limited number of open blocks, WARCIP merges such blocks and closes an inactive open block. The merge process is shown in Figure 2(b). When an open block has not received enough pages in a time window, WARCIP closes the cluster by merging it with its closest neighbor cluster as shown in Figure 2(b). D-SAM operation is triggered periodically in our WARCIP implementation. Let N be the number of open blocks that have been closed (written to flash) since the last D-SAM operation. Whenever N reaches a threshold T , D-SAM starts. T can be tuned depending on applications. Our experiments have shown that D-SAM performs very well by setting $T = 256$. That is, the Dynamic Split-and-Merge operation is activated whenever 256 blocks are filled or closed.

2.4 GC Feedback

To this point, WARCIP's clustering has been solely based on current rewrite interval information. Whether or not such clustering delivers the desired performance can be assessed

Figure 2: Examples of split (a) and merge (b) operations.

by what happens at GC time. If the clustering is done well, then the number of valid pages in a GC block is small. Otherwise, we may have a large number of valid pages that have to be migrated, indicating a poor clustering. In order to give proper credits and blames to the past clustering actions so that the system can reach an optimal state, WARCIP is equipped with a feedback and rewarding mechanism, adding another layer of learning capability.

In a properly clustered block, all pages in that flash block should be rewritten in a short time window after one of them is updated. If this were the case, most pages in the block would have been invalidated when the block is to be erased. Such clustering should receive a positive reward. In contrast, if pages are not rewritten before the block is selected to be erased in a GC process, those valid pages need to be copied to other blocks before being erased. This implies those pages have been placed in a wrong cluster. In this case, WARCIP should blame the wrong action of clustering for these pages and give a negative reward to these pages because they have a longer RWI than other pages in the block.

To quantify the rewarding mechanism, positive or negative value should be assigned to the past actions on clustering. Considering the fact that FTL does not maintain the mapping table for invalid physical pages, it may incur extra overhead to provide positive rewards. Our strategy is “no news is good news”, but bad actions should be reported. Reporting bad actions is relatively easy since wrongly grouped pages are still valid and their information is still available in FTL. To feedback negative reward to indicate wrong grouping, WARCIP doubles the RWI of these valid pages, and then they will be grouped with other pages that have larger update intervals in the future. Doubling RWI incurs very little overhead and achieve the purpose of giving negative rewards to the wrong clustering actions.

3 IMPLEMENTATION

In this section, we discuss implementation details of WARCIP by taking into account the practical factors of real systems.

3.1 WARCIP Architecture

The overall WARCIP architecture is shown in Figure 3. WARCIP consists of WARCIP driver, WARCIP dispatcher and WARCIP reward collector. The WARCIP driver executes clustering algorithms as described in Section 2.3. It calculates the RWI of each write request, then assigns it to the open block with the nearest average RWI by tagging the open block number in the NVMe write command. WARCIP dispatcher guides FTL to dispatch write commands to the corresponding open blocks according to the attached block number. WARCIP reward collector module collects the information of WA, such as LBAs of pages that are migrated to other blocks

Figure 3: The system architecture of WARCIP.

due to garbage collection, and waits for polling by WARCIP driver. Clustering algorithms are implemented in the driver because our SSD prototype is still under development (pre-alpha version) and being tuned for better I/O performance. In alpha version, all algorithms will be embedded in FTL which makes SSD with WARCIP a plug and play device.

3.2 WARCIP Driver Initialization

When WARCIP driver is loaded into the operating system for the first time, it obtains the configuration (e.g., page size, number of pages in a block, number of open blocks) of SSD by sending a query to it. WARCIP driver uses this information to allocate the necessary resources and configure the learning logic.

The WARCIP driver maintains time stamps in a table to track rewrite intervals. Each time stamp is a 4-byte variable that keeps the latest access time of a page. As mentioned in Section 2.2, each time stamp tracks 8 consecutive pages. If any of the 8 pages is updated, the corresponding time stamp are updated to the current time. The WARCIP driver also maintains the status of open blocks. The maximum number of open blocks is determined by the SSD. To keep time stamps persistent across reboots, all time stamps are stored in NAND flashes during system shutdown and restored to DRAM after reboot. When a power failure happens, the time stamp table is built from scratch.

3.3 Request Processing

After getting the open block number of current write request, WARCIP embeds it in DWORD 12 of the write command according to the NVM Express specification [33] and sends it to SSD. Inside SSD, WARCIP dispatcher decodes the open block number from the write command and passes it to FTL. FTL finds the corresponding open block and translates the write request’s LBAs to physical page addresses of that block.

3.4 GC Feedback

WARCIP reward collector is implemented in SSD firmware. It collects WA information (LBAs of the pages that need to

migrate) during each GC and sends them back to the WARCIP driver. WARCIP driver periodically takes these LBAs back to host. After getting LBAs of those pages that were migrated during a GC and caused WA, WARCIP driver punishes all migrated page's RWIs by recalculating the RWIs based on current system time. The RWIs of these pages are effectively doubled by subtracting the RWIs from the time stamp that is stored in the table.

4 EXPERIMENTAL SETUP AND WORKLOAD CHARACTERISTICS

Experimental Settings: We have implemented a fully functional WARCIP on our enterprise level NVMe SSD prototype. The NVMe SSD prototype uses Marvell's "Zao" NVMe SSD controller and Micron TLC NAND Flashs with a total storage capacity of 8TB. The fully functioning prototype has 4GB on-board DDR4 SDRAM. At the back-end of the SSD, there are 8 channels, 4 dies per channel, 2 luns per die and 4 planes per lun. Each plane contains 504 flash blocks, each of which holds 2304 flash pages. It is plugged into a PCIe Gen3x8 slot of a Linux server with Intel i7-7700K CPU, 8GB DRAM, EXT4 filesystem and its Kernel version is 4.4. During experiments, the size of SSD is limited to 180GB with 18% over-provisioned (OP) space to make SSD quickly reaches its steady-state (i.e., all pages in SSD have been written at least once, and GC has been triggered).

Since WARCIP is time-dependent, our measurement experiment takes an entire week to complete run through the one week's I/O traces. In order to better evaluate WARCIP's performance under various hardware configurations compared with the state-of-the-art systems and fine-tune WARCIP parameters, we also use a full-scale simulator to generate more performance results for the purpose of analyses and discussions. The simulator uses the same FTL design as the SSD prototype with page-level mapping for address translation and AVL-trees to maintain valid page number, and erasure count for each physical block. The write amplification factor and erasure count are accurate in our simulator since its Flash-Transfer-Layer design is the same as our SSD prototype. Two AVL-trees are used in the FTL: One maintains used blocks based on their valid page counts and the other maintains free blocks ordered based on their ages (erasure counts). Whenever an open block is full, the simulator chooses the youngest block which has the least erasure count as the new open block. 8 open blocks are supported in the simulator. When the number of free blocks is lower than a threshold, which is 6 in our experiments, the simulator grabs the block that has the least valid pages as the erase candidate. To analyze WARCIP under heavy GC loads, we use 1% OP space in the simulator to minimize the impact of OP space.

The hot-cold data classification method [31] is implemented and used as baseline (BL) for our performance comparison purpose. BL distinguishes user write requests from GC write requests and stores them to different blocks accordingly. PCStream [17], FStream [25] and AutoStream [37] are the three state-of-the-art algorithms of I/O streams separation. We chose AutoStream for comparison because AutoStream is a block-level solution same as our work, while PCStream and FStream leverage system call or filesystem that cannot be integrated into SSD. AutoStream's Sequentiality Frequency Recency (SFR) algorithm is implemented both in the simulator and our SSD prototype for comparison purpose. We implemented SFR based on Algorithm 2 presented in [37] and optimizations of pre-defined parameters of SFR such as $decay_{period}$ to make sure SFR gets its best results in our experimental platform.

Workloads: Standard I/O benchmarks and I/O traces from real-world I/O workloads are used in our experiments. The first benchmark we used is TPC-C [30]. TPC-C is a well-known benchmark used to simulate the execution of a set of distributed and online transactions (OLTP). That TPC-C combines five types of transactions and nine types of tables that generate I/O requests with different complexities makes it a good benchmark for pattern-based evaluations.

Since the objective of WARCIP is to minimize rewrite interval variance of pages in a physical block. Traces with different I/O access patterns are essential. Therefore, we choose the traces from Microsoft Research Cambridge (MSRC) data center [23]. These I/O traces were collected from 13 typical servers in the data center and covered a period of one week (168 hours). We believe that MSRC traces are the most suitable trace group available today to evaluate WARCIP for its diversity in I/O patterns and length in execution times. Before starting each measurement experiment using the I/O traces, we warm up the SSD by sequentially writing the entire SSD once so that our experiments were not started on an empty SSD. To minimize the influence of filesystem, all traces are sent to SSD using direct I/O.

5 RESULTS AND DISCUSSIONS

In this section, we report and discuss our experimental results to demonstrate the effectiveness of WARCIP. Performance analysis and comparison will be carried out in terms of write amplification, erasure count, I/O performance, and additional resources required.

5.1 Write Amplification Reduction

To quantitatively evaluate WA reduction of WARCIP, we first run TPC-C benchmark and MSRC traces on our SSD prototype. TPC-C was set up and run for 6 hours on the SSD using 1750 warehouses (~157GB in size) and 128 clients. TPC-C run

on MySQL database with InnoDB as its storage engine. The buffer pool size of InnoDB is 128MB. Thus, a large portion of the database is located in SSD (storage-resident). To enable WARCIP to work on the SSD, the number of open blocks was configured as 4 and 8, respectively. The Write Amplification Factors (WAF) of TPC-C experiments are shown in Figure 4. The WAF of the BL SSD is 1.71 times larger than that of WARCIP with 8 open blocks. Notice that our SSD prototype is still under development. Its current version is pre-alpha with strict GC and user write flow control to allow the prototype to run reliably. The flow control restricts the ratio of GC write requests to user write requests to be less than 2, which means the maximum WAF is limited to 3. We expect that the WAF reduction of WARCIP will be much larger once this restriction is removed when our SSD stabilizes. It is also observed in our experiments that the more open blocks configured, the better performance obtained. As shown in Figure 4, WARCIP with 8 open blocks shows lower WAF than with 4 open blocks. These results indicate that WARCIP can utilize open blocks well to achieve higher WAF reduction. Without loss of generality, we used 8 open blocks in all of the following evaluations.

In order to see how WARCIP reduces WA in real-world I/O workloads, we use MSRC traces to drive our experiments. Three write intensive MSRC traces *prn_1*, *proj_1* and *usr_1* are fed to our SSD prototype after warming up the SSD as mentioned in Section 4. The measured results are listed in Table 1. For all three cases, WARCIP reduces WA dramatically. The extra data migrations of traces *prn_1* and *proj_1* driven tests are less than 1MB, while they are over hundreds to thousands of megabytes on the baseline SSD. Even the maximum WA of WARCIP-SSD (*usr_1*) is less than one-tenth of that of the BL SSD. Fewer data migrations mean less back-end bandwidth consumption that can otherwise be used to serve user I/O requests. As a result, the latency of user I/O requests can be reduced, and I/O throughput is increased. This will be further discussed in later part of this section.

Figure 4: SSD WAF of WARCIP under different open block numbers in TPC-C benchmark (The lower, the better).

Table 1: Write Amplification of three MSRC traces (The amount of data copied during GC operations in megabytes).

	prn_1	proj_1	usr_1
BL	1459.250	863.668	7123.0
WARCIP	0.023	0.441	554.9

Besides comparing our WARCIP with the baseline SSD, we have also compared it with the most recent work on minimizing WA, specifically AutoStream [37]. Since the clustering algorithm of WARCIP is time-dependent, the 36 one week-long MSRC traces may take 36 weeks to complete for one configuration. Therefore, we ran all 36 MSRC traces on our simulator and measured WAs of WARCIP and AutoStream (AS), and compared them with the baseline SSD. We plotted the WA improvement of WARCIP and AutoStream relative to the baseline for all 36 I/O traces as shown in Figure 5. While both WARCIP and AutoStream reduce WA of the I/O traces, WARCIP did a much better job than Autostream in all cases. For two-thirds of the 36 traces, WARCIP is able to improve WA by over 4 times as compared to the baseline SSD. The average WA improvement of WARCIP over the baseline SSD is 4.46 times while AutoStream is 2.79 times. Although the improvements vary across different workloads, WARCIP achieves better results than AutoStream in all situations without exception. These consistent performance improvements can be mainly attributed to the fact that WARCIP is adaptive and self-optimized to different kinds of workloads.

5.2 Erasure Count Reduction

The lifetime of a flash memory SSD is mainly determined by its PE cycles, or erasure counts. Lower WA means fewer data pages being written to flash giving rise to fewer block erasures. In other words, WARCIP can increase the endurance of SSD by having less GC and erasure operations. For example, If WARCIP reduces erasure counts by half, the SSD's lifetime is likely doubled under the same I/O workloads. To illustrate this, we collect the erasure counts of WARCIP and the baseline SSD on our simulator driven by all 36 MSRC traces, and plot them in Figure 6 (read dominated traces that do not trigger erasure operations are not shown here). For better illustration, we divide the results into two groups with different scales. The trace group shown on the left-hand side of Figure 6 has higher write intensity than the group shown on the right-hand side of the figure. Since WARCIP is a learning algorithm that needs write requests to train its clustering model. More write requests give better model accuracy. As shown on the left-hand side of Figure 6, WARCIP reduces the erasure counts up to an order of magnitude for some traces. The overall average reduction of erasure counts of all

Figure 5: SSD Write Amplification improvement in MSRC traces (The higher, the better).

Figure 6: SSD erasure count reduction (BL/WARCIP) by WARCIP in MSRC traces (The higher, the better).

traces shown in the figure is 4.45 times as compared to the baseline SSD.

5.3 Performance

As mentioned previously, GC and WA have a negative impact on front-end I/O performance. Reducing WA can help improving I/O performance. We measured the average I/O throughput of our prototype SSD equipped with WARCIP (WARCIP-SSD) per 10 seconds while running TPC-C benchmark and compared it to AutoStream and the baseline SSD. TPC-C benchmark is set up to run 900 seconds with 60 seconds to warm up, and using 1750 warehouses and 128 clients. Please note that our SSD prototype is still under development, our main objective here is to show the relative performance under the same hardware conditions. The absolute throughput value will be higher after our SSD prototype is fine-tuned. Figure 7 plots the TPC-C experiment results. At the beginning, WARCIP, AutoStream (AS) and the baseline (BL) SSDs have similar throughput around 80MB/s. The throughput of AS is increased to 95MB/s after the 300th second, while BL remained around 80 MB/s. WARCIP accelerated when it learnt the I/O patterns around the 180th second, and its throughput increased gradually to 100MB/s. One may notice that there is a peak throughput of the baseline SSD at around the 150th second and a slowdown at around the 630th second of WARCIP-SSD and AS-SSD. To understand these abnormal

results, we look into the log of the SSD activities during our experiments. During the period around the 150th second, no GC is triggered since there are enough free blocks in the system. All back-end bandwidth is used by the front-end I/Os, giving rise to a high user I/O throughput. This also explains the small peaks throughout the tests. The performance drops of WARCIP and AutoStream are caused by database's internal processing during TPC-C test. The internal activities, such as log flush, changed I/O patterns by increasing queries response time and causing poor grouping results. However, with the D-SAM mechanism, WARCIP is able to adapt to the new pattern in a short period. As shown in Figure 7 from the 700th second to the 800th second, the throughput of WARCIP goes up again after self-optimization while the throughput of AutoStream SSD drops below 90MB/s because it can not adapt to the new I/O patterns automatically.

Besides throughput, we have also measured the response time of three write intensive MSRC traces on our SSD prototype. Table 2 shows the average response times of three traces running in all SSDs. In all cases, WARCIP is much more efficient than AS in terms of reducing read and write response times. We noticed that AS did not show much better performance than the BL for *prn_1* trace because of its lack of adaptivity to changing I/O patterns. The best improvement occurs on the *usr_1* trace. WARCIP reduced read and write response times by 29.1% and 65.0% while AS showed

Figure 7: SSD throughput of TPC-C benchmark (The higher, the better).

11.2% and 51.6% improvement, respectively. The *usr_1* trace contains the most I/O requests (over 45 million) among all the three traces. This indicates that WARCIP can achieve better performance after learning from more I/O requests. Table 2 also reveals that write requests are served faster and gain more improvement than read requests. This is due to the internal mechanism of our SSD prototype. In our SSD prototype, write requests are written to a write buffer and returned, then the SSD controller programs these requests' data to different flash dies in parallel to exploit SSD's internal parallelism. On the other hand, read requests are directly served by the die that contains requested data. Therefore, read requests are more likely to be blocked by programming or erase operations, making it difficult to take advantages of the back-end bandwidth that is released by WARCIP. What is more, when blockage happens, SSD firmware tries to serve all write commands alternatively giving rise to higher priority for write requests to be served.

To further illustrate WARCIP's advantages on reading blockage reduction over the baseline SSD, we ran all MSRC traces in the simulator and measured the number of blocked read requests under WARCIP-SSD and the baseline SSD, respectively. As shown in Figure 8, WARCIP reduces the number of blocked read requests significantly (Y-axis is in log scale). For some traces such as *prn_0*, WARCIP does not suffer from any read blockage but the baseline SSD does. The results also prove that WARCIP can reduce SSD's internal operations and improve the service quality of read requests, which is important for enterprise applications.

5.4 Overhead Analysis

As mentioned in Section 2.2, WARCIP needs to maintain a time stamp for every 8 consecutive logical pages and an average RWI for each open block. In our current implementation, each time stamp takes 4 bytes to record a maximal interval time of 25 days. The total DRAM required for this purpose is 1GB DRAM for our 8TB SSD prototype. In comparison, SFR needs 16 bytes per chunk. To achieve the same level of

Figure 8: Improvement on blockage of read I/Os (The less, the better).

precision as WARCIP (8 consecutive logical pages per unit), AutoStream takes 4GB DRAM space, 4 times larger than that of WARCIP. Regarding CPU usage, the RWI calculator and Clustering decision maker of WARCIP contains a few simple mathematical calculations and comparisons. The average CPU time used for each clustering decision is less than 400ns which is unnoticeable in practice.

6 RELATED WORK

Because of its importance, extensive research has been reported in the literature to reduce the negative impact of GC in flash memory SSDs [14, 20, 34] and to cluster streaming data for data mining [2, 7]. For a comprehensive review, readers are referred to [11, 29, 36]. In this section, we discuss prior researches that are closely related to our work.

Write amplification reduction: SDF [24] and LOCS [32] decrease write amplification by using large write unit (2MB to 8MB). They are designed for LSM-tree based workloads and do not work well on small writes dominated applications such as online transaction processing (OLTP). WA can also be reduced by invalidating pages ahead of GC [19] or only copying pages which do not have modified copies in main memory during a GC operation [35]. However, either way needs the upper layer cache to pass cache information to SSD controller. This requires the modifications in the operating system, which limits the usage. Other researches use greedy reclaiming policy [8, 12, 16] to reduce WA. They choose the block that has the least valid pages for GC. Although greedy reclaiming policy achieves minimum data migration during each GC, it cannot reduce lifetime variance of pages in that block. To eliminate the root cause of WA, separating data according to their RWIs is the way to go, which has been our focus in this work.

Another approach is to provide more free space [10, 18, 21, 26]. Large over-provisioned space increases GC trigger intervals and allows more pages to be invalidated in a block before being erased. Over-provisioning improves overall garbage

Table 2: I/O Response Time (μ s) of three MSRC traces (Less response time indicates better result).

	prn_1			proj_1			usr_1		
	BL	AS	WARCIP	BL	AS	WARCIP	BL	AS	WARCIP
Read	1023	1019	825	1434	1206	1067	2548	2261	1807
Write	528	540	484	389	328	208	985	477	345

collection performance since fewer data migrations are required. However, larger over-provisioning means less storage space available to users or higher cost of SSD. This is clearly not a cost-effective approach.

Write Interval Detection and Separation: SFS [22] separates “hot” and “cold” data to different segments in the buffer cache (DRAM) and writes them to SSD in batches. Nevertheless, SFS defers “small” writes in volatile memory until a segment is filled completely. It does not benefit applications such as the database that requires data to be persistent after a checkpoint or the completion of a flush operation. ETI [27] divides the address space into N hot/cold clusters based on extents. The computation complexity for the ordered search structure may not be suitable for high-end enterprise SSDs that require extremely high IOPS. Multi-Stream SSD [15] exposes open blocks to upper layer applications and let applications choose which block to use. In this way, applications put data with similar life-cycles in the same physical block to reduce write amplification. PCStream [17] automates the stream selection based on program counters in the Linux kernel. FStream [25] identifies journal and meta-data of file systems and maps these data to different streams. AutoStream [37] is transparent to applications for it detects I/O frequency in Linux device driver and divides writes into streams based on predefined parameters. While PCStream, FStream and AutoStream require modifications in the operating system, WARCIP is designed for working inside SSD and can plug and play without any modification of applications and the operating system. It should be also noted that the major distinctive features of WARCIP are its adaptivity to dynamic changing I/O workloads and self-optimization through a reward mechanism. As a result, no parameter tuning is needed for WARCIP to fit to different workloads.

Clustering: The k -means algorithm [13] is one of the most famous clustering algorithms. It calculates k cluster centers iteratively and assigns objects to their nearest centers by measuring the Euclidean Distance between the objects and cluster centers. k -means++ [5] improves the initialization procedure of k -means by selecting cluster centers that are far from each other. k -means++ runs faster than k -means and provides better results. Inspired by k -means++, Ailon et al. provide k -means# [3]. k -means# chooses a subset of the stream in a single pass and achieves a constant approximation to the k -means objective. StreamKM++ [1] is a

two-step stream clustering algorithm based on K -means++. StreamKM++ uses coresets tree and bucket to reduce-and-merge objects. Although the above clustering algorithms can cluster I/O stream, they are not suitable for reducing WA in SSD. To reduce WA, the cluster algorithm needs to have fixed cluster size which is equal to the block size and the number of clusters is unknown in advance. Our proposed WARCIP is designed with these constraints in mind. WARCIP has fixed cluster size with unlimited cluster number. WARCIP can adjust its clustering strategy to fit the changing I/O patterns and change the number of clusters for the purpose of better utilizing open block resources.

7 CONCLUSION

In this paper, we proposed WARCIP, a new approach to minimizing the negative impact of garbage collection on SSD’s I/O performance and endurance. WARCIP groups pages into a block according to their rewrite intervals to reduce write amplification caused by page migrations during garbage collection operations. Through learning the I/O patterns of write streams, WARCIP can reduce rewrite interval variance of pages in a physical block giving rise to less write amplification. We have implemented a fully functional WARCIP on an NVMe SSD card. Both simulation and real measurement experiments have been carried out on WARCIP using real-world I/O traces as well as standard benchmarks. Experiment results show that our WARCIP can reduce write amplification dramatically and decreases the number of block erasures by up to an order of magnitude.

8 ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their valuable comments that helped greatly in improving the quality of the paper. The authors are thankful to Ying Yang, Shaoquan Liu and Yuanpeng Ma for providing guidance to this work. This research is supported in part by the NSF grants CCF-1439011 and CCF-1421823. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF. It is also partly supported by a research contract between URI and Shenzhen Dapu Microelectronics Co., Ltd.

REFERENCES

- [1] Marcel R Ackermann, Marcus Märtens, Christoph Raupach, Kamil Swierkot, Christiane Lammersen, and Christian Sohler. 2012. StreamKM++: A clustering algorithm for data streams. *Journal of Experimental Algorithmics (JEA)* 17 (2012), 2–4.
- [2] Charu C Aggarwal, Jiawei Han, Jianyong Wang, and Philip S Yu. 2003. A framework for clustering evolving data streams. In *Proceedings of the 29th international conference on Very large data bases-Volume 29*. VLDB Endowment, 81–92.
- [3] Nir Ailon, Ragesh Jaiswal, and Claire Monteleoni. 2009. Streaming k-means approximation. In *Advances in neural information processing systems*. 10–18.
- [4] Dulcardo Arteaga and Ming Zhao. 2014. Client-side Flash Caching for Cloud Systems. In *Proceedings of International Conference on Systems and Storage (SYSTOR 2014)*. ACM, New York, NY, USA, Article 7, 11 pages. <https://doi.org/10.1145/2611354.2611372>
- [5] David Arthur and Sergei Vassilvitskii. 2007. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 1027–1035.
- [6] Yingyi Bu, Hongrae Lee, and Jayant Madhavan. 2013. Comparing SSD-placement Strategies to scale a Database-in-the-Cloud. In *Proceedings of the 4th annual Symposium on Cloud Computing*. ACM, 41.
- [7] Yixin Chen and Li Tu. 2007. Density-based clustering for real-time stream data. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 133–142.
- [8] M-L Chiang and R-C Chang. 1999. Cleaning policies in mobile computers using flash memory. *Journal of Systems and Software* 48, 3 (1999), 213–231.
- [9] Jeffrey Dean and Luiz André Barroso. 2013. The Tail at Scale. *Commun. ACM* 56 (2013), 74–80.
- [10] Gary Drossel. 2009. Methodologies for Calculating SSD Useable Life. In *Proc. Storage Developer Conf.*
- [11] Mohammed Ghesmoune, Mustapha Lebbah, and Hanene Azzag. 2016. State-of-the-art on clustering data streams. *Big Data Analytics* 1, 1 (2016), 13.
- [12] Longzhe Han, Yeonseung Ryu, and Keonsoo Yim. 2006. CATA: a garbage collection scheme for flash memory file systems. In *International Conference on Ubiquitous Intelligence and Computing*. Springer, 103–112.
- [13] J. A. Hartigan and M. A. Wong. 1979. Algorithm AS 136: A K-Means Clustering Algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 28, 1 (1979), 100–108.
- [14] Myoungsoo Jung and Joonhyuk Yoo. 2009. Scheduling garbage collection opportunistically to reduce worst-case I/O performance in solid state disks. *Proceedings of IWSSPS* (2009).
- [15] Jeong-Uk Kang, Jeeseok Hyun, Hyunjo Maeng, and Sangyeon Cho. 2014. The Multi-streamed Solid-state Drive. In *Proceedings of the 6th USENIX Conference on Hot Topics in Storage and File Systems (HotStorage'14)*. USENIX Association, Berkeley, CA, USA, 13–13.
- [16] Atsuo Kawaguchi, Shingo Nishioka, and Hiroshi Motoda. 1995. A flash-memory based file system.. In *USENIX*. 155–164.
- [17] Taejin Kim, Duwon Hong, Sangwook Shane Hahn, Myoungjun Chun, Sungjin Lee, Jooyoung Hwang, Jongyoul Lee, and Jihong Kim. 2019. Fully Automatic Stream Management for Multi-Streamed SSDs Using Program Contexts. In *17th USENIX Conference on File and Storage Technologies (FAST 19)*. USENIX Association, Boston, MA, 295–308. <https://www.usenix.org/conference/fast19/presentation/kim-taejin>
- [18] Kingston. 2019. Understanding Over-provisioning (OP). <https://www.kingston.com/us/ssd/overprovisioning>
- [19] Eunji Lee, Julie Kim, Hyokyung Bahn, Sunjin Lee, and Sam H. Noh. 2017. Reducing Write Amplification of Flash Storage Through Cooperative Data Management with NVM. *ACM Trans. Storage* 13, 2, Article 12 (May 2017), 13 pages. <https://doi.org/10.1145/3060146>
- [20] J. Lee, Y. Kim, G. M. Shipman, S. Oral, F. Wang, and J. Kim. 2011. A semi-preemptive garbage collector for solid state drives. In *(IEEE ISPASS) IEEE International Symposium on Performance Analysis of Systems and Software*. 12–21. <https://doi.org/10.1109/ISPASS.2011.5762711>
- [21] Ray Lucchesi. 2011. SSD Flash drives enter the enterprise. *Silverton Consulting*. accessed on 8 (2011), 2008.
- [22] Changwoo Min, Kangnyeon Kim, Hyunjin Cho, Sang-Won Lee, and Young Ik Eom. 2012. SFS: random write considered harmful in solid state drives.. In *FAST*, Vol. 12. 1–16.
- [23] Dushyanth Narayanan, Austin Donnelly, and Antony Rowstron. 2008. Write off-loading: Practical power management for enterprise storage. *ACM Transactions on Storage (TOS)* 4, 3 (2008), 10.
- [24] Jian Ouyang, Shiding Lin, Song Jiang, Zhenyu Hou, Yong Wang, and Yuanzheng Wang. 2014. SDF: Software-defined Flash for Web-scale Internet Storage Systems. In *Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '14)*. ACM, New York, NY, USA, 471–484. <https://doi.org/10.1145/2541940.2541959>
- [25] Eunhee Rho, Kanchan Joshi, Seung-Uk Shin, Nitesh Jagadeesh Shetty, Joo-Young Hwang, Sangyeon Cho, Daniel DG Lee, and Jaehoon Jeong. 2018. FStream: managing flash streams in the file system. In *16th USENIX Conference on File and Storage Technologies*. 257.
- [26] Seagate. 2019. SSD Over-Provisioning And Its Benefits. <https://www.seagate.com/tech-insights/ssd-over-provisioning-benefits-master-ti/>
- [27] Mansour Shafaei, Peter Desnoyers, and Jim Fitzpatrick. 2016. Write amplification reduction in flash-based SSDs through extent-based temperature identification. In *8th {USENIX} Workshop on Hot Topics in Storage and File Systems (HotStorage 16)*.
- [28] Dharma Shukla, Shireesh Thota, Karthik Raman, Madhan Gajendran, Ankur Shah, Sergii Ziuzin, Krishnan Sundaram, Miguel Gonzalez Guajardo, Anna Wawrzyniak, Samer Boshra, et al. 2015. Schema-agnostic indexing with Azure DocumentDB. *Proceedings of the VLDB Endowment* 8, 12 (2015), 1668–1679.
- [29] Jonathan A Silva, Elaine R Faria, Rodrigo C Barros, Eduardo R Hruschka, Andre CPLF De Carvalho, and João Gama. 2013. Data stream clustering: A survey. *ACM Computing Surveys (CSUR)* 46, 1 (2013), 13.
- [30] TPC-C. 2017. tpcc-mysql benchmark. <https://github.com/Percona-Lab/tpcc-mysql>
- [31] Benny Van Houdt. 2014. On the necessity of hot and cold data identification to reduce the write amplification in flash-based SSDs. *Performance Evaluation* 82 (2014), 1–14.
- [32] Peng Wang, Guangyu Sun, Song Jiang, Jian Ouyang, Shiding Lin, Chen Zhang, and Jason Cong. 2014. An Efficient Design and Implementation of LSM-tree Based Key-value Store on Open-channel SSD. In *Proceedings of the Ninth European Conference on Computer Systems (EuroSys '14)*. ACM, New York, NY, USA, Article 16, 14 pages. <https://doi.org/10.1145/2592798.2592804>
- [33] NVM Express Workgroup. 2017. NVM Express specification 1.3 a. <http://www.nvmexpress.org/specifications>.
- [34] Guanying Wu and Xubin He. 2012. Reducing SSD read latency via NAND flash program and erase suspension.. In *FAST*, Vol. 12. 10–10.
- [35] G. Xu, M. Wang, and Y. Liu. 2014. Swap-aware garbage collection algorithm for NAND flash-based consumer electronics. *IEEE Transactions on Consumer Electronics* 60, 1 (February 2014), 60–65. <https://doi.org/10.1109/TCE.2014.6780926>
- [36] Shiqin Yan, Huaicheng Li, Mingzhe Hao, Michael Hao Tong, Swaminathan Sundararaman, Andrew A. Chien, and Haryadi S. Gunawi.

2017. Tiny-Tail Flash: Near-Perfect Elimination of Garbage Collection Tail Latencies in NAND SSDs. In *15th USENIX Conference on File and Storage Technologies (FAST 17)*. USENIX Association, Santa Clara, CA, 15–28.
- [37] Jingpei Yang, Rajinikanth Pandurangan, Changho Choi, and Vijay Balakrishnan. 2017. AutoStream: Automatic Stream Management for Multi-streamed SSDs. In *Proceedings of the 10th ACM International Systems and Storage Conference (SYSTOR '17)*. ACM, New York, NY, USA, Article 3, 11 pages. <https://doi.org/10.1145/3078468.3078469>
- [38] Ning Zhang, Junichi Tatemura, Jignesh Patel, and Hakan Hacigumus. 2014. Re-evaluating designs for multi-tenant OLTP workloads on SSD-based I/O subsystems. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*. ACM, 1383–1394.