Storage Capacity Enhancement of SSD-based Image Deduplication

Berhe Yowhannes Kifle[1] and Chun-Feng Wu[1]

[1]Affiliation not available

October 11, 2024

Storage Capacity Enhancement of SSD-based Image Deduplication

Berhe Yowhannes Kifle, Chun-Feng Wu, *Member, IEEE,*

***Abstract*—As Cyber Physical Systems (CPSs), notably autonomous vehicles, generate increasing volumes of image-based data, efficient storage solutions become paramount. Leveraging high-density white-box SSDs and tailored data deduplication techniques presents a promising approach. However, conventional deduplication methods are ill-equipped to handle image file characteristics effectively, while existing image-specific solutions like imgDedup have limitations in format support and SSD integration. To address these challenges, we propose an SSD-based Approximate Image Deduplication (AID) system. SSD-AID comprises an image-customized deduplication, called AID, and an SSD-aware Image Consolidation. AID combines exact and approximate matching methods, along with optimized content compression, to enhance deduplication efficacy while integrating seamlessly with white-box SSDs. Furthermore, our SSD-aware Image Consolidation optimizes flash page utilization by grouping images based on access frequencies. Experimental results across multiple datasets demonstrate significant improvements in deduplication ratio and SSD page utilization. SSD-AID offers a comprehensive solution for efficient image data management in CPSs, addressing challenges in large-scale storage utilization and minimizing unused space.**

I. Introduction

Cyber Physical Systems (CPSs), like autonomous vehicles [1], [2], are in need of larger and more cost-effective storage solutions to handle the increasing data volumes from image-based applications, such as object or lane detection [3], [4]. Leveraging high-density white-box SSDs [5], such as open channel SSDs [6], [7] and zoned namespaces SSDs [8], [9], [10], and implementing data deduplication techniques offer a promising solution that combines software and hardware co-designing to address this demand. However, conventional data deduplication techniques are generally designed to handle various data types [11], [12]. They may not effectively utilize the unique characteristics of image files, such as their strong spatial locality in the color space. While a state-of-the-art design, imgDedup [13], focuses on image-specific deduplication, particularly image chunking, it only supports JPEG files and overlooks the specific access characteristics within SSDs. This gap presents an opportunity for innovative solutions that comprehensively address image data's unique properties in deduplication, including considerations for color spaces, approximate image processing, and redundancy removal, while also integrating seamlessly with SSDs.

Data deduplication plays a significant role in cloud storage systems [14], including platforms like Dropbox [15] and NetApp [16], where it enhances data density by identifying and removing redundant copies of recurring data file content within storage environments. Techniques for data deduplication can be broadly categorized into file-based and chunk-based approaches. File-based deduplication [17], [18] involves hashing the entire file to create a unique fingerprint, which is then compared with all other fingerprints in the system. While the file-based implementation is straightforward, it typically results in lower deduplication ratios[1][19]. On the other hand, chunk-based deduplication, such as Content-Defined Chunking (CDC)[20], sophistically divides each file into variable-sized chunks and generates fingerprints for each chunk through hashing. Although the chunk-based implementation is more complex than the file-based approach, it tends to offer higher data deduplication ratios[19], [21].

In addition to improving software designs, seamlessly integrating data deduplication techniques and storage devices is crucial to optimizing the whole system. To achieve the integration, white-box storage devices, such as Open-Channel SSDs [6], [7], [22], Zoned-Namespace SSDs [23] and Host-Managed Shingled Magnetic Drives (SMRs) [24], [25], offer design flexibility to let data deduplication techniques directly manage storage devices. Prior works, such as Sparse Indexing [26] and ChunkStash [27], focus on efficiently locating deduplicated data in the storage device, leveraging support from flash memory and/or DRAM. Besides, CAFTL [28] works on merging fingerprint lookups and the physical address translation in SSD to further reduce the cost of locating deduplicated data on SSD. SMR-aware Deduplication scheme [29] separates data chunks associated with different lifetimes in different SMR zones so as to further reduce the SMR space reclamation overhead.

Conventional data deduplication systems are typically designed to manage various data types, including documents, databases, and scientific datasets. However, the distinctive characteristics of image data set it apart, demanding specialized data deduplication systems tailored exclusively for images. One of the challenges arises from compressed image files, which pose particular difficulties for file- and chunk-based deduplication techniques. Compression algorithms aim

B. Y. Kifle is with the Department of Electrical Engineeringg and Computer Science, National Yang Ming Chiao Tung University, Hsinchu, Taiwan. E-mail: jhon.kifle@gmail.com

C.-F. Wu is with the Department of Computer Science, National Yang Ming Chiao Tung University, Hsinchu, Taiwan.
E-mail: cfwu417@cs.nycu.edu.tw

[1]The deduplication ratio is the ratio of the original data size to the size after deduplication.

to reduce the size of image files by encoding data with fewer bits, focusing on minimizing the bit representation of each data block or byte stream. In contrast, deduplication schemes eliminate duplicate content within the image, retaining only unique blocks. Compression algorithms produce a compressed representation of the original data, while deduplication preserves the original pixel values. Additionally, deduplication can further reduce file size when combined with compression algorithms. The current state-of-the-art, "imgDedup" [13], specializes in deduplicating images in the RGB color space. However, our observations reveal that imgDedup solely handles RGB images, hindering efficiency in image enhancement techniques. Additionally, the absence of integration with SSDs may lead to internal fragmentation issues and hurt the lifetime.

To overcome these obstacles, we propose a joint management solution between image deduplication techniques and SSDs, called SSD-based Approximate Image Deduplication system. Note that to technically embrace this solution, we target white-box SSDs. This system comprises two main components: an Approximate Image Deduplication (AID) and an SSD-aware Image Consolidation. AID demonstrates two main contributions. First, it includes an exact matching deduplication and an approximate window matching tailored for smooth-textured image areas. Second, it has a two-stage content compression, providing lightweight or heavy but strong hash functions for different types of content. Finally, our SSD-aware image consolidation works on grouping images sharing similar access frequencies so as to enhance the utilization within each flash page. Experimentation across three datasets demonstrates the efficacy of the proposed scheme. Notably, employing approximate window matching in smooth areas enhances the overall deduplication ratio by 12.3% to 20%. Moreover, optimizing SSD page utilization by consolidating inactive image files reduces total number of page writes by up to 47%. SSD-AID improves the deduplication ratio while handling large-sized image files and boasts the read performance while reading small-sized images. We list the following contributions in this work:

- **Approximate Image Deduplication** Achieves the best of two worlds by boosting responsiveness and increasing the deduplication ratio. We develop a new image deduplication design by comprehensively considering real-time constraints, color space and image spatial locality.
- **SSD-aware Image Consolidation** To vertically integrate with SSDs, this design consolidates small-sized inactive image files to alleviate page fragmentation resulting from the mismatch between image size and SSD page sizes.
- **Fast Responsiveness and High Deduplication Ratio** The combined impact of AID and SSD-aware image consolidation results in a higher deduplication ratio, particularly for large-sized image files, while also reducing the latency of reading image files by enhancing page utilization, especially for accessing small-sized image files. Furthermore, maintaining high page utilization can effectively minimize the number of page writes, thereby mitigating the degradation of the SSD's lifespan.

The rest of this article is structured as follows: Section II presents the background, observation, and motivation of this work. In Section III, we introduce our SSD-based Approximate Image Deduplication system (SSD-AID) to jointly manage image deduplication techniques and open-channel SSDs. Section IV outlines the analysis and experimental results. Finally, Section V provides the conclusion to wrap up this work.

II. Background, Observation, and Motivation

A. Background

1) Data Deduplication: Cyber-Physical Systems (CPSs) typically operate with finite resources, especially concerning storage and computing capabilities. This limitation underscores the importance of effectively managing available resources, particularly in storage utilization. Data deduplication emerges as a pivotal technique in optimizing storage efficiency. It typically encompasses three primary steps: (1) content extraction, (2) content comparison and redundant removal, and (3) metadata maintenance. In traditional data deduplication approaches, chunk-based content extraction is commonly employed. This method involves dividing each input file into fixed-size or variable-sized chunks using techniques like Fixed-Sized Chunking (FSC) [30] or Content-Defined Chunking (CDC) [20], respectively. Chunk-based deduplication divides files into smaller segments, generates fingerprints for each chunk by hashing their contents, and compares these chunks across different files. It then eliminates duplicates, storing only the unique chunks while retaining metadata for the removed ones, thereby maximizing storage efficiency. To facilitate efficient identification of redundant chunks, metadata maintenance becomes crucial. This metadata includes the mapping between fingerprints, chunks, and their respective files, enabling quick and accurate deduplication operations.

Image files, unlike other file types such as documents or scientific datasets, often exhibit distinct spatial locality. Typically, an image comprises background and foreground components. The background tends to feature uniform properties like color, saturation, and contrast, while the foreground contains objects with similar image attributes. This inherent redundancy among adjacent pixels in the background component offers a promising opportunity for achieving higher deduplication ratios. To minimize file size, image formats like JPEG, PNG, and GIF employ compression techniques [31], [32] that eliminate redundant information in similar areas. However, deduplicating compressed image files poses significant challenges, particularly in file- and chunk-based approaches [33]. Compression methods [34], [35] tend to intermingle areas with and without redundancy, complicating conventional deduplication efforts. As a result, specialized deduplication techniques are required to effectively handle the mixed nature of compressed image data and improve deduplication efficacy in storage systems.

The current state-of-the-art in image-based data deduplication, known as "imgDedup"[13], customizes the content extraction process specifically for images. Taking cues from

convolutional neural networks (CNNs)[36], [37], imgDedup adopts a window-based approach to extract pixel values from the RGB color space and generate a hash value for each window using a robust hash function. In this context, a window resembles the kernel concept in CNNs and represents an n-by-n matrix. All extracted windows are stored, allowing for the reconstruction of the original image by assembling the corresponding windows as needed.

2) NAND-Flash Solid State Drives (SSD): NAND-Flash SSDs have become ubiquitous as storage components in Cyber-Physical Systems (CPSs) [38], [39] due to their fast access times and the consistent decrease in price per gigabyte over the years [40]. Throughout the rest of this paper, we will refer to NAND flash-based SSDs simply as SSDs. Within an SSD, NAND flash memory cells serve as the primary units for storing data. Each memory cell technically represents different data bit values by retaining varying numbers of electrons, with the value determined by the voltage sensed within the cell. For instance, a Single-Level Cell (SLC) can discern low and high voltage values (i.e., with and without electrons) to signify binary data '0' and '1', respectively. Advancements in manufacturing have enabled memory cells to store more data bits by detecting multiple voltage levels. For example, contemporary Quad-Level Cell (QLC) cells can classify the number of electrons stored into 16 voltage levels, enabling the storage of 4 bits within each cell.

However, this increased capacity comes at the cost of reduced lifetime due to the complexity of clearly discerning each voltage level. In SSDs, a memory cell's thin oxide layer is responsible for trapping electrons. Repeated program (write) and erase operations degrade this layer's ability to trap electrons, leading to a worn-out cell. Lifetime, measured by the number of program-erase (P/E) cycles an SSD can undergo before the cell wears out, is a critical factor. An MLC cell is more sensitive to voltage fluctuations [41] than an SLC cell because it must distinguish between multiple voltage states within a single cell. While the maximum voltage for both SLC and MLC cells remains the same, the voltage range allocated for representing each MLC state is narrower. This requirement for precise voltage sensitivity ultimately reduces the cell's error-tolerance capability. Consequently, MLC cells exhibit shorter lifespans due to the quicker wear-out of the oxide layer.

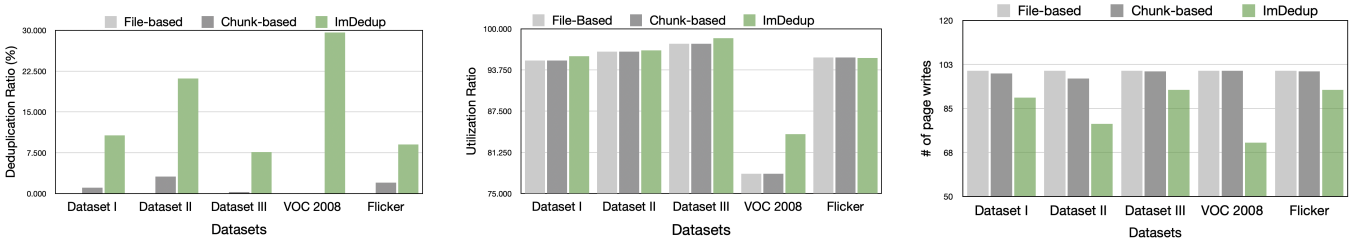
Memory cells attached to the same wordline collectively form a NAND flash page, typically ranging from 4KB to 32KB in size [42], [43]. When a control voltage is applied to memory cells on the same wordline to issue read or write operations, the minimum read/write granularity of an SSD aligns with the NAND flash page size. A NAND flash block comprises several hundred NAND flash pages, and according to the architecture, the erase operation must be applied at the block level. If a NAND flash page has been previously written (or programmed), updating the page shall follow the erase-before-write constraint, where the page shall be erased before writing a new value. However, the disparity between the granularity of write and erase accesses significantly increases the cost of data updating. Instead of directly erasing a block to update data in a NAND flash page, modern SSDs adopt the out-of-place update strategy. Technically, out-of-place update relies on a Read-Modify-Write (RMW) process. This process involves reading the NAND flash page into the SSD's DRAM buffer, updating it with new data, writing it to an unused NAND flash page, and subsequently marking the original page as invalid. By employing this strategy, SSDs can effectively manage data updates without the need for costly block-level erasures.

The RMW process helps to mitigate performance degradation caused by the erase-before-write constraint. However, it turns valid pages into invalid ones, thereby reducing available SSD capacity. To reclaim space occupied by invalid pages, SSDs must run garbage collection (GC) [44], [45], [46]. This involves two main steps: victim block selection and valid-page copy. During garbage collection (GC), SSDs identify multiple used blocks as victims, transfer valid pages from these blocks to unused ones, and then erase all victim blocks. Typically, GC is initiated by the SSD controller rather than by the host CPU, making it application-independent. However, white-box SSDs differ in this aspect, as they delegate GC management—such as triggering GC and selecting victim blocks—to the application rather than handling it internally like conventional SSDs [47], [10], [48].

B. Experimental Observation

The current state-of-the-art method, known as imgdedup, employs a deduplication process that relies solely on exact matches of window areas within an image. In addition, this technique is limited to JPEG image file formats, rendering it ineffective for other image formats. When applied to image files, both file-based and chunk-based deduplication techniques yield suboptimal results. Compression algorithms for image files employ different techniques to generate their byte representations. This variation in byte representation prevents file and chunk-based deduplication methods from effectively recognizing them as duplicates, even if the same image is saved in different formats. As a result, image deduplication tends to achieve a very low deduplication ratio when applied at the byte level. As shown in Figure 1, we evaluate the performance of file-based deduplication, chunk-based deduplication, and imgDedup by analyzing their deduplication ratio, SSD page utilization, and the number of SSD writes. The detailed information of the four datasets is presented in Table I.

Figure1(a) shows an experimental investigation that evaluated the challenges of applying existing approaches to image datasets. The x-axis represents the four image datasets, while the y-axis indicates the deduplication ratio. It is calculated as the ratio of the size difference between the original dataset and the reduplicated dataset, to the size of the original dataset. Both file-based and chunk-based deduplication schemes exhibited notably low deduplication ratios (lower than 10%) compared to the deduplication ratios (more than 20% [33], [29]) generated from 875 file system snapshots by Microsoft Research. It's noteworthy that most of these 875 file systems



(a) Execution on Deduplication Ratio (b) Execution on Page Utilization Ratio (c) Evaluation on Number of Page Writes

Fig. 1: Observational Experiments

are from general users. This disparity can be attributed to the byte-level comparison approach employed by these schemes, which proves ineffective for identifying duplicates in image data files. Particularly due to compression algorithms, even minor modifications in images may go undetected by these schemes. Furthermore, file-based deduplication schemes face a significant challenge wherein a single-byte variation can render a file unidentified. Across the five datasets utilized in the experiment, the deduplication ratios for file-based and chunk-based schemes ranged from 0% to 3%, indicating a lack of duplicate files despite substantial content similarity among the images. These results underscore the unsuitability of file-based and chunk-based schemes for image file formats.

When imgdedup was applied to the four image datasets, the deduplication ratio ranged from 8% to 30%. Although the result associated with imgdedup outperformed file-based and chunk-based deduplication schemes, it still fell short in several aspects. Deduplication ratios from datasets except VOC 2008 were still lower than the deduplication ratio (more than 20%) obtained by running a chunk-based deduplication scheme on file system snapshots by Microsoft Research. Moreover, imgdedup lacks compatibility with different types of images, being limited in its exclusive applicability to JPEG image formats.

Figure 1(b) illustrates the SSD page utilization across the four datasets after implementing the three designs. The x-axis represents the datasets, while the y-axis indicates the SSD page utilization ratio. Meanwhile, Figure 1(c) showcases the total number of SSD page writes for duplicating each dataset using the three designs. The x-axis displays the datasets, and the y-axis denotes the number of SSD page writes. Notably, regardless of the deduplication technique employed, high page utilization is observed after deduplicating large-sized images (Dataset I, II, and III). However, a significant drop in utilization ratio occurs after deduplicating small-sized images (VOC 2008). The reduced utilization factor scores are due to SSD write constraints, where the smallest writable unit is a fixed-size page. This constraint necessitates allocating at least one page for each write operation, so even very small files occupy a full page. Consequently, datasets with small-sized images result in low utilization factor scores. Although imgDedup outperforms conventional data deduplication techniques, the utilization ratio still decreases to below 84%. This decline in page utilization ratio results in extra page writes during image deduplication. Despite the smaller overall size of VOC 2008 compared to the other datasets (around 100x smaller), the total number of page writes generated during the execution of the three techniques closely resembles that of duplicating the larger datasets.

C. Motivation

Traditional data deduplication techniques cannot work well in handling compressed image files. While state-of-the-art research has redirected its focus to developing an image-specialized data deduplication technique called imgDedup. Despite advancements, based on our observations, we've identified four key unsolved challenges in imgdedup. Firstly, applications in autonomous vehicles often adhere to soft real-time constraints [49], [50], allowing for acceptable errors or early termination to meet deadlines. However, imgdedup doesn't fully leverage these characteristics, affecting its responsiveness in such contexts. Secondly, imgdedup exclusively processes RGB images. However, image enhancement and processing techniques, widely adopted in autonomous vehicles to extract information from camera images, are typically more efficient when utilizing YCbCr color spaces [51], [52]. These spaces separate luminosity from color data, potentially leading to more effective analysis and storage utilization. Thirdly, imgdedup encounters significant content comparison and redundancy removal overhead due to its window-based content extraction. This method extracts more content than chunk-based methods, which can result in increased computational costs. Despite this, imgdedup relies on a single heavy hash function designed for chunk-based extraction, impacting the efficiency of content comparison processes. Finally, imgdedup lacks integration with SSDs, which can lead to potential internal page fragmentation issues. This oversight could mix images with different access frequencies on flash pages, necessitating out-of-place updates and potentially causing SSDs to suffer from internal fragmentation without careful data placement strategies.

Addressing these challenges is crucial for enhancing the performance, efficiency, and applicability of imgdedup, particularly in demanding real-time environments such as those found in autonomous vehicle systems [49], [50]. Motivated by these challenges, this work is strongly inspired by the urgent

need to carefully design image-based data deduplication with SSDs for the CPSs. However, the scheme is applicable to any application with image-based processing. The technical contribution falls on (1) how to design an image deduplication technique with taking the image characteristics into consideration (e.g., spatial locality and color space) and (2) how to handle the fragmentation issue via consolidating image files in SSD pages with considering file size and access frequency behaviors. Our ultimate goal is to increase the deduplication ratio for storing more images in the CPS, boost the read performance and alleviate the lifetime degradation of SSDs.

D. Problem definition

1) Window-level Matching: In this section, we provide an explanation of the model formulas applied to address the issue examined in this research. The approach introduced in this study involves the utilization of a similarity checker for windows within an image file. To improve the deduplication ratio, the input image file undergoes decompression to obtain 2-dimensional data of the pixel values constituting the image content. This process is essential for comparing the similarity among window regions of the image.

By generating a signature for each potential window of the image, matches between two windows are identified, with duplicate signatures indicating duplicate content. Each window represents a grid of n by n pixels containing color information from the image file. The procedure involves iterative evaluating all windows within the image and identifying those that exhibit exact or closely resembling content, subsequently eliminating them from the image file. This comparison is carried out exhaustively among all windows until a matching pair is discovered. Hence, the error function can be explained in the following manner:

$$f_{Error} = \sum_{w,h} (w_i(x,y) - w_j(x,y))^2 \quad (1)$$

The windows are defined by their width (w) and height (h) dimensions. The error function quantifies the disparity between each pair of matching pixels in these two windows. This calculated degree of dissimilarity aids in determining whether the windows exhibit a high level of similarity or not. Because the error function is exclusively applied to pixels with lower information content, the threshold value for accepting the error measurement is contingent on the user's choice and preferences.

An image contains regions that convey significant information and regions that are less informative. For the purpose of deduplication, we specifically target the less critical pixel positions within the image. This choice is based on the observation that human perception prioritizes luminance changes over color variations, as they offer essential information. The similarity check and window removal process at this stage necessitates retaining metadata that indicates window positions within the image area. This metadata allows the image to be restored during a read operation. It is stored within the image pixel data to simplify the overall deduplication process and reduce complexity. However, this metadata introduces an overhead, increasing the size of the data file.

2) Victim file selection: Consolidating image files serves to mitigate the wastage of space resulting from internal fragmentation, which would otherwise occur due to the mismatch between the size of image files and the SSD page size. To address this issue and optimize SSD space utilization, the system identifies inactive and small-sized files as candidates for consolidation and packs them into a single file.

This consolidation process can be integrated with the garbage collection operations, where victim files are selected from a designated victim block. By incorporating the image consolidation process into these existing operations, such as during the relocation of valid pages in freed blocks during garbage collection, overall performance is minimized without the need for separate processes. This is attributed to the potential decrease in overall performance that may result from introducing additional management tasks to the SSD system, particularly impacting the efficiency of storage systems.

III. SSD-AID: SSD-based Approximate Image Deduplication System

A. Overview

This subsection shows a design overview of our SSD-based Approximate Image Deduplication system (AID). As shown in Figure 2 SSD-AID includes two main components: AID and SSD-aware image consolidation. The goal of AID is to enhance both responsiveness and the deduplication ratio. Technically, it involves two phases: an Exact_Window_Match to run a coarse-grained file deduplication and an Approx_Chunk_Match to run a fine-grained lossy chunk deduplication on the YCbCr color space. On the other hand, SSD-aware image consolidation aims to enhance SSD page utilization via consolidating files with considering file size and access behaviors. Technically, SSD-aware image consolidation is integrated with SSD's garbage collection process

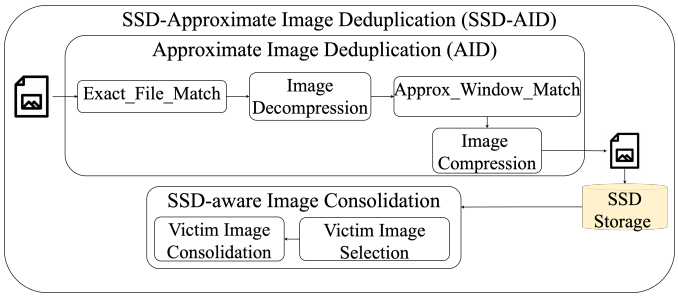


Fig. 2: System Overview

Technically, the metadata produced within the Image deduplication sub-system serves a crucial role by preserving information about the windows of the image, facilitating the reconstruction of images. This becomes particularly essential during image reading when the previously deduplicated or removed windows need to be reinstated. Notably, the metadata's window location value exhibits a lower correlation with the pixel values of the image file. To optimize compression

for the image file, it is advisable to store this metadata in a separate file. This recommendation stems from the fact that compression algorithms capitalize on correlations among pixel values, and integrating metadata within the image file would adversely impact the overall compression rate. Therefore, maintaining a distinction between the image data and metadata ensures a more efficient compression process.

B. Approximate Image Deduplication (SSD-AID)

1) Design Concepts: To enhance responsiveness while maintaining the deduplication ratio, AID incorporates two phases (See Figure 2): Exact_File_Match and Approx_Window_Match. In the Exact_File_Match phase, AID executes the inter-file deduplication to reach a higher deduplication ratio. AID hashes the entire input image file to generate a fingerprint and compares it with all file-based fingerprints in the system. This phase prioritizes fast responsiveness by conducting fingerprint matching at a coarse granularity. However, the complexity of running a sophisticated hash function (such as SHA-1 or even SHA-256) [53], [54] can lead to time-consuming operations aimed at achieving a higher deduplication ratio. To expedite this phase further, we propose a two-stage content comparison method (See Figure 3). Please find more technical details in Section III-C This method utilizes multiple hash functions to efficiently filter out candidates by initially applying simpler hash functions (e.g., MD5) before employing a complex one.

If the Exact_File_Match fails to find a matched file, AID switches from inter-file deduplication to intra-file deduplication. This means that AID divides the image into windows and deduplicates all windows associated with the same image. Image files exhibit strong spatial locality in the 2-dimensional space, making an N-by-N window beneficial for capturing this spatial locality. Conventional chunking solutions operate on a stream of bytes and thus cannot fully utilize this 2-dimensional spatial locality. AID avoids window-based deduplication across files. The reason is that each window would need to be compared with all windows from all files in the system, making duplication time unscalable as the system stores more image files.

Practically, AID decompresses the image file before running the Approx_Window_Match. This allows AID to extract additional image characteristics, such as color space and stronger spatial locality, leveraging the YCbCr color space instead of RGB like imgDedup. To enhance the deduplication ratio, AID divides the decompressed image file into windows using a window-sized kernel (e.g., 3x3, 5x5, 9x9). We evaluate the effectiveness of various window sizes in Section IV-B4 to demonstrate their impact on the deduplication process. Since an image file is divided into multiple windows, the hashing costs increase depending on the number of windows. To maintain responsiveness, AID selects simpler hash functions for this phase (See Figure 3). If AID cannot find a match for a window by comparing fingerprints, it resorts to approximate window matching on those windows covering non-critical features, such as edge detection. This lossy approach aims to identify nearly identical windows with minimal information loss, considering that autonomous driving systems relying on image processing and object detection can tolerate some errors.

Algorithm 1: Approximate Image Deduplication

```
Input  : Image file
Input  : Hash table
Output: Image file
// File-level deduplication;
1  if GetMD5(input_image) in MD5_table then
2      if GetSHA1(input_image) in SHA1_table then
           // Remove file and Update table ;
3          Delete(input_image);
4          Update(SHA1_table);
5  input_image = Decompress(input_image)
   // Dedup if window is exact match ;
6  for W_{i,j} ← input_image do
7      hashCode = GetMD5(W_{i,j});
8      if hashCode.is_exist() then
9          Remove(W_{i,j});
   // Dedup if window is similar ;
10 for W_{i,j} ← input_image do
11     for W_{k,l} ← input_image do
12         if W_{k,l} notDeduped then
13             if W_{k,l} inSmoothArea then
14                 Loss = MSE(W_{i,j}, W_{k,l});
15                 if Loss <= Threshold then
16                     Remove(W_{k,l});
```

Algorithm 1 presents the steps involved in the detailed procedure of running AID. Steps 1-4 execute the file deduplication process by generating a hash code for the file and verifying its existence in the store. An MD5 fingerprint is created for duplicate verification to expedite the process compared to generating SHA-1 hash codes, despite MD5 having a higher collision rate. Encountering duplicate files in images are rare due to frequent modifications to image files. If the images cannot be deduplicated in a file-level matching, AID decompress the image before running the window-level matching (Step 5). Steps 6-9 detail the procedure for eliminating duplicate windows containing exactly matching content throughout the entire image region. To expedite this process, MD5 hashing is utilized to generate a distinctive signature for each window, expediting the identification of duplicates. Windows sharing identical hash codes are deemed exact matches, enabling the removal of redundant windows. Subsequently, regions of the image containing smooth[2] pixel values undergo further

[2]Smooth regions in an image usually represent a non-critical region. Most features are associated with non-smooth regions.

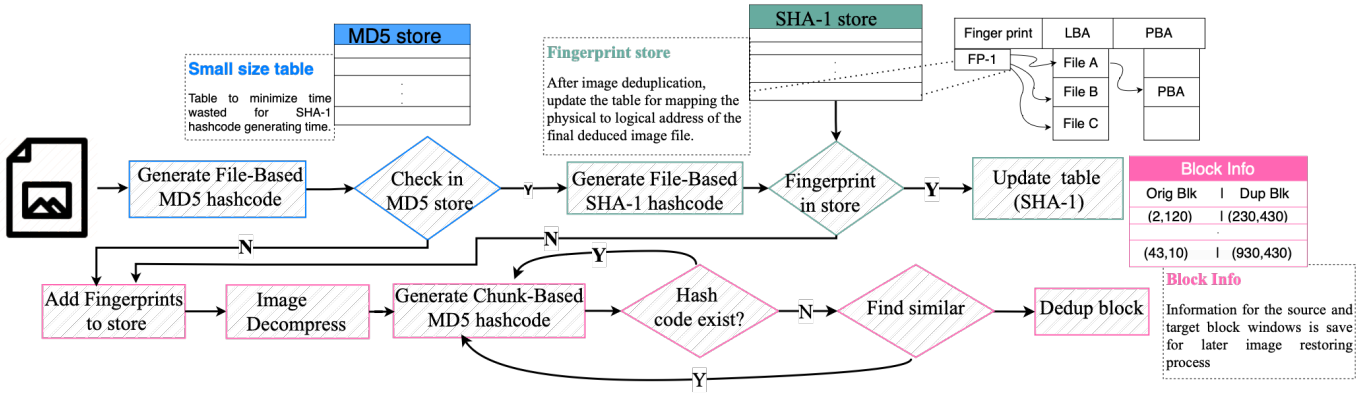


Fig. 3: Approximate Image Deduplication (AID)

examination to identify similar windows with minimal loss of information, customized according to user preferences in Steps 10-16.

2) YCbCr-support Content Extraction: The YCbCr color space divides image data into luminance and chrominance components, where changes in luminance are more perceptible than alterations in chrominance. Unlike RGB processing, YCbCr allows separate treatment of these components, enhancing image processing outcomes. Preserving luminance detail is crucial due to human eye sensitivity, while acceptable chrominance loss is imperceptible. In areas with smooth color content, pixels share nearly identical color data, minimizing deduplication in chrominance channels.

Leveraging this, our system employs comparable matching for chrominance channels with acceptable loss and exact matching for luminance duplicates. Due to gradual color changes, identical windows are less common in luminance than chrominance. Hence, the deduplication ratio is lower with larger window sizes in luminance. To optimize, smaller window sizes are used for luminance deduplication compared to chrominance. This strategy enhances deduplication efficiency, particularly in large image files. Additionally, by considering the human eye's limitations in discerning minor color variations, our approach ensures minimal perceptual loss while maximizing deduplication effectiveness.

3) Two-Stage Content Comparison:

A more efficient method for identifying duplicate files is to generate fingerprints for each file using hash functions like MD5 and SHA-1, avoiding the resource-intensive byte-by-byte comparison. Initially, an MD5 hash code is generated for each file, allowing for quick detection of duplicates. If a match is found, a SHA-1 hash code is subsequently generated to confirm the absence of collisions and validate the duplicate file status. Pre-generating the MD5 code minimizes the computational burden associated with SHA-1, known for its time-consuming process. This Two-Stage Content Comparison method not only conserves computational resources but also expedites the deduplication process. By employing MD5 and SHA-1 hashes, the method enhances efficiency by swiftly identifying duplicate files while minimizing space and time requirements, making it a valuable asset in storage management and optimization. We plan to incorporate this method into the SSD-AID framework, enhancing its capabilities and further streamlining image deduplication processes.

C. SSD-aware Image Consolidation for Inactive Data

The key concept of SSD-aware image consolidation is to pack inactive data together in a page so as to mitigate internal fragmentation. This design is particularly important due to the disparity between file sizes and SSD (Solid State Drive) page sizes. In SSDs, write operations are page-oriented, meaning that even a file smaller than a page size can occupy an entire page, resulting in internal fragmentation. To deal with this issue and minimize wasted space, the approach involves consolidating files identified as inactive data. It's worth noting that the extent of space wastage is influenced by page sizes, which typically range from 4KB to 32KB or even larger [42], [43]. This suggests that inefficiencies in space utilization become more pronounced with larger page sizes. Additionally, consolidating inactive data helps optimize SSD storage utilization and mitigate performance degradation associated with internal fragmentation.

Algorithm 2: File Merge

Input: Metadata for files in victim block , M

```
list = [ ]
for file in M do
    if file inactive then
        if file.size % page.size > 1/3 * page.size then
            list.append(file)
for (file_1,file_2) in list do
    Merge(file_1, file_2);
```

Due to the information gap between SSDs and the host system, the read and write operations within SSD storage

systems lack the flexibility to precisely match the required size for storing a file with the actual size allocated. In SSD systems, these operations are conducted on the smallest unit, known as a page. The size of a page can be determined based on system requirements and hardware specifications, imposing constraints and limitations on read and write operations. To bridge the information gap, SSD-AID relies on using the white-box SSDs.

Efficient storage utilization in SSDs is hindered by the presence of unused space within pages, inaccessible to write operations until the entire block undergoes erasure. This limitation underscores the importance of optimizing page allocation to enhance storage efficiency and minimize wastage. Strategies such as dynamic page resizing or advanced file allocation techniques can be employed to address these inefficiencies effectively. By dynamically adjusting page sizes or implementing precise file allocation methods, SSD systems can better utilize available space, mitigating the impact of unused page space on overall storage performance. These optimization strategies are crucial for maximizing the benefits of SSD technology and ensuring optimal utilization of storage resources in various applications and environments.

Implementing effective strategies to minimize space wastage not only optimizes storage efficiency but also contributes to prolonging the overall lifespan of the system. In our effort to enhance performance, we pinpoint files contributing to internal fragmentation by flagging those exceeding 1/3 of a page in size. Before proceeding with the merging process, we thoroughly evaluate the level of internal fragmentation each file may introduce. This cautious approach is necessary as even minor amounts of unused space within files can potentially impact system performance. It's crucial to avoid compromising system efficiency, especially considering the subsequent step involves reading the merged file, which entails splitting it into separate files—a task requiring both time and space resources. To uphold optimal performance, our focus lies on merging files exhibiting more than 1/3 internal fragmentation within a page, thereby streamlining storage usage and mitigating performance bottlenecks associated with excessive fragmentation.

In Algorithm 2, Steps2-5 pinpoint files that result in underutilized space within the page. Depending on the specifications of the system, the parameters of the scheme can be adjusted to a value conducive to the system's overall performance. For this study, a system utilizing a page size of 4KB is utilized. Taking this into account, the algorithm aims to identify files that could potentially lead to more than 1KB of unused space, designating them as candidates for the file merge process. By targeting such files, the algorithm aims to streamline storage utilization and optimize system performance by reducing wasted space within pages. Additionally, this proactive approach can help enhance the efficiency and longevity of the system by minimizing unnecessary storage fragmentation and wear on storage components. Furthermore, the scheme specifically targets image files of modest size. Managing large files poses a significant challenge in the relocation process, demanding more effort and resources at the management level. Given the scheme's mandate to minimize impact on overall performance, increasing the burden on file relocation could adversely affect system efficiency. To alleviate this strain on the relocation process, the scheme focuses solely on small-sized files for further merging. By concentrating on smaller files, the scheme aims to streamline the relocation process, thus optimizing system performance while ensuring efficient management of storage resources. This approach not only helps maintain system efficiency but also reduces the risk of performance degradation associated with managing larger files.

IV. Performance Evaluation

A. Performance Metrics and Evaluation Setup

In this section, the outcomes of the proposed system are detailed in contrast to prior approaches. The evaluation metrics include deduplication ratio, alleviation of internal fragmentation stemming from the sizing of image files, and the overall space demands for the algorithms across four distinct dataset inputs. The input data-sets are sourced from PASCAL ,Kaggle and youtube video, comprising real-world images. For the testbed, we run file-based deduplication, chunk-based deduplication, imgDedup (or ImDedup for short) and our SSD-AID on a system using Intel Core i7-12700 CPU, 128GB of DRAM, and one SamSung 870 4TB SSD. The operating system is Ubuntu 22.04.1 and the Linux kernel is 6.2.0-33-generic. With following the configuration provided by previous works [55], [29], we use"FIO" (version 3.16) to send requests to the SSD for measuring the overall read/write time. We will open-source the code of SSD-AID once the paper is accepted.

Name	Size	Count	Type	Average size
Dataset I	22GB	9,401	Generated	2.318MB
Dataset II	35.2GB	14,159	Generated	2.43MB
Dataset III	54.3GB	33273	Generated	2.35MB
VOC 2008	420MB	5011	Real world	34.7KB
Flicker	8.99GB	63,567	Real world	135.9KB

TABLE I: Dataset characteristics

Evaluation focus on following metric values:

- **Deduplication ratio** The deduplication ratio is a measure of the algorithm's effectiveness in eliminating duplicate window areas within the image, expressed as:

$$Dedup\ ratio = \frac{Original\ size - Output\ size}{Original\ size} X100\%$$

  Size indicates dataset size before and after the deduplication process.
- **Page utilization** indicates the percentage of SSD pages space used for data. And it is calculated as:

$$Page\ Util = \frac{Used\ space}{Total\ allocated\ space}$$

- **Total SSD write** represents the total space utilized by the SSD system for writing data.
- **Window size Vs deduplication ratio** to show the window size effect on the deduplication ratio.

- **Read and Write performance** measures the number of cycles required to read/write the deduped image datasets.

B. Evaluation Results

1) Image deduplication : Our research findings provide valuable insights into the efficacy of the strategies we've implemented. Through our analysis, we've observed significant enhancements in crucial metrics like deduplication ratio and internal fragmentation reduction, surpassing those achieved by conventional methods. Figure 4 depicts a comparison of the image deduplication ratios relative to two baseline methods: chunk-based deduplication and imgdedup. The x-axis shows four datasets and the y-axis indicates the deuplication ratio. Our proposed system demonstrates a notable increase in the deduplication ratio. This improvement is attributed to our system's emphasis on seeking more deduplication opportunities in areas with lower image information content. The results presented in Figure 4 illustrate that our proposed system enhances the deduplication ratio by a significant margin, ranging from 5% to 17% comparing to imgdedup.

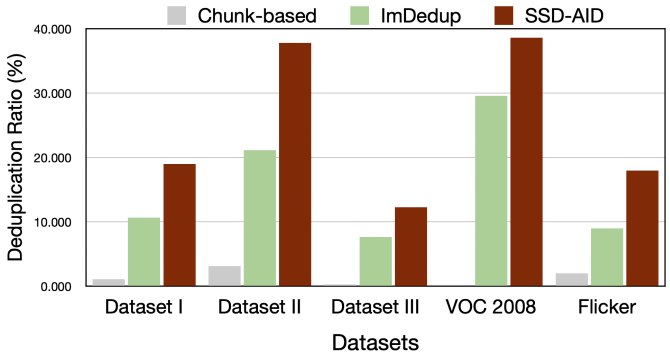


Fig. 4: Image Deduplication Ratio

The process of image deduplication is carried out with a window size of 5x5 and 3x3 pixels for the chrominance and luminance components respectively, chosen to account for the diverse characteristics of images and achieve an overall optimal result by considering an average window size. Our innovative system employs a nuanced deduplication method specifically targeting areas in the image that are less prominent to human eyes. Moreover, the extent of deduplication, coupled with the degree of data loss in these regions, can be adjusted based on user preferences.

2) Page utilization Ratio: Solid-state drive (SSD) systems function by storing data in page-sized units. Unlike conventional hard disk drive (HDD) systems, which can undergo defragmentation, SSD storage exhibits unique characteristics stemming from the simultaneous writing and erasing of cells within a page. This process frequently leaves behind unused cells, leading to what is known as internal fragmentation, a characteristic feature of SSDs. In our proposed system, we leverage the concept of data coldness to identify and merge inactive data segments, which remain unchanged over time and contribute to internal fragmentation exceeding half the size of read/write page size. By deliberately merging these inactive segments with others, we aim to alleviate internal fragmentation and optimize storage efficiency. The effectiveness of our approach is demonstrated in Figure 5, where the x-axis shows four datasets and the y-axis indicates the page utilization ratio. The result showcases a substantial reduction ranging 3%-12% in internal fragmentation compared to conventional systems. Through strategic identification and merging of inactive data segments, our system minimizes wasted space and enhances overall storage performance. This proactive management of internal fragmentation is crucial for maximizing SSD storage utilization and extending the lifespan of storage devices.

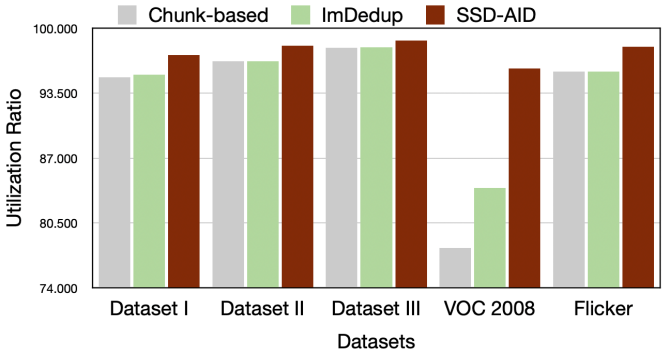


Fig. 5: Page Utilization ratio

The file merge subsystem delivers optimized outcomes, particularly in systems employing larger page sizes. This improvement stems from our system's practice of refraining from allocating a page size when data generates more than 2KB of unused space. In contrast to conventional systems, where image files of 2KB would still be allocated page sizes of 32KB, our scheme mandates page allocation only if data does not exceed 2KB of unused space. This approach ensures more efficient utilization of storage space, especially in scenarios involving smaller amounts of data. By minimizing wasted space and optimizing storage efficiency, our proactive strategy enhances overall system performance, especially in environments with larger page sizes.

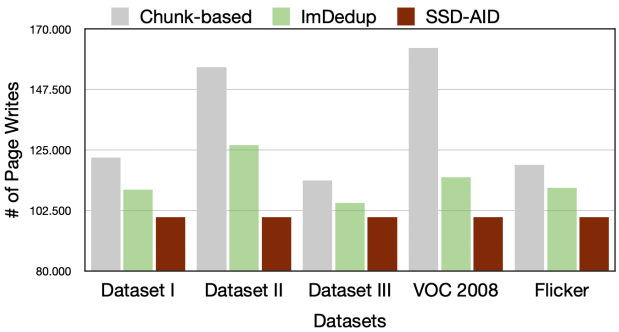


Fig. 6: Overall SSD Page Writes

3) Overall write to SSD : The total write size to the SSD is shown in Figure 6 for the three schemes. The conventional scheme exhibits high sensitivity to page size during write

operations. Similarly, imgDedup struggles to minimize the additional space consumption in write operations. As shown in Figure 6, the combined impact of image-customized deduplication and file consolidation processes significantly reduces the number of page writes. The x-axis shows four datasets and the y-axis indicates the number of SSD page writes. The numeric values indicate that SSD-AID can save 8% to 47% of the page writes compared with the state-of-art work. This result also suggests that running SSD-AID can alleviate the degradation of SSD's lifetime.

Allocating a large space for a page improves the overall write operation, particularly for smaller-sized files, in comparison to previously implemented approaches. This occurs because files marked as inactive and of smaller size are consolidated and grouped together into larger page sizes. This consolidation minimizes the allocation of pages to smaller and inactive files, preventing the creation of unused spaces within pages that would otherwise occur with smaller page sizes. The rationale behind this lies in the observation that larger file sizes result in less internal fragmentation within SSD storage devices.

4) Window size & Deduplication Ratio: The impact of window size on deduplication efficiency was investigated to explore its potential for enhancing deduplication ratios. The imgdedup scheme operates using an 8x8 window size for deduplication purposes. However, the effectiveness of the deduplication ratio overall is contingent upon the suitability of the window size with the contents of the image. Therefore, determining a fixed and universally optimal window size for all images is impractical due to resource limitations and potential performance implications.

In this experiment, an analysis was conducted to identify the most suitable window size for deduplication across datasets, comparing various window sizes against the performance of the current state-of-the-art, imgdedup. With different window size configurations, our proposed solution consistently outperformed imgdedup. This superior performance can be attributed to our solution's approach of deduplication through similarity checks, ensuring that any loss of pixel information remains imperceptible to the human eye.

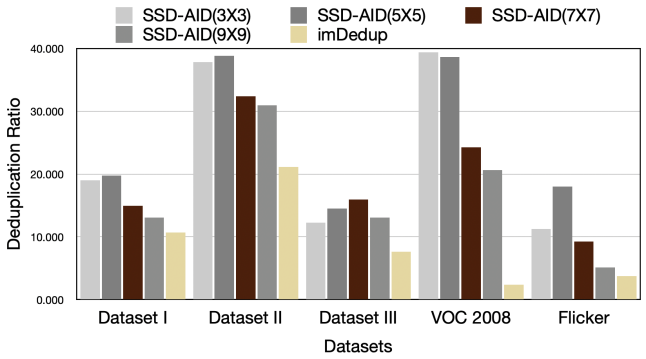


Fig. 7: Window size Vs deduplication ratio

Figure 7 shows the evaluation result, where the x-axis shows four datasets and the y-axis demonstrates the deduplication ratio. The findings indicate that the proposed solution resulted in a deduplication ratio improvement ranging from 3% to 36%. This enhancement is primarily influenced by both the window size and the content of the image. For instance, images containing larger objects and smoother areas are better suited for a larger window size, leading to higher deduplication ratios. Conversely, images featuring intricate details and smaller objects tend to achieve better deduplication ratios when employing a smaller window size. This show that the optimal window size is clearly dependent on the image content. The final size of the image file is influenced by the combined effects of the deduplication ratio, metadata size, and compression ratio. Due to the interplay of these three factors and the characteristics of the two image components, a smaller size of window for luminance and comparatively larger size for the chrominance channels provide optimal results.

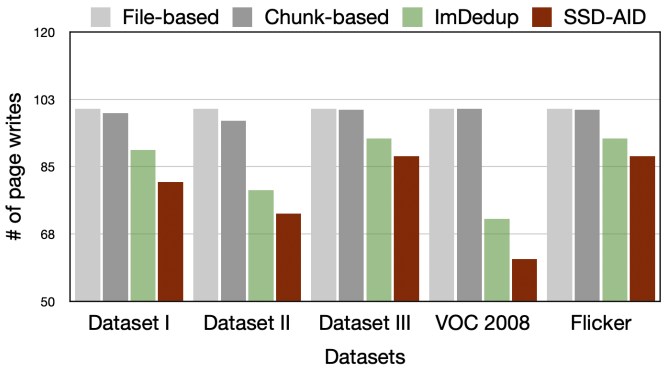


Fig. 8: Overall Write Time

5) Read/write performance: This study comprehensively evaluated the read/write cycles performed by three schemes, as shown in Figur 8 and 9. The x-axis in both figures shows the four datasets and the y-axis indicates the write and read latency, respectively. The proposed scheme demonstrated a reduction in the number of read/write cycles due to the combined impact of the deduplication process and image file consolidation.

Figure 8 illustrates a decrease in the number of write cycles made for the datasets by 4%-10%. This reduction suggests that merging files with prolonged periods of inactivity and those likely to cause internal fragmentation of SSD pages can alleviate the number of program cycles needed in an SSD system. Likewise, the proposed scheme offers a decrease in read cycles. Compared to imgdedup, the number of pages needed to read the datasets is reduced by 3% to 12% as depicted in Figure 9. The scheme exhibits a further reduction in read/write cycles, especially with datasets comprising smaller-sized files.

C. Performance Overhead

The overhead related to the proposed scheme comes from the fact that the scheme retains information about windows that are being removed from the image. The approximate window matching process evaluates every possible window

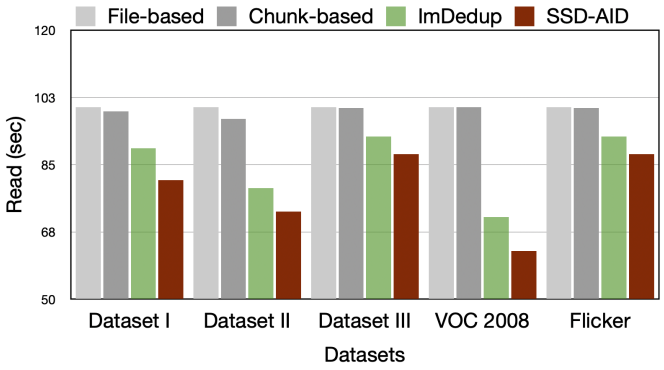


Fig. 9: Overall Read Time

(an area of the image with WxH dimensions) to determine if there are similarities. Similarity is assessed by calculating the MSE of two windows, and windows with an error below a set threshold are considered approximate matches. The AID will then eliminate these windows while retaining information about the eliminated windows within the image area. This allows the image to be restored to its original form if needed. Therefore, the final size discussed in the results includes the total size of the image after deduplication, incorporating the metadata within the image itself.

V. Conclusion

In conclusion, Cyber Physical Systems (CPSs), particularly autonomous vehicles, face escalating data volumes from image-based applications, necessitating larger and more efficient storage solutions. While leveraging high-density white-box SSDs and implementing data deduplication techniques presents a promising approach, conventional methods are not tailored to handle image file characteristics effectively. Existing image-specific solutions like imgDedup have limitations in format support and SSD integration, highlighting the need for innovative solutions comprehensively addressing image data's unique properties in deduplication. We identified four key challenges in imgdedup, including responsiveness in real-time environments, support for diverse color spaces, computational efficiency in content comparison, and integration with SSDs to mitigate fragmentation issues. To address these challenges, we proposed a joint management solution, SSD-based Approximate Image Deduplication system, targeting white-box SSDs. Our system integrates Exact and Approximate Image Deduplication methods with SSD-aware Image Consolidation to enhance deduplication efficacy and SSD utilization. Experimental results demonstrate up to a 20% increase in deduplication ratios and a substantial 47% of reduction on page writes, addressing the challenges of efficient SSD utilization for image storage in CPSs.

References

[1] J. M. Bradley and E. M. Atkins, "Optimization and control of cyber-physical vehicle systems," *Sensors*, vol. 15, no. 9, pp. 23 020–23 049, 2015.

[2] S. A. Seshia, S. Hu, W. Li, and Q. Zhu, "Design automation of cyber-physical systems: Challenges, advances, and opportunities," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 9, pp. 1421–1434, 2016.

[3] W. Xu, H. Zhou, N. Cheng, F. Lyu, W. Shi, J. Chen, and X. Shen, "Internet of vehicles in big data era," *IEEE/CAA Journal of Automatica Sinica*, vol. 5, no. 1, pp. 19–35, 2017.

[4] N. J. Zakaria, M. I. Shapiai, R. Abd Ghani, M. N. M. Yassin, M. Z. Ibrahim, and N. Wahid, "Lane detection in autonomous vehicles: A systematic review," *IEEE access*, vol. 11, pp. 3729–3765, 2023.

[5] I. L. Picoli, N. Hedam, P. Bonnet, and P. Tözün, "Open-channel ssd (what is it good for)," in *Conference on Innovative Data Systems Research*, 2020.

[6] M. Bjørling, J. Gonzalez, and P. Bonnet, "{LightNVM}: The linux {Open-Channel}{SSD} subsystem," in *15th USENIX Conference on File and Storage Technologies (FAST 17)*, 2017, pp. 359–374.

[7] J. González and M. Bjørling, "Multi-tenant i/o isolation with open-channel ssds," in *Nonvolatile Memory Workshop (NVMW)*, vol. 68, 2017.

[8] M. Bjørling, "From open-channel ssds to zoned namespaces," in *Proc. Linux Storage Filesyst. Conf.(Vault)*, vol. 1, 2019, p. 20.

[9] H. Bae, J. Kim, M. Kwon, and M. Jung, "What you can't forget: exploiting parallelism for zoned namespaces," in *Proceedings of the 14th ACM Workshop on Hot Topics in Storage and File Systems*, 2022, pp. 79–85.

[10] Y.-C. Chen, C.-F. Wu, Y.-H. Chang, and T.-W. Kuo, "Zonelife: How to utilize data lifetime semantics to make ssds smarter," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2022.

[11] Q. He, Z. Li, and X. Zhang, "Data deduplication techniques," in *2010 international conference on future information technology and management engineering*, vol. 1. IEEE, 2010, pp. 430–433.

[12] W. Xia, H. Jiang, D. Feng, F. Douglis, P. Shilane, Y. Hua, M. Fu, Y. Zhang, and Y. Zhou, "A comprehensive study of the past, present, and future of data deduplication," *Proceedings of the IEEE*, vol. 104, no. 9, pp. 1681–1710, 2016.

[13] C. Deng, Q. Chen, X. Zou, E. Xu, B. Tang, and W. Xia, "imdedup: A lossless deduplication scheme to eliminate fine-grained redundancy among images," in *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 2022, pp. 1071–1084.

[14] C.-F. Wu, T.-C. Hsu, H. Yang, and Y.-C. Chung, "File placement mechanisms for improving write throughputs of cloud storage services based on ceph and hdfs," in *2017 International Conference on Applied System Innovation (ICASI)*. IEEE, 2017, pp. 1725–1728.

[15] "Dropbox, inc., http://www.dropbox.com/."

[16] "Netapp, inc., http://www.netapp.com/."

[17] S. Neuner, M. Schmiedecker, and E. Weippl, "Effectiveness of file-based deduplication in digital forensics," *Security and Communication Networks*, vol. 9, no. 15, pp. 2876–2885, 2016.

[18] X. Chu, I. F. Ilyas, and P. Koutris, "Distributed data deduplication," *Proceedings of the VLDB Endowment*, vol. 9, no. 11, pp. 864–875, 2016.

[19] M. Dutch, "Understanding data deduplication ratios," in *SNIA Data Management Forum*, vol. 7, 2008.

[20] W. Xia, Y. Zhou, H. Jiang, D. Feng, Y. Hua, Y. Hu, Q. Liu, and Y. Zhang, "{FastCDC}: A fast and efficient {Content-Defined} chunking approach for data deduplication," in *2016 USENIX Annual Technical Conference (USENIX ATC 16)*, 2016, pp. 101–114.

[21] D. Kim, S. Song, and B.-Y. Choi, *Data deduplication for data optimization for storage and network systems*. Springer, 2017.

[22] C.-F. Wu, Y.-H. Chang, M.-C. Yang, and T.-W. Kuo, "When storage response time catches up with overall context switch overhead, what is next?" *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 11, pp. 4266–4277, 2020.

[23] M. Bjørling, A. Aghayev, H. Holmberg, A. Ramesh, D. Le Moal, G. R. Ganger, and G. Amvrosiadis, "{ZNS}: Avoiding the block interface tax for flash-based {SSDs}," in *2021 USENIX Annual Technical Conference (USENIX ATC 21)*, 2021, pp. 689–703.

[24] F. Wu, Z. Fan, M.-C. Yang, B. Zhang, X. Ge, and D. H. Du, "Performance evaluation of host aware shingled magnetic recording (ha-smr) drives," *IEEE Transactions on Computers*, vol. 66, no. 11, pp. 1932–1945, 2017.

[25] C.-F. Wu, M.-C. Yang, and Y.-H. Chang, "Improving runtime performance of deduplication system with host-managed smr storage drives,"

in *Proceedings of the 55th Annual Design Automation Conference*, 2018, pp. 1–6.
[26] M. Lillibridge, K. Eshghi, D. Bhagwat, V. Deolalikar, G. Trezis, and P. Camble, "Sparse indexing: Large scale, inline deduplication using sampling and locality." in *Fast*, vol. 9, 2009, pp. 111–123.
[27] B. Debnath, S. Sengupta, and J. Li, "{ChunkStash}: Speeding up inline storage deduplication using flash memory," in *2010 USENIX Annual Technical Conference (USENIX ATC 10)*, 2010.
[28] F. Chen, T. Luo, and X. Zhang, "{CAFTL}: A {Content-Aware} flash translation layer enhancing the lifespan of flash memory based solid state drives," in *9th USENIX Conference on File and Storage Technologies (FAST 11)*, 2011.
[29] C.-F. Wu, M. Kuo, M.-C. Yang, and Y.-H. Chang, "Performance enhancement of smr-based deduplication systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 9, pp. 2835–2848, 2021.
[30] P. Krishnaprasad and B. A. Narayamparambil, "A proposal for improving data deduplication with dual side fixed size chunking algorithm," in *2013 Third International Conference on Advances in Computing and Communications*. IEEE, 2013, pp. 13–16.
[31] G. K. Wallace, "The jpeg still picture compression standard," *IEEE transactions on consumer electronics*, vol. 38, no. 1, pp. xviii–xxxiv, 1992.
[32] J. Miano, *Compressed image file formats: Jpeg, png, gif, xbm, bmp*. Addison-Wesley Professional, 1999.
[33] D. T. Meyer and W. J. Bolosky, "A study of practical deduplication," *ACM Transactions on Storage (ToS)*, vol. 7, no. 4, pp. 1–20, 2012.
[34] M. A. Rahman and M. Hamada, "Lossless image compression techniques: A state-of-the-art survey," *Symmetry*, vol. 11, no. 10, p. 1274, 2019.
[35] D. Perra and J.-M. Frahm, "Cloud-scale image compression through content deduplication." in *BMVC*, 2014.
[36] R. Kaur, J. Bhattacharya, and I. Chana, "Deep cnn based online image deduplication technique for cloud storage system," *Multimedia Tools and Applications*, vol. 81, no. 28, pp. 40 793–40 826, 2022.
[37] C.-F. Wu, M.-C. Yang, Y.-H. Chang, and T.-W. Kuo, "Hot-spot suppression for resource-constrained image recognition devices with nonvolatile memory," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 11, pp. 2567–2577, 2018.
[38] F. C. Delicato, A. Al-Anbuky, I. Kevin, and K. Wang, "Smart cyber–physical systems: toward pervasive intelligence systems," pp. 1134–1139, 2020.
[39] J. Cui, Y. Zhang, L. Shi, C. J. Xue, J. Yang, W. Liu, and L. T. Yang, "Leveraging partial-refresh for performance and lifetime improvement of 3d nand flash memory in cyber-physical systems," *Journal of Systems Architecture*, vol. 103, p. 101685, 2020.
[40] C.-F. Wu, C.-J. Wu, G.-Y. Wei, and D. Brooks, "A joint management middleware to improve training performance of deep recommendation systems with ssds," in *Proceedings of the 59th ACM/IEEE Design Automation Conference*, 2022, pp. 157–162.
[41] Y.-H. Chang and T.-W. Kuo, "A reliable mtd design for mlc flash-memory storage systems," in *Proceedings of the tenth ACM international conference on Embedded software*, 2010, pp. 179–188.
[42] X. Jimenez, D. Novo, and P. Ienne, "Wear unleveling: Improving {NAND} flash lifetime by balancing page endurance," in *12th USENIX Conference on File and Storage Technologies (FAST 14)*, 2014, pp. 47–59.
[43] C.-Y. Liu, J. Kotra, M. Jung, and M. Kandemir, "{PEN}: Design and evaluation of {Partial-Erase} for 3d {NAND-Based} high density {SSDs}," in *16th USENIX Conference on File and Storage Technologies (FAST 18)*, 2018, pp. 67–82.
[44] M.-C. Yang, C.-F. Wu, S.-H. Chen, Y.-L. Lin, C.-W. Chang, and Y.-H. Chang, "On minimizing internal data migrations of flash devices via lifetime-retention harmonization," *IEEE Transactions on Computers*, vol. 70, no. 3, pp. 428–439, 2020.
[45] S.-H. Lim and K.-H. Park, "An efficient nand flash file system for flash memory storage," *IEEE Transactions on Computers*, vol. 55, no. 7, pp. 906–912, 2006.
[46] M.-C. Yang, Y.-M. Chang, C.-W. Tsao, P.-C. Huang, Y.-H. Chang, and T.-W. Kuo, "Garbage collection and wear leveling for flash memory: Past and future," in *2014 International Conference on Smart Computing*. IEEE, 2014, pp. 66–73.
[47] Z. Shen, F. Chen, Y. Jia, and Z. Shao, "Didacache: an integration of device and application for flash-based key-value caching," *ACM Transactions on Storage (TOS)*, vol. 14, no. 3, pp. 1–32, 2018.
[48] J. Zhang, Y. Lu, J. Shu, and X. Qin, "Flashkv: Accelerating kv performance with open-channel ssds," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 16, no. 5s, pp. 1–19, 2017.
[49] M. Teichmann, M. Weber, M. Zoellner, R. Cipolla, and R. Urtasun, "Multinet: Real-time joint semantic reasoning for autonomous driving," in *2018 IEEE intelligent vehicles symposium (IV)*. IEEE, 2018, pp. 1013–1020.
[50] B. Wu, F. Iandola, P. H. Jin, and K. Keutzer, "Squeezedet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2017, pp. 129–137.
[51] Z. A. Alwan, H. M. Farhan, and S. Q. Mahdi, "Color image steganography in ycbcr space," *International Journal of Electrical and Computer Engineering*, vol. 10, no. 1, p. 202, 2020.
[52] C. Patvardhan, P. Kumar, and C. Vasantha Lakshmi, "Effective color image watermarking scheme using ycbcr color space and qr code," *Multimedia Tools and Applications*, vol. 77, pp. 12 655–12 677, 2018.
[53] P. Gupta and S. Kumar, "A comparative analysis of sha and md5 algorithm," *architecture*, vol. 1, no. 5, 2014.
[54] G. P. Reddy, A. Narayana, P. K. Keerthan, B. Vineetha, and P. Honnavalli, "Multiple hashing using sha-256 and md5," in *Advances in Computing and Network Communications: Proceedings of CoCoNet 2020, Volume 1*. Springer, 2021, pp. 643–655.
[55] Q. Xu, H. Siyamwala, M. Ghosh, M. Awasthi, T. Suri, Z. Guz, A. Shayesteh, and V. Balakrishnan, "Performance characterization of hyperscale applicationson on nvme ssds," in *Proceedings of the 2015 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, 2015, pp. 473–474.

Berhe Yowhannes Kifle received his BSc degree in the department of Computer Science and Engineering from Mekelle University, Ethiopia in 2014. He is currently a student at National Yang Ming Chiao Tung University, Taiwan in masters program at the department of Electrical Engineering and Computer Science. He is studying in the master's degree program of Computer Science and Artificial Intelligence. He spent the year 2014 to 2020 as assistant lecturer scholar in the Department of Electrical and Computer Engineering at Debre Tabor University, Ethiopia. His primary research interests include memory/storage systems, embedded systems, operating systems and image system design.

Chun-Feng Wu received his M.S. degree in the Department of Computer Science from National Tsing-Hua University in 2016, and received his Ph.D. degree in the Department of Computer Science and Information Engineering from National Taiwan University, Taipei, Taiwan, in 2021. Currently, he is an assistant professor at the Department of Computer Science, National Yang Ming Chiao Tung University, Hsinchu, Taiwan. Previously, He was a postdoctoral scholar at the Department of Computer Science, Harvard University, Cambridge, USA, from 2021 to 2022. He served in R&D alternative service at the Institute of Information Science, Academia Sinica, Taipei, Taiwan, from 2017 to 2021. His primary research interests include memory/storage systems, embedded systems, operating systems and the next-generation memory/storage architecture designs. He is a member in IEEE.