Reducing Bit Writes in Non-volatile Main Memory by Similarity-aware Compression

Zhangyu Chen, Yu Hua, Pengfei Zuo, Yuanyuan Sun, Yuncheng Guo
Huazhong University of Science and Technology
Corresponding Author: Yu Hua (csyhua@hust.edu.cn)

Abstract—**Various applications use image bitmaps (data containing pixels) in main memory for fast accesses, thereby leading to lots of memory consumption. Unlike legacy DRAM, non-volatile memories (NVMs) have larger capacity. However, NVM writes consume higher energy and latency compared with reads. Existing data compression schemes leverage precise general-purpose data patterns or precision scaling to reduce data sizes, which suffer from limited compression performance for bitmaps due to large variance or serious quality loss. By exploiting the pixel-level similarity due to the analogous contents in adjacent pixels, we propose SimCom, an approximate Similarity-aware Compression scheme, to compress the write accesses to bitmaps in NVMs, thus efficiently improving the memory performance for image/video applications. SimCom reduces the data size by compressing data into base words and runs. The storage costs for small runs are further mitigated by reusing the least significant bits of base words. The adaptive compression scheme handles various data formats without user annotations on data types. Our experimental results with real-world image/video workloads demonstrate the efficacy and efficiency of SimCom.**

I. Introduction

Various applications (e.g., image/video processing, computer vision, and machine learning) operate on pixels and require images to be stored as bitmaps in main memory for fast accesses [1], which demand a large amount of memory in DRAM. Compressing bitmaps via conventional image compression algorithms (e.g., JPEG) is not applicable, since these image-based applications need to access raw images for computation. Compressed images are still required to be restored into bitmaps in memory for application uses.

Non-volatile memories (NVMs), such as PCM and ReRAM, offer high density with near-zero leakage power and are ideal for bitmaps, but the required energy and latency for writes are much higher than that of DRAM. The write process in NVMs requires high power to modify the physical states [2]. Due to the maximal current constraint during programing, the write process involves multiple write units [3]. Therefore, NVMs suffer from high write energy and latency [4]–[6].

Existing schemes propose lossless data compression inside the NVM module controller with general-purpose data patterns to reduce the bit writes and improve the memory performance [5], [6]. However, for write accesses of bitmaps, the partitioned words of data to be written are hard to match the general data patterns due to the large variance, which results in limited compression performance (§II-D). For example, frequent patterns used in frequent pattern compression (FPC) [6] are collected from general applications and do not fit the contents in bitmaps. Base-Delta-Immediate (BDI) [5] compresses data into base words with small deltas. However, for bitmaps, the word sizes in BDI (8/4/2 bytes) mismatch typical pixel sizes (e.g., 3 bytes for RGB images), which leads to large variance. Moreover, image bitmaps have various formats. Data compression designed for one format may fail in others due to the significant changes in data patterns.

Due to the error-tolerance in some workloads, e.g., machine learning, recent approximate storage leverages the precision scaling with data type annotations to reduce data sizes [4]. However, simple precision scaling for image bitmaps decreases the color depth and output quality. Bidirectional precision scaling (BiScaling) [4] partitions data using annotated word sizes and conducts precision scaling for error-tolerant data. Specifically, it truncates the most significant bits (MSB) and least significant bits (LSB) of data. However, pixel contents in bitmaps are often stored using the smallest data type, in which identical MSBs are usually unavailable. Moreover, indiscriminately truncating LSBs reduces the color depth and causes noticeable quality degradation.

To efficiently reduce the bit writes of bitmaps in NVM systems, we propose SimCom, an approximate Similarity-aware Compression scheme leveraging the pixel-level similarity for various data formats without user annotations for data types and accesses to other data blocks. By exploiting the pixel-level similarity, SimCom identifies similar words according to our similarity model and compresses multiple similar words into a **base word** (the representative word for similar words) with a **run** (the number of similar words), thus eliminating the writes of similar words with minor quality loss. The storage costs for small runs are optimized by reusing the LSBs of base words without significant accuracy loss. To handle different data formats, SimCom executes compression modes in parallel and adaptively selects a mode for efficient compression. In addition to images, our approximate compression scheme is also applicable to other error-tolerant data, as long as these data consist of fixed-size data units and the similarity exists in adjacent units. Our results with image/video-based workloads show 18.3%/22.2%/21.1% energy savings and 17.3%/24.9%/28.8% write latency reduction than FPC/BDI/BiScaling.

II. Background and Motivation

A. Bit-write Reduction in NVMs

Due to the higher programming power and time compared with reads, some schemes are proposed to reduce the number of bits to be written in NVMs, including data compression [5], [6] and data encoding [7]. Specifically, data compression is interpreted as the compaction process inside NVM module controllers. The data to be written (from CPU caches or DRAM) to NVMs are compressed into small sizes. FPC [6] compresses static frequent data patterns into short prefix bits. BDI [5] leverages the characteristics of narrow values in arrays to encode each word using the bases with small deltas. However, the data patterns of NVM writes in image-based applications depend on the visual contents and are hard to match the static patterns in existing schemes (e.g., FPC and BDI), thereby leading to limited compression performance (§II-D). In the meantime, data encoding techniques, e.g., Flip-N-Write (FNW) [7], reduce the bit flips by comparing old and new data, which are applicable after data compression. Hence, we focus on data compression in this paper.

B. Image Bitmap Structures

Image bitmaps are widely used as in-memory data storage for images due to fast accesses to pixels [1]. An image bitmap is a pixel storage structure containing the bits for all pixel colors. The pixel color consists of multiple primary colors. The values of one primary color for all pixels comprise a *channel*. For example, a RGB-based color image has 3 channels, i.e., red, green, and blue channels. For each pixel, the number of bits per channel is often 8 (16 for high quality image). In this paper, we use *channel count* (**CC**) to denote the number of channels in an image bitmap and *bits per channel* (**BPC**) to denote the number of bits per channel for each pixel.

C. Approximate Storage

For data tolerating minor errors, e.g., images and videos, the accuracy is possible to be relaxed to improve the efficiency for storage systems [8]. In the context of this paper, approximable data (or error-tolerant data) denote the image bitmaps and video frames. Biased MLC write schemes are proposed to improve the storage density of compressed images [9] or videos [10]. However, these schemes are established based on the significant entropy differences in encoded images/videos, which do not exist in raw bitmaps and frames. Recent scheme [1] exploits inter-block similarity, i.e., the similarity across different cache blocks, to reduce image sizes. However, searching similar data across blocks during each write access to NVMs incurs extra memory accesses and hardware overheads. Moreover, the inter-block similarity is orthogonal to the pixel-level (intra-block) similarity of our work. Biscaling [4] is proposed to compress the data to be written to DRAM. However, indiscriminately reducing the precision of all data significantly decreases the output quality. In the meantime, Biscaling requires user annotations on data types. Unlike them, our SimCom leverages pixel-level similarity without accesses to other data blocks and adaptively selects efficient modes for different data formats without data type annotations.

D. Motivation

To quantitatively evaluate the performance of existing precise compression schemes, we implement FPC [6] and BDI [5] in NVMain [11] and measure the compression ratios of NVM writes in six image-based workloads. More details about the experimental setup are available in §IV-A. As shown in Figure 1, the average compression ratios of FPC and BDI are 94.2% and 99.8%, which means general patterns often mismatch image bitmaps so that the compression performance of most approximable data is poor.

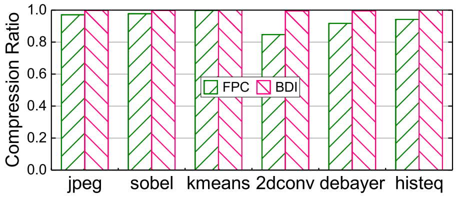

Fig. 1. The compression ratios (compressed data size relative to uncompressed data size) of NVM writes containing image bitmaps using FPC and BDI.

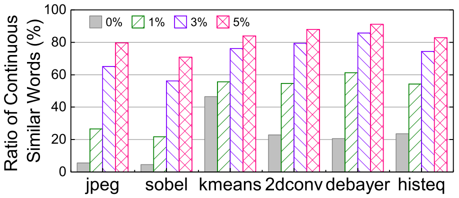

Fig. 2. The ratios of continuous similar words in approximable data with different error thresholds. *(The ratio is interpreted as all continuous similar words divided by total approximable data in bytes.)*

In these image-based workloads, we observe that the data in an NVM write access often show similarities due to the similar contents in the adjacent pixels, called *pixel-level similarity*. In order to verify the existence of pixel-level similarity in NVM writes, we record the continuous similar words in each write access. Specifically, we partition the approximable data at pixel boundaries and determine if two words are similar using Equation 1: if the result of Equation 1 is smaller than the provided error threshold, two words are similar; otherwise, they are not similar words. Figure 2 shows the percentage of continuous similar words in approximable data with different error thresholds. When we increase the error threshold, the ratio of continuous similar words increases up to 82.8% on average. It is worth noting that even when the error threshold is 0% (i.e., two identical words), the ratio of continuous similar words is still more than 4.5% and up to 46.5%. The substantial similarity in images motivates us to exploit the pixel-level similarity for bit-write reduction in NVMs.

III. The SimCom Design

Figure 3 shows the architecture overview of SimCom. The compression and decompression workflows of SimCom are respectively implemented in the *Adaptive Approximate Compression Logic* and *Decompression Logic*. Since NVM module controllers cannot determine whether data are approximable, SimCom requires users to annotate some metadata about image bitmaps (i.e., start and end addresses, error thresholds). These metadata are delivered through memory-mapped registers and stored in the on-chip LRU cache, called *Quality Table* [1], [4]. The quality table contains only a few entries (e.g., 64) so that the hardware overheads are negligible.

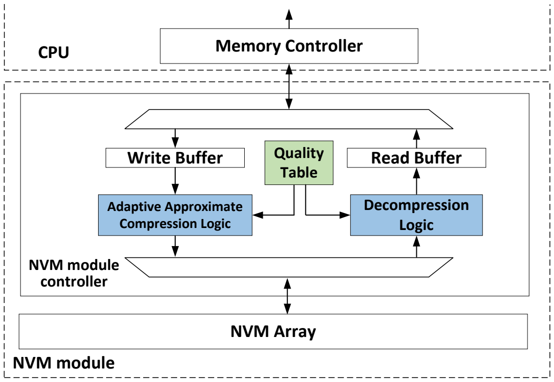

Fig. 3. The architecture overview of SimCom.

A. Similarity Model

In order to quantitatively measure the pixel-level similarity, we propose a model to determine if two words are similar. When a write access arrives at the NVM module controller, the data block (usually 64 bytes) is partitioned at the granularity of pixel size to conserve the pixel-level similarity. Though the ideal partition points are pixel boundaries, it's too complicated to find the boundaries at runtime. We observe that the data form an approximate periodic cycle of the pixel size due to the pixel-level similarity. Therefore, instead of pixel boundaries, we propose to partition the data block at the positions whose indexes are multiples of the pixel size. For example, Figure 4(a) shows a 16-byte data block with the pixel size of 3 bytes and Figure 4(b) presents the partitioned words of the pixel size, among which the similarity still exists. The last byte is left as a partial word.

In our similarity model, we use *normalized difference* to quantify the similarity between two partitioned words. The normalized difference is interpreted as the maximal absolute difference in different channels normalized to the maximal value of a primary color (called *maxValue*). maxValue is a constant determined by BPC. When BPC is 8, maxValue is 255. The normalized difference between words p and q is calculated using Equation 1. The $p[i]$ and $q[i]$ correspond to primary colors in the same channel. If one of the two words is a partial word, "CC" is substituted by the number of channels in the partial word. When the normalized difference is smaller than the error threshold provided by users, two words are similar. The idea behind our similarity model is that primary colors are close if they correspond to similar pixels. By calculating the normalized maximal absolute difference of primary colors, we can get a quantitative estimation of the similarity between words for future compression.

$$normDiff = \frac{max\{|p[i] - q[i]|\}}{maxValue}, i \in [0, CC) \qquad (1)$$

B. Similarity-aware Compression for NVMs

Compression for NVM writes: If the addresses of data to be written are within the regions stored in the quality table, the data are error-tolerant (i.e., approximable data) and compressed using SimCom in the following steps: (1) after data partition, the first word is taken as the initial base word and the run is set to 1. (2) If the next word is similar to current base word according to our similarity model, the run increases by 1; otherwise, take the word as a new base word and reset the run to 1. (3) When a partial word exists and is not similar to current base word, record the partial word in compressed data; otherwise, just increase the run by 1. The number of base words is recorded in the first byte of the compressed data. Figure 4(c) shows the compressed data from 5 continuous similar words (including the partial word).

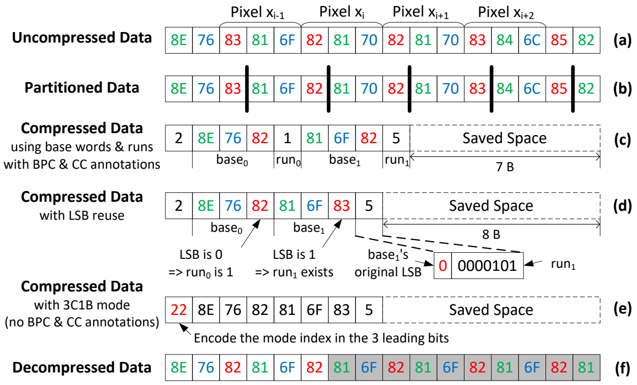

Fig. 4. An example of SimCom compression and decompression for 16-byte data. *(Pixel x_{i-1}-x_{i+2} are four adjacent pixels. $base_i$ denotes a base word. "xCyB" denotes a compression mode for data with "x" channels and "y" bytes for each channel. The threshold for normalized difference is 0.05.)*

The above approximate compression workflow is a variant of run-length encoding: the base word is a representative word for near-duplicate words and the run denotes the repeated time. The compression scheme is efficient for data when the run is large, since multiple similar words are replaced by a base word with a run. However, the storage overheads for runs become high when the runs are small. For example, due to the first run in Figure 4(c), the compressed data for the first partitioned word is one byte larger than the original word. To mitigate the metadata overheads of small runs, we propose to reuse the LSBs of base words to encode small runs. Specifically, the LSB of a base word is used to indicate whether its corresponding run exists. If the LSB of the base word is 1, the run exists in compressed data and we further reuse the MSB of the run to store the original LSB of its corresponding base word (e.g., in Figure 4(d), the original LSB of $base_1$ is stored in the MSB of run_1); otherwise, the run is 1 and not stored (e.g., $base_0$ in Figure 4(d)). As a result, only LSBs of words not similar to adjacent ones (e.g., $base_0$ in Figure 4(d)) are affected and the accuracy loss is limited. Base words with runs larger than 1 are not affected by LSB reuse, since the original LSB is stored in the MSB of corresponding runs (e.g., $base_1$ in Figure 4(d)). According to our evaluation results, the worst quality degradation of reusing 1 LSB (i.e., indiscriminately truncating 1 LSB of all base words) is 1.35% when BPC is 8. Reusing 2 bits leads to nonnegligible quality loss of 2.62%, since the typical output quality constraint for images is 3% [4]. Hence, SimCom only reuses 1 LSB of base words.

If the data are not approximable, the data are compressed using existing precise data compression schemes, e.g., FPC.

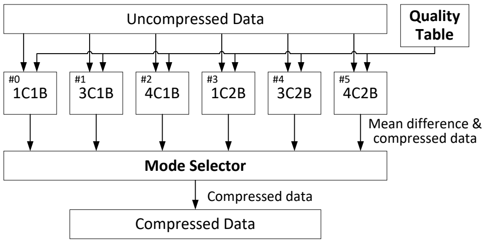

Fig. 5. Adaptive compression scheme overview. *("#n" indicates the index for the compression mode.)*

Decompression for NVM reads: For read accesses to approximable data, which fall within the annotated regions stored in the quality table, the data are restored in the reverse order of approximate compression. Specifically, the base words are used to fill the read buffer, as shown in Figure 4(f). If the base words are not enough to fill up the buffer, which means a partial word not similar to its previous base word is stored in the compressed data, read the encoded partial word and fill the buffer. If the read accesses are for precise data, these data are decompressed using corresponding precise decompression schemes (e.g., FPC).

C. Adaptive Compression Scheme

Our approximate compression scheme presented in previous section (§III-B) efficiently compresses error-tolerant data with additional requirements for metadata annotations including CC and BPC. These metadata need to be confirmed and annotated manually, which leads to heavy user burdens. In this Section, we propose an adaptive compression scheme by leveraging the image characteristics to avoid the annotation requirements for CC and BPC.

SimCom conducts multiple compression modes in parallel and selects a compression mode to achieve efficient a tradeoff between image quality and memory performance. We design six compression modes with different CCs and BPCs for general image bitmaps. CC has three possible values: 1 for grayscale images, 3 for color images in RGB color space, and 4 for color images with an optional alpha channel storing transparency information. The CC configurations are also applicable to other color space, e.g., CC = 3 is used for the YUV color space of videos. In the meantime, BPC has two optional values: 8 for common images and 16 for high-definition images. Images with other BPCs are supposed to be downscaled in color space to fit predefined compression modes. Such scaling in error-tolerant color images is acceptable, since the common RGB color images (BPC = 8) represent more than 16 million colors while the number of colors discriminated by the human eye is up to 10 million [12].

With multiple compression modes, how to select a compression mode with limited accuracy loss and small compressed data size becomes a new challenge. A straightforward approach is to sample some write accesses for an efficient compression mode to be applied on later write accesses. However, sampling doesn't work for random access patterns generated by applications using different bitmap formats when they are running in the same system. During our experiments, we observe that the compression mode matching the bitmap format often has small normalized difference between adjacent words due to the pixel-level similarity. Hence, we propose to select the compression mode with the smallest average normalized difference between words and corresponding base words, called *mean difference*. The mean difference is calculated following Equation 2, in which $w[i]$ denotes the i-th word in a data block, n denotes the number of words including the possible partial word, and $w[base(i)]$ denotes the base word for $w[i]$. Noting that the calculation of mean difference reuses the intermediate normalized difference generated during the compression progress. Therefore, the calculation of mean difference only consists of one summation and one division.

$$meanDiff = \frac{1}{n}\sum\nolimits_{i=0}^{n-1} normDiff(w[i], w[base(i)]) \quad (2)$$

The overview of our adaptive compression scheme is shown in Figure 5. The *Mode Selector* first selects the compression mode with the smallest mean difference. Hence, our SimCom tends to select the compression mode that matches current bitmap format, thus leading to limited accuracy loss. If multiple modes have the smallest mean difference, the one with minimal compressed data size is selected for high compression performance. After selecting a compression mode, the mode index is encoded into the first 3 bits of the compressed data. Figure 4(e) shows the compressed data encoded with the index of 3C1B mode (CC = 3, BPC = 8).

There are two classes of metadata in SimCom. The first-class metadata include the number of base words and the choice of compression mode. We encoded the two variables into 1 byte for all compression modes except for 1C1B. Specifically, for 1C1B, the first 3 bits of the first byte store the compression mode index and the second byte stores the number of base words. For other five compression modes, the mode index is encoded like 1C1B. However, unlike 1C1B, the number of base words is encoded in the rest 5 bits of the first byte. The reason is that, for these five compression modes, the pixel size is at least 2 bytes and the number of base words for a compressed data block is less than 32 (data block size is 64 bytes), which is possible to be encoded using 5 bits. The second-class metadata is one bit to indicate whether a data block is compressed or not. The bit is also used in existing data compression schemes [5], [6] and stored in a separate region in NVMs.

D. Software Interface

SimCom adopts the following software interface to transfer user annotated approximable data regions ([s_addr, e_addr]) and error thresholds of normalized difference (TH) to NVM module controllers via memory-mapped registers [4]. A practical way to determine the threshold is to search a suitable value using small canary inputs to be applied on full size inputs [13], [14].

```
setApproxRegion(s_addr, e_addr, TH)
```

TABLE I
SYSTEM CONFIGURATIONS.

Processor	
CPU	1 core, x86-64 processor, 2 GHz
L1 I/D cache	32 KB, 2 ways, LRU
L2 cache	1024 KB, 8 ways, LRU
Cache block size	64 B
Main Memory using PCM	
Memory controller	FCFRFS
Read/Write latency	120 ns/150 ns
Memory organization	4 GB, 8 B write unit size

IV. Performance Evaluation

A. Experimental Setup

We implement SimCom in GEM5 [15] with NVMain [11]. The system configurations are listed in Table I. The memory read/write latency comes from the default PCM configurations provided in NVMain. Since GEM5 is too slow to run video-based workloads, we leverage zsim [16], a fast pin-based x86-64 simulator, to measure the performance for videos. We evaluate the performance with six image-based workloads (jpeg, sobel, and kmeans from AxBench [17] and 2dconv, debayer, and histeq from PERFECT [18]) and one video-based workload (x264 from PARSEC [19]). These workloads are selected for various domains (jpeg, sobel, 2dconv, debayer, and histeq for image processing, kmeans for machine learning, and x264 for video processing) and color spaces (x264 for YUV and others for RGB). As suggested in [17], we use root-mean-square error (RMSE) to measure the output errors of images. The input images come from Kodak dataset [20] and we report the average RMSE of 6 images. Structural similarity (SSIM) [21] is used to quantify the output errors of videos by 1-$SSIM$. The output error constraints are set to 3% following BiScaling [4] and 5% for aggressive approximation.

We have evaluated the following compression schemes (FNW [7] is used to further reduce bit flips in all schemes):

- **FPC [6]**: We enhance this scheme by adding approximation. Specifically, if a word is able to match a data pattern by flipping few bits, this pattern is used to compress the word.
- **BDI [5]**: This scheme is enhanced by relaxing the narrow value constraints.
- **BiScaling [4]**: This scheme uses bidirectional precision scaling to approximately compress the data to be written.
- **ApproxCom**: This is our proposed scheme that requires additional annotations for BPC and CC.
- **SimCom**: This is our proposed adaptive compression scheme, which eliminates the annotations on data formats used in BiScaling and ApproxCom.

Since BiScaling, ApproxCom, and SimCom focus on approximate compression on approximable data, we use precise FPC to compress precise data in these schemes. For fairness, we tune the approximation degrees in different schemes (e.g., the number of truncated bits for BiScaling, normalized difference thresholds for ApproxCom/SimCom) to achieve same output error constraints and compare the memory performance.

B. Bit-write Reduction

Figure 6 shows the bit-write ratios using different schemes. The bit-write ratio denotes the percentage of bit flips after data compression and FNW. A lower bit-write ratio implies a higher NVM performance improvement. Due to the efficiency of pixel-level similarity, our SimCom achieves 32.3%/37.8%/30.7% lower bit-write ratios compared with FPC/BDI/BiScaling when the output error constraint is 3%. When the output error constraint is relaxed to 5%, the benefits of SimCom increase. An interesting point is that SimCom obtains slightly lower bit-write ratio than ApproxCom. The reason is that the adaptive compression scheme in SimCom optimizes the compression of grayscale images which are stored in color image formats. For example, if a grayscale pixel is stored in the RGB color space, the bits for each channel are the same. When the mean differences generated by 1C1B and 3C1B are identical (e.g., 0), 1C1B obtains smaller compressed data size than 3C1B. SimCom prefers the modes with small compressed data sizes (e.g., 1C1B), thus leading to more bit-write reduction than ApproxCom (e.g., 3C1B).

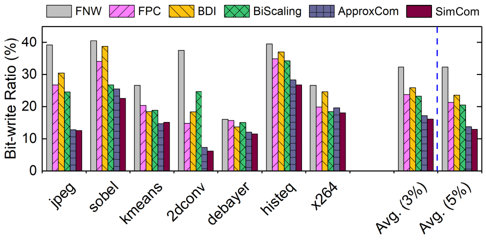

Fig. 6. Bit-write ratio in NVM writes. *(The number after "Avg." denotes the output error constraint.)*

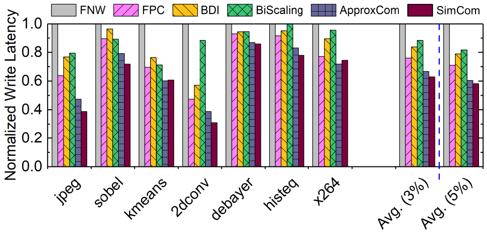

Fig. 7. Write latency normalized to FNW.

C. Write Latency

Figure 7 shows the normalized write latency. Though BiScaling truncates few bits for each word, the benefits in write latency are limited due to the large compressed data sizes. SimCom avoids the writes for similar words, thus efficiently decreasing the data sizes and reducing the time to complete write accesses. As a result, the average write latency of SimCom is 17.3%/24.9%/28.8% (18.0%/26.2%/28.8%) shorter than FPC/BDI/BiScaling when the output error constraint is 3% (5%).

D. Energy Consumption

The energy consumption of NVMs is shown in Figure 8. x264 workload is not measured as zsim doesn't support energy modeling. Since the power consumption mainly comes from the data programming during write operations [7], the low bit-write ratio of SimCom leads to more energy savings than other schemes. In summary, compared with FPC/BDI/BiScaling, SimCom reduces the consumed energy by 18.3%/22.2%/21.1% and 21.4%/25.6%/23.3% within 3% and 5% output error constraints, respectively.

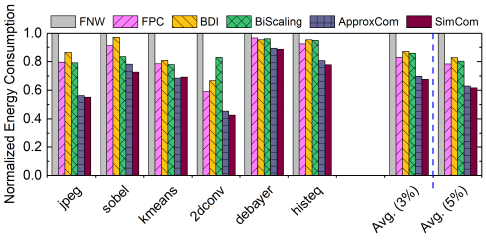

Fig. 8. Energy consumption normalized to FNW.

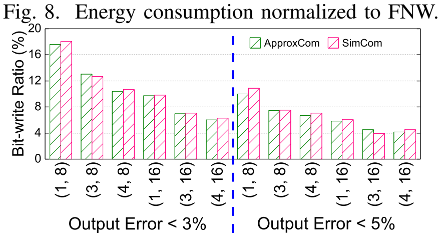

Fig. 9. The bit-write ratio in jpeg with bitmaps of different formats. *(The two integers in bitmap formats denote CC and BPC, respectively.)*

E. Sensitivity to Data Formats

SimCom leverages an adaptive compression scheme without the needs of user annotations on data formats. In order to verify the adaptability of SimCom, we feed the jpeg workload with images of different formats. As shown in Figure 9, SimCom achieves comparable bit-write ratios to those of ApproxCom (within 1%). We further record the selection of compression modes in SimCom (output error $<$ 3%). Table II shows SimCom is able to select the compression mode matching the data formats in most cases (the numbers in boldface). However, when the data have 3 channels, SimCom may select the mode in which CC is 1. The reason is that the jpeg workload converts a color image into grayscale and the channels for a pixel have the same value. Hence, the compression mode with 1 channel often compresses the data with small mean difference due to the fine granularity. Moreover, when the similarity among words is high, using 1 channel is able to obtain smaller compressed data size than 3 channels. As a result, due to the adaptive compression scheme, SimCom achieves slightly higher performance than ApproxCom in some workloads (e.g., jpeg and sobel).

V. Conclusion

The writes in NVM lead to high energy consumption and latency for storage systems. SimCom leverages the pixel-level similarity in bitmaps to efficiently reduce the writes of similar words for NVM-based main memory, thus improving the memory performance in terms of energy efficiency and write latency. By exploiting adaptive approximate compression, SimCom mitigates the programmer annotations used for compression. Experiments show that compared with state-of-the-art FPC and BDI, SimCom decreases 18.3%, 22.2% energy and 17.3%, 24.9% write latency with slight quality loss of 3%.

Acknowledgments

This work was supported by National Key Research and Development Program of China under Grant 2016YFB1000202, and National Natural Science Foundation of China (NSFC) under Grant No. 61772212 and No. 61821003.

TABLE II
The ratios of compression modes in SimCom. *(A bitmap format of "(m, n)" indicates CC = m and BPC = n.)*

Mode Ratio (%)	Bitmap Formats					
	(1, 8)	(3, 8)	(4, 8)	(1, 16)	(3, 16)	(4, 16)
1C1B	**82.4**	47.2	0.6	0.2	0.2	0.2
3C1B	0.2	**34.1**	0	0	0	0
4C1B	0.1	0.4	**96.9**	0	0	0
1C2B	15.3	7.4	0	**98.5**	58.3	1.0
3C2B	0	7.1	0	0.2	**38.5**	0
4C2B	0.1	0.1	2.2	0.6	1.6	**97.9**
Incompressible	1.9	3.7	0.3	0.5	1.4	0.9

References

[1] H. Zhao, L. Xue, P. Chi, and J. Zhao. 2017. Approximate Image Storage with Multi-level Cell STT-MRAM Main Memory. In *Proc. ICCAD*.

[2] J. Xu, D. Feng, Y. Hua, W. Tong, J. Liu, C. Li, G. Xu, and Y. Chen. 2019. Adaptive Granularity Encoding for Energy-efficient Non-Volatile Main Memory. In *Proc. DAC*.

[3] Y. Guo, Y. Hua, and P. Zuo. 2018. DFPC: A Dynamic Frequent Pattern Compression Scheme in NVM-based Main Memory. In *Proc. DATE*.

[4] A. Ranjan, A. Raha, V. Raghunathan, and A. Raghunathan. 2017. Approximate Memory Compression for Energy-efficiency. In *Proc. ISLPED*.

[5] G. Pekhimenko, V. Seshadri, O. Mutlu, P. B. Gibbons, M. A. Kozuch, and T. C. Mowry. 2012. Base-Delta-Immediate Compression: Practical Data Compression for On-Chip Caches. In *Proc. PACT*.

[6] D. B. Dgien, P. M. Palangappa, N. A. Hunter, J. Li, and K. Mohanram. 2014. Compression Architecture for Bit-write Reduction in Non-volatile Memory Technologies. In *Proc. NANOARCH*.

[7] S. Cho and H. Lee. 2009. Flip-N-Write: A Simple Deterministic Technique to Improve PRAM Write Performance, Energy and Endurance. In *Proc. MICRO*.

[8] A. Sampson, J. Nelson, K. Strauss, and L. Ceze. 2013. Approximate Storage in Solid-State Memories. In *Proc. MICRO*.

[9] Q. Guo, K. Strauss, L. Ceze, and H. S. Malvar. 2016. High-Density Image Storage Using Approximate Memory Cells. In *Proc. ASPLOS*.

[10] D. Jevdjic, K. Strauss, L. Ceze, and H. S. Malvar. 2017. Approximate Storage of Compressed and Encrypted Videos. In *Proc. ASPLOS*.

[11] M. Poremba, T. Zhang, and Y. Xie. 2015. NVMain 2.0: A User-Friendly Memory Simulator to Model (Non-)Volatile Memory Systems. *CAL*, vol. 14, no. 2, pp. 140-143.

[12] D. B. Judd. 1975. Color in Business, Science and Industry. Wiley-Interscience.

[13] M. A. Laurenzano, P. Hill, M. Samadi, S. A. Mahlke, J. Mars, and L. Tang. 2016. Input Responsiveness: Using Canary Inputs to Dynamically Steer Approximation. In *Proc. PLDI*.

[14] R. Xu, J. Koo, R. Kumar, P. Bai, S. Mitra, S. Misailovic and S. Bagchi. 2018. VideoChef: Efficient Approximation for Streaming Video Processing Pipelines. In *Proc. ATC*.

[15] N. L. Binkert, B. M. Beckmann, G. Black, S. K. Reinhardt, A. G. Saidi, A. Basu, J. Hestness, D. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. S. B. Altaf, N. Vaish, M. D. Hill, and D. A. Wood. 2011. The gem5 Simulator. *SIGARCH Computer Architecture News*, vol. 39, no. 2, pp. 1-7.

[16] D. Sánchez and C. Kozyrakis. 2013. ZSim: Fast and Accurate Microarchitectural Simulation of Thousand-Core Systems. In *Proc. ISCA*.

[17] A. Yazdanbakhsh, D. Mahajan, H. Esmaeilzadeh, and P. Lotfi-Kamran. 2017. AxBench: A Multiplatform Benchmark Suite for Approximate Computing. *IEEE Design & Test*, vol. 34, no. 2, pp. 60-68.

[18] K. Barker, T. Benson, D. Campbell, D. Ediger, R. Gioiosa, A. Hoisie, D. Kerbyson, J. Manzano, A. Marquez, L. Song, Nathan R. Tallent and A. Tumeo. 2013. PERFECT (Power Efficiency Revolution For Embedded Computing Technologies) Benchmark Suite Manual. *Pacific Northwest National Laboratory and Georgia Tech Research Institute*.

[19] C. Bienia, S. Kumar, J. P. Singh, and K. Li. 2008. The PARSEC Benchmark Suite: Characterization and Architectural Implications. Tech. Rep. TR811-08, Princeton University.

[20] R. Franzen. Kodak Lossless True Color Image Suite. http://r0k.us/graphics/kodak/.

[21] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE Trans. Image Processing*, vol. 13, no. 4, pp. 600-612.