ASTES
Advances in Science, Technology and Engineering Systems Journal Vol. 5, No. 2, 561-566 (2020)
www.astesj.com
ASTESJ
ISSN: 2415-6698

Minimizing the Cleaning Cost in Flash Based Storage Sub-Systems by Proliferating the Valid Data Copying Process

Hisyamuddin Kasim, Amir Rizaan Abdul Rahiman[*], Nor Asilah Wati Abdul Hamid, Thinagaran Perumal

Department of Computer Science, Universiti Putra Malaysia

A R T I C L E I N F O

Article history:
Received: 27 November, 2019
Accepted: 22 February, 2020
Online: 08 April, 2020

Keywords:
Sector
Flash memory
Garbage collection
Storage

A B S T R A C T

Flash memory reliability and the performance have been confirmed as an improvement in the storage subsystem technological advancement, especially in supporting an embedded system solution. Conversely, the main constraint of the storage device is the erase-before-write characteristic in handling both write and re-write I/O operations. More attention must be taken into consideration to handle both I/O operations. To overwhelm this constraint, a time-consuming garbage collection scheme has been introduced. This paper presents an optimized cleaning scheme that significantly reduces the required cleaning collection cost without interfering the memory I/O operations. The candidate sector to be erased is being determined by a score factor together with an erasure count before selecting the actual victim sector to be cleaned. The experimental results show the proposed approach assists in reducing the garbage collection cost since the number of sectors used in handling the I/O operations is being minimized. Even the sector becomes the candidates, but the score factor will be the tiebreaker to determine whether the sector needs to be cleaned or not. On the contrary, the proposed scheme increases the number of copy operations due to new sector requirement while the degree of the wear-levelling emulate the standing sector cleaning scheme.

1. Introduction

Recently, flash memory has become the predominant semiconductor storage device that offers some outstanding features such as zero noise, solid-state reliability, low power consumption, small and lightweight natures, and great shock-resistant [1 – 4]. Its primary application is a storage device used for storing and transferring data in various electronic appliances such as smartphones, digital cameras, a global positioning system (GPS) device, just to name a few.

On the contrary, the flash memory has two operating characteristics namely, i) out-place-updating policy, and ii) high cleaning cost requirement. Those characteristics bring some technical challenges in designing and implementing an efficient storage sub-system, especially in an embedded system. Rewrite an existing data at the recent location in the device results in I/O operational overheads. Thus, the out-place-updating policy has been employed where the updated data is being stored into a new location while the initial copy of the data is set as garbage [5 – 6]. Conversely, the frequent data rewrite increases the amount of garbage which resulting in lesser device-free spaces. The garbage needs to be cleaned to guarantee the continuous data storing process. Moreover, the device could wear out and leads to its end lifetime as compared to the classical hard drive. The cleaning in the flash memory is being implemented on a sector unit rather than page unit where it may contain valid data that is used by particular applications. Thus, the entire valid data in the sector need to be transferred or copied into the available free pages in the available sectors. The cleaning process and the sector utilization level, the ratio between the number of valid data and the total pages per sector have been considered the stimulus of flash memory I/O access performance, energy consumption and device persistence. Several studies those focusing on recognizing the accurate victim sector selection to be erased have been proposed in the literature. The objectives of the studies are to minimize the cleaning cost and lengthen the storage device lifespan [7 – 13]. The cleaning cost refers to the required I/O accessing time in executing the cleaning operation.

This paper proposes a sector cleaning scheme, called candidate selection garbage collection (CSGC). Unlike the existing schemes, the candidate or the victim sector in the CSGC scheme is being determined according to the score factor together with the erasure count before the selection of the actual sector to be cleaned. The scoring factor helps in selecting the actual victim sectors which result in the least cleaning cost. According to the simulation results

[*] Amir Rizaan Abdul Rahiman, Email: amir_r@upm.edu.my

performed, the CSGC scheme offers lower erasure operation among the existing sector cleaning and emulate the similar degree of wear-levelling with the existing cleaning scheme. Conversely, the scheme requires higher copy operation as compared to the previous schemes since it repopulated the valid data more rapidly after the cleaning process had been initiated. The remainder of this paper is organized as follows. Section 2 discusses the related works on a garbage collection scheme. Section 3 discusses and explain the proposed CSGC scheme. The experimental setup and the performance evaluation are being discussed in Section 4 while Section 5 presents the conclusions of this study.

2. Backgrounds and Related Works

In general, there are two types of flash memory in the current market, i) NOR-flash and ii) NAND-flash [14]. The characteristics comparisons between both flash memory types are being tabulated in Table 1. In this study, we limit our discussion on the NOR-flash and will refer to it as the flash memory in this paper.

2.1. Flash Memory Characteristics

Flash memory is a tiny and lightweight semiconductor storage device and known as a block-and-page-based storage device. The page is a basic accessing unit and a group of pages form a sector unit or block. Both read and write I/O accesses are executed in the page unit while the erase is being executed in the sector unit. Each page has two dissimilar sized areas, data and a spare area. The data area is a location where the actual data being stored while the spare area keeps assisting information about the data area.

Table 1: NAND-flash and NOR-flash Characteristics [15]

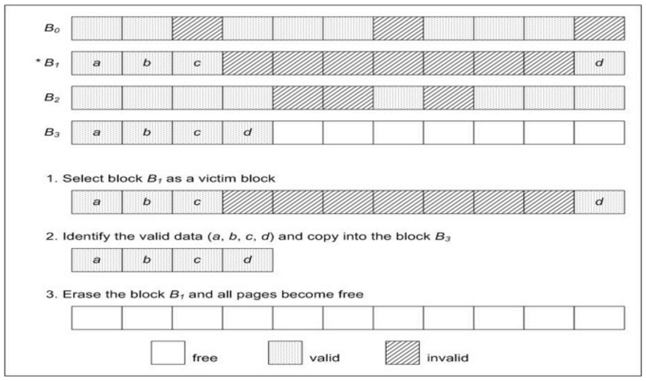
Operations	NAND-flash	NOR-flash
Read unit	Page (512 + 16 bytes)	Byte/word
Access time	7 µs (initial access) 50 ns (serial access)	90 ns (random access)
Write unit	Page (512 + 16 bytes)	Byte/word
Write time	200 µs	8 µs/byte 16 µs/Word (4 ms/528 bytes)
Erase unit	Block (8k + 256 bytes)	Sector (8k/64k bytes)
Erase time	2 ms	1 s

As tabularized in Table 1, the NAND-flash has slower read operation but offer faster write operation. Therefore, it is being used as a substitution for the classic storage sub-system (e.g., magnetic hard disk) and known for its enormous capacity. On the other hand, the NOR-flash memory offers fast read operation which is being carried out in a byte unit. However, it has slower written I/O operation. Moreover, the NOR-flash is known for its XIP capability where it runs the BIOS directly from the memory device. Therefore, the NOR-flash memory is being used as an alternative to the main memory of a personal computer or embedded microcontroller system. Rather than copying a program into a RAM, the XIP capability can execute a program directly from the storage. Thus, reduce the total memory usage as an extension of a portion of memory.

In flash memory, updating the existing data is done via the out-place-policy. Therefore, the page unit falls into three states (*free*, *valid*, and *invalid*). The free page is a page unit without any data and is ready for storing new and updated data while the valid page is a page containing the latest version or actual data. The invalid page is a page containing garbage or dead data. Moreover, each memory sector has a limited number of erasure cycles. Therefore, excessive sector erasures can cause the block to be permanently spoilt or damage. To overcome this limitation, a wear-leveling policy that wears down all memory sectors as evenly as possible is essential in flash memory.

2.2. Sector Cleaning in Flash Memory

Figure 1 below shows the cleaning process in the flash memory is carried out in three underlying phases. First, the victim sector/s is being determined based on the cleaning thresholds. Second, the entire valid data reside in the sector are being recognized and being copied into available free pages located in a spare sector. Third, upon completion of the copying, the victim sector is being erased. Moreover, if all pages in the victim sector are invalid, the sector can automatically be erased without requiring the copying process cost since the cleaning requires only the constant erasure cost.



B_0
$*B_1$
B_2
B_3
a
b
c
d
1. Select block B_1 as a victim block
2. Identify the valid data (a, b, c, d) and copy into the block B_3
3. Erase the block B_1 and all pages become free
free
valid
invalid

Figure 1: Three Phases in The Semi-automatic Cleaning Process in Flash Memory

2.3. Cleaning Algorithms in Flash Memory

There are various studies are focusing on defining the victim sector and re-organizing valid data when the garbage collection is initiated in the flash memory. Most studies are focusing on the NAND-flash type. The two objectives of the studies are minimizing the cleaning cost and guaranteeing the memory sectors are evenly worn. In the NOR-flash environment, the sector cleaning process is handled by freeing up both valid and invalid pages for the next iteration of live pages or bytes to be written in the erased victim sector. The life expectancy of the device is being determined according to the status of sector usage. For example, to utilize the sector life cycle, the proportional of sector erasure count and the re-use level are being taken into consideration. Moreover, the dense and the energy usages of the NOR-flash are less as compared to the NAND-flash. Thus, in NOR-flash, the cleaning process becomes a challengeable task since the cleaning operation is quite slow as compared to other I/O operations. Following are the cleaning schemes those focusing on the NOR-flash type.

The flash resident file system (FLAREFS) scheme dynamically proposed sector cleaning scheme with a wear-leveling policy that keeps a region of sectors clean and available for storing the updated data [16]. To determine the victim sector, the scheme will keep tracking the status of every physical busy page in a meta-table. A such number of sectors is required for the meta-table that makes the scheme consumes extra storage space. Moreover, the scheme requires extra space for handling the valid data copying process when the cleaning process is being initiated. When the cleaning is being initiated, the established greedy technique (sector with the highest amount of garbage) is being used to select the victim sector.

In the least first garbage collection (LFGC) scheme [17], all the dirty sectors (another name for a victim block) are being clustered according to the candidate region and resident region. Unlike the FLAREFS, the LFGC scheme uses hot and cold classification technique in reorganizing the valid data found in the victim sector. The well-defined cold-detection scheme has been used to classified hot and cold valid data. Those sectors belong to the candidate region are being selected to be erased when the cleaning is being initiated. On the contrary, if the candidate region does not have any sector, the least erase count sector in the device will be the victim sector.

Niv and Phillipe have proposed the two garbage collection schemes called Lazy Gecko and Logarithmic Gecko [18]. Both schemes require metadata to perform the sector cleaning with the minimum cleaning cost. In the Lazy Gecko, the RAM-resident bitmap called Page Validity Bitmap (PVB) is being used to store the up-to-date status of the pages to allow the victim sector selection and valid page identification. However, the scheme works efficiently when the PVB is ample to store the copying process information. Note that, the size of the metadata is very important for caching the frequently accessed I/O in the embedded system. Thus, to minimize the metadata size, the Logarithmic Gecko has been proposed to overcome the highly RAM consumption in the Lazy Gecko solution. The scheme similar to Lazy Gecko, but the information in the PVB is being stored in the flash-resident reverse map.

From the above studies, we have observed that the sector cleaning schemes that combine the victim block selection with the valid data reorganizing procedures assists in achieving the sector cleaning process objectives. Most of the schemes use the greedy approach in determining the victim block to be erased. Using the similar approach, this paper presents the score factor consideration together with the sector erasure count to verify the suitable victim sector that requires the minimum cleaning cost. The victim sector will be select as the candidate for the cleaning before the cleaning can be executed. Moreover, these two screening parameters will give the exact victim sector. Therefore, it has given us the motivation to propose an optimized sector cleaning scheme that guarantees the I/O performance in the flash memory.

3. Candidate Selection Garbage Collection (CSGC)

Table 2 lists the data structures used in the phases to determine the appropriate candidate sector. As illustrated in Figure 2, the cleaning process in the CSGC scheme will go through three (3) consecutive phases without exception in regulating the candidate sector selection. In the figure, the sector refers to the available block in the flash memory device.

First, when the cleaning process has been initiated, the scheme will check if all the sectors in the device are either free or not. This stage becomes a preliminary task to verify whether the cleaning process can be executed or not. Then, move to the second stage. In this stage, the availability for executing the copying process is being determined according to the *free_pages* variable. This stage is required to proceed with the candidate selection for the copying process when the candidate block was selected. Note that, the valid data/pages reside in the candidate sector must be copied into the available free sector in the memory. The busy pages within the sector will not be selected if the file where the pages reside is in the current execution state. The reason is, copying and migrating these pages require longer waiting time for the sector to be ready for the cleaning process.

Table 2: Data Structure in The CSGC Candidate Sector Selection

Variables	Description
fs-	The sectors in the flash memory are in free state
p_free	All pages in the sector are empty
free_pages	The available free pages in the sector
pages_per_sector	Total pages in each block of the memory
p_busy	The valid pages in the sector

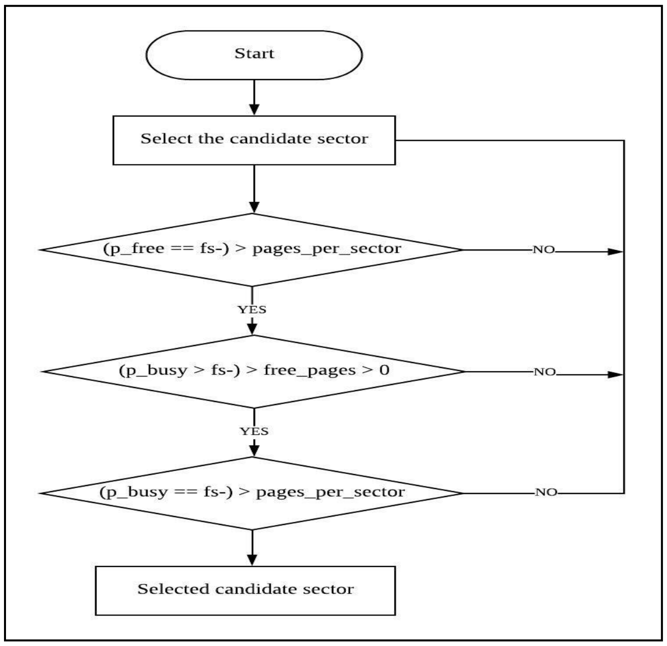

Start
Select the candidate sector
(p_free == fs-) > pages_per_sector
NO
YES
(p_busy > fs-) > free_pages > 0
NO
YES
(p_busy == fs-) > pages_per_sector
NO
Selected candidate sector

Figure 2: Candidate Sector Selection Process in The CSGC Scheme

Furthermore, if the sector has no free pages but has the non-busy pages, then the copying process become the main priority. At this stage, if the sector has the only free pages but in a different sector, this sector will not be chosen as the candidate sector. After completing the third stage, then the candidate sector is being decided. Only the sector that has the largest amount of non-busy pages will select for the cleaning process. This selection also can fully utilize sector usage lifecycle. Unlike the previous cleaning schemes, the CSGC scheme uses a scoring system to select the candidate sector for the erasure. The scoring *sec_ers_diff* parameter is being calculated according to the difference value between the maximum sector erasure count (denoted by *max_ers*) and the sector header erase (denoted by *header_ers*) count as shown below:

$$sec\_ers\_diff = max\_ers - header\_ers \quad (1)$$

Parameter *sec_ers_diff* in (1) defines the score factor for every sector in the device. The sector with the highest score factor will be selected as the candidate sector. However, if the sector has the same highest score, it will choose based on the sector index number. For example, both sector 1 and sector 2 have the same score value, then sector 1 will be chosen as the candidate. Then, the candidate sector cleaning cost (*cost*) is being calculated according to the following equation:

$$cost = (sec\_ers\_diff \times 100) + (free\_pages \times 100) + (invalid\_pages \times 100) + (p\_busy \times 100) \quad (2)$$

To ensure the efficiency of the victim sector selection, the calculation is being measured for every I/O accessing iterations. As shown in Figure 3, each sector will have the erase count, and the score factor. To maintain the consistency of the erased sector to prolong its lifespan, the erasure count of each sector will keep updates and monitoring. The only thing separating each sector is based on highest score factor and the sector starting digit.

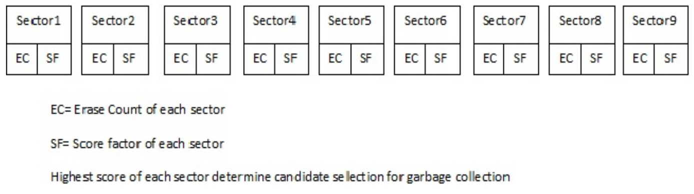

Sector1
Sector2
Sector3
Sector4
Sector5
Sector6
Sector7
Sector8
Sector9
EC
SF
EC= Erase Count of each sector
SF= Score factor of each sector
Highest score of each sector determine candidate selection for garbage collection

Figure 3: The Score Factor Counting in The Sector

4. Simulation and Experiment Results

4.1. Simulation Model and Parameter

The CSGC scheme is evaluated through proper computer simulation. The evaluation performance of the scheme is being compared with the existing sector cleaning schemes in terms of the total block erasure count and the copying process involved. In addition, the degree of wear-levelling is being discovered for the proposed scheme. The simulation model was programmed using the STM32F1 architecture that targeted for the embedded system simulation. Table 3 shows the actual technical specifications of the memory device in the STM32FI architecture. In the simulation, the threshold level of the available free space in the device is being set to 15% for initiating the cleaning process. Table 4 shows the actual device specification being executed on the Linux Kernel 4.1 environment during the simulation.

Table 3: STM32FI Simulation Parameters

Sector Header size	Page size	Page Header size
niffs_sector_hdr	*niffs_page_hdr:data*	*niffs_page_hdr:data*
ers_cnt	*id : flag* | …. \obj_id \spix	*id : flag* | …. \obj_id \spix

Table 4: The Evaluation Parameter Framework

Parameters	**Value**
Number of sectors in device	2^{16}
Sector size	8 Kbytes
Page per sector	512
Page size	16 bytes
Working buffer size	16 bytes

Figure 4 shows how the wireless sensor data is being collected for the I/O accessing pattern that is used in the simulation as the simulation data benchmarking.

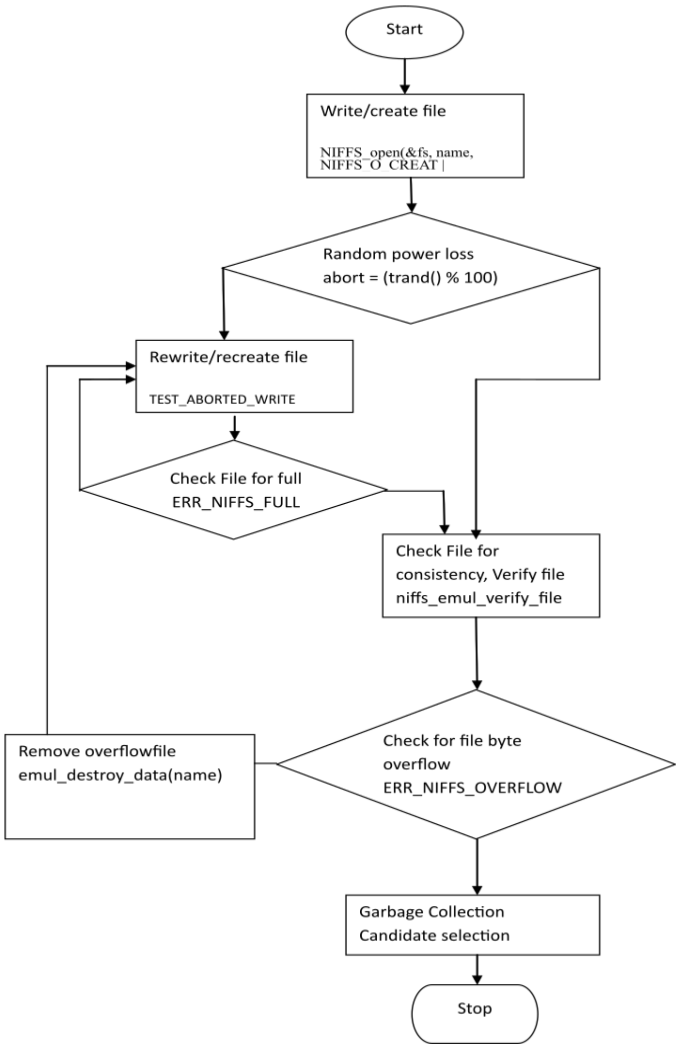


Figure 4: The Random Data Establishment in Simulation

4.2. Results and Discussions

The three performance matrices, i) number of erase operation, ii) number of copying the valid data and iii) degree of wear-levelling are being discovered in the simulation when the cleaning process is being initiated. The matrices are based on the storing and updating the random I/O accessing pattern captured from the designated testing framework. The following figures illustrate the evaluation results for the matrices.

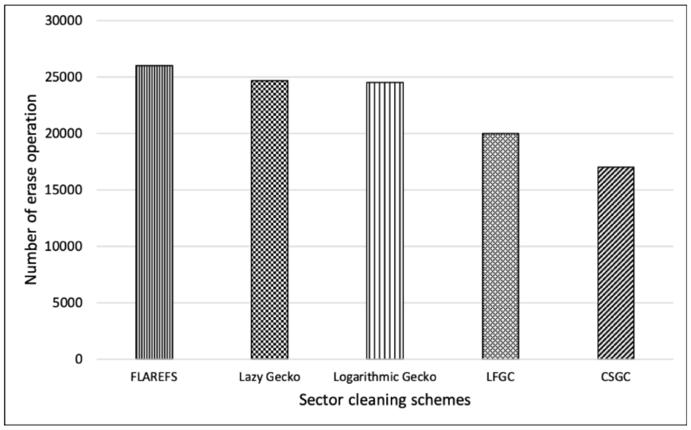


Figure 5: The Sector Erasure Count

As compared to existing garbage collection schemes, the CSGC scheme requires the lowest sector erasure counts. The lowest block erasure count is aligning with the minimum cleaning process cost since the number of involved sectors during the garbage collection process is being reduced. This is due to the candidate sector in the scheme will be evaluated again according to the score factor to confirm whether the sector can be erased or not.

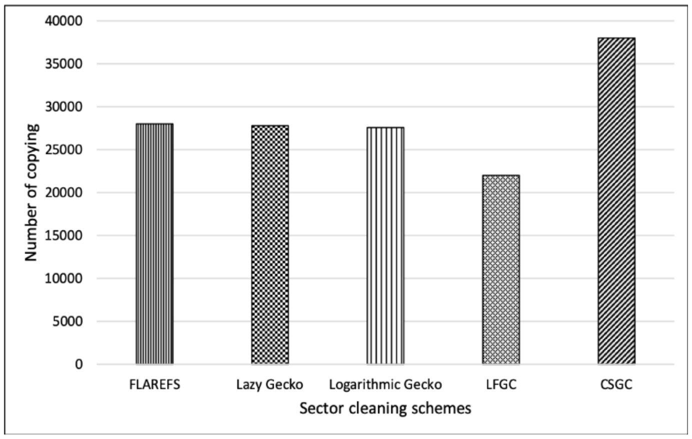


Figure 6: The Valid Data Copying Amount

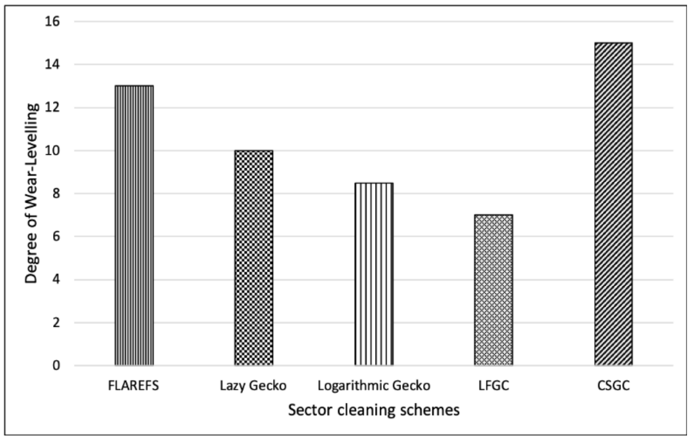


Figure 7: Degree of Wear-Levelling

On the contrary, as shown in Figure 6, the CSGC requires the most copying process. The scoring factor has increased the CSGC scheme to employ the new sector during the valid data copying process. As compared to the existing schemes, the gap difference is a considerable disparity. Note that, in the CSGC scheme, during the candidate sector selection, the existing valid data in the sector might be transferred into a new sector according to the state of the candidate sector. In the simulation, we have counted this copying process and that is the main reasons why the gap difference is quite large as compared to the existing schemes. Moreover, the degree of wear-levelling exceeding the existing sector cleaning schemes.

5. Conclusion and Future Work

The garbage collection scheme is a mandatory process in flash memory due to the necessarily erase-before-write characteristic. Several studies have been carried out to optimize the time-consuming erasure process due to the characteristic. Unlike the existing sector cleaning scheme, the CSGC scheme emphasizes on selecting the candidate sector for the valid data copying prior in selecting the actual victim block to be cleaned. The candidate sector is not necessarily the victim block to be erased. With the sector scoring factor and the erasure count, the actual victim sector with low cleaning cost can be determined appropriately. Moreover, the consistency of the erased sector is a pledge. Even though the number of the cleaning process is reduced significantly, the proposed CSGC scheme requires more copying processes when the garbage collection process is being initiated. Also, the degree of wear-levelling increase significantly. This is due to the busy pages are being copied earlier when the candidate sector has been selected. The scoring factor determines this approach earlier before the sector is being confirmed to be erased. For further study, an efficient calibration measurement will be used in reducing the process of a valid copy of the data once the sector candidate is determined. In the present study, the selection of candidates is based on the score factor derived from the difference between the maximum sector erasure count and the sector header erasure count.

Conflict of Interest

The authors declare no conflict of interest.

Acknowledgment

The work is supported by the Geran Putra (GP) GP/2018/9596300 and we would also like to thank Universiti Putra Malaysia for providing technical assistance.

References

[1] R. Agarwal, M. Marrow, "A Closed-form Expression for Write Amplification in NAND Flash" in 2010 IEEE Globecom Workshops, Miami, FL, 2010. https://doi.org/10.1109/GLOCOMW.2010.5700261

[2] Y. Zhang, H. Zhang, C. Wang, "Reliability-aware low energy scheduling in real time systems with shared resources" Journal Micro. and Microst., 52(C), 312 – 324, 2017. https://doi.org/10.1016/j.micpro.2017.06.020

[3] Y. Zhou, F. Wu, P. Huang, X. He, C. Xie, J. Zhou, "An Efficient Page-level FTL to Optimize Address Translation in Flash Memory" in Proc. ACM the 10th European Conference on Computer Systems, Bordeaux, France, 2015. https://doi.org/10.1145/2741948.2741949

[4] F. Chen, B. Hou, R. Lee, "Internal parallelism of flash memory-based solid-state drives" ACM Trans. on Storage, 12(3), 1 – 39, 2016. https://doi.org/10.1145/2818376

[5] S. K. Panigrahi , C. Maity, A. Gupta, "A simple wear leveling algorithm for NOR type solid storage device" CSI Trans. on ICT, 2(1), 65 – 76. 2014. https://link.springer.com/article/10.1007/s40012-014-0047-3

[6] S. y. Park, E. Seo, J.y. Shin, S. Maeng, J. Lee, "Exploiting internal parallelism of flash-based SSDs" IEEE Comp. Arch. Letters, 9(1), 9 –12. 2010. https://doi.org/10.1109/L-CA.2010.3

[7] P. Olivier, J. Boukhobza, E. Senn, “Micro-benchmarking Flash Memory File-system Wear Leveling and Garbage Collection: A Focus on Initial State impact” in Proc. 15th International Conference on In Computational Science and Engineering (CSE), Nicosia, Cyprus, 2012. https://doi.org/10.1109/ICCSE.2012.67

[8] M.C. Yang, Y.M. Chang, C.W. Tsao, P.C. Huang, Y.H. Chang, T.W. Kuo, “Garbage Collection and Wear Leveling for Flash Memory: Past and Future” in Proc. International Conference on Smart Computing (SMARTCOMP), Hong Kong, China, 2014. https://doi.org/10.1109/SMARTCOMP.2014.7043841

[9] O. Kwon, K. Koh, J. Lee, H. Bahn, “FeGC: An efficient garbage collection scheme for flash memory based storage systems. Journal of Syst. and Soft., 84(9), 1507 – 1523, 2011. https://doi.org/10.1016/j.jss.2011.02.042

[10] S. Mittal, J.S. Vetter, “A survey of software techniques for using non-volatile memories for storage and main memory system” IEEE Trans. on Parallel and Dist. Sys. 27(5), 1537 – 1550, 2016. https://doi.org/10.1109/TPDS.2015.2442980

[11] J. Y. Paik, R. Jin, T.S. Chung, “Data recovery aware garbage collection mechanism in flash-based storage devices” IEICE Trans. on Info. and Sys., E101.D(9), 2404 – 2408, 2018. https://doi.org/10.1587/transinf.2017EDL8255

[12] C. C. Chung, D. Sheng, N.M. Hsueh, “A high-performance wear-leveling algorithm for flash memory system” IEICE Electronics Express, 9(24), 1874 – 1880, 2012. https://doi.org/10.1109/APCCAS.2012.6419105

[13] Y.P Hu, N. Xiao, X.F. Liu, “An elastic error correction code technique for NAND flash-based consumer electronic devices” *IEEE Transactions on Consumer Electronics*, vol. 59, no. 1, pp. 1-8, 2013.

[14] M. Lin, S. Chen, “Efficient and intelligent garbage collection policy for NAND flash-based consumer electronics” IEEE Trans. on Cons. Electronics, 59(3), 538 – 543, 2013. https://doi.org/10.1109/TCE.2013.6626235

[15] A. R. Rahiman, P. Sumari, “Minimizing the Garbage Collection Time in Flash Memory Using Efficient Data Allocation Scheme” in Proc. 2009 IEEE Region 10 Conference (TENCON2009), Singapore, 2009. https://doi.org/10.1109/TENCON.2009.5396010

[16] S. Fazackerley, R. Lawrence, “A Flash Resident File System for Embedded Sensor Networks” in Proc. IEEE 24th Canadian Conference on Electrical and Computer Engineering (CCECE), Niagara Falls, ON, Canada, 2011. https://doi.org/10.1109/CCECE.2011.6030693

[17] G. Xu, Y. Liu, X. Zhang, M. Lin, “Garbage collection policy to improve durability for flash memory” IEEE Trans. on Cons. Elect., 58(4), 1232 – 1236, 2012. https://doi.org/10.1109/TCE.2012.6414990

[18] N. Dayan, P. Bonnet, “Garbage collection techniques for flash-resident page-mapping FTLs". *arXiv preprint* arXiv:1504.01666