International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-9 Issue-12, October 2020

Garbage Collection Algorithms in Flash-Based Solid State Drives

Rishabh Gogna

Abstract*: Solid state drives (SSDs)have emerged as faster and more reliable data storages over the last few years. Their intrinsic characteristics prove them to be more efficient as compared to other traditional storage media such as the Hard Disk Drives (HDDs). Issues such as write amplification, however, degrade the performance and lifespan of an SSD. This issue is in turn handled by the Garbage Collection (GC) algorithms that are put in place to supply free blocks for serving the writes being made to the flash-based SSDs and thus reduce the need of extra unnecessary writes. The LRU/FIFO, Greedy, Windowed Greedy and D choices algorithms have been described to lower write amplification for incoming writes which are different in nature. The performance of the GC algorithms varies based on factors such as pre-defined hot/cold data separation, hotness of data, uniform/non-uniform nature of incoming writes, the GC window size and the number of pages in each block of the flash memory package. Finally, it can be seen that the number of write frontiers so used, can dictate the separation of hot/cold data and increase the performance of a GC algorithm.

Keywords*: Flash Memory, Garbage Collection, Solid State Drive, Write Amplification.

I. INTRODUCTION

The past two decades have seen a gradual increase in the interest towards the improvement of storage media. The HDDs have and continue to serve as one of the most widely used storage device. However, HDDs have steadily been over shadowed due to the arrival of Solid State Drives (SSDs) that now work on flash memory. Solid State Drives are used to store data persistently in semiconductor cells that typically contain floating gate MOSFETS. SSDs provide several advantages over the traditional HDDs. HDDS store data in spinning magnetic disks. Read and Write heads are used to read and write data on specified locations on the disk. As a result of physical components being present within an HDD, Hard Disk Drives show very low resistance to shock, high latency due to the presence of a rotating magnetic disk and comparatively higher power consumption. SSDs however prove to be shock resistant (due to absence of physical moving parts), consume lesser power, are compact in size, and most importantly, are non-volatile in nature i.e. they retain data even in the absence of power. Solid State Drives are typically designed using the NAND flash memory [3]. The NAND flash memory is of two types depending on the number of bits one flash cell can store, SLC (Single Level Cell) in which one cell can store only 1 byte of data and MLC (Multi Level Cell) which can store two or more bits of data. SLCs are faster and more reliable than MLCs due to a longer lifespan and hence are more expensive and used in commercial or industrial applications as compared to MLCs. SSDs use packages of flash memory that are connected to a controller using numerous channels. The controller is then connected to the host via interfaces such as Serial Advanced Technology Attachment (SATA). Flash memory packages are divided into thousands of blocks each of which contain several pages (around 64 to 128). Each page contains a 2KB to 4KB data store and some storage for metadata [3]. Three major operations define the flash memory based SSDs; read, write and erase. This is facilitated by the Flash Translation Layer (FTL) algorithms implemented in the controller which works on the basis of Logical to Physical Mapping using the Logical Block Addresses (LBA) and Physical Block Addresses (PBA) that are used by the SSDs to serve the requests from the hosts and make changes to the data within the memory accordingly [8]. Flash memory based SSDs are as such that they do not support in-place writes. Rather, such devices make use of the out-of-place writes, i.e. rather than overwriting prewritten and already existing data, they write data in other free locations on the physical addresses in the memory. If no free space is available, data is first erased from a given location and the then empty location is updated with new data. SSDs make use of a Log Structured File System (LFS) [10] that pertain to increase write performance by allowing higher bandwidth to be utilized for write operations. Data is read from/written into flash memory with pages as the basic unit whereas erasure is done at block level. Typically, read operations take 25µs (SLC) to 60µs (MLC), write operations take 250µs (SLC) to 900µs (MLC) while erase operations take the longest, around 1.5ms (SLC) to 3.5ms (MLC). Erasing of data is carried out at a block level due to the consumption of large amounts of voltage by the floating gate MOSFETS [7]. Each block of flash memory can only be erased a given number of times which is generally around 100,000 for the more expensive SLC and 10,000 for MLC [3]. An issue that threatens the lifespan of an SSD, thus resides in the writes that are made by the drive into the memory blocks. At any given point of time, a page is either in the valid, invalid or erase state. Erased pages are validated when data is written to those pages while invalid pages hold data that is no longer required. Writing data into the memory is only possible for pages that are in the erase state [1]. Therefore when data is to be written to pages in the invalid or valid state, the pages containing valid data must first be copied to another block (to RAM or another memory buffer), the original block data deleted, and the valid pages are then copied back to the empty block. The pages block therefore contains pages that are now in the erase state and hence writes can be made to these pages.

Revised Manuscript Received on October 30, 2020.
* Correspondence Author
Rishabh Gogna*, Department of Electronics and Communication Engineering, Amity University Mumbai, (M.H), India

This method however leads to reduction in the lifespan and performance of the SSD, as flash memory cells can only be erased a given number of times before becoming inefficient. Sequential writes therefore help in maintaining the write performance of an SSD. In the absence of free pages for direct writing, the above procedure of copying valid pages to memory, erasing the block, and copying the data back leads to the creation or addition of extra unnecessary data within the flash memory.

As a result, at times, for writing data that may originally be around 4KB, 40KB of total data may be written (in the form of original data + copied data + metadata) in the memory due to all the required copying and erasure. This leads to an issue called Write Amplification which severely affects the performance and lifespan of the SSD. During the PE (program erase) cycles carried out by the SSDs, it sometimes may happen that some blocks may be written into and erased more frequently than others. This leads to some blocks wearing out faster than others in the flash memory that causes the issue of Wear Levelling which again affects the performance and lifespan of an SSD as, if even one block wears out in the flash memory packages, the whole SSD is affected severely. To counter these issues SSDs use Garbage Collection and Wear Levelling algorithms.

II. GARBAGE COLLECTION

As seen earlier, for data to be written, it is necessary for pages to be in the erased state. For this, blocks must be erased regularly to obtain free erased pages for incoming data to be written directly into and hence manage the data traffic efficiently [1]. This job is carried out by the Garbage Collection algorithm employed within an SSD. The Garbage Collection algorithm (GC) regularly victimizes a block, copies its valid pages to memory, erases the block and copies the valid pages along with the data to be written through write requests back to the block. This leads to the formation of a write frontier (WF). The role of a GC algorithm can therefore be summarized to work in order to maintain a write frontier at all times so as to allow subsequent write operations to make use of the write frontier till the GC algorithm has to be invoked again for the creation of a new write frontier once the previous one is filled. Once a write frontier is filled it is sent to a storage pool and another block is selected from the free list which then becomes the new write frontier. When the number of usable blocks drops below a certain threshold, certain number of filled blocks are erased till the number of blocks exceeds the minimum threshold for the free list. Different GC algorithms react differently to the type of writes made into the memory. Sequential writes are often easier to handle and are less degrading for the lifespan of the SSD whereas random writes prove to wear down the SSD more [5,6,9]. Write amplification therefore serves as a very important metric to dictate the efficiency of the GC algorithm used in an SSD. Write amplification is defined as the ratio of the number of writes actually made to the number of writes requested by the host [1,5,6,13]. Following are some of the GC algorithms that have been studied and summarized in the following survey: 1) Greedy Cleaning Algorithm: Is a form of GC algorithm that victimizes blocks containing the least number of valid pages/ greatest number of invalid pages [1,5,6,13,11,14]. 2) FIFO/LRU Cleaning Algorithm: Is a form of GV algorithm that victimizes blocks in a sequential manner such that the block that enters the free list at first is the block that is chosen for Garbage Collection [6,13]. 3) Windowed Greedy Cleaning Algorithm: Is a form of GC algorithm that considers blocks containing the least number of valid pages within a fixed window of blocks in the list of blocks to be cleaned [5,6]. 4) D choices Cleaning algorithm: Is a form of GC algorithm that selects blocks with the greatest number of invalid pages from a set of d blocks chosen at random [1,11].

A. Greedy Cleaning Algorithm

For most GC algorithms, many parameters are predefined before the analytic or probabilistic study of the algorithm. Over provisioning factor and spare factor are two such factors. Over provisioning is done in SSDs to allot extra memory for the SSD to perform its operations. As a result, the logical memory space available to the host is kept less than the physical memory available within the SSD. This helps in reduction of write amplification. The extent of over provisioning is defined by the spare factor which is defined as: SPARE FACTOR= (physical capacity – user visible capacity)/ physical capacity Or by the overprovision factor which is the ratio of total physical storage capacity to the user visible storage capacity. GC performance depends also on the type of mapping done by the FTL. For the sake of generality, Peter D. [6], assumes a fully page mapped model for the analysis of the GC algorithms so proposed omitting the issues of wear levelling as well as the cost or read and erase operations (although erase operations prove to be costly in SSDs). As mentioned earlier, the greedy cleaning algorithm victimizes blocks containing the greatest number of invalid pages. This method proves to be extremely effective in reducing the write amplification of an SSDs in case of uniform random writes as depicted by Peter D. [6], Hu [5]. Greedy cleaning focuses on the arrival of valid pages to a certain state. A minimal state is set below which the number of valid pages is not put under consideration by the algorithm. As soon as the threshold state is crossed, the block is victimized for reclamation by the GC algorithm and it is brought into a small pool of blocks which contain other blocks that are eligible for greedy cleaning algorithm and when a block is selected for garbage collection, all pages within the block have an equal probability of getting invalidated [13]. Experiment results from [6] show that LRU cleaning is more efficient than greedy cleaning in terms of the number of pages within the block i.e. as the number of pages in a block increases the efficiency of the greedy cleaning algorithm decreases. The graphical results from the same, show that for 100000 blocks containing 64 pages each, write amplification decreases effectively as the spare factor increases and when compared to blocks containing only 16 pages, the write amplification is reduced even more for smaller values of spare factor. Proofs by contradiction also show that greedy cleaning is an optimal algorithm for systems with memory less workloads [11]. While the above studies are able to demonstrate the efficiency of greedy cleaning for uniform traffic, real world traffic is typically non uniform in nature. As a result, many studies take into account the hot/cold model proposed by Rosenblum [12]. The model accounts for the hot and cold data location such that fraction f of the total address storage contains hot data while

1-f fraction of the total storage holds cold data of which fraction r of the total writes are made to address locations containing hot data while fraction 1-r of the total writes are made to addresses containing cold data. Peter D. [6] therefore assumed the model to have the value of ≤ 0.2 and r ≥0.8 which simply meant that more than 80% of the writes were made to less than 20% of the address locations that contained hot data. Using this model for non-uniform random writes, it was concluded that the write amplification increases for greedy cleaning for similar values of spare factor in the case of non-uniform traffic and worsens as hot data gets hotter when there is no data separation [6]. In case of hot/cold data separation, it is assumed that the FTL has prior knowledge of which pages contain hot and cold data [5,6]. In Peter D's [6] study, more free space is allotted to hot data than cold data thus decreasing write amplification for hotter blocks and increasing the write amplification for cold blocks. This is done as a reduction in the write amplification for hot blocks compensates easily, for the increase in the write amplification for cold blocks. It was noted that on having a clear hot/cold data separation write amplification decreased from 6.24 to 1.86 when 90% of the writes were restricted to only 5% of address space a spare factor of 0.1 and 64 pages per block. In [5] Hu assumes two model for comparative analysis. A "mixed" model that contains blocks with both cold and hot pages is considered first. The second model assumes that the flash controller is aware of the LBAs of the hot and cold data and thus separates the hot and cold data into separate pools of blocks. Another assumption so made is that, when the PE cycles of the blocks containing hot data get exhausted the block is replaced by a block containing cold data pages only from the cold data block pool. Analysis on these models show that the mixed model shows larger extents of write amplification as the amount of cold data pages increases in blocks. In this "mixed" model case, over provisioning is also inefficient in its ability to reduce the write amplification. This occurs due to the random placement cold data in the blocks. On the other hand, analysis of the second model with separated hot/cold pools of blocks, shows that write amplification is significantly reduced even when the amount of cold data is high but increases (less significantly) for lower values of cold data availability. These models described by Hu [5] were run on 400000 blocks each containing 64 pages. Theoretical and simulated results by W. Bux [14] show that write amplification: increases as the system workload increases, decreases as the number of blocks in the memory increases and increases as the number of pages in the memory blocks increases.

B. FIFO/LRU Cleaning Algorithm

The LRU cleaning algorithm is one of the simplest garbage collection algorithms in SSDs. It is comparable to a FIFO algorithm as both use the method of block cycling to obtain the block that will become the new write frontier (in case of single writes) [6,13]. The algorithm revolves around blocks that have been least recently written which are processed as the write frontier. When a block is filled, it is pushed into an LRU queue which contains a series of filled blocks. As more filled blocks are put into queue the first block is pushed towards end of the queue. On reaching the end of the queue the LRU algorithm picks up the block for cleaning and the block is then used as the write frontier. Analysis of this algorithm involves consideration of the length of the LRU queue, velocity with which the blocks move in the queue, number of pages and the rate of write arrival. The analysis so done and verified through simulation in [6] involved the use of a high-speed simulator programmed using 1000 lines of C and Python code which simulates a 256GB SSD that uses 3 million simulated writes per second. The analysis was carried out on simulated flash memory containing 100000 logical blocks. The results of the analysis show that the for uniform writes, the LRU cleaning algorithm does not depend much on the number of pages within a block, and that the change in number of pages per block has negligible effects on the cleaning algorithm's performance in lowering write amplification. The LRU cleaning however shows inefficiencies in performance under non uniform traffic [6]. This is because the LRU algorithm cleans blocks that have been in the cleaning queue for a fixed duration and hence all the blocks wait for an equal amount of time before their turn for cleaning arrives. As a result, hot data is forced to wait longer within the blocks before getting reclaimed while cold data that does not require urgent cleaning is cleared faster than it should be needed to. This results in a slight increase in write amplification resulting to the conclusion that LRU GC algorithm is somewhat insensitive towards the timely or temporal location of data within the memory blocks. LRU and Greedy GC algorithms both suffer in terms of performance as hot data gets hotter in cases where there is no separation of hot data from cold.

C. Windowed Greedy Cleaning Algorithm

The window greedy cleaning policy is a combination of the greedy and the nature of FIFO/LRU algorithm. This garbage collection algorithm victimizes the blocks containing most invalid pages but only within a window of blocks added to the cleaning block pool. Each filled block added to the pool is allotted an index number with the first one having index number 0. The greedy GC algorithm is invoked only when certain number of blocks are added to the pool. To select a block with the most number of invalid pages, the processing unit of the computing system would need too many cycles to read through all the blocks in the cleaning pool and decide which one is to be cleaned to produce a write frontier. As a result, a window size is chosen to limit the search for a block to be cleaned within the cleaning block pool and reduce the CPU cycle requirement to find the block that is to be cleaned. This window size is kept smaller than the number of blocks in the pool and hence the block with the greatest number of invalid pages is then chosen from this smaller set of blocks. This method is comparable to the age threshold based GC garbage collection algorithms which has not been included for the purpose of this survey, but unlike the age threshold method the window greedy GC algorithm considers a set of oldest blocks in the storage pool rather than blocks defined by an age threshold [5]. Probabilistic analysis for this garbage collection algorithm assumes that data is written sequentially into the block [5]. Analytical results from the same study show that the ideal greedy policy shows better results in terms of write amplification and that the write amplification worsens depending on the window size so chosen for the window greedy cleaning policy i.e. write amplification increases as the window size is decreases.

Comparison has been made between the LRU and window greedy policy in [6] for a 50000 user visible blocks each containing 64 pages and for a window size of 500. The results show that the difference in performance in terms of amplification factor is negligible between the LRU and window greedy algorithm for the same values of spare factor. The only difference in operation between the window greedy and LRU occurs when there is invalidation of data in one of the blocks that comes under the window chosen for the former.

The window greedy policy is thus just thought of as a combination of the greedy GC algorithm with the simplistic nature of the LRU or FIFO GC algorithm.

D. D Choices Cleaning Algorithm

As mentioned earlier, the d choices GC algorithm works on the principle of selecting a block with the greatest number of invalid pages out of a set of d randomly chosen blocks. The difference between this algorithm and the window greedy algorithm resides in the fact that the d choice policy chooses the set of blocks to be cleaned randomly rather than considering a fixed number of the oldest blocks in the pool of blocks that need cleaning. For the purpose of this survey, specifically, the Benny Van Houdt analysis and validation [1] is summarized which in turn considers the Rosenblum Model [12] for studying the performance of the d choice GC algorithm. The models suggested in [1] consider non uniform random writes for single and double log structures following the model in [12] to show how the performance of d choice algorithm is affected and how it is optimal as hot data gets hotter where the greedy algorithm may show degradation in performance for the same. A single log structure makes use of a single write frontier. Therefore, when data is to be written, valid pages from the block are copied to RAM, the block erased and then valid pages along with data to be written are copied back to the erased block thus forming a single write frontier. A single WF however does not lead to the separation of hot and cold data which is why studies like [5,6] put forward analytical and probabilistic studies based on the assumption that the system is aware of the location of hot and cold pages within the blocks. The hot/data identification can be carried out using bloom filters [2] so as to convey the identity of hot and cold data to the system before studying the write amplification performance for a given GC algorithm. The work shown in [1] does not assume predefined hot cold data separation awareness, rather it shows that the use of a double log structure provides an automatic form of hot and cold data separation. The work is modelled and validated with the consideration of page mapped FTLs despite its costly RAM usage, and no issue of wear levelling is taken into account for the same. To establish a base for study, [1] defines a hot/cold model using [12] (also mentioned under the greedy cleaning policy in this survey) where fraction f of the total logical address space is located to hot data and the remaining to cold data in which fraction r of incoming writes is made to hot data and the remaining to cold data. The double log structure, as the name suggests, makes use of two log structures and therefore two write frontiers are required. One write frontier deals with the external write requests made by the host, termed as WFE (Write Frontier External) and the other write frontier deals with the system's internal write requests i.e. the WFI (Write Frontier Internal). It is therefore expected that the WFE will mostly consist of hot data whereas the WFI will consist of mainly cold data. In the case of two write frontiers, the GC policy is invoked when the external write frontier becomes full. Assuming that the GC algorithm selects a block with number of the valid pages b, and the last p out of q pages are in the erased state in the WFI then:

1) If $b \leq q-p$, then the b valid pages are copied to the WFI after which a new WFE is created.

2) If $b \geq q-p$, q-p of the b valid pages are copied to the WFI and the rest are copied to the RAM which are then moved to a different block after it has been erased. The new block therefore becomes the WFI.

It is to be noted that if r=f, the case becomes that of uniform random writes as the probability of invalidation of each becomes equal. A mean field model is therefore defined to validate the case of a single log structure using the Markov chain and ODEs which are solved numerically using the Euler's method. The numerical results were validated with simulated results, ran (10 times) using a custom simulator for 10,000 blocks for different values of spare factor and number of pages in each block. It can be seen that for 32 pages in each block, the write amplification worsens as hot data gets hotter for different values of d, f and r (when the value of f decreases, or the value of r increases). The results show that the method of usage of a single write frontier does not act as a form of hot/cold data separation and even if the system is used with predefined hot/cold data separation, the separation does not last long enough while using a single write frontier. Another mean field model was defined for the double write frontier structure and studied numerically. The results show that for different values of spare factor, number of pages per block as 16,32,64, d ranging from 3 to 24 in steps of 3, r= 0.8,0.9 and f=0.05, 0.2, write amplification decreases as the spare factor increases or as the number of pages per block decreases and that for an optimal value of d which is around 10, the greedy algorithm no longer remains the optimal for the double write frontier model as explained in [1]. The optimal values of d prove to be beneficial as it allows the blocks created by WFE to be emptied more often and help avoid the selection of blocks with a lot of valid pages as assumed, that the WFE contains majorly hot data while the WFI contains majorly cold data. The double log structure therefore provides an automatic form of data separation as the write amplification decreases as the hot data gets hotter. The section of d choices GC algorithm in this survey majorly summarizes the studies and results from [1]. Studies like the ones mentioned in [6] show how blocks can be optimally chosen as the next block to be cleaned based on the CAT (cost/age/time) parameter and how it acts a beneficial metric to select blocks from hot and cold block pools. Peter D. [6] also used a recency-based simulation model that used a weighted moving average parameter which defines a hot and cold page depending on the time interval with which writes are made to the particular page. The model was then run a certain number of iterations to let the model "warm" up and in a way create its own form of data separation. Another study from Benny V. and Robin V. [13] describes how the various system parameters affect SSD endurance, something that not many researches have focused on, as most studies focus on the effect of varied parameters on the write amplification rather than the lifespan of the SSD.

III. CONCLUSION

This paper describes how flash-based SSDs show distinctive advantages over traditional storage media such as Hard Disk Drives (HDDs) due to their design and working. Despite this, flash memory too has its own set of drawbacks because each cell in the flash memory can only be written into, a specific number of times before the cell wears out.

As a result, issues such as write amplification, wear levelling and data fragmentation are commonly seen in flash-based SSDs and threaten their performance and lifespan. The performance of GC algorithms such as LRU/FIFO, Greedy, Windowed Greedy and D choices algorithm has been descried. The Greedy algorithm serves as a basic cleaning policy as it simply choses the block with the least number of valid pages. Computational cost is high as all the blocks must be searched before finding the ideal block for the GC algorithm. The FIFO/LRU algorithm simply adds filled blocks to a queue and blocks at the end of the queue are victimized for cleaning and Write Frontier creation. Greedy and LRU/FIFO policies show similar results with write amplification reduction, however unlike the latter, the Greedy GC policy shows inefficiencies in performance as the number of the pages in a block increase. The Windowed Greedy policy simply chooses a set of the oldest blocks in the storage pool for cleaning and hence reduces computational cost as the search for the block containing the least number of valid pages is limited to a window of blocks. Write amplification, however, depends on the size of the window i.e. write amplification increases as the window size decreases. Studies make use of the Rosenblum model with page, block or hybrid mapping in the Flash Translational Layer for describing the performance of the cleaning policies. It is concluded that, as hot data gets hotter, write amplification due to the greedy GC algorithm worsens where the D choices algorithm, that considers a set of d randomly chosen blocks of which the block with the least number of valid pages is victimized, helps reduces write amplification as the hotness of data increases for an optimal value of d, which is around 10. The introduction of two write frontiers, acts as a form of data separation in which hot data mostly resides in the write frontier allotted for external write requests (WFE) and the cold data mostly resides within the write frontier allotted for internal write requests (WFI).

REFERENCES

1. Benny Van Houdt. Performance of garbage collection algorithms for flash-based solid state drives with hot/cold data. Perform. Eval., 70(10):692–703, 2013.
2. D. Park and D. Du. Poster: Hot and cold data identification for flash memory using multiple bloom filters. USENIX Conference on File and Storage Technologies, 2011.
3. F. Chen, D.A. Koufaty, and X. Zhang. Understanding intrinsic characteristics and system implications of flash memory based solid state drives. ACM SIGMETRICS Perform. Eval. Rev., 37(1):181–192, 2009.
4. J. Hsieh, T. Kuo, and L. Chang. Efficient identification of hot data for flash memory storage systems. ACM Trans. on Storage, 2:22–40, 2006.
5. X. Hu, E. Eleftheriou, R. Haas, I. Iliadis, and R. Pletka. Write amplification analysis in flash-based solid state drives. In Proceedings of SYSTOR 2009: The Israeli Experimental Systems Conference, SYSTOR '09, pages 10:1–10:9, New York, NY, USA, 2009.
6. P. Desnoyers. Analytic modeling of SSD write performance. In Proceedings of International Systems and Storage Conference (SYSTOR 2012), 2012.
7. Jin Youngbin & Lee Ben. (2019). A comprehensive survey of issues in solid state drives. 10.1016/bs.adcom.2019.02.001.
8. T.S. Chung, D.J. Park, S. Park, D.H. Lee, S.W. Lee, and H.J. Song. A survey of flash translation layers. Journal of Systems Architecture, 55:332–343, 2009.
9. C. Min, K. Kim, H. Cho, S. Lee, and Y. I. Eom. SFS: Random write considered harmful in solid state drives. USENIX Conference on File and Storage Technologies, pages 139–155, 2012.
10. R Verschoren, B Van Houdt. On the Impact of Garbage Collection on Flash-Based SSD Endurance. 4th Workshop on Interactions of NVM/Flash with Operating Systems and Workloads(2016).
11. Yang Yudong,Vishal Misra & Rubenstein D. (2015). On the Optimality of Greedy Garbage Collection for SSDs. ACM SIGMETRICS Performance Evaluation Review. 43. 63-65.
12. M. Rosenblum and J. K. Ousterhout. The design and implementation of a log-structured file system. ACM Trans. Comput. Syst., 10(1):26–52, February 1992.
13. L. Xiang and B. Kurkoski. An improved analytical expression for write amplification in NAND flash. In International Conference on Computing, Networking, and Communications (ICNC), pages 497–501, 2012.
14. W. Bux and I. Iliadis. Performance of greedy garbage collection in flash-based solid-state drives. Perform. Eval., 67(11):1172–1186, November 2010.

AUTHORS PROFILE

Rishabh Gogna: currently pursuing a Bachelor of Technology degree in the course of Electronics and Communication Engineering (2017-2021) from Amity University Mumbai as a part of Amity School of Engineering and Technology (ASET).
E-mail ID: rgogna00@gmail.com