A closed-form expression for Write Amplification in NAND Flash

Rajiv Agarwal and Marcus Marrow

{ragarwal, mmarrow}@link-a-media.com

***Abstract*—The log-structured filesystems typically used in current solid-state drive's (SSD) exhibit write amplification, whereby multiple NAND writes are required for each host write. Write amplification negatively affects the SSD endurance and write throughput. This performance loss depends on the drive over-provisioning and the garbage collection method. This paper presents a novel probabilistic model to analytically quantify the impact of over-provisioning on write amplification under a uniformly-distributed random workload and a greedy garbage collection policy. The analysis shows write amplification approximately independent of NAND block size and number of blocks in the SSD. The analysis is verified by full drive simulations.**

I. Introduction

In recent years, flash memory has become an important storage medium because it has many attractive features such as small size, shock resistance, high reliability, low power consumption, and lightweight. Flash memory is a non-volatile solid state memory whose density and I/O performance have improved to a level at which it can be used widely. Flash memory is partitioned into blocks and each block has a fixed number of pages, typically 64 pages of 4 KiB each [1]. Data are written in a unit of one page, and the erase is performed in a unit of one block. Reads are performed in units of pages.

In flash-based systems, out-of-place write [2], [3], [4] is used: When a page or a fraction thereof needs to be rewritten, the new data does not overwrite the memory location where the data is currently stored. Instead, the new data is written to another page and the old page is marked invalid. A mapping scheme is needed to maintain a map of logical addresses of data to physical locations on flash memory, and this map is updated consequently. Over time, the SSD accumulates invalid pages and the number of free pages decreases. To make space for new incoming host data, invalid pages must be rewritten. The limitation of the SSD is: in order to write new data to replace invalid data in a page, the page must be free, which requires erasing the entire block the page pertains to. Under the current technology, a block could tolerate a limited number of cycles before the block becomes unreliable. The number is typically 100K cycles for single-level cells (SLC) and reduces drastically for multi-level cells (MLC) flash to less than 10K. As applications continually update data on flash memory (as digital camera users store and later delete files to store new ones), the need to limit the erase operations to a minimum becomes critical.

As block(s) of flash memory need to be erased before they are rewritten, the software layer of a flash memory device contains a garbage-collection mechanism to translate invalid pages to free pages. Reclamation or garbage collection happens in multiple steps: First, one of the blocks is selected for reclamation. Second, the valid pages of this block are copied to a reserved free block. Third, the reclaimed block is erased and becomes free. On-demand reclamation, which is triggered when the free space has been completely exhausted, is preferred in order to minimize relocating the valid pages in the block being reclaimed and the consequent extra page write operations. Whereas the reclaiming policy that selects the blocks to garbage-collect is usually based only on the amount of free space to be gained [3], the policy defined in [5] also included the time elapsed since the last writing of the block with data. The effectiveness of the garbage-collection mechanism is influenced by the policy according to which blocks are selected for reclamation. This paper considers the so-called "greedy" policy, in which the block with the smallest number of valid pages is selected for garbage collection. This approach minimizes the number of valid pages that need to be re-written in the course of garbage collection.

Additionally, in a NAND Flash, strong localities in user write pattern can lead to repeated erasing of a certain small portion of the drive, thus reducing drive lifetime. Cold data comprises of files such as the operating system (OS), which once stored is not deleted for a long time, whereas hot data comprises of users work files such as log files which are updated/deleted on a regular basis. Hence, blocks storing cold-data tend to not have their program erase (PE) cycle count increased at all, whereas blocks storing hot-data experience a continuous increase in their PE cycle count. As hot data and cold data differently wear flash memory, its overall lifespan could be unexpectedly short under such workloads. Wear leveling refers to system activities that prevent blocks from being unevenly worn so as to lengthen the overall lifespan.

Therefore, in flash, write amplification corresponds to the additional writes caused by garbage collection and by wear leveling. Hence, the total number of user writes that can be served depends on the total cycle budget available, write amplification, and the eventual unconsumed cycle budget due to wear-leveling inadequacy.

This paper derives a closed-form expression for write-amplification as a function of flash system parameters. The analysis assumes a simplistic model of the NAND-flash memory, in which, all the incoming host data is hot. The analysis pertains to applications or usage scenarios that uniformly update data on flash memory exclusively, e.g. digital camera users store and later delete files to store new ones. In these cases, the entire flash memory is evenly worn. This paper assumes absence of cold-data and workloads of uniformly-distributed random writes; hence, no wear-leveling is required.

Furthermore, a greedy garbage collection policy, which

978-1-4244-8865-0/10/$26.00 ©2010 IEEE

minimizes the number of additional writes, is considered. Specifically, when garbage collecting to accommodate for a new host write request, block(s) with the most 'amount of free space to be gained' or equivalently 'least number of valid pages' is selected. Such a policy will be optimal for the usage scenario described in the previous paragraph [6]. The derivation studies the evolution of the distribution of valid pages in blocks over time and shows that it converges to a uniform distribution numerically. Thereby, the paper derives a closed-form expression for write-amplification.

The rest of the paper is organized as follows. Section 2 reviews the relevant work in garbage collection. Section 3 introduces an analytical model based on a probabilistic approach, which is followed with write-amplification analysis for this model in Section 4. Based on this, the paper presents essential analytical and simulation results to quantify write amplification in Section 5. Finally, a discussion of results concludes the paper.

II. Prior Work

Due to the immense impact garbage collection and resulting write amplification have on flash memory's performance [7], several works in literature have studied efficient garbage collection policies. This section mentions a few of them, which are of direct interest. Most current systems implement the "greedy" garbage collection policy [6], in which the block that has the largest number of invalid pages will be recycled. For systems with a large number of blocks, implementing the optimal greedy policy may become prohibitive because of the potentially long searches needed to find the optimal block for reclamation. Papers [6], [1], [8] have studied less expensive sub-optimal selection algorithms, which are of practical interest.

Since both garbage collection and wear leveling contribute to writes in addition to those requested by the host, several works have proposed combining garbage collection and wear leveling. The one described by Chang et al. [9] avoids unnecessary reclamations in garbage collection and combines this with wear leveling in the form of a periodical task that performs a linear search for blocks with a small erase count to identify blocks to be recycled. Agrawal et al. [10] describe another combined algorithm called modified greedy garbage-collection strategy. The algorithm generally selects the block with the most invalid pages for garbage collection, while avoiding a large spread in the remaining cycle budget among all blocks and limiting the frequent movement of cold data. Ben-Aroya et al. [11] performed a worst-case competitive analysis with focus on endurance-based randomized algorithms. On a different note, Jagmohan et al. [12] proposed using a multilevel code, which allows rewriting without erase, to reduce write amplification.

Although, many flash memory papers briefly mention performance impacts from garbage collection and wear leveling, derivation of a simple closed-form expression for write amplification in flash-based storage systems, and how it relates to parameters, such as the over-provisioning factor, remains largely unstudied in the literature. Bux models the garbage collection process as a Markov chain to evaluate the performance of a 'select-a-random-block-for-reclamation' garbage collection policy, and finds that performance depends on the total memory space, the fraction of the memory available for storing user data, and the number of pages per block. Rosenblum et al. [3] studied write amplification as a function of disk utilization. Their analysis distinguishes between hot and cold data and their write-cost comparison includes time for seeks, rotational latency, and cleaning costs, which makes analysis for this model fairly involved and a neat closed-form expression becomes harder to derive. Analysis for write-amplification was done in [1] for the case when both garbage-collection and wear-leveling are in operation using a windowed 'greedy' garbage-collection policy, however, in this case, the system dynamics are more involved; hence, once again, a simple closed-form expression is hard to derive.

III. System Model

The flash memory is organized in terms of blocks, with each block containing a fixed number N_p of pages, typically $N_p = 64$. A flash driver is used to translate any given logical block addresses (LBAs) to physical block addresses (PBAs), where each LBA represents a storage unit corresponding to a page. When a page addressed by an LBA is rewritten, a free flash memory physical page will be allocated to store the new data, and the controller updates the LBA-PBA map: the LBA is mapped to the new PBA and the old PBA is marked as invalid data page. Garbage collection is used to reclaim space that stores invalid data, whenever there are no sufficient free pages.

A. Write-Amplification and Over-Provisioning

There is no garbage collection needed for a sequential write workload as an entire Flash block is made invalid during the write process, and it can be erased and reclaimed without any data movement. For write requests that come in random order of LBA's, after a while, the number of free pages in flash memory becomes low. The garbage-collection mechanism then identifies a candidate block for cleaning based on a given policy. All valid pages in the candidate block are relocated into a new block with free pages, and finally the candidate block is erased so that N_p pages become available for rewriting. Consequently, this mechanism introduces additional read and write operations, the extent of which depends on the specific policy deployed, as well as on the system parameters. These additional writes result in the multiplication of user writes, a phenomenon referred to as "write amplification" defined below:

Write Amplification: In a SSD, write amplification, W, due to garbage collection is defined as the average of actual number of page writes per user page write [1].

Suppose there are N_{ip} invalid pages in a block selected for garbage collection. The block has N_{vp} valid pages, where $N_{ip} + N_{vp} = N_p$. These valid pages have to be written to a different block, before the block selected for garbage

collection can be erased. Hence, to (re)write N_{ip} pages worth of new user data, the number of physical pages that have to be written is $N_{ip} + N_{vp}$. Therefore, the write amplification is

$$W = \frac{N_{ip} + N_{vp}}{N_{ip}} = 1 + \frac{N_{vp}}{N_{ip}} \quad (1)$$

The term $\frac{N_{vp}}{N_{ip}}$ corresponds to the extra write requests arising from the relocation of valid pages. The garbage collection policy should attempt to minimize write amplification. The greedy reclaiming policy leads to the lowest contribution to write amplification with independently, randomly, uniformly distributed writes of hot data only [13].

The impact of garbage collection on write amplification is influenced by the following factors: the level of over-provisioning, the choice of reclaiming policy, and the types of workloads. For convenience of analysis, the paper assumes a pathological workload, i.e., the random write workload, for which the 'greedy' garbage collection policy was shown to be optimal [13]. Over-provisioning refers to a common practice that the user address space can only take a fraction of the raw Flash memory capacity. Because of out-of-place writes, over-provisioning exists in practically all Flash SSDs. Over-provisioning is defined as follows.

Over-Provisioning: Suppose an SSD with a raw storage capacity of T blocks, of which, the user can only use a part, say, U blocks, and where $U \leq T$. Then the over-provisioning factor, ρ , is defined as $\rho = \frac{T-U}{U}$. So, for an SSD with $\rho = 0.25$, there is 25% over-provisioning, only 80% of the drive is available for user-data.

The over-provisioning or conversely the utilization is closely tied to write amplification due to garbage collection. The larger the utilization, the more likely a block selected to be reclaimed has many valid pages that have to be relocated, and the worse the write amplification.

B. Greedy Reclaiming Policy

This subsection describes the greedy garbage collection policy under consideration, which is the optimal garbage collection policy in the sense of minimizing write amplification under the random write workload [13]. Incoming write requests and relocation write requests are both serviced by writing to free pages/blocks. Once a free block has been filled up, it is removed from the free block pool and moves to the end of the occupied block queue. Each time garbage collection is triggered, a single occupied block is selected, and all its valid data pages are read and written to another location by issuing relocation write requests. Upon completion of relocation, the selected block is erased and joins the free block pool again. Denoting the number of free blocks by P and the total number of physical Flash blocks by T, garbage collection using the greedy reclaiming policy only starts once $T - P$ blocks are occupied, with P blocks reserved so that garbage collection can operate in parallel. This pool of free blocks is kept very small, because it eats into the over-provisioning. This paper assumes that this pool is negligibly small i.e. $P \ll T$ and hence it can be neglected from analysis. Henceforth, $T - P \approx T$.

The greedy reclaiming policy, based on its description above, would consume too many CPU cycles to select the block with the least number of valid pages from all T blocks, because each host write may decrease the number of valid pages in already written blocks. As a result, this would require to constantly update the number of valid pages in all T blocks. This paper assumes that CPU cycles to do this update are available. In practice, this list is updated only so often, and therefore at any given point of time, the list may not be 'fully' sorted. The impact of this delay in sorting is studied via simulations and is skipped during analysis. It is worth mentioning there though, that this delay, results in an effective reduction in over-provisioning and hence a higher write-amplification.

IV. Derivation of Write-Amplification

The analysis assumes that each (re)write request has a fixed request size of a single block, with the page addresses distributed i.i.d. according to a uniform distribution specified as Unif$[0, UN_p - 1]$. Garbage collection is triggered when there are no free pages available. Once the garbage collection process kicks-off, at every instant the block with the most number of invalid pages is selected and freed and the incoming host rewrite request of one block is met. The LBA-PBA map is updated accordingly. Due to the random uniform nature of host writes, the distribution of valid pages in a block right before the start of garbage-collection process, denoted by $f^{(0)}(v)$, will be the binomial distribution and the mean number of valid pages in a block will be $\frac{1}{1+\rho}N_p$. Specifically:

$$f^{(0)}(v) = \text{Prob}(N_{vp} = v) = \binom{N_p}{v} p^v \, (1-p)^{(N_p - v)} \quad (2)$$

where $\frac{1}{1+\rho} = p$.

To analyze write amplification, the evolution of the distribution of valid pages $f(v)$ in a block is studied. Since the greedy garbage collection policy selects the block with the least number of valid pages, the write amplification only depends on the statistics of this block. Let x denote the minimum possible number of valid pages in a block i.e. $x = \{\min v : f(v) > 0\}$.

Through extensive simulations, it was seen that the distribution of valid pages in a block $f(v)$ converges to a steady state distribution. Figure 1 plots the distribution of valid pages for $T = 1280$, $N_p = 256$ and $\rho = 0.25$, after 500 host block writes, averaged over its 10^3 realizations. Also shown in the figure 1 is the uniform distribution approximation to the empirical histogram, with the same non-zero support. The Uniform distribution approximation to the distribution of valid pages given as Unif$[x, N_p]$, for some x, is seen to be a reasonable fit.

With the above approximation, write amplification can be derived as follows. Use of Eqn (1) gives

$$W = \frac{N_p}{N_p - x} \quad (3)$$

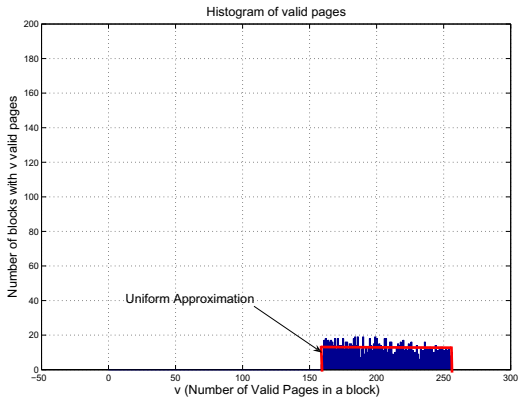

Fig. 1. Histogram of valid pages.

However, if in steady state $f(v)$ becomes the uniform distribution i.e. $v \sim \text{Unif}[x, N_p]$,

$$\mathbb{E}[v] = \frac{N_p + x}{2} = \frac{1}{1+\rho} N_p = pN_p \tag{4}$$

Hence,

$$\begin{aligned} W &= \frac{N_p}{N_p - (2pN_p - N_p)} \\ W &= \frac{1}{2}\left(\frac{1+\rho}{\rho}\right) \end{aligned} \tag{5}$$

which is seen to be independent of the number of pages per block. As a sanity check for the derived expression, when over-provisioning ρ is 0 (minimum value), W is infinity (maximum value). Similarly, when ρ is 1 (maximum value), W is 1 (minimum value). Hence, both these extreme values are correct. Write amplification, in general, depends on the number of pages in a block. For example, in the extreme case, if $N_p = 1$, then $W = 1$ regardless of over-provisioning ρ. However, as shown via simulation results in the next section, for reasonable values of N_p (64 and higher), write-amplification W is seen to be independent of N_p, supporting the result in this section.

V. Monte-Carlo Simulations

This section performs Monte-Carlo simulations to compare the derived analytical expression with actual write-amplification values observed for the case under study.

Figure 2 shows write-amplification over time using Monte-Carlo simulations for a fixed drive size $U = 1024$. The write-amplification converges to a value 2.67 fairly quickly regardless of the number of pages in a block. This value is close to the theoretically derived approximate value of 2.5 when $\rho = 0.25$. Figure 3 shows write-amplification over time using Monte-Carlo simulations for a fixed number of pages per block $N_p = 256$. Both the over-provisioning and drive size U are varied. Figure 3 shows that write-amplification depends only on ρ regardless of the value of drive size U. Furthermore, the steady-state value of write-amplification from Monte-Carlo simulations is seen to be 2.67, 3.18 and 3.96, which is fairly close to the corresponding analytical approximation of 2.5, 3 and 3.83, when over-provisioning ρ is 25%, 20% and 15% respectively.

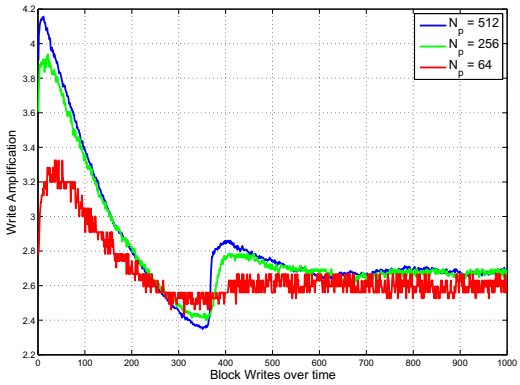

Fig. 2. Write-Amplification over time for 25% over-provisioning with fixed $T = 1024$.

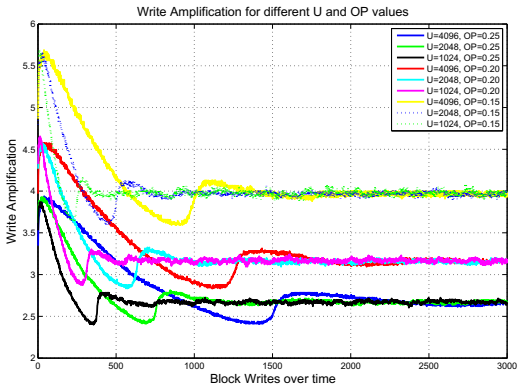

Fig. 3. Write-Amplification over time for {25%, 20%, 15%} over-provisioning with fixed $N_p = 256$.

Finally, figure 4 shows the effect of a delay in updating the sorted list of number of valid pages in a block for the whole drive. The delay is measured in terms of the number of host block rewrite requests. Figure 4 shows that this delay translates to an increase in the write-amplification value, hence effectively reducing the over-provisioning. The system designer can trade-off these two design parameters - over-provisioning (decreases effective storage space) and delaying the sort (reduces CPU cycles spent in book-keeping). An expression for the fit of this curve as a function of number of blocks U (in the figure denoted as NBlock) and delay D is also provided.

VI. Summary and Discussion

This paper studied write-amplification arising from garbage-collection in a NAND Flash. A simple closed-form expression

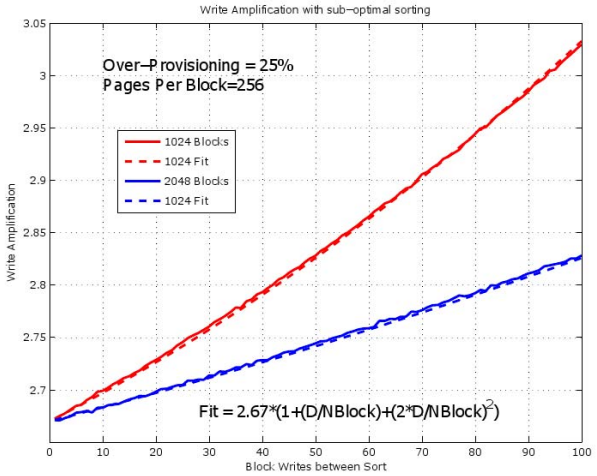

Fig. 4. Write-Amplification over time with delayed sort v/s the delay.

for write-amplification was derived and it was shown to depend only on over-provisioning and shown to be independent of drive size or number of pages in a block. Monte-Carlo simulations verified the same. Extending this analysis in the presence of cold-data and hence a wear-leveling algorithm remains future work.

REFERENCES

[1] X.-Y. Hu, E. Eleftheriou, R. Haas, I. Iliadis, and R. Pletka, "Write amplification analysis in flash-based solid state drives", in *SYSTOR 2009: The Israeli Experimental Systems Conference*, 2009.

[2] J. K. Ousterhout and F. Douglis, "Beating the I/O bottleneck: A case for log-structured file systems", in *Operating Systems Review*, vol. 23, no. 1, pp. 1128, Jan. 1989.

[3] M. Rosenblum and J. K. Ousterhout, "The design and implementation of a log-structured file system", in *ACM Transactions on Computer Systems*, vol. 10, no. 1, pp. 2652, Feb. 1992.

[4] J. Menon, "A performance comparison of RAID-5 and log-structured arrays", in proceedings of *Fourth IEEE International Symposium on High Performance Distributed Computing*, pp. 167178, Aug. 1995.

[5] J. Menon and L. Stockmeyer, "An age-threshold algorithm for garbage collection in log-structured arrays and file systems", in *J. Schaeffler, editor, High Performance Computing Systems and Applications*, pp. 119132. Kluwer Academic Publishers, 1998.

[6] W. Bux, "Performance evaluation of the write operation in flash-based solid- state drives", in *IBM Research Report*, RZ 3757, Nov. 2009.

[7] A. Kawaguchi, S. Nishioka, and H. Motoda, "A flash-memory based file system", in proceedings of *USENIX Technical Conference*, pp. 155164, Jan. 1995.

[8] J. Menon and L. Stockmeyer, "An age-threshold algorithm for garbage collection in log-structured arrays and file systems", in *J. Schaeffler, editor, High Performance Computing Systems and Applications, Kluwer Academic Publishers*, pp. 119132, 1998.

[9] L.-P. Chang, T.-W. Kuo, and S.-W. Lo, "Real-time garbage collection for flash-memory storage systems of real-time embedded systems", in *ACM Transactions on Embedded Computing Systems*, vol. 3, no. 4, pp. 837863, Nov. 2004.

[10] N. Agrawal, V. Prabhakaran, T. Wobber, J. D. Davis, M. Manasse, and R. Panigrahy, "Design tradeoffs for SSD performance," in proceedings of *USENIX Annual Technical Conference*, June 2008.

[11] A. Ben-Aroya and S. Toledo, "Competitive analysis of flash-memory algorithms," in proceedings of *14th Annual European Symposium on Algorithms (ESA)*, pp. 100111, Sep. 2006.

[12] A. Jagmohan, M. Franceschini and L. Lastras, "Write amplification reduction in NAND Flash through multi-write coding", in *IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, pp. 1-6, 2010.

[13] X.-Y. Hu and R. Haas, "The fundamental limit of flash random write performance: understanding, analysis and performance modelling", in *IBM Research Report*, RZ 3771, Mar. 2010.