An Improved Analytic Expression for Write Amplification in NAND Flash

Xiang Luojie, Brian M. Kurkoski
Dept. of Information and Communications Engineering
The University of Electro-Communications
1-5-1 Chofugaoka, Chofu
Tokyo, Japan

Abstract—Agarwal et al. gave an closed-form expression for write amplification in NAND flash memory by finding the probability of a page being valid over the whole flash memory. This paper gives an improved analytic expression for write amplification in NAND flash memory by finding the probability of a page being invalid over the block selected for garbage collection. The improved expression uses Lambert W function. Through asymptotic analysis, write amplification is shown to depend on overprovisioning factor only, consistent with the previous work. Comparison with numerical simulations shows that the improved expression achieves a more accurate prediction of write amplification. For example, when the overprovisioning factor is 0.3, the expression proposed by this paper gives a write amplification of 2.36 whereas that of the previous work gives 2.17, when the actual value is 2.35.

I. INTRODUCTION

Flash memory is a storage medium with growing significance. It has many appealing features including non-volatility, small size, low-cost, mechanical reliability, low power consumption and low read latencies particularly when compared to hard disk drives [1] [2].

Flash memory is organised in blocks. A block has a fixed number of pages (typically 64 pages). A page has a fixed size (typically 4KiB) [3]. There are three kinds of operations on flash memory: read, write and erase. Read and write operation can be performed on a page basis [4].

Flash memory has limitations that challenge the design of flash memory systems. One fundamental limitation of flash memory is nonsupport of overwriting and block erase. After data are written into a page, new data can not be written into the page by overwriting. The page must be erased before new data can be written to it. But erase operation can be performed only on a block basis [5]. Another fundamental limitation of flash memory is limited endurance. Flash memory can tolerate a limited number of program and erase cycles before it becomes unreliable. The latest multi-level cell (MLC) memories can endure 5000 ~ 10000 program and erase cycles [1] [2]. Thus, it is a critical problem to limit the number of program and erase cycles to a minimum to increase the flash memory’s lifetime.

Erasing an entire block when a page is needed to be updated is inefficient. Moreover, this can wear out the flash memory very quickly. Thus, in flash memory, out-of-place write is used [2]. When a page already written needs to be updated, the new data are not written into that page but to a new page free for writing and the old page is marked as invalid. A mapping table is maintained to record the mappings of logical address and physical address. In this paper, pages are classified into three categories and their definition is given as follows. A free page is a page into which no data have been written. It is available to accommodate new user writes. A valid page is a page into which data have been written but hasn’t been updated. A valid page is the page that stores the user data. An invalid page is a page into which data have been written and has been updated. An invalid page once stores user data, but it is updated and no longer stores user data due to out-of-place write.

Out-of-place results in invalid pages. Invalid pages consume the flash memory but do not store user data. Thus, when the number of invalid pages accumulate to some extent, the invalid pages should be reclaimed for new user writes. The mechanism in flash memory to reclaim invalid pages and translate them into free pages is called garbage collection [2]. Garbage collection is performed in the following way: first a block is selected for garbage collection. Then, the valid pages in the selected block are copied to some other free space in the flash memory. After this, the selected block is erased and becomes free. Various algorithms for garbage collection have been proposed in previous work [6] [7] [8].

During garbage collection, valid pages in the selected block are copied to some other free space before the selected block is erased and copied back to the flash memory afterwards. This copy operation causes additional writes. The actual number of writes on the flash memory would be more than the number of pages needed to be written. This phenomenon is called write amplification. Write amplification reduces the flash memory’s lifetime and therefore should be minimized. In flash memory, a common practice is that, user can only use a portion of the raw flash memory space. The portion the user cannot use is called overprovisioning. Overprovisioning provides flash memory with an increased endurance and an improved performance. Increasing the amount of overprovisioning decreases write amplification.

This research was supported in part by the Ministry of Education, Science, Sports and Culture; Grant-in-Aid for Scientific Research (C) number 21560388 and Grant-in-Aid for Scientific Research (C) number 23560439. Contact email: kurkoski.ice.uec.ac.jp
Due to the significance of minimizing write amplification, previous works analyzed write amplification. Hu et al. first developed a probabilistic model to describe the write amplification [3]. They then developed an empirical model to compute the write amplification [4].

Agarwal et al. gave a closed-form expression for write amplification [2]. They assumed that initially, the user space of the flash memory is full and the valid pages are randomly distributed over the physical flash memory. The number of valid pages in an arbitrary block is then binomially distributed. After a sufficiently large number of user writes, the distribution of the number of valid pages in an arbitrary block is empirically approximated by a uniform distribution. Because both the total number of pages and the total number of valid pages remains the same, the average number of valid pages in an arbitrary block remains the same. Thus, the expected value of these two distributions is equal. By solving this equation, an expression for write amplification is obtained which is a function of overprovisioning factor.

This paper follows this work, and proposes an improved analytical expression for write amplification in NAND flash memory. First, $X$, the number of invalid pages freed by each garbage collection is studied. After a sufficiently large number of user writes, $X$ evolves into a stationary state. We assume that, in the stationary state, the expected value of $X$ is a constant $x$. Another assumption is made and justified that, $X$ has a binomial distribution with probability $p_e$. However, $p_e$ and the expected value of the binomial distribution depend on $x$. Since the expected value of the binomial distribution is $x$, we form an equality which can be solved for $x$. An asymptotic value of $x$ and an asymptotic value of write amplification $A$ are obtained using Lambert W function when the total number of pages accessible to the user is very large. As with the previous work, $A$ is a function of overprovisioning factor $\rho$.

The key of both papers is finding the probability of a page being invalid or valid. Agarwal et al. find the probability of a page being valid over the whole flash memory while this paper finds the probability of a page being invalid over the block selected for garbage collection. Comparison shows that the improved expression achieves a more accurate prediction of write amplification than the previous work. For example, when the overprovisioning factor is 0.3, the improved expression gives a write amplification of 2.36 and that of the previous work gives 2.17 whereas the actual value is 2.35.

The rest of the paper is organized as follows: Section II specifies the system model assumed. The analysis of write amplification and derivation of closed-form expression are given in Section III. In Section IV, the assumptions in Section III is justified and the comparison between the result of this paper and previous work is given.

II. SYSTEM MODEL

The flash memory is organized in blocks. There are a total of $T$ physical blocks in the flash memory. Each block has a fixed number of pages, denoted by $N_p$. The portion of the flash memory space that user can use is $U$ blocks. Overprovisioning factor $\rho$ is defined by:

$$\rho = \frac{T - U}{U}. \quad (1)$$

The flash memory maintains a free block pool and an occupied block queue. Blocks that have at least one free page are in the free block pool and blocks with no free pages are in the occupied block queue. Initially, the flash memory is empty and all the blocks are empty and are in the free block pool. This paper assumes random writes which are uniformly distributed on the user address space. Each write request has a fixed size of one page. When a write request comes, data are written to a free page in a block in the free block pool. If the write request falls on a user address that has not been written before, this write request is called a “write”. If the write request falls on a user address that has already been written, this write request is called an “update”. When it is a write, a free page is found and the data are written into that page. The mapping between logical address and physical page address is recorded in a mapping table. When it is an update, the physical page storing the old data is found through the mapping table and marked as invalid. A free page would be found to write the new data and the mapping table is updated accordingly. Gradually, the blocks in the free block queue become full and are moved into the occupied block queue. When the number of free pages reaches a predefined threshold, garbage collection is triggered to collect the invalid pages. Greedy garbage collection is assumed where garbage collection is triggered when no free page exists. The block with the maximum number of invalid pages is selected for garbage collection. The valid pages in the selected block is copied to some auxiliary space in the flash memory. Then the selected block is erased. Finally the valid pages is copied back to this block. This block now joins the free block pool and is the only block in the free block pool accommodating write requests.

Suppose there are $N_{ip}$ invalid pages freed by garbage collection. Notice that, copying the valid pages contributes $N_p - N_{ip}$ writes on the flash memory. And as a result, $N_{ip}$ free pages are obtained and therefore $N_p$ new user writes can be accommodated. Write amplification, $A$, is defined as the average number of physical page writes per user page write [2]. It is given by:

$$A = \frac{N_p - N_{ip} + N_{ip}}{N_{ip}} = \frac{N_p}{N_{ip}}. \quad (2)$$

Due to the greedy garbage collection policy, at most one block can have free pages and be in the free block pool. Thus, the flash memory model can be further simplified into a single block queue. Blocks are sorted by their time in the queue since they were last selected for garbage collection. They are labeled 0, 1, 2, \ldots, $T - 1$ from the head of the queue. When no free pages exist in any block, the block with the maximum number of invalid pages is chosen and removed from the queue. All blocks after this block moves one step towards the head of the queue. After the erase operation, this selected block is added.
to the tail of the queue with $N_{ip}$ free pages and $N_p - N_{ip}$ valid pages. In this model, only the last block (block $T - 1$) can have free pages and garbage collection is triggered when this block is full.

A demonstration of this model is shown in Fig. 1. “V” stands for valid, “I” for invalid and “F” for free. The demonstration includes four blocks labeled block 0,1,2,3 and each block has four pages. The overprovisioning factor is $\frac{2}{3}$. This means the user can only use 12 pages out of the 16 pages. In step (a), there are 12 valid pages in total indicating that the user address space is already full and only update can be performed. At this time the flash memory is full and garbage collection is triggered. Block 1 should be selected for garbage collection because it has the maximum number of invalid pages. Block 1 is removed from the queue and block 2,3 moves one step towards the head of the queue becoming block 1,2. The valid pages in the selected block shown as blue V’s in step (a) are copied to some auxiliary space and then the selected block is erased and added to the tail of the queue becoming block 3 as is shown in step (b). The valid pages are copied into Block 3 shown as blue V’s in step (b). The original two invalid pages now becomes free pages accommodating new user writes. Two user writes updates the red pages shown in (b). These pages are first marked as invalid shown as red I’s in (c) and new data are written into block 3 shown as red V’s in (c). Now garbage collection is triggered again. Block 0 should be selected for garbage collection which is performed in the same way as in step (a).

III. WRITE AMPLIFICATION ANALYSIS

For the convenience of explanation later, first the definition of a block cycle is given. A block cycle is the process from when garbage collection is performed on a block to the time when this block is selected again for garbage collection. From the point of view of the block queue, a block cycle is the process from a block entering the block queue being block $T - 1$ to the time when it is removed from the block queue for garbage collection.

Let $X$ denote the number of invalid pages freed from garbage collection. The following assumption was used in [2] and is further supported by simulation in Section IV.

Assumption 1: After a sufficiently large number of user writes, $X$ evolves into a stationary state and has a constant expected value $E(X) = x$.

For any block selected for garbage collection, the following assumption is given and will be justified later.

Assumption 2: When a block is selected for garbage collection, all the pages in this block have the same probability $p_{ie}$ to be invalid.

Due to this assumption, $X$ is a binomial random variable with $N_p$ trials and each trial has a probability $p_{ie}$ to be successful. The expected value of $X$ is $N_p p_{ie}$. Therefore, using Assumption 1

$$N_p p_{ie} = x. \quad (3)$$

The following derives $p_{ie}$. Consider a certain valid page $\alpha$. It corresponds to one page of data in the user space. Because the user writes are randomly distributed on the user space and are mutually independent and the total user space is $UN_p$ pages, a user write has a probability of $\frac{1}{UN_p}$ to update the data page $\alpha$ stores, and make it invalid. After $k$ user writes, page $\alpha$ remains valid if none of these $k$ user writes update the data that page $\alpha$ stores. Let $p_{valid}$ denote the probability that a valid page $\alpha$ remains valid after $k$ user writes,

$$p_{valid} = (1 - \frac{1}{UN_p})^k. \quad (4)$$

Then, $p_{invalid}$, the probability that page $\alpha$ becomes invalid after $k$ user writes:

$$p_{invalid} = 1 - p_{valid} = 1 - (1 - \frac{1}{UN_p})^k. \quad (5)$$

Let $k_e$ denote the total number of user writes during an entire block cycle. According to Assumption 2, $p_{ie}$ is the value of $p_{invalid}$ when $k = k_e$. Thus, to determine $p_{invalid}$, $k = k_e$ should be determined. According to Assumption 1 each garbage collection collects $x$ invalid pages. This means, after each garbage collection, there are $x$ free pages in the block $T - 1$ translated from invalid pages freed by garbage collection. Then, $x$ user writes fills this block and triggers the next garbage collection. Thus, between two garbage collections, one block (block $T - 1$) becomes full and there are $x$ user writes.

Assumption 3: A number of $T$ blocks become full during an entire block cycle.

![Fig. 1. Demonstration of simplified model](image-url)
This assumption will be justified by simulation in Section IV. Using this assumption,

\[ k_e = T x. \]

Apply equation (6) to equation (5), \( p_{ie} \), the value of \( p_{invalid} \) when \( k = k_e \), is:

\[ p_{ie} = 1 - (1 - \frac{1}{U N P})^{T x}. \]

Apply equation (7) to equation (3),

\[ N_P(1 - (1 - \frac{1}{U N P})^{T x}) = x, \]

where \( N_P, T, U \) are constants for a specific flash memory. Let \( W(x) \) denote the Lambert W function [9]. Solving this equation for \( x \),

\[ x = \frac{W(N_P T (1 - (1 - \frac{1}{U N P})^{T N_P} \ln(1 - \frac{1}{U N P})))}{T \ln(1 - \frac{1}{U N P})} + N_P. \]

Thus, write amplification \( A \) is,

\[ A = \frac{N_P}{x} = \frac{N_P T \ln(1 - \frac{1}{U N P})/((N_P T \ln(1 - \frac{1}{U N P}) - W(N_P T (1 - (1 - \frac{1}{U N P})^{T N_P} \ln(1 - \frac{1}{U N P}))))}}{N_P (1 - \frac{W((-1 - \rho)e^{-1-\rho})}{-1 - \rho})}. \]

As was mentioned, practical flash memory has a very large number of blocks so that \( U N P \gg 1 \). Consider \( \hat{x} \), the asymptotic value of \( x \) as \( U N P \) becomes large:

\[ \hat{x} = \lim_{U N P \to \infty} \frac{W(N_P T (1 - (1 - \frac{1}{U N P})^{T N_P} \ln(1 - \frac{1}{U N P})))}{T \ln(1 - \frac{1}{U N P})} + N_P \]

\[ = N_P (1 - \frac{W((-1 - \rho)e^{-1-\rho})}{-1 - \rho}). \]

The asymptotic write amplification \( \hat{A} \) is:

\[ \hat{A} = \frac{N_P}{\hat{x}} = \frac{N_P}{1 - \rho - W((-1 - \rho)e^{-1-\rho})}. \]

For comparison, the closed-form expression for write amplification \( A_{pre} \) proposed in the previous work [2] is:

\[ A_{pre} = \frac{1}{2} (\frac{1 + \rho}{\rho}). \]

Comparison of equation (12) and equation (13) shows that, the asymptotic write amplification expression obtained by this analysis is a function of overprovisioning factor only, which is consistent with the previous work. It is independent of the number of pages per block.

Now, Assumption 2 is justified. Consider a certain block, at the very beginning of its block cycle with \( x \) free pages and \( N_P - x \) valid pages. After \( k_e \) user writes, this block is again selected for garbage collection. Thus, all the valid pages in the beginning of the block cycle have the same probability \( p_{ie} \) to be invalid at the end of the block cycle. The free pages in the beginning of the block cycle are written consecutively and each of them experience one less user write than the previous page. Practical flash memory has a very large number of blocks so that \( T x \gg 1 \). Thus, it is reasonable to assume that each page in a block has the same probability \( p_{ie} \) to be invalid when selected for garbage collection.

IV. SIMULATION

A flash memory simulator was written and simulations were run to justify the assumptions and verify the validity of equation (12). All simulations below start with an empty flash memory.

First, Assumption 1 is justified. Simulations were run with

selected for garbage collection. Thus, all the valid pages in the beginning of the block cycle have the same probability \( p_{ie} \) to be invalid at the end of the block cycle. The free pages in the beginning of the block cycle are written consecutively and each of them experience one less user write than the previous page. Practical flash memory has a very large number of blocks so that \( T x \gg 1 \). Thus, it is reasonable to assume that each page in a block has the same probability \( p_{ie} \) to be invalid when selected for garbage collection.

IV. SIMULATION

A flash memory simulator was written and simulations were run to justify the assumptions and verify the validity of equation (12). All simulations below start with an empty flash memory.

First, Assumption 1 is justified. Simulations were run with

selected for garbage collection. Thus, all the valid pages in the beginning of the block cycle have the same probability \( p_{ie} \) to be invalid at the end of the block cycle. The free pages in the beginning of the block cycle are written consecutively and each of them experience one less user write than the previous page. Practical flash memory has a very large number of blocks so that \( T x \gg 1 \). Thus, it is reasonable to assume that each page in a block has the same probability \( p_{ie} \) to be invalid when selected for garbage collection.
different combinations of $U$, $N_p$ and $\rho$. Fig. 3 shows that when $\rho = 0.20$ and $N_p = 256$, $X$, the number of invalid pages freed by garbage collection, evolves into a stationary state after a sufficiently large number of garbage collections. It also shows that $X$ converges to the same value, which is close to the analytical result 80.31 given by equation (11), regardless of the value of $U$. This complies with equation (11) that $\bar{x}$ is independent of $U$. Fig. 4 shows that $X$ evolves into a stationary state regardless of the value of $N_p$ and $\rho$ as well when the number of user writes is sufficiently large. Therefore, after a sufficiently large number of user writes, the average value of $X$ always converges to a constant $x$. It also closely approaches the analytical value.

Fig. 4 shows that for different values of $U$, $N_p$ and $\rho$, $T$ is always a good approximation of the number of blocks becoming full in a block cycle. This justifies Assumption 3.

Finally, equation (12) is evaluated and compared to the previous work equation (13). From equation (12), write amplification is independent of $U$ and $N_p$. Thus, comparison is made only for different $\rho$ values. Simulation is run using $N_p = 256$, $U = 1024$, and $\rho$ varies from 0.15 to 1.00. The comparison of results from simulation, this analysis (equation (12)) and previous work (equation (13)) is shown in Fig. 5 and Table I. This shows that, the result of equation (12) is much closer to the simulation value than the result of previous work. For example, when $\rho = 0.30$ which is a commonly adopted value in practical flash memories, equation (12) predicts $A = 2.36$ whereas equation (13) gives $A_{pre} = 2.17$, when the actual value is 2.35.

**REFERENCES**

[1] Ashish Jagmohan, Michele Franceschini, Luis Lastras, “Write amplification reduction in NAND flash through multi-write coding”, in *IEEE 26th Symp. Mass Storage Syst. Technologies (MSST)*, pp. 1-6, 2010.

[2] Rajiv Agarwal and Marcus Marrow, “A closed-form expression for write amplification in NAND flash”, in *IEEE Globecom 2010 workshop on Applicat. of Commun. Theory of Emerging Memory Technologies*, pp. 1908-1912.

[3] X.-Y. Hu, E. Eleftheriou, R. Haas, I. Iliadis, and R. Pletka, “Write amplification analysis in flash-based solid-state drives”, in *Proc. of the ACM SystStore: The Israeli Experimental Syst. Conference*, May 2009.

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

[5] A. Kawaguchi, S. Nishioka, and H. Motoda, “A flash-memory based file system”, in *Proc. USENIX Tech. Conference*, pp. 155-164, Jan. 1995.

[6] M. Rosenblum and J. K. Ousterhout, “The design and implementation of a log-structured file system”, in *ACM Trans. Comput. Syst.*, vol. 10, no. 1, pp. 26-52, Feb. 1992.

[7] 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 Syst. Application*, pp. 119-132. Kluwer Academic Publishers, 1998.

[8] 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 Trans. Embedded Computing Syst.*, vol. 3, no. 4, pp. 837-863, Nov. 2004.

[9] Robert M. Corless, G. H. Gonnet, D. E. G. Hare, D. J. Jeffrey, and D. E. Knuth, “On the Lambert W function”, *Advances in Computational Math.*, vol. 5, no. 4, pp. 329-359, 1996.