WOM Codes Reduce Write Amplification in NAND Flash Memory

Xiang Luojie
Department of Computer Science
Purdue University
West Lafayette, IN 47907, USA
xiang7@purdue.edu

Brian M. Kurkoski
School of Information Science
Japan Advanced Institute of
Science and Technology (JAIST)
Nomi, Japan
kurkoski@ieee.org

Eitan Yaakobi
California Institute of Technology
yaakobi@caltech.edu

***Abstract*—This paper proposes a NAND flash system that uses Write-Once Memory (WOM) codes to encode the data stored. It is shown through both analysis and simulation that, with proper parameters, flash memories which use WOM codes to encode data can achieve a lower write amplification than in a non-WOM-coded system. For example, in a 16-level per cell flash memory, when a two-write MLC WOM code is used with a total overprovisioning of 0.8, the write amplification is 15% lower than a non-WOM-coded system. A closed-form expression for the write amplification in a WOM-coded system is given for a system with a greedy garbage collection policy and a uniform random workload. The proposed expression is a function of the total overprovisioning factor, number of WOM code writes, and number of values per cell. The expression is applicable for both SLC and MLC flash.**

I. Introduction

Recently, flash memories have become widespread, being used in cell phones, memory cards, SSDs, USB disks etc. This is because flash memory has many advantages over traditional hard disk drives, such as lower power consumption, lower read latencies, mechanical reliability, and smaller size [1] [2]. MLC flash memory has gained substantial attention because it has higher density, lower cost per bit and a comparable read speed compared to SLC flash [10] [14]-[16]. In this paper, MLC refers to two or more bits per cell.

Flash memory is organized physically in blocks of fixed size. Each physical block contains a fixed number of pages with fixed size. The page is the unit of reads and writes. All flash memory has an 'erase-before-write' characteristic [3]. Conventionally, this means, once a physical page is written, it must be erased before it can be written again. Flash memory adopts block erase. This makes erasing whenever an update on a physical page is needed extremely inefficient.

To solve this problem, a controller is implemented to present a logical address space over the physical space. When an update on a logical page is issued, the controller writes the data to an empty physical page, then invalidates the old page. This scheme is called out-of-place write. Out-of-place write introduces invalidated pages (referred to as invalid page later). Invalid pages accumulate over time, consuming the physical space without storing data. Garbage collection is then periodically used to change invalid pages into empty pages. The general steps of garbage collection are: first select a physical block, then copy the valid pages into some auxiliary space, then erase the block, and finally copy the valid pages back. The copy operation introduces unwanted physical writes. This phenomenon is referred to as write amplification. Write amplification WA is defined as the average number of physical writes per logical write:

$$WA = \frac{\text{average number of physical writes}}{\text{average number of logical writes}}. \quad (1)$$

Write amplification reduces the lifetime of flash memory and causes significant system overhead.

Write amplification reduces the lifetime because flash memory can tolerate a limited number of program-erase cycles before it becomes unreliable. The latest multi-level cell (MLC) memories can endure around 10^4 program-erase cycles and single-level cell (SLC) memories can endure around 10^5 cycles [1] [2]. Therefore, reducing the number of program-erase cycles is very important for flash memory, especially MLC flash. This means write amplification should be minimized.

To reduce write amplification, a common practice is to allow the user to access only a portion of the raw flash memory space. This is referred to as overprovisioning and the amount of overprovisioning is the ratio between the additional space and the total logical space. It has been shown that write amplification is reduced by increasing the amount of overprovisioning. Previously, most work analyzed the influence of flash system parameters over write amplification, such as overprovisioning, total number of blocks, number of pages per block etc., as well as the influence of usage patterns such as data placement strategy for hot cold data, and access pattern (sequential, random) etc. [4]-[6].

WOM codes are a coding technique that allow multiple updates on a flash memory cell without erases. The original WOM codes, proposed by Rivest and Shamir [8] were binary codes that are suitable for SLC flash. Since then, several recent works studied MLC WOM codes and the capacity of both SLC and MLC WOM code [9] [11]-[13]. Fu and Han Vinck found the WOM capacity for t writes to q-ary storage cells. Jagmohan et al. proposed a multi-write coding scheme to encode the data stored in flash to reduce write amplification

978-1-4673-0921-9/12/$31.00 ©2012 IEEE

[1]. Their work achieved a lower write amplification than a non-coded system. However, they assumed data compression which causes system overhead. In addition, their work was based on simulation rather than analysis.

This paper extends Jagmohan et al's work and examines write amplification when the stored data are encoded with WOM codes. In a non-WOM-coded system, an update results in writing on a free page and an invalid page that is erased later (recall out-of-place write). In a WOM-coded system, however, some updates can be accommodated by eraselessly reprogramming on the old data. This reduces the number of program-erase cycles per logical write, which is called memory wear. Despite the benefit of reducing memory wear, WOM codes can cause an increased write amplification in SLC flash memory reducing its contribution.

This paper shows through both analysis and simulation that, with proper system parameters, applying WOM codes can achieve a lower write amplification than non-WOM-coded flash memory. For a 16-level-cell flash memory, when a two-write WOM code is used with a total overprovisioning of 0.8, the write amplification is 15% lower than that of a non-WOM-coded system. This paper also gives a closed-form expression for the write amplification when WOM codes are used. The closed-form expression is a function of the overprovisioning factor ρ_{total}, the number of WOM code writes t, and the number of values per physical bit q. This work does not use data compression.

II. System Model

A. Capacity-Achieving WOM Code

For a t-write WOM code, the rate on the i-th write R_i bits/cell, for $i = 1, 2, \ldots, t$. Then, the capacity is the upper bound on the achievable sum rate [6]:

$$R_1 + R_2 + \cdots + R_t \leq \log_2 \binom{q+t-1}{t}, \tag{2}$$

where q is the number of levels stored in a cell. A penalty expansion factor indicates the additional memory required for the WOM code:

$$r \geq \frac{t \log_2 q}{\log_2 \binom{q+t-1}{t}}. \tag{3}$$

It is possible to achieve equality in Eq. (2) only when all individual rates on each write are different. This paper uses WOM codes with equal rates. So, we use Eq. (3) in order to form a lower bound on the write amplification. The actual construction scheme of WOM codes is beyond the scope of this paper. The write amplification value while using specific WOM code constructions remains an open problem.

B. WOM-Coded System Model

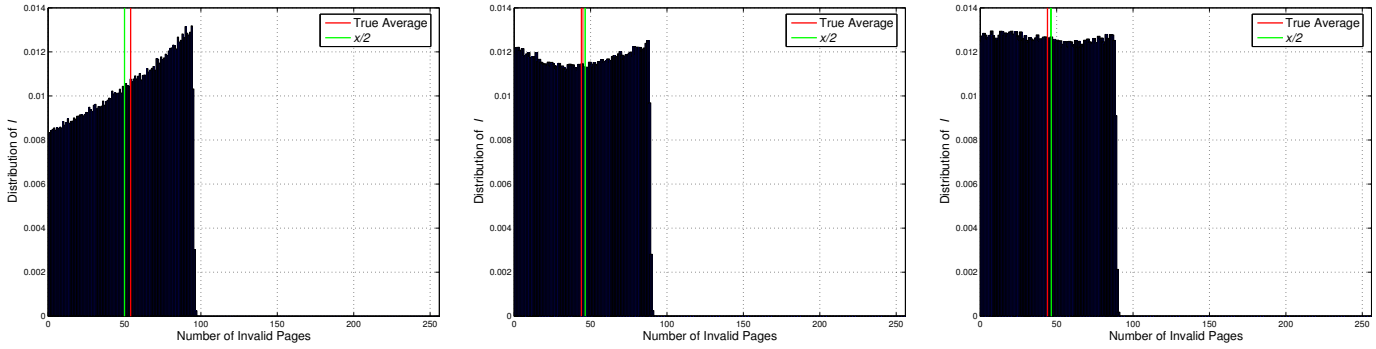
This paper models the physical flash memory as a block pool containing T blocks. Each block has a fixed number of N_{p} pages. The logical memory space is modeled as a continuous space with UN_{p} pages, where each logical page has a fixed size S_{p}. WOM codes are used to encode the data stored in the physical space. In order to maintain the physical page alignment, the size of a physical page is assumed to be the same as a WOM code word. Because the WOM code has an expansion factor r, the size of a physical page is $S_{\mathsf{p}}r$. A physical page has $t + 2$ states. A free page, denoted by state 0, is a physical page that has not been written and can accommodate a new write. A valid page is a physical page that has been written and stores valid data. Because a t-write WOM code supports t writes on a physical page, a valid page has t states from state 1 to state t. An invalid page, denoted by state $t + 1$, is a page that has been marked invalid due to an out-of-place write.

The controller handles the physical details of the flash memory and presents a logical space similar to hard disk drive to the operating system. It implements the WOM encoder and maintains metadata. The WOM encoder encodes the data of each write request into a WOM codeword. The metadata includes a LPA-PPA (logical page address - physical page address) mapping table which maps logical page addresses into physical page addresses. Another table is maintained in the metadata to store the current state of each physical page $(0, 1, \ldots, t+1)$ which indicates the number of writes available. Once there is a change in the LPA-PPA mapping table or the state of a physical page, the controller updates the metadata accordingly.

This paper assumes a random uniform workload on the logical space. This means the write requests are randomly and independently distributed on the logical space. We are interested in this scenario as the write amplification has the worst performance under this random workload.

The following describes the operation of the system. Initially, the logical space is empty. Write requests come to the logical space in a random manner. A new write is referred to as the case where a write request addresses an empty logical page. The controller finds a free physical page using the metadata, encodes the data into a WOM codeword and writes the codeword into that physical page. The LPA-PPA mapping table is updated accordingly. If a write request addresses a written logical page, this is called an update. Upon an update, the controller looks up the corresponding physical page of the addressed logical page in the LPA-PPA mapping table. The controller checks the state of this physical page. If it is smaller than t, the physical page can still be eraselessly reprogrammed (in the case of t-write WOM code). The controller encodes the data into a WOM codeword, eraselessly reprograms this codeword into that physical page and adds one to the state of this physical page in the metadata. If this physical page is at state t, an out-of-place write takes place. The controller marks that physical page as invalid (state $t+1$), finds another free physical page, writes the encoded codeword into that free physical page and updates the LPA-PPA table.

When there are no more free pages for a requested update, garbage collection is performed. This work assumes a greedy garbage collection policy. That is, garbage collection is triggered only when there are no free pages in the physical space. The block with the maximum number of invalid pages



(a) $t = 1$ (b) $t = 2$ (c) $t = 3$

Fig. 1: Distribution of number of invalid pages in a random block I

is selected for garbage collection. Once a block is selected for garbage collection, all valid pages in this block are copied out to some auxiliary space. The whole block is erased and becomes free. The valid pages are then copied back and finally the LPA-PPA mapping table is updated. We note that we use the greedy garbage collection as it was proved to be optimal under random workload [6]. Wear leveling algorithms are not used. Therefore write amplification is only a result of the copy operation in the garbage collection. We also note that we are aware that the page writes in flash memory blocks should be performed sequentially due to the restrictions imposed by flash memory chips. In our model, it may happen that page updates are not written sequentially in a block. It is possible to guarantee this sequential write property. However, we leave this for a future work, while in this work we allow non-sequential writes in a block as was done in [1].

In a non-WOM-coded system, the traditional overprovisioning comes from the invalid pages. In a WOM-coded system, WOM code causes data expansion by a factor of r and requires more overprovisioning. In order to determine the expression for both levels of overprovisioning, we propose an imaginary memory space called 'Apparent Memory' which lies between the logical and physical memory. The page size of the apparent memory is the same as that of logical memory S_{p}, and it has the same block and page organization with the physical memory. The only difference between the apparent memory and physical memory is that the physical page size is larger by a factor of r due to the WOM code. The size of the logical memory is:

$$S_{\mathsf{logical}} = UN_{\mathsf{p}}S_{\mathsf{p}}. \tag{4}$$

The size of the apparent memory is:

$$S_{\mathsf{apparent}} = TN_{\mathsf{p}}S_{\mathsf{p}}. \tag{5}$$

The size of the physical memory is:

$$S_{\mathsf{physical}} = TN_{\mathsf{p}}S_{\mathsf{p}}r. \tag{6}$$

Therefore, the traditional overprovisioning factor is:

$$\rho = \frac{S_{\mathsf{apparent}} - S_{\mathsf{logical}}}{S_{\mathsf{logical}}} = \frac{TN_{\mathsf{p}}S_{\mathsf{p}} - UN_{\mathsf{p}}S_{\mathsf{p}}}{UN_{\mathsf{p}}S_{\mathsf{p}}} = \frac{T-U}{U}. \tag{7}$$

The total overprovisioning factor is:

$$\rho_{\mathsf{total}} = \frac{S_{\mathsf{physical}} - S_{\mathsf{logical}}}{S_{\mathsf{logical}}} = \frac{TN_{\mathsf{p}}S_{\mathsf{p}}r - UN_{\mathsf{p}}S_{\mathsf{p}}}{UN_{\mathsf{p}}S_{\mathsf{p}}} = \frac{Tr-U}{U}. \tag{8}$$

The two overprovisioning factors are related by:

$$\rho = \frac{\rho_{\mathsf{total}} + 1}{r} - 1. \tag{9}$$

This work applies for both SLC and MLC flash memory. The distinction is that, q-level MLC WOM codes are used for q-level MLC flash.

III. Write Amplification Analysis

This section derives an expression for the write amplification in a WOM-coded system. There are two techniques that were developed for non-WOM coded systems. The first was proposed by Agarwal et al [2]. They expressed the average number of invalid pages in a random physical block in two ways and obtained the expression for write amplification by setting these two expressions equal. The second technique finds two ways to express the number of invalid pages in the block selected for garbage collection, and derived the expression for write amplification by setting these two equations equal [7],

$$WA = \frac{1 + \rho_{\mathsf{total}}}{1 + \rho_{\mathsf{total}} + \mathsf{W}((-1-\rho_{\mathsf{total}})e^{-1-\rho_{\mathsf{total}}})}, \tag{10}$$

where $\mathsf{W}(x)$ is the Lambert W Function. This technique has also been used in the scenario where TRIM commands are used [17]. In a non-WOM coded system, this technique gives a more accurate expression. Accordingly, Eq. (10) will be used for the write amplification in a non-WOM-coded system. In this analysis, for WOM-coded system however, Agarwal et al's technique will be used because the latter technique cannot find a closed-form solution for write amplification in WOM-coded system.

It is shown later that write amplification depends on the number of invalid pages in the block selected for garbage collection, denoted x. Therefore, an expression for x must be found in order to obtain the expression for write amplification.

Two methods are used to find the expression for the average number of invalid pages in a block over the physical space denoted a_{p}. The expression for x is obtained by setting these two expressions equal.

Let I denote the number of invalid pages in a randomly selected block in the physical space. Extensive simulation shows that, after a sufficiently large number of logical writes this distribution becomes stationary [2]. Another interesting point is that, the use of WOM codes changes the distribution of I. Fig. 1a shows this distribution in a non-WOM-coded system, namely $t = 1$ where $U = 1280,\ \rho = 0.25,\ N_{\mathsf{p}} = 256$. Simulation shows that, when $t \geq 2$, the distribution of I appears roughly symmetric. Fig. 1b and Fig. 1c show the distribution of I when $U = 1280,\ N_{\mathsf{p}} = 256,\ q = 1024$ with $t = 2,\ \rho_{\mathsf{total}} = 0.3$ and $t = 3,\ \rho_{\mathsf{total}} = 0.4$ respectively. The minimum value of the distribution is 0 and the maximum value is x, the number of invalid pages in the block selected for garbage collection (recall the greedy garbage collection policy). The first way to find the expression for a_{p} is to analyze the distribution of I.

When the distribution is symmetric, the following is a good approximation:

$$\mathsf{E}(I) = \frac{x}{2}. \tag{11}$$

The true average $\mathsf{E}(I)$ and the approximation $\frac{x}{2}$ are shown in Fig. 1. Agarwal et al. approximated the distribution of I when $t = 1$ (Fig. 1a) as a uniform distribution. Clearly, this approach achieves higher accuracy when $t \geq 2$, that is, in WOM-coded system shown in Fig. 1b, 1c. This justifies using Agarwal's approach in this analysis.

The second way to obtain an expression for a_{p} is to average the total number of invalid pages over the number of blocks. After a sufficiently large number of logical writes, there are UN_{p} valid pages in the physical space and therefore $(T - U)N_{\mathsf{p}}$ invalid pages. Referring to Eq. (7), let p denote the proportion of invalid pages in the entire physical space:

$$p = \frac{T - U}{T} = \frac{\rho}{1 + \rho}. \tag{12}$$

The average number of invalid pages in a block is then:

$$a_{\mathsf{p}} = N_{\mathsf{p}} p = N_{\mathsf{p}} \frac{\rho}{1 + \rho}. \tag{13}$$

Setting Eq. (11) and Eq. (13), the two expressions for a_{p}, to be equal,

$$x = N_{\mathsf{p}} \frac{2\rho}{1 + \rho}. \tag{14}$$

Notice that $x < N_{\mathsf{p}}$. Substituting Eq. (14) into this equation yields, $\rho < 1$. It should also hold that $\rho > 0$ (refer to Eq. (7)). Therefore, Eq. (14) is valid only when

$$0 < \rho < 1. \tag{15}$$

The following derives the expression for write amplification. From the definition of write amplification Eq. (1), the write amplification for the $t \geq 2$ WOM-coded system is approximated by:

$$WA = \frac{N_{\mathsf{p}} + (t - 1)x}{tx}, \tag{16}$$

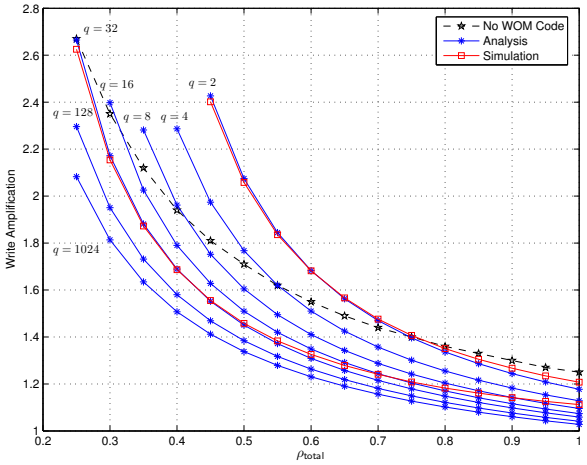


Fig. 2: Simulation results for different q when t=2.

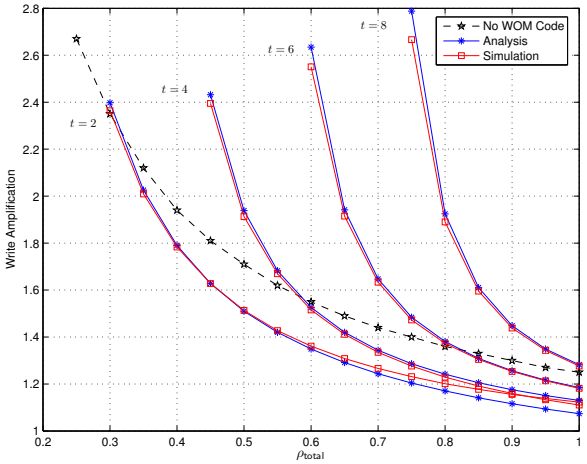


Fig. 3: Comparison of simulation and analysis for different t when q=16.

which is demonstrated in Appendix. Substituting Eq. (14) the expression for x into Eq. (16),

$$WA = \frac{2t\rho - \rho + 1}{2t\rho}, \quad (t \geq 2). \tag{17}$$

Substitute ρ_{total} for ρ using Eq. (9),

$$WA = \frac{1}{2t}(2t - 1 + \frac{r}{\rho_{\mathsf{total}} + 1 - r}), \quad (t \geq 2). \tag{18}$$

Substitute ρ_{total} into Eq. (15) using Eq. (9), the condition under which Eq. (18) holds is,

$$\frac{r}{\rho_{\mathsf{total}} + 1 - r} > 1. \tag{19}$$

Using the equality of Eq. (3), the expansion penalty factor of capacity-achieving t-write WOM code, into Eq. (18), the lower bound of write amplification is

$$WA \geq \frac{1}{2t}(2t - 1 + \frac{t}{(\rho_{\mathsf{total}} + 1)\log_q \binom{q+t-1}{t} - t}), \quad (t \geq 2). \tag{20}$$

Substitute Eq. (3) into Eq. (19), Eq. (20) is valid only when

$$\frac{t}{(\rho_{\mathsf{total}}+1)\log_q\binom{q+t-1}{t}-t} > 1. \tag{21}$$

From Eq. (20) we can see that the write amplification decreases when ρ_{total} increases. It is easily shown that write amplification decreases when q increases. Therefore, from the point view of reducing write amplification, ρ_{total} and q should be as large as possible, within the limits of practical constraints. The influence of t over write amplification is not obvious. Through evaluation of Eq. (20) for practical q ($q \leq 16$), $t = 2$ yields the minimal write amplification. This is not true when q is large. For example, when $q = 128$ and $\rho_{\mathsf{total}} = 0.5$, $t = 3$ yields the best result.

IV. Simulation and Discussion

Simulations are run to show the validity of Eq. (20), shown in Fig. 2, 3. Clearly, Eq. (20) gives an accurate prediction to the simulation value. Fig. 2 is obtained when $U = 1024$, $t = 2$ with varying q and ρ_{total}. r is determined by Eq. (3). Fig. 2 shows that the write amplification decreases as ρ_{total} and q increases. It can be seen that when q is small, considerable reduction on the write amplification can be achieved by increasing q. However, the influence of q over the write amplification becomes small when q is large enough. This figure also shows that when q is large enough, write amplification can be smaller over all ρ_{total} values than that without WOM code. Fig. 3 is obtained when $U = 1024$, $q = 16$ with varying t and ρ_{total}. r is determined by Eq. (3). We can see that when $q = 16$, a lower write amplification is achieved with smaller t. The write amplification becomes large very quickly as t grows. Applying two-write WOM code results in a better write amplification than that without WOM code when $\rho_{\mathsf{total}} > 0.3$.

At this point, it is clear that applying WOM code may achieve a lower write amplification than that without WOM code. One interesting point is to find out the condition under which applying WOM code can achieve a lower write amplification. Setting the expression for write amplification of a WOM-coded system Eq. (20) and a non-WOM-coded system Eq. (10) to be equal,

$$\begin{aligned}&\frac{1}{2t}\left(2t-1+\frac{t}{(\rho_{\mathsf{total}}+1)\log_q\binom{q+t-1}{t}-t}\right)\\ =\;&\frac{1+\rho_{\mathsf{total}}}{1+\rho_{\mathsf{total}}+\mathsf{W}((-1-\rho_{\mathsf{total}})e^{-1-\rho_{\mathsf{total}}})}.\end{aligned} \tag{22}$$

Obviously this equation is hard to solve. However, it can be used for numeric evaluation to find out when applying WOM code can achieve a lower write amplification. Values of ρ_{total} where the WOM-coded system achieves the same write amplification as the non-WOM-coded system is shown in Fig. 4. The parameter points $(q, \rho_{\mathsf{total}})$ for a specific t below the corresponding curve means a lower write amplification without WOM code. The points above means applying WOM codes yields better results.

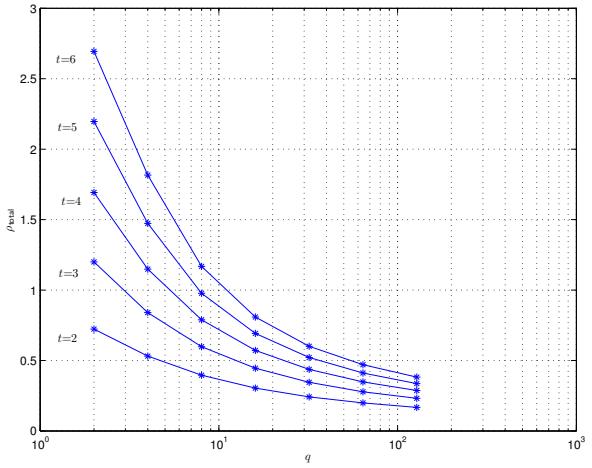


Fig. 4: Values of ρ_{total} for which the WOM coded and non-WOM-coded system have the same write amplification.

For a 16-level-cell flash memory [10], when two-write WOM code is used with a total overprovisioning of 0.8, a write amplification of 1.1704 can be achieved, whereas the write amplification without WOM code is 1.3653.

Appendix

Write amplification is defined as the average number of physical writes per logical write. The following determines the average number of physical writes and logical writes between two garbage collections respectively. Between two garbage collections, x invalid pages are collected, which means, $N_{\mathsf{p}}-x$ valid pages are copied out and back (recall the general steps of garbage collection). This results in $N_{\mathsf{p}} - x$ physical writes. It is shown in Lemma 1 that, between two garbage collections, there are an average of tx logical writes. These tx logical writes corresponds to tx physical writes. Therefore, there are $N_{\mathsf{p}} - x + tx = N_{\mathsf{p}} + (t-1)x$ physical writes and tx logical writes between two garbage collections. Write amplification should be

$$WA = \frac{N_{\mathsf{p}} + (t-1)x}{tx}. \tag{23}$$

Lemma 1: There are tx logical writes between two garbage collections.

The following proves Lemma 1. When t-write WOM code is used, one physical page has $t+2$ levels. Denote page state i as V_i ($0 \leq i \leq t+1$). To prove Lemma 1, the following assumption is needed.

Assumption 1: The number of V_i pages on the entire physical space after a garbage collection V_{E}^i converge to a constant after a sufficiently large number of logical writes, $\mathsf{E}(V_{\mathsf{E}}^i) = v_{\mathsf{E}}^i$.

This is verified by simulation, Fig. 5, which shows the convergence of V_{E}^i $(i = 0, 1, ..., 5)$ when $U_{\mathsf{R}} = 1024$, $\rho_{\mathsf{total}} = 1$, $q = 2$, $t = 4$ and r is determined by Eq. (3).

Consider two adjacent garbage collections. Suppose the total number of logical writes between these two garbage collections is t_{u}. Assume all these logical writes fall on different pages because the probability of one logical page

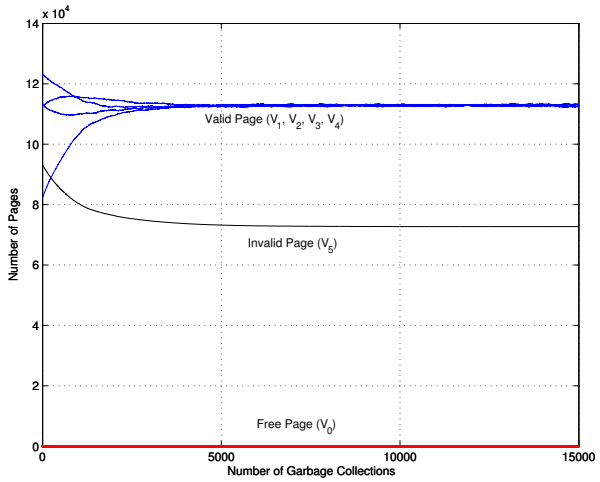


Fig. 5: Convergence of V_{E}^1 and V_{E}^2 to constants after sufficiently large number of user writes

being updated more than once between two adjacent garbage collections is estimated to be at the magnitude of 10^{-7} for a small flash of 128 MB. This probability becomes smaller as the size of the flash grows larger.

Right after the first of the two adjacent garbage collections, there are x free (V_0) pages and v_{E}^i number of V_i pages in the physical space, where $i = 1, 2..., t+1$. Before the next garbage collection, some logical pages are updated causing their corresponding valid pages (V_i pages where $1 \leq i \leq t$) updated into V_{i+1} pages. Because the total number of logical pages is $\sum_{k=1}^{t} v_{\mathsf{E}}^k$, which equals the total number of valid pages in the physical space, and there are v_{E}^i ($1 \leq i \leq t$) logical pages corresponding to V_i ($1 \leq i \leq t$) valid pages in the physical space, the probability of a V_i ($1 \leq i \leq t$) valid page being updated between two garbage collections is

$$\frac{v_{\mathsf{E}}^i}{\sum_{k=1}^{t} v_{\mathsf{E}}^k}, \tag{24}$$

due to the randomly distributed nature of logical writes. There are t_{u} logical writes between two garbage collections. Therefore, between the two garbage collections, the number of V_i ($1 \leq i \leq t$) pages updated into V_{i+1} pages is

$$\frac{v_{\mathsf{E}}^i}{\sum_{k=1}^{t} v_{\mathsf{E}}^k} t_{\mathsf{u}}. \tag{25}$$

At the same time, the number of V_{i-1} pages updated into V_i ($2 \leq i \leq t+1$) pages between the two garbage collections is

$$\frac{v_{\mathsf{E}}^{i-1}}{\sum_{k=1}^{t} v_{\mathsf{E}}^k} t_{\mathsf{u}}. \tag{26}$$

According to Assumption 1, v_{E}^i remains constant. Therefore, setting Eq. (26), the number of V_{i-1} pages becoming V_i pages and Eq. (25), the number of V_i pages becoming V_{i+1} pages to be equal

$$\frac{v_{\mathsf{E}}^i}{\sum_{k=1}^{t} v_{\mathsf{E}}^k} t_{\mathsf{u}} = \frac{v_{\mathsf{E}}^{i-1}}{\sum_{k=1}^{t} v_{\mathsf{E}}^k} t_{\mathsf{u}}. \tag{27}$$

That is

$$v_{\mathsf{E}}^i = v_{\mathsf{E}}^{i-1}, \quad (i = 2, ..., t). \tag{28}$$

Fig. 5 illustrates this when $t = 4$. Between the two garbage collections, $\frac{v_{\mathsf{E}}^t}{\sum_{k=1}^{t} v_{\mathsf{E}}^k} t_{\mathsf{u}}$ V_t pages are updated into V_{t+1} namely invalid pages. The new data of these updates are written to the x free pages. Therefore, the number of V_t pages updated is equal to the number of free pages x.

$$\frac{v_{\mathsf{E}}^t}{\sum_{k=1}^{t} v_{\mathsf{E}}^k} t_{\mathsf{u}} = x. \tag{29}$$

Because of Eq. (28),

$$\frac{v_{\mathsf{E}}^t}{\sum_{k=1}^{t} v_{\mathsf{E}}^k} = \frac{1}{t}. \tag{30}$$

Substitute Eq. (30) into Eq. (29),

$$t_{\mathsf{u}} = tx \tag{31}$$

This completes the proof.

REFERENCES

[1] A. Jagmohan et al., "Write amplification reduction in NAND flash through multi-write coding", in *IEEE 26th Symp. Mass Storage Syst. Technologies (MSST)*, 2010, pp. 1-6.

[2] R. Agarwal and M. Marrow, "A closed-form expression for write amplification in NAND flash", in *IEEE Globecom 2010 workshop Applicat. Commun. Theory Emerging Memory Technologies*, 2010, pp. 1908-1912.

[3] T.-S. Chung et al., "A survey of Flash Translation Layer", *J. Syst. Archit*, vol. 55, pp. 332-343, May, 2009.

[4] W. Bux, "Performance evaluation of the write operation in flash-based solid-state drives", IBM Research, Zurich, Rschlikon, Rep. RZ3757, 2009.

[5] X.-Y. Hu et al., "Write amplification analysis in flash-based solid-state drives", in *Proc. ACM SysStor: Israeli Experimental Syst. Conf.*, May 2009.

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

[7] L.J. Xiang and B. M. Kurkoski, "An improved analytical expression for write amplification in NAND flash", in *Proc. Int. Conf. Computing, Networking Commun.*, Maui, Hawaii, USA, Jan. 2012.

[8] R. L. Rivest and A. Shamir, "How to reuse a write - once memory", in *Proc. ACM STOC*, 1982, pp. 105-113.

[9] F.-W. Fu and A. J. Han Vinck, "On the capacity of generalized write-once memory with state transitions described by an arbitrary directed acyclic graph", *IEEE Trans. Inf. Theory*, vol. 45, no. 1, Jan. 1999.

[10] N. Shibata et al. "A 70 nm 16 gb 16-Level-Cell NAND ash memory", *IEEE J. Solid-State Circuits*, vol. 43, no. 4, pp. 929-937, 2008.

[11] R. Gabrys et al., "Non-binary WOM-codes for multilevel flash memories", in *Inform. Theory Workshop*, pp. 40-44, Oct. 2011.

[12] S. Kayser et al., "Multiple-write WOM-codes", in *Proc. of 2010 48th Annual Allerton Conf. Commun., Control, Computing* , Allerton, pp. 1062-1068, Otc. 2010.

[13] R. Gabrys and L. Dolecek, "Characterizing capacity achieving write once memory codes for multilevel flash memories", in *Proc. 2011 IEEE Int. Symp. Inform. Theory*, pp. 2517-2521, Jul. 2011.

[14] Y. Li et al. "A 16 Gb 3-bit per cell (X3) NAND flash memory on 56 nm technology with 8 MB/s write rate", *IEEE J. Solid-State Circuits*, vol. 44, no. 1, pp. 195-207, Jan. 2009.

[15] T. Futatsuyama et al., "A 113mm2 32Gb 3b/cell NAND flash memory", in *IEEE Int. Solid-State Circuits Conf.*, pp. 242-243, Feb. 2009.

[16] C. Trinh and N. Shibata et al., "A 5.6MB/s 64Gb 4b/cell NAND flash memory in 43nm CMOS", in *IEEE Int. Solid-State Circuits Conf.*, pp. 246-247, Feb. 2009.

[17] T. Frankie et al., "The effect of TRIM requests on write amplification in solid state drives", in *Seventh Int. Conf. Commun., Internet, Inform. Technology*, Baltimore, MD, USA, May, 2012.