ON THE DISTRIBUTION OF VALID PAGES WITH GREEDY GARBAGE COLLECTION FOR NAND FLASH

Borja Peleato, Rajiv Agarwal and John Cioffi

Electrical Engineering Department
Stanford University, CA 94305-9510

ABSTRACT

Flash memory uses relocate-on-write, also called out-of-place write for performance reasons. Relocate-on-write requires a garbage-collection process, which results in additional write operations referred to as write-amplification. The reclaiming policy that selects the blocks to garbage-collect is greedy if it is based only on the amount of free space to be gained-Write amplification is a critical factor limiting the write performance and endurance in solid-state drives and the greedy garbage collection policy minimizes it. In this paper, we derive an expression for the steady-state distribution of valid-pages using a novel probabilistic model for flash-based SSDs for a greedy reclaiming policy. This distribution lends itself directly to evaluation of write-amplification and gives insight into designing newer garbage collection policies, which would further reduce write-amplification. Numerical evaluation of the derived expression matches the distribution obtained via Monte-Carlo simulation for several different drive configurations, thereby validating the derivation.

1. INTRODUCTION

Recently, flash memories have emerged as a faster and more efficient alternative to hard drives. The introduction of Solid State Devices (SSD) based on Flash NAND memories has revolutionized mobile, laptop, and enterprise storage, among others. The main difference between flash memories and hard drives is that flash memories offer random access to the information, without any moving parts. SSDs significantly reduce power consumption and dramatically improve robustness and shock resistance.

NAND-flash memories have unique characteristics that pose challenges to the SSD system design, especially the aspects of random write performance and write endurance. They are organized in terms of blocks, each block consisting of a fixed number of pages. A block is the elementary unit for erase operations, whereas reads and writes are processed in terms of pages. Before data can be written to a page, the block must have been erased. Moreover, NAND flash memories have a limited program-erase (PE) cycle count, or equivalently there is a limit to the number of times information can be re-written. Typically, flash chips based on single-level cells (SLC) sustain 10^5 and those based on multi-level cells (MLC) 10^4 PE cycles [1].

Flash memory uses relocate-on-write, also called out-of-place write, mainly for performance reasons: If write-in-place is used instead, flash will exhibit high latency due to the necessary reading, erasing, and re-programming of the entire block in which data is being updated. However, relocate-on-write requires a garbage-collection process, which results in additional read and 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, the policy defined in [2] also included the time elapsed since the last writing of the block with data. In general, the objective is to minimize the number of valid pages in the blocks selected for garbage collection, thereby minimizing the number of read and write operations resulting from garbage collection.

In contrast to disks, flash memory blocks eventually wear out with progressing number of PE cycles until they can no longer be written. Wear-leveling techniques are therefore used to maximize endurance in terms of host writes served. In flash, write-amplification corresponds to the additional writes caused by garbage collection and wear-leveling. Hence, the total number of host 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. Finally, the management of out-of-place updates involves a mapping between logical block addresses (LBA), i.e., the host address space, and physical block addresses (PBA). This mapping may be used to distinguish dynamic from static data.

Another aspect of interest for NAND flash memories is whether the incoming host data is hot or cold. Cold data comprises of files such as the OS, which once stored is not deleted for a long time, whereas hot data comprises of user's work files such as log files which are updated/deleted on a regular basis. Hence, blocks storing cold data tend to not have their PE cycle count increased at all, whereas blocks storing hot data experience a continuous increase in their PE cycle count. In the absence of cold data, with workloads of uniformly-distributed random short writes, the entire flash memory is evenly worn. Furthermore, a greedy garbage collection policy will be optimal for this usage scenario. This policy ensures that, when garbage collecting to accommodate for a new host write request, blocks with the most 'amount of free space to be gained' or equivalently 'least number of valid pages' are selected, thereby minimizing write-amplification.

This paper studies the number of valid pages per block in a flash drive with greedy garbage collection. This problem has been studied in [3], where the authors present an exact Markov chain model for small systems and an approximate one for larger systems. The exact Markov chain model could not be extended to larger systems due to an exponential growth in the number of states with the number of pages per block. In [4], the authors analyze the write-amplification resulting from variants of the greedy garbage collection policy and how it could be improved by classifying the data into hot and cold. Another study was done in [5], where it was proven that in the asymptotic case, write-amplification depends exclusively on over-provisioning, and is independent of drive size and number of pages per block.

In this paper, a novel approach is taken to derive an expression for the steady-state distribution of valid pages in a block. Specifically, we derive an expression for the evolution of the distribution of valid pages over time and solve it to obtain a fixed-point solution in the steady state. The rest of the paper is organized as follows. Section 2 describes the model used for the analysis and introduces the notation used throughout the paper. Section 3 derives a simple expression for the expected number of blocks with a given number of valid pages, once the drive has reached its steady state. Section 4 compares the previous expression with the results obtained through simulations. The paper concludes with Section 5 which summarizes the paper and proposes directions for further work.

2. SYSTEM MODEL

Let B denote the total number of blocks in the drive and N the number of pages per block. In practice, the host can only use a part of that raw capacity, say W blocks, where $W \leq B$. Then the over-provisioning factor ϵ is defined as $\epsilon = \frac{B-W}{W}$. So, for a drive with $\epsilon = 0.25$ there is 25% over-provisioning and only 80% of the drive is available for host data. Equivalently, the drive is considered full when it has WN valid pages and $(B-W)N$ invalid pages.

When the drive is full and a new write request arrives, garbage collection must be done. In this paper greedy garbage collection is used, which selects the block with the least number of valid pages to be collected. Once a block has been selected to be reclaimed, all valid pages in that block are relocated into a new block. The selected block can then be erased and all N pages on that block become available to accommodate new data. The efficiency of garbage collection is measured by write-amplification, defined as the actual number of page writes per host page write. Specifically, suppose that c valid pages were relocated from the block that was garbage collected, then this block can accommodate $N-c$ pages of new data. In this instant write-amplification $\mu = \frac{N}{N-c} = 1 + \frac{c}{N-c} \geq 1$. A greater-than-1 write-amplification means that each host-page-write causes extra drive-page-writes due to relocation of valid pages. The garbage collection policy should try to minimize write amplification. The greedy policy, that always picks the block with least number of valid pages, achieves that.

Under normal usage of the drive, the over-provisioning ϵ is always kept constant. Once the drive is full, i.e. there are $U = WN$ valid pages, garbage collection is done to accommodate new host data, say R pages. This must be followed by deletion/invalidation of the same amount of existing host data, i.e. R valid pages, to maintain over-provisioning. Assuming workloads of uniformly-distributed random short writes, invalidation of existing host data amounts to deleting R randomly chosen valid pages in the drive.

3. ANALYSIS

Let $x^{(t)} \in \mathbf{Z}^{N+1}$ be a vector representing the distribution of valid pages in the blocks at time t, so that $x^{(t)}(0)$ denotes the number of blocks without any valid pages and $x^{(t)}(k)$ the number of blocks with k valid pages at time t. This section derives an explicit expression for $E[x^{(\infty)}]$, the expected value of $x^{(t)}$ as t approaches infinity, as a function of N, B, and U. First, an expression to transition from $E[x^{(t)}]$ to $E[x^{(t+1)}]$ will be found. Then, t will be assumed large enough for the drive to have reached its steady state, in which $E[x^{(t)}] = E[x^{(t+1)}] = E[x^{(\infty)}]$. The system transitions from $x^{(t)}$ to $x^{(t+1)}$ in two steps:

1. *Garbage collection*: The block with the minimum number of valid pages gets collected, erased, and refilled with new data. Let c be the number of valid pages in the garbage collected block. This step decreases the number of blocks with c valid pages by one and increases the number of blocks with N valid pages by one.
2. *Invalidation*: $(N-c)$ valid pages, chosen randomly across all the blocks, get invalidated to maintain over-provisioning.

After the garbage collection step there are $(U+N-c)$ valid pages in the drive, $(N-c)$ of which need to be invalidated. Let $p_{ij}(c)$ denote the probability that a block with i valid pages has $(i-j)$ of them invalidated, where $(i-j) \leq (N-c)$. An explicit expression for p_{ij} can be found as follows.

Let the $(N-c)$ pages be invalidated sequentially. The probability that the first invalidated page belongs to a block with i valid pages is $\frac{i}{U+N-c}$. After the first page has been invalidated there are $U+N-(c+1)$ valid pages left in the drive and $N-(c+1)$ of them need to be invalidated. Therefore, a recursive equation for $p_{ij}(c)$ can be written as:

$$p_{ij}(c) = \frac{i}{U+N-c} p_{(i-1)j}(c+1) + \frac{U+N-c-i}{U+N-c} p_{ij}(c+1).$$

This recursion can be expressed in matrix notation as $P(c) = A(c)P(c+1)$ where

$$P(c) = \begin{bmatrix} p_{00}(c) & 0 & \cdots & 0 \\ p_{10}(c) & p_{11}(c) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ p_{N0}(c) & p_{N1}(c) & \cdots & p_{NN}(c) \end{bmatrix},$$

$$A(c) = \frac{1}{T}\begin{bmatrix} T & 0 & \cdots & 0 \\ 1 & T-1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & N & T-N \end{bmatrix}$$

and $T = U + N - c$. The terms above the diagonal of $P(c)$ are zero because the number of valid pages cannot increase (hence $p_{ij} = 0$ for $j > i$). The transition matrix $A(c)$ can be decomposed as $A(c) = V^{-1}D(c)V$ where

$$(V)_{ij} = \begin{cases} 0 & j > i \\ (-1)^{i+j}\begin{pmatrix} i-1 \\ j-1 \end{pmatrix} & j \leq i \end{cases}$$

$$D(c) = \frac{1}{T}\begin{bmatrix} T & 0 & \cdots & 0 \\ 0 & T-1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & T-N \end{bmatrix}$$

and V^{-1} can be found by taking the element-wise absolute value of V. Since V and V^{-1} do not depend on c and $D(c)$ is diagonal, it is possible to repeatedly apply the equation $P(c) = V^{-1}D(c)VP(c+1)$ to obtain $P(c) = V^{-1}\left(\prod_{k=c}^{N-1} D(k)\right)VP(N)$. If the garbage collected block has N valid pages no invalidation happens, hence $P(N)$ is the identity matrix. Consequently,

$$P(c) = V^{-1}\left(\prod_{k=c}^{N-1} D(k)\right)V.$$

The expected number of blocks with k valid pages at time $(t+1)$, denoted by $E[x^{(t+1)}(k)]$, can be found using the method of indicators. An indicator variable s_b is defined for each block b, where $b = 1, \ldots, B$. The indicator variable s_b takes value 1 if block b has k valid pages after the invalidation step, and takes value 0 otherwise. The expected number of blocks with k valid pages at time $(t+1)$ is then given by $E[x^{(t+1)}(k)] = \sum_{b=1}^{B} E[s_b]$ where the expectation is taken with respect to c and the pages that get invalidated. Using iterated expectation,

$$\begin{aligned} E[x^{(t+1)}(k)] &= \sum_{j=1}^{B} E_c[E[s_j|c]] \qquad (1) \\ &= \sum_{j=k}^{N} E_c\big[p_{jk}(c)(E[x^{(t)}(j)|c] \\ &\quad +\delta(j-N) - \delta(j-c-1))\big], \qquad (2) \end{aligned}$$

where $\delta(n) = 1$ if $n = 0$ and $\delta(n) = 0$ otherwise. Equation (2) results from grouping the terms in Equation (1) by the number of valid pages in each block and including the contribution of the garbage collection step. The expectation with respect to c is very hard to compute but [3, 5] showed that, once the memory has reached its steady state, c is approximately constant. If a deterministic value c^∞ is assumed for c, the probabilities $p_{ji}(c)$ also become deterministic for all i and j. Therefore,

$$E[x^{(t+1)}(k)] = \sum_{j=k}^{N} p_{jk}(c^\infty)(E[x^{(t)}(j)] + \delta(j-N) - \delta(j-c^\infty))$$

or equivalently $E[x^{(t+1)}] = P^T(c^\infty)(E[x^{(t)}] + e_N - e_{c^\infty})$, where e_k denotes the vector which has zeros in all components except the k-th, in which it has a 1.

Considering the system in steady state, that is $E[x^{(t+1)}] = E[x^{(t)}] = E[x^{(\infty)}]$ and $c = c^\infty$, it holds that $x^{(\infty)}(k) = 0$ for $k < c^\infty$. It is then possible to discard the first c^∞ components of $x^{(\infty)}$ as well as the first c^∞ rows and columns of $P(c^\infty)$ obtaining the reduced system

$$E[\tilde{x}] = \left(\tilde{V}^{-1}\tilde{D}\tilde{V}\right)^T (E[\tilde{x}] + e_{(N-c^\infty+1)} - e_1), \qquad (3)$$

where

$$\tilde{x} = \begin{bmatrix} x^{(\infty)}(c^\infty) \\ x^{(\infty)}(c^\infty+1) \\ \vdots \\ x^{(\infty)}(N) \end{bmatrix},$$

$$(\tilde{V})_{ij} = \begin{cases} 0 & j > i \\ (-1)^{i+j}\begin{pmatrix} i-1+c^\infty \\ j-1+c^\infty \end{pmatrix} & j \leq i \end{cases},$$

$$\begin{aligned} \tilde{D} &= \prod_{k=c^\infty}^{N-1} \begin{bmatrix} \frac{T^\infty-k}{U+N-k} & 0 & \cdots & 0 \\ 0 & \frac{T^\infty-k-1}{U+N-k} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{U-k}{U+N-k} \end{bmatrix} \\ &= \frac{U!}{T^\infty!}\begin{bmatrix} \frac{(T^\infty-c^\infty)!}{(U-c^\infty)!} & 0 & \cdots & 0 \\ 0 & \frac{(T^\infty-c^\infty-1)!}{(U-c^\infty-1)!} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{(U-c^\infty)!}{(U-N)!} \end{bmatrix} \end{aligned}$$

and $T^\infty = U + N - c^\infty$.

Solving the system in Equation (3) for $E[\tilde{x}]$ gives the equation in Figure 1. The inverse is well defined because the elements on the diagonal are all non-zero. The next section will compare this expression with the simulation results.

$$
\begin{aligned}
E[\tilde{x}] &= \tilde{V}^T (I-\tilde{D})^{-1}\tilde{D}\tilde{V}^{-T}(e_{N-c^\infty}-e_1) \\
&= \tilde{V}^T \begin{bmatrix} \frac{T^\infty!(U-c^\infty)!}{(T^\infty-c^\infty)!U!}-1 & 0 & \cdots & 0 \\ 0 & \frac{T^\infty!(U-c^\infty-1)!}{(T^\infty-c^\infty-1)!U!}-1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{(U-N)!(U+N-c^\infty)!}{(U-c^\infty)!U!}-1 \end{bmatrix}^{-1} \begin{bmatrix} \binom{N}{c^\infty}-1 \\ \binom{N}{c^\infty+1} \\ \vdots \\ \binom{N}{N} \end{bmatrix}.
\end{aligned}
$$

Fig. 1. Equation to compute the steady state distribution of valid pages along the blocks.

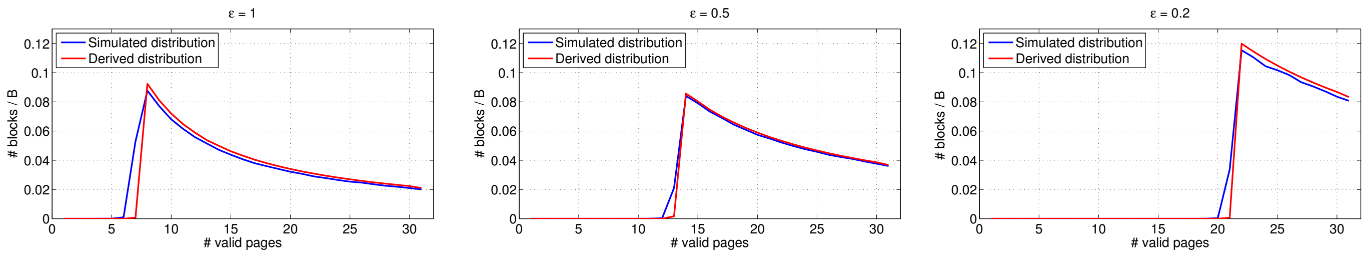

Fig. 2. Distribution of valid pages along the blocks for different values of over-provisioning $\epsilon = \frac{BN-U}{U}$.

4. NUMERICAL RESULTS

This section provides numerical results that validate the derivation in the previous section. Figure 2 shows the steady state distributions of valid pages. The red curve is obtained by numerically evaluating the equation in Figure 1 and the blue curve is obtained via Monte-Carlo simulations of full drive runs. For each full drive run, the simulation starts with a full drive and garbage collection is done 10^6 times[1]. The distribution of valid pages in simulations was then obtained by averaging the steady state histograms for distribution of valid pages over 10^5 full drive runs. As seen in Figure 2, the derived expression for steady state distribution of valid pages matches the one obtained through simulations, validating the derivation. In all plots $N = 32$ and $B = 900$; and the value of U was varied to achieve different values of over-provisioning ϵ. In practice, a block typically has 32, 64 or 128 pages. The smaller value of 32 was chosen because the equation in Figure 1 is prone to numerical errors for large values of N.

5. CONCLUSION

This paper presented a novel formulation to derive the distribution of valid pages in the blocks of a flash drive with greedy garbage collection, once it has reached its steady state. It was shown using Monte-Carlo simulations that the derived expression captures the actual distribution fairly accurately. In future work, the authors intend to use this formulation to investigate and characterize new garbage collection algorithms that improve upon the greedy strategy in terms of reducing write amplification. Since write-amplification depends entirely on the distribution of valid pages, the formulation proposed here may be useful in characterizing the write-amplification of other algorithms.

[1]It was found that 10^6 runs of garbage collection was sufficient to reach steady state, as also seen in [5]

6. REFERENCES

[1] B. Peleato and R. Agarwal, "Maximizing mlc nand lifetime and reliability in the presence of write noise," in *Proc. of IEEE International Conf. Comm.*, 2012.

[2] J. Menon and L. Stockmeyer, "An age-threshold algorithm for garbage collection in log-structured arrays and file systems," *KLUWER INTERNATIONAL SERIES IN ENG. AND COMP. SCIENCE*, pp. 119–132, 1998.

[3] W. Bux and I. Iliadis, "Performance of greedy garbage collection in flash-based solid-state drives," *Performance Evaluation*, 2010.

[4] X.Y. 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*. ACM, 2009, p. 10.

[5] R. Agarwal and M. Marrow, "A closed-form expression for write amplification in nand flash," in *GLOBECOM Workshops, IEEE*, 2010, pp. 1846 –1850.