On the Optimality of Greedy Garbage Collection for SSDs

Yudong Yang
Columbia University
yyd@cs.columbia.edu

Vishal Misra
Columbia University
misra@cs.columbia.edu

Dan Rubenstein
Columbia University
danr@cs.columbia.edu

ABSTRACT

Solid state drives have been widely applied in modern computer systems. The lifetime of the SSD depends heavily on the efficiency of the implementation of the garbage collection (GC) algorithm that reclaims previously used pages. In this paper, we present the first detailed proof that the greedy GC algorithm has the optimal performance (minimized write amplification) for memoryless workloads.

1. INTRODUCTION

Solid State Drives (SSDs) have been widely applied as persistent storage in personal computers and cloud servers due to their high performance and low energy consumption. In a SSD, space is partitioned into *blocks*, with each block containing the same number of *pages*. Data to be stored is written into pages, and a page can be reused only when it holds data that is no longer needed (invalid) and the block containing the page is **cleaned**. Cleaning is a costly operation, in that a block becomes unusable for storage after being cleaned too often (in the range of the tens or hundreds of thousands, depending on the specific design of SSD [2]). Hence, it is important to design efficient garbage collection (GC) algorithms that decide when to perform a clean, and which block to clean.

Intuitively, it is preferable to clean a block with more invalid pages since such a clean frees more pages for reuse. It is therefore also intuitive that an algorithm that greedily chooses the block with the most invalid pages as its next block to clean will perform well, perhaps even optimally, in general.

Despite this intuition that greedy is optimal and a large body of prior work studying the greedy GC policy, to our knowledge, no formal proof of this exists prior to our work here. **Here, we prove that greedy GC is optimal for workloads where the lifetimes of pages are exponentially distributed, or, more simply put, the workload is *memoryless*.**

In [5], the authors analyze the write amplification of greedy GC algorithm for memoryless workloads and propose a windowed greedy policy to ease the algorithm complexity. Another closed-form solution for write amplification under greedy GC algorithm for memoryless workloads is presented in [3], the result is more precisely compare to [5] especially when the spare factor is small ($S_f < 0.2$). Bux et al. [1] also give a closed form of the write amplification for greedy GC in large systems under memoryless workloads. Lin et al. [7] propose a Dual-greedy heuristic and compare it with the greedy GC algorithm and other heuristics, suggesting that the greedy GC algorithm outperforms several heuristics on memoryless workloads but Dual-greedy performs best on traces of workloads. The authors of [8] and [6] use a mean field model to analyze the performance of the D-choice GC algorithm which cleans the block with minimal valid pages among d randomly selected blocks. In [4], the authors give a proof sketch about the optimality of greedy GC algorithm for random write workload, in which the utilization factor μ is fixed, which is a special case of our memoryless workload model.

2. SYSTEM MODEL

A SSD contains N blocks, each block contains B pages, where at any time a page is either **valid**, **invalid** or **empty**. Each arrival is written to an empty page which, upon writing, becomes valid. Page lifetimes are independent and exponentially distributed with identical rate μ, such that at any time, all valid pages have equal probability of becoming the next invalid page. When a block is **cleaned**, all invalid pages in that block become empty. We assume that cleaning can be completed (almost) instantaneously, such that arrivals do not queue up to be placed. The arrivals of new pages which must be written to storage can be described by arbitrary process, but we assume that the number of valid pages never exceeds NB such that an arriving page can always be stored. If run for long enough, an SSDs pages will all be valid or invalid, having no empties to place incoming arrivals. Hence, a **Garbage Collection Algorithm** must, from time to time, select a block to clean.

A common measure of performance of a GC Algorithm is the **write-amplification**, defined as

$$lim_{M\to\infty}\frac{MB}{\sum_{j=1}^{M} i_j},$$

where M is the number of cleans performed and i_j is the number of invalid pages that were emptied on the jth clean. A smaller write amplification is better, since this implies that on average, more space was freed for future storage per clean.

The Greedy GC algorithm waits until the SSD is filled, and then cleans a block with maximal number of invalids[1]. There are two variants, previously considered, which we also consider here:

[1] Ties are broken arbitrarily

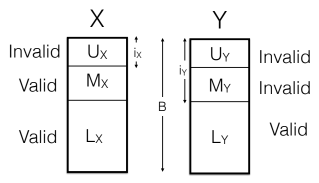
- **D-choice**: when choosing a block to clean, restrict the algorithm to choose among D randomly chosen blocks. $D = 1$ is a "pure" random algorithm, and $D = N$ is the most generalized (no restriction on choice). Greedy would choose the block from the set of D choices with the maximal number of invalids.
- **Clean-and-move:** The process of cleaning a block involves buffering valid pages in a temporary space, erasing the entire block, and then rewriting the valid pages back to the block. Clean-and-move allows the buffered valid pages to be written to other blocks (in the case where the other blocks have empty pages).

3. OPTIMALITY OF GREEDY

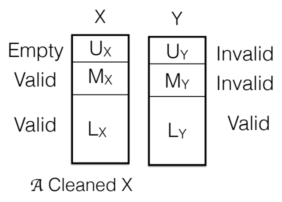
We show that Greedy is optimal in comparison to any other on-line algorithm $\mathcal{A}$, which can, at any time t, choose blocks to clean based only on the history of events up to time t (i.e., any previous arrivals, departures, moves and cleans prior to time t).

Lemma 1. *For any algorithm, a block should only be cleaned when it is full (of valid and invalid pages).*

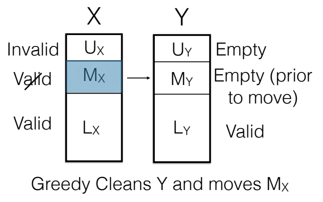
Proof. Consider a block that has been cleaned while containing empties, and let a be the next arrival to be stored in the block. Then there was space for a prior to the cleaning, and hence a could have been placed prior to the cleaning. Furthermore, by delaying the cleaning, additional pages may have expired, such that a later cleaning would free up more invalid pages. ☐



(a) Blocks before clean



(b) After $\mathcal{A}$ cleans



(c) After Greedy cleans and moves

Figure 1: A visual perspective of the respective clean operations.

Theorem 1. *In clean-and-move systems with memoryless workloads, Greedy is optimal.*

Proof. The proof is by contradiction. Consider an optimal algorithm $\mathcal{A}$ that chooses a block X other than the block containing the largest number of invalids to clean, and let Y be a block (as chosen by Greedy) with the largest number of invalids. By Lemma 1, neither X nor Y would have empty pages (since an optimal algorithm would delay their cleaning until they were indeed full.) Hence, block $S \in \{X, Y\}$ has i_S invalid pages and $B - i_S$ valid pages where $i_X < i_Y$. We divide the pages each block S into 3 regions: the upper U_S is a fixed collection of i_X invalid pages, the lower L_S is a fixed collection of $B - i_Y$ valid pages, and the middle of size $i_Y - i_X$ where the pages in M_X are valid and the pages in M_Y are invalid. These regions are depicted in Figure 1(a).

An important observation is that due to the memoryless property, the pages in L_X are identical in a stochastic sense to those in L_Y: they are all valid and the distributions on each page's remaining lifetime is identical.

When $\mathcal{A}$ cleans block X, block X will contain i_X empty pages, and $B - i_X$ valid pages, while block Y maintains i_Y invalid pages and $B - i_Y$ valid pages, as shown in Figure 1(b).

After Greedy cleans block Y, there are $B - i_Y$ valid pages and i_Y empty pages, and block X contains $B - i_X$ valid pages and i_X invalid pages. Assume the pages in region M_X are immediately copied over to (empty region M_Y in Y, and all blocks in M_x are invalidated. By doing this, we now have an additional $i_Y - i_X$ valid pages in Y making the total $B - i_X$, $i_Y - i_X$ fewer empty pages in Y making the total i_X, $i_Y - i_X$, with the number of invalids equalling 0. We also created $i_Y - i_X$ additional invalid pages in X making the total number of invalids equal to i_Y, and reduced the number of valid pages by $i_Y - i_X$ making the valid number $B - i_Y$, as shown in Figure 1(c).

In other words, from a stochastic standpoint, block X in Greedy-with-move is stochastically equivalent to Y in $\mathcal{A}$, while block Y in Greedy-with-move is stochastically equivalent to X in $\mathcal{A}$, and Greedy chose the block with more invalid pages to be cleaned.

The proof is concluded by constructing another algorithm $\mathcal{B}$ which duplicates $\mathcal{A}$ until the first time t when $\mathcal{A}$ selects a block X that Greedy would not select, and $\mathcal{B}$ instead selects the block Y that Greedy would select and peform the move described above. Since the states of systems are stochastically equivalent after these events, $\mathcal{B}$ could "imitate" algorithm $\mathcal{A}$, except that it would treat block X as $\mathcal{A}$ treats Y, and treating Y as $\mathcal{A}$ would treat block X, thereby making the j_i equal in expectation for all subsequent cleans. By repeating the process on $\mathcal{B}$ at each point where it deviates from Greedy, we reduce write amplification and converge toward a completely Greedy algorithm. ☐

Theorem 2. *For memoryless workloads, there is no advantage to moving active pages between blocks.*

Proof. Assume for the sake of contradiction that some number $n > 0$ of valid pages in block X must be moved to block Y at time t to minimize the write amplification.

First, we note that any move and clean events of a block Z can be delayed until an arrival of some page p that needs to be placed in either Z or some block involved in a move with Z. Hence, without loss of generality, we can assume that there is a new arrival p at time t that is to be placed in either X or Y, and that p cannot be placed prior to the move (without increasing write amplification). Hence at time t, the move will occur, followed by (possibly) cleaning block X and/or block Y, followed by placement of p in block X or block Y.

We first consider the case where p is to be placed in Y. Y must have exactly n empty pages, or we could place p in an empty page of Y prior to the move and still have enough pages to move the remaining n pages from X to Y. Hence, the n pages are moved, then Y is cleaned, freeing some invalid pages, where p can then be placed. However, the cleaning of Y could have been performed prior to the move (the n empty pages would remain empty, the remaining pages have the same status whether the order of events is move, clean or clean, move). We could therefore, at time t, clean Y, place p, have at least n empty pages in Y for a move that would not need to occur until some later arrival q. Thus, the move can be delayed until after time t wihtout impacting the number of invalids cleaned, and we can keep delaying the move as long as the page is to be placed into Y.

Next, consider the case where p is to be placed in X. X must have no empty pages at time t, or p could be placed there prior to the move, contradicting the assumption. Hence, the move must be performed, followed by a cleaning of X, followed by the placement of p. We distinguish one particular page q of the n valid pages being moved, and propose that instead of moving the n valid pages, we instead move $n-1$ of the pages to Y, and also place p in Y. Because of the memoryless assumption, the remaining lifetimes of p and q are stochastically equivalent, such that the state of the two systems (moving n pages versus moving $n-1$ and p) are stochastically equivalent. Furthermore, at time t, we can simply place p in Y, and delay the move of the remaining $n-1$ pages until a subsequent arrival. Note that this argument holds for the case where $n=1$ (there is simply no move in our proposed modification). As before, the proposed change does not modify the number of invalids performed by the clean.

We have shown that at time t, it is possible to delay the move to a later time,[2] contradicting the assumption that the move must take place at time t. $\square$

Note that the proof extends to the case where one block X to multiple other blocks by separately applying the proof to the subsets of blocks moved to a single block.

Note also that in a memoryless system, Theorem 2 provides a mapping from an algorithm $\mathcal{A}$ with moves to a comparable (or better) algorithm without moves. If $\mathcal{A}$ moves n pages from block X to block Y, we would instead abort the move and place the next n arrivals that would have been destined for block X into block Y, following the same cleaning schedule for Y as in $\mathcal{A}$, and postponing a cleaning of X until at least after Y filled. This non-move algorithm maintains identical write amplification.

4. DISCUSSION

First, we wish to point out that our proofs apply directly to D-choice variants (for $D > 1$, as our proof only looking at 2 blocks. Theorem 1 applies where both $\mathcal{A}$ and Greedy are allowed to choose from the same (restricted) set of D blocks.

While the proof demonstrates optimality with respect to the write-amplification metric, it also applies to other natural metrics for the problem, such as minimizing (in expectation) the number of cleans that need to be performed by a time t.

5. FUTURE WORK

While we have demonstrated that Greedy is optimal in memoryless systems, the optimality of greedy GC algorithm for more general workloads remains unclear. Prior work[9] suggests that greedy GC algorithm is not optimal for Rosenblum workloads, or the workloads where the lifetime is long-tailed distributed[7]. We conjecture that Greedy, or some variant therein, may still be the optimal for short-tailed workloads. Our proofs do provide us with some insights on the problem:

- It seems useful to consider Clean-and-Move variants in addition to strict Clean versions, as they should be implementable in practice, possibly offer improved performance, and may be useful in proving bounds on the performance on the non-move counterparts.

- The memoryless property allowed to abstract out the differences that might exist (in terms of remaining lifetime) of valid pages. Presumably, for other distributions, the lifetimes must be taken into account when selecting a block. In other words, one does not want to clean a block now which may, in a very short time, have a much larger number of invalid pages. Hence, some scheme that "weights" valid blocks might be needed for optimality.

Last, we have also designed an off-line heuristic which has knowledge of future arrivals and lifetimes, and have shown experimentally that such knowledge can be used to outperform Greedy. Thus, the problem of understanding the on-line competitive ratio of Greedy is still open.

[2]and/or possibly reducing, but never increasing, the number of moved pages.

6. REFERENCES

[1] W. Bux and I. Iliadis. Performance of greedy garbage collection in flash-based solid-state drives. *Performance Evaluation*, 67(11):1172–1186, 2010.

[2] F. Chen, D. A. Koufaty, and X. Zhang. Understanding intrinsic characteristics and system implications of flash memory based solid state drives. In *ACM SIGMETRICS Performance Evaluation Review*, volume 37, pages 181–192. ACM, 2009.

[3] P. Desnoyers. Analytic modeling of ssd write performance. In *Proceedings of the 5th Annual International Systems and Storage Conference*, page 12. ACM, 2012.

[4] X. Haas and X. Hu. The fundamental limit of flash random write performance: Understanding, analysis and performance modelling. Technical report, IBM Research Report, 2010/3/31, 2010.

[5] 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*, page 10. ACM, 2009.

[6] Y. Li, P. P. Lee, and J. Lui. Stochastic modeling of large-scale solid-state storage systems: analysis, design tradeoffs and optimization. In *Proceedings of the ACM SIGMETRICS/international conference on Measurement and modeling of computer systems*, pages 179–190. ACM, 2013.

[7] W.-H. Lin and L.-P. Chang. Dual greedy: Adaptive garbage collection for page-mapping solid-state disks. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2012*, pages 117–122. IEEE, 2012.

[8] B. Van Houdt. A mean field model for a class of garbage collection algorithms in flash-based solid state drives. In *ACM SIGMETRICS Performance Evaluation Review*, volume 41, pages 191–202. ACM, 2013.

[9] B. Van Houdt. Performance of garbage collection algorithms for flash-based solid state drives with hot/cold data. *Performance Evaluation*, 70(10):692–703, 2013.