Performance of garbage collection algorithms for flash-based solid state drives with hot/cold data

Benny Van Houdt
Department of Mathematics and Computer Science,
University of Antwerp - iMinds, Belgium
benny.vanhoudt@ua.ac.be

Abstract

To avoid a poor random write performance, flash-based solid state drives typically rely on an internal log-structure. This log-structure reduces the write amplification and thereby improves the write throughput and extends the drive's lifespan. In this paper, we analyze the performance of the log-structure combined with the d-choices garbage collection algorithm, which repeatedly selects the block with the fewest number of valid pages out of a set of d randomly chosen blocks, and consider non-uniform random write workloads. Using a mean field model, we show that the write amplification worsens as the *hot* data gets *hotter*.

Next, we introduce the double log-structure, which uses a separate log for internal and external write requests. Although the double log-structure performs identical to the single log-structure under uniform random writes, we show that it drastically reduces the write amplification of the d-choices algorithm in the presence of hot data. In other words, the double log-structure yields an automatic form of data separation. Further, with the double log-structure there exists an optimal value for d (typically around 10), meaning the greedy garbage collection algorithm is no longer optimal. Finally, both mean field models introduced in this paper are validated using simulation experiments.

1 Introduction

Flash-based solid state drives (SSDs) have many advantages over hard disk drives such as performance and power consumption, however, the limited lifespan and random write performance remains a concern [5, 15]. To understand the need for and impact of garbage collection algorithms on the SSD write performance it is vital to understand the internal structure of flash memory.

Data on flash-based SSDs is organized in blocks that each consist of b pages with a fixed size. Typical page sizes are 2 or 4 Kbyte and b is a power of two often as large as 128. The unit of data exchange with an SSD is a page, meaning any read (write) operation reads (writes) an entire page. At any point in time a page is either in the *erase*, *valid* or *invalid* state. A page can be read whenever it is in the valid state (often in less than 50μs) and reads do not affect the state of a page. Writes can only be performed on pages that are in the erase state. Hence, whenever data needs to be written to a page that is in the valid or invalid state, the page must be erased first. However, SSDs can only erase entire blocks and not individual pages. Thus, in principle, write operations to (in)valid pages can be implemented by first copying all the valid pages from the block (to memory), then erasing the entire block and finally writing the valid and new pages back to the block.

However, erase operations on SSDs are very time consuming (up to a few ms) and affect the lifespan of the SSD as each block can only be erased a limited number of times before becoming unstable. More specifically, single-level cells (SLC), which are the most expensive per Gbyte, can sustain as many as 100,000 cycles, while the cheaper multi-level (MLC) and triple level (TLC) cells can typically sustain less than 10,000 cycles [5, 9]. Therefore, the above-mentioned solution would be very damaging for the performance and lifespan of an SSD, especially under random writes. Sequential writes are less harmful as they tend to write data to a sequence of pages that belong to the same block and one erase operation suffices to write a sequence of pages (as opposed to just one for a random write). To avoid a poor random write performance SSDs typically rely on a log-structure and support out-of-place writes [5]. More specifically, if a write occurs to a valid/invalid page, the state of the page is simply set to invalid and the data is written elsewhere (on a page in the erase state). To allow for such out-of-place writes the SSD maintains a table that maps the logical page numbers to physical page numbers.

To determine the location of the page where the data is actually written, the SSD relies on a garbage collection (GC) algorithm. This algorithm will repeatedly select a block (preferably with as few valid pages as possible), copy the valid pages to memory, erase the block and copy the valid pages back. This results in a block, called the *write frontier*, that contains erased and valid pages only. After executing the GC, the next few write operations will make use of the erased pages of the write frontier (changing their state to valid). When the write frontier is full (meaning it no longer contains any erased pages), the GC is activated again to create a new write frontier.

Whenever a block is selected by the GC algorithm that contains $k > 0$ valid pages, k additional writes take place (to copy the valid data to memory and back). The ratio of the total number of writes and the number of externally requested writes is termed the *write amplification* and it is a measure for the effectiveness of a GC algorithm. The write amplification of various GC algorithms has been analyzed:

- The FIFO GC algorithm [18, 14, 22, 7], which selects the blocks in a cyclic manner.
- The Greedy GC algorithm [4, 7], which selects the block containing the fewest number of valid pages.
- The Windowed GC algorithm [11], which tries to combine the simplicity of the FIFO algorithm with the performance of the Greedy algorithm.
- The d-choices GC algorithm [20, 13], which selects the block with the fewest number of valid pages out of a set of d randomly chosen blocks.

Although the Greedy algorithm minimizes the write amplification in case of uniform random writes [11], the d-choices algorithm was shown to perform only slightly worse than the Greedy algorithm even for moderate values of d, e.g., $d = 10$ [20]. The FIFO GC algorithm is the easiest to implement, but results in the worst write amplification. The Windowed GC algorithm reduces the write amplification of FIFO, but unless the window size is very large the reduction is quite limited. As such the Windowed GC algorithm offers a less attractive tradeoff between simplicity and performance than the d-choices GC algorithm [20].

Most of the above models consider uniform random writes. The write amplification under non-uniform random writes was also analyzed in [7] for both the FIFO and Greedy algorithm, by relying on the hot/cold data model of Rosenblum [19]. In this model pages that regularly receive write requests are termed hot, while the others are termed cold (see Section 2.1). The results in [7] indicate that the write amplification worsens as the hot data gets hotter for the FIFO and Greedy GC algorithm (without data separation). Data separation techniques can be used to reduce the write amplification by using different blocks to store cold and hot data [17, 12, 10].

This paper contains the following contributions. First, we extend the mean field model presented in [20], which analyzes a class of GC algorithms under uniform random writes, to study the impact of non-uniform random writes. More specifically, we use the hot/cold data model of Rosenblum [19] to model the workload and show that the write amplification of the d-choices algorithm also *worsens* as the hot data gets hotter. Next, we propose the use of two write frontiers (i.e., a double log-structure): one for internal writes and one for externally requested writes. Although using one or two write frontiers results in the same write amplification under uniform random writes, we show by means of a mean field model, that the write amplification of the d-choices algorithm now *reduces* as the hot data gets hotter. In other words, the use of the double write frontier yields an automatic form of data separation. Moreover, we show that there exists an optimal value for d which implies that the Greedy algorithm is no longer optimal.

Although some prior analytic studies also looked at the impact of data separation [7, 11], they assume that the system is aware of the identity of the hot and cold pages, meaning some mechanism needs to be in place that identifies the hot and cold data. The use of the double write frontier does not require any such mechanism, but only assumes that the system is able to distinguish between an externally and internally requested write.

This paper and the above-mentioned studies focus on the write amplification in page-mapped flash translation layers (FTLs). Block- and hybrid-mapped FTLs [8, 6] have the advantage that they limit the amount of memory required to translate logical to physical page numbers, but at the same time they also limit the flexibility of the GC algorithm as it is no longer possible to map any logical page on any physical page. We do not consider the issue of wear leveling either. Wear leveling mechanisms try to prolongate the lifetime of the SSD by making sure that the number of write-erase cycles on a block does not vary too much. Some static wear leveling algorithms simply swap entire blocks (basically to move cold data to more worn out blocks), for instance by swapping the least and most worn out block or by swapping the free block with a randomly selected block as in Ban's algorithm [1]. When this type of swapping is used, the distribution of the number of valid pages is not affected by the wear leveling algorithm.

The remainder of the paper is structured as follows. Section 2 contains the description of the workload model, lists the GC algorithms under consideration and discusses the single and double log-structure operation. Section 3 presents the mean field model for non-uniform random writes for the single log-structure, validates the model using simulation experiments and discusses some numerical examples. Section 4 introduces the mean field model for the system with two write frontiers, validates this model and shows that the write amplification improves as the hot data gets hotter, while the greedy algorithm is no longer optimal. In Section 5 conclusions are drawn and some open issues are discussed.

2 Problem description

2.1 Hot/cold data model

Consider a flash-based SSD consisting of N (physical) blocks that are each able to store b pages (e.g., $N = 16,384$ for a device with a physical capacity of 8GB that stores $b = 64$ pages of size $8K$

per block). To limit the write amplification, the total storage capacity bN (=number of physical pages) on an SSD exceeds the user-visible capacity (=number of logical pages). This guarantees that a fraction of the pages is in the erase or invalid state at all times. A commonly used measure for the amount of over-provisioning is the *spare factor* S_f, defined as one minus the ratio of the user-visible to the total storage capacity. If we denote the user-visible storage capacity as bU pages, then the device *utilization* $\rho = U/N$ and the *spare factor* $S_f = 1 - U/N = 1 - \rho$.

Whenever a file is deleted, the file system will add the space previously occupied by the file to its free space. However, by default it does *not* pass this information down to the SSD, which implies that the pages that were occupied by the file remain in the valid state even though the corresponding file is deleted[1]. This implies that *exactly bU pages are marked as valid at all times*, provided that the system has been operational for a while, and each write operation in some sense "updates" a page.

Under uniform random writes there is no spatial or temporal locality and each page is updated with probability $1/bU$. Let n_i denote the number of the physical blocks containing exactly i valid pages and note that $\sum_{i=0}^{b} in_i = bU$. Under uniform random writes, the probability that an external write updates a page stored on a block with exactly i valid pages is therefore equal to $n_i i/bU$.

In this paper we consider non-uniform random writes, meaning not all the pages are updated with equal probability. More specifically, we rely on the hot/cold data model of Rosenblum [19]. In this model a fraction f of the logical address space corresponds to *hot* data and the remaining fraction to *cold* data. The fraction of write operations to the hot data is denoted as r. Typical case studies with hot and cold data assume that $f \leq 0.2$ and $r \geq 0.8$, meaning more than 80% of the writes are to less than 20% of the data [7].

Let $n_{i,j}$ denote the number of the physical blocks containing exactly i hot and $j - i$ cold valid pages. As the total number of hot and cold pages equals bUf and $bU(1-f)$, respectively, we have $\sum_{j=0}^{b}\sum_{i=0}^{j} in_{i,j} = bUf$ and $\sum_{j=0}^{b}\sum_{i=0}^{j}(j-i)n_{i,j} = bU(1-f)$. As a fraction r of the writes go to hot pages, the probability $u_{i,j}^{(hot)}$ that an arbitrary write updates a hot page on a block with exactly i hot and $j - i$ cold valid pages is given by

$$u_{i,j}^{(hot)} = r\frac{in_{i,j}}{bUf}. \tag{1}$$

[1]It is possible to mark the pages of deleted files as invalid on the SSD by means of the ATA TRIM command (if supported by the operating system and SSD). As the early version of this command is a non-queued command, it may cause an execution penalty if executed after each delete command.

Similarly, an arbitrary write updates a cold valid page on such a block with probability

$$u_{i,j}^{(cold)} = (1-r)\frac{(j-i)n_{i,j}}{bU(1-f)}. \tag{2}$$

2.2 Garbage collection algorithms

The mean field models introduced in this paper can be used to assess the write amplification of any algorithm belonging to the class $\mathcal{C}$ of GC algorithms introduced in [20]. In order to define class $\mathcal{C}$, let $m_{i,j}$ be the fraction of physical blocks containing exactly j valid pages, i of which are hot, and denote $m_j = \sum_{i=0}^{j} m_{i,j}$. Further, define $\vec{v} = (m_0, \ldots, m_b)$ and $\vec{m} = (\vec{m}_0, \ldots, \vec{m}_b)$ with $\vec{m}_j = (m_{0,j}, \ldots, m_{j,j})$, for $j = 0, \ldots, b$. A GC algorithm belongs to class $\mathcal{C}$ if and only if the following two conditions are met:

C1: There exist a set of probabilities $p_j(\vec{v})$ such that $p_j(\vec{v})$ reflects the probability that a block containing exactly j valid pages is selected by the GC algorithm. In other words, whether a block is selected by the GC algorithm should only depend on the total number of valid pages it contains and on the fraction of blocks m_j containing exactly j valid blocks, for $j = 0, \ldots, b$.

C2: For $j = 0, \ldots, b$, the probabilities $p_j(\vec{v})$ should be smooth in $\vec{v}$ with $\vec{v} \in \Delta_v = \{\vec{v} \in \mathbb{R}^{b+1} | 0 \leq m_j \leq 1, \sum_{j=0}^{b} m_j = 1, \sum_{j=1}^{b} j m_j = b\rho\}$.

Note that the above definition does *not* specify whether the valid pages are hot or not. In other words, the GC algorithms in $\mathcal{C}$ do not need to know the identity of the hot and cold pages. The selection is solely based on the total number of valid pages in a block. In fact, the mean field models in this paper can be readily used to study a class of GC algorithms that take the cold/hot nature of the pages into account. Such algorithms do however need to rely on an additional mechanism to identify the hot and cold data.

Examples of GC algorithms belonging to $\mathcal{C}$ are the Random, Random++ and the d-choices GC algorithm [20]. For the numerical experiments we will focus on the d-choices algorithm only as this algorithm is simple and has a close to optimal performance under uniform random writes even for small values of d, e.g., $d = 10$.

For the d-choices algorithm the probabilities $p_j(\vec{v})$ are given by

$$p_j(\vec{v}) = \left(\sum_{\ell=j}^{b} m_\ell\right)^d - \left(\sum_{\ell=j+1}^{b} m_\ell\right)^d, \tag{3}$$

as all the selected pages must contain at least j valid pages, but not $j+1$. For further use, we also define $p_{i,j}(\vec{m})$ as the probability that the GC algorithm selects a block with i hot and $j-i$ cold valid pages. For the algorithms belonging to class $\mathcal{C}$ we have

$$p_{i,j}(\vec{m}) = p_j(\vec{v})\frac{m_{i,j}}{\sum_{i=0}^{j} m_{i,j}},$$

provided that $\sum_{i=0}^{j} m_{i,j} > 0$ (otherwise $p_{i,j}(\vec{m}) = 0$). To ease the notation we denote $\sum_{i=0}^{j} p_{i,j}(\vec{m}) = p_j(\vec{v})$ also as $p_j(\vec{m})$.

2.3 Single and double log-structure

In this section we describe the system operation in case of the single and double log-structure, and introduce an expression for the write amplification as a function of the probabilities $p_j(\vec{m})$ characterizing the GC algorithm and the vector $\vec{m}$.

The *single* log-structure uses at all times a single special block called the *write frontier* (WF). Pages will be written sequentially to the WF, until it is full. Assume that the first j pages of the WF are in the (in)valid state, while the last $b-j$ are in the erase state at some point. Next, assume a write operation takes place on a logical page that is physically stored on page k of block number x. This operation first writes the new content to page $j+1$ of the WF (changing its state from erase to valid) and afterwards invalidates page k on block number x. When the WF is full, the GC algorithm creates a new WF as follows: it first selects a new block, say block number y, copies all the valid pages of block y to the random-access memory (RAM), erases block number y and copies the valid pages back from RAM to block y. Note, in practice one avoids the need to copy the valid pages to RAM and back by making use of a single *free* block [7].

The *double* log-structure relies on two WFs: a WF for external writes (WFE) and a WF for internal writes (WFI). The basic idea behind the use of the WFE and WFI is that all externally requested writes are written to the WFE, while the internal write operations make use of the WFI. In this manner, we expect that the WFE contains mostly hot data, while the WFI should contain a lot of cold data. We will show that the double structure indeed yields a form of data separation that results in a reduced write amplification.

With the two WFs, the GC algorithm is invoked whenever the WFE becomes full. Assume that the last $b-j^*$ pages of the WFI are in the erase state when the GC algorithm is invoked, while the first j^* are in the valid/invalid state. Further assume a block with j valid pages is selected by the GC algorithm. Consider the following 2 cases:

1. If $j \leq b - j^*$, the j valid pages of the selected block are simply copied to the WFI leaving

the last $b - j^* - j$ pages in the erase state. After copying the j valid pages to the WFI, a new WFE is created by erasing the selected block. Hence, the next b externally requested write operations can make use of the WFE before the GC algorithm is invoked again.

2. If $j > b - j^*$, $b - j^*$ of the j valid pages are selected[2] and copied to the WFI. The remaining $j - (b - j^*)$ valid pages are copied to RAM and back to the selected block after the selected block has been erased. In this case, the selected block becomes the new WFI and the GC algorithm is immediately invoked again.

The write amplification (WA) is defined as the ratio of the total number of requested writes and the number of externally requested writes. In case of a single WF $b-k$ internal and k external writes are performed whenever a block is selected by the GC algorithm that contains $b - k$ valid pages. Hence, the WA($\vec{m}$) can be expressed as

$$WA(\vec{m}) = \frac{b}{\sum_{k=0}^{b} k p_{b-k}(\vec{m})}. \tag{4}$$

Note that this expression depends on $\vec{m}$ and only when we replace $\vec{m}$ by the equilibrium distribution of the number of valid pages per block we get the true WA.

In case of the two WFs, we can easily express the WA in $\vec{m}$ provided that there are j^* valid pages in the WFI. In this case we get b external and k internal writes if a block with $k \leq b - j^*$ valid pages is selected by the GC and zero external and k internal writes otherwise, hence

$$WA(\vec{m}, j^*) = \frac{\sum_{k=0}^{b} k p_k(\vec{m}) + b \sum_{k=0}^{b-j^*} p_k(\vec{m})}{b \sum_{k=0}^{b-j^*} p_k(\vec{m})}. \tag{5}$$

In Section 4.2 we will argue that the probability of having j^* valid pages in the WFI is equal to $1/b$ for $j^* = 1, \ldots, b$ and $\vec{m} > 0$ (that is when all the entries of $\vec{m}$ are larger than zero), which implies that (5) reduces to (4) when we decondition on j^* with $\vec{m} > 0$.

3 Single write frontier performance

3.1 Mean field model definition

The mean field model in this section generalizes the model of [20] to non-uniform random writes, in fact if we set $r = f$ all the pages are requested equally often and the model reduces to the one for the uniform random writes.

[2]In the model the selection of these $b - j^*$ pages is at random, in the simulation experiments we also consider the case where the first $b - j^*$ pages are copied to the write frontier. The latter method causes a slight reduction in the write amplification (typically less than 2%).

Consider a system consisting of N blocks and observe this system at the time epochs just prior to the execution of the GC algorithm. Let $H_n^N(t) \in S' = \{0, 1, \ldots, b\}$, for $n = 1, \ldots, N$, be the number of hot valid pages on block number n at time t (i.e., when the GC algorithm runs for the t-th time) and $C_n^N(t) \in S'$, for $n = 1, \ldots, N$, be the number of cold valid pages on block number n at time t. Denote $X_n^N(t) = (H_n^N(t), H_n^N(t) + C_n^N(t)) \in S = \{(i,j) | 0 \leq i \leq j \leq b\}$.

Let $M^N(t)$ be the occupancy measure of $X_n^N(t)$, that is, $M^N(t) = \{M_{i,j}^N(t) | (i,j) \in S\}$, while

$$M_{i,j}^N(t) = \frac{1}{N} \sum_{n=1}^{N} 1[X_n^N(t) = (i,j)],$$

for $(i,j) \in S$. In other words, $M_{i,j}^N(t)$ is the fraction of the total number of blocks N that contains j valid pages, i of which are hot. To ease the discussion, we will refer to such blocks as type (i,j) blocks.

It is easy to see that $\{M^N(t), t \in \mathbb{N}\}$ is a Markov chain. However, computing the steady state probabilities of this Markov chain for realistic values of N (e.g., $N = 10,000$) does not appear to be feasible. Therefore we introduce a mean field model that is characterized by a set of ODEs, the solution of which can be used to approximate the evolution of the Markov chain.

More specifically, we define $\bar{M}^N(\tau)$ as the re-scaled process such that $\bar{M}^N(t) = M^N(\lfloor tN \rfloor)$, for $t \geq 0$. We will argue that the limit process of $\bar{M}^N(t)$ as N tends to infinity is a deterministic process $\vec{\mu}(t)$, the evolution of which is captured by the set of ODEs given by (6). In other words, for N large and finite t, we can approximate $M^N(t)$ by $\vec{\mu}(t/N)$, which is the unique solution of (6) with $\vec{\mu}(0) = M^N(0)$.

Next, we define the mean field model by means of the deterministic process $\vec{\mu}(t) = (\vec{\mu}_0(t), \ldots, \vec{\mu}_b(t))$ and $\vec{\mu}_i(t) = (\mu_{0,i}(t), \ldots, \mu_{i,i}(t))$ for $i \in S'$, the evolution of which is given by the following set of ODEs:

$$\frac{d\vec{\mu}(t)}{dt} = \vec{f}(\vec{\mu}(t)), \tag{6}$$

with $\vec{f}(\vec{m}) = \{f_{(i,j)}(\vec{m}) | (i,j) \in S\}$, where $f_{i,j}(\vec{m})$ is defined below. Note that $f_{i,j}(\vec{m})$ represents the change in the number of type (i,j) blocks, in between two executions of the GC algorithm provided that the system state is $\vec{m}$.

Let $u_{i,j}^{(hot)}(\vec{m})$ be the probability that an arbitrary external write updates a hot valid page on a type (i,j) block and define $u_{i,j}^{(cold)}(\vec{m})$ similarly. Hence, by dividing the numerator and denominator of (1) and (2) by N, we get

$$u_{i,j}^{(hot)}(\vec{m}) = \frac{i m_{i,j} r}{b\rho f}, \qquad u_{i,j}^{(cold)}(\vec{m}) = \frac{(j-i) m_{i,j}(1-r)}{b\rho(1-f)}.$$

Define $u_{i,j}(\vec{m}) = u_{i,j}^{(hot)}(\vec{m}) + u_{i,j}^{(cold)}(\vec{m})$. For $j < b$, define

$$f_{(i,j)}(\vec{m}) = \left(\sum_{k=1}^{b} p_{b-k}(\vec{m})k\right) \left[u_{i+1,j+1}^{(hot)}(\vec{m}) + u_{i,j+1}^{(cold)}(\vec{m}) - u_{i,j}(\vec{m})\right] - p_{i,j}(\vec{m}). \tag{7}$$

This expression for the limiting ODE can be understood intuitively by noting that $\sum_{k=1}^{b} p_{b-k}(\vec{m})k$ represents the mean number of externally requested writes in between two executions of the GC algorithm (see also (4)). Any such request to a hot page on a type $(i+1, j+1)$ or to a cold page on a type $(i, j+1)$ block creates a type (i, j) block, while any request to a valid page of a type (i, j) block reduces the number of type (i, j) blocks by one. Finally, we also lose a type (i, j) block if the GC algorithm selected such a block (as it becomes full). For $i = b$, define

$$f_{(i,b)}(\vec{m}) = \sum_{k=0}^{i} \sum_{j=i-k}^{b-k} p_{i-k,j}(\vec{m}) B_k^{b-j,r} - p_{i,b}(\vec{m}) - \left(\sum_{k=1}^{b} p_{b-k}(\vec{m})k\right) u_{i,b}(\vec{m}), \tag{8}$$

where $B_j^{n,p} = \binom{n}{j} p^j (1-p)^{n-j}$. The latter two terms can be understood as before, while the first term states that a type (i, b) block is created (in the write frontier) if k out of the $b - j$ externally requested writes demand a hot page whenever the GC algorithm selects a type $(i - k, j)$ block.

3.2 Convergence and validation

Similar to [20, Section 5] we can prove the following theorem by relying on Corollary 1 in [2]:

Theorem 1. *If $M^N(0) \to \vec{m}$ in probability as N tends to infinity, then $\sup_{0 \le t \le T} ||\bar{M}^N(t) - \vec{\mu}(t)|| \to 0$ in probability, where $\vec{\mu}(t)$ is the unique solution of the ODE (6) with $\vec{\mu}(0) = \vec{m}$.*

This theorem states that for N large and finite t, we can approximate $M^N(t)$ by $\vec{\mu}(t/N)$, which is the unique solution of the ODE (6) with $\vec{\mu}(0) = M^N(0)$. In order to prove that the convergence extends to the stationary regime, Corollary 2 in [2] shows that it suffices that the ODE given by (6) has a unique fixed point that is also a global attractor. Although proving that there exists a global attractor may be possible for some GC algorithms belonging to class $\mathcal{C}$, such a proof seems hard for set of ODEs in (6) corresponding to the d-choices algorithm. In fact, even for the case of uniform random writes a simple closed-form expression for the fixed point does not appear to exist and a proof of global attraction was only provided for $b = 2$, see [20]. Various numerical experiments however suggest that a unique global attractor exists.

The numerical results presented in the paper for the write amplification A and distribution of the number of valid pages were obtained by numerically solving the ODE (6) with $\mu_{i,j}(0) = \mu_j \binom{j}{i} f^i (1-f)^{j-i}$ using Euler's method with a (variable) step size h until $||\vec{\mu}(t+h) - \vec{\mu}(t)||_1 < 10^{-7}$, where $(\mu_0, \ldots, \mu_b)$ is the fixed point of the uniform random writes model in [20]. Note, when $r = f$

b	S_f	d	r	f	ODE	simul. (95% conf.)
16	0.10	16	0.92	0.23	4.5925	4.5925 $\pm$0.0006
16	0.14	13	0.94	0.21	3.7272	3.7275 $\pm$0.0006
32	0.07	9	0.81	0.06	7.6481	7.6490 $\pm$0.0024
32	0.08	5	0.94	0.25	6.5347	6.5349 $\pm$0.0008
32	0.11	14	0.79	0.19	4.6507	4.6507 $\pm$0.0008
32	0.13	14	0.87	0.12	4.4551	4.4554 $\pm$0.0005
32	0.14	15	0.84	0.21	3.8505	3.8507 $\pm$0.0004
64	0.06	4	0.85	0.17	9.2976	9.2985 $\pm$0.0015
64	0.08	2	0.82	0.19	8.6973	8.6976 $\pm$0.0028
64	0.09	6	0.79	0.08	6.5886	6.5885 $\pm$0.0007
64	0.11	11	0.94	0.28	4.8997	4.9002 $\pm$0.0003
64	0.13	15	0.84	0.26	4.1587	4.1588 $\pm$0.0004

Table 1: Comparison of ODE-based results and simulation experiments for a system with $N = 10,000$ blocks for various parameter settings (10 runs). Relative errors are less than 0.01%.

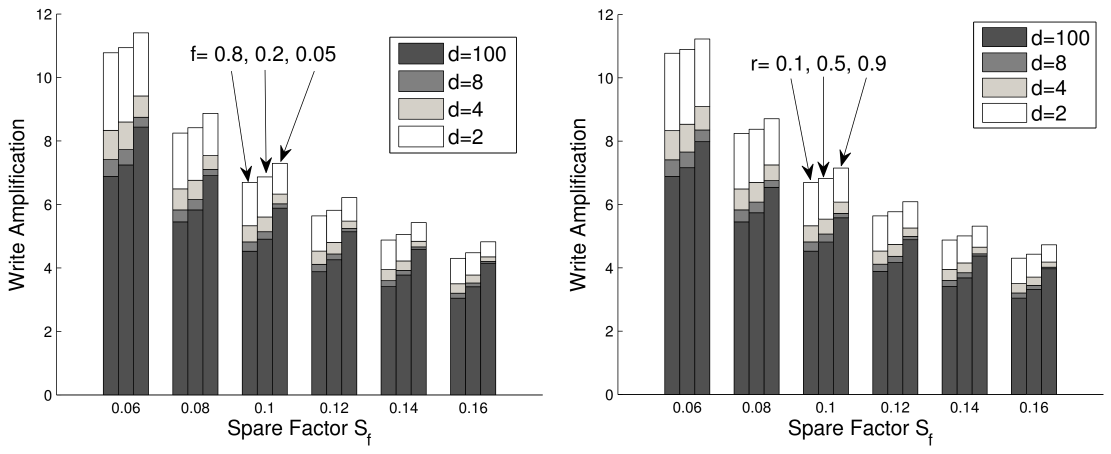


Figure 1: Write amplification for $b = 32$ pages per block as a function of S_f and d with $r = 0.8$ (left) and $f = 0.1$ (right).

the hot/cold data model reduces to the uniform random writes model and $\mu(0)$ as defined above is a fixed point. For all the numerical experiments reported in this paper convergence occurred within seconds.

Table 1 compares various simulation results with the ODE-based prediction for a system consisting of $N = 10,000$ blocks and shows a near perfect agreement. The simulation results were obtained by a customized simulation program that basically simulates the Markov chain $\{M^N(t), t \in \mathbb{N}\}$ defined in Section 3.1 with $N = 10,000$. These simulation results were based on 10 runs with a length of $3tN$, where t was determined using the ODE such that $||\vec{\mu}(t+h) - \vec{\mu}(t)||_1 < 10^{-7}$ and the length of the warm-up period was equal to tN.

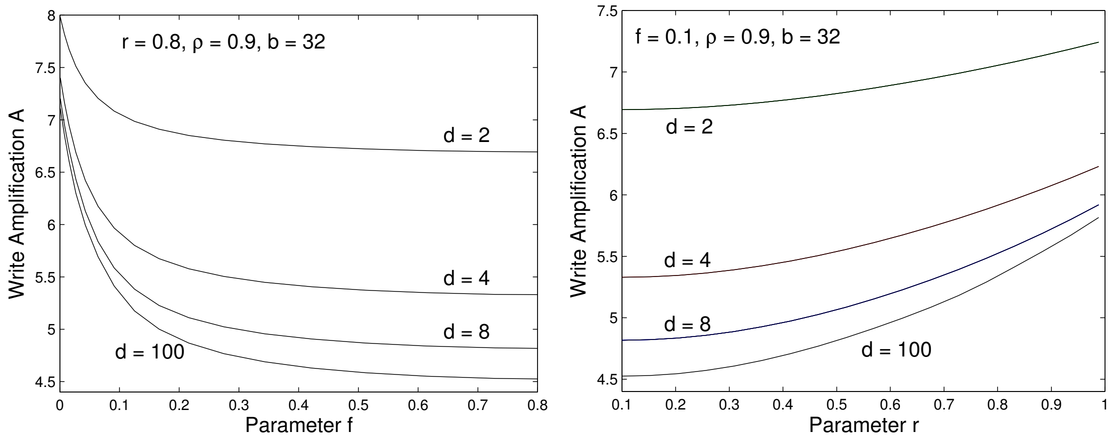


Figure 2: Write amplification for $b = 32$ pages per block, $S_f = 0.1$ and as a function of f with $r = 0.8$ (left) and r with $f = 0.1$ (right).

3.3 Numerical results

Figure 1 depicts the write amplification A as a function of the spare factor S_f and number of choices d in a system with $b = 32$ pages per block and this for different combinations of r and f. In the left part of the figure $r = 0.8$, while $f = 0.1$ in the right part of the figure. As setting $r = f$ results in uniform random writes, the first set of bars corresponds to the model in [20] without hot/cold data. Setting $f = 0.2$ and 0.05 (with $r = 0.8$) implies that 80% of the writes go to 20% and 5% of the address space, respectively, while $r = 0.5$ and 0.9 (with $f = 0.1$) means that 50% and 90% of the writes go to 10% of the address space, respectively.

Figure 1 clearly indicates that the write amplification increases as the hot data gets *hotter*, that is, if either f decreases or r increases. Further, the increase in the write amplification is more pronounced for larger d values, as further demonstrated by Figure 2. In other words, the difference in write amplification between two particular choices for d decreases as the data gets hotter. This implies that even smaller d values suffice to approximate the write amplification of the Greedy GC algorithm[3].

To understand the increase in the write amplification as the hot data gets hotter, Figure 3 shows the distribution of the number of valid pages in an arbitrary block for $b = 16$, $S_f = 0.1$ and $d = 10$, for different r and f values. It shows that as the hot data gets hotter, the number of valid pages per block becomes less variable and blocks with only a limited number of valid pages become more and more scarce. A block with b valid pages that consists mostly of cold data, will quickly

[3]Simulation experiments not reported here indicate that the write amplification of the d-choices algorithm still converges to that of the Greedy algorithm as d tends to infinity.

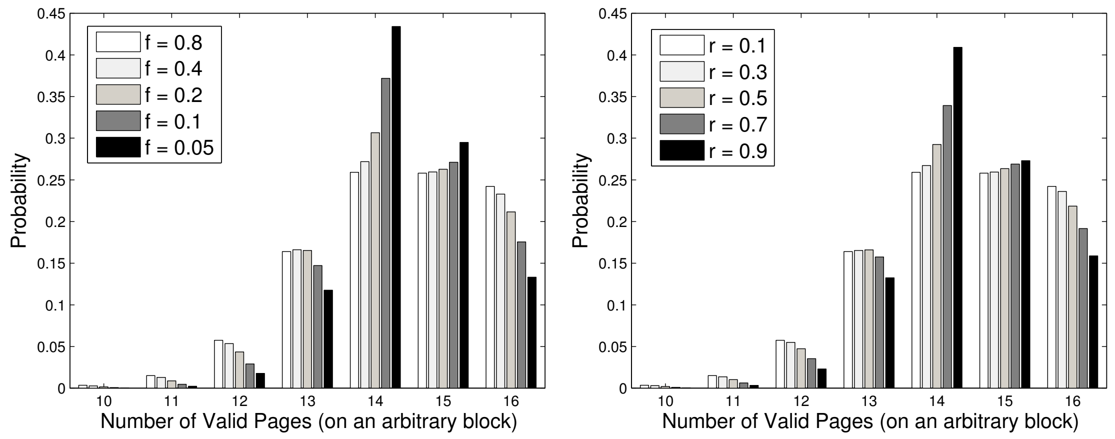


Figure 3: Distribution of the number of valid pages for $b = 16$ pages per block, $S_f = 0.1$ and $d = 10$, for different f values with $r = 0.8$ (left) and r values with $f = 0.1$ (right).

invalidate its hot data, but invalidating the cold data becomes harder as the hot data gets hotter. Blocks with b valid pages that consist mostly of hot data on the other hand are emptied more easily, but some form of data separation between the hot and cold data is needed to create such blocks. The results indicate that the GC algorithm that relies on a single write frontier hardly ever creates such blocks and thus does not achieve any form of data separation. In fact, even if we initialize the system such that the hot and cold data is perfectly separated, the single write is not able to maintain any form of data separation over time. Closer investigation of the results (not depicted here) also indicates that the GC algorithm mostly selects blocks that contain a lot of cold data and nearly no hot data, which implies that the write frontier also contains a lot of cold data.

4 Double write frontier performance

4.1 Mean field model definition

As in Section 3, define the occupancy vector $\hat{M}^N(t) = \{\hat{M}^N_{i,j}(t) | (i,j) \in S\}$, where $\hat{M}^N_{i,j}(t)$ is the fraction of the total number of blocks N that contains j valid pages, i of which are hot, at time t. Define $\tilde{S} = \{(i,j) | 1 \leq j \leq b, 0 \leq i \leq j\}$. Further, let $J(t) \in \tilde{S}$ reflect the content of the WFI just prior the time epoch where the GC algorithm is activated for the t-th time (note, the WFI is never empty). In this case, one readily finds that $\{(\hat{M}^N(t), J(t)), t \in \mathbb{N}\}$ is a Markov chain, which we approximate once more by means of a mean field model.

The mean field model used to analyze the impact of the double WF also fits within the framework developed in [2], but in contrast to the single WF model it relies on a resource $J(t)$. The resulting ODE is therefore of the form

$$\frac{d\vec{\mu}(t)}{dt} = \vec{F}(\vec{\mu}(t)), \quad \text{with } \vec{F}(\vec{m}) = \sum_{(i^*,j^*)\in\tilde{S}} \pi_{i^*,j^*}(\vec{m})\hat{f}(\vec{m},i^*,j^*) \tag{9}$$

where $\pi(\vec{m}) = \{\pi_{i^*,j^*}(\vec{m})|(i^*,j^*) \in \tilde{S}\}$ is the invariant probability vector of $K(\vec{m})$ and $(K(\vec{m}))_{s^+,s^*} = P[J(t+1) = s^*|J(t) = s^+, M(t) = \vec{m}]$, with $s^+ = (i^+,j^+), s^* = (i^*,j^*) \in \tilde{S}$.

If the WFI contains j^* valid pages, b external writes follow the execution of the GC algorithm if the selected block contains at most $b - j^*$ valid pages, otherwise no external writes are performed. Hence, similar to (7) we find for $j < b$

$$\hat{f}_{(i,j)}(\vec{m},i^*,j^*) = \left(\sum_{k=0}^{b-j^*} p_k(\vec{m})b\right)\left[u^{(hot)}_{i+1,j+1}(\vec{m}) + u^{(cold)}_{i,j+1}(\vec{m}) - u_{i,j}(\vec{m})\right] - p_{i,j}(\vec{m}). \tag{10}$$

while for $j = b$

$$\begin{aligned}\hat{f}_{(i,b)}(\vec{m},i^*,j^*) = {} & \left(\sum_{k=0}^{b-j^*} p_k(\vec{m})b\right)\left[B^{b,r}_i/b - u_{i,b}(\vec{m})\right] - p_{i,b}(\vec{m}) \\ & + 1[i^* \le i \le i^* + b - j^*] \sum_{j'=b-j^*+1}^{b} \sum_{i'=i-i^*}^{j'-(b-j^*-i+i^*)} p_{i',j'}(\vec{m}) \frac{\binom{i'}{i-i^*}\binom{j'-i'}{b-j^*-i+i^*}}{\binom{j'}{b-j^*}}.\end{aligned} \tag{11}$$

Indeed, if a block with at most $b - j^*$ valid pages is selected, b external writes occur and a type (i,b) block is created in the WFE if i of these external writes involve hot pages. If the selected block is of type (i',j') with $j' > b - j^*$, then $b - j^*$ valid pages are randomly selected among the j' valid pages and added to the WFI. In order to create a type (i,b) block, $i - i^*$ of these $b - j^*$ valid pages should contain hot data.

To express the entries of the transition matrix $K(\vec{m})$ more easily let $p_{i,j}(\vec{m}) = 0$ for $i > j$. A transition from (i^+,j^+) to (i^*,j^*) occurs with $j^* > j^+$ if a type $(i^* - i^+, j^* - j^+)$ block is selected by the GC algorithm. For $j^* < j^+$ a block with $b - j^+ + j^*$ valid pages needs to be selected such that the new WFI contains j^* valid pages, while i^* of the j^* pages need to be hot. Finally, for $j^* = j^+$ both scenario's apply, which yields

$$\begin{aligned}K_{(i^+,j^+),(i^*,j^*)}(\vec{m}) = {} & 1[j^+ \le j^*, i^+ \le i^*]p_{i^*-i^+,j^*-j^+}(\vec{m}) \\ & + 1[j^* \le j^+] \sum_{i=i^*}^{b-j^++i^*} p_{i,b-j^++j^*}(\vec{m}) \frac{\binom{i}{i^*}\binom{b-j^++j^*-i}{j^*-i^*}}{\binom{b-j^++j^*}{j^*}},\end{aligned} \tag{12}$$

for $(i^+,j^+),(i^*,j^*) \in \tilde{S}$.

4.2 Convergence and validation

To prove that the re-scaled process $\tilde{M}^N(t) = \hat{M}^N(\lfloor tN \rfloor)$, for $t \ge 0$, converges to the solution of the ODE given by (9) as N tends to infinity, it suffices again to check the five conditions of

Corollary 1 in [2]. As a fraction ρf and $\rho(1-f)$ of the physical pages contains valid hot and cold data, respectively, at all times, $\hat{M}^N(t) \in \Delta = \{\vec{m} \in \mathbb{R}^{(b+1)(b+2)/2} | 0 \leq m_{i,j} \leq 1, \sum_{(i,j)\in S} m_{i,j} = 1, \sum_{(i,j)\in S} j m_{i,j} = b\rho, \sum_{(i,j)\in S} i m_{i,j} = b\rho f\}$ for all N. In this case condition H1 of [2], which demands among others that $K(\vec{m})$ is ergodic for any $\vec{m} \in \Delta$, does not hold. More specifically, ergodicity is not always guaranteed when some of the entries of $\vec{m}$ are equal to zero (e.g., if $\vec{m}$ is such that all the blocks contain either 0 or b valid pages). However, for $\vec{m} > 0$ (in each entry), it is easy to see that $K(\vec{m})$ is ergodic. Further, any trajectory that starts in the interior of Δ, denoted as $int(\Delta)$, stays in $int(\Delta)$. As indicated in [3], we can therefore still apply the results of [2] if we restrict the state space to the open subset $int(\Delta)$. Hence,

Theorem 2. *If $\hat{M}^N(0) \to \vec{m} \in int(\Delta)$ in probability as N tends to infinity, then $\sup_{0\leq t\leq T} ||\tilde{M}^N(t) - \vec{\mu}(t)|| \to 0$ in probability, where $\vec{\mu}(t)$ is the unique solution of the ODE* (6) *with $\vec{\mu}(0) = \vec{m}$.*

As in Section 3.2, numerical experiments suggest that the ODE given by (9) has a unique fixed point that is a global attractor and numerical results are obtained in a similar manner using Euler's method. We should however note that due to the presence of the resource $J(t)$, this implies that we need to determine the invariant vector $\pi(\vec{m})$ of a $K(\vec{m})$ matrix during each step. Thus, in order to make the method numerically tractable for realistic values of b, it is important that $\pi(\vec{m})$ can be computed efficiently and without the need to explicitly construct $K(\vec{m})$. This can be done as follows. By (12), we have

$$\pi_{i^*,j^*}(\vec{m}) = \pi_{i^*,j^*}(\vec{m}) p_{0,0}(\vec{m}) + \sum_{j^+=1}^{j^*-1} \sum_{i^+=0}^{i^*} \pi_{i^+,j^+}(\vec{m}) p_{i^*-i^+,j^*-j^+}(\vec{m}) + \\ + \sum_{j^+=j^*}^{b} \pi_{j^+}(\vec{m}) \sum_{i=i^*}^{b-j^++i^*} p_{i,b-j^++j^*}(\vec{m}) \frac{\binom{i}{i^*}\binom{b-j^++j^*-i}{j^*-i^*}}{\binom{b-j^++j^*}{j^*}}, \tag{13}$$

with $\pi_{j^+}(\vec{m}) = \left(\sum_{i^+=0}^{j^+} \pi_{i^+,j^+}(\vec{m})\right)$.

Next, we note that the Markov chain characterized by $K(\vec{m})$ can be lumped with respect to the partition $A_1, \ldots, A_b$ with $A_j = \{(i,j) | i = 0, \ldots, j\}$ for $j = 1, \ldots, b$. Further, the lumped transition matrix is a stochastic Toeplitz matrix with its first row equal to $(p_0(\vec{m}) + p_b(\vec{m}), p_1(\vec{m}), \ldots, p_{b-1}(\vec{m}))$. This implies that $\pi_j(\vec{m}) = 1/b$ for $j = 1, \ldots, b$. Hence, by means of (13) we get

$$\pi_{i^*,1}(\vec{m}) = \frac{1}{(1 - p_{0,0}(\vec{m}))b} \sum_{j^+=1}^{b} \sum_{i=i^*}^{b-j^++i^*} p_{i,b-j^++1}(\vec{m}) \frac{\binom{i}{i^*}\binom{b-j^++1-i}{1-i^*}}{\binom{b-j^++1}{1}} \tag{14}$$

b	S_f	d	r	f	ODE	simul. (95% conf.) copy RND	simul. (95% conf.) copy OLDEST.
16	0.05	12	0.83	0.24	6.7745	6.7754 ±0.0005	6.7205 ±0.0004
16	0.06	5	0.94	0.22	6.0320	6.0326 ±0.0005	5.9081 ±0.0005
32	0.05	6	0.74	0.15	8.4562	8.4574 ±0.0007	8.3745 ±0.0007
32	0.08	11	0.81	0.22	5.5623	5.5628 ±0.0004	5.5247 ±0.0003
32	0.12	18	0.90	0.23	3.9199	3.9202 ±0.0002	3.8833 ±0.0002
32	0.13	2	0.91	0.24	4.9148	4.9152 ±0.0006	4.8844 ±0.0008
32	0.14	14	0.93	0.10	2.7982	2.7982 ±0.0003	2.7636 ±0.0003
64	0.05	6	0.87	0.12	8.2524	8.2540 ±0.0007	7.9759 ±0.0008
64	0.05	10	0.71	0.07	8.4387	8.4401 ±0.0009	8.3144 ±0.0008
64	0.05	20	0.94	0.26	8.9138	8.9154 ±0.0004	8.7054 ±0.0005
64	0.09	3	0.94	0.06	4.6364	4.6368 ±0.0014	4.6015 ±0.0013
64	0.13	12	0.92	0.08	2.9317	2.9329 ±0.0003	2.8889 ±0.0003

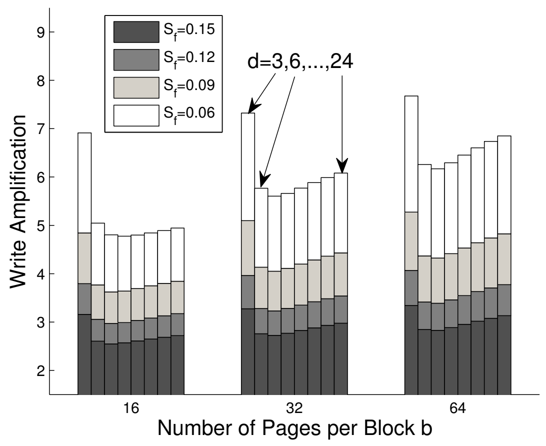
Table 2: Comparison of ODE-based results and simulation experiments for a system with $N = 50,000$ blocks for various parameter settings (25 runs). RND: ODE is optimistic, relative error less than 0.02%. OLDEST: ODE is pessimistic, relative error between 0.5% and 3.5%.

with $i^* = 0, 1$ and

$$\pi_{i^*,j^*}(\vec{m}) = \frac{1}{1-p_{0,0}(\vec{m})}\left[\sum_{j^+=1}^{j^*-1}\sum_{i^+=0}^{i^*}\pi_{i^+,j^+}(\vec{m})p_{i^*-i^+,j^*-j^+}(\vec{m}) + \right.$$
$$\left. +\frac{1}{b}\sum_{j^+=j^*}^{b}\sum_{i=i^*}^{b-j^++i^*} p_{i,b-j^++j^*}(\vec{m})\frac{\binom{i}{i^*}\binom{b-j^++j^*-i}{j^*-i^*}}{\binom{b-j^++j^*}{j^*}}\right], \tag{15}$$

for $0 \leq i^* \leq j^*$ and $j^* > 1$, meaning we can compute $\pi_{i^*,j^*}(\vec{m})$ recursively starting with $j^* = 1$. With this approach we can compute the ODE-based write amplification within a few seconds for $b = 32$, while the computation time is around 1 minute for $b = 64$ (the exact computation times depend on the parameter setting).

Table 2 compares the ODE-based results with two sets of simulation results: RND and OLDEST. The only difference between the ODE model and the RND results lies in the system size N, that is, the RND results were obtained by simulating the Markov chain $\{(\hat{M}^N(t), J(t)), t \in \mathbb{N}\}$ with $N = 50,000$ and we see a near perfect agreement with the ODE-based results. The OLDEST results correspond to a system were the choice of the pages that are moved from the selected block to the WFI is not random, but instead they are selected based on the order in which they were written to the selected block. In this case we see a 0.5% to 3.5% reduction of the write amplification. Note that $\{(\hat{M}^N(t), J(t)), t \in \mathbb{N}\}$ is not a Markov chain in this case as one needs to keep track of the order of the hot and cold pages in the WFI to obtain a Markov chain.



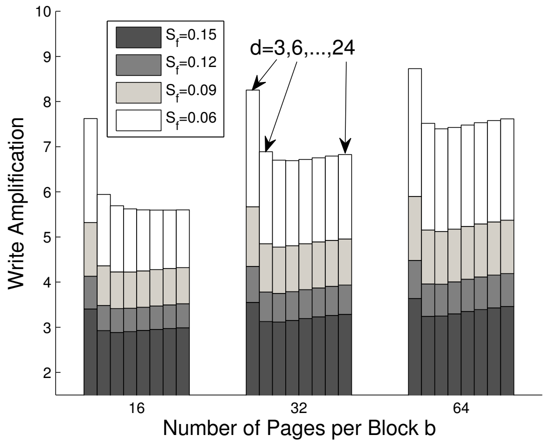


Figure 4: Write amplification A for a spare factor $S_f = 0.06, 0.09, 0.12$ and 0.15, for $b = 16, 32$ and 64 pages per block and for $d = 4, 8, \dots, 32$ choices with $r = 0.8$ and $f = 0.05$ (left) and $r = 0.9$ and $f = 0.20$ (right).

4.3 Numerical results

Figure 4 depicts the write amplification of the d-CHOICES GC algorithm for 4 different spare factors S_f, for $b = 16, 32$ and 64 and for $d = 3, 6, \dots, 24$ with $r = 0.8$ and $f = 0.05$ (left) and $r = 0.9$ and $f = 0.20$ (right). As expected, the write amplification increases as either S_f decreases or b increases. More importantly and in contrast to the system that operates with a single WF, there exists an optimal d value. In other words, the greedy strategy is no longer optimal in this setup.

To understand why being greedy is no longer optimal, consider two blocks, one with mostly hot data and one with mostly cold data. Further, assume the block with the cold data contains one or two more valid pages. A greedy strategy would select the block containing mostly hot data. However, it may be beneficial to select the other block first as the block containing mostly hot data may invalidate many more pages during the time that elapses until the other block is selected, thereby creating a reduction in the write amplification in the long run.

One might initially expect that blocks containing a small number of valid pages contain mostly cold data as in the system with a single WF, which implies that the above reasoning does not apply. However, under the current hot/cold data model, the ratio of hot versus cold data in a block created in the WFE is on average $r/(1-r)$ (as the block is initially empty and only external requests write data to the WFE). Under the Rosenblum model a tagged hot page is invalidated with a rate that is $r(1-f)/((1-r)f)$ times as large as a tagged cold page, hence the hot/cold page ratio in the block created by the WFE decreases as more pages are invalidated. However, unless the write amplification is very low, these blocks still contain a large proportion of hot data when

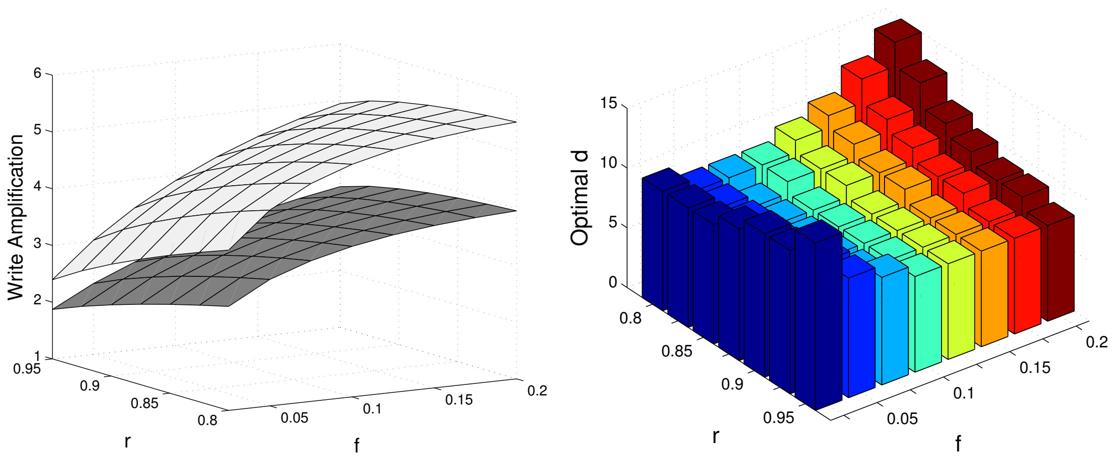


Figure 5: Left: Write amplification A as a function of r and f with d optimal for a spare factor $S_f = 0.08$ and 0.12, $b = 32$ pages per block. Right: Optimal d as a function of r and f for $S_f = 0.08$ and $b = 32$ pages per block.

selected by the GC algorithm. The ratio of hot versus cold data turns out to be much smaller in the blocks created by the WFI. The optimal d value provides the best balance between allowing the blocks created by the WFE to empty more, while trying to avoid the selection of blocks with lots of valid data.

Figure 5(left) depicts the write amplification as a function of r and f for $S_f = 0.08$ and 0.12 with $b = 32$ when the optimal d is used, while Figure 5(right) shows the corresponding optimal d (obtained by brute-force) for $S_f = 0.08$. It shows that the write amplification decreases significantly as the hot data gets hotter (i.e., if r increases or f decreases). The optimal d value is not a monotone function of r and f, but appears to have a fairly simple shape. Similar numerical experiments with other b and S_f values not depicted here, further show that

- Larger b values increase the write amplification under the optimal d value, while the optimal d value itself decreases slightly.
- Smaller spare factors increase the write amplification with d optimal (as expected) and the optimal d value also increases somewhat, except for very hot data.

5 Conclusions and open issues

We studied the write amplification in flash-based SSDs running the d-choices GC algorithm under non-uniform random writes. We showed that the write amplification *worsens* as the hot data gets hotter when a single write frontier is in place. Next, we introduced the double write frontier, which separates internally and externally requested writes, and showed that the write amplification now *reduces* significantly as the hot data gets hotter. Preliminary simulation experiments, not reported here, based on trace data collected in [16, 21] indicate that replacing the single write frontier by the double write frontier also reduces the write amplification significantly in case of real workloads.

A key characteristic of the d-choices algorithm studied in this paper is that it does not rely on any information with regard to the hot/cold nature of the data, meaning it does not require a data separation technique. In fact, the mean field models introduced in this paper can also be used to study a class of GC algorithms that do exploit such information. It might be interesting to see how the performance of the *blind* GC algorithms (when combined with a double write frontier) compare to the GC algorithms that use available hot/cold data information. Other possible future topics include the study of the impact of the TRIM command on the findings in this paper.

References

[1] A. Ban. Wear leveling of static areas in flash memory. US patent 6,732,221. Filed June 1, 2001; Issued May 4, 2004; Assigned to M-Systems., 2004.

[2] M. Benaïm and J. Le Boudec. A class of mean field interaction models for computer and communication systems. *Performance Evaluation*, 65(11-12):823–838, 2008.

[3] L. Bortolussi, J. Hillston, D. Latella, and M. Massink. Continuous approximation of collective system behaviour: A tutorial. *Performance Evaluation*, 70:317–349, 2012.

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

[5] F. Chen, D.A. Koufaty, and X. Zhang. Understanding intrinsic characteristics and system implications of flash memory based solid state drives. *ACM SIGMETRICS Perform. Eval. Rev.*, 37(1):181–192, 2009.

[6] T.S. Chung, D.J. Park, S. Park, D.H. Lee, S.W. Lee, and H.J. Song. A survey of flash translation layers. *Journal of Systems Architecture*, 55:332–343, 2009.

[7] P. Desnoyers. Analytic models of SSD write performance. *ACM Trans. Storage*, 10(2):8:1–8:25, March 2014.

[8] E. Gal and S. Toledo. Algorithms and data structures for flash memories. *ACM Computing Surveys*, 37:138–163, 2005.

[9] L. M. Grupp, J. D. Davis, and S. Swanson. The bleak future of NAND flash memory. In *Proc. of USENIX Conference on File and Storage Technologies*, 2012.

[10] J. Hsieh, T. Kuo, and L. Chang. Efficient identification of hot data for flash memory storage systems. *ACM Trans. on Storage*, 2:22–40, 2006.

[11] X. 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*, SYSTOR '09, pages 10:1–10:9, New York, NY, USA, 2009.

[12] S. Jung, Y. Lee, and Y. Song. A process-aware hot/cold identification scheme for flash memory storage systems. *IEEE Transactions on Consumer Electronics*, 56:339–347, 2010.

[13] Y. Li, P.P.C. Lee, and J.C.S. Lui. Stochastic modeling of large-scale solid-state storage systems: Analysis, design tradeoffs and optimization. *ACM SIGMETRICS Perform. Eval. Rev.*, 41(1):179–190, 2013.

[14] J. Menon. A performance comparison of RAID-5 and log-structured arrays. In *Proceedings of the 4th IEEE International Symposium on High Performance Distributed Computing*, HPDC '95, pages 167–178, Washington, DC, USA, 1995.

[15] C. Min, K. Kim, H. Cho, S. Lee, and Y. I. Eom. SFS: Random write considered harmful in solid state drives. In *Proc. of USENIX Conference on File and Storage Technologies*, pages 139–155, 2012.

[16] D. Narayanan, A. Donnelly, and A. Rowstron. Write off-loading: Practical power management for enterprise storage. *Trans. Storage*, 4(3):10:1–10:23, November 2008.

[17] D. Park and D. Du. Poster: Hot data identification for flash memory using multiple bloom filters. In *Proc. of USENIX Conference on File and Storage Technologies*, 2011.

[18] J.T. Robinson. Analysis of steady-state segment storage utilizations in a log-structured file system with least-utilized segment cleaning. *SIGOPS Oper. Syst. Rev.*, 30(4):29–32, October 1996.

[19] M. Rosenblum and J. K. Ousterhout. The design and implementation of a log-structured file system. *ACM Trans. Comput. Syst.*, 10(1):26–52, February 1992.

[20] B. Van Houdt. A mean field model for a class of garbage collection algorithms in flash-based solid state drives. *ACM SIGMETRICS Perform. Eval. Rev.*, 41(1):191–202, 2013.

[21] A. Verma, R. Koller, L. Useche, and R. Rangaswami. SRCMap: energy proportional storage using dynamic consolidation. In *Proceedings of the 8th USENIX conference on File and storage technologies*, FAST'10, pages 267–280, Berkeley, CA, USA, 2010.

[22] L. Xiang and B. Kurkoski. An improved analytical expression for write amplification in NAND flash. In *International Conference on Computing, Networking, and Communications (ICNC)*, pages 497–501, 2012.