ToMPECS 2022: impact factor 0.6

On the cost of near-perfect wear leveling in flash-based SSDs

BENNY VAN HOUDT
DEPT. OF COMPUTER SCIENCE, UNIVERSITY OF ANTWERP, BELGIUM

Wear leveling techniques in flash-based SSDs aim at distributing the erase cycles as uniformly as possible across the memory blocks within the SSD in order to extend its life span. The downside of any wear leveling technique is that it causes additional internal write operations, thereby increasing the so-called write amplification factor, which equals the ratio between the total number of writes performed and the number of writes requested by the host system.

In this paper we address the question whether near-perfect wear leveling is possible at low costs in terms of the write amplification factor. We answer this question affirmatively by presenting a simple randomized algorithm that combines wear leveling with garbage collection. This algorithm guarantees that the wear is nearly perfectly balanced at all times, while causing a low increase in the write amplification. This is demonstrated mathematically using a mean field model in case of uniform random writes and using trace-driven simulation experiments for general workloads.

ACM Reference Format:
Benny Van Houdt, Dept. of Computer Science, University of Antwerp, Belgium. 2022. On the cost of near-perfect wear leveling in flash-based SSDs. 1, 1 (December 2022), 22 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Wear leveling (WL) techniques in a flash-based SSD [9] are used to improve the life span of the device by making sure that the wear among the memory blocks on the device is more or less balanced, where the wear reflects the number of times that a block has been erased. In other words, WL techniques aim at distributing the erase cycles as uniformly as possible across the data blocks within the SSD. This is important as a memory block can only be erased a limited number of times (which can be as low as a few thousand [11, 15]) without losing the guarantee that the data stored on the block can be retained for a sufficiently long time. The downside of any WL technique is that it causes additional internal write operations, thereby increasing the so-called write amplification (WA) factor, which is defined as the ratio between the total number of writes performed by the SSD and the number of writes requested by the host system. In a recent paper [13] the increase in the WA factor was shown to be very substantial for a number of representative WL algorithms making the authors question the usefulness of wear leveling in modern SSDs.

Due to the limitations of flash-based memory, SSDs make use of out-of-place writes and a garbage collection (GC) algorithm that reclaims free space to support these out-of-place writes. The main objective of the GC algorithm is to select blocks with a low number of valid pages [4, 7] and to relocate the valid pages on a selected block to some other block before the block is erased. Memory blocks on an SSD contain a fixed number of pages and a copy of a page becomes invalid when it

Author's address: Benny Van Houdt
Dept. of Computer Science, University of Antwerp, Belgium.

XXXX-XXXX/2022/12-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

is rewritten on another block. Note that the GC algorithm aims a keeping the WA factor low by selecting memory blocks with a low number of valid pages.

In some sense the GC algorithms and WL techniques counteract each other. WL wants to erase all memory blocks equally often, but the GC algorithm is designed to select and erase blocks with a low number of valid pages. Moreover, to guarantee the existence of blocks with a low number of valid pages so-called hot and cold pages are separated (either explicitly using a hot/cold identification technique [6, 12, 18, 20] or implicitly by separating host requested writes from internal writes [23]). Hot pages are pages that are frequently rewritten and therefore invalidate quickly, while cold pages are mostly read and remain valid for long periods of time. Thus by grouping hot pages in blocks, one automatically creates blocks holding a low number of valid pages and these are frequently selected by the GC algorithm. Blocks containing mostly cold pages on the other hand are far less regularly selected as it takes a long time before its pages become invalid. Thus any WL technique needs to make sure that the cold pages also migrate regularly (e.g., by performing periodic swaps between blocks), but this clearly comes at the expense of an increase in the WA as demonstrated in [13] for a number of representative WL algorithms.

The question we address in this paper is:

Can near-perfect wear leveling be achieved at low costs in terms of the write amplification factor?

We answer this question in the affirmative by presenting a simple randomized algorithm that combines wear leveling with garbage collection. This algorithm achieves near-perfect wear leveling at all times by design, while we show both mathematically and using trace-based simulation experiments that it causes a low increase in the write amplification factor.

To measure the unevenness in the wear, we make use of the *program/erase (PE) fairness* [26]. It is defined as the ratio between the average number of erase operations performed on a block divided by the maximum number of erasures performed on a block. Hence the PE fairness lies between 0 and 1 and a PE fairness close to 1 indicates that the wear of the blocks in the systems is well balanced. The PE fairness affects the *SSD endurance* which corresponds to the total number of full drive writes that can be performed before a memory block is erased for the $(W_{max} + 1)$-th time, where W_{max} is the number of erasures that a block can tolerate without losing the guarantee that the data on a block can be retained sufficiently long. Note that the SSD endurance can be expressed as W_{max} times the PE fairness divided by the WA factor. When the PE fairness is close to one, the SSD endurance is close to W_{max}/WA and is upper bounded by W_{max}.

In this paper we focus on the setting where hot and cold data is separated implicitly (by separating host requested writes from internal writes), as this avoids the complicated task of explicitly labeling pages as hot or cold. In fact such an implicit hot and cold data separator often performs nearly as well as an explicit separator [24]. The algorithm presented in this paper can be regarded as an advanced d-choices GC algorithm that guarantees near-perfect wear leveling. The d-choices GC algorithm [22] selects blocks by picking a block with the least number of valid pages among d randomly selected blocks. Even for moderate values of d, the d-choices GC performs very similar to the Greedy GC algorithm that always selects the memory block holding the least number of valid pages [23, 28].

The algorithm presented has near-perfect wear leveling at all times by design. To demonstrate that it causes a low increase in the WA factor we use two approaches. First, we assess its performance using a mean field model for the setting with uniform random writes (i.e., without hot and cold data). While uniform random writes hardly even occur in practice, the Greedy GC algorithm is known to minimize the WA factor this setting [28], which means we can precisely compute how much larger the WA factor of the presented algorithm is compared to the optimal WA factor. Second, we perform simulation experiments for trace-based workloads that demonstrate that for real workloads the WA factor of the presented algorithm is quite close to that of the vanilla d-choices GC algorithm (which causes very unbalanced wear in the absence of a wear leveling technique [26]).

The paper is structured as follows. In Section 2 we present some background on flash-based SSDs. The algorithm that has near-perfect wear by design is outlined in Section 3, while its performance under uniform random writes is studied using a mean field model in Section 4, Some of the more technical details of this model are deferred to Section 5. Section 6 presents some simulation experiments using trace-based workloads. Conclusions are drawn in Section 7.

2 FLASH-BASED SSD BACKGROUND

Modern SSDs make use of several channels each connected to a small number of flash chips. Each flash chip in turn may accommodate multiple dies, each consisting of a few planes. Finally, a single data plane contains thousands of blocks each holding a fixed number b of pages, e.g., $b = 64$ pages per block. Data reads and writes are performed at the granularity of flash pages. In order to write data on a page, it must be in the *erase* state first. However, pages cannot be erased individually, only entire blocks can be erased. As erasing an entire block and restoring all the data on the block for each page update would make the device very slow, SSDs use out-of-place writes: whenever data is written to a page, the old data is simply marked as *invalid* and the new data is written elsewhere and marked as *valid*. Hence, any page is either in the erase, valid or invalid state at any point in time [5]. Page writes are also called program operations and therefore the number of erase operations performed on a block is also called the number of program-erase (PE) cycles. In this paper we use the terms PE cycles and erasures interchangeably.

A possible design is to map logical page addresses to planes in a deterministic manner, that is, the channel, chip, die and plane numbers are fully determined by the logical address [21]. Within a plane writes are performed in an out-of-place manner. Therefore the flash translation layer (FTL) maintains a table that maps logical page numbers to physical page numbers for each plane. However, recent SSDs do not have such restrictions and may use a single map for the full logical address space in order to maximize resource efficiency.

The garbage collection algorithm is responsible for selecting the blocks that are erased and subsequently used to store new data. Garbage collection is only effective if some of the blocks contain only a limited number of valid pages. To guarantee the existence of such blocks SSDs rely on over-provisioning, meaning the physical storage capacity of the device, say N blocks, exceeds the logical capacity, say U blocks, such that at least a fraction $S_f = 1 - U/N$ (e.g., 0.1), called the spare factor, of the total number of pages is not in the valid state. For further use we denote $\rho = 1 - S_f = U/N$.

A flash cell can correctly hold stored data for a limited amount of time, called the retention time. If a block is not rewritten before the retention time, its data can no longer be retrieved correctly. The retention time of a block decreases as the number of PE cycles performed on it increases. For instance, in order to guarantee retention for as long as 3 years, the number of PE cycles on a block may be limited to as few as 3K [15]. Thus to prolong the lifetime of the device a wear leveling technique is used to balance the number of erasures performed among the blocks.

A natural choice for the GC algorithm is the Greedy GC algorithm. This GC algorithm selects a victim block that currently holds the least number of valid pages among all blocks. In fact this algorithm is known to minimize the WA under uniform random writes [28] and its performance was shown to be close (within 5%) to an offline near-optimal GC algorithm for 10 MSR traces in case only a *single* block is used to write the new data [19]. However, in the presence of hot and cold data the WA can be greatly reduced (by more than 50%) by relying on *multiple* blocks to store the new data [19, 23].

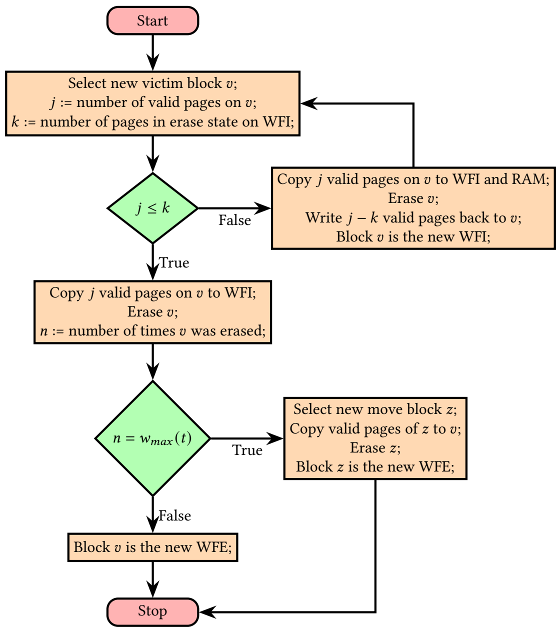

Start
Select new victim block v;
j := number of valid pages on v;
k := number of pages in erase state on WFI;
$j \leq k$
False
Copy j valid pages on v to WFI and RAM;
Erase v;
Write $j - k$ valid pages back to v;
Block v is the new WFI;
True
Copy j valid pages on v to WFI;
Erase v;
n := number of times v was erased;
$n = w_{max}(t)$
True
Select new move block z;
Copy valid pages of z to v;
Erase z;
Block z is the new WFE;
False
Block v is the new WFE;
Stop

Fig. 1. Flow diagram of the randomized GC algorithm

3 A SIMPLE RANDOMIZED ALGORITHM

In this section we introduce a simple randomized garbage collection algorithm such that the number of erase cycles performed on any two blocks differs by at most Δ_w at all times, where Δ_w is an input parameter. This means that the algorithm has a guaranteed PE fairness of at least $1 - \Delta_w/W_{max}$, where W_{max} is the maximum number of erasures a block can tolerate. Thus, if we pick $\Delta_w = W_{max}/100$ we get a PE fairness of at least 0.99.

At any point in time our algorithm marks two blocks (initially selected at random) and these two blocks are used to support write operations. One of these two blocks is called the write frontier for external writes (WFE) and the other the write frontier for internal writes (WFI). Any write requested by the host system is written to the WFE and changes the state of one of its pages from the erase state to the valid state. Internal writes triggered by the GC algorithm are written to the WFI. Whenever the WFE becomes full (meaning none of its pages is in the erase state) the GC algorithm is triggered to select a new WFE.

For every block a we maintain an erase counter c_a, which counts the number of PE cycles performed thus far on block a. Our algorithm guarantees that the number of PE cycles performed thus far on any two blocks differs by at most Δ_w. Thus, if $w_{min}(t)$ reflects the least number of PE cycles performed on any block at time t, then all blocks experienced at most $w_{max}(t) = w_{min}(t) + \Delta_w$ at time t. If we pick Δ_w such that it is of the form $2^k - 1$ for some k, then $k = \log_2(\Delta_w + 1)$ bits per block suffice to maintain these erase counters as only the offset with respect to $w_{min}(t)$ matters. In our experiments we typically set $\Delta_w = 31$ or 63 and therefore we only require $k = 5$ or 6 bits per block to maintain these erase counters.

We now discuss the manner in which our randomized algorithm operates. A flow diagram of the algorithm is presented in Figure 1. The algorithm is triggered when the WFE becomes full (due to a write requested by the host system). Assume this happens at time t. Our algorithm starts by selecting d blocks at random[1] among all the blocks with an erase counter strictly less than $w_{max}(t)$. The block v that contains the least number of valid pages among these d blocks is called the *victim* block (ties are broken at random). Thus, contrary to the classic d-choices GC algorithm [22], all the blocks with an erase counter equal to $w_{max}(t)$ are excluded from becoming the victim block. Suppose that the victim block contains j valid pages. We consider two cases.

(1) If there are $k \geq j$ pages left on the WFI that are in the erase state, the j valid pages of the victim block v are copied to the WFI and subsequently the victim block is erased. As a result the number of pages in the erase state on the WFI reduces by j (as these pages became valid) and the erase counter of the victim block increases by one. As the value of erase counter of the victim block was at most $w_{max}(t) - 1$, it remains upper bounded by $w_{max}(t)$.

   (1A) If the number of erasures on the victim block v remained below $w_{max}(t)$, the victim block becomes the new WFE and contains b pages in the erase state.

   (1B) If the number of erasures on the victim block v became $w_{max}(t)$ an additional *move* operation is performed. More specifically, our algorithm selects d^* blocks at random that have been erased exactly $w_{min}(t)$ times [2]. The block z that contains the *most* number of valid pages among these d^* blocks is termed the move block (ties are broken at random). Next, all the valid pages of the move block z are written to the victim block v, the move block z is erased and its erase counter is increased to $w_{min}(t) + 1$. In this case the move block z becomes the new WFE. Note that all the valid pages on the move block can be copied to the victim block v as block v was erased just prior to the move.

(2) If there are fewer than j pages left on the WFI that are in the erase state, say $k < j$, then our algorithm first copies the first k of the j valid pages of the victim block to the WFI and temporarily stores the remaining $j - k$ valid pages (in RAM). Next our algorithm erases the victim block v, copies the remaining $j - k$ valid pages back to the victim block and labels the victim block as the new WFI (that now contains $j - k$ valid pages and $b - (j - k)$ pages in the erase state). In this case our algorithm is immediately executed again in search of a new WFE.

Note that if during one of these two cases the last block with an erase counter equal to $w_{min}(t)$ got erased, $w_{min}(t)$ increases by one (and so does $w_{max}(t)$ by definition).

There are two important reasons for implementing the move operation in Case (1B).

(1) In order to limit the write amplification, blocks with a low number of valid pages should preferably have fewer than $w_{max}(t)$ erasures (as such a block cannot be selected as a victim block). As hot data tends to become invalid quickly, we want to avoid that hot data is written on a block that experienced exactly $w_{max}(t)$ PE cycles. In other words, we want to avoid that the WFE has an erase counter equal to $w_{max}(t)$. The move operation does exactly this as it makes sure that the new WFE has an erase counter equal to $w_{min}(t) + 1$ instead of $w_{max}(t)$. The idea of selecting a move block with many valid pages is based on the fact that after the move, the victim block has an erase counter equal to $w_{max}(t)$ and should therefore contain as much cold data as possible. Thus although moving the valid pages of a block with the most number of valid pages is more expensive, it gives us the best odds that cold data is moved instead of hot data.

[1]In the (unlikely) event that there are only $c < d$ such blocks left, we simply select these c blocks.
[2]If there are only $c < d^*$ blocks with exactly $w_{min}(t)$ PE cycles, these c blocks are selected.

(2) It is important to limit the number of blocks with $w_{max}(t)$ PE cycles as much as possible (as these cannot be selected as a victim block). Without the move operations the number of blocks left with less than $w_{max}(t)$ erasures before the last block with $w_{min}(t)$ erasures is selected may be very low. In fact this will often be the case as without the move operation, the last blocks with $w_{min}(t)$ erasures contain many valid pages and are therefore unlikely to be selected as a victim block (unless d is very small). This also explains why selecting a block with many valid pages as a move block is useful. Not only does it increase our chance to move cold data, as explained before, it also guarantees that far more blocks with less than $w_{max}(t)$ erasures remain when the last blocks with exactly $w_{min}(t)$ erasures are selected. In short, without the move operations there will be a period of high write amplification whenever the value $w_{min}(t)$ is about to increase by one.

Yet another way to think about this algorithm is to consider the following thought experiment. Suppose we have a host system that repeatedly writes the same sequence $\mathcal{S}$ of b logical pages (mixed with read operations). Assume the sequence $\mathcal{S}$ is initially placed on the same block, say block x_1, there is a single free block, say block x_2, and all the other blocks contain b valid pages that are read only. Further assume all erase counters are initially equal to zero and d is equal to the number of blocks on the device (meaning we select the victim in a greedy manner among all the blocks that have an erase counter below $w_{max}(t)$). To achieve a high PE fairness in such a system, the sequence $\mathcal{S}$ should migrate regularly from one block to the next. However, in order to write these b pages on a block with read only data, the read only data must be moved, which causes internal writes and thus write amplification.

With the above algorithm the sequence $\mathcal{S}$ is first written $2(\Delta_w - 1)$ times in an alternating fashion between block x_1 and x_2. At this point block x_1 contains the sequence $\mathcal{S}$, block x_2 is free and both blocks have been erased $\Delta_w - 1$ times. After the sequence $\mathcal{S}$ moves to block x_2 for the Δ_w-th time, GC is triggered and a new block x_3 with an erase counter equal to zero is selected as the move block and becomes the new WFE (while its read only data is moved to block x_1). Next the sequence $\mathcal{S}$ is written to block x_3, GC is triggered and block x_2 becomes the victim block. Therefore x_2 is erased for the Δ_w-th time and another move is performed that moves the read only data from some block x_4 to x_2 and now x_4 is free and becomes the new WFE. This process is repeated where blocks x_3 and x_4 take the roles of x_1 and x_2. In this manner we only need to perform 2 expensive move operations every $2\Delta_w$ times that the sequence $\mathcal{S}$ is written. This implies that we have a write amplification of only $(2\Delta_w b + 2b)/(2\Delta_w b) = 1 + 1/\Delta_w$.

4 UNIFORM RANDOM WRITES

In this section we introduce a mean field model to assess the write amplification factor of the GC algorithm with guaranteed PE fairness presented in Section 3 under uniform random writes. Uniform random writes means that all the logical pages are written equally likely and there is no correlation between the logical page numbers of consecutive writes. For uniform random writes the greedy garbage collection algorithm (that always selects a block with as few valid pages as possible among all the blocks) is known to minimize the write amplification [28]. While assuming uniform random writes is restrictive from a practical point of view, the main insights provided by this mean field model turn out to be in good agreement with the trace-based simulation experiments presented in Section 6. This mean field model also provides us with the ability to detect very minor changes in the system performance as it is not hampered by the noise present in simulation experiments.

We focus on the write amplification as the PE fairness is guaranteed to be close to one and the endurance is close to W_{max} divided by the write amplification if the PE fairness is close to one. Nevertheless we also present some numbers for the PE fairness. In Section 4.1 we present the mean field model, while Section 4.2 discusses the changes required to the model when modifying some aspects of the randomized algorithm. The model is validated in Section 4.3, where we also discuss convergence. Finally, numerical results and insights are presented in Section 4.4.

4.1 Mean field model

The mean field model presented next is inspired by the mean field models introduced in [22, 23, 26], but bears some significant differences as discussed in Section 4.3. The model is characterized by a set of ordinary differential equations (ODEs) that describe the evolution of some variables over time. In our case the variables are $m_{i,w}(t) \in [0,1]$ for $i = 0, \ldots, b$ and $w = w_{min}(t), \ldots, w_{max}(t)$, where $m_{i,w}(t)$ represents the fraction of all the memory blocks at time t holding i valid pages that have been erased w times. If we collect these variables in a vector

$$\vec{m}(t) = (m_{0,0}(t), m_{1,0}(t), \ldots, m_{b,0}(t), m_{0,1}(t), m_{1,1}(t), \ldots, m_{b,1}(t), m_{0,2}(t) \ldots),$$

the mean field model is characterized by a set of ODEs of the form:

$$\frac{d}{dt}\vec{m}(t) = F(\vec{m}(t)),$$

for some drift function F with component functions $f_{i,w}$ with $i = 0, \ldots, b$ and $w = w_{min}, \ldots, w_{max}$, that is, $\frac{d}{dt}m_{i,w}(t) = f_{i,w}(\vec{m}(t))$. After obtaining an expression for the drift functions $f_{i,w}$ we assess the performance of the algorithm presented in Section 3 by numerically solving this set of ODEs on the interval $(0, T)$, where T is the smallest real number such that $w_{max}(T) = W_{max}$.

The drift functions $f_{i,j}$ are the fluid limits of the discrete time model that observes the state of the memory blocks whenever a call to the GC algorithm takes place. When such a call takes place the WFE is full, but the WFI may contain some valid pages. We denote the state of the WFI as

$$(j^*, w^*) \in \{1, \ldots, b\} \times \{w_{min} + 1, \ldots, w_{max}\},$$

where j^* is the number of valid pages and w^* the number of erasures performed on the WFI. As the operation of the algorithm depends on the state of the WFI, the evolution of the variables $m_{i,w}$ depends not only on $\vec{m}(t)$, but also on the state (j^*, w^*) of the WFI.

The mean field model corresponds to a rescaled process of the discrete time Markov chain that observes the system whenever a call to the GC algorithm is made. It is clear that the WFI makes one transition per time unit in this discrete time Markoc chain. However, if we tag a random block and look at its evolution, the probability that the state of this tagged block changes during a single transition scales as $1/N$ due to the manner in which the victim block is selected, where N is the number of blocks. Thus the intensity of the transitions of the tagged block vanishes as N tends to infinity. This suggests that a time scale separation argument can be used [3]. Hence, the drift functions $f_{i,w}$ can written in terms of the steady state probabilities $\pi_{j^*,w^*}(\vec{m}(t))$ of the state of the WFI given $\vec{m}(t)$. More precisely,

$$f_{i,w}(\vec{m}(t)) = \sum_{j^*,w^*} \pi_{j^*,w^*}(\vec{m}(t)) f_{i,w}(\vec{m}(t), j^*, w^*),$$

where $f_{i,w}(\vec{m}(t), j^*, w^*)$ is the conditional drift function of $m_{i,w}$ given that the WFI is in state (j^*, w^*). In Section 5 we prove that

$$\sum_{w^*=w_{min}(t)+1}^{w_{max}(t)} \pi_{j^*,w^*}(\vec{m}(t)) = 1/b,$$

which means that *the number of valid pages j^* on the WFI has a uniform distribution.* In the remainder of this section we present the resulting drift functions $f_{i,w}$ *given that j^* has a uniform distribution.* The derivation is split in two steps: (1) we derive an expression for the probabilities $p_{i,w}(\vec{m})$ that the GC algorithm selects a victim block with i valid pages that has been erased w times, given that the system is in state $\vec{m}$ and (2) we express the drift functions $f_{i,w}(\vec{m})$ in terms of the probabilities $p_{i,w}(\vec{m})$.

Step 1: the probabilities $p_{i,w}(\vec{m})$. Let $p_{i,w}(\vec{m})$ be the probability that the GC algorithm selects a victim block with i valid pages that has been erased w times given that the occupancy vector equals $\vec{m} = (m_{0,0}, m_{1,0}, \ldots, m_{0,b}, m_{0,1}, m_{1,1}, \ldots, m_{0,1}, \ldots)$. Let $w_{min}(\vec{m})$ be the smallest w for which $m_{i,w} > 0$ for some i, that is, $w_{min}(\vec{m}) = \min\{w | \sum_{i=0}^{b} m_{i,w} > 0\}$. Define $w_{max}(\vec{m}) = w_{min}(\vec{m}) + \Delta_w$. By design, we have $m_{i,w} = 0$ for $w > w_{max}(\vec{m})$ and $w < w_{min}(\vec{m})$. To ease the notation, we simply refer to $w_{min}(\vec{m})$ and $w_{max}(\vec{m})$ as w_{min} and w_{max}, respectively. Further define $p_i(\vec{m}) = \sum_{w=w_{min}}^{w_{max}-1} p_{i,w}(\vec{m})$. Note that $p_i(\vec{m})$ is the probability that the GC algorithm selects a victim block containing exactly i valid pages. It can be computed as

$$p_i(\vec{m}) = \frac{\left(\sum_{\ell=i}^{b} \sum_{w=w_{min}}^{w_{max}-1} m_{\ell,w}\right)^d - \left(\sum_{\ell=i+1}^{b} \sum_{w=w_{min}}^{w_{max}-1} m_{\ell,w}\right)^d}{\left(\sum_{\ell=0}^{b} \sum_{w=w_{min}}^{w_{max}-1} m_{\ell,w}\right)^d}, \tag{1}$$

as each of the d randomly selected blocks have an erase counter below w_{max}, must contain at least i valid pages and not all of them contain more than i valid pages. The probability $p_{i,w}(\vec{m})$ can then be expressed as

$$p_{i,w}(\vec{m}) = \frac{m_{i,w}}{\sum_{w=w_{min}}^{w_{max}-1} m_{i,w}} p_i(\vec{m}), \tag{2}$$

as the GC algorithm selects the block with the least number of valid pages among the d randomly selected blocks without taking the erase counter values into account.

Step 2: the drift functions $f_{i,w}(\vec{m})$. The drift $f_{i,w}(\vec{m})$ indicates how $m_{i,w}$ changes over time when the occupancy vector equals $\vec{m}$. We write the drift as the sum of 3 terms:

$$f_{i,w}(\vec{m}) = f_{i,w}^{(nomove)}(\vec{m}) + f_{i,w}^{(move,1)}(\vec{m}) + f_{i,w}^{(move,2)}(\vec{m}).$$

The first term represents the drift in case that we never perform the move operation in Step (1B) of the algorithm. The two remaining terms take care of the changes made to $m_{i,w}$ due to the move operation.

We now derive the term $f_{i,w}^{(nomove)}(\vec{m})$ which has a similar form as the drift functions in [26]. First note that $im_{i,w}/b(1 - S_f)$ is the probability that a random write invalidates a page on a memory block with i valid pages and w erasures (as a memory block contains $b(1 - S_f)$ valid pages on average). Thus a single random write causes a drift of

$$\frac{(i+1)m_{i+1,w} - im_{i,w}}{b(1 - S_f)}$$

to $m_{i,w}$. The number of random writes that take place in between two calls to the GC algorithm equals b times the probability that all the pages on the victim block can be transferred to the WFI. We now use the fact that the number of valid pages on the WFI has a uniform distribution. Therefore, a victim block holding $b - j$ valid pages can copy all of its valid pages to the WFI with probability j/b. Hence, the mean number of writes between two GC calls equals b times $\sum_{j=0}^{b} p_{b-j}(\vec{m}) \frac{j}{b}$. The mean field model is the fluid limit of the discrete system that observes the system just prior to a GC call, therefore the random writes cause a total drift of

$$\frac{(i+1)m_{i+1,w} - im_{i,w}}{b(1 - S_f)} \left(\sum_{j=1}^{b} p_{b-j}(\vec{m}) j\right),$$

to $m_{i,w}$. Finally, $m_{i,w}$ decreases when the GC algorithm selects a block with i valid pages and w erasures and increases if a victim block was selected with $w-1$ erasures provided that $i=b$ (as the victim block is full when the next GC call occurs). Hence,

$$f_{i,w}^{(nomove)}(\vec{m}) = \frac{(i+1)m_{i+1,w} - im_{i,w}}{b(1-S_f)} \left(\sum_{j=1}^{b} p_{b-j}(\vec{m})j\right) \\ - p_{i,w}(\vec{m}) + 1[i=b, w>w_{min}] \sum_{\ell=0}^{b} p_{\ell,w-1}(\vec{m}). \quad (3)$$

To specify the terms in $f_{i,j}(\vec{m})$ related to the move operation, we first define $p_{move}(\vec{m})$ which represents the probability that a GC call requires a move operation given that the occupancy measure equals $\vec{m}$. It is clearly given by

$$p_{move}(\vec{m}) = \sum_{\ell=0}^{b} \frac{b-\ell}{b} p_{\ell,w_{max}-1}(\vec{m}), \quad (4)$$

as a move occurs when a block with exactly $w_{max}-1$ erasures is chosen as the victim block and the ℓ valid pages on the victim can be copied to the WFI (where we again rely on the fact that the number of valid pages on the WFI is uniformly distributed). The first term due to the presence of the move is defined as

$$f_{i,w}^{(move,1)}(\vec{m}) = (1[w=w_{min+1}] - 1[w=w_{max}])p_{move}(\vec{m})1[i=b]. \quad (5)$$

This term is due to the fact that after a move operation the WFE is a block with $w_{min}+1$ erasures as opposed to being a block with w_{max} erasures. The next term captures the change due to the valid pages that are moved from the move to the victim block:

$$f_{i,w}^{(move,2)}(\vec{m}) = (1[w=w_{max}] - 1[w=w_{min}])p_{move}(\vec{m})q_i^{most}(\vec{m}), \quad (6)$$

where $q_i^{most}(\vec{m})$ represents the probability that the move block contains exactly i valid pages. This probability equals

$$q_i^{most}(\vec{m}) = \frac{\left(\sum_{\ell=0}^{i} m_{\ell,w_{min}}\right)^{d^*} - \left(\sum_{\ell=0}^{i-1} m_{\ell,w_{min}}\right)^{d^*}}{\left(\sum_{\ell=0}^{b} m_{\ell,w_{min}}\right)^{d^*}}, \quad (7)$$

as the move block contains the most number of valid pages among d^* randomly selected blocks with exactly w_{min} erasures.

These mean field equations are used as follows to determine the write amplification of the proposed algorithm. We consider the system up to the point where the value of w_{max} reaches some predefined value W_{max}. The mean field approach exists in numerically solving the set of equations $\frac{d}{dt}m_{i,w}(t) = f_{i,w}(\vec{m}(t))$ given $\vec{m}(0)$, which reflects the occupancy measure at time 0 (i.e., the initial state of the drive). We use a simple Euler iteration to solve these equations except that some extra care is needed whenever we hit a surface where $\sum_{\ell=0}^{b} m_{\ell,w_{min}}$ hits zero.

The write amplification can be expressed as E_{tot}/E_{host}, where E_{tot} is the mean number of host and internal writes that is performed per GC call and E_{host} is the mean number of host writes that can be supported per GC call. If the GC algorithm selects a victim block at time t with less than $w_{max}-1$ erasures, this block is eventually filled with b host or internal writes (either as WFE or WFI). With probability $p_{move}(m(t))$ the victim block was erased exactly $w_{max}-1$ times and is filled with $\sum_{i=1}^{b} iq_i^{most}(m(t))$ internal page writes on average, while the move block is filled with b host

b	d	d^*	ρ	Δ_w	mean field	simulation
16	50	2	0.9	7	4.3198	4.3195 ± 0.2030 E-3
16	10	10	0.9	15	4.3864	4.3859 ± 0.1208 E-3
32	5	30	0.9	31	5.1335	5.1326 ± 0.1320 E-3
32	50	30	0.8	63	2.5237	2.5242 ± 0.0543 E-3
64	10	5	0.85	15	3.5176	3.5185 ± 0.2835 E-3
64	20	3	0.88	7	4.2875	4.2888 ± 0.2156 E-3

Table 1. Validation of the mean field model using simulation (averaged over 5 runs) for a system with $N = 10,000/\rho$ blocks and $W_{max} = 2000$. Relative errors well below 1% are observed in all cases.

writes. Further, E_{host} is simply the mean number of pages on the victim block that are not in the valid state. Hence, the write amplification can be computed as

$$\frac{b + \frac{1}{T_{W_{max}}} \int_0^{T_{W_{max}}} p_{move}(m(t)) \left(\sum_{i=1}^b i q_i^{most}(m(t))\right) dt}{b - \frac{1}{T_{W_{max}}} \int_0^{T_{W_{max}}} \left(\sum_{i=1}^b i p_i(m(t))\right) dt}, \tag{8}$$

where T_x is the smallest t value for which $m_{i,x+1}(t)$ becomes larger than zero for some i.

4.2 Model variations

The mean field model introduced in Section 4.1 can be easily modified to investigate the impact of a number of algorithm design choices. For instance, to understand what the impact of the move operation is, we can simply set $p_{move}(\vec{m})$ equal to zero. To see what happens if we adapt the move operation such that a block with the least number of valid pages and exactly w_{min} PE cycles is selected it suffices to use $q_i^{least}(\vec{m})$ instead of $q_i^{most}(\vec{m})$ with

$$q_i^{least}(\vec{m}) = \frac{\left(\sum_{\ell=i}^b m_{\ell,w_{min}}\right)^{d^*} - \left(\sum_{\ell=i+1}^b m_{\ell,w_{min}}\right)^{d^*}}{\left(\sum_{\ell=0}^b m_{\ell,w_{min}}\right)^{d^*}}. \tag{9}$$

Another interesting question is to see how the performance alters when we use a single WF (instead of the WFI/WFE). With a single WF all host writes are written to the WF. If the WF is full, the GC selects a victim block (with at most $w_{max} - 1$ erasures), copies its valid pages to memory, erases the block and places the valid pages back on the victim block unless the erase counter of the victim equalled $w_{max} - 1$. In the latter case a move operation is performed as in Case (1B) of our algorithm. Regarding the mean field model, the only required change is that $p_{move}(\vec{m})$ must be replaced by

$$p_{move}^{singleWF}(\vec{m}) = \sum_{\ell=0}^{b} p_{\ell,w_{max}-1}(\vec{m}),$$

as we always perform a move operation whenever a block is selected with exactly $w_{max} - 1$ erasures. Note that without the move operation the model with the single WF and WFI/WFE are identical and therefore so is the performance in case of uniform random writes (as in [22]).

4.3 Convergence and model validation

A first difference between the model in this paper and those in [22, 23] is that we are interested in the system behavior over a finite time interval (that is, until a block is erased for the W_{max} + 1-th time) and not in a fixed point. In fact the current model does not have a fixed point even if we only work with the offset of w with respect to w_{min}, that is, make use of the variables $v_{i,w} = m_{i,w+w_{min}}$ with $w = 0, \dots, \Delta_w$. Typically convergence towards the mean field limit is easier to prove over finite time scales as there is no need to prove global attraction of the fixed point.

The fact that the mean field equations capture the proper limit behavior as the number of blocks N on the SSD device tends to infinity as was proven in [22, 23] using the framework in [3] over finite time scales. However, contrary to these models, the drift functions $f_{i,w}(\vec{m})$ of our model are not a smooth function in $\vec{m}$ due to their dependence on w_{min} which occasionally jumps up by one whenever $\sum_{\ell=0}^{b} m_{\ell,w_{min}}$ hits zero. As a result assumptions H4 and H5 in [3] do not hold. These two assumptions demand that the drift equations are smooth in $\vec{m}$ (this is H5), and the transition probabilities of the so-called *resource* in [3] are also smooth in $\vec{m}$ (this is H4). The latter corresponds to the WFI state in our case (see (14)). Although traditional mean field results such as Kurtz' theorem [8] have been generalized to settings with discontinuous drifts [10], no such relaxations for assumptions H4 and H5 in [3] have appeared in the literature. In fact, examples of mean field models can be constructed where convergence does not occur if the smoothness of assumptions H4 and H5 in [3] are violated. This implies that proving convergence is challenging and requires model specific arguments. Intuitively it seems that system can be decoupled in periods where the drift is Lipschitz continuous (when $w_{min}(t)$ is ϵ bounded away from 0) and periods that contain the discontinuities. The duration of the latter periods should be small/negligible compared to the length of the Lipschitz continuous periods, which suggests that the convergence to the mean field model is likely to remain valid. It is however unclear how to formalize such an argument. As such coming up with a formal proof that the mean field model captures the proper limit behavior remains an open problem.

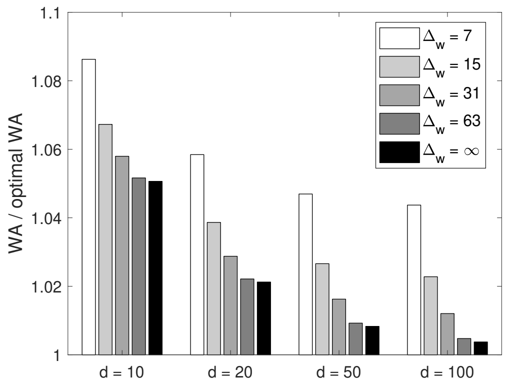
In the absence of a convergence proof, we illustrate the accuracy of the mean field model using simulation instead. We simulate a system consisting of N blocks and a workload consisting of uniform random writes. Hence the simulated system is identical to the mean field model, except that it considers a finite number of blocks N. More specifically we simulate a system with $10000 \times b$ logical pages and thus $N = 10000/\rho$ physical blocks. The number of erasures that a block can tolerate W_{max} was set equal to 2000. Further increasing this value has no noticeable impact on the results. The time until the first block reached 500 erasures was used as a warm up period. When computing the mean field write amplification we therefore used (8), but integrated from time T_{500} until T_{2000} instead of integrating from time $T = 0$. The logical pages were initially placed at random random over the Nb physical pages in the simulation, while for the mean field model the initial state was given by $m_{i,0} = \binom{b}{i}\rho^i(1-\rho)^{b-i}$ (that is, the number of valid pages on a block is binomially distributed with parameters (b, ρ)).

Table 1 presents the comparison for a number of arbitrarily chosen parameter settings. In all cases we observe a relative error well below one percent. The accuracy even further improves when increasing the system size (i.e., number of blocks). The write amplification values listed in this table may appear to be quite high, but this is merely due to the uniform random writes. For instance the greedy algorithm, that minimizes the write amplification under uniform random writes, has a write amplification of 3.9814 for $b = 16$ and $\rho = 0.9$ and of 2.5136 for $b = 32$ and $\rho = 0.8$.

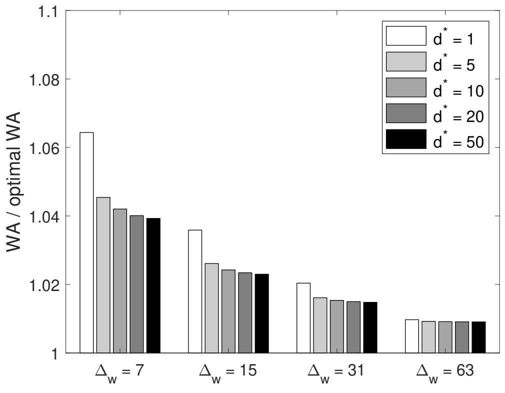
4.4 Mean field numerical results

In this section we present various numerical results using the mean field model. The main insight gained from these experiments is:

- Under uniform random writes the proposed algorithm in Section 3, with d sufficiently large, has a write amplification that is close to optimal for moderate values of Δ_w, e.g., $\Delta_w = 31$ or 63. This indicates that it is indeed possible to achieve near-perfect wear without significantly



(a) $d^* = 5$, $b = 32$ and $S_f = 0.1$.



(b) $d = 50$, $b = 32$ and $S_f = 0.1$.

Fig. 2. Impact of d (left) and d^* (right) on the write amplification under uniform random writes for $\Delta_w = 7, 15, 31$ and 63.

increasing the WA factor, answering our initial question affirmatively in case of uniform random workloads.

In addition the following observations are made:

- We show that the WA factor is fairly insensitive to the choice of the parameter d^* (the number choices made during the move operation).
- We demonstrate that the increase in the WA factor is much more substantial if we make use of only a single WF (instead of working with the WFI and WFE) or if we select the move block by picking the block with the least number of valid pages among the d^* blocks during the move operation.

We consider systems with $\Delta_w = 7, 15, 31$ and 63, which corresponds to having a 3, 4, 5 and 6 bit erase counter, respectively. We present results for $b = 32$ and $\rho = 0.9$, i.e., $S_f = 0.1$. Similar observations are made for other b and ρ values. In our plots we normalized the write amplification by the write amplification of the greedy GC algorithm (which is optimal for uniform random writes, but does not offer any guarantees in terms of the PE fairness).

Figure 2a depicts the impact of d and Δ_w on the normalized WA when $d^* = 5$. We also included results for $\Delta_w = \infty$, which corresponds to using the classic d-choices GC algorithm. We note that in all cases the relative increase in the WA compared to the optimal algorithm is below 10% and even below 5% if we set $d = 100$. We also note that the results for $\Delta_w \geq 31$ are less than one percent above the results with $\Delta_w = \infty$. Recall that setting $\Delta_w = 31$ with $W_{max} = 2000$ implies that the PE fairness is lower bounded by $1 - \Delta_w/W_{max} = 0.9845$. In other words we can guarantee a very high PE fairness with only a very limited increase in the WA. For completeness the PE fairness values are listed in Table 2. Note that even the classic d-choices GC algorithm (which corresponds to setting $\Delta_w = \infty$) achieves a high PE fairness under uniform random writes, however the PE fairness of the classic d-choices GC algorithm drops quickly in the presence of hot and cold data (see Table **??**).

In Figure 2b we look at the sensitivity of the WA factor in terms of the parameter d^* with $d = 50$ (as larger d values perform better). The parameter d^* is used during the move operation, where we selected the block with the most number of valid pages among d^* randomly selected pages with exactly w_{min} erasures. As such larger d^* values imply that we are more likely to find a block with more valid pages as a move block. The results indicate that increasing d^* reduces the WA, however

d	$\Delta_w = 7$	$\Delta_w = 15$	$\Delta_w = 31$	$\Delta_w = 63$	$\Delta_w = \infty$
10	0.9979	0.9955	0.9907	0.9821	0.9680
20	0.9978	0.9954	0.9904	0.9818	0.9695
50	0.9978	0.9953	0.9903	0.9817	0.9707
100	0.9978	0.9953	0.9903	0.9817	0.9714
bound	0.9965	0.9925	0.9845	0.9685	0

Table 2. Impact of d and Δ_w on the PE fairness for $d^* = 5$, $b = 32$ and $S_f = 0.1$ under uniform random writes with $W_{max} = 2000$.

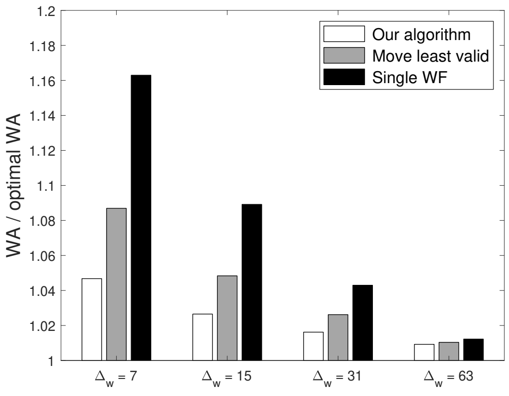


Fig. 3. Impact of changing some of the design choices in our algorithm on the write amplification under uniform random writes for $d = 50, d^* = 5$, $b = 32$ and $S_f = 0.1$.

the reduction is much more limited compared to the impact of d and setting d^* as low as 5 already gives results that are quite close to much larger d^* values, e.g., $d^* = 50$. This is especially true for larger Δ_w values.

We end this section by looking at some of the algorithm design choices. More specifically we look at the setting where the move block is selected as the block with the *least number of valid pages* instead of the most number of valid pages and at the setting where we use a single as opposed to the WFI/WFE write frontiers (see Section 4.2 for details). We note that selecting the move block based on the least number of valid pages increases the WA, which is not unexpected given our earlier observation that increasing d^* lowered the WA. The impact of this design choice however lowers as Δ_w increases since fewer move operations are performed with increasing Δ_w. When replacing the WFI/WFE write frontiers by the single write frontier we see a more pronounced increase in the WA. Thus contrary to the classic d-choices GC the write amplification is no longer the same for the WFI/WFE write frontiers and the single write frontier under uniform random writes. We do note that the difference is again very limited for larger Δ_w values. In the Section 6 we consider trace based workloads in which case using a double write frontier is key to achieve a low write amplification.

5 MEAN FIELD DETAILS

In this section we provide some of the fine details that yield the drift equations in (3), (5) and (6). The mean field model observes the system whenever a call to the GC algorithm is made. As stated before, the system state at such points in time is represented by the fractions $m_{i,w}$, for $i = 0, \ldots, b$

and $w = w_{min}, \ldots, w_{max}$ and the state $(j^*, w^*) \in \{1, \ldots, b\} \times \{w_{min}+1, \ldots, w_{max}\}$ of the WFI, where j^* is the number of valid pages on the WFI and w^* the number of erasures performed so far on the WFI. Note that the WFI plays the role of the resource in the mean field framework presented in [3]. The drift

$$f_{i,w}^{(nomove)}(\vec{m}) = \sum_{j^*=1}^{b} \sum_{w^*=w_{min}+1}^{w_{max}} \pi_{j^*,w^*}(\vec{m}) f_{i,w}^{(nomove)}(\vec{m}, j^*, w^*), \tag{10}$$

where $f_{i,w}^{(nomove)}(\vec{m}, j^*, w^*)$ is the drift conditioned on the state of the WFI and $\pi_{j^*,w^*}(\vec{m})$ are the steady state probabilities of the WFI given that the occupancy vector equals $\vec{m}$.

As explained below the conditional drift is given by

$$\begin{aligned} f_{i,w}^{(nomove)}(\vec{m}, j^*, w^*) &= \frac{(i+1)m_{i+1,w} - i m_{i,w}}{b(1-S_f)} \sum_{\ell=j^*}^{b} b p_{b-\ell}(\vec{m}) - p_{i,w}(\vec{m}) \\ &+ 1[i=b, w > w_{min}] \sum_{\ell=j^*}^{b} p_{b-\ell,w-1}(\vec{m}) + 1[i=b, w^*=w] \sum_{\ell=0}^{j^*-1} p_{b-\ell}(\vec{m}). \end{aligned} \tag{11}$$

The first two terms are identical to the first two terms in (3), except for the sum. Given that the WFI contains j^* valid pages, $\sum_{\ell=j^*}^{b} p_{b-\ell}(\vec{m})$ is the probability that the valid pages of the victim can be copied to the WFI and in such case we have a new WFE that can support b host writes. The third term can be understood by noting that $\sum_{\ell=j^*}^{b} p_{b-\ell,w-1}(\vec{m})$ is the probability that a victim block with $w-1$ erasures is selected that becomes the new WFE. This block will be erased and filled with host writes, creating a new block with w erasures and b valid pages. Finally, $\sum_{\ell=0}^{j^*-1} p_{b-\ell}(\vec{m})$ is the probability that the victim block becomes the new WFI and we get a new block with w^* erasures and b valid pages, being the WFI.

To obtain (3) from (10) we need to argue that

$$\sum_{j=1}^{b} p_{b-j}(\vec{m}) j = \sum_{j^*=1}^{b} \sum_{w^*=w_{min}+1}^{w_{max}} \pi_{j^*,w^*}(\vec{m}) \sum_{\ell=j^*}^{b} b p_{b-\ell}(\vec{m}) \tag{12}$$

and

$$\sum_{\ell=0}^{b} p_{\ell,w-1}(\vec{m}) = \sum_{j^*=1}^{b} \sum_{w^*=w_{min}+1}^{w_{max}} \pi_{j^*,w^*}(\vec{m}) \sum_{\ell=j^*}^{b} p_{b-\ell,w-1}(\vec{m}) + \sum_{j^*=1}^{b} \pi_{j^*,w}(\vec{m}) \sum_{\ell=0}^{j^*-1} p_{b-\ell}(\vec{m}). \tag{13}$$

We first look at the probabilities $\pi_{j^*,w^*}(\vec{m})$. These are the steady state probabilities of the WFI and the state of the WFI evolves according to the Markov chain with with the following size $b\Delta_w$ transition probability matrix (see Figure 4 for an illustration):

$$K_{(j^*,w^*),(j^+,w^+)}(\vec{m}) = \begin{cases} p_{j^+-j^*}(\vec{m}) & j^+ > j^*, w^+ = w^*, \\ p_0(\vec{m}) + p_{b,w^+-1}(\vec{m}) & j^+ = j^*, w^+ = w^*, \\ p_{b,w^+-1}(\vec{m}) & j^+ = j^*, w^+ \neq w^*, \\ p_{b-j^*+j^+,w^+-1}(\vec{m}) & j^+ < j^*, \\ 0 & \text{elsewhere.} \end{cases} \tag{14}$$

Here $p_{j^+-j^*}(\vec{m})$ is the probability that the number of valid pages increases from j^* to j^+ and the WFI remains the same, hence $w^+ = w^*$. With probability $p_{b-j^*+j^+,w^+-1}(\vec{m})$ the WFI becomes full and the victim is erased and becomes the new WFI with j^+ valid pages.

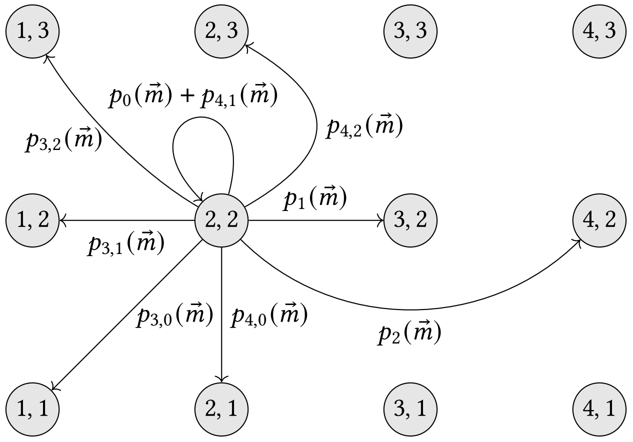


Fig. 4. Illustration of the transition probabilities of the WFI from state $(j^*, w^*) = (2, 2)$ for $b = 4$, $\Delta = 3$ and $w_{min} = 0$.

If we only focus on the evolution of the number of valid pages on the WFI, we obtain a Markov chain with the following transition probability matrix:

$$\tilde{K}_{j^*,j^+}(\vec{m}) = \begin{cases} p_{j^+-j^*}(\vec{m}) & j^+ > j^* \\ p_{b-j^*+j^+}(\vec{m}) & j^+ < j^*, \\ p_0(\vec{m}) + p_b(\vec{m}) & j^+ = j^* \end{cases} \tag{15}$$

which is a circulant matrix. Therefore its steady state probabilities equal $1/b$ and we find that

$$\sum_{w^*=w_{min}+1}^{w_{max}} \pi_{j^*,w^*}(\vec{m}) = 1/b, \tag{16}$$

for all j^*, meaning the number of valid pages on the WFI follows a uniform distribution. Using (16) we immediately find that (12) holds and (13) simplifies to showing

$$\sum_{\ell=1}^{b}(1 - \frac{\ell}{b})p_{b-\ell,w-1}(\vec{m}) = \sum_{j^*=1}^{b} \pi_{j^*,w}(\vec{m}) \sum_{\ell=0}^{j^*-1} p_{b-\ell}(\vec{m}). \tag{17}$$

By summing the balance equations of $K(\vec{m})$ using (14), we have

$$\begin{aligned}\sum_{j^+=1}^{b} \pi_{j^+,w^+}(\vec{m}) &= \sum_{j^+=1}^{b}\sum_{j^*=1}^{j^+} \pi_{j^*,w^+}(\vec{m})p_{j^+-j^*}(\vec{m}) + \sum_{j^+=1}^{b}\sum_{j^*=j^+}^{b}\sum_{w^*=w_{min}+1}^{w_{max}} \pi_{j^*,w^*}(\vec{m})p_{b-j^*+j^+,w^+-1}(\vec{m}) \\ &= \sum_{j^*=1}^{b} \pi_{j^*,w^+}(\vec{m}) \sum_{\ell=0}^{b-j^*} p_\ell(\vec{m}) + \sum_{j^+=1}^{b}\sum_{\ell=0}^{b-j^+} \frac{1}{b}p_{b-\ell,w^+-1}(\vec{m}) \\ &= \sum_{j^*=1}^{b} \pi_{j^*,w^+}(\vec{m}) \sum_{\ell=0}^{b-j^*} p_\ell(\vec{m}) + \sum_{\ell=0}^{b-1}(1 - \frac{\ell}{b})p_{b-\ell,w^+-1}(\vec{m}),\end{aligned}$$

where we used (16). (17) now follows as $\sum_{\ell=0}^{b} p_\ell(\vec{m}) = 1$.

We end this section by looking at the drifts $f_{i,w}^{(move,1)}(\vec{m})$ and $f_{i,w}^{(move,2)}(\vec{m})$. These drifts are only affected by the probabilities $\pi_{j^*,w^*}(\vec{m})$ through the probability $p_{move}(\vec{m})$ that a move operation

trace	page reads	page writes	page read footprint (GB)	page write footprint (GB)	total page footprint (GB)
w33	1082234	37993401	0.8038	11.1661	11.7080
w76	1354159	11665083	1.7712	7.9410	9.1567
w95	7999488	7628304	5.8055	3.1197	8.3469
prxy0	806528	21330164	0.3015	0.7067	0.8782
rsrch0	364955	2888684	0.0793	0.2894	0.3574
online	1488693	4211806	0.5244	0.2638	0.7508
webmail	1413830	6381985	1.2051	0.8353	1.8641

Table 3. Workload characteristics *after* preprocessing.

takes place as

$$p_{move}(\vec{m}) = \sum_{j^*=1}^{b} \sum_{w^*=w_{min}+1}^{w_{max}} \pi_{j^*,w^*}(\vec{m}) \sum_{\ell=j^*}^{b} p_{b-\ell,w_{max}-1}(\vec{m}),$$

which simplifies to (4) due to (16).

6 TRACE-BASED WORKLOADS

In this section we study the performance of the algorithm presented in Section 3 when subject to trace-based workloads (instead of uniform random writes). The main insight can be summarized as follows:

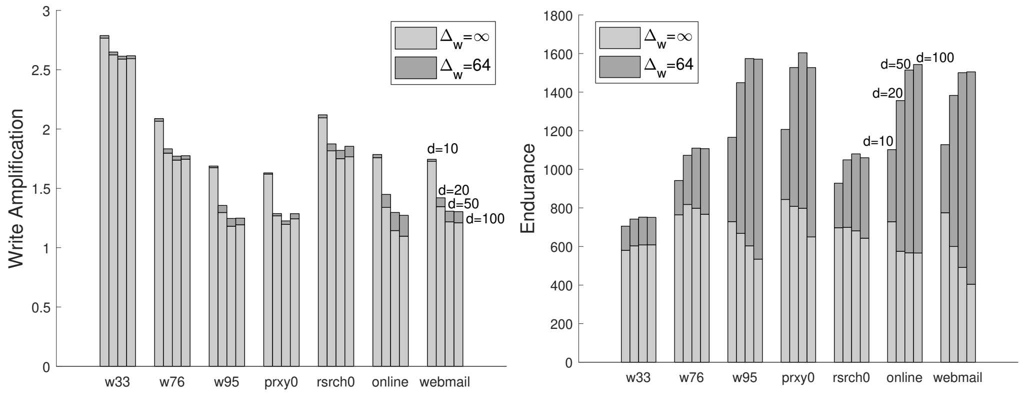
- The algorithm presented in Section 3 has a WA factor close to the WA factor of the classic d-choices algorithm, while its PE fairness is much higher and guaranteed to be close to one. This implies that the SSD endurance increases sharply compared to the classic d-choices GC algorithm (more than doubling in a number of cases). This shows that we can indeed guarantee near-perfect wear leveling without paying a high cost in terms of the WA factor, answering our initial question affirmatively for trace-based workloads.

In addition we further demonstrate that

- The impact of d^*, the number of blocks selected during the move operation, is very minor and that selecting a block with the most number of valid pages in a move operation is superior to selecting a block with the least number of valid pages.
- Without the move operation we can still achieve a near-perfect wear, but at the expense of a major increase in the WA factor. In other words the move operation is key to keep the growth of the WA factor limited.
- Replacing the WFI/WFE by a single WF in case of trace-based workloads results in a very high WA factor, which deteriorates the system performance. This confirms the know fact that separating hot and cold data (either implicitly or explicitly) is important for workloads with hot and cold data.

Note that these conclusions are very much in agreement with the conclusions drawn from the mean field model experiments in Section 4.4 (which was restricted to uniform random writes).

In our trace-based simulation experiments we make use of two (prxy0 and rsrch0) MSR block traces [1, 17], two (online and webmail) SyLab block traces [2, 25] and three (w33, w76 and w95) more recent CloudPhysics block traces [27]. We first preprocessed the above traces by aligning the offset of each request to a multiple of 4 KB. Requests with sizes above 4 KB were subsequently split into several (sequential) requests such that all requests have a size of at most 4 KB. The SSD



(a) WA factor

(b) Endurance

Fig. 5. WA factor and Endurance for $d = 10, 20, 50, 100$, $d^* = 5$, $b = 64$, $S_f = 0.1$ and $W_{max} = 2000$.

used in the trace-driven simulation experiments is composed of $U = \lfloor x/b \rfloor$ logical blocks, where x equals the number of different logical pages accessed by the trace. The number of physical blocks $N = U/(1 - S_f)$. The pages are initially placed on the first U physical blocks, while the remaining $N - U$ blocks contain b pages in the erase state. To make the simulation runs sufficiently large we adopted the *replay* method used in prior SSD work [14, 16]. Some characteristics of the workloads considered are listed in Table 3.

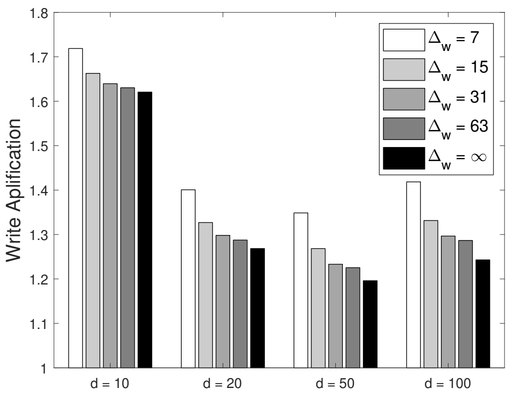
Main insight: To demonstrate that the algorithm proposed in Section 3 can achieve near-perfect wear leveling at a low cost in terms of the WA factor, we compare the WA factor, PE fairness and Endurance of this algorithm for $\Delta_w = 63$ with the d-choices algorithm (which corresponds to setting $\Delta_w = \infty$) for each of the workloads considered. We set $b = 64, S_f = 0.1, d^* = 5$ and consider four different choices of d. Similar results were also obtained for other parameter settings. The maximum number of erase cycles W_{max} is set to 2000, such that for $\Delta_w = 63$ the PE fairness is guaranteed to exceed $1 - 63/2000 = 0.9685$. Figure 5a presents the results for the WA factor. The light gray bars show the WA factor of the d-choices GC algorithm, while the dark gray bars indicate the increase in the WA factor of the algorithm when $\Delta_w = 63$. It shows that for most of these workloads the increase in the WA factor is indeed limited. Note that in some cases setting $d = 100$ yields a higher WA factor than setting $d = 50$ which is in agreement with the fact that the greedy GC algorithm does not minimize the WA factor for non-uniform writes.

The question is now whether this guaranteed near-perfect wear yields a strong improvement for the life span of the drive. Figure 5b confirms that this is indeed the case. It shows a large increase in terms of the Endurance (often doubling for larger d values), which represents the number of full drive writes that can be supported before any block is erased W_{max} times. This huge improvement can be understood by looking at Table 4 which indicates that the classic d-choices GC algorithm causes a very unequal wear that becomes more pronounced as d increases. Note that for the classic d-choices GC algorithm there is a trade-off between the WA factor and the PE fairness: the d value that minimizes the WA factor often yields a poor PE fairness.

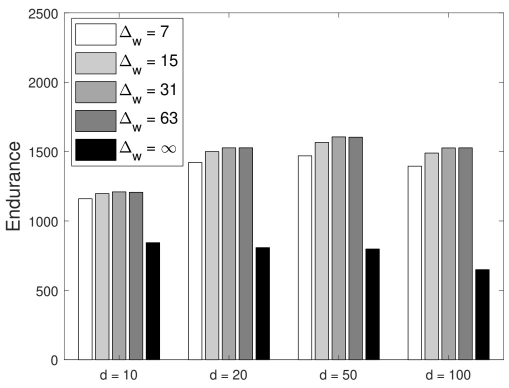
We now look at the influence of some of the other parameters. We present results for the prxy0 trace (similar observations are made for other traces).

trace	Δ_w	$d = 10$	$d = 20$	$d = 50$	$d = 100$
w33	63	0.9836	0.9836	0.9835	0.9836
	∞	0.8022	0.7917	0.7873	0.7889
w76	63	0.9836	0.9833	0.9832	0.9833
	∞	0.7890	0.7347	0.6937	0.6696
w95	63	0.9841	0.9828	0.9813	0.9817
	∞	0.6104	0.4329	0.3561	0.3188
prxy0	63	0.9839	0.9836	0.9834	0.9835
	∞	0.6836	0.5125	0.4773	0.4036
rsrch0	63	0.9836	0.9836	0.9835	0.9837
	∞	0.7301	0.6348	0.5961	0.5678
online	63	0.9839	0.9829	0.9826	0.9827
	∞	0.6404	0.3851	0.3244	0.3105
webmail	63	0.9839	0.9826	0.9820	0.9822
	∞	0.6692	0.4037	0.2993	0.2446

Table 4. The PE fairness for $d^* = 5$, $b = 64$ and $S_f = 0.1$ with $W_{max} = 2000$ for various d and Δ_w.



(a) WA factor

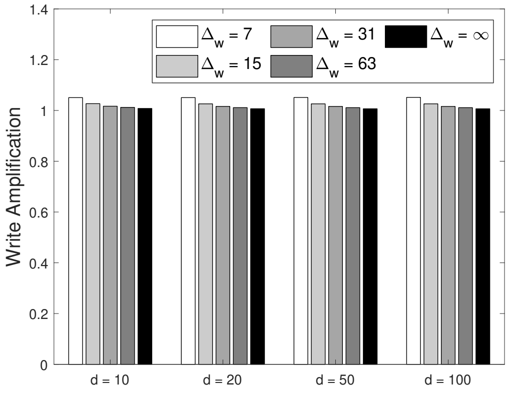


(b) Endurance

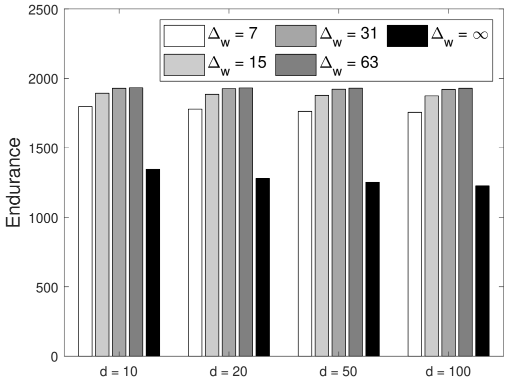
Fig. 6. Impact of d and Δ_w for $d^* = 5$, $b = 64$ and $\rho = 0.9$ for the prxy0 trace with $W_{max} = 2000$.

Influence of Δ_w*:* Figure 6a depicts the impact of Δ_w on the write amplification for the prxy0 trace with $b = 64$ and $\rho = 0.9$. The main observation is that smaller Δ_w values yield a higher WA factor (as expected) and the increase becomes more significant for small Δ_w. We therefore see a trade-off in the choice of Δ_w: smaller Δ_w improve the PE fairness, but lower the WA. This implies that for the Endurance there exists an optimal Δ_w value as shown in Figure 6b. It also shows that a near-perfect wear suffices to get a high endurance instead of striving for perfect wear leveling.

Low utilization: We now repeat the previous experiment using the same parameters, except that we lower ρ from 0.9 to 0.3. This reflects a setting where the SSD has a low utilization. In such a setting the WA factor should be close to one, but the PE fairness may still be well below one when some of the data on the SSD is cold or read only data, where larger fractions of read only data result in lower PE fairness numbers. In Figure 7a we see that the WA factor is indeed close to 1, especially for $\Delta_w = 63$ and $\Delta_w = \infty$. Looking at Figure 7b we see that there is a huge difference



(a) WA factor



(b) Endurance

Fig. 7. Impact of d and Δ_w for $d^* = 5$, $b = 64$ and $\rho = 0.3$ for the prxy0 trace with $W_{max} = 2000$.

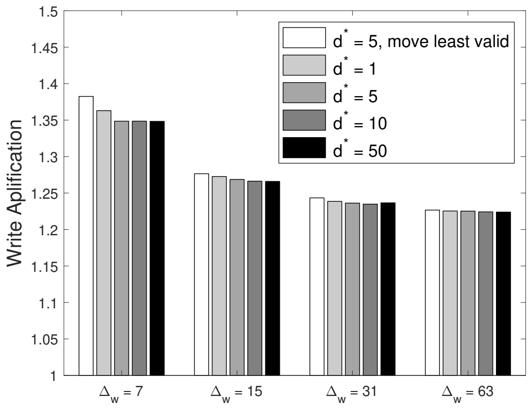
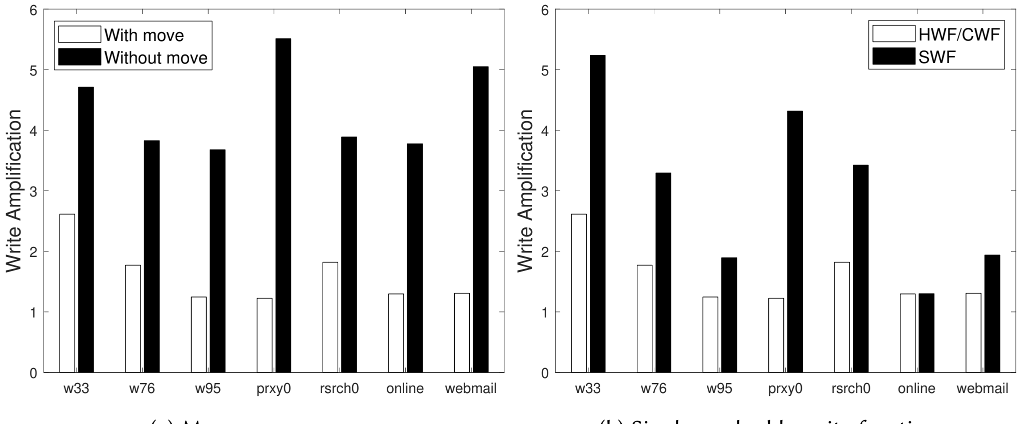


Fig. 8. Impact of d^* and Δ_w on the WA factor for $d = 50$, $b = 64$ and $S_f = 0.1$ for the prxy0 trace.

in the endurance between $\Delta_w = 63$ and $\Delta_w = \infty$. The former has an endurance close to 2000 full drive writes, as the PE fairness and the WA factor are both close to one, while with $\Delta_w = \infty$ the PE fairness varies between 0.61 and 0.68. This experiment therefore shows that the proposed algorithm is also very effective in low utilization scenarios.

Influence of d^*: The impact of d^*, the number of choices used to select the move block among the blocks with w_{min} erasures, is illustrated in Figure 8 for the prxy0 trace. In addition this figure also indicates what happens if the block with the least number of valid pages among the d^* selected blocks is picked as the move block. As in the uniform random writes case the impact of d^* is limited and larger d^* values result in a very small decrease in the WA (close to zero for $\Delta_w = 63$). In fact the impact of d^* is very minor as most of the blocks with w_{min} erasures contain mostly cold data and therefore a small d^* yields a high probability of selecting a block with cold data. We further note that selecting the block with the least number of valid pages as the move block increases the WA factor, but the increase is quite limited.



(a) Move vs. no move. (b) Single vs. double write frontier.

Fig. 9. WA factor for $d = 50$, $d^* = 5$, $b = 64$, $\Delta_w = 63$ and $S_f = 0.1$.

Importance of move operation: In Figure 9a we present results for all the workloads in Table 3 for $\Delta_w = 63$ and study the impact of disabling the move operation in Case (1B) of the algorithm. It is immediately apparent that disabling the move operation results in a very poor WA. This can be understood by noting that without the move operation, the new WFE that is created by the GC algorithm may be a block that experienced w_{max} writes. The WFE is typically filled with lots of hot data that is quickly invalidated, creating a block with a low number of valid pages. However if the erase counter of the WFE equals w_{max}, it cannot be selected by the GC algorithm. Thus without the move operation, a lot of the hot invalidated pages get stuck on the blocks with exactly w_{max} erasures, which has a detrimental effect on the WA. Note that the move operation itself also causes some additional internal writes, but their contribution to the total number of internal writes is limited. More specifically for the experiment in Figure 9a the percentage of internal writes caused by a move operation on the total number of internal writes equals 1.00%, 2.31%, 6.26%, 2.12%, 5.15% and 5.11% for the seven traces considered, where the traces with a larger fraction of read only data tend to require more move operations.

Importance of data separation: In Figure 9b we look at the impact of having a single write frontier instead of the WFI/WFE write frontiers for our algorithm for each of the traces in Table 3 for $\Delta_w = 63$. As anticipated the write amplification surges up significantly for all the traces (except for the online trace), showing the importance of separating hot and cold data (implicitly or explicitly). This increase is in line with earlier observations, e.g. [19, Figure 9].

7 CONCLUSIONS

In this paper we addressed the question whether near-perfect wear leveling is possible at low costs in terms of the write amplification factor. We answered this question in the affirmative by presenting a simple randomized algorithm that combines wear leveling with garbage collection. This algorithm guarantees that the wear is nearly perfectly balanced at all times, that is, the number of erase cycles of any two blocks is guaranteed to differ by at most Δ_w (a parameter of the algorithm), while causing a low increase in the write amplification. This was demonstrated mathematically using a mean field model in case of uniform random writes and using trace-driven simulation experiments.

ACKNOWLEDGEMENT

The author likes to thank Peter Desnoyers for some useful comments and suggestions on an earlier version of the paper and for helping out with parsing the CloudPhysics traces.

REFERENCES

[1] ftp://ftp.research.microsoft.com/pub/austind/msrc-io-traces/. MSRC-io-traces.

[2] http://sylab-srv.cs.fiu.edu/dokuwiki/doku.php?id=projects:srcmap:start. SyLab Energy Proportional Storage Systems Traces.

[3] 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.

[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] Mei-Ling Chiang, Paul C. H. Lee, and Ruei-Chuan Chang. Using data clustering to improve cleaning performance for flash memory. *Softw. Pract. Exper.*, 29(3):267–290, March 1999.

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

[8] S.N. Ethier and T.C. Kurtz. *Markov processes: characterization and convergence.* Wiley, 1986.

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

[10] N. Gast and B. Gaujal. Markov chains with discontinuous drifts have differential inclusion limits. *Perform. Eval.*, 69(12):623–642, 2012.

[11] 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.

[12] 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.

[13] Ziyang Jiao, Janki Bhimani, and Bryan S. Kim. Wear leveling in ssds considered harmful. In *Proceedings of the 14th ACM Workshop on Hot Topics in Storage and File Systems*, HotStorage ’22, page 72–78, New York, NY, USA, 2022. Association for Computing Machinery.

[14] 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.

[15] Yixin Luo, Yu Cai, Saugata Ghose, Jongmoo Choi, and Onur Mutlu. Warm: Improving nand flash memory lifetime with write-hotness aware retention management. In *MSST*, pages 1–14. IEEE Computer Society, 2015.

[16] M. Murugan and D. Du. Rejuvenator: A static wear leveling algorithm for NAND flash memory with minimized overhead. In *Proc. of IEEE MSST*, 2011.

[17] 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.

[18] 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.

[19] M. Shafaei and P. Desnoyers. Near-optimal offline cleaning for flash-based SSDs, 2017. http://storageconference.us/2017/Papers/CleaningFlashBasedSSDs.pdf.

[20] M. Shafaei, P. Desnoyers, and J. Fitzpatrick. Write amplification reduction in flash-based ssds through extent-based temperature identification. In *8th {USENIX} Workshop on Hot Topics in Storage and File Systems (HotStorage 16)*, 2016.

[21] A. Tavakkol, M. Arjomand, and H. Sarbazi-Azad. Unleashing the potentials of dynamism for page allocation strategies in SSDs. *SIGMETRICS Perform. Eval. Rev.*, 42(1):551–552, June 2014.

[22] 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.

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

[24] B. Van Houdt. On the necessity of hot and cold data identification to reduce the write amplification in flash-based SSDs. *Perform. Eval.*, 82:1 – 14, 2014.

[25] 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.

[26] R. Verschoren and B. Van Houdt. On the endurance of the d-choices garbage collection algorithm for flash-based SSDs. *ACM Trans. Model. Perform. Eval. Comput. Syst.*, 4(3):13:1–13:23, July 2019.

[27] C. A. Waldspurger, N. Park, A. Garthwaite, and I. Ahmad. Efficient MRC construction with SHARDS. In *13th USENIX Conference on File and Storage Technologies (FAST 15)*, pages 95–110, Santa Clara, CA, February 2015. USENIX Association.

[28] Y. Yang, V. Misra, and D. Rubenstein. On the optimality of greedy garbage collection for SSDs. *ACM SIGMETRICS Perform. Eval. Rev.*, 43(2):63–65, September 2015.