On the Necessity of Hot and Cold Data Identification to Reduce the Write Amplification in Flash-based SSDs

B. Van Houdt

Department of Mathematics and Computer Science
University of Antwerp - iMinds

Abstract

The write performance and life span of a solid state drive is greatly influenced by the garbage collection algorithm. This algorithm selects the data blocks to be erased which can be subsequently used for storing new data. Any valid data left on a selected block needs to be written elsewhere before the block can be erased and contributes to the so-called write amplification.

As all of the data on a solid state drive is not accessed equally often, data identification techniques have been proposed that identify the more frequently accessed, called hot, from the less frequently accessed, termed cold, data. These data identification techniques have been shown to be quite effective in reducing the write amplification essentially by using different blocks to store the hot and cold data, but they also contribute to the complexity of the device.

Write approaches that use different blocks for writes triggered by the operating system and writes triggered by the garbage collection algorithm have also been proposed. These approaches do not require a data identification technique and thus simplify the design of the device, while also reducing the write amplification.

In this paper we compare the performance of such a write approach with write approaches that do rely on data identification using both mean field models and simulation experiments. The main finding is that the added gain of identifying hot and cold data is quite limited, especially as the hot data gets hotter. Moreover, the write approaches relying on hot and cold data identification may even become inferior if either the fraction of data labeled hot is not ideally chosen or if the probability of having false positives or negatives when identifying data is substantial (e.g. 5%).

1. Introduction

Data on a solid state drive (SSD) is partitioned in blocks that each contain a fixed number of fixed size pages (e.g., 64 pages of 2 Kbyte each) and the unit

Email address: benny.vanhoudt@uantwerpen.be (B. Van Houdt)

of data exchange is a single page. 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 [6].

The garbage collection (GC) algorithm is responsible for selecting the blocks that are erased and subsequently used to store new data. More specifically, when activated the GC algorithm selects a block (in some algorithm specific manner), copies the valid pages from the selected block to RAM, erases the entire block and writes the valid pages back, leaving the remaining pages in the erase state. The additional write operations caused by the GC algorithm contribute to the so-called write amplification, defined as the ratio of the total number of physical writes performed by the SSD divided by the number of logical writes requested by the operating system.

The write amplification can be quite high if the logical storage capacity is (nearly) fully used, as any block selected by the GC algorithm is packed with valid data. As such 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.

The write amplification can be further reduced by making sure that some blocks contain a small number of valid pages, while others contain many valid pages, as this creates opportunities for the GC algorithm to select blocks with a small number of valid pages. One way to achieve this exists in implementing a hot and cold data identification technique (e.g.,[5, 11, 7, 23]). Such a technique attempts to identify the more frequently accessed logical pages, called the hot pages, and uses different blocks to store hot and cold pages (where any pages that is not hot is termed cold). This helps to reduce the write amplification as blocks packed with hot valid pages are invalidated more quickly and therefore tend to create blocks with a small number of valid pages.

An approach discussed in [28] exists in using different blocks for writes requested by the operating system and for writes triggered by the GC algorithm. Thus, instead of separating hot and cold pages, this approach separates internally and externally requested writes. Numerical results in [28] indicate that this approach also reduces the write amplification, basically because the majority of the externally requested writes go, by definition, to the hotter data. In other words, this approach also achieves a form of hot and cold data separation, but without the need to identify the hot pages.

The main objective of this paper is to compare the write amplification of the approach of [28] with approaches that rely on a data identification technique to see how much additional gain such an identification technique offers, that is, we wish to examine whether the added complexity of implementing a data identification technique is worth the trouble when focusing on the write amplification.

The main contributions and findings of the paper can be listed as follows:

1. We compare the write amplification of write approaches relying on a data identification technique with [28] using both synthetic and trace-based workloads. To derive results for the synthetic workload model we rely on the mean field model of [28] as well as on a new mean field model developed and validated in Section 3. For the trace-based workloads we make use of a customized simulation program.
2. For the synthetic workloads we show that the added gain of implementing a hot and cold data identification technique diminishes as the hot data gets hotter. Furthermore if the number of false positives (i.e., a cold page that is identified as hot) and negatives (i.e., a hot pages that is marked as cold) is substantial (e.g., +5%), the approaches based on data identification may even become inferior. The synthetic workload consists of the so-called Rosenblum model for hot and cold data, which partitions the pages in two classes such that all the pages within a class are accessed equally often. As such one class of pages corresponds to the hot pages and the other to the cold pages.
3. The trace-based workloads correspond to real life I/O traces. In this case drawing a line between the hot and cold data is less obvious and the optimal fraction of the data that should be marked as hot to minimize the write amplification is workload dependent. The main finding indicates that the write approaches using data identification do not always provide much additional gain compared to the write approach of [28] and may even become inferior if the fraction of the data that is labeled hot is not properly chosen.

It is important to note that the conclusions drawn in this paper apply to any hot and cold data identification technique, in the sense that we do not focus on a specific technique, but instead assume some data identification technique is in place that either works perfectly or has a specific false positive and/or negative probability.

Before proceeding there are a few more issues that are worth addressing. To support out-of-place writes the SSD must maintain a mapping between the logical and physical page numbers, which is stored in the flash translation layer (FTL) [8, 10]. In this paper we focus on a page-mapped FTL, meaning any logical page can be mapped to any physical page at the expense of requiring as many entries in the map as there are logical pages on the SSD. Many flash-based devices rely on a hybrid-mapped FTL (e.g., [14, 16, 13, 15, 18]) as this reduces the size of the FTL map. We note that some of these solutions were designed specifically for mobile embedded systems (e.g., MP3 and PDAs) and are not very suitable for the workloads with a substantial fraction of random writes encountered in general-purpose computing.

Another relevant issue of an SDD is that a block can only be erased a limited number of times [20]. A reduced write amplification helps in this regard as overall the GC algorithm needs to perform fewer erase operations. However, the approaches studied in this paper erase blocks used to store hot pages far more

Symbol	Meaning
b	Number of pages in a block
d	Number of choices in d-choices GC algorithm
f	Fraction of the pages that are hot in the Rosenblum model
f_p/f_n	Probability of a false positive/negative
p	Fraction of S_f assigned to hot partition in the STAT approach
r	Probability that a hot page is requested in the Rosenblum model
N	Number of physical blocks on the SSD
S_f	Spare factor defined as $1 - U/N$
U	Number of logical blocks on the SSD
ρ	Ratio of the number of logical to physical blocks (U/N)

Table 1: Table of notations.

often. As such the approaches presented in this paper need to be complemented with a wear-leveling mechanism such that the number of erase operations on each block is approximately the same.

Finally, many modern SSDs and operating systems support the so-called TRIM command which allows a file system to inform the SSD which pages can be marked as invalid whenever a file is deleted. Without this command information regarding file deletions is not passed down to the SSD and the fraction of pages in the valid state will therefore become and remain equal to one minus the spare factor (i.e., U/N) when the drive has been operational for a sufficiently long time. In this paper we assume that the TRIM command is not supported.

The paper is structured as follows. In Section 2 we introduce four different write approaches, two of which rely on a hot/cold data identification technique, and discuss the GC algorithms used. In Section 3 a mean field model is introduced and validated to determine the write amplification of one of the write approaches relying on data identification under a synthetic workload model. Sections 4 and 5 compare the different approaches using synthetic and trace-based workloads, respectively. Conclusions are drawn in Section 6

2. System operation

2.1. Write approaches

We start by introducing two write approaches that do not rely on a hot and cold data identification technique, being the Single Write Frontier approach, studied in [24, 19, 12, 4, 9, 27], and the Double Write Frontier discussed in [28]. Denote b as the number of pages in a block.

Single Write Frontier (SWF). A single block is labeled the write frontier (WF) at all times and new data is written sequentially to the WF. When the WF becomes full, the GC algorithm selects a block, temporarily removes the valid data from this block, erases all of its pages, places the removed valid data back

and labels the block as the new WF. Thus, if the selected block contained j valid pages, $b-j$ pages of the WF are in the erase state when the GC algorithm is finished.

Double Write Frontier (DWF). One block is labeled the external write frontier (WFE) and another the internal write frontier (WFI) at all times. New data is sequentially written to the WFE. When the WFE becomes full, the GC algorithm selects a block. Assume the selected block contains j valid pages, while the WFI has k pages in the erase state when the GC algorithm is activated.

- If $k \geq j$, the j valid pages of the selected block are copied to the WFI, the selected block is erased and becomes the new WFE (with b pages in the erase state).
- Otherwise, k of the j valid pages are copied to the WFI, the remaining $j-k$ pages are copied back to the selected block after all of its pages have been erased and the selected block becomes the new WFI.

In the latter case the GC algorithm is immediately reactivated in search of a new WFE.

It should be noted that the DWF approach bears some resemblance to the second chance policy discussed in [18] as this policy also copies valid pages part of the block selected for erasure to a separate block, but as opposed to the DWF approach this policy uses different blocks depending on whether a valid page already received a so-called second change.

Our main objective is to compare the write amplification of the DWF approach with the following two approaches that both rely on a hot and cold data identification technique. The first approach is closely related to [5] in the sense that it also uses a separate write frontier for the cold and hot pages:

Hot/Cold Write Frontier (HCWF). One block is labeled the hot write frontier (HWF) and another the cold write frontier (CWF) at all times. New hot (cold) data is sequentially written to the HWF (CWF). All the remaining blocks are marked as either hot or cold at all times, depending on whether the block was last used as a HWF or CWF. The initial marking of the blocks (when the drive is empty) is irrelevant. If the HWF becomes full the GC algorithm selects a block. Assume the selected block contains j valid pages, while the CWF has k pages in the erase state when the GC is activated.

- If the selected block was marked hot, the j valid pages are copied back to the selected block after its pages have been erased and the selected block becomes the HWF.
- If the selected block was marked cold and $k \geq j$, the j valid pages are copied to the CWF and the selected block becomes the new HWF (which contains b pages in the erase state) and is labeled hot. Otherwise if $k < j$, k of the j valid pages are copied to the CWF, the remaining $j-k$ pages

are written back to the selected block after its pages have been erased and the selected block becomes the new CWF.

In the latter case (i.e., the selected block is marked cold and $k < j$), the GC algorithm is immediately reactivated in search of a new HWF. Finally, when the CWF becomes full, instead of the HWF, the system operates as above if we exchange the terms hot and cold.

A key feature of the HCWF solution is that it dynamically distributes the spare fraction S_f among the hot and cold data partition (formed by the hot/cold block markings) and blocks can move from one partition to the other. Recall, without the TRIM command the fraction of pages in the valid state equals $1-S_f$ if the drive has been operational sufficiently long. The next approach, discussed in [9], also separates the hot and cold pages in two partitions, but the spare space is distributed in a static offline manner among the two partitions, while the SWF approach is used within each partition.

Static Drive Partitioning (STAT). The blocks are partitioned in two sets in a static manner: one for the hot pages and one for the cold pages. Let f_{hot} be the fraction of the pages marked as hot by the data identification algorithm and r_{hot} be the probability that a random request updates a hot page. The size of the hot partition is defined as

$$f_{hot}(1 - S_f) + pS_f,$$

while the cold partition has size

$$(1 - f_{hot})(1 - S_f) + (1 - p)S_f.$$

In other words, we assign a fraction p of the spare factor S_f to the hot partition and a fraction $1-p$ of S_f to the cold partition. Within each partition the SWF approach is used. The basic idea is to let p exceed f_{hot} such that the write amplification of the hot partition (W_{hot}) decreases at the expense of the write amplification on the cold partition (W_{cold}). As the overall write amplification is given by

$$r_{hot}W_{hot} + (1 - r_{hot})W_{cold},$$

this may result in an overall reduction of the write amplification. The parameter p is determined (numerically) such that the overall write amplification is minimized, as such the STAT scheme corresponds to the optimal static drive partitioning.

It is worth noting that the STAT approach basically creates two isolated SSDs that each operate using the SWF approach: one for hot and one for cold pages. As such this approach cannot be used directly if the id of the hot pages changes over time, as is often the case in practice. In order to cope with dynamic hot data, the fraction p of the spare space has to be determined in an online manner and the blocks cleaned by the GC algorithm should be assigned

dynamically. An online mechanism for dynamically calculating the fraction p was presented in [9]. The other approaches do not require any changes if the id and amount of hot data changes over time. For the numerical experiments reported in Sections 4 and 5, we limit ourselves to settings where the hot pages remain hot at all times.

2.2. Garbage Collection Algorithms

In the previous section we specified four different write approaches that each rely on a GC algorithm without discussing which GC algorithm is used to select a block. The following GC algorithms were introduced and analyzed in a number of earlier studies (using a page mapped FTL and without the TRIM command):

- The FIFO GC algorithm [24, 19, 30, 9] selects the blocks in a cyclic order.
- The Greedy GC algorithm [4, 9] selects the block containing the fewest number of valid pages among all the blocks.
- The d-choices GC algorithm [27, 17, 28, 26] selects the block with the fewest number of valid pages out of a set of d randomly chosen blocks.
- The Windowed GC algorithm [12] maintains a window containing the w least recently selected blocks and selects the block with the fewest number of valid pages in the current window.

Most of the above studies consider uniform random writes and focus on the SWF approach (while N tends to infinity). Under such a setting the Greedy algorithm is believed to be optimal, while the FIFO algorithm often has the highest write amplification. Both the d-choices and the Windowed algorithm provide a trade-off between the simplicity of FIFO and the performance of the Greedy algorithm. The d-choices GC algorithm however provides a much better trade-off than the Windowed algorithm as small values of d suffice, e.g., $d = 10$, to achieve a write amplification close to that of the Greedy algorithm (see [27] for details).

The performance of the FIFO and Greedy algorithm in the presence of hot and cold data was analyzed in [9] for a system using the SWF approach. These results indicated that the write amplification worsens significantly as the hot data gets hotter, especially for the FIFO GC algorithm. Similar results for the SWF approach were provided in [28] for the d-choices GC algorithm, while with the DWF approach the write amplification was shown to decrease as the hot data gets hotter. Further, under the DWF approach the Greedy algorithm is not optimal, instead there exists an optimal value for d. In this paper we make use of either the d-choices or Greedy GC algorithm depending on the write approach used (see Section 3–5 for more details).

3. Mean field model for HCWF

3.1. Model description

In this section we introduce a mean field model for the d-choices GC algorithm when combined with the HCWF approach and a perfect data identification technique. As in [9, 28], we consider non-uniform random writes modeled by the hot/cold data model of Rosenblum [25]. In this simple model a fraction f of the logical address space is termed hot and the remaining pages are termed cold, while the ids of consecutive write requests are independent and the probability that a hot page is requested equals 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 [9].

As the ids of the hot pages do not change over time in this model and we assume that the data identification technique identifies all the hot and cold pages as such (i.e., there are no false positives or negatives), the HCWF approach never mixes hot and cold pages within a block. Thus, at all times any block contains either hot or cold valid data only and can be labeled as hot or cold. In fact, even a block with no valid data at all can be classified as hot/cold depending on whether it was last used as a HWF or CWF. As opposed to [27, 28], we will observe the system not only just prior to any call to the GC algorithm, but also just prior to any write request. Let $X_n^N(t) \in S = \{0, \ldots, b\}$ denote the number of valid pages in block n and $Y_n^N(t) \in \{h, c\}$ reflect whether this block is labeled hot (h) or cold (c) at the t-th point of observation (i.e., the t-th time the GC algorithm is activated or a write request is received).

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

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

for $z \in \{h, c\}$ and $i \in S$. In other words, $M_{h,i}^N(t)$ ($M_{c,i}^N(t)$) is the fraction of the total number of blocks N that are labeled hot (cold) and contain i valid pages. To ease the notation we will refer to such blocks as type (z, i) blocks. Further let $J^N(t) \in \Omega = \{(k, l) | 0 \leq k, l \leq b\} \backslash \{(b, b)\}$ represent the number of pages written so far in the HWF and CWF. Note the GC algorithm is executed at the t-th point of observation if $J(t)$ is of the form (b, l) or (k, b).

It is easy to see that $\{(M^N(t), J^N(t)), t \in \mathbb{N}\}$ is a Markov chain. However, it clearly suffers from the curse of dimensionality for practical values of N (e.g., $N = 10,000$) and we therefore introduce a mean field model by defining $\bar{M}^N(\tau)$ as the re-scaled process such that $\bar{M}^N(t/N) = M^N(t)$, for $t \in \mathbb{N}$ and $\bar{M}^N(t)$ affine in $[t/N, (t+1)/N]$. Similarly, define $\bar{J}^N(\tau)$ as the re-scaled process of $J^N(t)$. We will argue that the limit process of $(\bar{M}^N(t), \bar{J}^N(t))$ 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 (1). 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 (1) with $\vec{\mu}(0) = M^N(0)$.

The mean field model is defined by means of the deterministic process $\vec{\mu}(t) = \{\mu_{z,i}(t)|z \in \{h,c\}, i \in S\}$, the evolution of which is given by the following set of ODEs:

$$\frac{d\vec{\mu}(t)}{dt} = \vec{F}(\vec{\mu}(t)), \tag{1}$$

with

$$\vec{F}(\vec{m}) = \sum_{(k,l)\in\Omega} \pi_{k,l}(\vec{m})\vec{f}(\vec{m},k,l)$$

where the drift $\vec{f}(\vec{m},k,l) = \{f_{(z,i)}(\vec{m},k,l)|z \in \{h,c\}, i \in S\}$ is defined below and $\vec{\pi}(\vec{m}) = \{\pi_{k,l}(\vec{m})|(k,l) \in \Omega\}$ is the invariant probability vector of $K(\vec{m})$, where $(K(\vec{m}))_{i,j} = P[J(t+1) = i|J(t) = j, M(t) = \vec{m}]$, with $i,j \in \Omega$. The entries of $K(\vec{m})$ are described in detail further on.

Given that the GC algorithm is executed while $M(t) = \vec{m}$, we denote the probability that the GC algorithm selects a type (z,i) block as $p_{z,i}(\vec{m})$. As we rely on the d-choices GC algorithm, which simply selects the block with the least number of valid pages among a set of d randomly selected blocks, we have

$$p_{z,i}(\vec{m}) = \left[\left(\sum_{s=i}^{b} m_s\right)^d - \left(\sum_{s=i+1}^{b} m_s\right)^d\right]\frac{m_{z,i}}{m_i},$$

if $m_i > 0$ (and zero otherwise), where $m_j = m_{h,j} + m_{c,j}$ for $j \in S$, as all the selected blocks must contain at least i valid pages, but not all should contain $i+1$ and ties are broken randomly. For further use define $p_{z,i+}(\vec{m}) = \sum_{s>i} p_{z,s}(\vec{m})$.

The drift $f_{(z,i)}(\vec{m},k,l)$ represents the expected change in the number of type (z,i) blocks in between two points of observation given that the occupancy measure equals $\vec{m}$ and the number of pages written to the HWF and CWF respectively equal k and l at the first point of observation. As explained below, this leads to (where $m_{h,b+1} = m_{c,b+1} = 0$ to ease the notation)

$$f_{h,i}(\vec{m},k,l) = \tag{2}$$
$$\begin{cases} r\frac{(i+1)m_{h,i+1} - im_{h,i}}{b\rho f} & k < b, l < b, \\ -p_{h,i}(\vec{m}) & i < b, k = b \text{ or } l = b, \\ 1 - p_{h,b}(\vec{m}) - p_{c,(b-l)+}(\vec{m}) & i = b, k = b, \\ p_{h,(b-k)+}(\vec{m}) - p_{h,b}(\vec{m}) & i = b, l = b, \end{cases}$$

and

$$f_{c,i}(\vec{m},k,l) = \tag{3}$$
$$\begin{cases} (1-r)\frac{(i+1)m_{c,i+1} - im_{c,i}}{b\rho(1-f)} & k < b, l < b, \\ -p_{c,i}(\vec{m}) & i < b, k = b \text{ or } l = b, \\ 1 - p_{c,b}(\vec{m}) - p_{h,(b-k)+}(\vec{m}) & i = b, l = b, \\ p_{c,(b-l)+}(\vec{m}) - p_{c,b}(\vec{m}) & i = b, k = b, \end{cases}$$

with $\rho = 1 - S_f$. First, consider the cases with $k, l < b$, meaning the first point of observation corresponds to a write request. If the request is for a page on a block of type $(z, i+1)$, it invalidates one of its $i+1$ pages and the number of type (z, i) blocks increases by one. Similarly, the number of type (z, i) blocks decreases by one if the request is for a page of type (z, i). As the fraction of hot pages equals f at all times and the spare factor equals $S_f = 1 - \rho$, we have $\sum_{j=1}^{b} j m_{h,j} = b\rho f$ and $\sum_{j=1}^{b} j m_{c,j} = b\rho(1-f)$. Hence, the probability that an arbitrary request is for a page on a block labeled hot with j valid pages given by

$$r j m_{h,j} / b\rho f$$

and similarly for a page on a block labeled cold with j valid pages by

$$(1-r) j m_{c,j} / b\rho(1-f).$$

This explains the expressions for the drift if $k, l < b$.

When either k or l equals b and $i < b$, the number of type (z, i) blocks decreases by one when such a block is selected by the GC algorithm, which occurs with probability $p_{z,i}(\vec{m})$, otherwise the number of type (z, i) blocks remains the same.

Let us now focus on the expected change in the number of (z, b) blocks when $k = b$, meaning the HWF is full and the GC algorithm is executed. When a full cold or hot block is selected (with probability $p_{h,b}(\vec{m}) + p_{c,b}(\vec{m})$) there is no change in the number of (z, b) blocks, while if a cold block with more than $l - b$ valid pages is selected (but not with b valid pages), the CWF becomes full and an extra type (c, b) block is created. In all the other cases, the number of type (h, b) blocks increases by one as the call to the GC algorithm creates a new HWF. The same reasoning applies to the case when $k = l$ and $i = b$.

We end this section by discussing the entries of the transition probability matrix $K(\vec{m})$. Let k and l denote the number of pages written to the HWF and CWF respectively at time t and k' and l' at time $t+1$. If k and l are less than b, the transition is caused by a write request and a single page is written to the HWF (CWF) with probability r $(1-r)$. If $k = b$ the HWF is full and the transition corresponds to a call to the GC algorithm. If the block selected by the GC algorithm is labeled hot and contains j valid pages, then $k' = j$ and $l' = l$. Otherwise, if the selected block is labeled cold with j valid pages, we have two possibilities: either $j \leq b - l$, in which case the valid pages are moved to the CWF and the selected block becomes the new HWF (i.e., $k' = 0$ and $l' = l + j$) or $j > b - l$, meaning $b - l$ of the valid pages go to the CWF and the selected block becomes the new CWF (i.e., $k' = b$ and $l' = j - (b - l)$). A similar argument for $l = b$ leads to

$$K(\vec{m})_{(k,l),(k',l')} = \tag{4}$$

$$\begin{cases} r & k < b, k' = k+1, l = l' < b, \\ 1-r & l < b, l' = l+1, k = k' < b, \\ p_{h,k'}(\vec{m}) & k = b, k' < b, l = l' < b, \\ p_{c,k'}(\vec{m}) & l = b, l' < b, k = k' < b, \\ p_{c,l'-l}(\vec{m}) & k = b, k' = 0, b \geq l' \geq l, \\ p_{h,k'-k}(\vec{m}) & l = b, l' = 0, b \geq k' \geq k, \\ p_{c,b-l+l'}(\vec{m}) & k = k' = b, 0 < l' < l, \\ p_{h,b-k+k'}(\vec{m}) & l = l' = b, 0 < k' < k, \\ p_{h,b}(\vec{m}) + p_{c,b}(\vec{m}) & k = k', l = l', k \text{ or } l = b, \\ 0 & otherwise. \end{cases}$$

3.2. Convergence and numerical solution

The process $\{(X_n^N(t), Y_n^N(t))_{n=1,\ldots,N}, J^N(t)), t \in \mathbb{N}\}$ is clearly a Markov chain. A key feature of this Markov chain is that the state changes of (X_n^N, Y_n^N), for $n = 1, \ldots, N$ depend on (X_k^N, Y_k^N), with $k \neq n$, only through the occupancy measure $M^N(t)$ and $J^N(t)$. As such the model belongs to the class of mean field interaction models studied in [3]. It is not hard to verify that the necessary conditions ($H1$ to $H5$ in [3]) hold and therefore the following theorem follows from Corollary 1 in [3]:

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

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 the ODE (1) with $\vec{\mu}(0) = M^N(0)$. As we are interested in the stationary regime of $M^N(t)$, the question remains whether the convergence extends to the stationary regime. Corollary 2 in [3] shows that it suffices to show that the ODE given by (1) has a unique fixed point that is also a global attractor.

Proving the existence of a global attractor for the set of ODEs in (1) seems problematic as the steady state probabilities $\pi_{k,l}(\vec{m})$ do not appear to have a closed form. Numerical experiments however indicate that there exists a unique fixed point which is a global attractor. Obtaining a closed form expression for this fixed point also appears unlikely (as for the models in [28]).

To generate numerical results we determine a fixed point $\vec{\nu} = \{\nu_{z,i} | z \in \{h, c\}, i \in S\}$ by solving the ODE numerically using Euler's method. The write amplification WA can subsequently be expressed as

$$WA = \frac{b}{b - \sum_{j=0}^{b} j(p_{h,j}(\vec{\nu}) + p_{c,j}(\vec{\nu}))}, \tag{5}$$

as there are on average $\sum_{j=0}^{b} j(p_{h,j}(\vec{\nu}) + p_{c,j}(\vec{\nu}))$ valid pages on a block selected by the GC algorithm and a new HWF or CWF is only selected when full. Euler's method is an iterative method that may require several thousand iterations. During iteration $i+1$ we need to compute the drifts $\vec{f}(\vec{m}^{(i)}, k, l)$ for all $(k, l) \in$

Ω, where $\vec{m}^{(i)}$ is the occupancy vector after i iterations. Hence, we need to determine the steady state probabilities $\pi_{k,l}(\vec{m})$ for some vector $\vec{m}$ during each iteration. As $K(\vec{m})$ is a $b(b+2)$ state Markov chain, this becomes very time consuming for realistic values of b, e.g., $b = 32$ or 64. This computational issue can be resolved as follows.

Partition Ω as $\Omega_{<b} \cup \Omega_{=b}$, where $\Omega_{<b}$ contains all the states with $k, l < b$ and partition $K(\vec{m})$ accordingly

$$K(\vec{m}) = \begin{bmatrix} K(\vec{m})_{<b,<b} & K(\vec{m})_{<b,=b} \\ K(\vec{m})_{=b,<b} & K(\vec{m})_{=b,=b} \end{bmatrix}.$$

First note that the drifts given by (2) and (3) do not depend on k, l whenever $(k, l) \in \Omega_{<b}$ and denote the drift with $k, l < b$ as $f(\vec{m}, < b)$. $F(\vec{m})$ can therefore be written as

$$\vec{F}(\vec{m}) = \sum_{(k,l)\in\Omega_{=b}} \pi_{k,l}(\vec{m})\vec{f}(\vec{m}, k, l) + \left(1 - \sum_{(k,l)\in\Omega_{=b}} \pi_{k,l}(\vec{m})\right) f(\vec{m}, < b),$$

and it suffices to compute the $2b$ steady state probabilities $\pi_{k,l}(\vec{m})$ with $(k, l) \in \Omega_{=b}$.

Second, looking at (4), we note that the b^2 rows of $K(\vec{m})$ corresponding to $(k, l) \in \Omega_{<b}$ do not depend on $\vec{m}$. Thus, $K_{<b,<b}(\vec{m})$ and $K_{<b,=b}(\vec{m})$ are independent of $\vec{m}$ and are denoted as $K_{<b,<b}$ and $K_{<b,=b}$, respectively. The transition matrix $K_{=b}(\vec{m})$ of the Markov chain obtained by censoring $K(\vec{m})$ on the states in $\Omega_{=b}$ can therefore be written as

$$K_{=b}(\vec{m}) = K(\vec{m})_{=b,=b} + K(\vec{m})_{=b,<b}(I - K_{<b,<b})^{-1}K_{<b,=b},$$

where the matrix $(I - K_{<b,<b})^{-1}K_{<b,=b}$ is independent of $\vec{m}$ and needs to be computed only once. Note, the entries of the stochastic invariant vector $\pi_{=b}(\vec{m})$ of the size $2b$ matrix $K_{=b}(\vec{m})$ are identical to $\pi_{k,l}(\vec{m})/c$ with $(k, l) \in \Omega_{=b}$, where c is a normalization constant equal to the fraction of time the Markov chain $K(\vec{m})$ spends in $\Omega_{=b}$:

$$c = \frac{1}{1 + \pi_{=b}(\vec{m})K(\vec{m})_{=b,<b}(-K_{<b,<b})^{-1}e},$$

where the vector $(-K_{<b,<b})^{-1}e$ is independent of $\vec{m}$. Using this approach the time complexity per iteration is reduced from $O(b^6)$ to $O(b^4)$, allowing us to compute a fixed point within seconds for $b = 32$ pages per block.

3.3. Model validation and numerical results

To validate the mean field model, the write amplification for the HCWF approach computed based on (5) is compared to the write amplification obtained

b	S_f	d	r	f	ODE	simul. (95% conf.)
64	0.15	4	0.96	0.24	2.5727	2.5727 $\pm$0.0014
64	0.12	9	0.81	0.08	2.6607	2.6606 $\pm$0.0019
64	0.09	12	0.94	0.02	1.8762	1.8756 $\pm$0.0008
64	0.06	5	0.86	0.13	5.3424	5.3406 $\pm$0.0034
32	0.15	15	0.8	0.07	2.1708	2.1711 $\pm$0.0009
32	0.12	50	0.77	0.2	3.5902	3.5912 $\pm$0.0008
32	0.09	3	0.92	0.12	4.4076	4.4043 $\pm$0.0013
32	0.06	8	0.88	0.03	3.2717	3.2706 $\pm$0.0032
16	0.15	4	0.8	0.05	2.4716	2.4717 $\pm$0.0015
16	0.12	20	0.95	0.15	2.2152	2.2158 $\pm$0.0011
16	0.09	6	0.7	0.2	4.1795	4.1791 $\pm$0.0009
16	0.06	10	0.9	0.1	3.2594	3.2590 $\pm$0.0003

Table 2: The write amplification for the HCWF approach: ODE-based results versus simulation experiments for a system with $N\rho = U = 10,000$ blocks for various parameter settings. Relative errors are less than 0.1%.

by simulating the Markov chain $\{(X_n^N(t), Y_n^N(t))_{n=1,\ldots,N}, J^N(t)), t \in \mathbb{N}\}$, with $10,000$ logical blocks (meaning, $N\rho = 10,000$). Table 2 presents the results for various choices of b, d, r, f and S_f, it shows an excellent agreement between the mean field model and simulation with relative errors below 0.1%. The 95% confidence intervals in Table 2 were computed based on 5 simulation runs, each consisting of $10,000,000$ write requests with a warm-up period of $1,000,000$ requests.

We end this section by briefly looking at the impact of d, the number of choices, on the write amplification for the HCWF approach. In Section 4, we also use this model to compare the performance of the HCWF approach with other approaches. As indicated in [28] the Greedy algorithm (which corresponds to setting $d = \infty$) is not optimal in the presence of hot and cold data and under the DWF approach discussed in [28] there exists an optimal value of d. This can be understood as follows. Consider two blocks, one with mostly hot data and one with mostly cold data and assume the block with the cold data contains one or two more valid pages than the block with the hot data. In this case, the greedy GC algorithm selects the block containing mostly hot data. However, it may be better 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. Thus, if d is set too large, blocks containing only some hot data may not get sufficient opportunities to further invalidate their data.

Figure 1 indicates that the same holds true for the HCWF approach where the optimal choice of d is marked by a star, for $b = 32$, $S_f = 0.08$ and various combinations of r and f. Note, the choice of r and f is denoted as $100r/100f$ in the legend of Figure 1. The optimal value of d clearly depends on the choice of r and f (as well as on b and S_f as indicated by other experiments). When comparing the HCWF with the other approaches we will always set $d = 10$ as

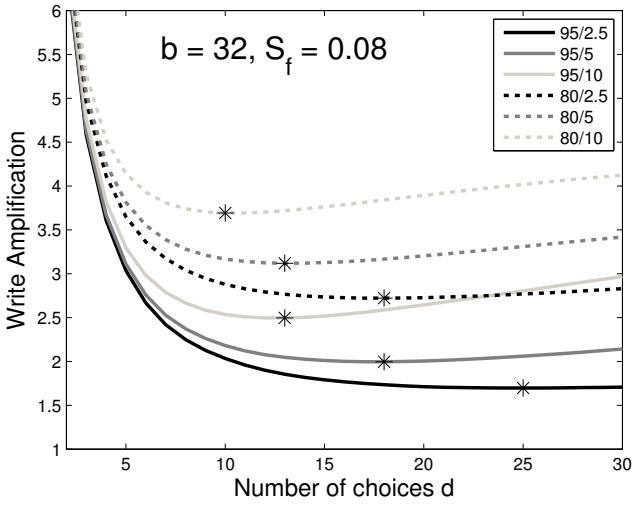

Figure 1: The impact of the number of choices d on the write amplification for the HCWF approach under a Rosenblum workload model for various choices of r and f, with $b = 32$ pages per block and a spare factor $S_f = 0.08$.

optimizing d is hard in practice.

4. Synthetic workloads

In this section we compare the write amplification of the SWF, DWF, HCWF and STAT approaches under a Rosenblum workload model characterized by the parameters r and f (with $r \geq f$). Under such a model there are two types of pages: hot and cold pages, where all the pages of the same type are accessed with equal probability. The probability that an arbitrary request accesses a hot page is denoted as r, while f represents the fraction of hot pages. For instance, for $r = 0.9$ and $f = 0.05$, 90% of the requests go to 5% of the logical address space, while the remaining 95% of the space is only accessed by 10% of the write requests. We focus on the case with $b = 32$ pages per block, but similar conclusions can be drawn for other b values.

The GC algorithm used by SWF, DWF and HCWF is the d-choices algorithm, where $d = 100$ for SWF and $d = 10$ for the DWF and HCWF approach. For the SWF approach numerical results in [28] indicated that setting $d = \infty$ is optimal and setting $d = 100$ gives a near optimal result. For the DWF and HCWF approaches there exists an optimal value for d (see Section 3.3). As this optimal d value depends on the workload characteristics we have set $d = 10$ instead, keeping in mind that a minor reduction in the write amplification can still be expected if d is optimized.

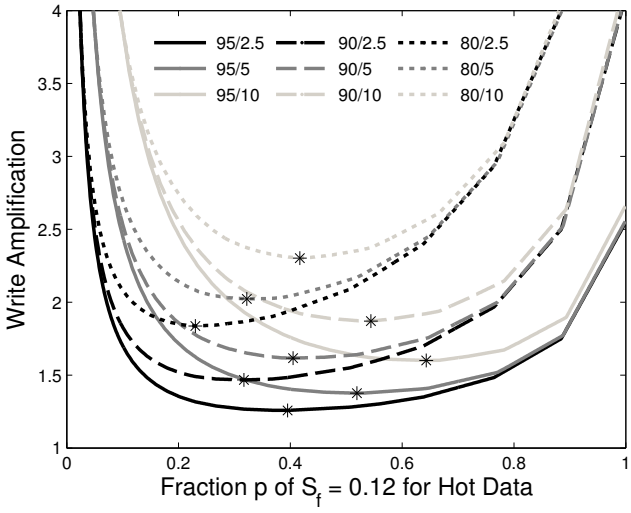

Figure 2: Impact of the fraction p assigned to the hot partition on the write amplification of the STAT approach in an SSD with $b = 32$ pages per block and a spare factor $S_f = 0.12$.

4.1. Perfect data identification

We start by assuming that the hot/cold data identification technique identifies the hot pages in a perfect manner, meaning all the hot/cold pages are correctly labeled and the hot and cold data is perfectly separated in the HCWF and STAT approaches. To determine the optimal fraction p of the spare factor S_f assigned to the hot partition by the STAT approach, we numerically minimize

$$W_{overall}(p) = rW_{hot}(p) + (1 - r)W_{cold}(p),$$

where $W_{hot}(p)$ ($W_{cold}(p)$) is the write amplification on the hot (cold) partition. To determine $W_{hot}(p)$ ($W_{cold}(p)$), we note that each partition can be regarded as an isolated SSD with a uniform random write workload. As such we use the Greedy GC algorithm on each partition (as it is believed to be optimal under uniform random writes) and determine p such that $W_{overall}(p)$ is minimized. Note that increasing p will decrease $W_{hot}(p)$, as the hot partition has a larger spare factor, but increases $W_{cold}(p)$. In other words, we pick p such that the spare factor S_f is distributed among the two partitions in an optimal manner. Figure 2 shows the impact of p on the overall write amplification of the STAT approach (for $b = 32$ and $S_f = 0.12$) and indicates that the optimum is quite broad, especially when the hot data is concentrated on a small fraction of the drive (i.e., 2.5%). It also indicates that the optimal p value is very workload dependent and picking a single near optimal p is not feasible.

Figure 3 depicts the write amplification of the SWF, DWF, HCWF and STAT approach for various choices of r and f with $b = 32$ and $S_f = 0.12$ and 0.08. The mean field models in [28] were used to compute the results for

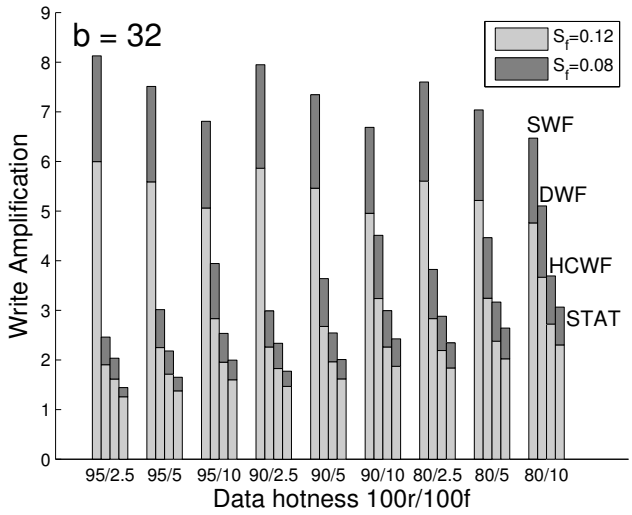

Figure 3: Write amplification of the SWF, DWF, HCWF and STAT approach under a Rosenblum workload model for various choices of r and f, with $b = 32$ pages per block and a spare factor $S_f = 0.12$ and 0.08.

the SWF and DWF approach, the results for HCWF were determined using the mean field model in Section 3, while the results for STAT were found by numerically optimizing p and using the results in [4] to determine $W_{hot}(p)$ and $W_{cold}(p)$.

As expected Figure 3 shows that STAT performs best, SWF is the worst and HCWF outperforms DWF. We also note that the reduction achieved by DWF, HCWF and STAT compared to the SWF approach, grows as the hot data gets hotter (increasing r or decreasing f). More importantly we observe that the reduction achieved by the DWF approach, which does not require a data identification technique, gets closer to the reduction of HCWF and STAT as the hot data gets hotter. In other words, if most of the write requests are concentrated on a small fraction of the drive, there is limited additional gain in implementing a hot/cold data identification technique even if the technique separates the data in a perfect manner. We also note that while the performance of STAT is superior to the HCWF approach, it does rely on the optimization of the fraction p which was noted to be quite workload dependent (see Figure 2).

4.2. Impact of false positives and negatives

In this section we investigate the impact of having false positives and negatives in the hot/cold data identification technique, where false positives are cold pages that are incorrectly labeled as hot and false negatives are hot pages that are labeled cold. False positives and negatives are a genuine concern for many data identification techniques, for instance in [11] as many as 30% of the identified hot pages may be false positives if the number of hash functions or the hash

table size is too small. In this section we will look at the impact of having much fewer false positives and negatives (up to 6%). False positives and negatives clearly only affect the write amplification of the HCWF and STAT approaches, as the SWF and DWF do not rely on hot/cold data identification. Further, as the SWF approach is clearly inferior to DWF, we will compare HCWF and STAT with DWF only. Let f_p (f_n) represent the probability that a false positive (negative) occurs.

For the STAT approach we need to determine the optimal choice of the parameter p (the fraction of S_f allocated to the hot partition). First note that in the presence of false positives/negatives STAT identifies a fraction

$$f_{hot} = f(1 - f_n) + (1 - f)f_p,$$

as hot data and a fraction $f_{cold} = 1 - f_{hot}$ as cold. Further, the probability that a request goes to the hot partition is given by

$$r_{hot} = r(1 - f_n) + (1 - r)f_p,$$

while $r_{cold} = 1 - r_{hot}$. For a given p, we need to determine the write amplification on each partition, which is done using the mean field model in [28] as the workload on each partition corresponds to a Rosenblum model with parameters r_{hot}, f_{hot} (and r_{cold}, f_{cold}). The optimal p is subsequently determined using a numerical optimization method.

The write amplification of the HCWF is determined via a customized simulation program as the mean field model of Section 3 does not incorporate false positives or negatives. Although it is possible to extend the mean field model of Section 3 to capture the impact of false positives and negatives, numerically solving the resulting ODE would be very time consuming and would offer little benefit compared to simulation.

Figure 4 compares the write amplification of the HCWF and STAT approach with DWF in the presence of an increasing number of false positives (and no false negatives), for $b = 32$ and $S_f = 0.08$. As the number of false positives increases (from 0% to 6%), the difference between HCWF and STAT on the one hand and DWF on the other becomes less significant, especially when the hot data is concentrated on a small fraction of the drive, e.g., 2.5%. Thus, a substantial part of the gain provided by implementing a hot/cold data identification technique vanishes even in the presence of a small number of false positives. We can also see that the STAT approach suffers more from false positives than HCWF. Similar results in case of false negatives (and no false positives) are presented in Figure 5, except that the increase in the write amplification is somewhat smaller.

The impact of having both false positives and negatives is depicted in Figure 6. As expected the DWF results are even closer to HCWF and STAT. In fact, in some cases (mostly with $f = 0.025$) DWF even outperforms the HCWF and STAT approach, while HCWF is also superior to STAT in some cases as it suffers less from having false positives and negatives. In short, if the hot data is highly concentrated, implementing a hot/cold data identification technique

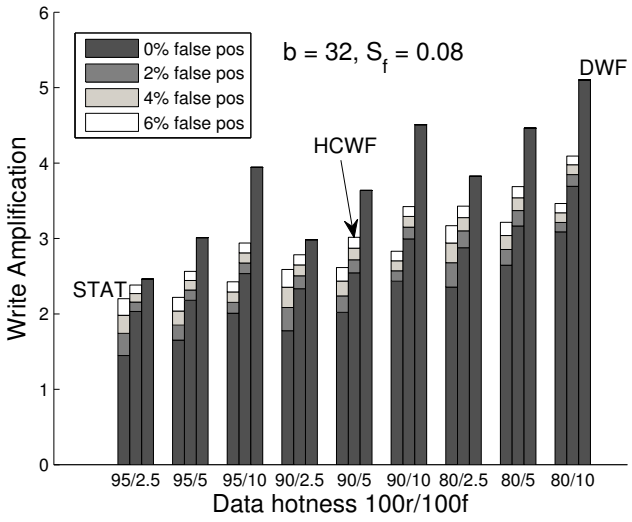

Figure 4: The effect of false positives on the write amplification of the STAT, HCWF and DWF approach under a Rosenblum workload model for various choices of r and f, with $b = 32$ pages per block and a spare factor $S_f = 0.08$.

with a false positive/negative probability above 5% may deteriorate the system performance compared to simply relying on the DWF approach.

5. Trace-based workloads

The synthetic workloads considered in Section 4 assume that all the pages can be classified into two types of pages, where pages of the same type are updated equally often. As such it is obvious to label one type of pages as hot and the other type as cold. Intuitively it is also clear that the HCWF and STAT approaches are very effective on such a workload. In reality we are however faced with a popularity distribution, where each page has its own access rate. As such it is far less obvious where to draw the line between hot and cold data. Clearly, the hot data corresponds to the most popular pages, but it is not clear which fraction of the pages should be labeled hot to achieve the greatest benefit. Note that most data identification techniques aim at identifying hot data [5, 11, 23], meaning they aim at partitioning the logical address space in two parts, although some schemes have been proposed that attempt to partition the space in more parts [7]. In this section we will compare the SWF, DWF and HCWF approaches using trace-based simulation experiments. We do not consider the STAT approach as determining the optimal fraction p using simulation would be very time consuming and not feasible in practice.

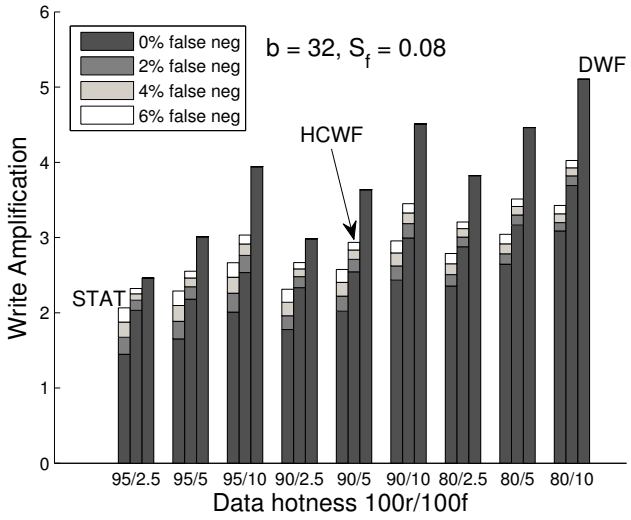

Figure 5: The effect of false negatives on the write amplification of the STAT, HCWF and DWF approach under a Rosenblum workload model for various choices of r and f, with $b = 32$ pages per block and a spare factor $S_f = 0.08$.

5.1. I/O Workloads

For the trace-based simulation experiments presented in this section, we made use of the following real-world I/O traces:

- **rsrch0** [22, 1]: an I/O trace collected at a server supporting research projects at Microsoft Research.
- **prxy0** [22, 1]: an I/O trace containing requests of a Firewall/web proxy server at Microsoft Research.
- **online** [29, 2]: an I/O trace of a coursework management workload on Moodle at a university.
- **webmail** [29, 2]: an I/O trace of webmail traffic on a university department mail server.

More specifically, we first preprocessed the above traces by aligning the offset of each request to a multiple of 4 KB (all the offsets in the traces are multiples of 512 bytes). Requests with sizes above 4 KB (if present) were subsequently split into several (sequential) requests such that all requests have a size of at most 4 KB. Some statistics on the trace files after this processing was done are listed in Table 3. It lists the percentage of the requests that are write requests (%Writes), the number of requests in the trace (#Requests), and the percentage of the accessed logical block address (LBA) space that is only read (%LBA_RO). Table 4 provides information regarding the data locality of the write operations.

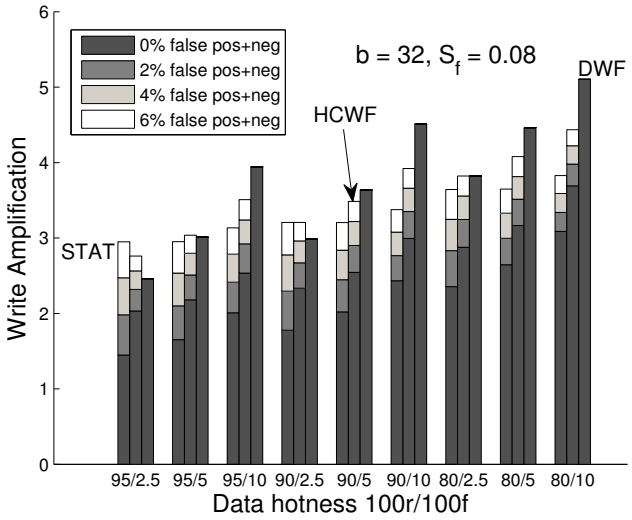

Figure 6: The effect of false positives and negatives on the write amplification of the STAT, HCWF and DWF approach under a Rosenblum workload model for various choices of r and f, with $b = 32$ pages per block and a spare factor $S_f = 0.08$.

I/O trace	%Writes	#Requests	%LBA_RO
rsrch0 [22]	88.87	3,253,639	19.02
prxy0 [22]	96.36	22,136,692	19.53
online [29]	73.88	5,700,499	64.87
webmail [29]	81.86	7,795,815	55.19

Table 3: Data set statistics. %Writes: percentage of writes, #Requests: number of request and %LBA_RO: the size of the LBA space that is only read.

More specifically, it lists the fraction of the accessed LBA space that corresponds to the most frequently written data, e.g., 80% of the writes in the rsrch0 trace go to 14.11% of the accessed logical block address space.

The SSD used in the trace-driven simulation experiments is composed of $U = \lfloor x/b \rfloor$ logical blocks, where x is equal to the number of logical pages accessed during the trace. The number of physical blocks N is determined by U by means of the spare factor S_f, i.e., $U = N(1 - S_f)$. In other words, a fraction $(1 - S_f)$ of the pages is in the valid state at all times during the simulation, while all of the logical pages are accessed at least once during the simulation, but data is only written to some of the pages. The data is initially placed in an unfragmented manner on the drive such that the first U physical blocks contain all the valid pages and the remaining $N - U$ blocks contain only pages in the erase state.

To make the simulation runs sufficiently large we also adopted the *replay* method used in prior SSD work [17, 21]. By replaying a trace, we simply mean

	rsrch0	prxy0	webmail	online
10%	0.0000	0.0003	0.0002	0.0001
20%	0.0001	0.0006	0.0007	0.0007
30%	0.0003	0.0009	0.0015	0.0202
40%	0.0005	0.0012	0.0026	0.0440
50%	0.0025	0.0015	0.0050	0.0680
60%	0.0348	0.0040	0.0221	0.0920
70%	0.0879	0.0252	0.0451	0.1161
80%	0.1411	0.0464	0.0685	0.1404
90%	0.1967	0.0677	0.1127	0.1649

Table 4: Data locality of the writes, e.g., 80% of the writes in the rsrch0 trace go to 14.11% of the accessed logical block address space.

that the I/O pattern of the trace is repeated a number of times without change such that the overall trace length exceeds $50,000,000$ requests. This implies that all pages are updated several times during a single run, unless the page is only read during the original trace. The numerical results presented in Table 5 are based on 10 simulation runs each such that the 95% confidence intervals are sufficiently small.

5.2. Numerical results

In this section we compare the write amplification of the SWF, DWF and HCWF approaches using trace-based simulation experiments. As the HCWF approach relies on a hot data and cold identification technique, we consider five different choices for the fraction f_{hot} of the pages labeled hot: $0.0025, 0.005, 0.01, 0.02$ and 0.04, meaning the fraction of pages labeled hot varies between 0.25% and 4%. Note that for any data identification technique the fraction of hot pages is either explicitly defined, e.g., as in [5] by the size of the hot list, or implicitly determined by the parameters of the identification technique, e.g., as in [11] and [23] where this fraction can dynamically change over time. In either case, it is hard to optimize this fraction (or the related parameters) to minimize the write amplification as it is workload dependent (as shown further on). We also limit ourselves to the setting with perfect hot data identification, introducing false positives or negatives would further increase the write amplification of the HCWF approach somewhat.

Table 5 compares the write amplification of the SWF, DWF and HCWF approaches for $S_f = 0.06, 0.10$ and 0.14, while $b = 64$ (similar results are observed for other b values). For each of the three approaches the d-choices GC algorithm was used, with $d = 10$. A first observation is that the optimal fraction of hot data f_{hot} (among the values considered) to minimize the write amplification is workload dependent: 1% for rsrch0, 2% for prxy0, 0.25% for online and 1% for webmail. Second, the results of the DWF approach are always much closer to the HCWF results than to the SWF results, so the DWF approach captures most of the gain that the HCWF approach offers when compared to the SWF approach.

S_f	SWF	DWF	HCWF 0.25%	HCWF 0.5%	HCWF 1%	HCWF 2%	HCWF 4%
rsrch0 trace							
0.14	2.843	1.785	1.604	1.559	**1.535**	1.549	1.583
0.10	3.739	2.095	1.979	1.915	**1.876**	1.897	1.945
0.06	5.601	2.826	2.825	2.752	**2.694**	2.731	2.799
prxy0 trace							
0.14	3.330	1.363	1.490	1.480	1.458	**1.443**	1.459
0.10	4.258	1.623	1.779	1.763	1.734	**1.719**	1.744
0.06	6.105	2.273	2.484	2.458	2.414	**2.393**	2.427
online trace							
0.14	1.513	1.429	**1.418**	1.421	1.424	1.430	1.431
0.10	1.907	1.761	**1.712**	1.718	1.723	1.733	1.743
0.06	2.830	2.506	**2.443**	2.452	2.464	2.483	2.502
webmail trace							
0.14	1.859	1.430	1.474	1.421	**1.399**	1.419	1.444
0.10	2.414	1.729	1.801	1.722	**1.685**	1.715	1.755
0.06	3.600	2.441	2.592	2.481	**2.408**	2.455	2.519

Table 5: Impact of the fraction of data labeled hot on the write amplification of the 10-choices GC algorithm when using the HCWF approach compared to the SWF and DWF approaches ($b = 64$ pages per block).

We also note that in some cases the HCWF approach outperforms the DWF approach for each of the fractions f_{hot} considered (e.g., rsrch0 trace), while in other cases the DWF approach is superior in all five cases (e.g., prxy0 trace). Overall, it is fair to state that the write amplification of the DWF approach is close to that of the HCWF approach and may even be below if the fraction of hot data is not properly chosen. Thus, even if the hot and cold data identification technique generates no false positives or negatives, the HCWF approach may be inferior to the DWF approach for some workloads given that a fixed fraction f_{hot} of the pages is labeled hot. This makes the added value of some existing data identification techniques questionable with respect to reducing the write amplification as it complicates the design of the FTL driver without creating a very clear additional benefit.

6. Conclusions

Hot and cold data identification techniques are known to be very useful in reducing the write amplification on flash-based solid state drives (by separating the hot and cold data), but also contribute to the complexity and memory requirements of the FTL driver. In this paper we compared the write amplification of the write approach discussed in [28], called the DWF approach, with two write approaches that do require hot and cold data identification. The DWF approach separates writes triggered by the GC algorithm and operating system, respectively, and therefore does not rely on any hot and cold data identification technique.

Numerical results, based on both mean field models and simulation experiments, demonstrated that the DWF approach can capture a substantial part of the reduction achieved by the two write approaches that rely on hot and cold data identification (especially as the hot data gets hotter). Further, in case the fraction of data labeled hot is not properly chosen or in case of a substantial number of false positives/negatives occur (e.g., 5%), the DWF approach may even become superior. These results make the necessity of implementing a hot and cold data identification technique questionable with respect to reducing the write amplification as the added value is rather limited.

For future work we also intend to compare these write approaches under dynamic hot and cold data workloads (meaning workloads for which the id and amount of hot pages changes over time) as well as for systems supporting the TRIM command.

Aknowledgement

This work was supported by the FWO Flanders via research project G024714N entitled *Design and analysis of garbage collection algorithms for flash-based solid state drives.*

[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] Li-Pin Chang and Tei-Wei Kuo. An adaptive striping architecture for flash memory storage systems of embedded systems. In *Proceedings of the Eighth IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'02)*, RTAS '02, pages 187–, Washington, DC, USA, 2002. IEEE Computer Society.

[6] 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.

[7] 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.

[8] 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.

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

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

[11] 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.

[12] 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.

[13] J. Kang, H. Jo, J. Kim, and J. Lee. A superblock-based flash translation layer for NAND flash memory. In *In EMSOFT 2006: Proceedings of the 6th ACM IEEE International conference on Embedded software*, pages 161–170. ACM, 2006.

[14] Jesung Kim, Jong Min Kim, Sam H. Noh, Sang Lyul Min, and Yookun Cho. A space-efficient flash translation layer for compactflash systems. *IEEE Transactions on Consumer Electronics*, 48:366–375, 2002.

[15] S. Lee, D. Shin, Y.-J Kim, and J. Kim. LAST: locality-aware sector translation for NAND flash memory-based storage systems. *SIGOPS Oper. Syst. Rev.*, 42(6):36–42, October 2008.

[16] S.-W. Lee, D.-J. Park, T.-S. Chung, D.-H. Lee, S. Park, and H.-J. Song. A log buffer-based flash translation layer using fully-associative sector translation. *ACM Trans. Embed. Comput. Syst.*, 6(3), July 2007.

[17] 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.

[18] Sang-Phil Lim, Sang-Won Lee, and Bongki Moon. FASTer FTL for enterprise-class flash memory SSDs. In *Proceedings of the 2010 International Workshop on Storage Network Architecture and Parallel I/Os*, SNAPI '10, pages 3–12, Washington, DC, USA, 2010. IEEE Computer Society.

[19] 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.

[20] 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.

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

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] B. Van Houdt. Analysis of the d-choices garbage collection algorithm with memory in flash-based ssds. In *Proceedings of Valuetools, Torino (Italy)*, DEC 2013.

[27] 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.

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

[29] 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.

[30] 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.