Algebraic Modeling of Write Amplification in Hotness-aware SSD

Yue Yang
University of Toronto
yyang@eecg.toronto.edu

Jianwen Zhu
University of Toronto
jzhu@eecg.toronto.edu

Abstract

Garbage collection is essential to the endurance of NAND flash-based solid state drives (SSDs). Analytic modeling of garbage collection reveals the non-trivial relationship between endurance, a performance metric manifested as write amplification, and the algorithmic design variables, device configurations and workload characteristics. This work builds on recent advances in analytic modeling that targets hotness-aware victim selection algorithms, and improves them using a simpler system of algebraic equations. We show that such a model retains the precision of predicting write amplification, with less than 5% error, while reducing the running time from several hours, as the case in previous model of ordinary differential equations, to merely a few seconds. Equipped with such models, we show how design insights can be obtained to guide performance tuning.

Categories and Subject Descriptors C.4 [*Performance of systems*]: Modeling techniques

General Terms modeling, design

Keywords Flash Translation Layer (FTL), Garbage Collection, Write Amplification, Performance Modelling

1. Introduction

The market development of NAND flash-based Solid State Drive (SSD) is outstripping that of hard disk drives [Troxel 2013]. The dramatic decrease in price and recent technology breakthroughs in capacity, reliability, and speed have made the flash memory a viable storage medium in both consumer devices and enterprise solutions. However, the underlying NAND flash technology imposes an endurance limit on the SSD device, ranging from a hundred to a hundred thousand erase/write cycles for each physical block. Once the limit is exceeded, the data stored are no longer considered reliable.

Unfortunately, the erase operation is required to precede any in-place update of a flash memory cell. To avoid high latency and shortened device lifetime due to this "erase-before-write" constraint, a log-structured write scheme must be employed. In this situation, the drive writes data at a new location while it marks the old data as *invalid*, with flash management software known as Flash Translation Layer (FTL). This scheme necessitates a *garbage collection* (GC) mechanism that reclaims invalid data and consolidates free space for subsequent updates. In this GC process, the copied valid data unproductively consume write budget, as shown by the dash arrows in Figure 1. As a result, the actual amount of data written to the flash memory usually exceeds the user requested write amount (solid arrows), a phenomenon known as *write amplification*. Understandably, the write amplification is destructive to the effective lifetime of flash-based storage devices, making them impractical in the long-term for large storage.

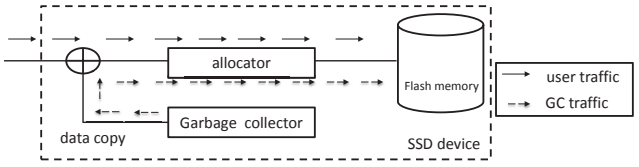

Figure 1. Write traffic in SSD

Thanks to contributions made in the context of log-structured file systems [Rosenblum and Ousterhout 1991] and, later, the flash community [Chen et al. 2011, Park and Du 2011], the side effects of the GC process are well-understood and have been mitigated. Nevertheless, these works are mainly experimental and validated usually by trace-driven simulations. Like any experimental methodology, trace-driven simulations suffer from long running time. In addition, they are incapable of uncovering the non-trivial relationship between write amplification and design variables. Analytical instruments are therefore urgently needed, particularly in large-scale SSD systems.

SYSTOR '15, May 26–28, 2015, Haifa, Israel.

http://dx.doi.org/10.1145/2757667.2757671

Recently SSD researchers propose several analytic models, allowing FTL designers to analyze design trade-offs in write amplification reduction [Bux and Iliadis 2010, Desnoyers 2014, Van Houdt 2013a]. Despite significant results, these models either adopt the impractical greedy method of the GC selection algorithm or they fail to incorporate the complexity of hotness characteristics in real workloads, preventing them from being used in real FTLs. Although [Yang and Zhu 2014] present a model that addresses the two concerns, their solution relies on an ordinary differential equation (ODE) based system derivation. Such a model could suffer from a long run-time, typically hours, when the design parameters scale. Therefore, a convenient solution for realistic, high-efficiency GC algorithms with a generic traffic model is still lacking in the literature to date.

To bridge this gap, we first employ the d-Choice selection algorithm that is demonstrated to provide an excellent trade-off between ease of use and GC performance [Van Houdt 2013a]. Second, we fit generic workloads into a multi-tiered traffic model in compliance with data separation schemes in recent SSD works [Hsieh et al. 2006, Park and Du 2011]. Third, we develop system state derivatives and characterize the steady state by a system of algebraic equations, from which the write amplification is derived. To the best of our knowledge, this is the first analytical work presenting an algebraic solution that evaluates the write amplification with d-Choice selection for a generic traffic. While demonstrating similar accuracy to the ODE based approach [Yang and Zhu 2014], this model enjoys a substantial run-time reduction, allowing efficient design trade-offs explorations. Based on analytical results, we provide insights on performance optimization for hotness-aware flash management algorithm designers.

The rest of this paper proceeds as follows. In Section 2 we explain the FTL concepts and define the problem. Section 3 presents model notations and analytic derivation towards the output, i.e. the write amplification factor. We demonstrate the accuracy of the proposed model by a comparison to simulated results in Section 4. With the validated predictive power, we provide design insights on performance optimization in Section 5. We give detailed review of related prior work in Section 6 and conclude the paper in Section 7.

2. Problem description

The I/O behaviour of flash memory is fundamentally different from that of hard disks owing to its unique erase-before-write and wear-out characteristics. Flash memory blocks must be erased before it can store new data, and can only sustain a limited number of erase-program cycles, usually from as little as 100 to a more typical 3,000 or 100,000 times. Flash memory is organized in units of pages and blocks. A flash page in modern models is 8KB or 16KB in size and a block has 128 or 256 flash pages.

FTL is a set of flash management functions designed to conceal the peculiarities of flash memory and to provide the traditional block-level interface, which exposes an array of logic blocks to the upper-level software stack. To avoid the costly erase-before-write cycle on blocks, overwriting a logical page is done by performing an *out-of-place update*. An unoccupied and erased location holds the updated data and this newly allocated page is marked as *valid* or *live*, leaving the physical page containing old data *invalid*. This is analogous to the write process in log-structured file systems.

2.1 Garbage collection

Out-of-place updates necessitate a garbage collection (GC) process to reclaim the space for invalid data. This cleaning process is triggered once the percentage of free blocks in the system drops below a threshold. The GC works by (1) selecting a victim block; (2) copying live data remaining in the block to a new place; and (3) erasing the block and moving it to the free block pool. If the victim block contains V live pages out of B, the total gain is $B - V$. As migrating the live pages unproductively consumes free pages, it is considered a source of write amplification. It is numerically defined as the ratio of total number of physical page written to number of logical page writes, or formally,

$$A = \frac{B}{B - \overline{V}} \tag{1}$$

where $\overline{V}$ is the mean of the number of live pages in selected victim blocks.

Throughout the paper, we use a d-Choice algorithm [Li et al. 2013, Van Houdt 2013a] as the selection algorithm in the GC process. Originating from the selection algorithm in load balancing [Mitzenmacher 2001], the d-Choice provides an excellent trade-off between ease of use and performance. It employs a selection window of size d that defines a random subset of d out of total N blocks to be selected. Moreover, the window size d is a tunable parameter that enables the designer to perform design trade-off exploration in the spectrum between the random and greedy policies. Intuitively, the random selection ($d = 1$) maximizes wear-leveling, while the greedy selection ($d \to \infty$) minimizes the cleaning cost, meaning the write amplification. In fact, the d-Choice manages to achieve a negligible write amplification difference to the greedy selection with a d value such as $d = 10$ [Van Houdt 2013a].

2.2 Free space characterization

We refer the space available to perform the out-of-place updates as *free space*. It is well known that the fraction of free space is a key parameter for write amplification in flash [Baek et al. 2007]. A *live data ratio*, ρ, characterizes this relation, which is numerically defined as the ratio of the number of written pages in user space to the number of available physical pages. In practice, we always have $\rho < 1$.

Understandably, ρ is traffic dependent and thus correctly characterizes the fraction of live data in the system.

2.3 Data Placement

The cleaning efficiency of GC largely depends on data placement. Collocated hot and cold data tend to create more GC traffic when the block is cleaned. This is so because the cold data invalidates slowly. It is thus wise to group data with similar update frequencies in blocks to form a skewed distribution of live data in the device, facilitating the selection algorithm in choosing a good candidate block for cleaning. This strategy is well researched in related literature [Hsieh et al. 2006, Park and Du 2011], known as *data separation by hotness*.

The separation is implemented by *write frontier* (WF) blocks, through which user data are written into flash memory. Pages in a frontier block, or *open* block, are sequentially written until the block becomes full. Once it is full, the flash controller considers the frontier *closed* and allocates a new one from the free block pool as a new WF. User data identified as the same hotness tiers are written into the same WF. In this work, we assume, during system operations, that no more than one WF is designated to one hotness tier.

Once the traffic is separated, written data forms multiple disjoint hotness regions in the flash array, where allocation and clean can be self-regulated with its own FTL. This provisioning allows us to handle the complexity of overall write amplification derivation by modeling the performance in the individual hotness region, as explained later in Section 3.

Moreover, in this work we only consider and measure the write amplification introduced by GC. In doing so we ignore the cost of the read operation, which is significant in a disk-based file system, but much less so in flash-based ones. We also ignore the extra writes introduced by algorithms other than GC, such as the address translation by FTL; we also ignore the cost of erase operations.

3. System Model

3.1 The overall framework

Write amplification, denoted by A, is a function of a n-tier traffic model T, device configurations C and a garbage collection algorithm G, or formally,

$$A = f(T, C, G) \tag{2}$$

Notations used in this paper are summarized in Table 1 and discussed in detail in the rest of this section.

3.2 Traffic model

The traffic model T characterizes the write skew in workload, abstracted by an *n-tier* model

$$T = \langle U, w, n, \vec{r}, \vec{l} \rangle \tag{3}$$

where U is the number of pages in LBA space; w is the portion of LBA space to which the write accesses are made;

Table 1: Modeling notations

Symbol		Description
Input parameters		
traffic model	$T.U$	number of pages in LBA space
	$T.w$	fraction of non-read-only data in LBA space
	$T.n$	number of hotness tiers
	$T.\vec{r}$	a vector of fractions of write count
	$T.\vec{l}$	a vector of fractions of LBA space
device config.	$C.N$	number of physical pages in device
	$C.B$	number of pages per block
	$C.\vec{R}$	a vector of free space provisioning
GC config.	$G.d$	selection window size of d-Choice
Output value		
A		write amplification factor
Key intermediate variables & constants		
ρ		live data ratio
M		system state matrix
π		steady system state
P		probability matrix of victim block selection

$\vec{r} \in \{\langle r_1, \ldots, r_n \rangle \in \mathbb{R}^n | r_i \geq 0 \wedge \sum_{i=1}^{n} r_i = 1\}$ and $\vec{l} \in \{\langle l_1, \ldots, l_n \rangle \in \mathbb{R}^n | l_i \geq 0 \wedge \sum_{i=1}^{n} l_i = 1\}$. T characterizes a distribution of logical page hotness. Each page falls into one of n disjoint tiers, denoted by $i \in \{1, 2, \ldots, n\}$. Each tier is characterized by r_i and l_i, meaning that a fraction r_i of write requests are made on the pages in tier i, taking a fraction l_i of the LBA space. The n-tier traffic model has two advantages: 1) it is simple enough to express the write skew analytically; and 2) it is sufficiently general to approximate real I/O workloads. To fit a real I/O workload into the n-tier model, a page rank is created with update frequency. Then a binning scheme is applied to the rank, forming bins where pages with similar hotness can be grouped. Unless otherwise stated, the write skew is assumed stationary over the time.

3.3 Device configuration

The device is characterized by $C = \langle N, B, R \rangle$, where N is the number of physical pages; B is the number of pages per block, or block size; and $\vec{R} \in \{\langle R_1, \ldots, R_n \rangle \in \mathbb{R}^n | R_i > 0 \wedge \sum_{i=1}^{n} R_i = 1\}$, where R_i is the fraction of free space provisioned for writing data in hotness tier i.

We can now derive *live data ratio* $\rho = \frac{U \times w}{N}$, where U and w are defined in Section 3.2. It represents the live data fraction in the device. For the multi-tiered case, each hotness region independently maintains a live data ratio ρ_i, numerically defined as the ratio the number of logical pages in tier i to the number of physical pages in region i, i.e.,

$$\rho_i = \frac{l_i}{l_i + R_i \times (\frac{1}{\rho} - 1)} \tag{4}$$

Table 2: An exemplary snapshot of system state in SSD with $N = 64$, $B = 4$ and $n = 2$ ($0 \le j \le B$ and $1 \le i \le n$). The number of blocks of a type is given in the bracket.

m_j^i	j=0	j=1	j=2	j=3	j=4
i=1	0	0	3.8%(1)	23.1%(6)	73.1%(19)
i=2	0	0	5.3%(2)	28.9%(11)	65.8%(25)

3.4 System state modeling

The type of a physical block, denoted by $\langle i, j\rangle$, is jointly defined by the hotness tier $i \in \{1, \ldots, n\}$, the block has been allocated for, and the number of live pages contained in the block $j \in \{0, 1, 2, \ldots, B\}$. Thus, the system state at time t can be described by a matrix

$$M(t) = \begin{pmatrix} m_0^1 & m_1^1 & \cdots & m_B^1 \\ m_0^2 & m_1^2 & \cdots & m_B^2 \\ \vdots & \vdots & \ddots & \vdots \\ m_0^n & \cdots & \cdots & m_B^n \end{pmatrix}$$

where m_j^i is the fraction of blocks of type $\langle i, j\rangle$ in hotness region i at time t. Note that the sum of each row in M is 1 at any time.

Table 2 shows an instantaneous system state of an exemplary device containing 64 blocks, 4 pages per block. The blocks are allocated for one of the two hotness tiers. In the example, 26 blocks have been allocated for tier 1, among which 73.1% (19 out of 26) contain 4 live pages, i.e., the fraction of blocks with type $\langle 1, 4\rangle$ is 73.1% at the moment.

3.5 Garbage collection

Using the d-Choice selection algorithm, garbage collection is characterized by a sole parameter d that defines a random subset of d out of a total of $N_i = N \times R_i$ blocks in region i. We define a victim probability matrix P, where each element p_j^i is defined to be the probability a type $\langle i, j\rangle$ block is selected as victim for cleaning in region i, as follows.

$$P = \begin{pmatrix} p_0^1 & p_1^1 & \cdots & p_B^1 \\ p_0^2 & p_1^2 & \cdots & p_B^2 \\ \vdots & \vdots & \ddots & \vdots \\ p_0^n & \cdots & \cdots & p_B^n \end{pmatrix}$$

Given $G = \langle d\rangle$, we have

$$p_j^i = \left(\sum_{k=j}^{B} m_k^i\right)^d - \left(\sum_{k=j+1}^{B} m_k^i\right)^d \tag{5}$$

The first term in Equation (5) gives the probability that all the randomly selected d blocks in tier i have no less than j live pages, while the second term is the complement of the probability that at least one has exactly j live pages. Thus, the difference is the probability that a block containing j live pages is selected. Note that the sum of each row in P is 1.

3.6 State transition

The state transitions are driven by both user writes and data migration in GC. More specifically, the user writes increase the number of valid pages in write frontiers and decrease the number in some of other blocks due to data invalidation. The GC process creates free blocks while increases the number of valid pages in write frontiers.

Within the region i, the probability that a valid page in a block of type $\langle i, j\rangle$ blocks is updated by an external write is the ratio of the number of valid pages contained in the blocks of type $\langle i, j\rangle$ to the total number of user pages in tier i, i.e.,

$$u_j^i = \frac{NR_i m_j^i j}{BNR_i\rho_i} = \frac{1}{B\rho_i} \times m_j^i \times j \tag{6}$$

where ρ_i is given in Equation (4).

Let Δm_j^i be the expected change to m_j^i between two consecutive cleanings of the victim blocks, or a *cleaning interval*. For $j < B$,

$$\Delta m_j^i = \left(\sum_{k=0}^{B} p_{B-k}^i \times k\right) \times (u_{j+1}^i - u_j^i) - p_j^i \tag{7}$$

The first term $\sum_{k=1}^{B} p_{B-k}^i \times k$ represents the mean of the number of user writes within a cleaning interval. Any such write to a block of type $\langle i, j+1\rangle$ increases the total number of type $\langle i, j\rangle$ blocks in the system, while a request to a block of type $\langle i, j\rangle$ decreases the number. In addition, we also lose a type $\langle i, j\rangle$ block due to the cleaning.

We notice that[1]

$$\sum_{k=1}^{B} p_{B-k}^i k = B - \sum_{k=1}^{B} (c_k^i)^d \tag{8}$$

where $c_j^i = \sum_{k=j}^{B} m_k^i$ is the cumulative fraction of blocks containing at least j live pages in hotness tier i. Equation (8) can be understood as follows. The right-hand side is the mean of the number of free pages gained from a GC process, which should match the mean of the number of user writes serviced between the cleaning interval, represented by the left-hand side in the equation. For simplicity, let $\beta_i = B - \sum_{k=1}^{B} (c_k^i)^d$.

For $j = B$,

$$\Delta m_j^i = \sum_{k=0}^{B} p_k^i \times b_0^{B-k, k/BWl(i)} - p_j^i - \beta_i \times u_j^i \tag{9}$$

where $b_i^{q,p} = \binom{q}{k} p^i (1-p)^{q-i}$ is the binomial probabilities. The latter two terms can be understood as before, while the

[1] Proof of Equation (8) is available upon request.

first term states that a type $\langle i, B\rangle$ block is formed (as a write frontier) by k pages copied from the immediately preceding GC process cleaning a block of type $\langle i, k\rangle$ and $B-k$ recent user writes demanding pages in a tier i block, none of which invalidates the former k pages. Note that $k/BWl(i) \ll 1$, $b_0^{B-k,k/BWl(i)} \approx 1$. Equation (9) can be reduced to

$$\Delta m_B^i = 1 - p_B^i - \beta_i \times u_B^i \tag{10}$$

3.7 Steady State

A necessary and sufficient condition of steady system state, denoted by π, is $\Delta m_j^i = 0$ at π, for all $i \in \{1, \ldots, n\}$ and $j \in \{0, \ldots, B\}$.

Uniform Traffic We first examine the steady state for uniform traffic, or $n = 1$. When $d = 1$, the victim block is randomly chosen and thus $p_j^1 = m_j^1$. From Equation (7) and Equation (10), $\Delta m_j^1 = 0$, $0 \le j \le B$, yields $\pi_1 = \langle m_0^1, \ldots, m_B^1\rangle$, where

$$m_j^1 = \begin{cases} \frac{1}{1+(1/\rho-1)\cdot B} & , j = B \\ m_B^1 \cdot \prod_{k=1}^{B-j} \frac{(1/\rho-1)\cdot(B-k+1)}{1+(1/\rho-1)(B-k)} & , 0 \le j < B. \end{cases} \tag{11}$$

We then generalize the d-Choice with $d \ge 2$. Assuming $(c_B^1)^d \ll 1$, we have $c_B^1 = \frac{\rho_1}{\beta_1}$ from Equation (10). For $j < B$, from Equation (7) and, we obtain a system of algebraic equations[2]

$$(c_j^1)^d + \frac{j\beta_1}{B\rho_1}(c_j^1 - c_{j+1}^1) = 1 \tag{12}$$

Note that β_1 is a function of c_j^1, $0 \le j \le B$. Thus the equations can be solved numerically [3].

Figure 2 shows a comparison of system steady states predicted by the proposed model and the simulation results for different ρ and d values. The simulation performed on a device with $B = 32$ and $N = 131,072$ is driven by a uniform workload of 20 million write requests. The x-axis represents the block type, and y-axis indicates the occupancy. The comparison demonstrates that, under all cases, our model provides a good approximation of the steady system state.

Skewed Traffic Directing skewed traffic into different hotness regions where garbage collection is independently performed implies that each hotness tier is regulated by an independent flash translation layer (characterized by ρ_i). In this way, the problem is reduced to the uniform traffic case by which the steady state of the whole device can be constructed, i.e.,

$$\pi = \langle \pi_1, \ldots, \pi_n\rangle \tag{13}$$

where π_i represents the steady state of hotness region i.

[2] Derivations of Equation (11) and Equation (12) are available upon request.

[3] MATLAB code for these and other numeric computations in this paper is available upon request.

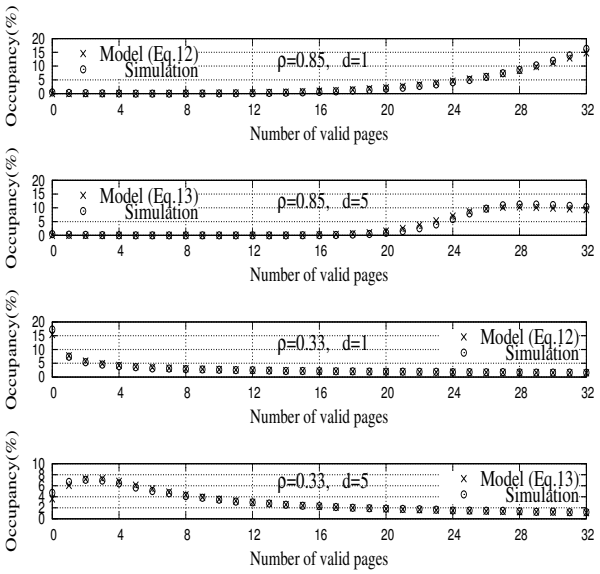

Figure 2. Comparison of model-predicted steady system state and simulation results. X-axis represents block type. Y-axis indicates the occupancy percents. The states are reached by 20 million random writes. ($B = 32$ and $N = 131,072$)

3.8 Write amplification

Once the steady system state π is obtained, write amplification can be immediately derived. As the frequency of GCs invoked in a hotness tier is proportional to the request arrival rate, the overall write amplification is given by

$$A = \sum_{i=1}^{n} r_i \times A_i = \frac{r_i \times B}{B - \sum_{k=0}^{B} p_k^i \times k} \tag{14}$$

where p_k^i is the element of the victim probability matrix under $\pi_i = \langle m_0^i, \ldots, m_B^i\rangle$ with m_j^i, $0 \le j \le B$, solved by replacing m_j^1 with m_j^i in Equation (11) or Equation (12).

4. Model Validation

4.1 Simulation Setup

4.1.1 Datasets

We use both real and synthetic traces to drive the simulations. Filebench [McDougall 2014] trace *fileserver* and *varmail* emulate block I/O behavior of a file server and a mail server, respectively. The real-world traces, obtained from UMass Trace Repository [Storage Performance Council 2002] and [Storage Networking Industry Association 2011], include an OLTP trace *fin2* collected from an online transaction process application running at a large financial institution and a database trace *rad-be* collected via 18-hour SQL replication on a back-end server running at Microsoft.

The write skew in real-world traffic is oftentimes high, and we observe so in the selected traces. The highly skewed traffic implies a low live data ratio and naturally leads to

Table 3: Trace characteristics and simulation configurations

Trace	Characteristics				SSD device configurations				
	Aligned page size (KB)	Number of normalized write requests (million)	Write range (GB)	Working set $N \times w$(# of pages in million)	Configured device capacity (GB)	Total number of dies in SSD	Total number of blocks per die	Block size	Free space partition
synthetic	4	20	15	2.5/3.6/3.8	16	16	8192	32	Equal
fileserver		3	15	1.4					
varmail		3.5	15	1.3					
fin2	8	3.8	9.3	0.8	12	6			
radbe		38.5	81.9	5.5	84	42			

a low write amplification falling into in a narrow range that is close to 1. In order to examine the model accuracy with diverse high live data ratios, we use synthetic traces. Characteristics of the used workloads are given in Table 3. As read requests do not influence our analysis, we record the write requests only.

In a trace, we rank the logical pages by their write counts (hotness) in a descending order and group pages with similar hotness into bins that each is created 2 times wider than the one preceding it. In this way, we obtain hotness tiers with the first bin contains the hottest pages and the last bin contains the coldest. Separation with 2,4 and 6 bins are examined.

4.1.2 SSD configuration

We have used a commercial flash array simulator offered by our industry partner, who is an SSD vendor. The simulator models real devices, and can simulate an arbitrary SSD array configuration with different flash device and parallelism characteristics. It can also drive the input using trace files in DISKSIM format, or using common I/O benchmarking tools such as iometers, vdbench etc. The capacity of the simulated flash array scales to terabyte range. To ensure reproducible and comparable results, we have chosen SSD configuration that is consistent with prior literature, and can be likely handled by academic simulators [Kim et al. 2009].

Some configurations are trace specific. For example, the device capacity is configured slightly larger than the largest LBA referenced in a trace, by using different number of dies. The total number of blocks per die and the block size remain unchanged in all simulations. In multi-tiered cases, by default the free space is equally divided for each hotness region, i.e., $R_i = 1/n$ for $i = 1, \ldots, n$, allowing each region maintains an FTL by itself. Note that the default provisioning setting is used to perform model validation but performance tuning. We later discuss performance optimization implications later in Section 5.

The garbage collection starts once the total number of free pages drops below 0.5% of total pages and ends when this threshold is surpassed. The d-Choice selection configured with $d = 2, 5, 10$ is used. While the garbage collector is busy, user requests are held. In multi-tiered case, the threshold applies to each hotness region.

4.1.3 Simulation behavior

The simulation is profile-based and the configuration space is swept for each of the traces. Each simulation starts with an *empty* state, meaning that the SSD is clean and no data has been written. To make sure that the device reaches a steady state with a sufficiently large number of GC performed, the exercising trace is replayed until more than 100,000 GC counted, a method has been used in prior SSD work [Li et al. 2013]. In multi-tiered case, the counts are ensured in each hotness region in the same way. The write amplification is reported when the simulation ends.

4.2 Validation results

We compare the predicted write amplifications to simulation results for different design parameters, shown in Table 4. Input profiles are differentiated by traffic models, d values as well as device configurations. For synthetic uniform traces, traffic models are differentiated by working set sizes, while filebench and real-world traces, whose working set sizes are fixed, are investigated with different number of tiers used to separate data. As a result, we find that the predictions are quite close in all examined cases. The absolute value of the maximum relative errors is 4.14% and the mean is 1.64%.

4.3 Results agreement

Parameterized modeling enables us to customize our model to some published analytic frameworks [Desnoyers 2014, Van Houdt 2013a] that have been shown highly accurate. In the remainder of this section, we compare our predictions with their results and present the agreement or even better approximations that advocate for our model's predictive power. Appreciating these pioneering studies, we point out that our model provides more analytic capabilities that are not all found in any of the other works, which is later discussed in detail in Section 6.

4.3.1 d-Choice selection algorithm

An ordinary differential equations (ODEs) based model is proposed by [Van Houdt 2013a]. It is shown to be highly accurate to evaluate GC performance with the d-Choice algorithm under uniform traffic. We plug the parameters used in the Table.1 of his paper into our model and find results agreement, as shown in Table 5. The spare factor S_f was

Table 4: Comparisons of predicted write amplifications and trace-driven simulation results. Traffic specific simulation configurations is given in Table 3. d-Choice with $d = 2, 5, 10$ and traffic separated by $n = 2, 4, 6$ tiers are examined.

Trace	d	A (model)	A (sim.)	rel. error	A (model)	A (sim.)	rel. error	A (model)	A (sim.)	rel. error
		$\rho = 0.6$			$\rho = 0.85$			$\rho = 0.9$		
Synthetic uniform ($n = 1$)	2	1.85	1.84	0.54%	4.62	4.61	0.22%	7.25	7.23	0.28%
	5	1.54	1.52	1.31%	3.57	3.54	0.85%	5.11	5.08	0.59%
	10	1.47	1.44	2.08%	3.34	3.30	1.21%	4.74	4.71	0.63%
		$n = 2$			$n = 4$			$n = 6$		
fileserver ($\rho = 0.35$)	2	1.20	1.18	1.69%	1.23	1.21	1.65%	1.30	1.28	1.56%
	5	1.08	1.08	0	1.12	1.10	1.82%	1.18	1.16	1.72%
	10	1.06	1.06	0	1.10	1.08	1.85%	1.15	1.14	0.87%
varmail ($\rho = 0.31$)	2	1.14	1.18	-3.39%	1.14	1.12	1.78%	1.20	1.17	2.56%
	5	1.05	1.07	-1.86%	1.07	1.05	1.90%	1.11	1.09	1.83%
	10	1.03	1.06	-2.83%	1.05	1.04	0.96%	1.09	1.07	1.87%
fin2 ($\rho = 0.48$)	2	1.39	1.45	-4.14%	1.42	1.40	1.42%	1.57	1.52	3.28%
	5	1.21	1.26	-3.97%	1.24	1.23	0.81%	1.36	1.33	2.26%
	10	1.17	1.21	-3.31%	1.20	1.20	0%	1.32	1.28	3.13%
radbe ($\rho = 0.18$)	2	1.05	1.07	-1.87%	1.06	1.05	0.95%	1.11	1.07	3.74%
	5	1.01	1.01	0%	1.02	1.01	0.99%	1.05	1.03	1.94%
	10	1.00	1.01	-0.99%	1.01	1.00	1%	1.04	1.02	1.96%

defined as the ratio of spare space to physical capacity, numerically equivalent to $1 - \rho$ in our work. The block size is set to 64 in the comparison.

Table 5: Model accuracy for d-Choice algorithm. Input parameters are borrowed from the Table.1 in [Van Houdt 2013a], where spare factor S_f is numerically equivalent to $1 - \rho$ in our model.

d	S_f	A ([Van Houdt 2013a])	A(our model)	A (simulation)
2	0.07	9.64	10.05	9.64
4	0.07	7.72	7.72	7.72
8	0.07	7.00	7.00	7.00
2	0.14	4.96	4.97	4.97
4	0.14	4.07	4.07	4.07
8	0.14	3.74	3.74	3.74
2	0.21	3.37	3.37	3.37
4	0.21	2.80	2.80	2.80
8	0.21	2.59	2.59	2.59

4.3.2 Greedy selection algorithm

Desnoyers [Desnoyers 2014] has presented a highly accurate approximation of the write amplification under greedy selection algorithm. We plug the parameters used in the Table.3 of his paper into our model and show in Table 6 the predicted results by the two as well as the simulation results. The variable α was defined as the over-provisioning factor in [Desnoyers 2014] and numerically equivalent to $\frac{1}{\rho}$ in our work. In order to approach the greedy policy, we customize the d-Choice by letting d be total number of physical blocks in device. In all the shown cases, our model yields very close results to the simulation and in addition we do better than the compared when α is close to 1.

Table 6: Model accuracy for greedy GC algorithm. Input parameters are borrowed from the Table.3 in [Desnoyers 2014],where the over-provisioning factor α is numerically equivalent to $\frac{1}{\rho}$ in our model.

α	A (Eq.16 in [Desnoyers 2014])	A (our model)	A (simulation)
1.03	13.71	13.86	13.86
1.05	9.19	9.20	9.20
1.07	7.00	7.00	7.01
1.12	4.53	4.53	4.53
1.20	3.05	3.05	3.05

Table 7: Matlab modeling run-time for modern flash array configurations. A' and T'_{run} are from [Yang and Zhu 2014]

B	ρ	d	A'	A (our model)	T'_{run}	T_{run} (our model)
256	0.93	5	7.80	7.80	4h 57m	4s
256	0.87	10	4.08	4.08	9h 50m	3s
128	0.93	5	7.66	7.66	27m	1s
128	0.87	10	4.03	4.03	1h 6m	1s

4.4 Modeling speed-up

While demonstrating the high accuracy, our model enjoys a substantial speed-up over the ODE-based approaches. Table 7 shows the run-time reduction for predicting the write amplification for random traffic with modern flash array configurations and selected ρ and d, in comparison with the ODE-based model presented in [Yang and Zhu 2014]. The run-time is measured by the 64-bit version Matlab (R2013a) [MathWorks 2013] under Windows 7 OS with 3.70GHz i7-4820K CPU and 16GB system RAM. Our model achieves significant speed-up with orders of magnitude, especially for large block sizes.

5. Design Implications

5.1 Optimization on provisioning

Because write amplification is a monotonically increasing function of the live data ratio, reserving higher share of free space for hotness region i decreases the write amplification A_i, however, at the cost of increasing write amplification in some other hotness regions, making the impact of free space assignment on the overall write amplification non-trivial. Figure 3 shows the write amplification as a function of free space provisioning for four 2-tiered traffic models T_1-T_4. Having the same live data ratio (0.72), they are differentiated by parameters $\vec{r}$ and $\vec{l}$. With $G.d = 5$, we observe that a global optimal provisioning yielding the minimum write amplification exists in each case. For example, $\vec{R} = \langle 0.5, 0.5\rangle$ yields $A_{opt} = 1.61$ for traffic input T_1, the Rosenblum [Rosenblum and Ousterhout 1991] traffic model.

Like Equation (42) in [Desnoyers 2014], A_{opt} can be analytically found by framing Equation (14) into an optimization problem:

$$\begin{aligned} \underset{\vec{R}=\langle R_1,\ldots,R_n\rangle}{\text{minimize}} \quad & A = \sum_{i=1}^{n} r_i \times A_i \\ \text{s.t.} \quad & \sum_{i=1}^{n} R_i = 1, R_i \geq 0. \end{aligned} \tag{15}$$

With solution of this equation[(4)], one can construct hotness regions by specifying the free space assignment determined by our optimization. Figure 4 demonstrates the performance improvement by comparing write amplifications given by the default (equal-sized) and the optimal provisioning for a set of 171 3-tiered traffic models. The traffic models are differentiated by parameter $T.\vec{r} = \langle r_1, r_2, r_3\rangle$ with multiple intervals of 0.05. Working set size $N \times w$ is set to 3 million ($\rho = 0.72$ for 4KB-page 16GB devices) and logical address space is logarithmically binned, i.e. $T.\vec{l} \equiv \langle 1/7, 2/7, 4/7\rangle$. In the figure, The x-axis and y-axis are values of r_1 and r_2, respectively, and the z-axis is the write amplification predicted by our model. The upper surface is the write amplification given by the default free space provisioning, while the lower surface represents the minimum write amplification obtained by the optimization. For example, for $T = \langle 4.2 \times 10^7, 0.72, 3, \langle 0.3, 0.2, 0.5\rangle, \langle 1/7, 2/7, 4/7\rangle\rangle$, the default gives $A = 1.97$ while the optimal yields $A_{opt} = 1.83$. On average, we observe a performance gain of 8% by the optimization, which suggests that the equal-sized free space reservation is a close-to-optimal provisioning scheme.

We have simulated this, creating synthetic workloads and provisioning free space with the default and optimal assignments. Sample results are shown in Table 8 and shown to correspond to the analytic results.

[(4)] Solution is found by the *MultiStart* class implemented in Matlab [MathWorks 2013].

[(5)] $T_5 = \langle 4.2 \times 10^7, 0.72, 3, \langle 0.40, 0.35, 0.25\rangle, \langle 1/7, 2/7, 4/7\rangle\rangle$

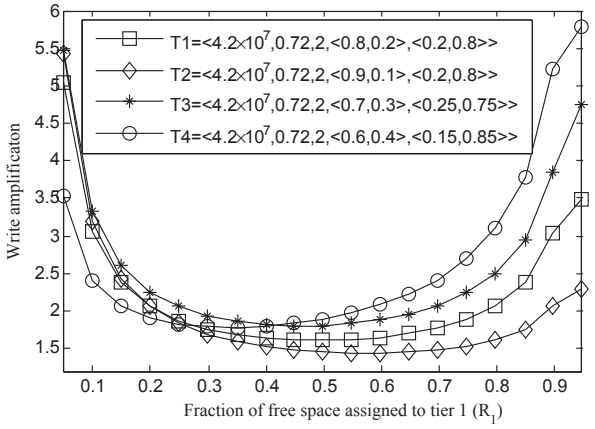

Figure 3. Write amplification as a function of free space provisioning for 2-tier traffic models T_1-T_4. SSD device is of capacity 16GB, block size 32 and 4KB page size. Garbage collection parameters include $d = 5$. The free space provisioning is characterized by $0 < R_1 < 1$ ($R_2 = 1 - R1$).

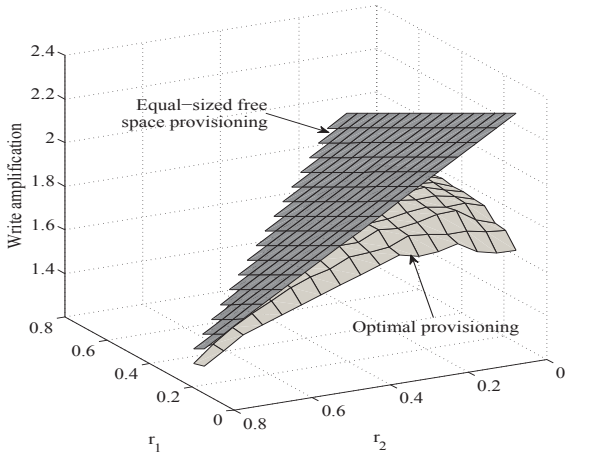

Figure 4. A comparison of write amplifications with equal-sized and optimal free space provisioning. The used 3-tier traffic models are differentiated by parameter $\vec{r} = \langle r_1, r_2, r_3\rangle$ with multiple intervals of 0.05.

Table 8: d-Choice cleaning with optimal free space provisioning for 3-tiered data. ($\rho = 0.72$, $B = 32$)

Traffic model	d	Free space provisioning	A (model)	A (sim.)
T_5[(5)]	2	Equal-size	1.98	1.97
	2	Optimal	1.95	1.94
T_6[(6)]	5	Equal-size	1.64	1.62
	5	Optimal	1.56	1.54

[(6)] $T_6 = \langle 4.2 \times 10^7, 0.72, 3, \langle 0.60, 0.35, 0.05\rangle, \langle 1/7, 2/7, 4/7\rangle\rangle$

5.2 Optimization on data separation

Understandably, increasing the number of hotness tiers can separate the data with a finer granularity and potentially facilitate the garbage collector in selecting good victim block candidates. However, it does not necessarily improve the overall write performance – we have already observed this in Table 4. We consider the overall write amplification A as a function of traffic model T and depict the relationship based on our analytic results, shown in Figure 5. The filebench and real-world traces are examined. For each trace, the traffic models are differentiated by the number of tiers $T.n = 1, 2, \ldots, 6$, assuming the working set is logarithmically binned by data hotness. In addition, $G.d = 5$ and the optimal free space provisioning are used in each case.

We observe that for the shown cases $n = 3$ or 4 yields the lowest write amplification. The optimal tier number depends on the write skew in the given traffic as well as the working set binning scheme. Highly skewed traffic requires less hotness regions partitioned in device, and can be well characterized by the simple hot/cold model. For less skewed traffic, a larger number of bins maybe needed as mixing data of different hotness is known to has negative impact on the cleaning. However, over-binning is bad as well because too many cold regions would reduce the share of free space reserved for hot regions. Analytically, the overall write amplification is dominated by the largest r_i in Equation (14). With the logarithmic binning, over-binning implies the write amplification in coldest region predominates, where the worst cleaning is performed, and it explains the upward tails in Figure 5 for all cases. Unfortunately, we cannot examine all possible binning schemes, of which there are an infinite number.

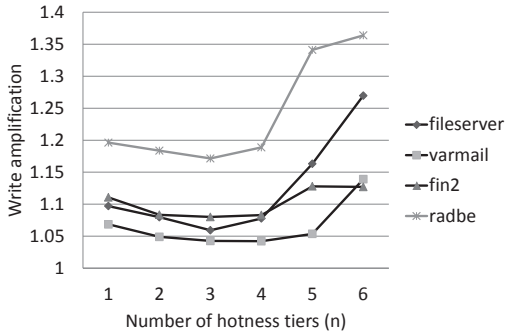

Figure 5. Write amplification on the number of hotness tiers with optimal free space provisioning. User address space is logarithmically binned by data hotness. $G.d = 5$.

5.3 Approximating greedy GC algorithm

Although the greedy selection algorithm yields the lower bound of write amplification under a given system state, it requires an expensive metadata management, preventing it from being used in large systems. The d-Choice algorithm provides a trade-off between the performance and practicality. Understandably, the write amplification is a monotonically decreasing function of d. When $d \to \infty$ the d-Choice corresponds to the greedy policy.

In fact, d-Choice can well approximate the greedy policy with a much smaller d value. Figure 6 demonstrates the approximation with different ρ values. The solid lines plot the write amplification reduction while d increases; and the non-solid reference lines are the lower bounds given by the greedy policy. With $d = 20$, the d-Choice performance is only 2.1% higher than the greedy for $\rho = 0.9$. When d reaches 100, the performance gap shrinks to less than 0.5% for all examined cases. This observation implies that d-Choice algorithm is a good candidate method for garbage collection given its selective power and implementation simplicity even when the system scales. Similar observation was made in [Van Houdt 2013a]. While Houdt [Van Houdt 2013a] demonstrates that small d values suffice to approximate the write amplification of the greedy algorithm for different block size, we confirm that this observation holds true for different live data ratio, in particular, when ρ is small.

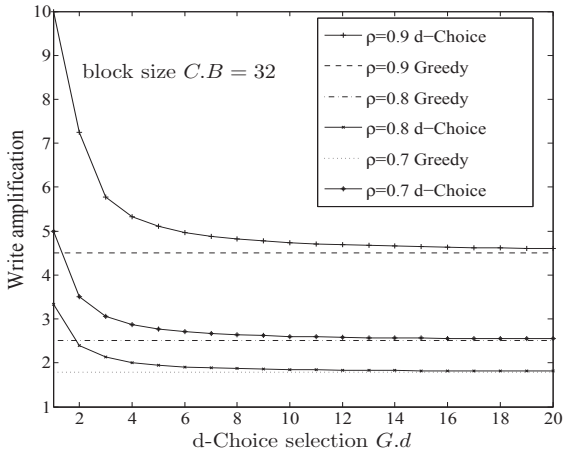

Figure 6. Approximation of greedy algorithm by d-Choice for uniform traffic.

6. Prior Work

Rosenblum's [Rosenblum and Ousterhout 1991] seminal work provides the foundation for developing most of the existing analytic models evaluating garbage collection performance for flash memory based storage system. His initial work is on log-structured file systems where he examines write performance in the context of hard disk drives.

[Ben-Aroya and Toledo 2006] first examines the cleaning cost in flash context by introducing a wear-leveling model. Such a model shows a lower bound on the total writes that a flash device running a simple randomized algorithm can service. Their study also identifies that clustering blocks according to update frequency is advantageous. Despite the significance of its results, some assumptions, such a single-block free pool, prevent this model from being used. [Baek et al. 2007] identifies free space as a key parameter. Unfortu-

nately, this work provides no further investigations on traffic models and the victim block selection method, both of which have substantial impacts on GC performance.

[Bux and Iliadis 2010] derives write amplification with the greedy algorithm from device's steady-state probabilities developed from Markov chain model. However, owing to practical limitations, the model can be used only for very small devices. In the same work, Bux presents a stochastic model that captures the behaviour of large systems, but lacking a closed-form solution. In addition, this work assumes purely random small write traffic, limiting its adoption.

Subsequent to this, [Van Houdt 2013a] employs the Markov model and achieves to characterize the system state for large systems by introducing the mean field theory (MFT). The system evolution is described by a system of ordinary differential equations. The model assesses the number of valid pages per block for a class of GC algorithms, including the d-Choice and Greedy selection algorithms, under uniform traffic. His later work [Van Houdt 2013b] extends this model with two write frontiers, separating GC traffic and user traffic. Houdt's models assume that cold and hot data co-exist in a physical block, while it has been well recognized that separating cold and hot data could significantly reduce the cleaning costs. Using MFT as well, [Li et al. 2013] extends the performance evaluation by revealing trade-off between cleaning cost and wear-leveling. However, the write behaviour in their framework does not rely on the write frontier scheme, which is not practical.

[Yang and Zhu 2014] advances Houdt's framework by introducing hotness-awareness where traffic is generalized by a multi-tier model. In addition, trace-driven simulations, not found in [Van Houdt 2013a,b], have been used for model validation. Despite demonstrated predictive power, this work still relies on ODE-based system derivation, and suffers from a long running time when design parameters scales. For example, given a block size 128 and 4 hotness tiers, a typical configuration in practice, it takes hours for the MATLAB implementation to reach a result.

[Desnoyers 2014] delivers closed-form expressions for write performance with LRU and greedy selection algorithms. Particularly, the work examines uniform and simple hot/cold traffic models. The author provides insights of design trade-off for the non-uniform case. In this case, he investigates an optimal free space partition that would minimize the overall write amplification. While this framework has achieved highly accurate approximation in demonstrated cases, it also has several limitations hindering its wide adoption. First, both LRU and greedy selection algorithms are rarely adopted in actual designs. While the former yields high cleaning cost, the latter is impractical when design parameter scales. Second, it is unclear if their approach may be extended to realistic workloads with the greedy algorithm - this is because no solution and simulation is described.

Prior to Desnoyers' work, several models present closed-form results as well. [Hu et al. 2009] develops a probabilistic model to approximate the performance of the greedy algorithm, but it is shown to be inaccurate under configurations that are commonly found in consumer SSDs. [Agarwal and Marrow 2010] and [Luojie and Kurkoski 2012] study the distribution of live and invalid pages, respectively, in selected blocks. Both works derive an expected value of the number for an asymptotic write amplification. Unfortunately, these studies use the greedy algorithm exclusively and overlook the data hotness.

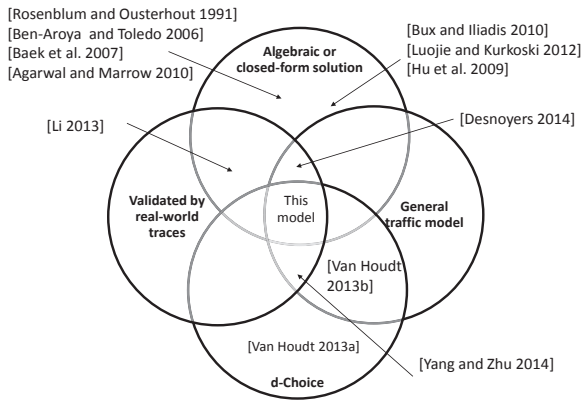

Figure 7. A capability comparison of state-of-the-art analytical frameworks for evaluating garbage collection performance in flash memory based SSD.

Figure 7 shows that our model advances the state-of-the-art by providing more analytical capabilities. These extra capabilities meet the needs of practical FTL design and performance evaluation. These needs include 1) a generic traffic model reflecting the complexity of real-world workloads; 2) a practical and efficient block selection algorithm for garbage collection and 3) an accurate and economically computational modeling process. In addition, simulations driven by both synthetic and real-world traces validate our model – a feature not found in many of previous proposals.

7. Conclusion

In this work, we have presented a highly accurate algebraic approach for evaluating the write amplification in the garbage collection with the d-Choice selection algorithm for general traffic. We validated the model accuracy with simulations, using both an application benchmark and real-world workloads; the errors against these simulations are shown to be within 5%.

With our study, we conclude that the algebraic model of garbage collection can compete in precision with models with ordinary differential equations, while reducing the running time from hours to seconds. It should therefore, as our discussion shows, serve as indispensable tools for design optimizations.

References

R. Agarwal and M. Marrow. A closed-form expression for write amplification in NAND flash. In *GLOBECOM Workshops*, pages 1846–1850, December 2010.

Seungjae Baek, Jongmoo Choi, Donghee Lee, and Sam H. Noh. Model and validation of block cleaning cost for flash memory. In *Proceedings of the 7th International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation*, SAMOS'07, pages 46–54, Berlin, Heidelberg, 2007. Springer-Verlag. ISBN 3-540-73622-0, 978-3-540-73622-6.

Avraham Ben-Aroya and Sivan Toledo. Competitive analysis of flash-memory algorithms. In Yossi Azar and Thomas Erlebach, editors, *Algorithms ESA 2006*, volume 4168 of *Lecture Notes in Computer Science*, pages 100–111. Springer Berlin Heidelberg, 2006. ISBN 978-3-540-38875-3.

Werner Bux and Ilias Iliadis. Performance of greedy garbage collection in flash-based solid-state drives. In *Perform. Eval.*, volume 67, pages 1172–1186. Elsevier Science Publishers B. V., November 2010.

Feng Chen, Tian Luo, and Xiaodong Zhang. CAFTL: A content-aware flash translation layer enhancing the lifespan of flash memory based solid state drives. In *Proceedings of the 9th USENIX Conference on File and Stroage Technologies*, FAST'11, pages 6–6, Berkeley, CA, USA, 2011. USENIX Association. ISBN 978-1-931971-82-9.

Peter Desnoyers. Analytic models of SSD write performance. In *Trans. Storage*, volume 10, pages 8:1–8:25, New York, NY, USA, March 2014. ACM.

Jen-Wei Hsieh, Tei-Wei Kuo, and Li-Pin Chang. Efficient identification of hot data for flash memory storage systems. In *Trans. Storage*, volume 2, pages 22–40, New York, NY, USA, February 2006. ACM.

Xiao-Yu Hu, Evangelos Eleftheriou, Robert Haas, Ilias Iliadis, and Roman 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. ACM, 2009. ISBN 978-1-60558-623-6.

Youngjae Kim, B. Tauras, A. Gupta, and B. Urgaonkar. Flashsim: A simulator for NAND flash-based solid-state drives. In *Advances in System Simulation, 2009. SIMUL '09. First International Conference on*, pages 125–131, September 2009.

Yongkun Li, Patrick P.C. Lee, and John C.S. Lui. Stochastic modeling of large-scale solid-state storage systems: Analysis, design tradeoffs and optimization. In *Proceedings of the ACM SIGMETRICS/International Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '13, pages 179–190. ACM, 2013. ISBN 978-1-4503-1900-3.

Xiang Luojie and B.M. Kurkoski. An improved analytic expression for write amplification in NAND flash. In *Computing, Networking and Communications (ICNC), 2012 International Conference on*, January 2012.

MathWorks. Matlab r2013a documentation, 2013. URL http://www.mathworks.com/.

R. McDougall. Filebench: Application level file system benchmark. http://sourceforge.net/projects/filebench/, 2014.

Michael Mitzenmacher. The power of two choices in randomized load balancing. In *IEEE Trans. Parallel Distrib. Syst.*, pages 1094–1104, Piscataway, NJ, USA, October 2001. IEEE Press.

Dongchul Park and David H. C. Du. Hot data identification for flash-based storage systems using multiple bloom filters. In *Proceedings of the 2011 IEEE 27th Symposium on Mass Storage Systems and Technologies*, MSST '11, pages 1–11, 2011. ISBN 978-1-4577-0427-7.

Mendel Rosenblum and John K. Ousterhout. The design and implementation of a log-structured file system. In *SIGOPS Oper. Syst. Rev.*, pages 1–15, New York, NY, USA, September 1991. ACM.

Storage Networking Industry Association. IOTTA repository, 2011. URL http://iotta.snia.org/.

Storage Performance Council. OLTP application I/O, 2002. URL http://traces.cs.umass.edu/index.php/Storage/Storage.

C. Troxel. SSD & flash storage:How fast is it growing?, 2013. URL http://www.zetta.net/blog/ssd-flash-storage-fast-growing.

Benny Van Houdt. A mean field model for a class of garbage collection algorithms in flash-based solid state drives. In *Proceedings of the ACM SIGMETRICS/International Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '13, pages 191–202. ACM, 2013a. ISBN 978-1-4503-1900-3.

Benny Van Houdt. Performance of garbage collection algorithms for flash-based solid state drives with hot/cold data. In *Perform. Eval.*, pages 692–703, Amsterdam, The Netherlands, The Netherlands, October 2013b. Elsevier Science Publishers B. V.

Yue Yang and Jianwen Zhu. Analytical modeling of garbage collection algorithms in hotness-aware flash-based solid state drives. In *Proceedings of the 30th International Conference on Massive Storage Systems and Technology*, MSST '14, 2014.