Analytical Modeling of Garbage Collection Algorithms in Hotness-aware Flash-based Solid State Drives

Yue Yang, Jianwen Zhu
Department of Electrical and Computer Engineering
University of Toronto
Toronto, ON, Canada
{yyang, jzhu}@eecg.toronto.edu

***Abstract*—Garbage collection plays a central role of flash-based solid state drive performance, in particular, its endurance. Analytical modeling is an indispensable instrument for design improvement as it demonstrates the relationship between SSD endurance, manifested as write amplification, and the algorithmic design variables, as well as workload characteristics. In this paper, we improve recent advances in using the mean field analysis as a tool for performance analysis and target hotness-aware flash management algorithms. We show that even under a generic workload model, the system dynamics can be captured by a system of ordinary differential equations, and the steady-state write amplification can be predicted for a variety of practical garbage collection algorithms, including the d-Choice algorithm. Furthermore, the analytical model is validated by a large collection of real and synthetic traces, and prediction errors against these simulations are shown to be within 5%.**

I. Introduction

Flash-based Solid State Drive (SSD) has seen a surge of adoption in consumer devices. The significant technology breakthroughs in capacity, reliability, speed and the dramatic drop in price, have made the flash memory a viable storage medium in enterprise solutions. Although exposing the same block device interface as hard disk drives, the SSDs are fundamentally different. For example, a write operation in flash has to be preceded by an erase operation. To avoid high latency and shortened device lifetime due to this "erase-before-write" procedure, a log-structured, or *out-of-place*, write scheme is employed, where the data are written at a new location while the old data are marked as *invalid*. Clearly, this scheme necessitates a cleaning mechanism, called *garbage collection* (GC), to reclaim invalid data and consolidate free space for subsequent updates.

To perform the cleaning a block is first chosen by a *victim selection* algorithm. Valid data in the block are copied to new locations, leaving the block with invalid data only. Then the block is erased and returned to the free blocks pool. This implies that extra write operations are needed, and as a result the actual amount of data written to flash usually exceeds the write amount user requested, a phenomenon known as *write amplification*.

978-1-4799-5671-5/14/$31.00

Unfortunately, write amplification not only degrades speed performance but also shortens the device's lifespan. Thanks to the extensive studies in the context of log-structured file systems [23], [15] and flash community [21], [10], [22], the side effects of GC process are well understood. Indeed, many algorithms are proposed to mitigated these effects. Nevertheless, these works are mainly experimental validated by trace-driven simulations. Like any experimental methodology, trace-driven simulations suffer from long running time, and more importantly, are incapable of revealing the relationship between write amplification and algorithmic design variables, as well as workload characteristics. Therefore, analytical tools are urgently needed to explore the design tradeoff, particularly in the instance of large-scale solid state drive systems. One example of such tradeoff is the data hotness, which manifest both as the workload characteristics and the algorithmic design variable, as evidenced by recent works that hotness-aware algorithms can often do significantly better [1], [5], [22], [14].

Luckily, recent progress in analytical modeling of garbage collection algorithms in the context of SSDs has been formed. Nevertheless, the proposed analytical frameworks either overlook the role of data hotness or simply take a simplistic hot/cold model [23] that does not ponder the complexity of hotness characteristics in real workloads.

To bridge this gap, we propose an analytical framework with a generic hotness model. To the best of our knowledge, this is the first analytical framework accommodating multiple hotness tiers for analyzing the d-Choice algorithm [16]. This is accomplished by the abstraction of flash block state space, and the stochastic modeling of the victim block selection process. In particular, we ramp up on the mathematical tool of *mean field theory* (MFT) [20], [2] to approximate the state transition of the large system, thereby reducing the intractable large scale stochastic process into a small deterministic process.

Of course, the idea of mean field analysis is not new and has been used in many different areas [20], [3], [9], [2], [24], [26], [16]. Most relevantly, Van Houdt [24] and Li [16] use MFT to model the performance of garbage collection process in flash-based storage. Our study is inspired by these recent advances and a more detailed discussion on our improvements

can be found in Section VI.

The rest of this paper proceeds as follows. Section II defines the problem. In Section III, we propose a system model, using a Markov chain to capture the system dynamics of an SSD and conduct the mean field analysis. Section IV develops analytic results for selected GC algorithms, followed by comparison to simulated results in Section V, and a discussion of the impact of hotness in GC performance. We give detailed review of related work in Section VI and conclude the paper in Section VII.

II. Problem Statement

We consider a case where $W \times B$ logical pages are written to an SSD device with $N \times B$ physical pages, where 1) W is the *working set* size, in number of blocks, in the workload of interest; 2) N is the total number of physical blocks and $N > W$; and 3) B denotes the number of pages per block, or the *block size*.

Independent of traffic model, hotness of a logical address is identified by one of n tiers. To facilitate the separation of data belonging to different tiers, one physical block in system is designated as a *write frontier* (WF) of each hotness tier; pages are written sequentially in the frontier until it becomes full. Once it is full, the block is *closed* and a new block from a free block list is *opened* as the frontier. As *out-of-place write* is used to overwrite a logical page, the newly written physical page is marked as *valid*, leaving the page containing old data *invalid*. This implies that exactly $W \times B$ pages are valid at all times, provided that the system has been operational for a while, and each logical address has been written at least once.

Once the percentage of the free blocks drops below a threshold, denoted by G, the cleaning, or formally called *garbage collection* (GC), process begins. The GC is done by (1) choosing a victim block by a selection algorithm; (2) migrating valid data remaining in the block to the frontier designated to the corresponding hotness tier; and (3) erasing the block and moving it to the free block list. If the victim block contains V valid pages and $B - V$ invalid pages, the total gain will be $B - V$. As migrating the valid pages unproductively consumes free pages, it is considered as a source of write amplification, numerically defined as the ratio of total number of physical pages written to the number of logical page write requests, or formally,

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

where $\overline{V}$ is the mean of valid pages in chosen victim blocks.

The free blocks preservation can be characterized by a factor $\rho = \frac{W}{N \times (1-G)}$, called *working set ratio*, which is the ratio of written logical address space to the in-use physical space when the system is under steady state. In practice, we have $\rho < 1$.

III. System Model

A. Data Hotness Model

Throughout the paper we consider general workloads abstracted by an n-*tier* hotness model that represents an write distribution where the write frequency of any given logical page falls into one of n disjoint hotness tiers, denoted by $h \in \mathbb{H} = \{1, 2, \ldots, n\}$. Each tier is characterized by $f(h)$ and $r(h)$, shown in Figure 1, meaning that a fraction $r(h)$ of write operations belong to tier h, addressing to a fraction $f(h)$ of the logical address space. A general I/O workload is abstracted through a n-tier workload by grouping pages with similar hotness into the same bin. Pages in a higher tier bin are considered *hotter* than those in a lower tier bin. This hotness model is a generalization of the hot/cold data model in Rosenblum [23] used in [8] and [25], where logical data are considered either hot or cold. In addition, we assume that the access pattern is stationary in this work.

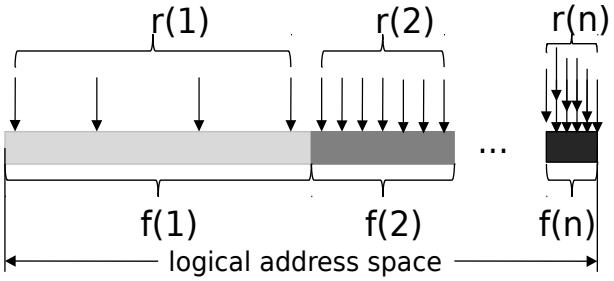

Fig. 1. A general traffic model. The fraction of the total number of page writes with hotness $h \in \mathbb{H}$ is $r(h)$; these pages represent a fraction of $f(h)$ of the overall written address space.

B. Markov Chain Formation

Let set $\mathbb{N} = \{1, 2, \ldots, N\}$ and set $\mathbb{B} = \{0, 1, 2, \ldots, B\}$. We classify each physical block into a type based on 1) the number of valid pages contained in the block; and 2) the hotness tier the block has been allocated for. Specifically, if a block containing tier $h \in \mathbb{H}$ data has exactly $j \in \mathbb{B}$ valid pages, we say the block is in state $\langle h, j \rangle$, or call it a block of type $\langle h, j \rangle$. Considering the free blocks that are not attached to any hotness tier, we extend $\mathbb{H}$ to $\mathbb{H}' = \mathbb{H} \cup \{0\}$, where 0 indicates an undefined hotness tier. Let $s(x, t) = \langle h, j \rangle$, $x \in \mathbb{N}$, denote the state of block number x at time t, then the state descriptor for the SSD is

$$\langle s(1,t), s(2,t), \ldots, s(N,t) \rangle \tag{2}$$

Figure 2 shows a state transition diagram with $B = 4$ and $n = 2$. We have 11 possible states denoted by the nodes and 5 types of state transitions. For example, a write operation performed in a block of type $\langle 1, 0 \rangle$ increases the number of valid pages in it and thus changes the block's type to $\langle 1, 1 \rangle$. At the same time, the updated ("over-written") page in a block of type $\langle 1, 3 \rangle$, for example, becomes invalid and thus change the block's type to $\langle 1, 2 \rangle$. In addition, an erased block is of type $\langle 0, 0 \rangle$ and a newly created write frontier is of type either $\langle 1, 0 \rangle$ or $\langle 2, 0 \rangle$ depending on hotness assignment.

Using the notations in [24], we define the set $\Delta^N = \{\vec{m} = (\vec{m}_1, \vec{m}_2, \ldots, \vec{m}_B)$, where $\vec{m}_j = (m_j^1, m_j^2, \ldots, m_j^n) \in \mathbb{R}^n$, $j \in \mathbb{B}$ such that $0 \le m_j^h \le 1$, $h \in \mathbb{H}$, $\sum_{j \in \mathbb{B}, h \in \mathbb{H}} m_j^h = 1$ and $\sum_{j \in \mathbb{B}, h \in \mathbb{H}} j \times m_j^h = B\rho$. We define *occupancy measure* [2]

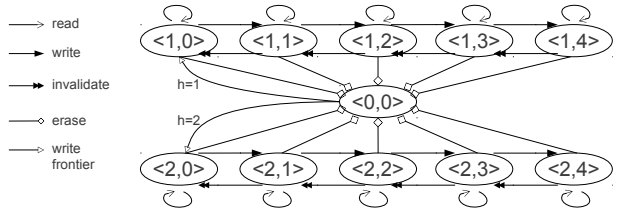

Fig. 2. Block state transitions. ($B = 4$ and $n = 2$)

TABLE I
AN EXEMPLARY SNAPSHOT OF OCCUPANCY FRACTIONS IN SSD WITH $B = 4$ AND $n = 2$

	j=0	j=1	j=2	j=3	j=4
h=0	1/100	0	0	0	0
h=1	5/100	2/100	7/100	3/100	15/100
h=2	10/100	9/100	20/100	8/100	20/100

$M^N : T \mapsto \Delta^N$ with $M(t) = \vec{m}$ as the vector of fractions of type $\langle h, j\rangle$ blocks in system at time t. In other words, the system state can be described by $\vec{m}$ with

$$m_j^h = \frac{1}{N}\Sigma_{x=1}^{N} 1[s(x,t) = \langle h, j\rangle] \tag{3}$$

where m_j^h is the occupancy fraction of type $\langle h, j\rangle$ blocks in SSD at time t.

Table I shows an exemplary instantaneous SSD state described by the occupancy fractions. For example, 9% of blocks are of type $\langle 2, 1\rangle$ at the moment. In addition, since $h = 0$ implies a free block, $m_j^0 = 0$, $\forall j \neq 0$.

We now describe the state transitions of a block by the occupancy fraction. First of all, read requests do not change m_j^h for any h and j. Second, because erasing a victim block and allocating a write frontier always appear in pair in one GC process, we know that $m_0^0 \equiv G$ once the SSD reaches the GC threshold for first time, i.e., $\forall j$, m_j^0 does not change between the GC intervals. Therefore, the values in row $h = 0$ in Table I can be treated as "don't care". For the sake of simplicity, we consider only the non-free blocks for Equation (3) from now on. Third, only write and invalidate (update) operations change the state. We derive the state transition probability as follows. Assuming each logical address has been written at least once, there are in total $m_j^h \times N \times (1 - G) \times j$ valid pages in $\langle h, j\rangle$ blocks. Since the total number of tier h logical addresses is $B \times W \times f(h)$, with an assumption that logical addresses in the same tier have equal probability to be updated, the probability that an external write updates a valid page in a block of type $\langle h, j\rangle$ under system state $\vec{m}$ is given by

$$\begin{aligned} u(h, j, \vec{m}) &= r(h) \times \frac{m_j^h \times N \times (1 - G) \times j}{B \times W \times f(h)} \\ &= \frac{r(h)}{B\rho f(h)} \times m_j^h \times j \end{aligned} \tag{4}$$

Clearly $\vec{m}$ is a Markov chain on space Δ^N, whose evolution is driven by not only the *external* requested writes but also *internal* data migration by GC process. The state transition is discussed in detail in Section III-E.

C. Mean Field Analysis

To understand the dynamics of $\vec{m}$, we employ the mean field model [2] characterized by a set of ODEs, the solution of which can be used to approximate the evolution of the Markov chain.

More specifically, the stochastic process $M^N(t)$ can be approximated by a mean field model by means of a deterministic process $\vec{\phi}(t) = (\vec{\phi_0}(t), \vec{\phi_1}(t), \ldots, \vec{\phi_B}(t))$, given $\vec{m}(t) = \vec{\phi}(t/N)$ as N approaches to infinity. Here $\vec{\phi_j}(t) = (\phi_j^0(t), \phi_j^1(t), \ldots, \phi_j^n(t))$ and $\phi_j^h(t)$ denotes the fraction of blocks of type (h, j) at time t in the deterministic process. The evolution of $\vec{\phi}(t)$ is captured by the following set of ODEs:

$$\frac{d\vec{\phi}(t)}{dt} = \vec{\theta}(\vec{\phi}(t)) \tag{5}$$

where $\vec{\theta}(\vec{m})$, called the *drift* [24] for $\vec{m} \in \Delta^N$, is defined as the expected change of $\vec{m}$ during one GC interval, with $\vec{\theta}(\vec{m}) = (\vec{\theta_0}(\vec{m}), \vec{\theta_1}(\vec{m}), \ldots, \vec{\theta_B}(\vec{m}))$ and $\vec{\theta_j}(\vec{m}) = (\theta_j^1(\vec{m}), \theta_j^2(\vec{m}) \ldots, \theta_j^n(\vec{m}))$ for $j \in \mathbb{B}$. $\vec{\theta_j^h}(\vec{m})$ represents the expected change of the number of type $\langle h, j\rangle$ blocks in GC process provided that the system is under state $\vec{m}$. In other words, for N large and finite t, we can approximate $M^N(t)$ by $\vec{\phi}(t/N)$, which is the unique solution of Equation (5) with $\vec{\phi}(0) = M^N(0)$.

D. Convergence

The convergence exists if five conditions, called *H1* to *H5*, given in [2], are satisfied. With similar reasoning in [24], all the conditions are met given Equation (5). Therefore, the following theorem follows from Corollary 1 in [2].

Theorem 1. *If $M^N(0) \to \vec{m}$ in probability as N tends to infinity, then $\sup_{0 \le t \le T} \left\| M^N(t) - \vec{\phi}(t) \right\| \to 0$ in probability, where $\vec{\phi}(t)$ is the unique solution of the ODE Equation (5) with $\vec{\phi}(0) = \vec{m}$.*

This theorem states that for N large and finite t, we can approximate $M^N(t)$ by $\vec{\phi}(t/N)$, which is the unique solution of the ODE Equation (5) with $\vec{\phi}(0) = M^N(0)$. Corollary 2 in [2] shows that it suffices to show that the ODE given by Equation (5) has a unique fixed point, or a *global attractor* [2].

E. State Transition

The drift $\vec{\theta}(\vec{m})$ changes the system state. We thus derive $\theta_j^h(\vec{m})$ by extending the results developed in [24] with the consideration of data hotness. We define $p(h, j, \vec{m})$ as the probability that a block of type $\langle h, j\rangle$ is chosen by the GC algorithm for cleaning under state $\vec{m}$.

As shown in Equation (5), $\theta_j^h(\vec{m})$ is the expected change to m_j^h between two GC processes. For $j < B$,

$$\begin{aligned} \theta_j^h(\vec{m}) = &\left(\sum_{h=1}^{n} \sum_{k=1}^{B} p(h, B-k, \vec{m}) \times k \right) \times \\ &[u(h, j+1, \vec{m}) - u(h, j, \vec{m})] - p(h, j, \vec{m}) \end{aligned} \tag{6}$$

This expression can be understood intuitively by noting that $\sum_{h=1}^{n}\sum_{k=1}^{B} p(h, B-k, \vec{m}) \times k$ represents the mean of externally requested writes between two executions of the GC algorithm. Any such request to a block of type $\langle h, j+1\rangle$ increases the total number of type $\langle h, j\rangle$ blocks, while a request to a block of type $\langle h, j\rangle$ decreases the total number of type $\langle h, j\rangle$ block. In addition, we also lose a type $\langle h, j\rangle$ block if the selection algorithm chooses such a block under state $\vec{m}$.

For $j = B$,

$$\theta_j^h(\vec{m}) = \sum_{k=0}^{B} p(h, k, \vec{m}) \times b_0^{B-k, k/BNf(h)} - p(h, j, \vec{m}) - \left(\sum_{h=1}^{n}\sum_{k=1}^{B} p(h, B-k, \vec{m}) \times k\right) \times u(h, j, \vec{m}) \tag{7}$$

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 first term states that a type $\langle h, B\rangle$ block is formed (as a write frontier) by k pages copied from the immediately preceding GC process cleaning a block of type $\langle h, k\rangle$ and $B-k$ recent externally requested writes demanding pages in a tier h block, none of which invalidates the former k pages. Note that $k/BNf(h) \ll 1$, $b_0^{B-k,k/bNf(h)} \approx 1$. Therefore, Equation (7) can be reduced to

$$\theta_j^h(\vec{m}) = \sum_{k=0}^{B} p(h, k, \vec{m}) - p(h, j, \vec{m}) - \left(\sum_{h=1}^{n}\sum_{k=1}^{B} p(h, B-k, \vec{m}) \times k\right) \times u(h, j, \vec{m}) \tag{8}$$

Note that Equation (1) is now transformed to

$$A = \frac{B}{B - \sum_{h=1}^{n}\sum_{j=1}^{B} j \times p(h, j, \vec{m})} \tag{9}$$

IV. Analytic Results

A. The d-CHOICES GC Algorithm

The d-Choice GC algorithm [16], [24] randomly selects d blocks and reclaims the block containing the least number of valid pages. d is a tunable parameter with a value range of $1 \le d \le N$. Note that in special cases where $d = 1$ (resp. $d = N$), the algorithm corresponds to the random (resp. greedy) GC algorithm.

A block containing j valid pages is chosen as the victim implies that 1) all selected blocks have at least j valid pages; and 2) at least one of the d blocks has exactly j valid pages. Furthermore, probability of the selected block, given it contains j valid pages, is of type $\langle h, j\rangle$ is $m_j^h / \sum_{h'=1}^{n} m_j^{h'}$. Therefore, with the d-Choice GC algorithm $p(h, j, \vec{m})$ can be jointly written as follows:

$$p(h, j, \vec{m}) = \frac{m_j^h}{\sum_{h'=1}^{n} m_j^{h'}} \left[\left(\sum_{h'=1}^{n}\sum_{k=j}^{B} m_k^{h'}\right)^d - \left(\sum_{h'=1}^{n}\sum_{k=j+1}^{B} m_k^{h'}\right)^d\right] \tag{10}$$

With $g_j^h = \sum_{k=j}^{B} m_k^h$ and $g_{B+1}^h = 0$, $j \in \mathbb{B}$, $h \in \mathbb{H}$, Equation (10) can be written as in Equation (11) by noting that $m_k^h = g_k^h - g_{k+1}^h$.

$$p(h, j, \vec{m}) = \frac{(g_j^h - g_{j+1}^h) \times \left[\left(\sum_{h'=1}^{n} g_j^{h'}\right)^d - \left(\sum_{h'=1}^{n} g_{j+1}^{h'}\right)^d\right]}{\sum_{h'=1}^{n} (g_j^{h'} - g_{j+1}^{h'})} \tag{11}$$

Then we derive dg_j^h/dt for $j \in \mathbb{B}$, $h \in \mathbb{H}$.

$$\frac{dg_j^h}{dt} = \sum_{k=j}^{B} \frac{dm_k^h}{dt} = \sum_{k=j}^{B} \theta_k^h(\vec{m}) \tag{12}$$

Using Equation (10) we can write

$$\sum_{h=1}^{n}\sum_{k=1}^{B} p(h, B-k, \vec{m})k = B - \sum_{k=1}^{B}\left(\sum_{h=1}^{n} g_k^h\right)^d \tag{13}$$

Equation (13) can be understood as follows. The right-hand side is the mean of free pages gained from a GC process, which should match the mean of externally requested writes between two GC executions, represented by the left-hand side in the equation.

Applying Equation (6), Equation (8) and Equation (13) to Equation (12), we obtain the following ODEs for $j \in B$.

$$\frac{dg_j^h}{dt} = \sum_{k=0}^{j-1} p(h, k, \vec{m}) - \beta u(h, j, \vec{m}) \tag{14}$$

where $\beta \triangleq B - \sum_{j=1}^{B}\left(\sum_{h'=1}^{n} g_j^{h'}\right)^d$ in the second term. Since $u(h, j, \vec{m})$ can be replaced by m_j^h, dg_j^h/dt can be further represented by g_j^h and g_{j+1}^h with $m_j^h = g_j^h - g_{j+1}^h$.

Now, we have a system of ODEs ready. The set of ODEs given by Equation (14) have an intuitive interpretation. The first term on RHS represents the rate at which tier h blocks with less than j valid pages are cleaned; or equivalently the rate at which tier h block with j or more pages are produced by GC. β in the second term represents the mean number of user writes that can be accommodated between two executions of GC. Then $\beta u(h, j, \vec{m})$ is the rate at which blocks of type $\langle h, j\rangle$ disappear. Therefore, the combined results on the right hand side is the rate at which the number of blocks of type $\langle h, j\rangle$ varies, which is exactly the meaning of dg_j^h/dt.

TABLE II
SSD PARAMETERS

Parameter	Value
page size	4KB
# of pages per block(B)	32
# of blocks per plane	4096
# of planes per die	2
# of dies per package	4
# of packages in SSD	4
SSD capacity	16GB
GC threshold (G)	5%

As β represents the mean number of pages freed by GC processes, the write amplification A can be written as:

$$A = \frac{B}{\beta} \tag{15}$$

We numerically solve the ODEs given in Equation (14) by Matlab [12] built-in solver $ode45$ assuming that the numbers of valid pages in blocks initially conform to a binomial distribution. We consider the system state to be steady if $\mathrm{Var}\,(dg/dt) < 10^{-4}$.

V. TRACE-DRIVEN EVALUATION

We have thus far characterized the dynamics of GC process and present an analytical model that predicts the write amplification. In this section we validate the accuracy of the model via simulations.

A. System Setup

1) 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 characteristics, number of channels, and number of dies per channel. It can also drive the input using trace files in DISKSIM format, or using common IO benchmarking tools such as iometers, vdbench etc. The capacity of the simulated flash array can scale to the terabyte range. To make sure our reported result can be repeated, and compared apple-to-apple, we have chosen SSD configuration that is consistent with prior literature, and can be likely handled by academic simulators [13].

Table II summarizes the default values of parameters that we use to configure an SSD in our evaluation. The flash array configuration is based on a common SLC SSD [6]. Specifically, the SSD contains 4 flash packages, each of which can process I/O requests in parallel. Each flash package contains 4 dies and each die contains 2 planes consisting of 4096 blocks. Each block contains 32 pages of size 4KB. Thus, each flash package contains 32768 physical blocks in total and the physical capacity of the SSD is 16GB.

We determine the cleaning threshold to 5%, meaning that GC will be activated when the number of free blocks in the system drops below 5% of the total number of blocks in the SSD device.

TABLE III
WORKLOAD CHARACTERISTICS

Trace	write ratio	# of 4KB writes	Write range (GB)	Working set ratio ρ
fileserver	72.5%	2837651	13.3	11.3%
oltp	27.2%	945505	13.3	6.0%
varmail	79.9%	8729692	13.3	1.0%
videoserver	17.8%	836614	13.3	18.8%
webproxy	64.6%	10061642	13.3	1.4%
webserver	26.7%	2059610	13.3	23.2%
fin2	21.5%	7621600	9.3	37.9%
hm	66.0%	7857142	13.9	10.9%

2) Workloads: We describe both synthetic and real traces that drive the simulation. As read requests do not influence our analysis, we record the write requests only. The write requests are normalized to aligned 4KB page accesses.

- *Filebench* [18] is shipped with a library of pre-defined workload personalities, emulating I/O activity for a variety of applications. The personalities include tunables for scaling workloads for specific systems. For instance, the *fileserver* workload file that emulates simple file-server I/O activity performs a sequence of creates, deletes, appends, reads, writes and attribute operations on a directory tree. In this work, we examine six pre-defined workloads including *fileserver*, *oltp*, *varmail*, *videoserver*, *webproxy* and *webserver* [17]. Characteristics of the created workloads are given in Table III. Understandably, with different working set sizes of the selected workloads, ρ, as defined in Section II, varies accordingly.
- *SPC Financial2* [7] is a block I/O trace collected from an online transaction process application running at a large financial institution. There are two financial traces in [7], namely Financia1.spc and Finanacial2.spc. Since the logical addresses in the former span over 10 terabytes, which is too large for the chosen SSD capacity, we only use Financial2.spc, abbreviated to *fin2*. 21.5% of 36 millions 4KB requests are writes, or 7621600 user write requests in total. The logical addresses span 9.3 GB and the working set ratio is 37.9%.
- *hm* [19] is collected from a hardware monitor server in the data center in Microsoft Research Cambridge for a week. The logical addresses span 13.9 GB and the working set ratio is 10.9%.

3) Hotness binning: In this work, hotness of a logical address is measured by update counts. The hotness distributions of selected workload are shown in Figure 3 - Figure 9. While the horizontal axis indexes the hotness, the vertical axis is the number of unique logical addresses. As expected, in general data hotness is inversely proportional to the number of unique logical addresses characterized by the hotness. For example, in Figure 9 a hundred thousand of logical addresses have been written exactly twice, while a few logical addresses have been written more than 10,000 times. By grouping addresses with a similar degree of hotness, the addresses are binned into n tiers using $n-1$ thresholds. In chosen application benchmark and real traces hotness distributions are highly skewed, meaning

TABLE IV
HOTNESS BINNING FOR SELECTED WORKLOADS WITH $n = 2$

Workload	Hotness binning	
fileserver	$r(1) = 0.468$ $r(2) = 0.532$	$f(1) = 0.903$ $f(2) = 0.097$
oltp	$r(1) = 0.403$ $r(2) = 0.597$	$f(1) = 0.672$ $f(2) = 0.328$
varmail	$r(1) = 0.361$ $r(2) = 0.639$	$f(1) = 0.465$ $f(2) = 0.535$
videoserver	$r(1) = 0.199$ $r(2) = 0.801$	$f(1) = 0.894$ $f(2) = 0.106$
webproxy	$r(1) = 0.091$ $r(2) = 0.909$	$f(1) = 0.406$ $f(2) = 0.594$
webserver	$r(1) = 0.445$ $r(2) = 0.555$	$f(1) = 0.904$ $f(2) = 0.096$
fin2	$r(1) = 0.349$ $r(2) = 0.651$	$f(1) = 0.762$ $f(2) = 0.238$
hm	$r(1) = 0.064$ $r(2) = 0.936$	$f(1) = 0.741$ $f(2) = 0.259$

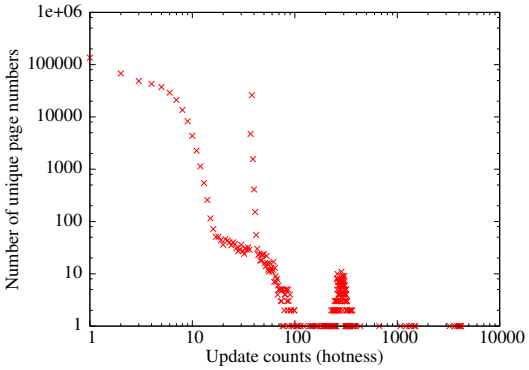

Fig. 3. Hotness distribution of write traffic in *fileserver*

that traffic are dominated by write operations targeting on a very small portion of logical address space. Thus binning the address space with equal width would result in highly skewed values for hotness variables, i.e., $r(h)$ and $f(h)$. We therefore manually choose thresholds ensuring reasonable partitions for both r and f. Binning with $n = 2$ for the selected workloads are given in Table IV. The second column of the table gives the fractions of total number of write operations, while the third column gives the fractions in working set space. For example, the hot (tier 2) data in *fin2*, occupying 65.1% of total writes, are written to 23.8% of working set space, denoted by $r(2)$ and $h(2)$, respectively, as specified in Section III-A.

4) Simulation behavior: One workload is exercised in one simulation. Each simulation starts with an *empty* state, meaning that the SSD is clean and no data has been written. As a warmup, the trace file is loaded once, ensuring each address has been written at least once, after which it is repeated. We collect the cleaning statistics from the second run. To make sure that the device reaches the steady GC state, the exercising trace is repeated until more than 50,000 GC are performed. The write amplification is reported when the simulation ends.

B. Validation by simulation

In Table V and Table VI, we show that with different settings of block size, hotness binning and d-Choice char-

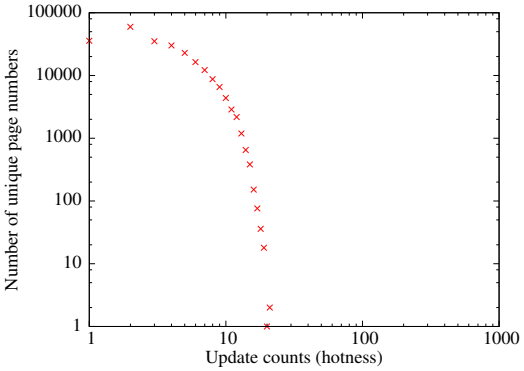

Fig. 4. Hotness distribution of write traffic in *oltp*

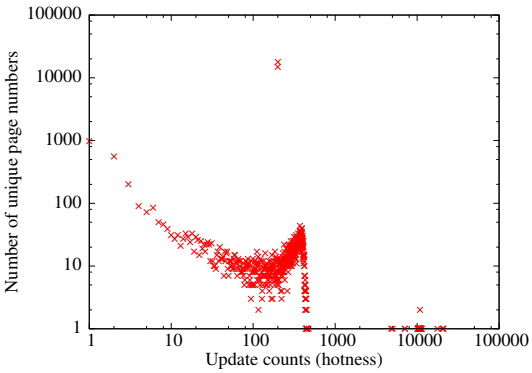

Fig. 5. Hotness distribution of write traffic in *varmail*

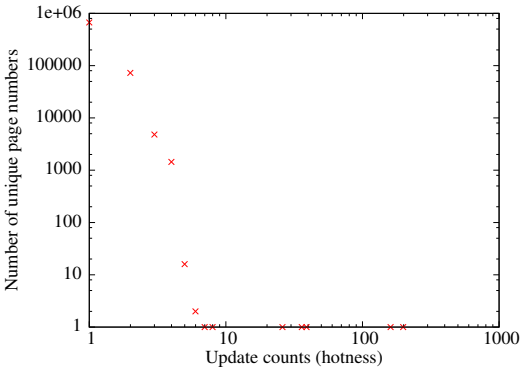

Fig. 6. Hotness distribution of write traffic in *videoserver*

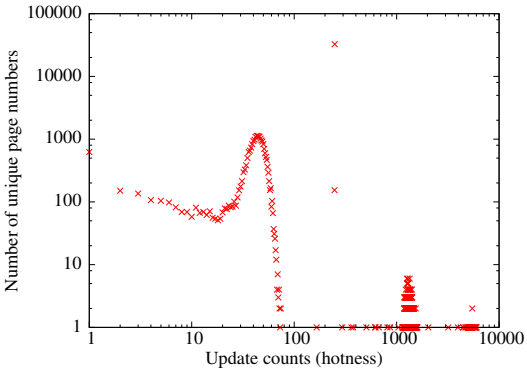

Fig. 7. Hotness distribution of write traffic in *webproxy*

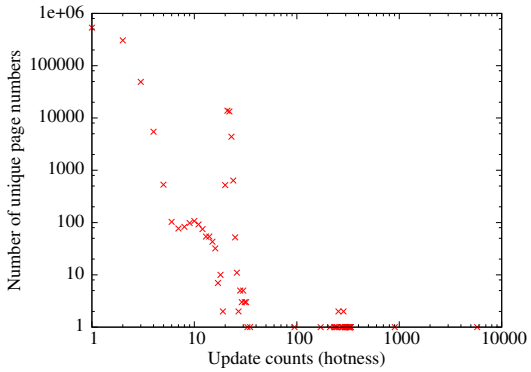

Fig. 8. Hotness distribution of write traffic in *webserver*

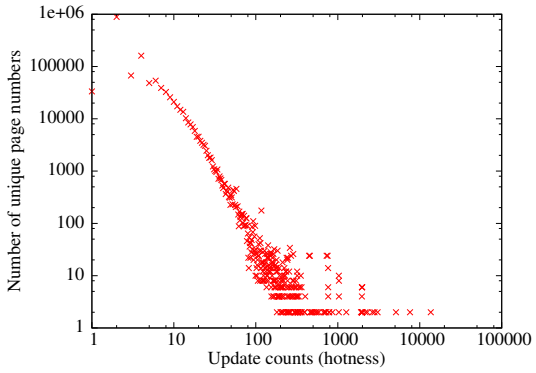

Fig. 9. Hotness distribution of write traffic in *fin2*

acterization, write amplifications predicted by our analytical model agree with simulation results for the selected application benchmarks and real traces with relative errors within 5%. The A (predicted) is the write amplification predicted by our model, while A (simulated) is the result of simulation. Besides, for *varmail* and *webproxy* benchmarks, the write amplifications converge to 1 for all given settings. This is caused by their small working set ratio making the "perfect" victim blocks dominant the disk. Our predictions conform to the simulation results for these two cases.

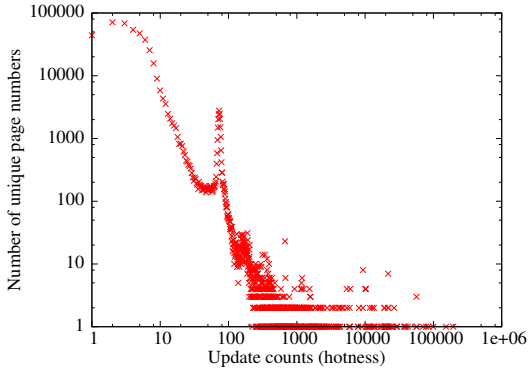

Fig. 10. Hotness distribution of write traffic in *hm*

TABLE V
ANALYTIC VS. SIMULATED WRITE AMPLIFICATIONS FOR APPLICATION BENCHMARK [18] WITH d-CHOICE GC ALGORITHM ($B = 32$)

trace	hotness awareness	d	A (predicted)	A (simulated)	rel. error
fileserver	unaware	1	1.127	1.128	0.1%
		2	1.024	1.024	0
		4	1.002	1.002	0
	aware $n = 2$	1	1.127	1.128	0.1%
		2	1.021	1.018	0.3%
		4	1.001	1.001	0
oltp	unaware	1	1.064	1.064	0
		2	1.007	1.006	0.1%
		4	1.000	1.000	0
	aware $n = 2$	1	1.064	1.064	0
		2	1.008	1.007	0.1%
		4	1.000	1.000	0
videoserver	unaware	1	1.231	1.233	0.2%
		2	1.066	1.037	2.7%
		4	1.017	1.004	1.3%
	aware $n = 2$	1	1.231	1.233	0.2%
		2	1.065	1.036	2.8%
		4	1.014	1.002	1.2%
webserver	unaware	1	1.303	1.306	0.2%
		2	1.102	1.100	0.2%
		4	1.033	1.019	1.4%
	aware $n = 2$	1	1.303	1.304	0.1%
		2	1.100	1.098	0.2%
		4	1.032	1.017	1.5%

C. Comparison with prior work

We compare our predictions with results given by the state-of-the-art analytical frameworks. When $n = 1$, β in Equation (14) reduces $B - \sum_{j=1}^{B} (g_j^1)^d$, conforming to Equation (18) in [24], which corresponds to one hotness tier, a special case of our framework. Table VII shows the agreement with the results based on ODE(18) in [24] with different values of d and ρ.

When $d = 1$, regardless of the value of n, the analytic results conform to the results given by Theorem 2 in [24] for the random GC algorithm. Table VIII shows the agreement between Equation (14) and the closed form results in Theorem 2 in [24] with different values of n and ρ.

When $d \to \infty$, it corresponds to the Greedy GC algorithm. From Equation (10) and Equation (14) we have the following two expressions, respectively:

$$\hat{p}(h, j, \vec{m}) = \begin{cases} \frac{m_j^h}{\sum_{h'=1}^{n} m_j^{h'}}, & \text{if } \sum_{h=1}^{n} g_j^h = 1 \text{ and } \sum_{h=1}^{n} g_{j+1}^h < 1. \\ 0, & \text{otherwise.} \end{cases} \tag{16}$$

and

$$\hat{\beta} = B - max\{j | \sum_{h=1}^{n} g_j^h = 1, \sum_{h=1}^{n} g_{j+1}^h < 0\} \tag{17}$$

, for $j \in \mathbb{B}$ and $h \in \mathbb{H}$.

We obtain Equation (16) and Equation (17) as a new set of ODEs for the Greedy GC Algorithm, expressed by

$$\frac{dg_j^h}{dt} = \sum_{k=0}^{j-1} \hat{p}(h, k, \vec{m}) - \hat{\beta} u(h, j, \vec{m}) \tag{18}$$

TABLE VI
ANALYTIC VS. SIMULATED WRITE AMPLIFICATIONS FOR TRACE *fin2* AND *hm* WITH d-CHOICE GC ALGORITHM

trace	B	hotness awareness	d	A (predicted)	A (simulated)	rel. error
fin2	32	unaware	1	1.611	1.587	1.5%
			2	1.271	1.229	3.4%
			5	1.128	1.083	4.2%
			10	1.096	1.054	4.0%
		aware $n = 2$	1	1.611	1.582	1.8%
			2	1.268	1.212	4.6%
			5	1.127	1.078	4.5%
			10	1.095	1.056	3.7%
	64	unaware	1	1.611	1.585	1.6%
			2	1.279	1.234	3.6%
			5	1.133	1.083	4.6%
			10	1.103	1.054	4.6%
		aware $n = 2$	1	1.611	1.573	2.4%
			2	1.278	1.221	4.7%
			5	1.130	1.093	3.4%
			10	1.097	1.040	5.5%
hm	32	aware $n = 2$	1	1.122	1.124	0.2%
			2	1.022	1.013	0.9%
			4	1.001	1.001	0
	64	aware $n = 2$	1	1.122	1.123	0.1%
			2	1.023	1.015	0.8%
			4	1.002	1.001	0.1%

TABLE VII
ANALYTICAL RESULTS AGREEMENT TO HOUDT [24] WITH $B = 64$ AND $n = 1$

d	ρ	ODE Equation (14)	ODE (18) in [24]
2	0.93	9.63	9.64
4	0.93	7.72	7.72
8	0.93	7.00	7.00
2	0.86	4.96	4.96
4	0.86	4.08	4.07
8	0.86	3.73	3.74
2	0.79	3.37	2.37
4	0.79	2.80	2.80
8	0.79	2.59	2.59

Another way to approximate performance of the Greedy GC algorithm is to use a large d value in d-Choice. $d \geq 10$ manages to achieve a negligible difference [24]. We therefore compare the results given by both Equation (18) and Equation (14) with $d = 100$, shown in Table IX, to Bux [4] where the Greedy cleaning is used. We reach the agreement with the results shown in Figure 1 and Figure 2 in [4] . We set $n = 1$ for this comparison because hotness is not considered in [4].

With hotness awareness, we compare our results to the modeling of the Greedy GC algorithm presented in [8]. For a fair comparison, we create synthetic traces in which 80% data are hot destined to 20% of working set space with different ρ values. Note that ρ is equivalent to the over-provisioning factor α defined in [8], with a value of $1/\alpha$. The traces contains 6-9 million writes ensuring the GC process is fully exercised. Focusing on the steady GC state, we record the numbers of valid pages in the selected blocks for the last 100,000 GC processes and report the write amplifications under the steady state. Predictions by our model demonstrate better approximation

[(1)] $r(1) : r(2) = 0.2 : 0.8$ and $f(1) : f(2) = 0.8 : 0.2$

TABLE VIII
ANALYTICAL RESULTS AGREEMENT TO HOUDT [24] FOR THE RANDOM GC ALGORITHM WITH $B = 32$ AND $d = 1$

n	ρ	ODE Equation (14)	Theorem 2 in [24]
$2^{(1)}$	0.24	1.32	1.32
$4^{(2)}$	0.24	1.32	1.32
2	0.19	1.23	1.23
4	0.19	1.23	1.23
2	0.17	1.20	1.20
4	0.17	1.20	1.20

TABLE IX
WRITE AMPLIFICATION PREDICATED BY EQUATION (18) AND EQUATION (14) FOR GREEDY GC ALGORITHM

ρ	B	A by Equation (18)	A by Equation (14), $d = 100$
0.8	8	2.000	2.157
0.8	16	2.286	2.371
0.8	32	2.461	2.494
0.6	8	1.333	1.310
0.6	16	1.333	1.380
0.6	32	1.391	1.417

D. Design Implications

Since write frontiers are employed to separate data with different hotness, the degree of hotness skewing affects the rates at which data in different hotness bins are updated. As a result, the numbers of valid pages in blocks polarize as the skewness increases. Figure 12 shows write amplifications under different degrees of skewness in a 2-tier hotness model for an SSD with $\rho = 0.95$ and $B = 16$. Lower values of $r(1)$ and higher values of $f(1)$ indicate a higher degree of skewness. With both $d = 2$ and $d = 20$, we observe that the write amplifications tend to decrease while the skewness becomes more intensive. In addition, the write amplification is more sensitive to the skewness change for a larger d value.

Intuitively, the skewness in data hotness affects the rates at which the numbers of different types of blocks vary. More specifically, the higher degree of skewness, 1) the higher rate at which the hot pages are updated; or/and 2) the higher number of blocks containing hot pages in system. The implication is that victim blocks with less valid pages have a higher chance to

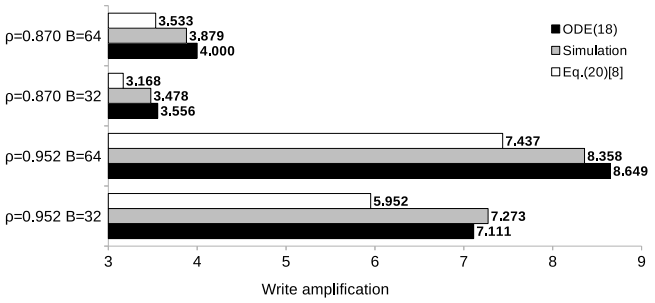

Fig. 11. Comparison of analytical results for Greedy GC algorithm with hotness awareness ($n = 2$, $r(1) = 0.2$, $f(1) = 0.8$)

[(2)] $r(1) : r(2) : r(3) : r(4) = 0.1 : 0.2 : 0.3 : 0.4$ and $f(1) : f(2) : f(3) : f(4) = 0.4 : 0.3 : 0.2 : 0.1$

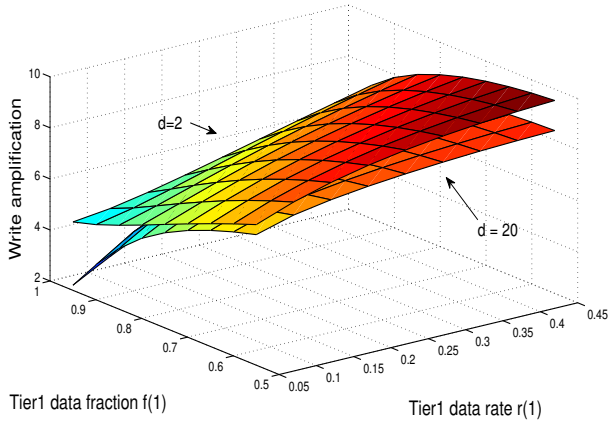

Fig. 12. Sensitivity of write amplifications on degree of hotness skewing. $\rho = 0.95$, $B = 16$, $n = 2$

be selected by the d-Choice algorithm. Analytically, the value of $u(h, j, \vec{m})$ in Equation (14) polarizes while the degree of skewness increases, making the max value of j that satisfies $\sum_{h'=1}^{n} g_j^h = 1$ smaller. Thus, the second term in the expression of β decreases, and the value of β thus increases. As a result, the write amplification A in Equation (15) drops. As $\sum_{h'=1}^{n} g_j^{h'} \leq 1$ for all $j \in \mathbb{B}$, the write amplification declines quicker when the value of d increases.

VI. Related Work

Most analytical frameworks evaluating garbage collection performance in log-structured systems have studied GC algorithms in one of two categories:

1) The w-Windowed GC algorithm [11] picks the block containing the least number of valid pages among $w \in [1, N]$ least-recently-written blocks.
2) The d-Choice GC algorithm [16] picks the block containing the least number of valid pages among $d > 0$ randomly chosen candidates.

Desnoyers [8] presents an analytic framework for the FIFO GC algorithm, a special case of the w-Windowed GC algorithm ($w = 1$). In general the FIFO GC algorithm results in the worst write amplification among windowed algorithms. Clearly, the amplification can be reduced by enlarging the value of w. However, the improvement is quite limited unless the window size is very large [8], [11].

On the other hand, the d-Choice algorithm manages to achieve close to optimal with a small d value [24]. As the greedy GC algorithm, a special case of the d-Choice GC algorithm, always selects the block carrying the least number of valid pages, it offers the best cleaning efficiency. Bux et al. [4] propose a model to analyze the Greedy GC algorithm under the uniform workload, and Desnoyers [8] presents a closed-form solution with an extension of hot/cold data model proposed by Rosenblum [23].

With the same hotness model, Houdt [24] introduces a mean field model to analyze the write amplification of a class of garbage collection algorithms including the greedy GC algorithm and confirms that the windowed GC algorithm is not very effective in reducing the write amplification for small w value. Relying on this framework, Houdt further proposes an approach [25] separating GC traffic and user traffic and shows that the amplification reduces significantly as the hot data gets hotter. Houdt's models assume that pages containing data with different hotness can co-exist in a physical block, while we vision that by designating a write frontier to each hotness the GC cleaning costs can be further reduced. Li et al. [16] unify a class of GC algorithms where the best victim block is selected among a group of randomly chosen blocks. Their work, based on the mean field analysis, extends the performance evaluation by revealing trade-off between cleaning cost and wear-leveling. However, the write behavior in their framework does not seem to rely on the write frontier scheme, which apparently is not practical.

While enjoying the high accuracy, our model advances the state-of-the-art by providing more analytical capability. First of all, our model is adaptive to generic workloads, not relying on any type of traffic model, such as the uniform or hyper-exponential distribution, which are rarely seen in real applications. The only assumption we make is that the statistics in the interested workload are stationary. Second, our framework has the capability of modeling a wider class of GC algorithms other than the Greedy algorithm. Third, the support of multiple write frontiers facilitates the investigation of hotness separation impact on cleaning performance. Furthermore, we perform comprehensive experimental validations and demonstrate the agreement with the results in [4], [24], [25], where the trace-driven evaluations were not presented. A comprehensive comparison of the state-of-the-art analytical frameworks is given in Table VI.

VII. Conclusion

In this paper, we generalize the use of the mean field analysis tool for analyzing the SSD garbage collection performance to A) a general workload traffic model; B) a wider class of garbage collection algorithms; and C) a write-frontier based hotness separation scheme. We find that the predictive power of the proposed analytical model is accurate, and can therefore serve its purpose to explore algorithmic design tradeoffs.

References

[1] Nitin Agrawal, Vijayan Prabhakaran, Ted Wobber, John D. Davis, Mark Manasse, and Rina Panigrahy. Design tradeoffs for SSD performance. In *USENIX 2008 Annual Technical Conference on Annual Technical Conference*, ATC'08, pages 57–70, Berkeley, CA, USA, 2008.

[2] Michel Benaïm and Jean-Yves Le Boudec. A class of mean field interaction models for computer and communication systems. *Perform. Eval.*, 65(11-12):823–838, 2008.

[3] Jean-Yves Le Boudec, David McDonald, and Jochen Mundinger. A generic mean field convergence result for systems of interacting objects. In *Proceedings of the Fourth International Conference on Quantitative Evaluation of Systems*, QEST '07, pages 3–18, Washington, DC, USA, 2007.

TABLE X
A COMPARISON OF STATE-OF-THE-ART ANALYTICAL FRAMEWORKS FOR GARBAGE COLLECTION PERFORMANCE IN FLASH-BASED SSDS

Framework	Workload model	Hotness separation by WF	GC algorithm	Trace-driven validation
Bux [4]	uniform	no	greedy	no
Houdt [24]	uniform	no	d-Choice	no
Houdt [25]	hyper-exponential	no	d-Choice	no
Desnoyers [8]	hyper-exponential	yes	greedy	yes
Li [16]	Poisson	no	d-Choice	yes
This work	general	yes	d-Choice	yes

[4] Werner Bux and Ilias Iliadis. Performance of greedy garbage collection in flash-based solid-state drives. *Perform. Eval.*, 67:1172–1186, 2010.

[5] Li-Pin Chang and Chun-Da Du. Design and implementation of an efficient wear-leveling algorithm for solid-state-disk microcontrollers. *ACM Trans. Des. Autom. Electron. Syst.*, 15(1):6:1–6:36, 2009.

[6] Feng Chen, David A. Koufaty, and Xiaodong Zhang. Understanding intrinsic characteristics and system implications of flash memory based solid state drives. In *Proceedings of the Eleventh International Joint Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS'09, pages 181–192, New York, NY, USA, 2009.

[7] Storage Performance Council. OLTP Application I/O. http://traces.cs.umass.edu/index.php/Storage/Storage, 2002.

[8] Peter Desnoyers. Analytic modeling of SSD write performance. In *Proceedings of the 5th Annual International Systems and Storage Conference*, SYSTOR'12, pages 12:1–12:10, New York, NY, USA, 2012.

[9] Nicolas Gast and Gaujal Bruno. A mean field model of work stealing in large-scale systems. *SIGMETRICS Perform. Eval. Rev.*, 38(1):13–24, 2010.

[10] Aayush Gupta, Youngjae Kim, and Bhuvan Urgaonkar. DFTL: A flash translation layer employing demand-based selective caching of page-level address mappings. In *Proceedings of the 14th International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS XIV, pages 229–240, New York, NY, USA, 2009.

[11] 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, New York, NY, USA, 2009.

[12] MathWorks Inc. Matlab R2013b documentation. http://www.mathworks.com/help/matlab/ref/ode45.html, 2013.

[13] Youngjae Kim, Brendan Tauras, Aayush Gupta, and Bhuvan Urgaonkar. FlashSim: A simulator for NAND Flash-based solid-state drives. In *Proceedings of the 2009 First International Conference on Advances in System Simulation*, SIMUL '09, pages 125–131, Washington, DC, USA, 2009.

[14] Jongsung Lee and Jin-Soo Kim. An empirical study of hot/cold data separation policies in solid state drives (SSDs). In *Proceedings of the 6th International Systems and Storage Conference*, SYSTOR'13, pages 12:1–12:6, New York, NY, USA, 2013.

[15] Sang-Won Lee, Dong-Joo Park, Tae-Sun Chung, Dong-Ho Lee, Sangwon Park, and Ha-Joo Song. A log buffer-based flash translation layer using fully-associative sector translation. *ACM Trans. Embed. Comput. Syst.*, 6(3), 2007.

[16] 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, New York, NY, USA, 2013.

[17] Richard McDougall. http://sourceforge.net/apps/mediawiki/filebench/index.php?title=Pre-defined_personalities.

[18] Richard McDougall. Filebench: application level file system benchmark. http://sourceforge.net/apps/mediawiki/filebench/index.php?title=Filebenchss.

[19] Dushyanth Narayanan, Austin Donnelly, and Antony Rowstron. Write off-loading: Practical power management for enterprise storage. *Trans. Storage*, 4(3):10:1–10:23, 2008.

[20] Manfred Opper and David Saad. *Advanced Mean Field Methods:Theory and Practice*. MIT Press, 2001.

[21] Chanik Park, Wonmoon Cheon, Jeonguk Kang, Kangho Roh, Wonhee Cho, and Jin-Soo Kim. A reconfigurable FTL (flash translation layer) architecture for NAND flash-based applications. *ACM Trans. Embed. Comput. Syst.*, 7(4):38:1–38:23, 2008.

[22] 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, Washington, DC, USA, 2011.

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

[24] 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, New York, NY, USA, 2013.

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

[26] Piet Van Mieghem, Jasmina Omic, and Robert Kooij. Virus spread in networks. *IEEE/ACM Trans. Netw.*, 17(1):1–14, 2009.