Analytic Models of SSD Write Performance

PETER DESNOYERS, Northeastern University

Solid state drives (SSDs) update data by writing a new copy, rather than overwriting old data, causing prior copies of the same data to be *invalidated*. These writes are performed in units of *pages*, while space is reclaimed in units of multi-page *erase blocks*, necessitating copying of any remaining valid pages in the block before reclamation. The efficiency of this cleaning process greatly affects performance under random workloads; in particular, in SSDs the write bottleneck is typically internal media throughput, and *write amplification* due to additional internal copying directly reduces application throughput.

We present the first nearly-exact closed-form solution for write amplification under greedy cleaning for uniformly distributed random traffic, validate its accuracy via simulation, and show that its inaccuracies are negligible for reasonable block sizes and over-provisioning ratios. In addition we also present the first models which predict performance degradation for both LRW (least-recently-written) cleaning and greedy cleaning under simple non-uniform traffic conditions; simulation results show the first model to be exact and the second to be accurate within 2%. We extend the LRW model to arbitrary combinations of random traffic, and demonstrate its use in predicting cleaning performance for real-world workloads.

Using these analytic models, we examine the strategy of separating "hot" and "cold" data, showing that for our traffic model such separation eliminates any loss in performance due to non-uniform traffic. We then show how a system which segregates hot and cold data into different block pools may shift free space between these pools in order to achieve improved performance, and how numeric methods may be used with our model to find the optimum operating point, which approaches a write amplification of 1.0 for increasingly skewed traffic. We examine on-line methods for achieving this optimal operating point, and show a control strategy based on our model which achieves high performance for a number of real-world block traces.

Categories and Subject Descriptors: B.3.3 [**Memory Structures**]: Performance Analysis and Design Aids—*formal models, simulation*; D.4.2 [**Storage Management**]: Garbage collection

General Terms: Design, Performance, Algorithms

Additional Key Words and Phrases: Solid State Drives, Solid State Storage Systems, Write Amplification, Flash Memory

ACM Reference Format:
Desnoyers, P. 2012. Analytic Models of SSD Write Performance. ACM Trans. Storage 0, 0, Article 0 (January 2012), 25 pages.
DOI = 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

A subset of this work was previously presented at the 2012 Annual International Systems and Storage Conference (SYSTOR 2012), Haifa Israel, June 2012.
The following additions and changes have been made to the SYSTOR 2012 version of this paper:

(1) A linear approximation to LRW cleaning performance is described.
(2) An approximate analytic result for LRW cleaning performance with hot/cold traffic is described, providing improved insight into the effect of hot area size and relative write rate.
(3) The discussion of related work is enlarged and improved.

This work has been supported by the National Science Foundation under grant CNS-1149232, and by a Faculty Award from IBM Corporation.
Author's address: P. Desnoyers, College of Computer and Information Science, Northeastern University.

DOI 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

1. INTRODUCTION

Log-structured file systems (LFSs) and SSD Flash Translation Layers (FTLs) both rely on a *cleaning* mechanism to consolidate free space and make it available for subsequent modification. In each case modifications are made in fixed-sized units (flash *pages* or file system *blocks*), and these writes are made in an *out-of-place* fashion—i.e. rather than modifying existing data, the previous data is *invalidated* and a new copy written. Storage is arranged in larger fixed-size units (flash *erase blocks* or LFS *segments*) which are written sequentially, destroying (either by over-writing or prior erasure) any previous data in the block. When all blocks on the media have been filled, it is necessary to select and *clean* blocks which may be used for further writes, by copying any remaining valid data so that it is not lost. The increase in internal writes caused by this cleaning process is referred to as *write amplification*, A, measured as the ratio of total internal to external writes.

Cleaning mechanisms have been extensively studied in the context of log-structured file systems [Blackwell et al. 1995; Wang and Hu 2002; Matthews et al. 1997] and flash translation layers [Lee et al. 2007; Chung et al. 2009; Gal and Toledo 2005], but to date almost all work has focused on experimental validation of new algorithms via trace-driven simulation, and few analytic results have been published.

We examine write amplification in the context of fully page-mapped FTLs, which are known to have optimal performance under limited conditions [Hu and Haas 2010], examining two cleaning policies in depth: Least-recently-written (LRW) and Greedy. We begin by describing an analytic model due to Robinson [Robinson 1996], with a closed-form solution due to Xiang [Xiang and Kurkoski 2012], which describes LRW cleaning under uniformly distributed random writes. Based on this approach, we provide a tractable near-exact approximation for Greedy cleaning, extending prior work by Bux et al. [Bux and Iliadis 2010] by providing a closed-form solution which is highly accurate under typical conditions.

We extend these models to cover the real-world case of non-uniformly distributed writes, which has not been addressed in the literature to date, and present analytic models with simple numerical solutions which accurately predict degradation of cleaning performance in the presence of simple non-uniform traffic. This approach is then applied to more complex traffic models, and shown to predict performance of simple cleaning algorithms for real-world traces.

Finally, we use our analytic results to provide insight into the well-known technique of separating "hot" and "cold" data. We show that simple separation of hot and cold data is insufficient to achieve performance gains possible in the presence of locality, and demonstrate how cleaning strategies informed by our models can achieve improvements of 3x or more in the presence of locality, vs. a 2x degradation for naïve cleaning. We construct a cleaning strategy which we argue may be optimal for a simple hot/cold traffic model, and present trace-driven simulation results demonstrating its applicability to real-world workloads.

2. PROBLEM STATEMENT

We consider the case where $U \cdot N_p$ pages of data are stored on $T \cdot N_p$ pages of media ($T > U$), in blocks of N_p pages each; this would correspond, for instance, to a flash translation layer with U blocks of user data stored across a total of T physical blocks, with an erase block size of N_p pages. We express the amount of *over-provisioning* as either the *spare factor* $S_f = \frac{T-U}{T}$, or equivalently as the *over-provisioning factor* $\alpha = \frac{T}{U}$, the ratio by which total storage exceeds user-visible data.

We consider only what are termed *fully page-mapped* flash translation layers, which place no restriction on the physical location of any data page; this is in contrast to

Table I. Variables used in this paper

N_p	block size (pages)
T, U	total and user-visible blocks
α	over-provisioning ratio
S_f	spare factor ($= \frac{\alpha-1}{\alpha}$)
r, f	hot/cold model parameters
$A_{lru}(\alpha)$	LRU write amplification
$A_{greedy}(\alpha)$	Greedy write amplification

fully or partially *block-mapped* algorithms, where placement restrictions allow the use of smaller in-RAM tables. We may classify the physical blocks in the system into three groups: *live data blocks*, which have been fully written and not yet selected for cleaning, the *free list* of blocks which have been cleaned and not yet written, and a single block designated as the *write frontier*. Pages are written sequentially into the write frontier until it is full, after which it joins the other live data blocks, and a new frontier is selected from the free list. When the free list drops below a low watermark $w \ll T$, blocks are selected and cleaned until the free list has reached w again. This is done by (1) selecting a block to clean, (2) copying any remaining valid pages within the block to the write frontier, and then (3) adding the block to the free list. If the block selected for cleaning contains N_v valid pages and $N_i = N_p - N_v$ invalid pages, the total gain in free pages will be N_i; the process will need to be repeated until N_p invalid pages have been reclaimed, increasing the length of the free list by one block.

For purposes of analysis we will ignore the write frontier and free list; a real system with low watermark w (and thus free list length w) and total blocks T may thus be modeled by an equivalent system with $T' = T - w - 1$.

In this work we are concerned with cleaning efficiency, which we express as *write amplification*, the ratio of total page writes (including copies due to cleaning) to external page writes. In doing so we ignore the following factors:

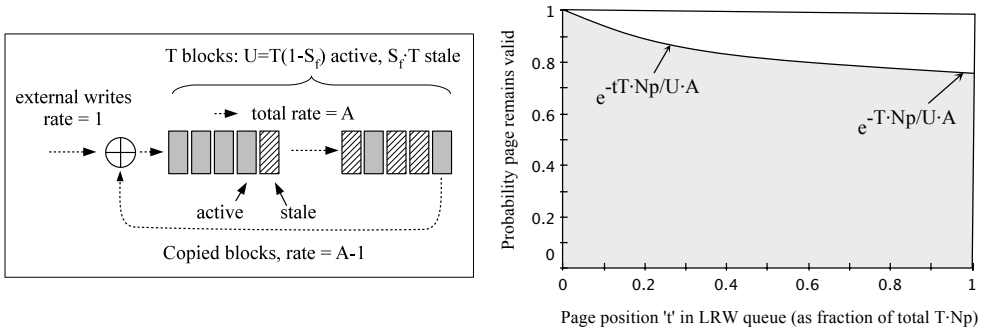
— Erasures. Although expensive, the ratio of erasures to writes is constant in page-mapped FTLs, and thus represents an adjustment of the write cost by a constant factor.
— Reads. Internal copies incur a page read, while external writes do not. Given specific values for read and write times, and a calculated write amplification factor A, the overall impact on performance will be $(1 + \frac{t_r}{t_w})A$.
— Implementation. Some implementations may perform additional writes to e.g. store mapping information, and some may be able to exploit internal parallelism.
— Wear leveling. *Static wear-leveling* algorithms (e.g. [Ban 2004]) periodically select non-optimal blocks to clean; this typically results in a small increase in write amplification, with the amount of degradation depending on the wear-leveling algorithm used.

3. UNIFORM TRAFFIC CASE

We first examine the case where traffic is uniformly distributed—i.e. an individual write is equally likely to invalidate any one of the N_pU pages of valid data.

3.1. LRW Cleaning

The simplest policy is to select blocks for cleaning in the same order as they are written, selecting the least-recently-written (LRW) block first. A diagram of this process may be seen in Figure 1(a); after a block is fully written it enters the *LRW queue*, and upon reaching the end it is selected for cleaning, with any remaining valid pages copied back to the current write frontier. (For purposes of analysis we assume that remaining



(a) Block model (b) Valid page probability

Fig. 1. Model of LRW block cleaning. (a) External writes enter at rate 1; blocks move through the queue at rate A, and are invalidated at a rate of $\frac{1}{U}$. (b) Probability that a page remains valid as it moves through the queue. In (a), note that at end of queue the expected valid fraction is $\left(1-\frac{1}{U}\right)^{T/A}$, which must also be equal to $\frac{A-1}{A}$.

valid pages are always copied, even in the case where there are no invalid pages in the block.)

The behavior of this system was first described mathematically by Robinson [Robinson 1996], and later by Menon and Stockmeyer [Menon and Stockmeyer 1998], both of whom independently derive Equation 3 as an approximation to the behavior of Greedy cleaning. The closed-form solution in Equation 5 was first reported by Xiang and Kurkoski [Xiang and Kurkoski 2012].

We assume that external page writes arrive at a constant rate of 1 page write per unit time. Taking write amplification into account, the overall rate at which pages are written to the write frontier is A, the write amplification factor. The queue thus moves at a rate of A pages per unit time; since its length is $T \cdot N_p$ (ignoring the write frontier, on the assumption that $T \gg 1$) it will take $\frac{TN_p}{A}$ units of time for a page to move through the queue. During this time, the chance of a page being invalidated by each external write is $\frac{1}{UN_p}$, and the probability that a page will remain valid after travelling through the queue is $(1-\frac{1}{UN_p})^{\frac{TN_p}{A}}$, as shown in Figure 1(b).

When a block containing v valid pages is selected for cleaning, v internal writes are generated, while $N_p - v$ external writes may be performed before another block must be cleaned; the mean write amplification is thus:

$$A = \frac{N_p}{N_p - E(v)} \tag{1}$$

At cleaning time the expected number of valid pages in a block is $(1-\frac{1}{UN_p})^{\frac{TN_p}{A}} N_p$, giving:

$$\frac{1}{1-(1-\frac{1}{UN_p})^{\frac{TN_p}{A}}} = A \tag{2}$$

Since $T = \alpha U$, the term $(1-\frac{1}{UN_p})^{\frac{TN_p}{A}}$ in the denominator is equal to $(1-\frac{1}{UN_p})^{\frac{\alpha}{A}UN_p}$, which approaches $e^{\frac{\alpha}{A}}$ as $U \cdot N_p \to \infty$. Letting $t = \frac{\alpha}{A}$, this gives the following equation:

$$\frac{1}{1-e^{-t}} = \frac{\alpha}{t} \tag{3}$$

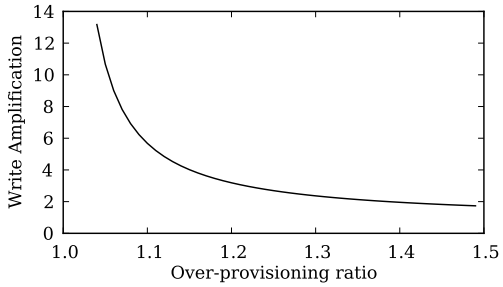


Fig. 2. Write amplification for LRW cleaning as a function of over-provisioning ratio α.

Table II. LRW Cleaning—Simulation vs. Eq. 5

S_f	Eq. 5	simulation	95% CI
0.03	16.837	16.835	$\pm$0.0036
0.07	7.318	7.317	$\pm$0.0020
0.11	4.725	4.725	$\pm$0.0013
0.17	3.129	3.129	$\pm$0.0008
0.23	2.371	2.371	$\pm$0.0008

Simulated results are for $U = 10^6$.

The closed-form solution to Equation 3 may be found in terms of Lambert's W function, $W(z)e^{W(z)} = z$:

$$t = \alpha + W(-\alpha e^{-\alpha}) \tag{4}$$

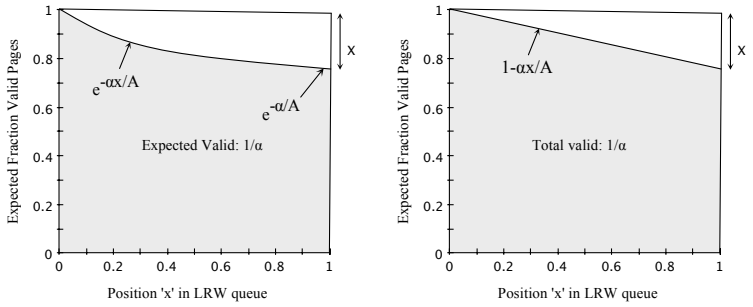
and

$$A_{LRW}(\alpha) = A = \frac{\alpha}{t} = \frac{\alpha}{\alpha + W(-\alpha e^{-\alpha})} \tag{5}$$

which may be seen in Figure 2.

In Table II simulation results for LRW cleaning, using 10^6 pages, are compared to analytic results from Equations 4 and 5; essentially exact correspondence is seen. We note that Equations 4 and 5 are independent of the block size N_p. This independence has been validated in simulation, and may be shown intuitively (for the case where $U, T \gg 1$) by comparing a system with block size N_p to an equivalent one with $N_p' = 1$ and $(T', U') = (T, U) \cdot N_p$. If we run the two systems in parallel, on the same sequence of operations, we see that they run in lockstep with each other in every case except that where a page in the N_p system is copied for cleaning, but is invalidated less than N_p steps later in the $N_p' = 1$ system. The rate at which such events will occur is less than $\frac{1}{U}$, or negligible by our assumptions.

For validating this and other models in this paper we have constructed a high-speed simulator, available in open source from `www.ccs.neu.edu/~pjd/ftlsim`, consisting of approximately 1000 lines of C and Python code. This simulator ignores page contents, but is designed to be highly configurable and to enable detailed instrumentation of behavior such as block occupancy. In the simplest case (LRW cleaning, write amplification statistics only) we are able to simulate the equivalent of a 256 GB SSD at a rate of 3 million simulated writes per second, while for greedy cleaning a 128 GB SSD may be simulated at 0.5 million simulated writes per second.



(a) Exact model (b) Approximate model

Fig. 3. Exact and approximate graphs of expected number of valid pages vs. position in LRW queue.

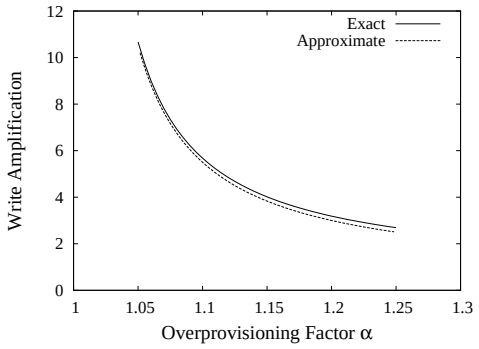


Fig. 4. Simple linear approximation for LRW performance (lower line) compared to exact solution.

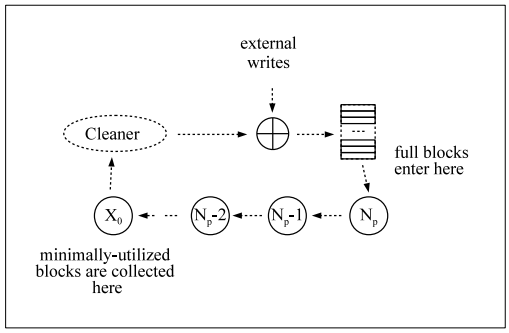


Fig. 5. Markov model for a single block with greedy cleaning: block state is the number of remaining valid pages.

3.2. Linear Approximation

An approximate formula for write amplification with LRW cleaning can be derived by assuming $e^{-\lambda} = 1 - \lambda$ for small values of λ; in Figure 3 we see a representation of the exact model (a) compared to this approximate model (b). In each figure the grey area represents (expected) valid pages vs. position in the LFW queue, which is represented on the X axis. Since the graph area is 1, the valid and invalid areas are thus of area $\frac{1}{\alpha}$ and $\frac{\alpha-1}{\alpha}$ respectively. The write amplification may be determined from this graph, as the interval X corresponds to the fraction of free space in blocks chosen for cleaning, and by Equation 1 the write amplification is $A = \frac{1}{X}$.

In the approximate model we do not know the slope of the line between white and grey areas, as it depends on the write amplification; however given the area of the white triangle, a simple geometric argument gives us X:

$$X = \frac{2(\alpha - 1)}{\alpha} \tag{6}$$

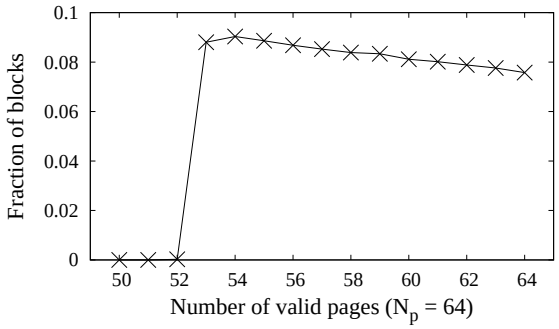
and thus our approximated value is:

$$A = \frac{1}{X} = \frac{\alpha}{2(\alpha - 1)} \qquad \left(= \frac{1}{2S_f}\right) \tag{7}$$

In other words, write amplification is approximately linear in the inverse of the spare factor S_f, tending to infinity as $S_f \to 0$. Compared to the exact solution in Figure

Valid pages	Fraction
50	0.0000
51	0.0022
52	0.9535
53	0.0443
54	0
55	0

(a) Valid page distribution, blocks selected for cleaning.



(b) Valid page distribution, all blocks.

Fig. 6. Greedy cleaning: distribution of block states (i.e. number of valid pages) overall and when selected for cleaning. Simulation parameters: $S_f = 0.089$, $N_p = 64$, $U = 5 \times 10^4$.

4 the approximation has an absolute error of about 0.2, becoming significant for larger S_f (and thus smaller A): e.g. for $S_f = 0.5$ the approximation gives 1.0, while the exact value is 1.225. We note that Equation 7 is identical to that given by Agarwal and Morrow [Agarwal and Marrow 2010]; however their result is derived by approximating (as uniform) the distribution of block utilizations in Greedy cleaning.

3.3. Greedy Cleaning

In contrast to LRW, which selects the oldest block for cleaning, the Greedy algorithm chooses the block with the most invalid pages. This choice results in the fewest valid pages to be copied in the cleaning process; for uniformly-distributed random traffic it is believed to be optimal [Hu and Haas 2010]. Unlike LRW, Greedy cleaning performance is affected by block size; in the limit of $N_p = 1$, for instance, no write amplification will occur, as we may always select a block containing no valid pages.

We begin by presenting a Markov model, due to Bux and Iliadis [Bux and Iliadis 2010], of the states through which an individual block passes between being written and selected for cleaning. We present Bux and Iliadis' balance equations for this system, which they use numerically to find exact solutions for Greedy cleaning performance; we then derive an approximate but highly accurate closed-form solution for this model.

We see the model in Figure 5: the states of a block correspond to the number of valid pages, starting at N_p and decreasing as pages are invalidated by external writes. Again assuming external writes arrive at a rate of one page per unit time, blocks enter state N_p at rate $\frac{A}{N_p}$. Since the chance of a single write invalidating a particular page is $\frac{1}{UN_p}$, the transition rate from state i (with i valid pages) to $i-1$ is $\frac{i}{UN_p}$ transitions per unit time.

Bux and Iliadis have shown that in the equilibrium state, the number of valid pages in blocks selected for cleaning will take on either a single value, or be split between two consecutive values. (this may be seen empirically in simulation results, shown in Figures 6 and 7) For the purpose of analysis we begin by assuming that all blocks are selected for cleaning in a single state $X_0 - 1$, where X_0 is the lowest-numbered state with non-negligible occupancy, and postpone examination of the bimodally-distributed case.

Based on this assumption, only states $X_0 \cdots N_p$ will be occupied, and we obtain the equilibrium state as follows. First, transitions may only occur from state i to state

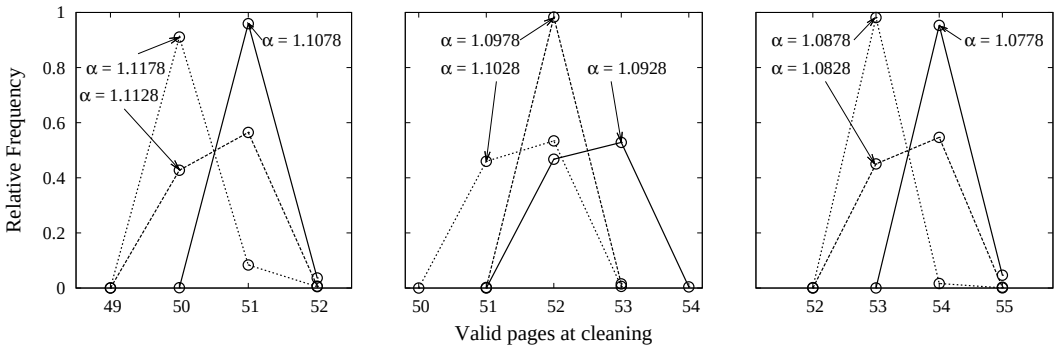


Fig. 7. Block states (i.e. number of valid pages) when selected for cleaning, for values of α ranging from 1.0778 to 1.1178. $N_p = 64$, $U = 5 \times 10^5$. Note that depending on parameters, block occupancy at cleaning takes on either a single value or one of two adjacent values.

$i - 1$, with rate proportional to the number of valid pages i. Letting f_i be the fraction of blocks in state i, and k be the proportionality constant, we have:

$$i \cdot f_i = k \tag{8}$$

for some constant k. If we sum $i \cdot f_i$ across all blocks we have the mean number of valid pages per block, which is the global utilization fraction $(1 - S_f)$ times the number of pages in a block:

$$\sum_{i=X_0}^{N_p} i \cdot f_i = \sum_{i=X_0}^{N_p} k = \frac{N_p}{\alpha} \tag{9}$$

and thus

$$k = \frac{N_p}{\alpha \left(N_p - (X_0 - 1)\right)} \tag{10}$$

Next, the sum of all state probabilities is 1:

$$\sum_{i=X_0}^{N_p} f_i = \sum_{i=X_0}^{N_p} \frac{k}{i} = 1 \tag{11}$$

which can be expressed in terms of the Harmonic number function $H_n = \sum_{i=1}^{n} \frac{1}{i}$:

$$H_{N_p} - H_{X_0 - 1} = 1 \tag{12}$$

Using the approximation $H_n \approx ln\left(n + \frac{1}{2}\right) + \sigma$ [Qi and Guo 2009], we have:

$$k = \frac{1}{\log(2N_p + 1) - \log(2X_0 - 1)} \tag{13}$$

Finally, from Equations 10 and 13 we derive the equilibrium value of X_0:

$$X_0 = \frac{1}{2} - \frac{N_p}{\alpha} W\left(-(1 + \frac{1}{2N_p})\alpha e^{-(1+\frac{1}{2N_p})\alpha}\right) \tag{14}$$

and noting that reclaimed blocks have $X_0 - 1$ pages active out of N_p, we obtain the write amplification:

Table III. Greedy Cleaning—Simulation vs. Eq. 16

α	$A(\alpha)$	$A(simulated)$	95% CI
1.030	13.7055	13.8560	$\pm$0.0012
1.050	9.1918	9.1964	$\pm$0.0007
1.070	7.0014	7.0101	$\pm$0.0006
1.120	4.5286	4.5308	$\pm$0.0003
1.200	3.0529	3.0527	$\pm$ <0.0002

Simulation parameters: $N_p = 64$, $U = 300000$.

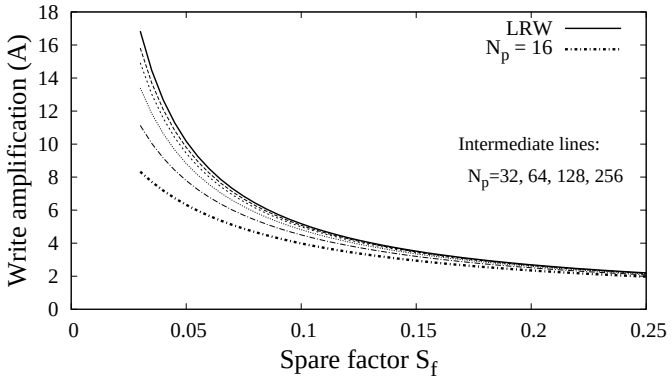


Fig. 8. Greedy cleaning performance for varying block sizes, converging to the performance of LRW cleaning for large values of N_p.

$$A = \frac{N_p}{N_p - (X_0 - 1)} \tag{15}$$

$$= \frac{1}{1 + \frac{1}{2N_p} + \frac{1}{\alpha} W\left(-\left(1 + \frac{1}{2N_p}\right)\alpha e^{-\alpha\left(1+\frac{1}{2N_p}\right)}\right)} \tag{16}$$

With some manipulation we can express this as:

$$A = \frac{A_{LRW}(\alpha')}{1 + \frac{1}{2N_p}} \tag{17}$$

where $\alpha' = (1 + \frac{1}{2N_p})\alpha$ and $A_{LRW}(\alpha)$ is the LRW cleaning throughput for a given over-provisioning ratio, from Equation 5.

In Table III we see simulation results compared to Equations 14 and 16 for $N_p = 64$; for realistic values of α (i.e. 1.07 or higher) relative error is seen to be 10^{-3} or less.

Unlike LRW, the efficiency of greedy cleaning is dependent on N_p, decreasing as the block size increases. In Figure 8 we see performance for values of N_p ranging from 16, the erase block size in early flash devices, to 256, the largest erase block size in production today. (typical values are 64 for SLC and 128 for MLC) As N_p increases, we see performance converge to that of LRW cleaning, with a more rapid convergence for higher values of S_f.

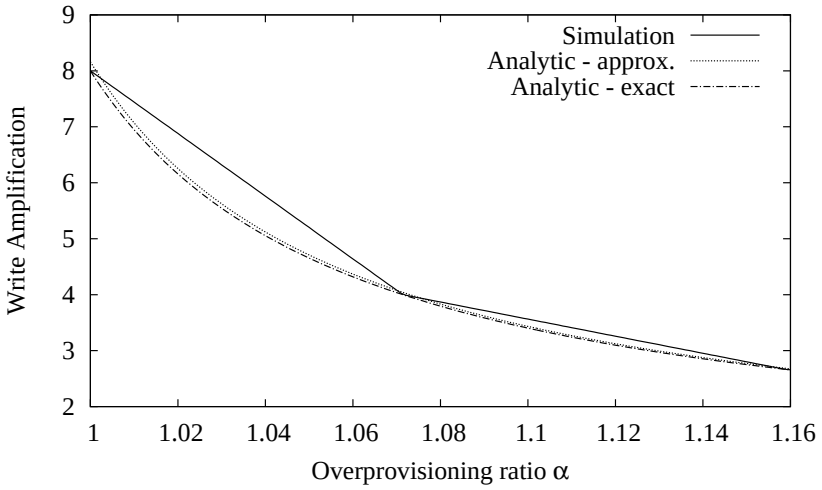


Fig. 9. Simulated and analytic performance for small values of N_p and α. Exact correspondence is seen between the lower curve (Equation 18) and simulation results for cases where X_0 is integral.

3.4. Limits of Approximation

In deriving the results in Equation 16 we make two approximations: we approximate H_n by $ln\left(n+\frac{1}{2}\right)+\sigma$, and in Equation 9 we assume that X_0 is integral.

The error in approximating the harmonic summation may be evaluated directly; it is small but noticeable for small values of N_p and α; e.g. for $N_p = 8, \alpha = 1.0 + \epsilon$ the resulting write amplification should be 8, while Equation 16 gives a value of 8.163. Precise answers may be obtained by using the digamma function ψ to precisely express $\sum_{i=x}^{y} \frac{1}{i}$, and then solving numerically. In particular, Equation 13 becomes:

$$k = \frac{1}{\psi(N_p+1) - \psi(X_0)} \tag{18}$$

We then solve numerically for a value of X_0 which satisfies Equations 18 and 10; this is shown as the lower curve in Figure 9.

The second and more significant approximation is in assuming that a non-integral value of X_0 satisfying the balance equations directly corresponds to the mean block occupancy at cleaning time. In Figure 9 we see simulated and analytic results for write amplification with 8-page blocks—i.e. $N_p = 8$, showing both precise (Equation 18) and approximate (Equation 16) analytic results. The simulated algorithm performance is seen to be piecewise linear, where each corner point corresponds to an integer solution for X_0; for these integer solutions the analytic result from Equation 18 is seen to be exact.

Bux and Iliadis [Bux and Iliadis 2010] show how given N_p the values of α corresponding to integral values of X_0 may be determined, and in addition show that behavior is piecewise linear between these integral solutions.

This may be seen by examining the region in Figure 9 between $\alpha = 1.00$ and $\alpha = 1.0741$, where A decreases from 8.0 to 4.0. In the limiting case where $\alpha = 1.0 + \epsilon$ there is a single unused page in a single block; that block will be selected and cleaned (incurring 7 writes) before each new write, for an overall write amplification of 8.0. As we increase the over-provisioning ratio between 1.0 and 1.0741 virtually all blocks will be in state 8 or state 7, as we are selecting blocks for cleaning faster than invalidations are arriving to blocks in state 7. Let f_7 be the fraction of blocks in state 7; this will be

Table IV. LRW vs. Windowed Greedy (simulated)

S_f	Greedy	LRW	Windowed Greedy	95% CI
0.040	10.6283	12.6712	12.469	±0.0042
0.060	7.5465	8.5070	8.396	±0.0027
0.080	5.8696	6.4261	6.356	±0.0027
0.110	4.4235	4.7254	4.682	±0.0017
0.140	3.5662	3.7554	3.727	±0.0019

parameters: U=50000, N_p=64, w=500. $S_f = \frac{\alpha-1}{\alpha}$

directly proportional to the over-provisioned space $\alpha - 1$, and the rate at which writes arrive to blocks in state 7 (resulting in near-immediate cleaning in state 6) will be proportional to f_7.

3.5. Windowed Greedy

Prior work by Hu [Hu et al. 2009] has examined the *Windowed Greedy* cleaning algorithm, a hybrid of LRW and Greedy where the search for a block with minimal valid pages is restricted to a *greedy window* of the w oldest blocks in the LRW queue, in order to reduce computational cost.

If we assume that w is sufficiently small to allow linear search (i.e. $w = O(1)$) then we argue that Windowed Greedy offers little improvement over LRW cleaning. We again compare two systems of identical size, in this case one using strict LRW cleaning and the other Windowed Greedy with a window size of w. We note that although the two systems may choose blocks in different order for cleaning, differences in performance will arise only if some block B has a different number of valid pages when selected for cleaning in one system than in the other. For this to happen, a write must occur to block B while it is one of the w least-recently-used blocks; since by assumption $w = O(1)$, the rate at which such writes occur is very small.

This may be seen in Table IV, where analytic values for Greedy and LRW cleaning are compared to simulated results for Windowed Greedy with $U = 50000$, $N_p = 64$ (the lowest value seen in current devices), and a window size $w = 500$. While Greedy cleaning offers relative improvements in write amplification of 5% to 16% over LRW, improvements due to Windowed Greedy reach at most 1.6%.

Larger values of w would improve this performance, but would result in unacceptable computational complexity if simple linear search were used. Sub-linear algorithms for greedy selection are known, such as the $O(1)$ algorithm described in [Lin and Chang 2012] and used in our simulator; however this algorithm has similar overhead for implementing Windowed Greedy (with $w = O(T)$) and full Greedy cleaning.

4. NON-UNIFORM TRAFFIC CASE

Unlike the uniformly distributed workload assumed above, real storage workloads typically display significant amounts of both temporal and spatial locality. As an illustration, in Table V we see statistics on write operations in several of the widely used Microsoft Research storage traces [Narayanan et al. 2008]; in many cases 50% or more of the writes are destined to one percent or fewer of the storage. In this section we present the first analytic examination of the effect of non-uniform write distribution on cleaning performance, deriving an accurate analytic model for LRW cleaning and an approximate model for Greedy.

Temporal locality refers to the correlation between observance (or non-observance) of an address during an interval and the expected time until that address is next seen. We address the simplest form of such locality: the case where individual writes are still independent, but the distribution of writes over the address space is non-uniform.

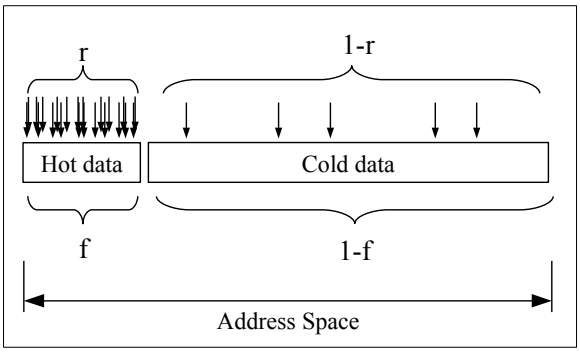


Fig. 10. Hot/Cold data model used for analysis. The fraction of the total write rate destined for "hot" pages is r; these hot pages represent a fraction f of the overall address space—e.g. 80% of writes destined to 20% of the address space.

Table V. Sample trace locality

		Fraction of LBA space			
		mds_0	prxy_0	rsrch_0	rsrch_2
Fraction of Traffic	70%	4.20%	5.60%	13.1%	59.2%
	50%	0.70%	0.30%	1.20%	32.0%
	20%	0.00%	0.12%	0.01%	0.30%
	10%	0.00%	0.06%	0.00%	0.00%

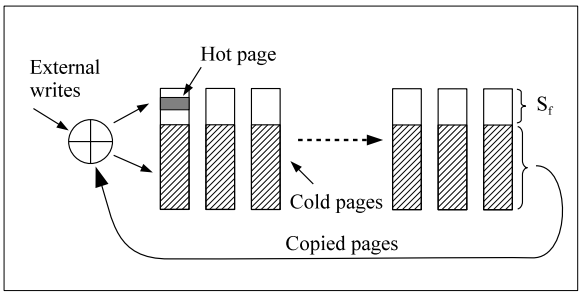


Fig. 11. Limiting behavior for hot/cold traffic with LRW cleaning: $f \to 0$ (i.e. one hot page) and $r \to 1$ (no writes to cold pages).

(Under this assumption a recently-seen address is more likely to be from a high-rate "hot" address, and thus have to a shorter expected time until its next occurrence.)

We use Rosenblum's hot/cold data model [Rosenblum 1992] as shown in Figure 10: some fraction f of the address space is "hot"; a fraction r of overall writes are distributed uniformly and randomly across these hot pages. Conversely, the remaining $1 - f$ of the address space is "cold", and fraction $1 - r$ of overall writes are uniformly distributed across these cold pages.

4.1. LRW Cleaning

LRW cleaning is known to suffer degradation in performance when there is significant temporal locality [Rosenblum and Ousterhout 1991; Wu and Zwaenepoel 1994]. We can gain some insight by considering the limiting case where a single page is repeatedly re-written—i.e. $f \simeq 0$ and $r = 1$, as seen in Figure 11. In this case we see that recycled blocks contain $N_p(1 - S_f)$ cold pages; this is in fact the population average, and thus no better than if blocks had been selected randomly. Because LRW cleaning forces all

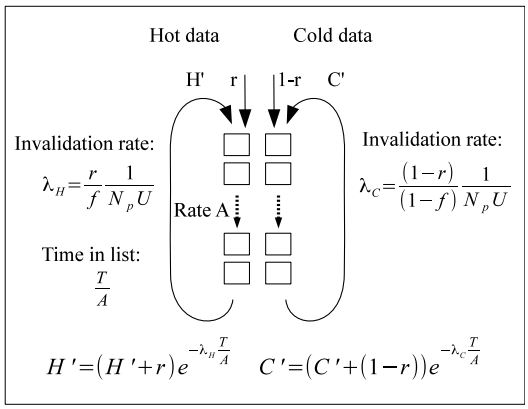


Fig. 12. Balance equations for LRW cleaning with hot/cold traffic. At each step (i.e. external write) the odds of a hot page being invalidated are $\frac{r}{fN_pU}$. Since the list moves on average A pages for each external write, during a block's passage through the list the number of external writes is $\frac{N_pT}{A}$.

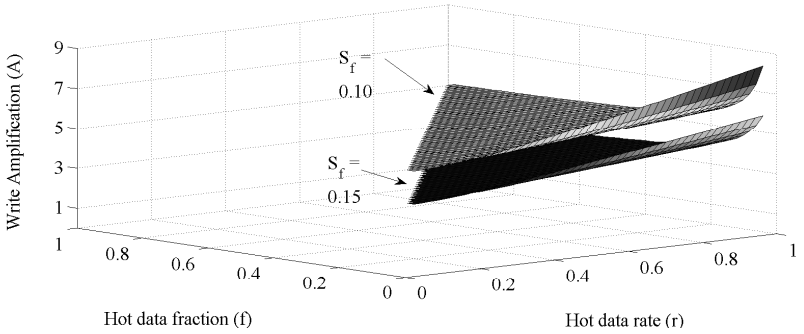


Fig. 13. Performance of LRW cleaning with mixed hot/cold traffic for different values of r and f, for $S_f = 0.1$ (upper graph) and $S_f = 0.15$ (lower one). As $(r, f) \to (1, 0)$ write amplification increases from 5.2 and 10 ($S_f = 0.1$) and from 3.5 to 6.66 ($S_f = 0.15$).

blocks to be cleaned at the same age, hot data must wait too long to be cleaned (in this case it is invalidated and thus ready to reclaim almost immediately) while cold data does not wait long enough–in this limiting case it should in fact never need cleaning.

Menon and Stockmeyer derive the balance equations for LRW cleaning with hot/cold traffic, depicted graphically In Figure 12. The rate at which hot and cold pages are copied during cleaning is H' and C' respectively, while new hot and cold pages arrive at rates r and $1-r$; the total rate at which pages enter the top of the queue is A, giving:

$$A = H' + r + (1 - r) + C' \tag{19}$$

Using $\alpha = \frac{T}{U}$ as before, and assuming without loss of generality that $N_p = 1$, we have:

$$A = 1 + \frac{r}{e^{\frac{r}{f}\frac{\alpha}{A}} - 1} + \frac{1 - r}{e^{\frac{(1-r)}{(1-f)}\frac{\alpha}{A}} - 1} \tag{20}$$

This equation is easily and efficiently solved numerically; in Table VI we see computed values compared to simulation results for several values of S_f, r, and f, with essentially exact correspondence.

Table VI. LRW Cleaning with mixed hot/cold traffic

S_f	r	f	A (computed)	A (simulated)	95% CI
0.03	0.90	0.05	19.064	19.065	$\pm$ 0.002
0.07	0.8	0.2	7.682	7.681	$\pm$ 0.001
0.07	0.90	0.05	9.240	9.240	$\pm$ 0.0007
0.11	0.8	0.2	5.083	5.083	$\pm$ 0.0008
0.11	0.90	0.05	6.409	6.409	$\pm$ 0.0005
0.20	0.8	0.2	3.035	3.034	$\pm$ 0.0006
0.20	0.90	0.05	3.973	3.972	$\pm$ 0.002

Simulation parameters: U = 3×10^6 pages.

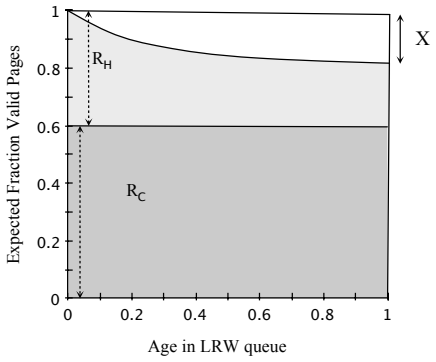


Fig. 14. Invalid pages vs. age in the LRW queue for hot/cold traffic with r=1 (i.e. no writes to cold pages).

To give a better picture of cleaning behavior as locality increases, in Figure 13 we see A plotted against values of r and f for $S_f = 0.1$ and $S_f = 0.15$ (i.e. $\alpha = 1.111$ and $\alpha = 1.177$). We only consider cases where the hot pages are in fact hot—i.e. $r \geq f$—and thus the values plotted begin at the diagonal $r = f$, or no locality, and approach the corner where $r = 1$ and $f = 0$. As we approach the corner of the graph performance degrades; the increase in write amplification is fairly linear in r, but is relatively unaffected for most values of f, rising sharply as f nears zero. Although only two values of S_f are shown here, behavior relative to the uniform case is similar for other values of S_f.

4.2. Approximate Analytic Results

To gain more insight into the behavior of LRW cleaning with non-uniform traffic we derive some analytic approximations. For insight into the case where r is near 1, we examine its value *at* $r = 1$—i.e. cold pages are never written—which may be solved exactly. This is seen in Figure 14: R_H is the total rate at which hot pages are written at the front of the queue (i.e. host writes plus copies due to cleaning) as a fraction of all internal writes, while R_C is the total rate (again as fraction of all internal writes) at which cold pages are written.

As seen in the figure, since cold pages are never invalidated, they constitute a constant fraction of total pages (valid + invalid) of any age; thus

$$R_C = \frac{(1-f)U}{T} = \frac{1-f}{\alpha} \tag{21}$$

and

$$R_H = 1 - R_C \tag{22}$$

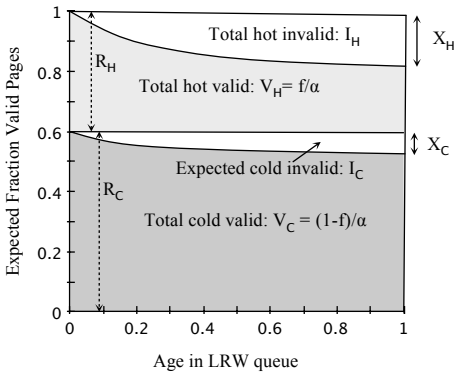


Fig. 15. Full Hot/Cold traffic model with LRW cleaning.

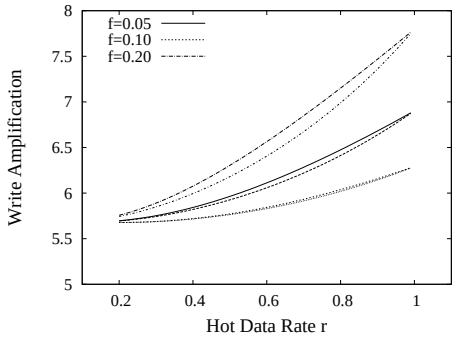


Fig. 16. Approximate vs. Exact (upper curve) results, $\alpha = 1.1$ (max relative error = 2.6%).

To calculate X in Figure 14 we note that the chance of a hot page being invalidated by an external write is $\frac{1}{fUN_p}$; X is the complement of its survival probability after $\frac{\alpha UN_p}{A}$ external writes:

$$X = R_H - R_H e^{-\frac{\alpha}{fA}} \tag{23}$$

giving the following result:

$$A = \frac{1}{X} = \frac{\alpha}{(\alpha - 1 + f) + fW\left(-\frac{\alpha-1+f}{f}e^{-\frac{\alpha-1+f}{f}}\right)} \tag{24}$$

Putting this in terms of the uniform LRW performance function, we have:

$$A = \frac{\alpha}{\alpha - (1-f)} A_{LRW}\left(\frac{\alpha-(1-f)}{f}\right) \tag{25}$$

Examining Figure 14, we see that $\frac{\alpha-(1-f)}{f}$ is the ratio of total space to valid pages within the hot (light gray and white) area, while $\frac{\alpha}{\alpha-(1-f)}$ is the ratio of the entire graph to that hot section. In other words, we can look at this as a system with only hot pages and free space (i.e. light gray and white areas), but for each hot page that we copy, we have to carry along an additional $\frac{\alpha}{\alpha-(1-f)} - 1$ unmodified cold pages. In addition, we note that the limit of this function as $f \to 0$ is $\frac{\alpha}{\alpha-1}$ or $\frac{1}{S_f}$.

We next examine the more general case where cold pages receive a non-zero fraction of writes, shown in Figure 15. We note that with LRW cleaning all pages—whether originally holding hot or cold data—spend the same amount of time in the system, while the rate at which invalid hot pages are generated is higher by a factor of $\frac{r}{1-r}$ than that at which invalid cold pages are generated. We thus assume that the ratio of total invalid hot to total invalid cold pages (white areas I_H and I_C in the figure) is the same:

$$I_H = \frac{r}{1-r} I_C \tag{26}$$

Since the two areas sum to $\frac{\alpha-1}{\alpha}$, we have:

$$I_H = \frac{r(\alpha-1)}{\alpha} \tag{27}$$

$$I_C = \frac{(1-r)(\alpha-1)}{\alpha} \tag{28}$$

$$R_H = \frac{f}{\alpha} + I_H \tag{29}$$

$$R_C = \frac{(1-f)}{\alpha} + I_C \tag{30}$$

Next we calculate α_h and α_c, the over-provisioning ratios in the hot and cold sections:

$$\alpha_h = \frac{f + r(\alpha-1)}{f} \tag{31}$$

$$\alpha_c = \frac{(1-f) + (1-r)(\alpha-1)}{1-f} \tag{32}$$

X_H and X_C may be calculated exactly via the uniform-traffic LRW performance function, as the same exponential relation between e.g. R_C, $R_C - X_C$, and α_c must hold as in the uniform case. This gives

$$X_h = \frac{R_H}{A_{LRW}(\alpha_h)} \tag{33}$$

$$X_c = \frac{R_C}{A_{LRW}(\alpha_c)} \tag{34}$$

and

$$A = \frac{1}{\frac{R_H}{A_{LRW}(\alpha_h)} + \frac{R_C}{A_{LRW}(\alpha_c)}} \tag{35}$$

In Figure 16 we see the approximation compared to exact numeric results for $\alpha = 1.1$ and several values of f; relative error is less than 3% in the worst case. The important intuition which may be gained from this approximation is the idea of "dividing" the total free space between the hot and cold regions in amounts proportional to the relative rate of overwrites experienced by the regions, and then assuming that the regions operate independently. Although clearly an over-simplification—since hot and cold pages are mixed within blocks, they cannot be independent—this simplified model provides fairly accurate predictions of the behavior of the real system.

4.3. Greedy Cleaning

We may approximate performance of greedy cleaning for hot/cold traffic by noting from Equation 17 that Greedy performance for uniform traffic may be expressed in terms of LRW performance:

$$A = \frac{A_{LRW}(\alpha')}{1 + \frac{1}{2N_p}} \tag{36}$$

where $\alpha' = (1 + \frac{1}{2N_p})\alpha$ and $A_{LRW}(\alpha)$ is the LRW write amplification function.

We approximate the performance of Greedy for hot/cold data using the same adjustment to $A_{LRW}(\alpha, r, f)$, the write amplification for LRW cleaning from Equation 20:

$$A_{Greedy}(\alpha, r, f) \approx \frac{A_{LRW}((1 + \frac{1}{2N_p})\alpha, r, f)}{1 + \frac{1}{2N_p}} \tag{37}$$

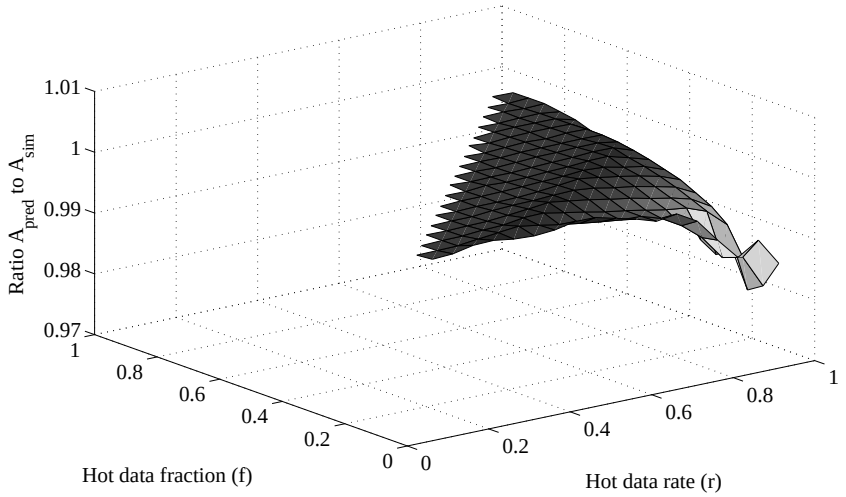


Fig. 17. Ratio of predicted Greedy performance A_{pred} (Eq. 37) to simulated performance A_{sim} for $S_f = 0.10$, $N_p = 64$ across the range of values for r and f. For highly skewed data, relative error remains less than 2%.

Table VII. Greedy cleaning for hot/cold traffic

S_f	N_p	r	f	Eq. 37	A (sim.)	rel. error
0.03	32	0.9	0.05	13.199	13.433	1.77%
0.07	64	0.9	0.05	8.461	8.608	1.74%
0.07	128	0.8	0.20	7.302	7.325	0.31%
0.11	64	0.9	0.05	6.058	6.112	0.89%
0.11	32	0.8	0.20	4.509	4.537	0.62%
0.20	64	0.9	0.05	3.845	3.826	-0.49%
0.20	128	0.8	0.20	2.984	2.992	0.27%

All simulation 95% confidence intervals are < 0.0025.

In Figure 17 we see Equation 37 compared with simulation results for a single configuration, with $S_f = 0.10$ and $N_p = 64$. In Table VII we see a sampling of cases for different values of S_f and N_p; again agreement is close but not exact, with error increasing slightly for small values of N_p and high degrees of locality.

4.4. Real-World Traffic

Equation 20 may be generalized for the case of k traffic classes, each with rate $r_i 1 \leq i \leq k$ and occupying fraction f_i of the LBA space, where $\sum r_i = \sum f_i = 1$:

$$R_i = (R_i + r_i)\, e^{-\frac{r_i}{f_i}\frac{\alpha}{A}} \tag{38}$$

$$= \frac{e^{-\frac{r_i}{f_i}\frac{\alpha}{A}}}{1 - e^{-\frac{r_i}{f_i}\frac{\alpha}{A}}} r_i \tag{39}$$

and

$$1 + \sum R_i = A \tag{40}$$

In applying this model to real-world traffic, we first note that the page-mapped LRW and greedy algorithms analyzed in this work are unaffected by strict spatial locality—unlike e.g. hybrid block-mapped FTLs [Lee et al. 2007], logical addresses are not examined other than to test for equality. In other words, given a workload

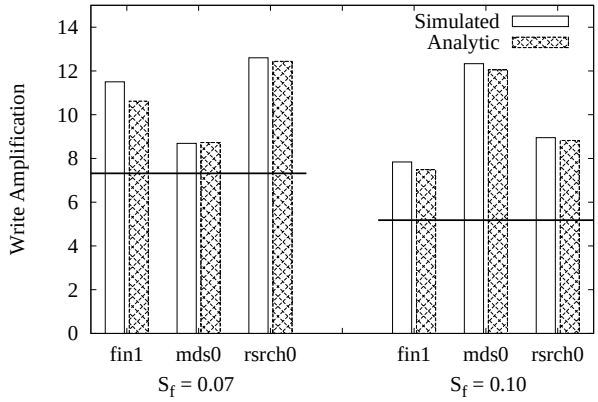


Fig. 18. Predicted and simulated performance for selected traces with LRW cleaning and spare factors of 0.07 and 0.10. Horizontal lines show uniform traffic write amplification for comparison.

trace $(lba_1, lba_2, \ldots)$, if we shuffle the addresses by e.g. applying a mapping f to obtain $(f(lba_1), f(lba_2) \ldots)$, the two traces will perform identically regardless of whether f preserves locality.

In addition, LRW cleaning appears to be fairly insensitive to temporal correlation: for instance, randomizing the order of writes in the traces examined causes in only small changes in LRW write amplification, indicating that results are primarily determined by the frequency of arrivals for each address. We may therefore estimate cleaning performance by measuring workload statistics and using Equations 39 and 40.

In Figure 18 we see predicted and simulated performance for LRW cleaning and three block traces: the `fin1` OLTP trace from the UMass Trace Repository [Bates and McNutt 2007] and the `mds0` and `rsrch0` traces from Microsoft Research [Narayanan et al. 2008]. In each case per-page access frequencies were grouped by quintile (i.e. those addresses responsible for the top 20% of accesses, then the next 20%, etc.) and used to construct a 5-part model for Equations 39 and 40. Prediction accuracy is seen to be quite close in almost all cases.

5. HOT/COLD DATA SEPARATION

It is well known that separating hot and cold data can increase flash translation layer performance [Lee et al. 2009; Hsieh et al. 2006; Chang and Kuo 2002]. To examine this, we begin by separating the problem of *identifying* hot data from *handling* it, and assume that the FTL has perfect knowledge of which data pages are hot and which are cold. Simulation results have shown that for all tested values of S_f, r, and f, with our simple hot/cold traffic model, separating hot and cold data into separate blocks and using global greedy cleaning results in performance that is statistically identical to that for uniformly distributed traffic. In effect, the hot and cold blocks form essentially two independent flash translation layers, as hot pages are written to blocks going into one pool, and cold pages go into a separate pool. Due to greedy cleaning, hot and cold blocks will be selected for cleaning with the same number of invalid pages, thus incurring the same write amplification. Since write amplification is a monotonic function of S_f, they will each have the same spare fraction, which must then be the same as the total spare fraction $\frac{T-U}{T}$, resulting in performance equal to that of the uniformly-distributed traffic case.

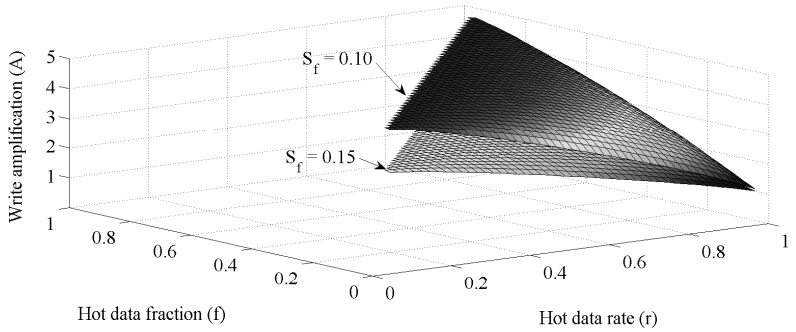


Fig. 19. Write amplification with hot/cold separation and optimal division of free space, greedy garbage collection, $N_p = 64$.

Table VIII. Optimized Greedy Cleaning

S_f	N_p	r	f	A (computed)	A (simulated)
0.07	64	0.90	0.05	2.325	2.335
0.07	128	0.80	0.20	4.693	4.823
0.11	32	0.80	0.20	2.919	2.991
0.11	64	0.90	0.05	1.760	1.762
0.20	64	0.90	0.05	1.311	1.312
0.20	128	0.80	0.20	1.966	2.008

We can do better, however, by reserving a higher share of free space for the hot traffic, decreasing write amplification for hot blocks while increasing it for cold blocks. Since there are more writes to hot pages, the reduction in write amplification for these pages will more than compensate for any increase in amplification suffered by cold writes. Again the limiting case of $f \to 0$ and $r \to 1$ is illustrative; given several blocks reserved for hot traffic, the single hot page may be re-written repeatedly with no additional copying of either hot or cold data.

We can frame this as a simple optimization problem: if $A(\alpha)$ is the write amplification for uniform traffic with over-provisioning factor α, then we want to minimize:

$$A_{total} = r \cdot A(\alpha_h) + (1 - r) \cdot A(\alpha_c) \tag{41}$$

where α_h and α_c are the over-provisioning factors for hot and cold data resulting from some unequal division of free space. In particular, assume that of the entire free space $T - U$ we assign some fraction p to the hot traffic, giving

$$\alpha_h = \frac{p(\alpha - 1) + f}{f} \tag{42}$$

$$\alpha_c = \frac{(1 - p)(\alpha - 1) + (1 - f)}{(1 - f)} \tag{43}$$

Using the performance function for greedy cleaning from Equation 16 and optimizing numerically, we obtain results shown in Figure 19. As hot/cold disparity increases, write amplification decreases towards 1. For example, for $S_f = 0.1$ and $N_p = 64$, with uniform arrivals $A = 4.82$; if 90% of the traffic is restricted to 5% of the address space, naïve greedy cleaning gives $A = 6.56$ while hot/cold separation with an optimal division of free space yields $A = 1.86$. Given a desired assignment of free space between hot and cold blocks, we may construct a cleaner which greedily chooses either a cold block or a hot block so as to maintain the specified free space division. We have simulated this,

using knowledge of the synthetic workload to divide hot and cold pages perfectly, and specifying the free space assignment determined by our optimization; sample results are shown in Table VIII and shown to correspond to our analytic results.

In each case the optimal assignment results in a lower write amplification for hot data than for cold, meaning that cold blocks selected for cleaning will have few free pages than hot blocks selected. (In the case above where $S_f = 0.1$ and $N_p = 64$, hot blocks will have about 14 valid pages when cleaned, while cold blocks will be selected when 55 blocks are still valid and only 9 are free.)

5.1. Optimized Online Hot/Cold Cleaning

We can do better, however, by examining the marginal effect on write amplification of removing a block from either the hot or cold pool. If we let $\omega = W\left(-\alpha\left(1+\frac{1}{2N_p}\right)e^{\alpha\left(1+\frac{1}{2N_p}\right)}\right)$ for brevity, the derivative of the greedy performance function is:

$$dA = \frac{2\omega N_p}{(1+\omega)(\alpha+2(\alpha+\omega)N_p)}d\alpha \tag{44}$$

If we let L be the number of blocks in a pool (hot or cold) and i be the number of free (i.e. invalid) pages in that pool, then the number of valid pages v is $L \cdot N_p - i$; and assuming v is fixed we have:

$$\begin{aligned} \alpha &= \tfrac{LN_p}{v} \\ d\alpha &= \tfrac{N_p}{v}dL \end{aligned} \tag{45}$$

Each time a block is to be selected for cleaning, we may then select it from the hot or cold pool according to which is less: $r\alpha_h d\alpha_h$ or $(1-r)\alpha_c d\alpha_c$. Given perfect knowledge of hot vs. cold pages, it is clear that this will result in the optimal division of free space that minimizes Equation 41; we show below that even with probabilistic identification of access frequency it can converge in many cases to near-optimal performance.

Matlab code for numerical results presented above is available from `www.ccs.neu.edu/~pjd/ftlsim`.

6. EXPERIMENTAL RESULTS

We have modified the simulator used in prior sections to incorporate a simple recency-based hot/cold data separation algorithm. Write sequence numbers are used rather than actual timestamps, and for each page in the LBA space an exponentially weighted moving average R of recency (i.e. the number of writes since a page was last overwritten) is calculated. If a cold page receives two writes in a row with inter-arrival time less than a cutoff of $0.7R$ it is considered hot; if a hot page is cleaned and has not been overwritten in $1.4R$ it are demoted to cold. At startup all pages are considered cold, and the first write to any page is ignored as no recency information is available.

This classifier was chosen for its ability to converge rapidly and accurately on the synthetic hot/cold workloads used in our experiments. We make no claims as to its utility on real workloads; this is an area of active research at this time [Park and Du 2011; Hsieh et al. 2006].

Traces were quantized into aligned 4K page accesses, and simulated device size was determined by the largest LBA referenced in a trace. Two OLTP traces from the UMass Trace Repository [Bates and McNutt 2007] were used (*fin1* and *fin2*) as well as a number of more general-purpose computing traces from Microsoft Research [Narayanan et al. 2008]: home (*rsrch_0*) and project (*proj_2*) directories, a source code control server (*src1_1*) and a web proxy (*prxy_0*).

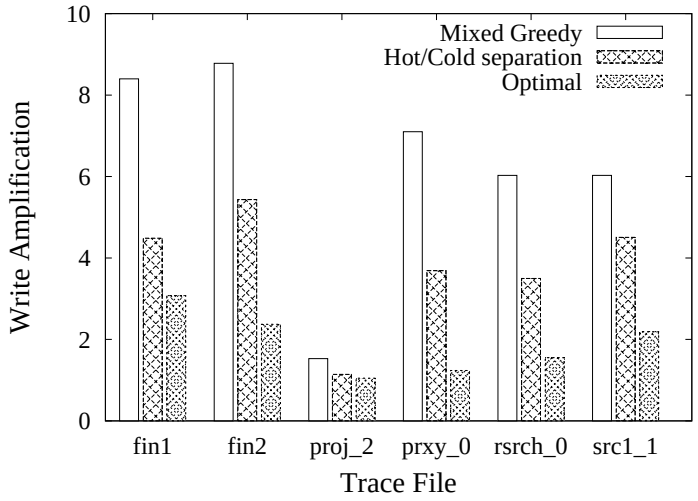


Fig. 20. Write amplification for write-only storage traces using global greedy, greedy with hot/cold separation, and optimized hot/cold cleaning; $N_p = 64, S_f = 0.07$.

In Figure 20 we see results for these traces, simulated with $S_f = 0.07$ and $N_P = 64$. The algorithms simulated are:

— Greedy: the naïve greedy cleaning algorithm, with no hot/cold separation.
— Hot/cold separation: separation of hot and cold pages into two pools, with global greedy cleaning.
— Optimal: hot/cold separation, cleaning based on Equation 45.

On the general-computing workloads, cleaning with optimal division of free space between empirically-determined hot/cold pools is seen to result in a write amplification of less than 2, even with this small spare factor, with naïve greedy cleaning between $2\times$ and $4\times$ worse. The OLTP traces have significantly less spatial locality, resulting in poorer performance, although still a factor of 2 better than naïve greedy.

7. PRIOR WORK

Several researchers, starting with Rosenblum [Rosenblum 1992], have examined the performance of block cleaning in the context of log-structured file systems. Analytic results for LRW cleaning have been independently derived and published at least four times: [Robinson 1996], [Menon and Stockmeyer 1998], [Xiang and Kurkoski 2012], [Desnoyers 2012]. Two of these papers—Robinson as well as Xiang and Kurkoski—describe the result as applying to Greedy cleaning rather than LRW cleaning, although for the large segment sizes considered by Robinson the difference is minor. The earlier two publications derive Equation 3 and obtain results numerically, while the more recent two provide the equivalent closed-form solution, Equation 5.

The earliest general performance model of FTL garbage collection known to the author is Ben-Aroya and Toledo's 2006 work, which proved hard analytic bounds for a simplified case [Ben-Aroya and Toledo 2006]. Despite the significance of this result, however, the assumptions used in their proof prevent their model from being used to predict performance in typical configurations. In particular, they assume that free space ($T-U$ in the above analysis) is a single block, causing severe performance degradation.

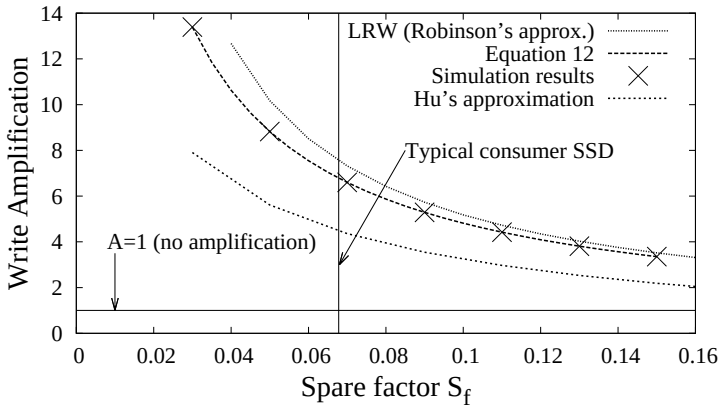


Fig. 21. Hu's approximation and Robinson's LRW model compared to Greedy cleaning simulation and analytic results (Equation 16), $N_p = 64$.

Subsequent to this, Baek et al. [Baek et al. 2007] identified free space ("utilization") as a key performance parameter, providing analytic results for block cleaning performance. However, the other main parameter of their model ("uniformity") is a measure of system state, itself the result of workload and cleaning algorithm. The problem has thus been transformed from one of predicting performance of an algorithm to one of predicting the device state resulting from that algorithm, but no solution is given for the transformed problem.

Boboila and Desnoyers have published mathematic models [Boboila and Desnoyers 2011] for performance of several hybrid page/block-mapped FTLs; however this work does not extend to fully page-mapped FTLs. Hu's 2009 paper [Hu et al. 2009] provides a compute-intensive procedure (typically taking more CPU time than accurate simulations) for numerically approximating the performance of the Greedy algorithm, with fairly good correspondence between predicted and simulated results for $S_f > 0.2$. Hu begins by using Windowed Greedy as an approximation of Greedy; we argue in Section 3.5 above that this approximation is unfounded, as for uniform traffic Windowed Greedy performance is in fact nearly identical to that of LRW.

Hu's approximation has large errors for values of S_f below 0.2, however; this range includes virtually all consumer SSDs, which tend to have free space ratios ranging from under 7% (e.g. the Intel device examined above, with $S_f = 0.069$) to about 13% (e.g. $U = 60 \times 10^9$ for a drive with 64×2^{30} of raw capacity gives $S_f = 12.7\%$). In Figure 21 we see Hu's approximation evaluated for lower S_f values and $N_p = 64$, compared with Equation 16, results from simulation, as well as LRW cleaning performance from Robinson's model. We see that Hu's approximation is not only less precise than our greedy cleaning model, but is in fact significantly less accurate than Robinson's LRW cleaning results; however we have not determined which step in Hu's derivation introduces this inaccuracy.

Early work by Bux described a Markov model of greedy cleaning performance [Bux 2009] for very small devices; the complexity of this model makes it unfeasible for realistic systems. Most recently, Bux and Iliadis [Bux and Iliadis 2010] describe the results described above in Section 3.3; this work improves on their results by providing a closed-form solution and extensions to non-uniform workloads.

In particular, we note that our optimal controller relies on the differentiability of the close-form performance function, an extension which is not possible with Bux and Iliadis' solution.

Agarwal and Marrow [Agarwal and Marrow 2010] give a closed-form expression for Greedy cleaning which is identical to the linear approximation to LRW performance in Section 3.2, based on a misinterpretation of empirical results. (their work assumes that the distribution of state occupancies shown in Figure 6 is uniform, rather than a function of the number of valid pages.) In 2012 the closed-form expression for LRW write amplification in Equation 5 was published independently by Xiang and Kurkoski [Xiang and Kurkoski 2012] and by the author [Desnoyers 2012].

Efforts to identify and exploit spatial locality in cleaning again begin with log-structured file systems, including work by Rosenblum [Rosenblum and Ousterhout 1991] using the simple two-parameter hot/cold traffic model used in this work, and proposing separation of hot and cold data at cleaning time and a *cost/benefit* calculation for cleaning decisions as an improvement over greedy cleaning. Chiang in turn proposed the Cost/Age/Times metric [Chiang et al. 1997] as well as a multiple-pool method for separating hot and cold data [Chiang and Chang 1999]. Chang proposed a *dual-pool* algorithm [Chang and Kuo 2002] for separating hot and cold data, similar to Chiang's multiple pools; however later versions of the dual-pool algorithm [Chang 2007] emphasize wear leveling rather than FTL performance, and the pools are used group physical blocks (rather than logical addresses) according to their write/erase cycle counts.

These methods have all used cleaning-time decisions between one of two write frontiers to implicitly group hot and cold data. Kim and Lee [Kim and Lee 2002] assume that the identity of hot and cold data is known exactly by the file system, while Jung [Jung et al. 2010] uses OS process information to inform hot/cold data separation. Hsieh [Hsieh et al. 2006] uses counting Bloom filters to track access frequency of data pages, while Park and Du [Park and Du 2011] use rotating Bloom filters to track recency instead of frequency.

We believe that our optimized cleaning strategy is relatively independent of the mechanism used to track hot and cold pages; Park and Du's Bloom filter-based recency estimator is likely to be a good substitute for our resource-intensive algorithm, and other tracking mechanisms may work as well. Finally, and most significantly, unlike prior work, our work provides models for predicting the effect of non-uniformity on performance, and for parameterizing an FTL to achieve peak performance in the presence of non-uniformity.

8. CONCLUSIONS

In this work we have presented a nearly-exact closed-form solution for write amplification of greedy block cleaning under uniformly distributed traffic, and an exact model for LRW cleaning with a simple hot/cold data mix which is also applicable to more complex workloads; in addition we provide a highly accurate approximation for greedy cleaning with hot and cold data. We validate the predictions of these models in simulation, on both synthetic data and (in the case of LRW cleaning) real workloads. In addition we use these models to derive new insight into the behavior of cleaning algorithms when data is segregated by update frequency ("hot/cold data separation"), providing a control algorithm which appears near-optimal for simple traffic mixes and achieves high performance for real-world workloads.

ACKNOWLEDGMENTS

The author would like to thank the anonymous reviewers for their extensive and invaluable feedback, and Andy Twigg for alerting me to Robinson's overly-neglected prior work in this area.

REFERENCES

AGARWAL, R. AND MARROW, M. 2010. A closed-form expression for write amplification in NAND flash. In *IEEE Globecom 2010 Workshop on Application of Communication Theory to Emerging Memory Technologies*. IEEE, Miami, Florida, 1846–1850.

BAEK, S., CHOI, J., LEE, D., AND NOH, S. H. 2007. Model and validation of block cleaning cost for flash memory. In *Int'l Conf. on Embedded Computer Systems: Architectures, Modeling, and Simulation*. Springer-Verlag, Samos, Greece, 46–54.

BAN, A. 2004. Wear leveling of static areas in flash memory. United States Patent 6,732,221.

BATES, K. AND MCNUTT, B. 2007. OLTP Application I/O. `http://traces.cs.umass.edu`. available from the UMass Trace Repository.

BEN-AROYA, A. AND TOLEDO, S. 2006. Competitive analysis of Flash-Memory algorithms. In *Algorithms - ESA 2006*. Springer-Verlag, Zurich, Switzerland, 100–111.

BLACKWELL, T., HARRIS, J., AND SELTZER, M. 1995. Heuristic cleaning algorithms in log-structured file systems. In *Proceedings of the 1995 USENIX Technical Conference*. USENIX Association, New Orleans, Louisiana, 277–288.

BOBOILA, S. AND DESNOYERS, P. 2011. Performance models of flash-based Solid-State drives for real workloads. In *IEEE Symposium on Massive Storage Systems and Technologies*. IEEE, Denver, Colorado.

BUX, W. 2009. Performance evaluation of the write operation in flash-based solid-state drives. IBM Research Report RZ 3757, IBM. Nov.

BUX, W. AND ILIADIS, I. 2010. Performance of greedy garbage collection in flash-based solid-state drives. *Performance Evaluation 67,* 11, 1172–1186.

CHANG, L. 2007. On efficient wear leveling for large-scale flash-memory storage systems. In *ACM Symposium on Applied Computing*. ACM, Seoul, Korea, 1126–1130.

CHANG, L. AND KUO, T. 2002. An adaptive striping architecture for flash memory storage systems of embedded systems. In *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'02)*. IEEE Computer Society, San Jose, California, 187.

CHIANG, M. AND CHANG, R. 1999. Cleaning policies in mobile computers using flash memory. *Journal of Systems and Software 48,* 3, 213–231.

CHIANG, M.-L., LEE, P. C. H., AND CHUAN CHANG, R. 1997. Managing flash memory in personal communication devices. In *Int'l Symposium on Consumer Electronics (ISCE'97)*. IEEE, Singapore, 177–182.

CHUNG, T.-S., PARK, D.-J., PARK, S., LEE, D.-H., LEE, S.-W., AND SONG, H.-J. 2009. A survey of flash translation layers. *Journal of Systems Architecture 55,* 5-6, 332–343.

DESNOYERS, P. 2012. Analytic modeling of ssd write performance. In *SYSTOR 2012: The Israeli Experimental Systems Conference*. ACM, Haifa, Israel.

GAL, E. AND TOLEDO, S. 2005. Algorithms and data structures for flash memories. *ACM Computing Surveys 37,* 2, 138–163.

HSIEH, J., KUO, T., AND CHANG, L. 2006. Efficient identification of hot data for flash memory storage systems. *ACM Transactions on Storage 2,* 1, 22–40.

HU, X., ELEFTHERIOU, E., HAAS, R., ILIADIS, I., AND PLETKA, R. 2009. Write amplification analysis in flash-based solid state drives. In *SYSTOR 2009: The Israeli Experimental Systems Conference*. ACM, Haifa, Israel, 1–9.

HU, X.-Y. AND HAAS, R. 2010. The fundamental limit of flash random write performance: Understanding, analysis and performance modelling. IBM Research Report RZ 3771, IBM Research - Zurich. Mar.

JUNG, S., LEE, Y., AND SONG, Y. 2010. A process-aware hot/cold identification scheme for flash memory storage systems. *IEEE Transactions on Consumer Electronics 56,* 2, 339 –347.

KIM, H. AND LEE, S. 2002. An effective flash memory manager for reliable flash memory space management. *IEICE Transactions on Information and Systems 85,* 6, 950–964.

LEE, H., YUN, H., AND LEE, D. 2009. HFTL: hybrid flash translation layer based on hot data identification for flash memory. *IEEE Transactions on Consumer Electronics 55,* 4, 2005–2011.

LEE, S.-W., PARK, D.-J., CHUNG, T.-S., LEE, D.-H., PARK, S., AND SONG, H.-J. 2007. A log buffer-based flash translation layer using fully-associative sector translation. *ACM Transactions on Embedded Computer Systems 6,* 3, 18.

LIN, W.-H. AND CHANG, L.-P. 2012. Dual greedy: Adaptive garbage collection for page-mapping solid-state disks. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2012*. 117 –122.

MATTHEWS, J. N., ROSELLI, D., COSTELLO, A. M., WANG, R. Y., AND ANDERSON, T. E. 1997. Improving the performance of log-structured file systems with adaptive methods. In *Proceedings of the sixteenth ACM symposium on Operating systems principles*. ACM, Saint Malo, France, 238–251.

MENON, J. AND STOCKMEYER, L. 1998. An age-threshold algorithm for garbage collection in log-structured arrays and file systems. In *High Performance Computing Systems and Applications*, J. Schaeffer, Ed. Vol. 478. Springer US, 119–132.

NARAYANAN, D., DONNELLY, A., AND ROWSTRON, A. 2008. Write off-loading: practical power management for enterprise storage. In *USENIX Conference on File and Storage Technologies*. USENIX Association, San Jose, California, 1–15.

PARK, D. AND DU, D. 2011. Hot data identification for flash-based storage systems using multiple bloom filters. In *2011 IEEE 27th Symposium on Mass Storage Systems and Technologies (MSST)*. 1 –11.

QI, F. AND GUO, B.-N. 2009. Sharp inequalities for the psi function and harmonic numbers. arXiv e-print 0902.2524. Feb.

ROBINSON, J. T. 1996. Analysis of steady-state segment storage utilizations in a log-structured file system with least-utilized segment cleaning. *SIGOPS Operating Systems Review 30,* 4, 29–32.

ROSENBLUM, M. 1992. The design and implementation of a log-structured file system. Ph.D. thesis, University of California at Berkeley.

ROSENBLUM, M. AND OUSTERHOUT, J. K. 1991. The design and implementation of a log-structured file system. In *ACM Symposium on Operating Systems Principles*. ACM, Pacific Grove, California, 1–15.

WANG, J. AND HU, Y. 2002. WOLF - a novel reordering write buffer to boost the performance of Log-Structured file systems. In *Proceedings of the Conference on File and Storage Technologies*. USENIX Association, Monterey, California, 47–60.

WU, M. AND ZWAENEPOEL, W. 1994. eNVy: a non-volatile, main memory storage system. In *Int'l Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. ACM Press, San Jose, California, 86–97.

XIANG, L. AND KURKOSKI, B. 2012. An improved analytic expression for write amplification in NAND flash. In *2012 International Conference on Computing, Networking and Communications (ICNC)*. IEEE, Maui, Hawaii, 497–501.