Solid State Disk Object-Based Storage with Trim Commands

Tasha Frankie, Member, IEEE, Gordon Hughes, Fellow, IEEE, and Ken Kreutz-Delgado, Senior Member, IEEE

Abstract—This paper presents a model of NAND flash SSD utilization and write amplification when the ATA/ATAPI SSD Trim command is incorporated into object-based storage under a variety of user workloads, including a uniform random workload with objects of fixed size and a uniform random workload with objects of varying sizes. We first summarize the existing models for write amplification in SSDs for workloads with and without the Trim command, then propose an alteration of the models that utilizes a framework of object-based storage. The utilization of objects and pages in the SSD is derived, with the analytic results compared to simulation. Finally, the effect of objects on write amplification and its computation is discussed along with a potential application to optimization of SSD usage through object storage metadata servers that allocate object classes of distinct object size.

Index Terms—Trim, Solid State Drive, Object-Based Storage, Write Amplification, NAND Flash, Markov processes.

1 INTRODUCTION

In recent years, NAND flash solid state drives (SSDs) have become ubiquitous, appearing in small mobile electronics to large data centers. Part of the popularity of flash-based SSDs over traditional hard disk drives (HDDs) is the increased speed and performance that they offer. However, SSDs have their own set of problems, including a slowdown in performance as the device fills with data due to write amplification, and a limited lifetime due to wearout of the floating-gate transistors. Additionally, SSDs currently need to conform with legacy interfaces from HDDs, which makes optimization in using the device more challenging.

Object based storage offers multiple advantages for systems using solid state storage devices [1]. When the multiple thousand pages in a flash block are divided into object records of larger size, the resultant fewer objects per block directly translates into fewer data write relocations during garbage collection, leading to a lower write amplification and faster performance. This offers an opportunity for user application storage metadata, via metadata servers, to actively allow flash storage devices to self-optimize their hardware, to fully utilize SSD write speed advantages over magnetic HDD while ameliorating SSD limitations like block erase and finite write lifetime. Object storage metadata servers could allocate SSDs to object classes [2] of distinct object size and thereby allow SSD to internally manage their object physical mapping and relocation. This has the potential to improve I/O rate and QoS, user access blocking probability, and SSD flash chip lifetime through the reduction of write amplification.

In this paper, we present a model for computing utilization and write amplification under a modified random workload including Write and Trim commands, and data records of varying sizes. This model is an extension of models proposed for simpler workloads by previous researchers [3], [4], [5], [6] and builds on our previous models of the Trim command [7], [8], [9]. Good models assist in the development of algorithms for managing the challenges of SSDs; our model adds flexibility to the theory in order to more closely match real-world workload characteristics.

2 NAND FLASH SSDs

In this section, we provide background about NAND Flash SSDs which is necessary to understand the model presented in this paper.

2.1 Layout

The organization of a NAND flash SSD gives it characteristics that cause unique challenges. The smallest storage unit of flash is the page, which typically holds 2K, 4K, or 8K of data [10], [11]. Pages are grouped into blocks, with blocks typically containing 64, 128, or 256 pages, varying by manufacturer [10]. One challenge of flash SSDs is that a page must be erased before it can be programmed again, but the smallest erase unit of flash SSDs is the block, and not all pages on the block are ready to be erased at the same time. This means that write-in-place schemes are impractical, and most SSDs employ a dynamic logical-to-physical mapping...
called the flash translation layer (FTL) [12], [13]. With this dynamic mapping, when data is overwritten, the old physical location of the data is marked as invalidated in the FTL mapping. In this paper, we utilize a log-structured file system, in which data is written to successively available pages, as described in [3].

2.2 Garbage Collection

Garbage collection is the process of selecting and erasing a block to reclaim space for writing. When a block is chosen for erasure, any valid pages on the block first need to be copied elsewhere to avoid data loss. Copying these valid pages leads to write amplification, a phenomenon in which the device performs more writes than are requested of it.

There are many algorithms used to select a block for garbage collection, most of which are based on one of two schemes originally described in the seminal paper by Rosenblum and Ousterhout [14]. Their first method of selection chooses the block that creates the most free space, and their second method also accounts for the age of the block. In this paper, we utilize greedy garbage collection, in which the block with the fewest valid pages is selected.

2.3 Improving Performance

Reducing write amplification creates an improvement in performance by increasing the average write speed due to the fewer number of extra writes performed by the device. Additionally, by reducing the number of program/erase cycles, SSD lifetime is extended, since flash devices wear out and lose the ability to store data after being written many times.

There are several ways to reduce write amplification. One common way is for manufacturers to provide extra storage space on the SSD beyond what the user is allowed to access, a practice called overprovisioning. Overprovisioning reduces write amplification because the amount of time between garbage collections increases, which tends to lead to a decrease in the number of valid pages in the block selected for reclamation. We measure the amount of overprovisioning with the spare factor, \( S_f = (T - u)/T, 0 \leq S_f \leq 1 \), where \( T \) is the total number of pages on the SSD, and \( u \) is the number of pages in the user space.

Another way to reduce write amplification is through use of the Trim command [15], which is a way for the file system to communicate to the SSD that certain data pages no longer need to be retained. The pages associated with the data are invalidated in the FTL mapping, meaning that there are fewer valid pages that need copying during garbage collection.

3 ANALYZING PERFORMANCE

Several researchers have created analytic models to help understand the performance of SSDs. In this section, we summarize previous work that we build on in this paper.

3.1 Uniform Random Workload

The type of workload used can have a large effect on the performance of a storage device [16]. We view real-world workloads as existing along a continuum from purely sequential workloads to completely random workloads. For SSDs, analysis of a purely sequential workload is uninteresting, because entire blocks of data are invalidated before garbage collection, leading to no write amplification [3]. However, at the other end of the spectrum, completely random workloads are very interesting to statistically analyze, since the random nature of the writes results in most blocks still containing valid pages at the time of garbage collection, leading to potentially high write amplification. Hu, et al. [3], [4], Agarwal and Marrow [5], and Xiang and Kurkoski [6] have all proposed models for write amplification of the uniform random workload under greedy garbage collection. Their workloads consist of write requests distributed uniformly randomly over the user space, with each logical block address (LBA) requested requiring one page to store the data. Each of their models found that the single most important factor in determining the write amplification was the level of overprovisioning provided on the SSD.

3.2 Trim-Modified Uniform Random Workload

Previously, we modified the uniform random workload to incorporate Trim requests, then analyzed the resulting utilization and write amplification [7], [8]. In the Trim-modified uniform random workload, both write and Trim requests are allowed, and the data associated with each LBA occupies one page. As in the standard uniform random workload, write requests are uniformly randomly chosen over all user LBAs, and Trim requests are uniformly randomly chosen over all In-Use LBAs, where an LBA is defined as being In-Use if its most recent request was a Write. Trim requests occur with probability \( q \), and write requests occur with probability \( 1 - q \). This workload was modeled as a Markov birth-death chain with the state being the number of In-Use LBAs, and the steady state solution was approximated as Gaussian with mean \( us \) and variance \( u\bar{s} \), where \( s = (1 - 2q)/(1 - q) \) and \( \bar{s} = q/(1 - q) \) are dependent on the rate of Trim.

3.2.1 Effective Overprovisioning

Since the number of In-Use LBAs is the same as the number of valid pages, the level of effective overprovisioning is straightforward to calculate. The effective overprovisioning is measured in terms of the effective spare factor, \( S_{\text{eff}} = (T - X_n)/T \), where \( X_n \) is the...
number of valid pages at time \( n \). The mean effective spare factor was found to be

\[
\bar{S}_{\text{eff}} = \frac{T - us}{T} = \bar{s} + sS_f
\]

The variance of the effective spare factor was also calculated, and found to be negligible in practical devices (as \( T \to \infty \)) \cite{7, 9}.

### 3.2.2 Write Amplification

We also explored write amplification under the Trim-modified random workload. After paralleling the derivation of Xiang and Kurkoski \cite{6}, we discovered that all three previously proposed models for write amplification could be modified by using the effective level of overprovisioning. In simulation, the Trim-modified model of Xiang and Kurkoski provided the best prediction; the Trim-modified models of Hu, et al., and of Agarwal and Marrow were optimistic in their predictions of write amplification \cite{5, 9}.

Additionally, we separated the workload into hot (frequently accessed) and cold (infrequently accessed) data in order to closer approximate a real-workload. Analysis of this workload led to a more general workload with \( N \) temperatures, and found that the write amplification of this workload type was lowest when the \( N \) temperatures were kept physically separated into different blocks \cite{9}.

### 4 Model Augmentation

In the previous work described in Section 3, one page stored the data associated with one LBA. In this section, we augment the model to allow larger amounts of data to be stored under a single identifier, which lends itself naturally to a discussion of object-based storage.

#### 4.1 From LBAs to Objects

In object-based storage, the filesystem does not assign data to particular LBAs. Instead, the filesystem tells the object-based storage disk to store certain data; the disk chooses where to store it, and gives the filesystem a unique object ID with which the data can be retrieved and/or modified \cite{17}. Object-based storage allows for variable-sized objects with identifiers that need not indicate any information about the physical location of the data \cite{18}. Rajimwale, et al. have proposed that object-based storage is an optimal interface for flash-based SSDs \cite{19}. We use object-based storage as motivation for extensions to the theory of effective overprovisioning and write amplification described in Section 3.

The utilization problem we solved previously \cite{7} can be viewed in object space instead of LBA space. Instead of LBAs that are In-Use or not In-Use, there are object IDs that are In-Use or not In-Use. Previously, there were \( u \) LBAs allowed; now we allow object IDs in the range \([1, u]\), for a maximum of \( u \) objects. The object-view of the Trim-modified uniform random workload is the same as the LBA view of the problem when each object occupies 1 page.

To remove the restriction that one object occupies only one page, we start by examining the workload in which each object occupies \( b \) pages, where \( b \) is constant and identical for all objects in the workload. Then, we examine the workload in which \( b \) varies each time an object is written. For all workloads, we assume that when an object ID is selected for trimming (or, more appropriately for objects, deletion), all pages associated with that object ID are invalidated. Likewise, when an object is rewritten, all pages previously containing data associated with that object ID are invalidated. Note that there are now two quantities of interest: the number of object IDs that are in-use, and the number of pages that are valid. When each object occupies one page, these two quantities are the same.

Define the following variables:

\[
\chi_{i, t} = (\chi_{1, t}, \chi_{2, t}, \ldots, \chi_{u, t})^T
\]

\( \chi_{i, t} \) is a vector of indicator random variables at time \( t \).

\[
\chi_{i, t} = \mathbf{1}_{A_i}(\omega) = \begin{cases} 1 & \omega \in A_i \text{ (object i is In-Use)} \\ 0 & \text{otherwise (object i is not In-Use)} \end{cases}
\]

where

\( A_i = \{ \omega | \text{the STATE(\omega) is such that location i is occupied} \} \)

\[
X_t = \sum_{i=1}^{u} \chi_{i, t}
\]

is a random variable representing the number of objects in-use at time \( t \). The mean and variance of this random variable at steady state were computed previously by Frankie, et al. \cite{7, 9}.

\[
Z_t = (Z_{1, t}, Z_{2, t}, \ldots, Z_{u, t})^T
\]

is a vector of the size of each object at time \( t \).

Now, compute the number of valid pages at time \( t \):

\[
Y_t = Z_t^T X_t = \sum_{i=1}^{u} \chi_{i, t} Z_{i, t}
\]

We are concerned with the values at steady state, and drop the time index to indicate the steady state solution.

#### 4.2 Fixed Sized Objects

This workload is an extension of the original workload whose steady state pdf was derived in \cite{7}. However, objects now occupy \( b \) pages instead of 1 page. Recall that the number of In-Use objects is the same as the number of In-Use LBAs previously derived; only the number of valid pages will change. We assume that at least \( ub \) pages are allowed for the user to write to, so that it is not possible to request more space than is available.
For this workload, $Z_t = (b, b, \ldots, b)^T \forall t$. This means

$$Y_t = \sum_{i=1}^{u} \chi_{i,t} Z_{i,t}$$

$$= \sum_{i=1}^{u} \chi_{i,t} b$$

$$= b \sum_{i=1}^{u} \chi_{i,t}$$

$$= bX_t$$

To compute the steady state mean and variance of $Y$, we can use the known steady state mean and variance of $X$.

$$E[Y] = E[bX]$$

$$= bE[X]$$

$$= bus$$

Similarly,

$$Var[Y] = Var[bX]$$

$$= b^2 Var[X]$$

$$= b^2us$$

### 4.3 Variable Sized Objects

In this workload, the size of each object varies independently over time. Each time an object ID is selected to be written, the size of the object is chosen randomly, according to the distribution from which $Z_{i,t}$ is drawn, regardless of any size that object ID may have had in the past. $Z_{i,t}$ is drawn from a distribution with a minimum value $B_1 \geq 1$, a maximum value $B_2 \geq B_1$, a mean value $m_Z$, and a standard deviation $\sigma_Z$. Again, we assume there is sufficient space for writing all requests; this means there are at least $ubZ$ pages allowed for the user. Because $Z_{i,t}$ is not constant, $Y_t = \sum_{i=1}^{u} \chi_{i,t} Z_{i,t}$ cannot be simplified, as it was in the fixed-sized object case.

The mean is still relatively straightforward to calculate, since $\chi_i$ and $Z_i$ are independent, and the $Z_i$s are iid:

$$E[Y] = E\left[ \sum_{i=1}^{u} \chi_i Z_i \right]$$

$$= \sum_{i=1}^{u} E[\chi_i] E[Z_i]$$

$$= \sum_{i=1}^{u} E[\chi_i] m_Z$$

$$= m_Z \sum_{i=1}^{u} E[\chi_i]$$

$$= m_Z E\left[ \sum_{i=1}^{u} \chi_i \right]$$

The variance is more complicated because of cross-terms. The $\chi_i$ values are not independent, so it is not simple to compute the variance. To compute the variance of $Y$, compute

$$Var(Y) = E[Y^2] - E[Y]^2$$

The value of $E[Y]$ was computed in Equation[1] Computing $E[Y^2]$ is more complicated. When following the steps below, note that $\chi_i^2 = \chi_i$ , $\chi_i$ is independent of $Z_i$ and $Z_j$, and the $Z_i$ values are iid.

$$E[Y^2] = E\left[ \left( \sum_{i=1}^{u} \chi_i Z_i \right)^2 \right]$$

$$= E \left[ \sum_{i=1}^{u} \chi_i Z_i^2 + 2 \sum_{i<j} \chi_i \chi_j Z_i Z_j \right]$$

$$= \sum_{i=1}^{u} E[\chi_i Z_i^2] + 2 \sum_{i<j} E[\chi_i \chi_j Z_i Z_j]$$

$$= \sum_{i=1}^{u} E[\chi_i] E[Z_i^2]$$

$$+ 2 \sum_{i<j} E[\chi_i \chi_j] E[Z_i] E[Z_j]$$

Most of these expected values are straightforward to calculate. However, care must be taken when computing $E[\chi_i \chi_j]$, because the $\chi_i$ values are not independent. See Section 4.3.1 for details.

Continuing,

$$E[Y^2] = (\sigma_Z^2 + m_Z^2)us + 2m_Z^2 \sum_{i<j} \frac{s(us - (u-1)\frac{s}{u})}{u-1}$$

$$= (\sigma_Z^2 + m_Z^2)us + 2m_Z^2 \frac{(u-1)us(us - 1 + \frac{s}{u})}{u-1}$$

$$= \sigma_Z^2us + m_Z^2us + m_Z^2us \left( us - 1 + \frac{s}{u} \right)$$

$$= \sigma_Z^2us + (m_Zus)^2 + m_Z^2us$$

Now it is possible to compute

$$Var(Y) = E[Y^2] - E[Y]^2$$

$$= \sigma_Z^2us + (m_Zus)^2 + m_Z^2us - (m_Zus)^2$$

$$= \sigma_Z^2us + m_Z^2us$$

#### 4.3.1 Computing $E[\chi_i \chi_j]$

As mentioned previously, care must be taken when computing $E[\chi_i \chi_j]$, because the $\chi_i$ values are not independent.

$$E[\chi_i \chi_j] = P(A_i \cap A_j)$$

$$= P(A_i|A_j)P(A_j)$$

1. If the $\chi_i$ values were independent, $X$ would have a binomial distribution, but because we know from previous analysis that the variance is $us$ and not $us^2$, this is obviously not the case.
Obviously, when \( i = j \), \( P(A_i|A_j) = 1 \). However, calculating this value when \( i \neq j \) is important. Since there is nothing special about any particular \( i \) or \( j \), assume that \( P(A_i|A_j) \) is the same for all \( i \neq j \), and thus is a constant with respect to \( i \) and \( j \).

It is tempting to claim that \( P(A_i|A_j) = (us-1)/(u-1) \), since \( us \) is the expected number of In-Use objects at steady state. However, this is an invalid statement because it implicitly assumes a zero variance. In fact, using this value in the calculation of \( Var(Y) \) results in a statement that \( Var(Y) = 0 \), which is exactly what is expected because of the implicit zero variance assumption.

Instead of trying to compute \( P(A_i|A_j) \) directly, it is possible to compute it by using the result of the one object occupies one page model. Working backwards to calculate this probability, and noting that there are \((u-1)/u\) terms in the sum over \( i < j \):

\[
Var(Y) = Var\left( \sum_{i=1}^{n} \chi_i \right) = \sum_{i=1}^{u} Var(\chi_i) + 2 \sum_{i < j} Cov(\chi_i, \chi_j)
\]

\[
= \sum_{i=1}^{u} s\bar{s} + 2 \sum_{i < j} C
\]

\[
us\bar{s} = us\bar{s} + 2u(u-1)C
\]

Solving for \( C \) yields \( \frac{s^2}{u-1} \), which is the value of the covariance. Continuing with the calculations,

\[ Cov(\chi_i, \chi_j) = C = E[\chi_i\chi_j] - P(A_i)P(A_j) = E[\chi_i\chi_j] - s^2 \]

so that

\[ E[\chi_i\chi_j] = C + s^2 = \frac{s^2}{u-1} + s^2 = s \left( \frac{us - 1 + \frac{s}{u}}{u - 1} \right) \]  \hspace{1cm} (2)

### 4.4 Simulation Results

Monte-Carlo simulation was used to verify the analytic results, with various values of \( u, q \), and distributions of \( Z \). In this paper, results are shown for \( u = 1000 \). Simulations were run for 1 million requests to ensure steady state was reached, then the simulation means and variances were averaged over the next 9 million requests. The results in each table are the average of 64 Monte-Carlo simulations. Although results are only shown for \( u = 1000 \), results for larger values of \( u \) are comparable.

Table 4 shows the analytic and simulation values of the mean and standard deviation of the number of In-Use objects as well as the mean and standard deviation of the number of valid pages at steady state when the objects are all a fixed size of 32 pages. When rounded to the nearest integer, the mean number of objects is predicted exactly by analysis; the mean number of valid pages is predicted to within 3 pages. The slight error in the prediction of the number of valid pages is likely due to the extremely large variances seen when \( q \geq 0.3 \). An overlay of the simulation pdf and the analytic pdf for the number of valid pages is shown in Fig. 1 for \( q = 0.2 \).

Table 2 shows the analytic and simulation values of the mean and standard deviation of the number of In-Use objects as well as the mean and standard deviation of the number of valid pages at steady state when the size of the objects is drawn from a discrete uniform distribution over \([1, 32]\). Analytic and simulation values match exactly when rounded to the nearest integer. An overlay of the simulation pdf and the analytic pdf for the number of valid pages is shown in Fig. 2 for \( q = 0.2 \).

Table 3 shows the analytic and simulation values of the mean and standard deviation of the number of In-Use objects as well as the mean and standard deviation of the number of valid pages at steady state when the size of the objects is drawn from a binomial distribution with parameters \( p = 0.4 \) and \( n = 32 \). Analytic and simulation values match exactly when rounded to the nearest integer. An overlay of the simulation pdf and the analytic pdf for the number of valid pages is shown in Fig. 3 for \( q = 0.2 \).
### Table 1
Mean and $\sigma$ of Number of In-Use Objects and Number of Valid Pages for Fixed Sized Objects. Values in this table were computed with $u = 1000$. The size of the objects was fixed at 32. Simulation values are the average of 64 Monte-Carlo simulations.

| Trim Amount $q$ | Mean Objects | $\sigma$ Objects | Mean Pages | $\sigma$ Pages |
|----------------|--------------|-----------------|------------|---------------|
|                | Analytic     | Simulation      | Analytic   | Simulation    |
| 0.05           | 947.37       | 947.37          | 7.25       | 7.24          |
| 0.1            | 888.89       | 888.89          | 10.54      | 10.54         |
| 0.2            | 750.00       | 750.01          | 15.81      | 15.82         |
| 0.3            | 571.43       | 571.47          | 20.70      | 20.73         |
| 0.4            | 333.33       | 333.43          | 25.82      | 25.83         |
| 0.45           | 181.82       | 181.89          | 28.60      | 28.62         |

### Table 2
Mean and $\sigma$ of Number of In-Use Objects and Number of Valid Pages for Variable Sized Objects. Values in this table were computed with $u = 1000$. The size of the objects was drawn from a discrete uniform distribution over [1, 32]. Simulation values are the average of 64 Monte-Carlo simulations.

| Trim Amount $q$ | Mean Objects | $\sigma$ Objects | Mean Pages | $\sigma$ Pages |
|----------------|--------------|-----------------|------------|---------------|
|                | Analytic     | Simulation      | Analytic   | Simulation    |
| 0.05           | 947.37       | 947.36          | 7.25       | 7.27          |
| 0.1            | 888.89       | 888.92          | 10.54      | 10.53         |
| 0.2            | 750.00       | 750.00          | 15.81      | 15.80         |
| 0.3            | 571.43       | 571.41          | 20.70      | 20.79         |
| 0.4            | 333.33       | 333.35          | 25.82      | 25.79         |
| 0.45           | 181.82       | 181.81          | 28.60      | 28.58         |

### Table 3
Mean and $\sigma$ of Number of In-Use Objects and Number of Valid Pages for Variable Sized Objects. Values in this table were computed with $u = 1000$. The size of the objects was drawn from a binomial distribution over with parameters $p = 0.4$ and $n = 32$. Simulation values are the average of 64 Monte-Carlo simulations.

| Trim Amount $q$ | Mean Objects | $\sigma$ Objects | Mean Pages | $\sigma$ Pages |
|----------------|--------------|-----------------|------------|---------------|
|                | Analytic     | Simulation      | Analytic   | Simulation    |
| 0.05           | 947.37       | 947.36          | 7.25       | 7.27          |
| 0.1            | 888.89       | 888.88          | 10.54      | 10.53         |
| 0.2            | 750.00       | 750.01          | 15.81      | 15.80         |
| 0.3            | 571.43       | 571.37          | 20.70      | 20.68         |
| 0.4            | 333.33       | 333.32          | 25.82      | 25.79         |
| 0.45           | 181.82       | 181.84          | 28.60      | 28.63         |

Fig. 2. Comparison of analytic and simulation pdf for the number of valid pages when the size of objects is drawn from a discrete uniform distribution over $[1, 32]$. For this example, $u = 1000$ and $q = 0.2$.

Fig. 3. Comparison of analytic and simulation pdf for the number of valid pages when the size of objects is drawn from a binomial distribution over $p = 0.4$ and $n = 32$. For this example, $u = 1000$ and $q = 0.2$. 
5 APPLICATION TO WRITE AMPLIFICATION

In this section, we examine write amplification when one object requires a fixed number of pages to store data. We assume that the number of pages required to store the data associated with the object evenly divides the number of pages per block (i.e. \( \text{mod}(n_p, b) = 0 \)), so that an entire object is stored within a single block. As in previous work, data is stored in the fashion of a log-structured file system with greedy garbage collection.

We argue that fixed sized objects with sizes that evenly divide the number of pages per block is equivalent to reducing the number of pages in a block. For instance, if two objects can fit in one block, this is equivalent to having a block with two (large size) pages, and objects that require one of these (large) pages to store the associated data. With this equivalence between reduced number of pages per block and increased object size, the formulas for write amplification with Trim that we previously developed [8] should apply. However, the Trim-modified Agarwal and the Trim-modified Xiang models do not take \( n_p \) into account! The derivation of the Trim-modified Agarwal model causes \( n_p \) to drop out naturally; it is not a good candidate for modification. The Trim-modified Xiang model is a better candidate, as asymptotically removes \( n_p \) in the derivation. By eliminating this asymptotic approximation, the write amplification is computed as \( n_p/y \), where

\[
y = n_p + \frac{W\left(T(1+\delta)\left(1 - \frac{1}{\pi(1+\delta)}\right)\ln \left(1 - \frac{1}{\pi(1+\delta)}\right)\right)}{\frac{T}{n_p}(1+\delta)\ln \left(1 - \frac{1}{\pi(1+\delta)}\right)}
\]

and \( W(\cdot) \) is the Lambert W function [20], [6], [8]. The Trim-modified Hu model does not need to be changed in order to take \( n_p \) into account, as it already does this. The model can be found in Figure 4.

Results comparing the simulated write amplification and the computed write amplification are in Table 4 and Figure 5. In this simulation, there were 1280 blocks, with a manufacturer-specified spare factor of 0.2, and a Trim factor of 0.1. As the number of pages per block changed, the effective spare factor was held constant to enable comparisons. As expected, the write amplification was reduced as the number of pages per block decreased, with the minimum possible write amplification of 1 achieved with 1 page in a block. The modification of the Xiang model shown in Equation 5 was a very poor approximation as the number of pages in a block decreased; this is likely due to the assumption that the number of invalid pages freed in a block by garbage collection is binomial; with a small number of pages per block, this approximation becomes too coarse. However, the Trim-modified Hu model provides a reasonable, if optimistic, prediction of write amplification. Note that the Trim-modified Hu model does not make sense to compute if there is one page per block, so the entry in the table is left blank.

\[
A_{\text{Hu}} = \frac{n_p}{n_p - \sum_{k=0}^{n_p} kp_k^*}
\]

Computations of Trim-Modified Write Amplification Factor

| Compute \( p = (p_1, \cdots, p_j, \cdots, p_w) \) |
| \( p_1 = e^{-1.9 \left( \frac{T}{us} - 1 \right)} \) (4) |
| \( p_j = \min \left( 1, \frac{1.1 p_{j-1}}{(1 - \frac{1}{us})^{n_p}} \right) \) (5) |

Compute \( v = (v_{j,k}) \), a matrix of size \( w \times n_p \), \( j = 1, \cdots, w \) & \( k = 0, \cdots, n_p - 1 \)

\[
v_{j,k} = 1 - \sum_{m=0}^{k} \left( \frac{n_p}{m} \right) p_j^m (1 - p_j)^{n_p-m}
\]

Compute \( V = (V_1, \cdots, V_k, \cdots, V_{n_p}) \)

\[
V_k = \prod_{j=0}^{w-1} v_{j,k}
\]

Compute \( p^* = (p_0^*, \cdots, p_k^*, \cdots, p_{n_p}^*) \)

\[
p_0^* = 1 - V_0 \quad p_k^* = V_{k-1} - V_k \quad \text{for } k = 1, \ldots, n_p - 1 \quad p_{n_p}^* = 1 - V_{n_p-1}
\]

Compute the write amplification,

\[
A_{\text{Hu}} = \frac{n_p}{n_p - \sum_{k=0}^{n_p} kp_k^*}
\]

Fig. 4. Equations needed to compute the Trim-modified empirical model-based algorithm for \( A_{\text{Hu}} \) (adapted from [3], [4] and [8]). \( w \) is the window size, and allows for the windowed greedy garbage collection variation of greedy garbage collection. Setting \( w = \frac{n_p}{n_p} - r \) is needed for the standard greedy garbage collection discussed in this paper. For an explanation of the theory behind these formulas, see [3], [4].

When objects do not fit neatly within blocks, such as the case for \( \text{mod}(n_p, b) \neq 0 \), for variable sized objects, or for objects larger than one block, the computation of write amplification is not as straightforward. First, design decisions must be made: should an object be split over multiple blocks, or should empty, unwritten pages be left on a block, that perhaps could be filled in later by a smaller object? The design decision will affect the analysis of the write amplification; this is an area for future research.
Write amplification for varying number of pages per block, equivalent to fixed sized objects that evenly divide $n_p$. The level of effective overprovisioning was kept constant for each simulation. The amount of Trim was $q = 0.1$. All models referenced are Trim-modified; the Xiang model was additionally modified to account for $n_p$, as shown in Equation 3.

| Pages / Block | Write Amplification |
|---------------|---------------------|
|               | Simulation | Modified | Xiang | Hu |
| 1             | 1.000      | 1.936    | -     | -  |
| 2             | 1.191      | 1.937    | 1.000 | -  |
| 4             | 1.432      | 1.938    | 1.065 | -  |
| 8             | 1.631      | 1.938    | 1.274 | -  |
| 16            | 1.771      | 1.938    | 1.628 | -  |
| 32            | 1.853      | 1.938    | 1.732 | -  |
| 64            | 1.896      | 1.938    | 1.793 | -  |
| 128           | 1.918      | 1.938    | 1.828 | -  |
| 256           | 1.929      | 1.938    | 1.847 | -  |

Fig. 5. Write amplification versus pages per block for constant effective overprovisioning. The modified Xiang model (Equation 3) fails when the number of pages per block is low. The Trim-modified Hu model provides an optimistic, but reasonable, prediction.

6 CONCLUSION

In this paper, the Trim model we previously presented in [7], [8], [9] was extended to encompass the idea of object-based storage and allow objects to utilize more than one page for data storage. A utilization model was derived and shown to be accurate via simulation. Additionally, models for write amplification were modified and compared to simulated values, demonstrating the failure point of one model and the need for a more accurate model in another. Models such as the ones presented in this paper are useful tools for understanding the problems that an FTL design needs to address, and can lead to improved future designs.

The model extension presented in this paper helps make the workload a more characteristic of real-world workloads. By using this model in conjunction with object-based storage, we envision metadata servers that separate object classes of distinct sizes to unique flash SSDs in order to assist in self-optimization of hardware that would improve overall data access performance and improve the lifetime of the SSD.

REFERENCES

[1] Y. Kang, J. Yang, and E. Miller, “Efficient storage management for object-based flash memory,” in 2010 18th Annual IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems. IEEE, 2010, pp. 407–409.
[2] D. Feng and L. Qin, “Adaptive object placement in object-based storage systems with minimal blocking probability,” in Advanced Information Networking and Applications, 2006.AINA 2006. 20th International Conference on, vol. 1. IEEE, 2006, pp. 611–616.
[3] X. Hu, E. ELEFTHERIOU, R. HAAS, I. Iliadis, and R. Pletka, “Write amplification analysis in flash-based solid state drives,” in Proceedings of SYSTEM 2009: The Israeli Experimental Systems Conference. ACM, 2009.
[4] X. Hu and R. Haas, “The fundamental limit of flash random write performance: Understanding, analysis and performance modelling,” IBM Research Report, vol. RZ 3771, Mar 2010.
[5] R. Agarwal and M. Marlow, “A closed-form expression for Write Amplification in NAND Flash,” in GLOBECOM Workshops (GC Wkshps), 2010 IEEE. IEEE, 2010, pp. 1846–1850.
[6] L. Xiang and B. Kurkoski, “An Improved Analytical Expression for Write Amplification in NAND Flash,” in International Conference on Computing, Networking, and Communications, 2012, Also available at Arxiv preprint arXiv:1110.4245.
[7] T. Frankie, G. Hughes, and K. Kreutz-Delgado, “A Mathematical Model of the Trim Command in NAND-Flash SSDs,” in Proceedings of the 50th Annual Southeast Regional Conference. ACM, 2012.
[8] ——, “The Effect of Trim Requests on Write Amplification in Solid State Drives,” in Proceedings of the IASTED-CIIT Conference. IASTED, 2012.
[9] ——, “Analysis of trim commands on overprovisioning and write amplification in solid state drives,” ACM Transactions on Storage (TOS), 2012, in Submission.
[10] L. Grupp, A. Caulfield, J. Coburn, S. Swanston, E. Yaakobi, P. Siegel, and J. Wolf, “Characterizing flash memory: anomalies, observations, and applications,” in Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture. ACM, 2009, pp. 229–240.
[11] D. Roberts, T. Kgil, and T. Mudge, “Integrating NAND flash devices onto servers,” Commun. ACM, vol. 52, no. 4, pp. 98–103, 2009.
[12] T. Chung, D. Park, S. Park, D. Lee, S. Lee, and H. Song, “A survey of flash translation layer,” Journal of Systems Architecture, vol. 55, no. 5-6, pp. 332–343, 2009.
[13] A. Gupta, Y. Kim, and B. Urraoaokar, “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. ACM, 2009, pp. 229–240.
[14] M. Rosenblum and J. Ousterhout, “The design and implementation of a log-structured file system,” ACM Transactions on Computer Systems (TOCS), vol. 10, no. 1, pp. 26–52, 1992.
[15] Information Technology - ATA/ATAPI Command Set - 2 (ACS-2), INCITS Working Draft T13/2015-D Rev. 7 ed., June 2011, http://www.t13.org/documents/UploadedDocuments/docs2011/d2015r7-ATAATAPI_CommandSet-2-ACS-2.pdf.
[16] N. Agrawal, V. Prabhakaran, T. Webber, J. Davis, M. Manasse, and R. Panigrahy, “Design tradeoffs for SSD performance,” in USENIX 2008 Annual Technical Conference and Annual Technical Conference. USENIX Association, 2008, pp. 57–70.
[17] Information Technology - SCSI Object-Based Storage Device Commands-3 (OSD-3), INCITS Working Draft T10/2128-D Rev. 0 ed., March 2009, http://www.t10.org/cgi-bin/ac.pl?t=1&id=osd3r00.pdf.
[18] Y. Kang, J. Yang, and E. Miller, “Object-based SCM: An Efficient Interface for Storage Class Memories,” in *Mass Storage Systems and Technologies (MSST), 2011 IEEE 27th Symposium on*. IEEE, 2011, pp. 1–12.

[19] A. Rajimwale, V. Prabhakaran, and J. Davis, “Block management in solid-state devices,” in *Proceedings of the 2009 conference on USENIX Annual technical conference*. USENIX Association, 2009, pp. 21–21.

[20] R. Corless, G. Gonnet, D. Hare, D. Jeffrey, and D. Knuth, “On the LambertW function,” *Advances in Computational mathematics*, vol. 5, no. 1, pp. 329–359, 1996.