A Novel Simulation Methodology for Accelerating Reliability Assessment of SSDs

University of Virginia Dept. of Computer Science Technical Report CS-2013-03

Luyao Jiang[2] and Sudhanva Gurumurthi[1, 2]

lj5bp@virginia.edu
sudhanva.gurumurthi@amd.com
[1]AMD Research, Advanced Micro Devices, Inc.
[2]Department of Computer Science, University of Virginia

June 25, 2013

Reliability is an important factor to consider when designing and deploying SSDs in storage systems. Both the endurance and the retention time of flash memory are affected by the history of low-level stress and recovery patterns in flash cells, which are determined by the workload characteristics, the time during which the workload utilizes the SSD, and the FTL algorithms. Accurately assessing SSD reliability requires simulating several years' of workload behavior, which is time-consuming. This paper presents a methodology that uses snapshot-based sampling and clustering techniques to reduce the simulation time while maintaining high accuracy. The methodology leverages the key insight that most of the large changes in retention time occur early in the lifetime of the SSD, whereas most of the simulation time is spent in its later stages. This allows simulation acceleration to focus on the later stages without significant loss of accuracy. We show that our approach provides an average speedup of 12X relative to detailed simulation with an error of 3.21% in the estimated mean and 6.42% in the estimated standard deviation of the retention times of the blocks in the SSD.

1 Introduction

Flash memory based solid-state drives (SSD) have gained tremendous popularity in recent years. SSDs are widely used in a variety of computing devices, from phones and tablets to desktops and servers. SSDs offer several advantages over hard disk drives (HDDs), including higher performance, lower power, improved acoustics, and ruggedness. Despite these positives, a major concern with SSDs is that the underlying flash memory technology has a limited lifetime, which affects both the number of writes that can be reliably done to flash, referred to as endurance and quantified in terms of program/erase (P/E) cycles, and the retention time of stored data. The retention time is period of time during which data written to a flash memory cell can be read reliably [1]. The retention time decreases with the number of P/E cycles. These reliability concerns are paramount in data centers, where workloads are I/O-intense, data integrity requirements are high, and SSD replacement costs can be significant [2].

Accurate estimation of SSD reliability is important in data centers for several reasons. One of the key factors that determines the total cost of ownership (TCO) of a data center is the cost of the IT equipment. Typically, hardware refresh cycles span a period of 3-5 years during which the computing equipment is expected to operate reliably. Having an accurate means of estimating SSD reliability for the workloads to be hosted in a data center is essential for calculating TCO. Because the data retention time decreases with cycling, knowledge of how the retention time of an SSD changes overtime allows the storage system designer to make informed choices about how best to use the SSD over its lifetime – as long-term storage or as a cache for short-term storage.

There has been prior work on reducing the number of P/E cycles to ensure high endurance and long data retention times [3][4][5]. While reducing cycling is beneficial, it has been shown that merely counting the number of P/E cycles is insufficient to accurately model endurance and retention time accurately [1][6][7]. Both endurance and retention time are determined by stresses (P/E cycles) to the flash memory cells as well as a recovery process wherein the cell has the ability to partially heal itself during idle times between stresses. Several recent publications have proposed reliability and performance enhancements that leverage this observation [8][9][10][11].

While SSD performance can be evaluated using relatively short workload traces, accurate reliability assessments require capturing the impact of the workload access patterns on flash memory during an extended period, typically on the order of years of simulated time. This is because the reliability of flash memory cells depends on the *history* of stresses and recovery over time. Existing SSD simulators focus on performance modeling and use relatively simplistic approaches to quantifying endurance, such as counting the number of P/E cycles [12][13]. Also, all of the prior works that attempt to quantify lifetime more accurately by using the physically realistic reliability models simulate a workload for only a very short interval of time or use simplistic extrapolation of the results from a short simulation to a longer duration [14][1]. Because reliability is affected by the interplay of the workload behavior, the flash-translation layer (FTL) algorithms for page mapping, wear-leveling, garbage collection, and the distribution of stress and recovery events, any simplistic extrapolation of a short-duration simulation over a longer timescale is inherently error-prone. On the other hand, simulating a workload over a long time-scale is time-consuming. Having the means to perform reliability assessments of SSDs for different workloads with low turnaround time can help reduce the costs of capacity planning for deploying the storage systems.

Table 1: Simulation time of enterprise workloads for 5 years

Workload	MSNFS	EXCHANGE	DAPPS	RADIUS
Simulation Time (hours)	185.93	127.51	86.52	49.31

Table 1 shows the simulation time of several data center workload traces [15] during a 5-year timescale in the DiskSim simulator [16] with the SSD extension [12]. These simulations were run on an 8-CPU quad-core 2.3 GHz Intel Xeon$^{\text{TM}}$ machine with 48 GB of RAM. We choose a 5-year timescale because it represents the typical hardware refresh interval in a data center [11]. We simulate 5 years worth of activity by repeatedly replaying the I/O traces, each of which capture one representative day's I/O traffic. As the table indicates, detailed simulation of years of workloads is very expensive in terms of time. In this paper, we develop a methodology for reducing this simulation time.

Accelerating simulation time has been studied in the field of computer architecture for processor simulation [17][18]. However, these prior techniques cannot be applied directly to SSD reliability simulation. Architecture performance metrics, such as IPC, depend more on the instantaneous behavior of a workload running on a given microarchitecture whereas SSD reliability metrics depend on the history of the utilization patterns. Moreover, as mentioned previously, retention time is a complex function of the workload, FTL, stress distributions, etc. As a result of these differences, direct application of techniques such as workload reduction or sampling can lead to large inaccuracies.

In this paper, we present an acceleration framework to accurately measure SSD reliability by performing sampling over time on carefully trimmed workloads. Our methodology is generic and can be used for any workload or SSD architecture, thereby allowing for flexible design-space exploration studies. This paper makes the following specific contributions:

- We study the history patterns of retention time and characterize its behavior on an SSD during its entire lifetime. We show that data retention time changes rapidly early in the lifetime and tends to stabilize afterward, changing very slowly.
- We exploit this trend in retention time variation by developing a sampling-based approach to accelerate the simulation.
- We characterize the spatial and temporal stress characteristics of the workloads and identify further opportunities to decrease simulation time by reducing the size of the workload to be simulated. We develop and evaluate a clustering-based approach to trim down the workload size. Overall, we find that using sampling and clustering provides a 12X speed-up on average compared to detailed simulation, with errors in the estimated mean and standard deviation of 3.21% and 6.42%, respectively.

In the rest of this paper, Section 2 provides an overview of SSDs and flash memory reliability and Section 3 describes the experimental setup. Section 4 characterizes the variation in the retention time during a multi-year timescale and motivates the sampling-based acceleration approach. Section 5 describes our simulation acceleration methodology and Section 6 presents the experimental evaluation of the accerlation and accuracy of our approach. Section 7 concludes the paper.

2 Background of SSD Architecture and Flash Reliability

A typical SSD consists of a host interface logic (e.g., PCI Express, SATA), a flash memory controller, an internal buffer, and several flash memory packages that are connected to the controller via multiple channels [12]. A flash package is organized into one or more flash elements. Each element is composed of multiple planes, usually 4 or 8. Each plane contains a large number of blocks (e.g., 8K). Each block in turn consists of 64 or 128 flash pages that store data. An SSD has very different internal mechanisms compared to a HDD. To hide these differences from the upper layers of the system, SSDs implement a software layer called the flash-translation layer (FTL) to abstract the low-level implementation details [19].

Because NAND flash does not support efficient in-place writes, the FTL must maintain some mechanism that translates logical page addresses (LPA) to physical page addresses (PPA)[12]. In general, there are two mapping strategies: static mapping and dynamic mapping. Static mapping maps an LPA to a fixed PPA. Dynamic mapping does not pre-determine the mapping between LPA and PPA; instead, when handling a write request, the FTL computes the corresponding physical position in flash on the fly according to the wear-leveling logic that takes into account the current wear-out conditions across the flash memory packages. In practice, a typical hybrid FTL design often combines the two strategies by first mapping a portion of the LPA statically to a fixed pre-determined pool of flash memory, which is referred to as the allocation pool[12], and then mapping the non-static portion of LPA dynamically to some physical address inside this specific allocation pool. Therefore, for an incoming write request, its associated physical address is allocated from a fixed allocation pool, which can be as small as a flash plane or as large as several flash packages, based on the specific implementation. Because the hybrid mapping policy is used most widely in practice, our simulations use this policy.

In this paper, we focus on data retention time to characterize flash reliability. We use the model developed by Mohan et al. [11] to calculate the data retention time:

Data retention time, $t_{retention}$, can be calculated by the following equation:

$$t_{retention} = \frac{(Q_{th,spread} - C \cdot \delta V_{th})}{J_{SILC}} \tag{1}$$

where $Q_{th,spread}$ is the total charge stored in the FGT corresponding to a logical bit, C is the capacitance between the control gate and the floating gate of a FGT, and δV_{th} is the threshold voltage shift due to stress events. $C \cdot \delta V_{th}$ can be viewed as the amount of charge trapped in the insulators. J_{SILC} is the leakage current. Therefore, data retention time $t_{retention}$ is the amount of time taken for the charge to leak from the FGT. J_{SILC} can be approximated as a constant value over time [11]. As a consequence, the threshold

Table 2: Configuration of the simulated SSD

Elements per package	16
Planes per element	8
Blocks per plane	1024
Pages per block	128
Page size	4KB
Over-provisioning	10%
Cleaning threshold	5%

voltage shift δV_{th}, which determines the amount of charges trapped in the insulators, dominates the variation of retention time.

Mohan et al. [1] present a model to calculate the threshold voltage shift δV_{th}. According to their model, δV_{th} increases with charge-trapping in the insulators, which is affected by the accumulation of stress events and decreases with charge de-trapping which is caused by the recovery process. Therefore, δV_{th} can be expressed as follows:

$$\delta V_{th} = \Delta V_{th,s} - \Delta V_{th,r} \tag{2}$$

where $\Delta V_{th,s}$ and $\Delta V_{th,r}$ are the threshold voltage shifts due to charge trapping and detrapping respectively.

Their study also shows $\Delta V_{th,s}$ has a power-law relationship with the number of P/E cycles and $\Delta V_{th,r}$ is affected logarithmically by the length of the recovery time. δV_{th} is a monotonically increasing function. [1] and [11] include more details about flash memory reliability.

3 Experimental Setup

We carried out our simulations using Disksim[16] with the SSD extension module developed by Microsoft Research[12]. We modified Disksim to record statistics that impact reliability, such as the recovery time between successive stresses to flash, and augmented it with reliability models to calculate the retention time [1][11]. We simulate an enterprise-class 64GB 2-bit MLC SSD, whose characteristics are given in Table 2. The FTL of this SSD uses a greedy-based garbage collection approach with wear-leveling aware heuristics and cold data migration to distribute stress events evenly across all blocks [12]. The FTL uses a hybrid page-mapping policy that combines static and dynamic mapping mechanisms, with an allocation pool that operates at the granularity of a flash element. *Although we use Disksim to model a specific SSD in this paper, our statistical acceleration methodology is generic enough to be applied to other FTLs, SSD organizations, and potentially other simulators.*

To the best of our knowledge, there are no recent publicly available workload traces that span multiple years of activity. *Cello* [20] is the longest trace that we know of; it spans one year. However, it is relatively old and its I/O activity might not be representative of modern data center workloads. The workloads we choose in this paper consist

Table 3: Properties of enterprise workloads used for evaluation[11].

Workload	Trace Duration (hours)	Total I/Os (millions)	Write Traffic (GBs/day)	Request Inter-arrival average(ms)
Display Ads Platform Payload Server (DAPPS)	24	1.09	44.06	79.63
Exchange Server (Exchange)	24	5.50	75.86	15.79
MSN File Server (MSNFS)	6	2.22	83.21	9.78
Radius Authentication Server (RADIUS)	15	2.21	30.83	24.90

of four enterprise-class block I/O traces captured from modern Microsoft's data centers [15]. The key characteristics of these workloads are given in Table 3. Because each trace spans only between 6 hours and 1 day of representative I/O activity, we repeatedly replay each I/O trace until the simulation time reaches 5 years, which aligns with the typical hardware refresh period in a data center. This approach of replaying I/O traces for long timescales is also used by those in prior work on reliability simulation [1][8][11][14]. We believe that this approach is reasonable because prior research has shown that disk I/O traffic exhibits spatial and temporal self-similarity characteristics [21][15].

4 The Opportunity for Simulation Acceleration

In this section, we study history patterns of data retention time and present an analysis of how the distribution of retention time across flash memory blocks in an SSD vary with time. We use this analysis to formulate a sampling-based approach to decrease the simulation time.

The retention time of the flash memory cells is affected by the pattern of stresses and recovery periods over time. These patterns are governed by the workload, page-mapping, wear-leveling, and cleaning operations in the FTL. In our simulator, we track the retention times at the granularity of individual flash memory blocks. We simulate each of our workloads using the methodology described in Section 3 for a 5-year period and take snapshots of the entire system state, including all metrics related to retention time, every 15 days of simulated time. The histograms of the retention times over each 15-day period for each of the four workloads are shown in Figure 1. Each curve in the graph shows the histogram of the number of blocks in the SSD with a given retention time value at each 15-day interval. The x-axis in the graph is retention time. As noted in Section 1, retention time decreases as P/E cycles increase, so the curves on the right represent retention time histograms early in the lifetime of an SSD.

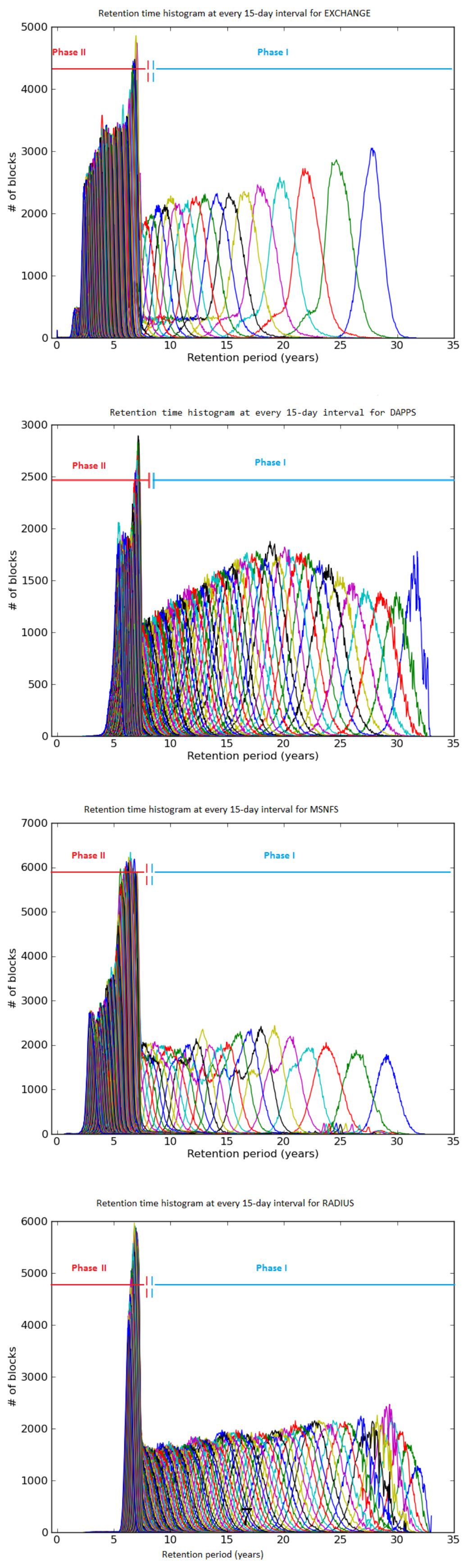

Figure 1: Retention Time Histograms during 5 Years of Simulated Time
Curves on the right represent retention time histograms early in the lifetime of an SSD.

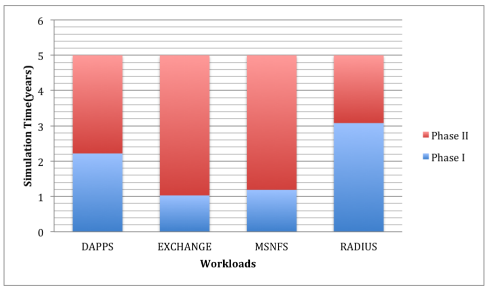

Figure 2: Ratio of Simulation Time between Phase I and Phase II of the Retention Time Variation

For all four workloads, the retention times of the blocks decrease rapidly early in the lifetime (denoted as *Phase I* in the figures) and then the retention times decrease at a much slower rate afterward (designated as *Phase II*). The *Phase I* and *Phase II* markers in Figure 1 are not the actual duration of these phases; rather, they show when the retention time distributions show a marked change in trends. The key reason for this trend is that $\delta V_{th,s}$ has a power-law relationship with the number of P/E cycles [1]:

$$\delta V_{th,s} = \frac{(A.cycle^{0.62} + B.cycle^{0.3}).q}{C_{ox}} \tag{3}$$

where A and B are constants, *cycle* is the number of P/E cycles, q is the charge of an electron, and C_{ox} is the oxide capacitance. Therefore, the rate of change of the retention time decreases with cycling. The ratio of the total simulated time in the two phases is shown in Figure 2.

The graphs highlight two key trends that directly influence the ability to accelerate the simulation:

- It is important to simulate in detail the activity during Phase I. Otherwise, even small inaccuracies in the reliability estimation potentially can accumulate into large errors in the result.
- The bulk of the simulation time is spent in Phase II, where potential opportunity exists to skip regions of the simulation and extrapolate the behavior without resulting in large errors.

The RADIUS workload exhibits a different trend; Phase I dominates the overall simulation time because this workload is not write-intense and therefore does not stress the flash memory cells to the same extent as the others and also requires far less simulation time, as shown in Table 1.

We use these two observations to develop a sampling-based approach to accelerate the simulation. This approach consists of two simulation modes: a detailed mode and a fast-forward functional mode in which the change in the reliability metrics are approximated. This bifurcated simulation approach is similar to those used for accelerating processor simulations [17][18]. Detailed simulation is performed only during certain intervals, which we refer to as sampling units, whereas functional simulation is performed between the sampling units.

5 Acceleration Methodology

This section explains how we decrease the time required for reliability simulations using a sampling-based approach, leveraging the insights from the Section 4 about how retention time varies over time. We first present an overview of our approach and then discuss each step of the simulation acceleration methodology in detail.

5.1 Overview of our Methodology

As mentioned in Section 4, our sampling-based simulation approach consists of two modes: a detailed mode and a fast-forward functional mode in which the change in the reliability metrics are approximated. To obtain high accuracy, sampling is performed only in Phase II of a simulation, where the changes in the retention time stabilize. To decrease simulation time further, we augment our sampling-based approach with a snapshot-based clustering technique to reduce the size of the workload so that only a subset of requests are simulated during the detailed simulation mode. We will show that our trimmed workload is representative of the original workload in terms of its stress behavior and impact on SSD reliability.

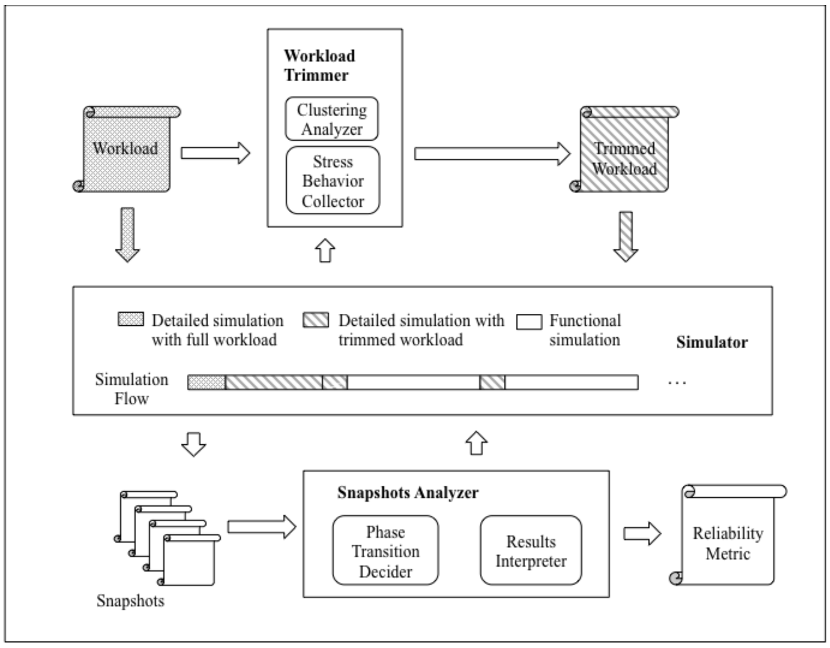

Figure 3: Overview of the Acceleration Framework

The overall simulation acceleration framework is shown in Figure 3. The framework consists of three major components: simulator, workload trimmer and snapshots analyzer. The simulator, integrated with reliability models, performs simulation in the detailed and fast-forward modes. It takes workloads as inputs and periodically dumps snapshots to track the reliability-related characteristics. The simulation flow is also depicted in Figure 3. The simulation begins with detailed mode on the full workload (the dotted region) with no acceleration techniques applied. The stress behavior collector in the workload trimmer collects information about the stress patterns of the full workload as the simulation runs. As soon as sufficient information is collected, the clustering analyzer in the workload trimmer is triggered to perform a clustering-based analysis on the information collected and reduce the size of the workload by selecting representative requests from the full workload. The rest of the simulation takes trimmed workload as inputs. During simulation, the simulator consults the phase-transition decider inside the snapshots analyzer periodically to see whether the simulation has entered Phase II and whether sampling can be performed. The snapshots analyzer is also responsible for interpreting the snapshots to report the reliability metrics. Except for the beginning of the simulation (dotted region), the bulk of the simulation flow alternates between detailed simulation on the trimmed workload (striped region) and the fast-forward functional simulation (blank region), which speeds up the original simulation.

We now discuss our approach in detail, including how we quantify the transition from Phase I to Phase II, how we perform functional simulation, and how our snapshot-based clustering algorithm works.

5.2 Quantifying the Phase Transition

To develop a sampling-based technique, we first require a way to quantify the distance between the retention time histograms. We use earth mover's distance (EMD) as the metric to quantify this, which gives a measure of distance between histograms. We choose EMD because it captures the shift along the x-axis of two histograms and their difference in shape. A precise mathematical definition of EMD between two histograms can be found in [22]. We calculate the EMD between every pair of adjacent data retention histograms for each workload, which gives a way to quantify the amount of variation in retention time distribution for every 15-day simulation interval. The EMD results for EXCHANGE are presented in Figure 4. We present only the EMD results for EXCHANGE here because other workloads share similar trends. The x-axis plots time in terms of the ID of the each 15-day snapshot and the EMD between the given snapshot and its following adjacent snapshot.

We can see that the EMD value is high early in the lifetime of the SSD and drops to a very low value later, which captures the two phases of behavior shown in Figure 1. Therefore, we can choose a threshold value of EMD, T_{EMD}, to distinguish between Phase I and Phase II in an automated manner and use this threshold to decide when to begin sampling the simulation. The choice of T_{EMD} value provides a tradeoff between simulation acceleration and accuracy.

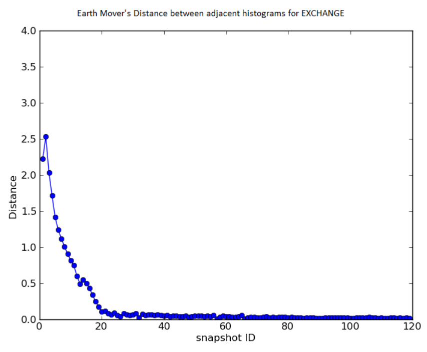

Figure 4: EMD between Two Adjacent Data Retention Histograms for EXCHANGE The K_{th} dot in the graph denotes EMD between the $(K-1)_{th}$ and the K_{th} Histograms

5.3 Sampling Strategy

Our acceleration framework consists of two simulation modes: a detailed simulation mode that simulates wear-leveling and garbage collection in the FTL, and a fast-forward mode that performs functional simulation. The detailed mode essentially runs full-fledged Disksim, whereas the fast-forward mode only updates flash reliability related statistics, such as the number of stress events and the timestamp of the last stress to a block, and does not change other FTL parameters such as the mapping between LPAs and PPAs over time. Compared to the detailed mode, the fast-forward mode attempts to estimate the stress behavior instead of simulating the complex FTL algorithm and requires far less simulation time.

Our sampling strategy works as follows:

- We first start with detailed simulation and take snapshots to record reliability related characteristics at the granularity of blocks at short time intervals during the course of simulation. We calculate the EMD between two adjacent data retention histograms from the information in the snapshots.

- If the EMD between two adjacent snapshots drops below a predetermined EMD threshold, T_{EMD}, we consider it to be the entry point into Phase II and sampling is performed. In Phase II, detailed simulation is performed only in chosen sampling units and fast-forward simulation is performed between the sampling units. We use a systematic sampling approach[17] in which simulation repeatedly alternates between d rounds of the detailed mode and f rounds of the fast-forward mode. In our experiments, we set the d to f ratio to be $1:10$.

5.3.1 Characterizing the detailed simulation mode behavior for fast-forwarding

While fast-forwarding can decrease simulation time, care needs to be taken to design it to minimize inaccuracies. In general, the fast-forward mode needs to consider two key aspects of the stress behavior: spatial behavior, which is the distribution of stress events across various blocks in the SSD, and temporal behavior, which is the distribution of recovery times corresponding to individual blocks. While prior work has also pointed out the need to model these two aspects [1], their approach has been to use simplistic extrapolation techniques to estimate reliability.

We sub-divide the SSD, which has M blocks, into m contiguous regions in which each region has $\frac{M}{m}$ blocks. We also sub-divide the simulation time into n intervals. We compute a stress distribution matrix, S_{mn} (size $m \times n$), of the spatial and temporal access patterns as follows. The i^{th} row of S_{mn} corresponds to the spatial memory region, which contains blocks numbered in the range $[\frac{M}{m} \cdot (i-1), \frac{M}{m} \cdot i)$. Each row in S_{mn} is a vector of length n, which we call the temporal vector, which characterizes the temporal stress distribution of the associated sub-portion of flash memory. Assume T is the simulation time duration which S_{mn} is collected and t corresponds to the start time of this duration; the entry $S_{i,j}$ in the stress distribution matrix is the number of stress events that fall into the sub-memory portion that contains blocks numbered from $[\frac{M}{m} \cdot (i-1), \frac{M}{m} \cdot i)$ within time interval $[t + \frac{T}{n} \cdot (j-1), t + \frac{T}{n} \cdot j)$.

In the detailed simulation mode, we construct the S_{mn} for the sampling unit by recording the stress into each memory region at each n time interval to capture the history of spatial and temporal access patterns over that sampling unit. In the fast-forward mode, instead of simulating the operation of the FTL algorithm with the workload, we use the stress distribution matrix from the immediately prior detailed-mode simulation interval to extrapolate/predict the workload's stress behavior. Inside each entry of S_{mn} (i.e., for the blocks within each flash sub-region), we assume an uniform distribution for the stress behavior, similar to [1]. In this way, the fast-forward mode estimates the reliability-related characteristics with information collected from the most recent detailed simulation. In our experiments, we take snapshots every 15 days and choose a T_{EMD} value of 0.1.

5.4 Using Clustering to Further Decrease Simulation Time

The systematic sampling technique discussed in the previous section can decrease simulation time by avoiding the execution of the entire second phase of a workload in the detailed mode. We now explore further opportunities to accelerate the simulation. We conduct a detailed analysis of the stress behavior of the workloads, in terms of both their spatial and temporal characteristics. We carry out this analysis by recording the stress distribution matrix of the detailed simulation of each day's worth of simulation. From these stress distribution matrices, we characterize the spatial stress behavior by generating a histogram of the number of stress events to various regions of flash memory in the SSD. The frequency of the i^{th} bin of the histogram, which corresponds to the number of stresses that fall into the i^{th} contiguous region of flash, is calculated by summing up all the stress numbers across the i^{th} row of the stress distribution matrix. Similarly, we characterize the temporal stress behavior by generating a histogram of the number of stress events to different time intervals within the simulation time during which the data is collected, where the frequency of each bin is calculated by summing up all the

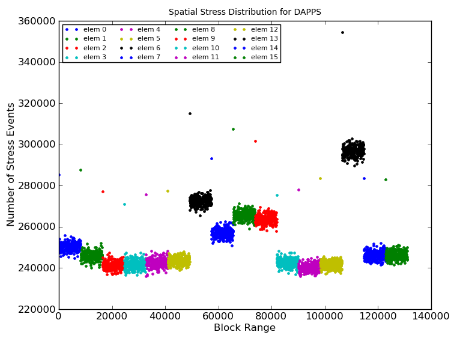

Figure 5: Spatial Stress Distribution of DAPPS after 15 Days of Simulation

stress numbers down the corresponding column in the Stress Distribution Matrix. We base our stress behavior analysis for each workload on the study of the corresponding 15-day stress distribution matrix generated by running the detailed simulation for 15 days and summing the stress distribution matrices for each trace-day. We choose to analyze the accumulated stress behavior of 15 days for two reasons: 15 days is the time duration between any two snapshots, and we find that the data collected for 15 days is sufficient to capture the overall stress patterns of the workloads. We present results for one workload - DAPPS. The trends for the other workloads are similar to DAPPS and we omit those graphs due to space reasons.

Figure 5 shows the spatial stress distribution during 15 days of simulation, where the x-axis is the block number in the SSD and the y-axis is the number of stress events. Each region of a specific color corresponds to the stress distribution for each flash element in the SSD. If we examine the spatial stress distribution for DAPPS in the graph, we observe similarities in stress behavior between certain elements. We can cluster the elements into five groups according to their similarity in stress behavior: Group 1 contains Elements 0-5, 10-12, 14, and 15; Group 2 contains Element 6; Group 3 contains Element 7; Group 4 contains Elements 8 and 9; and Group 5 contains Element 13.

If we look at the temporal stress distribution in Figure 6, we again observe similar clusters. The x-axis of Figure 6 is the sub-divided time intervals within the simulation time during which the workload is analyzed and the value in y-axis corresponds to the number of stress events that fall into the associated time domain. Each line in the graph corresponds to the temporal stress behavior of each element in the SSD. The temporal stress behavior of the workload again shows trends similar to the spatial stress patterns.

This analysis shows that elements in the SSD can be sub-divided into groups in which elements in each group share similar spatial and temporal stress behavior. Another important observation we make here is that at the element level, the short-term stress behavior in Figures 5 and 6 lines up closely with the long-term reliability behavior of SSD, which we show in Figure 7. Figure 7 shows the mean and standard deviation of the retention time on a per-flash element basis after 5 years of simulated time. We can see that Element 13 in DAPPS has a distinct reliability compared to other elements,

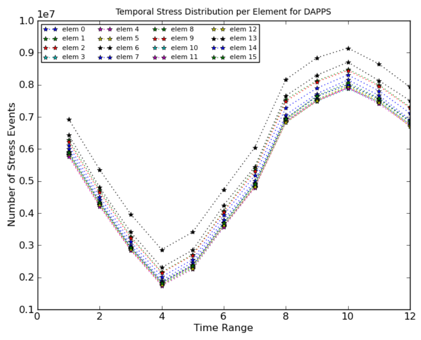

Figure 6: Temporal Stress Distribution of DAPPS after 15 Days of Simulation

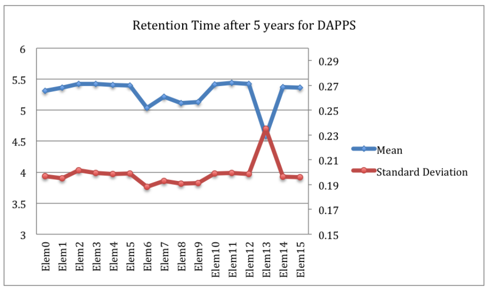

Figure 7: Mean and Standard Deviation of the Retention Time per Element after 5 years Simulation for DAPPS

which is quite similar to the 15-day trend.

The short-term behavior of the workloads correlates well with the longer term trend because the SSD uses a hybrid FTL in which an incoming write request from the workload is statically mapped to a specific allocation pool. Because the allocation pool is maintained at the granularity of flash elements and the longer timescale simulation is performed by repeatedly replaying the same trace, there is an inherent periodicity in the workload access patterns to each element. We exploit this observation to further accelerate the simulation by introducing a *snapshot-based clustering algorithm* to reduce the total workload size that is simulated. We can divide allocation pools into different clusters according to their stress behaviors during a short-term of detailed simulation and choose one allocation pool to be the *representative allocation pool* in each cluster. By simulating only requests that fall into the representative allocation pool, the workload is trimmed efficiently while still guaranteeing accuracy. We now describe our algorithm.

5.4.1 Snapshot-based clustering algorithm

Based on the preceding analysis, we develop the following workload trimming algorithm:

1. We run the detailed simulation for several trace days to collect the stress distribution matrix S_{mn}, which is a matrix of m temporal vectors of length n.

2. We apply the k-means clustering algorithm on the stress distribution matrix to classify the temporal vectors into $k_{optimal}$ clusters.
3. We extract a signature vector, Sig, for each allocation pool based on the clustering result of the second step. The signature vector provides a compact way to characterize the stress behavior of each allocation pool.
4. We run a hierarchical clustering algorithm [23] on the signature vectors of various allocation pools. This divides the allocation pools into several clusters. Within each cluster, allocation pools share similar stress behaviors. We choose the allocation pool with the smallest index value as the representative allocation pool.
5. The workload is trimmed by preserving only requests that fall in the representative Allocation Pool chosen in Step 4. The reliability metrics of the allocation pools that are not chosen are approximated with the result of the representative allocation pool in the associated cluster.

We now discuss each of these steps in detail.

Classifying temporal vectors using k-means: In order to classify the temporal vectors in the stress distribution matrix, we quantify the similarity and difference of two temporal vectors using the Manhattan distance. The Manhattan distance between temporal vectors p and q is defined as follows:

$$ManhattanDistance \quad = \sum_{i=1}^{n} |p_i - q_i|, \tag{4}$$

where n is the dimension of the temporal vector.

One problem that affects the quality of clustering of the k-means algorithm is the choice of the k value, which is the number of clusters. We apply the technique in [18] to calculate the optimal value of k valu $k_{optimal}$. In this approach, the k-means algorithm is tried with various k values, from 1 to the largest expected number of clusters (which is chosen to be 8 in our analysis, where 16 is the total number of elements in the SSD), resulting in a different clustering in each trial. To compare between different clustering approaches, the Bayesian information criterion (BIC) is used, which measures the "goodness of fit" of the clustering to the original data. The clustering with the smallest k value whose BIC exceeds 90% of the largest BIC value of all clusterings is chosen as the optimal clustering, and its associated k value is regarded as $k_{optimal}$. Our analysis shows that $k = 7$ is a suitable value for various workloads. Note that each temporal vector characterizes the temporal stress behavior for each region of flash memory. The clustering approach divides the memory regions into $k_{optimal}$ categories according to their temporal stress behavior.

Extracting signature vector for allocation pools: In Step 2, temporal vectors are grouped into $k_{optimal}$ clusters C_1, C_2, ..., $C_{k_{optimal}}$. We extract a signature vector, which is of length $k_{optimal}$, for each allocation pool based on the clustering result from the previous step. The signature vector for allocation pool r Sig_r is computed as follows. The i^{th} entry in Sig_r is the number of temporal vectors in cluster C_i whose associated memory region is contained in allocation pool r. A signature vector denotes how much a region of memory in the allocation pool exhibits certain types of temporal stress behavior. Therefore, a signature vector can be viewed as a compact way to characterize the spatial and temporal stress behavior of each allocation pool. The allocation pools with similar signature vectors tend to share similar stress behaviors.

Clustering allocation pools using a hierarchical clustering algorithm: Finally, we need a way to divide allocation pools into several clusters according to the similarity and difference of their signature vectors. Similar to temporal vectors, the distance between two signature vectors is quantified using Manhattan distance as the metric. We choose the agglomerative hierarchical clustering algorithm because it is flexible and produces good results. The hierarchical clustering algorithm works in a bottom-up manner, where each allocation pool starts in its own cluster and pairs of clusters are merged as we move up the hierarchy [23]. The merging terminates if the distance between all the stand-alone clusters exceeds a pre-determined threshold, which we set to be 20% of the maximum distance between two signature vectors in our analysis.

6 Evaluation of the Acceleration Framework

In this section, we evaluate the accuracy of our acceleration techniques and the speed-up achieved in the simulation time. We evaluate accuracy by comparing the histograms of the data retention times of the all the blocks in the SSD after a simulation period of 5 years for the accelerated variants to the base detailed simulation runs. The only exception is the EXCHANGE workload, which we simulate for only 3.5 years because this workload experiences block retention failures that exceed the capacity of the over-provisioning space of the SSD with the FTL algorithm we exploit. As mentioned earlier, we simulate the multi-year timescale by repeatedly playing back the workload trace. We call each such repetition simulation round.

Our acceleration framework allows the user to input five parameters: detailed simulation rounds (sampling unit size) d, functional simulation rounds f, earth mover distance threshold which denotes the workload phase transition T_{EMD}, size of stress distribution matrix m and n. As discussed in Section 5, the ratio of d to f provides a tradeoff between simulation speed and estimation accuracy. The choice of m and n determines how accurately a stress distribution matrix can capture the stress behavior in the detailed simulation mode. In our evaluation, we set T_{EMD} to be 0.1, $d : f$ ratio to be 1:10, and m and n to 131,072 (the total number of blocks in our simulated SSD) and 96, respectively. We now discuss how to choose the appropriate sampling unit size (d).

6.1 Choosing Appropriate Sampling Unit Size

In our sampling-based approach, detailed simulation is performed during sampling units, during which stress behavior of the given workload is captured using a stress distribution

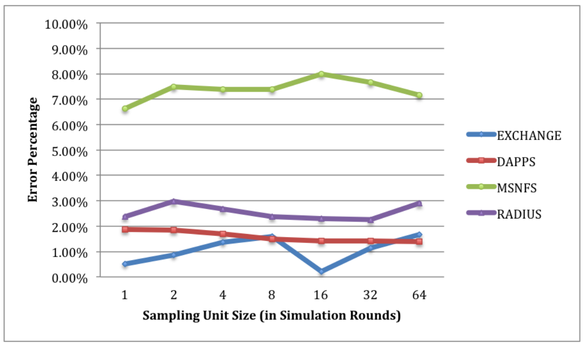

Figure 8: Error Percentage in Mean of Retention Times across Blocks after 5 Years of Simulated Time

matrix. In functional mode, stress behavior is approximated using the information collected from the sampling units. The sampling unit size can influence the simulation speed and accuracy. A small sampling unit size can reduce the required number of rounds of detailed-mode simulation. However, too small a sampling unit size might not be sufficient to capture the characteristics of the stress behavior and thus result in a larger estimation error during functional simulation. Figures 8 and 9 shows the error percentage with respect to the detailed simulation mode in the estimated retention times across all the blocks during 5 years of simulated time in terms of the mean and standard deviation. The analysis is based on the trimmed workload. We vary the sampling unit size from 1 to 64 simulation rounds. We can see that the choice of sampling unit size has greater impact on standard deviation than on mean. For standard deviation estimation, the error is high when sampling unit size is small, then drops and stabilizes afterwards. MSNFS is an exception; the error keeps falling and fails to stabilize within the range of sampling unit size in our analysis. After analyzing the spatial stress behavior of MSNFS at the flash-plane level for consecutive sampling units, we observe that the stress patterns between the two sampling units do not show any stable repetitive trend, whereas we do find a repetitive trend for the other workloads. This inherent complexity and instability in the stress behavior makes accurate approximation in the functional mode challenging for MSNFS at this sampling unit size. The downward slope of the standard deviation curve for MSNFS in Figure 9 suggests that it might possible to increase the accuracy with a larger sampling unit size. In the experiments that follow, we use $d = 8$, because the majority of the workloads stabilize at this value.

6.2 Evaluation of Accuracy and Speedup

Figure 10 compares the retention time distribution across blocks after 5 years of simulated time among the detailed-simulation mode, sampling-simulation mode on full workloads, sampling-simulation mode on trimmed workloads. We compare these to a simplistic extrapolation of retention time after detailed simulation phase (the approach used in [1]). Our simulation framework generates estimates much closer to detailed simulation

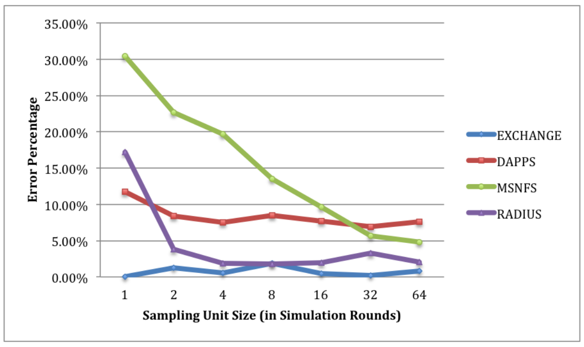

Figure 9: Error Percentage in Standard Deviation of Retention Times across the Blocks after 5 Years Simulation

compared to the simplistic extrapolation in terms of the position, shape and height of the retention time distribution. The histograms of the accelerated versions are very similar to the detailed simulation versions for DAPPS, EXCHANGE, and RADIUS. For MSNFS, the shape of the estimated distribution diverges with the original simulation due to the estimation bias in functional simulation, as already discussed. On average, our acceleration framework achieves a mean estimation error of 3.21% and a standard deviation estimation error of 6.42%.

Figure 11 shows the speed-up of sampling simulation mode on the full workload and trimmed workload with respect to the detailed simulation mode. Significant acceleration is achieved for all workloads except RADIUS. RADIUS is not write-intensive and does not incur as large a simulation time as the other workloads. With our sampling framework on the trimmed workload, an average of 12X speed-up is achieved.

7 Conclusion

Accurate estimation of SSD reliability is important for data centers. However, accurately measuring the reliability of SSD requires capturing the impact of workload access patterns on flash memory over timescales that span several years, which requires long simulation times. Simplistic extrapolation of reliability from short simulations is inherently error-prone and can lead to incorrect conclusions, and consequently, adversely affect system availability and TCO. To address this problem, we present a framework to accelerate SSD reliability simulation. This framework uses sampling and clustering techniques to significantly reduce the simulation overheads while still providing high accuracy.

8 Acknowlegement

We thank Sriram Sankar and Vidyabhushan Mohan for their valuable inputs. This research has been supported by NSF CAREER Award 0643925 and gifts from Google.

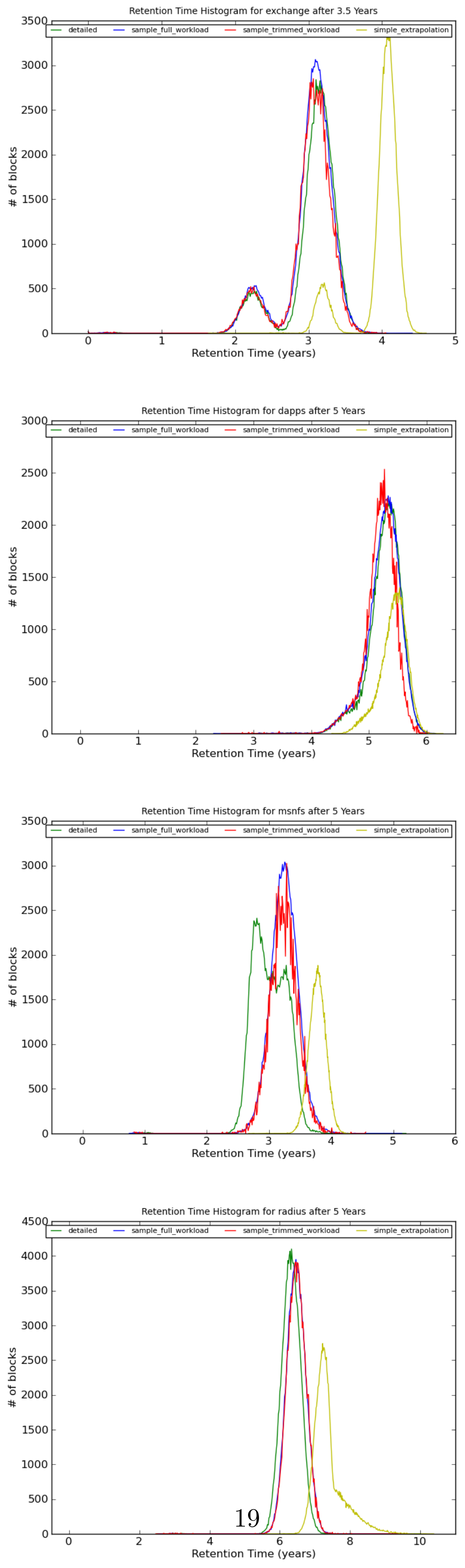

Figure 10: Comparison of Retention Time Histograms between Detailed-Simulation Mode, Sampling-Simulation Mode, Sampling-Simulation Mode on the Trimmed Workloads and Simplistic Extrapolation

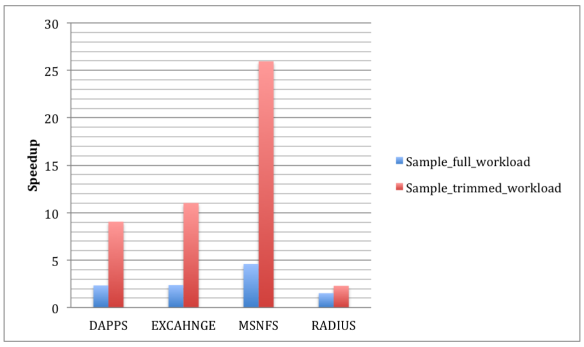

Figure 11: Speedup of Sampling-Simulation Mode and Sampling-Simulation Mode on Trimmed Workload Compared to Detailed-Simulation Mode

References

[1] V. Mohan, T. Siddiqua, S. Gurumurthi, and M. Stan, "How i learned to stop worrying and love flash endurance," in *Proceedings of the 2nd USENIX Conference on Hot Topics in Storage and File Systems.* USENIX Association, 2010, pp. 3–3.

[2] L. Barroso, "Warehouse-scale computing," in *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data.* ACM, 2010, p. 2.

[3] F. Chen, T. Luo, and X. Zhang, "Caftl: a content-aware flash translation layer enhancing the lifespan of flash memory based solid state drives," in *Proceedings of the 9th USENIX Conference on File and Storage Technologies.* USENIX Association, 2011, pp. 6–6.

[4] G. Soundararajan, V. Prabhakaran, M. Balakrishnan, and T. Wobber, "Extending ssd lifetimes with disk-based write caches," in *Proceedings of the 8th USENIX Conference on File and Storage Technologies.* USENIX Association, 2010, pp. 8–8.

[5] G. Sun, Y. Joo, Y. Chen, D. Niu, Y. Xie, Y. Chen, and H. Li, "A hybrid solid-state storage architecture for the performance, energy consumption, and lifetime improvement," in *High Performance Computer Architecture (HPCA), 2010 IEEE 16th International Symposium on.* IEEE, 2010, pp. 1–12.

[6] L. Grupp, A. Caulfield, J. Coburn, S. Swanson, E. Yaakobi, P. Siegel, and J. Wolf, "Characterizing flash memory: anomalies, observations, and applications," in *Microarchitecture, 2009. MICRO-42. 42nd Annual IEEE/ACM International Symposium on.* IEEE, 2009, pp. 24–33.

[7] S. Boboila and P. Desnoyers, "Write endurance in flash drives: measurements and analysis," in *Proceedings of the 8th USENIX Conference on File and Storage Technologies.* USENIX Association, 2010, pp. 9–9.

[8] S. Lee, T. Kim, K. Kim, and J. Kim, “Lifetime management of flash-based ssds using recovery-aware dynamic throttling,” in *Proc. of USENIX FAST*, 2012.

[9] Y. Pan, G. Dong, Q. Wu, and T. Zhang, “Quasi-nonvolatile ssd: trading flash memory nonvolatility to improve storage system performance for enterprise applications,” in *High Performance Computer Architecture (HPCA), 2012 IEEE 18th International Symposium on*. IEEE, 2012, pp. 1–10.

[10] Y. Pan, G. Dong, and T. Zhang, “Exploiting memory device wear-out dynamics to improve nand flash memory system performance,” in *Proceedings of the USENIX Conference on File and Storage Technologies*, 2011.

[11] V. Mohan, S. Sankar, and S. Gurumurthi, “Refresh ssds: enabling high endurance, low cost flash in datacenters,” University of Virginia, Tech. Rep. CS-2012-05, 2012.

[12] N. Agrawal, V. Prabhakaran, T. Wobber, J. Davis, M. Manasse, and R. Panigrahy, “Design tradeoffs for ssd performance,” in *USENIX 2008 Annual Technical Conference on Annual Technical Conference*, 2008, pp. 57–70.

[13] Y. Kim, B. Tauras, A. Gupta, and B. Urgaonkar, “Flashsim: a simulator for nand flash-based solid-state drives,” in *Advances in System Simulation, 2009. SIMUL’09. First International Conference on*. IEEE, 2009, pp. 125–131.

[14] Y. Cai, G. Yalcin, O. Mutlu, E. Haratsch, A. Cristal, O. Unsal, and K. Mai, “Flash correct-and-refresh: retention-aware error management for increased flash memory lifetime.”

[15] S. Kavalanekar, B. Worthington, Q. Zhang, and V. Sharda, “Characterization of storage workload traces from production windows servers,” in *Workload Characterization, 2008. IISWC 2008. IEEE International Symposium on*. IEEE, 2008, pp. 119–128.

[16] J. Bucy, J. Schindler, S. Schlosser, and G. Ganger, “The disksim simulation environment version 4.0 reference manual (cmu-pdl-08-101),” *Parallel Data Laboratory*, p. 26, 2008.

[17] R. Wunderlich, T. Wenisch, B. Falsafi, and J. Hoe, “Smarts: accelerating microarchitecture simulation via rigorous statistical sampling,” in *Computer Architecture, 2003. Proceedings. 30th Annual International Symposium on*. IEEE, 2003, pp. 84–95.

[18] T. Sherwood, E. Perelman, G. Hamerly, and B. Calder, “Automatically characterizing large scale program behavior,” in *ACM SIGARCH Computer Architecture News*, vol. 30, no. 5. ACM, 2002, pp. 45–57.

[19] C. Dirik and B. Jacob, “The performance of pc solid-state disks (ssds) as a function of bandwidth, concurrency, device architecture, and system organization,” in *ACM SIGARCH Computer Architecture News*, vol. 37, no. 3. ACM, 2009, pp. 279–289.

[20] C. Ruemmler and J. Wilkes, *UNIX disk access patterns.* Hewlett-Packard Laboratories, 1992.

[21] M. Gomez and V. Santonja, “Analysis of self-similarity in i/o workload using structural modeling,” in *Modeling, Analysis and Simulation of Computer and Telecommunication Systems, 1999. Proceedings. 7th International Symposium on.* IEEE, 1999, pp. 234–242.

[22] H. Ling and K. Okada, “An efficient earth mover’s distance algorithm for robust histogram comparison,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 29, no. 5, pp. 840–853, 2007.

[23] Wikipedia, “Hierarchical clustering — Wikipedia, the free encyclopedia,” 2012, [Online; accessed 20-January-2013]. [Online]. Available: http://en.wikipedia.org/wiki/Hierarchical_clustering