Accelerating Reliability Simulation of NAND-Flash Based Solid State Drives

A Thesis

Presented to

the faculty of the School of Engineering and Applied Science

University of Virginia

In Partial Fulfillment

of the requirements for the Degree

Master of Science

Computer Science

by

Luyao Jiang

May 2013

Approvals

This thesis is submitted in partial fulfillment of the requirements for the degree of

Master of Science

Computer Science

Luyao Jiang

Approved:

Sudhanva Gurumurthi (Advisor)

Mary Lou Soffa (Chair)

Mircea R. Stan

Accepted by the School of Engineering and Applied Science:

James H. Aylor (Dean)

May 2013

Abstract

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 impacted by the history of low-level stress and recovery patterns to Flash cells, which are determined by the workload characteristics, the time period over which the workload utilizes the SSD, and the FTL algorithms. Accurately assessing SSD reliability requires simulating the workload behavior over timescales that span several years, which is very time-consuming. This thesis 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 the latter stages. This allows for simulation acceleration to be focused on the latter stages without significant loss of accuracy. We show that our approach provides an average speedup of 12X over 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.

Acknowledgments

I would like to take this opportunity to thank many people without whom this thesis would not have been possible. First of all, I would like to thank my advisor Professor Sudhanva Gurumurthi, who is always so intelligent and helpful. When I first came to UVa to start my graduate career, this research area was a whole new world for me. Professor Gurumurthi was always so patient and supportive. He taught me not only knowledge in this research area, but also the way to conduct research and tackle problems which I believe would be beneficial throughout my career. He helped me identify and develop a strong interest in computer science research. Most importantly, his support and guidance helped me accumulate confidence along the way and realize what I am capable of. I would also like to thank my committee members, Professor Soffa and Professor Stan for taking the time to review my thesis and provide great suggestions.

I am fortunate to have a group of good friends around me. They are smart, pleasant, and inspiring. We went through a lot of tough times together. Their great company has made graduate life much more fun and enjoyable. I sincerely wish them luck and success in future career, whereever they may be.

Lastly, I want to thank my family without whom I cannot imagine what life would be like. Each time when I feel like I am about to lose hope, they are always there to support me. They are the reason that I can put myself together and keep on going. Without their support, I would have given up along the way. I wish them health and happiness all the time.

Contents

List of Figures

List of Tables

Chapter 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 the stored data. The retention time is period of time during which data written to a flash memory cell can be read reliably [18]. The retention time decreases with the number of Program/Erase (P/E) cycles. These reliability concerns are especially 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 over which time 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 important for calculating the TCO. Since 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 vs. 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 [7] [24] [25]. 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 [18] [11] [3]. Both endurance and retention time are determined by not only stresses (P/E cycles) to the flash memory cells but also 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 [15] [19] [20] [17]. More related works in this area are discussed in Chapter 2.

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 over an extended period of time, typically on the order of years of simulated time. The reason for this is because the reliability of the 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 [1] [14]. 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 of time [5] [18]. Since reliability is affected by the interplay of the workload behavior, the FTL algorithms for page mapping, wear-leveling, and garbage collection, 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 very 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 1.1 shows the simulation time of several data center workload traces [13] over a 5-year timescale in the DiskSim simulator [4] with the SSD extension [1]. 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 since it represents the typical hardware refresh interval in a data center [17]. We simulate 5 years worth of activity by repeatedly replaying the I/O traces, each of which capture one

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

Table 1.1: Simulation time of enterprise workloads for 5 years

representative day's I/O traffic. As the table indicates, detailed simulation of these workloads over a multi-year timescale is very expensive in terms of time. In this work, we develop a methodology for reducing this simulation time.

Accelerating simulation time has been studied in the field of computer architecture for processor simulation [30] [22]. However, these prior techniques cannot be directly applied 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, the 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.

The goal of this thesis is to present a methodology to accelerate reliability simulations of SSDs. We accelerate the simulation by performing sampling over time. Detailed simulation is only performed in sampling units, whereas fast forwarded simulation is performed between the sampling units. In order to further reduce simulation time, we augment our sampling-based approach with a clustering technique to reduce the size of the workload so that only a subset of requests are simulated during the detailed simulation mode. Our simulation framework is able to drastically accelerate the original simulation while still maintaining high accuracy. Moreover, our methodology is generic and can be used for any workload or SSD architecture thereby allowing for flexible design-space exploration studies.

This thesis makes the following specific contributions:

- We study the time-varying behavior of retention time and characterize its behavior of an SSD over the entire lifetime. We show that data retention time changes rapidly early in the lifetime and tends to stabilize afterward and change very slowly.

- We show how we can exploit this trend in retention time variation by developing a sampling-based approach to speedup the simulation.
- We characterize the spatial and temporal stress characteristics of the workloads and identify further opportunities to reduce the 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 the use of sampling and clustering provides a 12X speedup on average over detailed simulation with errors in the estimated mean and standard deviation to 3.21% and 6.42% respectively.

The rest of the thesis is organized as follows. Chapter 2 provides an overview of SSDs and flash memory reliability and dicusses the related work. Chapter 3 describes the experimental setup. Chapter 4 characterizes the variation in the retention time over a multi-year timescale which motivates the sampling-based acceleration approach and describes our simulation acceleration methodology. Chapter 5 presents the experimental evaluation of the speedup and accuracy of our approach. Chapter 6 concludes the work and discusses about the future works.

Chapter 2

Background and Related Work

2.1 SSD Basics

A typical SSD consists of a host interface logic (e.g., PCI Express, SATA), a flash memory controller, internal buffer, and several flash memory packages that are connected to the controller via multiple channels [1] [6]. The flash controller translates incoming requests and issues commands to the flash packages. The internal buffer holds pending data and metadata to/from flash. A flash package is organized into one or more elements. Each element comprises of multiple planes, usually 4 or 8 in number. 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. Inside each page, a small region is reserved for metadata, such as identification and error detection information.

The physical organization of an SSD is very different than that of a HDD. Since flash memory does not support in-place writes, a mapping is required between the logical block address to phisical media. Also SSDs need to move data aroound, so even data distribution inside SSD is required to prolong the life of the SSD. To address the above issues, SSDs implement a software layer called the Flash Translation Layer (FTL) to abstract the low-level implementation details [8]. The FTL is primarily responsible for maintaining an address mapping table that maps upper-level logical addresses to physical addresses in flash memory, performing garbage collection which cleans blocks for reuse, and conducting wear-leveling to ensure flash blocks are evenly stressed [23].

Since 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) [1]. In general, there are two mapping strategies: static mapping and dynamic mapping. Static mapping maps a 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 Allocation Pool [1], 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. Since the hybrid mapping policy is used most widely in practice, our simulations use this policy.

2.2 Flash Reliability

Flash memory supports three types of operations: writes (programs), erases and reads. Writes and erases are referred to as P/E cycles or stress events [17]. A NAND flash memory cell consists of a Floating Gate Transistor (FGT) whose threshold voltage can be programmed by tunneling some amount of charges into the floating gate and erased by tunneling charges out [19]. When data is programmed into an FGT, the insulators surrounding the floating gate prevent the charges from tunneling back out. This provides for non-volatility and thus the data stored in FGT can be sensed during a read operation. During P/E cycling, charges that tunnel through the oxide can get trapped in the insulator. When charges are trapped, it results in current leakage from the FGT, known as Stress Induced Leakage Current (SILC). After most charges inside the FGT have leaked through the insulator, read operations can no longer be guaranteed to return the correct data, thus rendering the FGT unusable. The estimated time for charges to leak through the insulator determines the amount of time data can be read reliably from a flash cell and is known as the data retention time. The endurance can be calculated by estimating the number of P/E cycles it takes to reach this reliability limit.

In this work, we focus on data retention time to characterize flash reliability. We use the model developed by Mohan et al. [17] to calculate the data retention time. We summarize the model below.

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

$$t_{retention} = \frac{(Q_{th,spread} - C \cdot \delta V_{th})}{J_{SILC}} \tag{2.1}$$

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 [17]. As consequence, the threshold voltage shift δV_{th}, which determines the amount of charges trapped in the insulators, dominates the variation of retention time.

Mohan et al. [18] 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:

$$\delta V_{th} = \Delta V_{th,s} - \Delta V_{th,r} \tag{2.2}$$

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. When δV_{th} gets sufficiently large, a read operation can no longer distinguish between different voltage levels and thus can no longer be reliably performed. The amount of time it takes to reach this point is the data retention time.

2.3 Related Work

There has been prior work on reducing the number of P/E cycles to ensure high endurance and long data retention times. CAFTL [7] explores an content-aware Flash Translation Layer(FTL) design that is able to remove duplicate writes and extend available free flash memory space. Griffin [24] improves lifetime of SSD by introducing a hybrid storage device that maintains a hard disk drive as write cache for the associated SSD and migrates cached data periodically. [25] proposes a hybrid architecture for the NAND flash storage that utilizes phase change random access memory to enhance performance, energy, as well as lifetime.

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 [18] [11] [3]. Mohan et al. [18] [17] have presented a model to characterize flash relianibility. Their model has indicated that both endurance and retention time are determined by not only stresses (P/E cycles) to the flash memory cells but also a recovery process wherein the cell has the ability to partially heal itself during idle times between stresses.

Despite reliability being one of the most important metrics to consider when using an SSD, there is a lack of tools that can provide accurate reliability estimates. Most of the current software simulators focus on performance rather than endurance [1] [14] [9]. SolidSim is a kernel-level simulator that is capable of providing insights for characterizing SSD endurance without the need of collecting workload traces [12]. Although SolidSim models the Flash Translation-Layer (FTL) in detail, the tool reports the endurance of the application relative to a sequential workload and does not directly quantify reliability. Moreover, all the above works simulate the given workload for a short time interval and fail to provide measurement over a timescale long enough for reliability issues to appear. Recent work [5] has tried to estimate long-term reliability via linear extrapolation. However, their approach overly-simplifies the relation of flash reliability over time. In this thesis, we evaluate one of the over simplistic extrapolation approaches and show that their results have a large divergence with the detailed simulation results.

Chapter 3

Experimental Setup

Our simulations are carried out using Disksim [4] with the SSD extension module developed in Microsoft Research [1]. We have 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 [18] [17]. We simulate an enterprise class 64GB 2-bit MLC SSD, whose characteristics are given in Table 3.1. The FTL of this SSD uses a greedy-based garbage collection approach with wear-leveling aware heuristics and cold data migration to evenly distribute stress events across all blocks [1]. 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. Note that although we model a specific SSD in this work, our statistical acceleration methodology is generic enough to be applied to other FTL or SSD organizations.

To the best of our knowledge, there are no recent publicly available workload traces that span

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%

Table 3.1: Configuration of the simulated SSD

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

Table 3.2: Properties of Enterprise Workloads used for evaluation [17].

multiple years of activity. Cello [21] is the longest trace that we know of which 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 work consist of four enterpriseclass block I/O traces captured from modern Microsofts data centers [13]. Each trace spans 6 hours to one day of representative I/O activity. The key characteristics of these workloads are given in Table 3.2. Since there are no publicly available workload traces that span multiple years of 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 similar to those in prior work on reliability simulation [18] [15] [17] [5]. We believe that this approach is reasonable since prior research has shown that disk I/O traffic exhibits spatial and temporal self-similarity characteristics [10] [13].

In this work, we use data retention time as the metric for reliability. Data retention time is the time during which data written to a flash memory cell can be read reliably. So larger data retention time indicates higher reliability. We evaluate the accuracy of the simulation by comparing the histograms of the data retention times of all the blocks in the SSD after a simulated period of 5 years for the accelerated variants to the detailed simulation. The speedup is measured by the ratio of detailed simulation time to accelerated simulation time.

Chapter 4

Acceleration Methodology

In this chaper, we first study how data retention time varies over time and present an analysis of how the distribution of retention time across flash memory blocks in an SSD vary with time. We then use this analysis to formulate a sampling-based approach to reduce the simulation time. We explain how we reduce the time required for reliability simulations using a sampling-based approach, leveraging the insights about how retention time varies over time. We present an overview of our approach and discuss each step of the simulation acceleration methodology in detail.

4.1 The Opportunity for Simulation Acceleration

The retention time of the flash memory cells are impacted 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 Chapter 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 4.1. 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. *Note that x-axis in the graph is retention time. As in Chapter 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.*

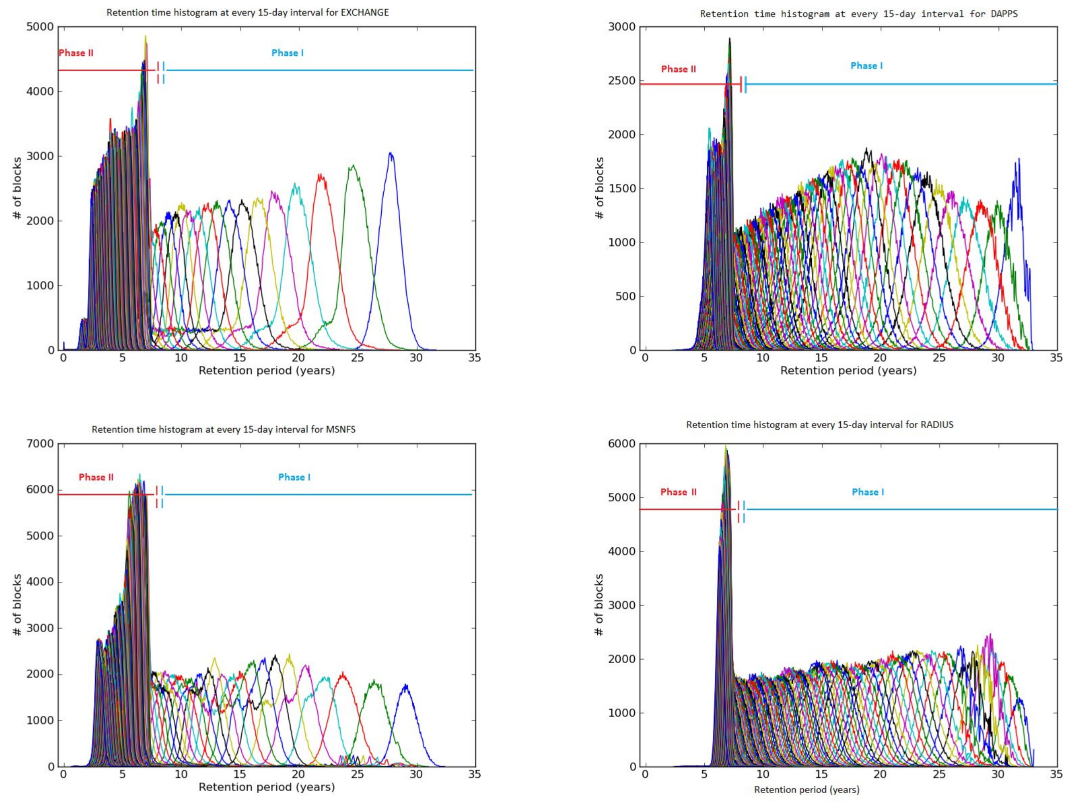


Figure 4.1: Retention Time Histograms over 5 Years of Simulated Time for EXCHANGE Curves on the right represent retention time histograms early in the lifetime of an SSD.

For all four workloads, we can see that the retention times of the blocks drop 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*). *Note that the Phase I and Phase II markers in Figure 4.1 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

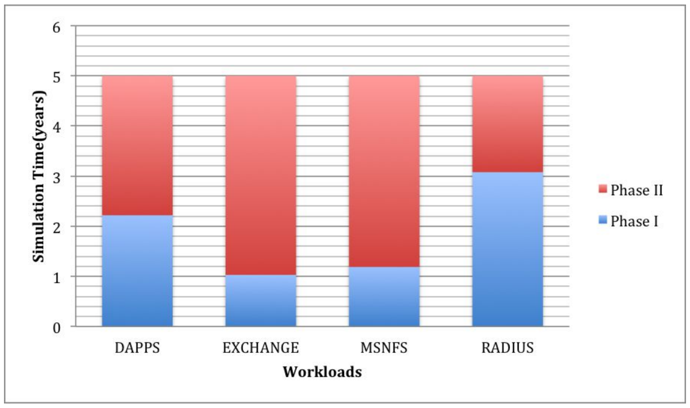


Figure 4.2: Ratio of Simulation Time between Phase I and Phase II of the Retention Time Variation

with the number of P/E cycles as [18]:

$$\delta V_{th,s} = \frac{(A.cycle^{0.62} + B.cycle^{0.3}).q}{C_{ox}} \tag{4.1}$$

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 4.2.

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 can potentially accumulate into large errors in the result.
- The bulk of the simulation time is spent in Phase II, where exists opportunity to potentially skip regions of the simulation and extrapolate the behavior without resulting in large errors.

The RADIUS workload exhibits a different trend, where Phase I dominates the overall simulation time. This is 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 1.1.

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 where the change in the reliability metrics are approximated. This bifurcated simulation approach is similar to those used for accelerating processor simulations [30] [22]. Detailed simulation is only performed during certain intervals, which we refer to as sampling units, whereas functional simulation is performed between the sampling units.

4.2 Overview of our Methodology

As mentioned in the previous section, our sampling-based simulation approach consists of two modes: a detailed mode and a fast-forward functional mode, where the change in the reliability metrics are approximated. To obtain high accuracy, sampling is only performed in Phase II of a simulation, where the changes in the retention time stabilizes. To further reduce simulation time, we augment our sampling-based approach with a 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 their stress behavior and impact on SSD reliability.

The overall simulation acceleration framework is shown in Figure 4.3. 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 keep track of the reliability related characteristics. The simulation flow in the simulator is also depicted in Figure 4.3. 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. 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. We can see that, 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 phase transition, how we perform functional simulation, and how our workload trimming algorithm works.

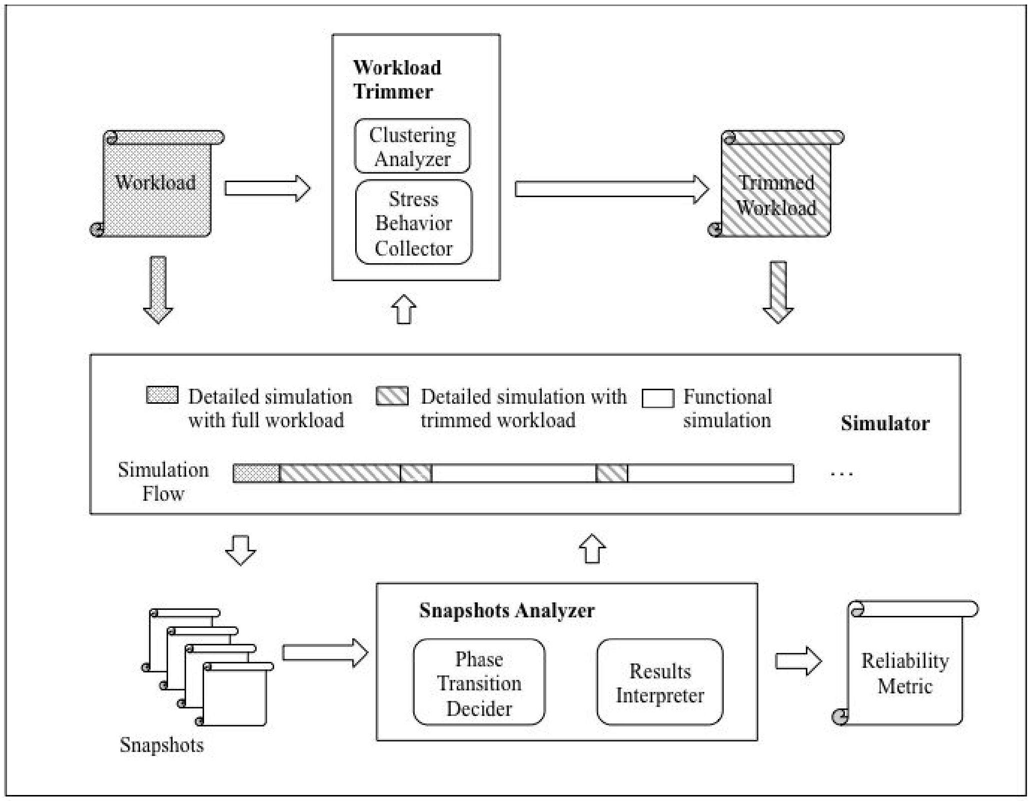


Figure 4.3: Overview of the Acceleration Framework

4.3 Quantifying the Phase Transition

In order 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. Informally, if two histograms are viewed as two different ways to pile up the same amount of dirt (the total number of blocks inside an SSD in this case), EMD computes the minimum cost of moving one pile into the other, where the cost is quantified as the amount of dirt moved times the distance by which it is moved [26]. EMD captures the shift along the x-axis of two histograms as well as the difference in shape of the distributions. A precise mathematical definition of EMD between two histograms is given in [16]. 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 the workloads are presented in Figure 4.4. The x-axis of each graph 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 down to a very low value later (except for RADIUS as explained previously), which captures the two phases of behavior shown in the previous chapter. 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 speedup and accuracy.

4.4 Sampling Strategy

Our acceleration framework consists of two simulation modes: (1) a detailed simulation mode which simulates wear-leveling and garbage collection in the FTL, and (2) a fast-forward mode which 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

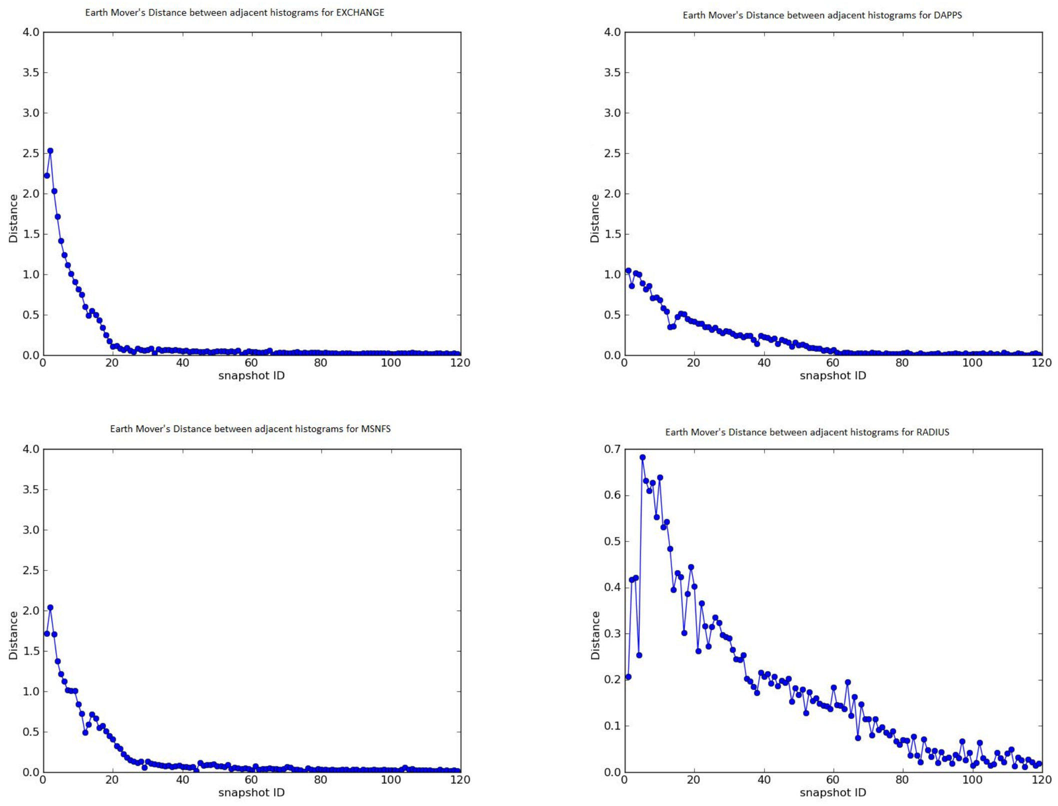


Figure 4.4: Earth Mover Distance between two adjacent Data Retention Histograms

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 pre-determined 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 only performed in chosen sampling units and fast-forward simulation is performed between the sampling units. We use a systematic sampling approach [30], where 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.

4.4.1 Characterizing the Detailed Simulation Mode Behavior for Fast-Forwarding

While fast-forwarding can reduce 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: (1) spatial behavior, which is the distribution of stress events across various blocks in the SSD, and (2) 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 [18], their approach has been to use simplistic extrapolation techniques to estimate reliability.

We subdivide the SSD, which has M blocks, into m contiguous regions where 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 which 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 [18]. 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 an T_{EMD} value of 0.1.

4.5 Using Clustering to Further Reduce Simulation Time

The systematic sampling technique discussed in the previous chapter can reduce simulation time by avoiding the execution of the entire second phase of a workload in the detailed mode. We now explore further opportunities to speed up the simulation. We conduct a detailed analysis of the stress behavior of the workloads, both in terms of 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 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 adding up Stress Distribution Matrices for each trace-day. We choose to analyze the accumulated stress behavior of 15 days for two reasons: (1) 15 days is the time duration any two snapshots, and (2) we find that the data collected for 15 days is sufficient to capture the overall stress patterns of the workloads.

Figure 4.5 shows the spatial stress distribution over 15 days of simulation of the workloads, 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 EXCHANGE in the graph, we can see that Elements

7 and 8 exhibit similar stress behavior but different from its other elements, all of which exhibit similar behavior. MSNFS tends to have a uniform stress distribution across all the elements. While DAPPS and RADIUS have more varied spatial stress behavior across the elements, we can still observe similarities in the stress behavior between certain elements. For example, in DAPPS, we can categorize elements exhibit similar stress distributions into 5 distinct groups: Group 1 contains Elements 0-5, 10-12, 14, 15; Group 2 contains Element 6; Group 3 contains Element 7; Group 4 contains Elements 8, 9; And Group 5 contains Element 13.

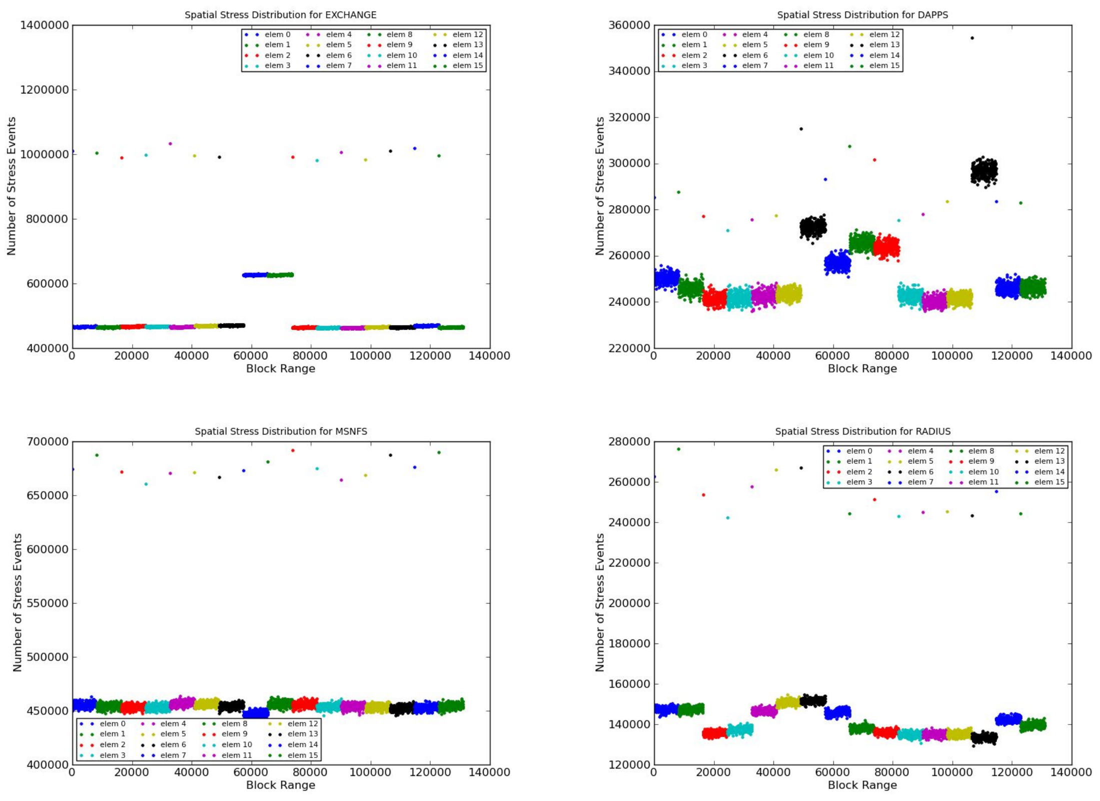


Figure 4.5: Spatial Stress Distributions of the Workloads

If we look at the temporal stress distribution for each workload in Figure 4.6, we can again observe similarities between the elements. The x-axis of Figure 4.6 is the sub-divided time interval within the simulation time 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.

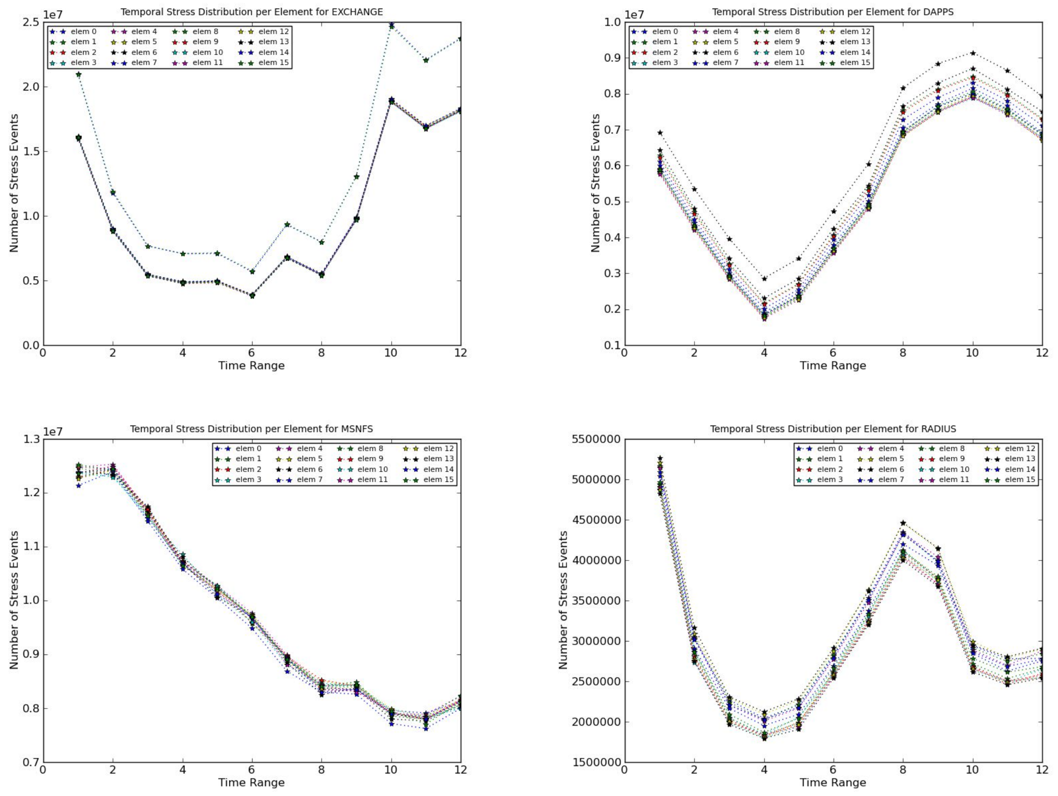


Figure 4.6: Temporal Stress Distributions of the Workloads

The temporal stress behavior of the workloads again shows trends similar to the spatial stress patterns. In EXCHANGE, Elements 7 and 8 share similar temporal stress behavior, which is different from the behavior of the other elements in the SSD. MSNFS exhibits a uniform temporal stress behavior across all the elements and in DAPPS and RADIUS, there are groups of elements that share similar temporal stress behavior.

From the above analysis, we can see elements in the SSD can be sub-divided into groups where elements in each group share similar spatial and temporal stress behavior to each other. Another important observation we make here is that at the element level, the short-term stress behavior in Figure 4.5 and Figure 4.6 lines up closely with the long-term reliability behavior of SSD, which we show in Figure 4.7. Figure 4.7 shows the mean and standard deviation of the retention time on a per flash element basis after 5 years simulation time. For example, if we look at the reliability behavior for EXCHANGE, we find that Elements 7 and 8 have similar mean and standard deviation of retention times, which are much lower than the values of the rest of the elements after 5 years simulation, which is similar to the 15-day trend. Similarly, MSNFS has uniform reliability behavior across various elements and Element 13 in DAPPS has a distinct reliability compared to the other elements.

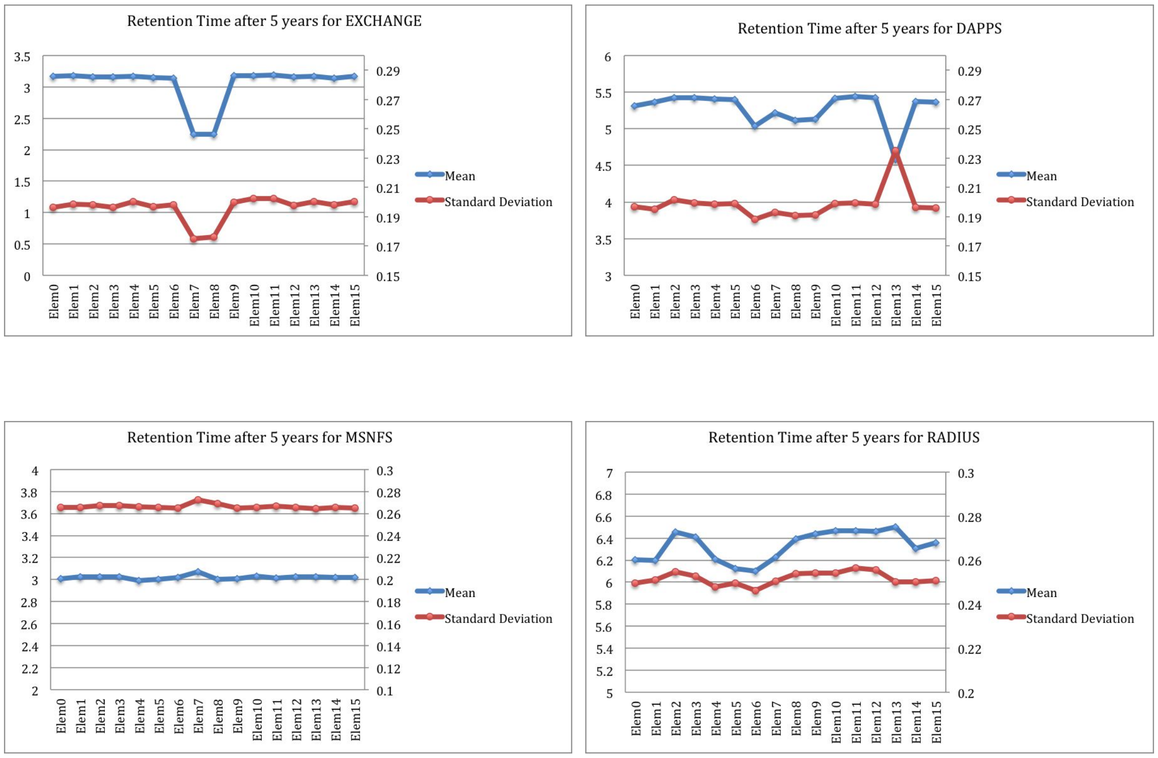


Figure 4.7: Mean and Standard Deviation of the Retention Time per Element after 5 years Simulation

The reason why the short-term behavior of the workloads correlates well with the longer term trend is as follows. The SSD uses a hybrid FTL where an incoming write request from the workload is statically mapped to a specific Allocation Pool. Since 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 the above observation to further speed up the simulation by reducing the total workload size that is simulated. We can divide Allocation Pools into different clusters according to their stress behavior over a short-term of detailed simulation and choose one Allocation Pool to be the *representative allocation pool* in each cluster. By only simulating requests that fall into the representative Allocation Pool, the workload is efficiently trimmed while still guaranteeing accuracy. We now describe our workload trimming algorithm.

4.5.1 Workload Trimming Algorithm

Based on the above 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 [28] 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 [27] on the Signature Vectors of various Allocation Pools, which divides Allocation Pools into several clusters. Within each cluster, Allocation Pools share similar stress behavior. We choose the Allocation Pool with the smallest index value as the representative Allocation Pool.

5. The workload is trimmed by only preserving requests that end 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 Clustering: K-Means algorithm is a clustering algorithm which partitions n observations into k clusters where each observation belongs to the cluster with the nearest mean [28]. The common approach is to start with k random observations as cluster centers and iteratively refine the choice of the center until convergence is reached [22]. During each iteration, we first assign the observation being clustered to the cluster with the shortest distance and then update the cluster center to be the centroid of all observations in the cluster taken into account the observation just added. The iteration reaches convergence until there is no membership change among the clusters.

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 [29]. Manhattan Distance between Temporal Vectors p and q is defined as follows:

$$ManhattanDistance = \sum_{i=1}^{n} |p_i - q_i|, \tag{4.2}$$

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 [22] to calculate the optimal value of k value - $k_{optimal}$. 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 exceed 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 temporal stress behavior of the memory regions into $k_{optimal}$ categories.

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 type 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 behavior to each other.

Clustering Allocation Pools using an 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 [27]. 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. Table 4.1 shows the clustering result for the Allocation Pools and the representative Allocation Pools chosen for various workloads.

Workload	Clustering Result
EXCHANGE	$\{\underline{0}, 1, 3, 4, 9, 10, 11, 12, 13, 15\}$
	$\{\underline{2}, 5, 6, 14\}$
	$\{\underline{7}, 8\}$
DAPPS	$\{\underline{0}, 1, 2, 3, 4, 5, 10, 11, 12, 14, 15\}$
	$\{\underline{6}\}$ $\{\underline{7}\}$ $\{\underline{8}, 9\}$ $\{\underline{13}\}$
MSNFS	$\{\underline{0}, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15\}$
RADIUS	$\{\underline{0}, 5, 7\}$ $\{\underline{1}\}$ $\{\underline{2}, 3, 8, 9, 10, 12, 13, 15\}$
	$\{\underline{4}\}$ $\{\underline{6}\}$ $\{\underline{11}\}$ $\{\underline{14}\}$

Table 4.1: Clustering Results. The Representative Allocation Pools are underlined.

Chapter 5

Evaluation of the Acceleration Framework

In this chapter, we evaluate the accuracy of our acceleration techniques and the speedup 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 case detailed simulation runs. The only exception is the EXCHANGE workload, which we simulate only for 3.5 years. This is 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 a *simulation round*.

Our acceleration framework allows the user to input 5 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 Chapter 4, 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 T_{EMD} to be 0.1, $d : f$ ratio to be 1:10, and m and n to 131072 (the total number of blocks in our simulated SSD) and 96 respectively. We now discuss how to choose the appropriate sampling unit size (d).

5.0.2 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 Matrix. In functional mode, stress behavior is approximated using the information collected from the sampling units. The sampling unit size can have an impact on the simulation speed and accuracy. A small sampling unit size can reduce the number of rounds of detailed-mode simulation required. However, too small of a sampling unit size might not be sufficient to capture the characteristics of the stress behavior and thus resulting in a larger estimation error during functional simulation. Figure 5.1 and Figure 5.2 shows the error percentage with respect to the detailed simulation mode in the estimated retention times across all the blocks after 5 years of simulation in terms of the mean and standard deviation. The analysis is based on the trimmed workload. We vary the sampling unit size from 1 through 64 simulation rounds. We can see that the choice of sampling unit size has more significant impact on standard deviation than the mean. For standard deviation estimation, the error is high when sampling unit size is small and drops down and stabilizes afterwards. MSNFS is an exception where the error keeps falling and fails to stabilize within the range of sampling unit size in our analysis. In order to understand why this happens, we analyze the spatial stress behavior of the workloads within a sampling unit. Figure 5.3 shows the spatial stress behavior of MSNFS at the flash plane level for two consecutive sampling units of size 16. We can 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 5.2 suggests that it might possible to increase the accuracy with a larger sampling unit size. In the experiments that follow, we use $d = 8$, since the majority of the workloads stabilize at this value.

spatial distribution for MSNFS

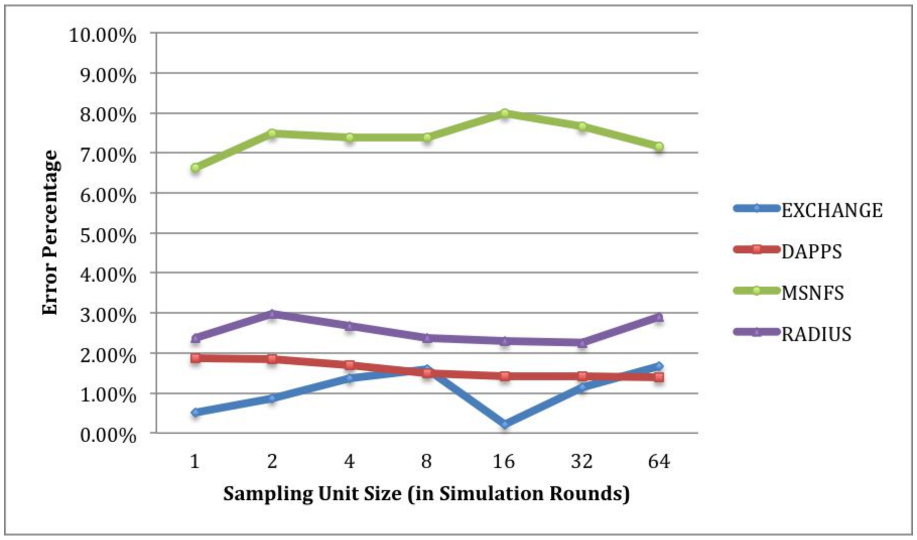


Figure 5.1: Error Percentage in Mean of Retention Times across the Blocks after 5 Years Simulation

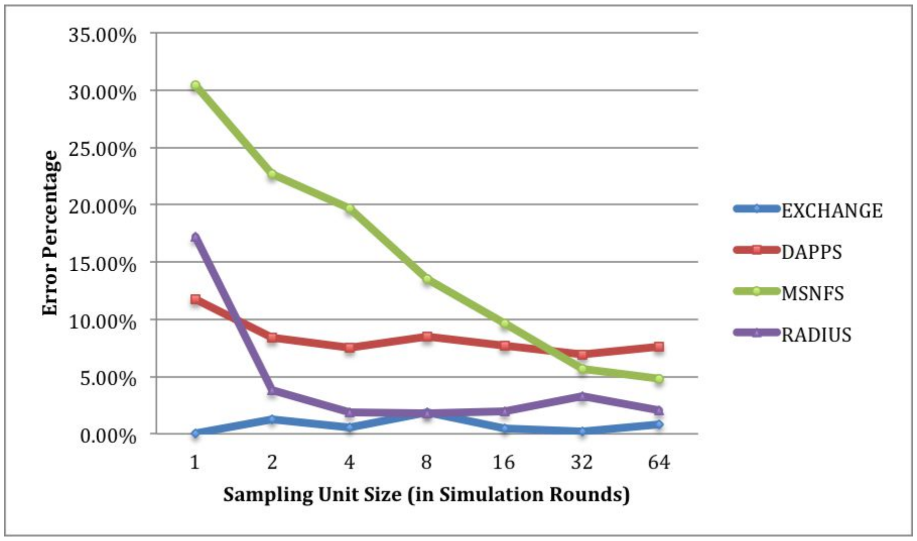
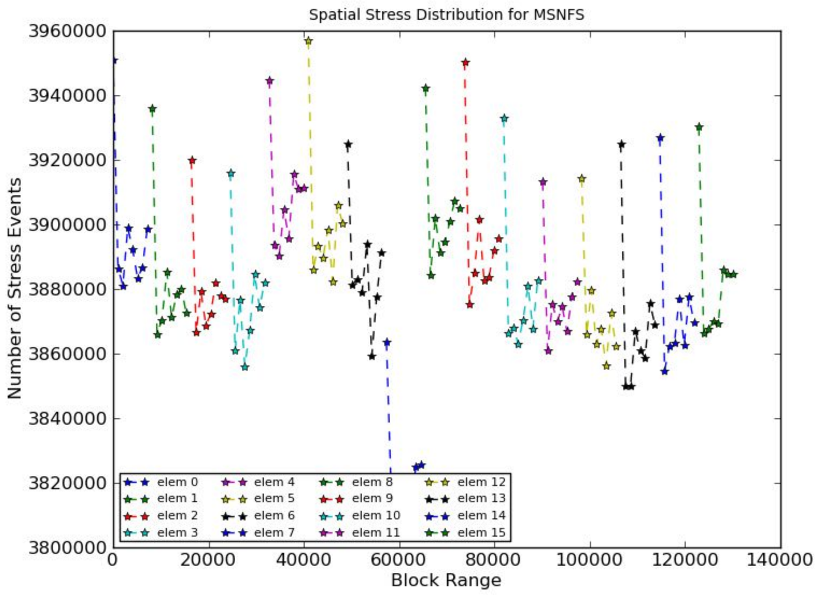


Figure 5.2: Error Percentage in Standard Deviation of Retention Times across the Blocks after 5 Years Simulation



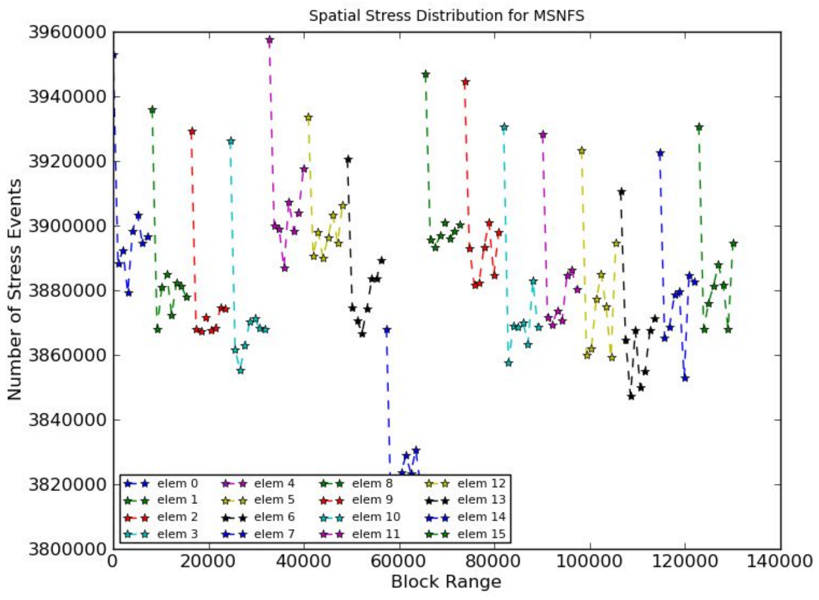


Figure 5.3: Spatial Stress Distribution for MSNFS in two Consecutive Sampling Units of Size 16

5.0.3 Evaluation of Accuracy and Speedup

Figure 5.4 compares the retention time distribution across blocks after 5 years of simulation time between 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 the detailed simulation phase, as is the approach used in prior work [18]. We can observe that our simulation framework generates much closer estimation to detailed 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 discussed above. On average, our acceleration framework achieves a mean estimation error of 3.21% and a standard deviation estimation error of 6.42%.

Figure 5.5 shows the speedup of sampling simulation mode on the full workload and trimmed workload with respective to the detailed simulation mode. Significant speedup is achieved for all the 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 speedup is achieved.

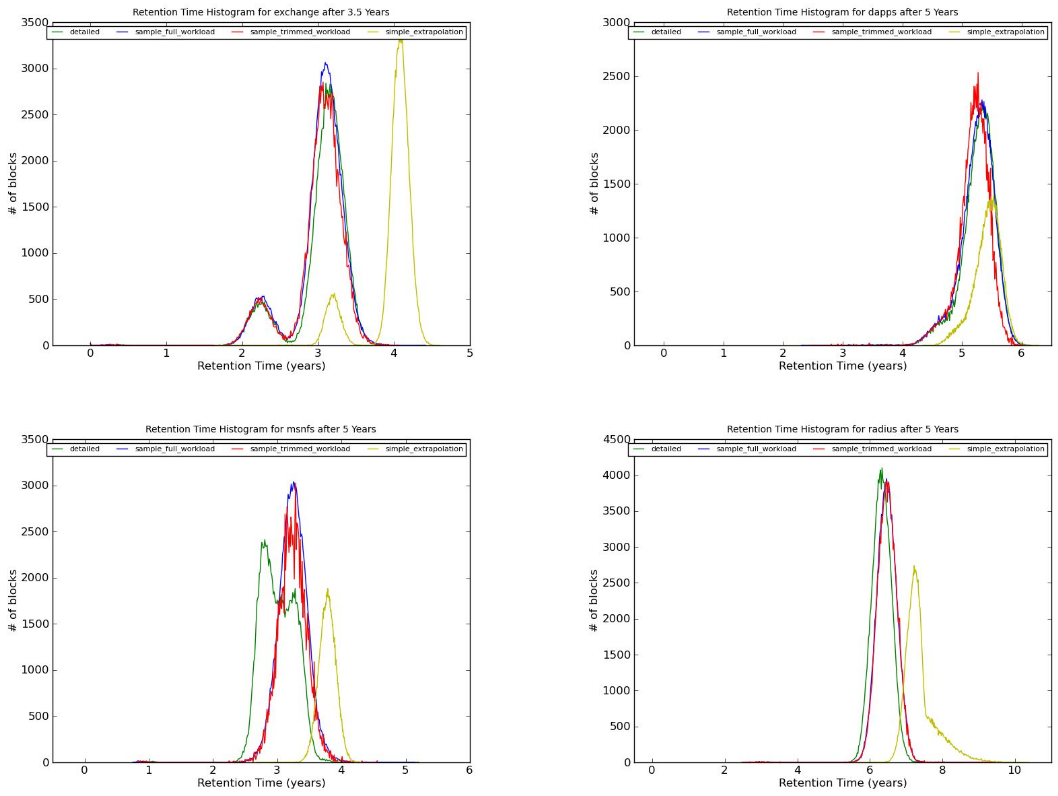


Figure 5.4: Comparison of Retention Time Histograms between Detailed Simulation Mode, Sampling Simulation Mode, Sampling Simulation Mode on the Trimmed Workloads and Simplistic Extrapolation

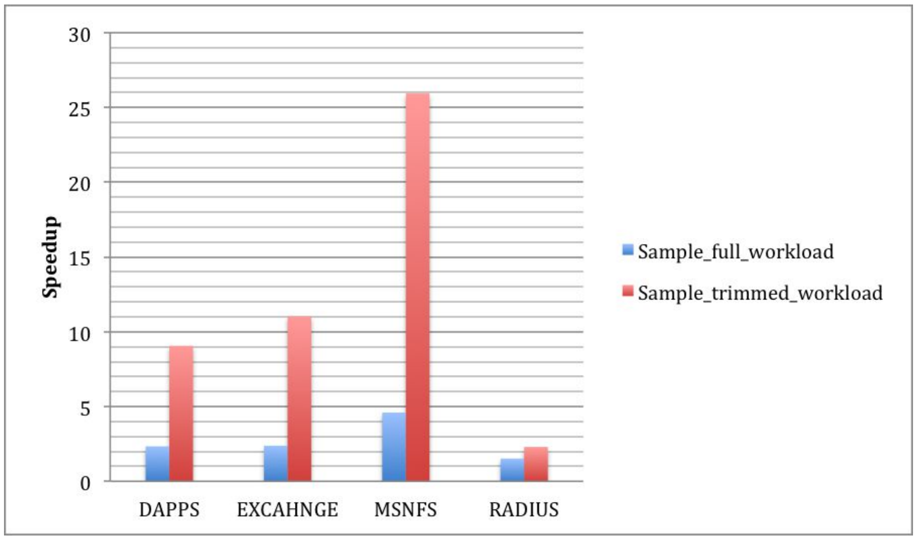


Figure 5.5: Speedup of Sampling Simulation Mode and Sampling Simulation Mode on Trimmed Workload Compared to Detailed Simulation Mode

Chapter 6

Conclusion and Future Work

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

While we believe that our framework is a good first step towards fast and accurate assessment of SSD reliability, there are several limitations of our current approach that we plan to address in future work:

- We model a workload's execution over a long timescale by repeatedly replaying shorter duration traces, leveraging the observations from prior work that disk I/O traffic tends to exhibit self-similar characteristics [10] [13]. We resort to this approach since there are no publicly available I/O traces that span large timescales. We believe that the growing attention being paid in the storage systems and architecture communities on modeling flash reliability will motivate capturing longer traces or traces that span various representative activities over long timescales (e.g., I/O traces from online retail outlets that capture activity during the holiday shopping season vs. other times). In these cases, one can still use our methodology for each such trace to construct the activity over the multi-year period.

- In our acceleration framework, we allow the user to input several parameters. The choice of parameters provides a tradeoff between simulation performance and accuracy and we currently choose these values statically which may not be optimal for all workloads. In future work, we plan to explore techniques that can automatically infer/tune the choice of parameters for each workload.

- An alternative, and possibly complementary, approach to speeding up the simulation is to construct a statistical black or gray-box model of the FTL so that one can infer the state of all the blocks of the SSD given a set of workload and FTL parameters. We plan to explore the development of such an approach in future work.

Bibliography

[1] N. Agrawal, V. Prabhakaran, T. Wobber, J.D. Davis, M. Manasse, and R. Panigrahy. Design tradeoffs for ssd performance. In *USENIX 2008 Annual Technical Conference on Annual Technical Conference*, pages 57–70, 2008.

[2] L.A. Barroso. Warehouse-scale computing. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, page 2. ACM, 2010.

[3] 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*, pages 9–9. USENIX Association, 2010.

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

[5] Y. Cai, G. Yalcin, O. Mutlu, E.F. Haratsch, A. Cristal, O.S. Unsal, and K. Mai. Flash correct-and-refresh: Retention-aware error management for increased flash memory lifetime.

[6] F. Chen, R. Lee, and X. Zhang. Essential roles of exploiting internal parallelism of flash memory based solid state drives in high-speed data processing. In *High Performance Computer Architecture (HPCA), 2011 IEEE 17th International Symposium on*, pages 266–277. IEEE, 2011.

[7] 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 stroage technologies*, pages 6–6. USENIX Association, 2011.

[8] 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*, volume 37, pages 279–289. ACM, 2009.

[9] K. El Maghraoui, G. Kandiraju, J. Jann, and P. Pattnaik. Modeling and simulating flash based solid-state disks for operating systems. In *Proceedings of the first joint WOSP/SIPEW international conference on Performance engineering*, pages 15–26. ACM, 2010.

[10] M.E. 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*, pages 234–242. IEEE, 1999.

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

[12] G. Kandiraju and K. El Maghraoui. A flexible os-based approach for characterizing solid-state disk endurance. In *Proceedings of the 9th conference on Computing Frontiers*, pages 223–232. ACM, 2012.

[13] 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*, pages 119–128. IEEE, 2008.

[14] 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*, pages 125–131. IEEE, 2009.

[15] 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.

[16] H. Ling and K. Okada. An efficient earth mover's distance algorithm for robust histogram comparison. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(5):840–853, 2007.

[17] V. Mohan, S. Sankar, and S. Gurumurthi. refresh ssds: Enabling high endurance, low cost flash in datacenters. Technical Report CS-2012-05, University of Virginia, 2012.

[18] V. Mohan, T. Siddiqua, S. Gurumurthi, and M.R. 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*, pages 3–3. USENIX Association, 2010.

[19] Y. Pan, G. Dong, Q. Wu, and T. Zhang. Quasi-nonvolatile ssd: Trading flash memory non-volatility to improve storage system performance for enterprise applications. In *High Performance Computer Architecture (HPCA), 2012 IEEE 18th International Symposium on*, pages 1–10. IEEE, 2012.

[20] 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.

[21] Chris Ruemmler and John Wilkes. *UNIX disk access patterns*. Hewlett-Packard Laboratories, 1992.

[22] T. Sherwood, E. Perelman, G. Hamerly, and B. Calder. Automatically characterizing large scale program behavior. In *ACM SIGARCH Computer Architecture News*, volume 30, pages 45–57. ACM, 2002.

[23] J.Y. Shin, Z.L. Xia, N.Y. Xu, R. Gao, X.F. Cai, S. Maeng, and F.H. Hsu. Ftl design exploration in reconfigurable high-performance ssd for server applications. In *Proceedings of the 23rd international conference on Supercomputing*, pages 338–349. ACM, 2009.

[24] 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*, pages 8–8. USENIX Association, 2010.

[25] 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*, pages 1–12. IEEE, 2010.

[26] Wikipedia. Earth mover distance — Wikipedia, the free encyclopedia, 2012. [Online; accessed 13-January-2013].

[27] Wikipedia. Hierarchical clustering — Wikipedia, the free encyclopedia, 2012. [Online; accessed 20-January-2013].

[28] Wikipedia. k-means clustering — Wikipedia, the free encyclopedia, 2012. [Online; accessed 13-January-2013].

[29] Wikipedia. Taxicab geometry — Wikipedia, the free encyclopedia, 2012. [Online; accessed 20-January-2013].

[30] R.E. Wunderlich, T.F. Wenisch, B. Falsafi, and J.C. Hoe. Smarts: Accelerating microarchitecture simulation via rigorous statistical sampling. In *Computer Architecture, 2003. Proceedings. 30th Annual International Symposium on*, pages 84–95. IEEE, 2003.