# reFresh SSDs: Enabling High Endurance, Low Cost Flash in Datacenters

Vidyabhushan Mohan[†] Sriram Sankar[‡] Sudhanva Gurumurthi[†]

[†]Department of Computer Science
University of Virginia
Charlottesville, VA 22904
{vm9u, gurumurthi}@virginia.edu

[‡] Microsoft Corporation
Redmond, WA
srsankar@microsoft.com

**Technical Report CS-2012-05**

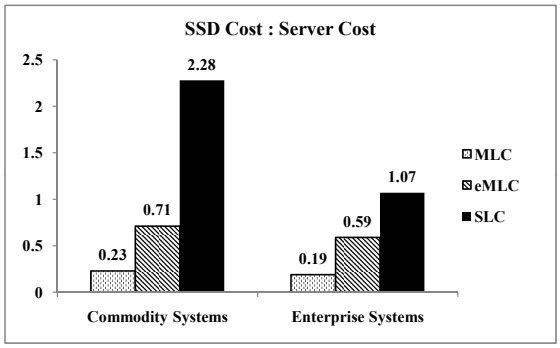
May 2012

**Abstract**

Storage performance and power are critical issues in modern datacenters. Solid State Drives (SSDs) offer both performance and power advantages over hard disk drives. With the advent of MLC flash, the cost-per-Gigabyte of Flash has dropped significantly enough to make it attractive for use in large-scale storage in datacenters. However, flash suffers from limited endurance and wears away after a certain number of writes and erases, thereby entailing higher replacement and service costs and hence an increase in the overall server infrastructure costs. In this paper, we develop a physically-accurate model of flash memory reliability and show that there exists a tradeoff between endurance and retention time. We then quantify this tradeoff and show how to exploit it without causing data integrity problems by *refreshing* the data in the flash memory cells. We propose and evaluate an Earliest-Deadline First (EDF) based refresh policy that can increase the endurance limit by 6-56% for an MLC-based SSD.

## 1 Introduction

As large enterprises move to the cloud, they host online services and store consumer data in large scale datacenters across the world. Moving consumer data to the cloud increases the demand for both storage capacity and performance. Storage accounts for a significant portion of the Total Cost of Ownership (TCO) and contributes upto 30% of the total power consumption in a datacenter [10]. This number is projected to become more significant given the uptrend in user data being stored on the cloud. IDC reports that user data stored in datacenters has been growing at an exponential rate [11]. Given the increasing demand, the cost of designing storage is expected to increase significantly. In order to keep pace with increasing data growth, enterprises are forced to look for new storage technologies that can achieve performance and capacity requirements at an optimal cost.




| | Commodity Systems | Enterprise Systems |
|---|---|---|
| CPU | Dual (8-16 cores) | Dual (8-24 cores) |
| Memory | DDR3 (24GB-32GB) | DDR3 (48GB-128GB) |
| Disk | Direct Attached SATA (4TB-8TB) | SAS (6TB-12TB) |
| SSD | 5% of capacity (Avg: 300GB) | 5% of capacity (Avg: 450GB) |
| System Cost (without SSD) | $2,000 - $3,000 | $6,000 - $10,000 |
| **SSD Actual Cost. Based on [2]** | | |
| MLC **Avg $/GB : 1.94** | $582 | $873 |
| eMLC **Avg $/GB : 5.91** | $1,773 | $2,660 |
| SLC **Avg $/GB : 19** | $5,700 | $8,550 |

Figure 1: Cost analysis of SSDs in Datacenters. Each bar represents the ratio of the cost of a SSD to the cost of the overall server solution without SSDs.

Solid State Drives (SSDs) have become increasingly relevant for datacenter storage since they achieve these objectives. The $/GB of flash based devices has been dropping steadily, making them attractive for use in large scale storage systems. SSDs offer several advantages including lower power and higher performance when compared to hard disk drives that were the traditional storage medium in large datacenters. Since enterprise workloads are mostly random in nature [35], they benefit significantly from the superlative random I/O performance of SSDs. Large datacenters hosting online services, such as Web Search, OLTP, and Cloud services employ SSDs as main memory extension buffers, Hard Disk Drive (HDD) caching solutions in the storage hierarchy, and sometimes as actual HDD replacements. Flash memory comes in three flavors - (i) Single Level Cell (SLC), (ii) Multi-Level Cell (MLC), where multiple bits are stored per memory cell, and (3) Enterprise MLC (eMLC). SLC provides the highest performance, but is the most expensive and least dense of all three types. MLC provides the lowest cost and highest density, albeit at a lower performance than SLC. However, MLC is still a strong performance value proposition compared to HDDs, since HDD performance is still significantly lower than MLC flash. eMLCs are MLC based devices with additional firmware and ECC geared towards enterprise class customers but at higher cost than MLC [31]. In a datacenter scale system, a key figure-of-merit is the the ratio of the cost of the SSD to the cost of the overall server solution without SSDs. It is important to minimize this ratio in order to be able to deploy SSDs at a large scale in a cost-effective manner. Figure 1 presents this ratio for the three types of flash SSDs for both commodity and enterprise class servers whose configurations are shown in the table. The cost prices for the servers and SSDs are obtained from list prices available online [2]. As we can see, among the three types of flash, MLC is clearly the best in terms of the SSD:Server cost ratio for both commodity and enterprise servers.

In this paper, we evaluate MLC based solutions, given its performance and cost benefits. However, the major impediment in adopting MLC SSDs at datacenter scale has been the endurance limitation. Enterprise storage has to support significant write traffic, when compared to client or desktop storage. This requirement directly impacts the endurance limits of flash technology. Flash memory blocks can wear out after a certain number of write (program) and erase operations, a property referred to as limited write endurance. Typical enterprise servers are designed to be cost- amortized for a period of 3-5 years based on the respective TCO model that an enterprise adopts. Based on this expected lifetime, replacement and service costs are projected

to be 15% of the overall server infrastructure cost [40]. Hence, increasing the cost of the server due to costly SSD replacements impacts the financial bottom line of a large enterprise. If the write endurance limit is reached before the expected TCO lifetime of a server, then this results in a significant cost to the enterprise. In a cloud environment, where efficiency of operation is key to revenue margins, failures due to write endurance limitations need to be avoided. Hence there is a need for a flash device that is cost optimal and provides all the benefits of flash technology, but at the same time, does not have the disadvantage of limited write endurance. While, among the three flash technologies, MLC has the lowest SSD:Server cost ratio, it is also the one that has the least amount of endurance. The key objective of this paper is to demonstrate how we can leverage MLC in datacenters while boosting its endurance.

Writes and erases to flash memory cells also has a detrimental impact on the retention time. The retention time is the metric that determines the non-volatility of flash. In this paper, we show that there exists a tradeoff between endurance and retention time, wherein a flash memory cell can undergo a larger number of programs and erases if one is willing to sacrifice some retention time. To ensure that sacrificing data retention does not violate the integrity of the stored data, we propose that the flash memory cells be periodically *refreshed* and we evaluate the performance and reliability impact of implementing an Earliest Deadline First (EDF) based refresh policy within the SSD for a set of datacenter workloads.

To summarize, our paper makes the following major contributions:

- We develop an analytical model based on actual physical parameters that captures the effect of cycling and recovery on the data retention property of NAND flash memory.
- We use the analytical model to quantify the tradeoffs between endurance and data retention for data-center workloads.
- We propose and evaluate an EDF-based refresh policy and show that it increases the endurance limits for an MLC-based SSD by 6-56% for our workloads.

The rest of the paper is organized as follows: Section 2 provides an overview of flash reliability. Section 3 explains the analytical model for flash reliability. Section 4 describes the experimental setup while Section 5 explains the benefits of the EDF based refresh policy on the reliability of MLC based SSDs.

## 2 Overview of Flash Reliability

Compared to a Hard Disk Drive (HDD), which uses rotating magnetic media to store data, flash uses transistors as the storage medium. It it important to understand the physical characteristics of flash before we delve into a discussion on its reliability. In this section, we provide an overview of flash memory architecture and its operation and then discuss the various reliability issues that affect flash memory.

A flash memory chip supports three operations: read, program (write) and erase. In addition to the flash memory array, a flash memory chip has many other components, including control and status registers, decoders, buffers, and address and data buses. An external memory controller sends a read, program, or erase command to the flash chip along with the physical address associated with that operation. Reads and programs are performed at a "page" granularity ($4-8$KB), while an erase is performed at a "block" granularity ($64-256$ pages). NAND flash does not support in-place writes and requires that a page be in the erased state before it can be written to. The Flash Translation Layer (FTL) inside the SSD manages the mapping from the logical block address, which is exported at the storage interface, to the actual physical mapping of that address on flash. The FTL also attempts to level the wear on all the pages and performs garbage collection on the superseded pages on flash that exist as a result of the out-of-place writes.

A flash memory cell consists of a Floating Gate Transistor (FGT). An FGT is similar to a NMOS transistor, but in addition to the control gate, there is another gate called the floating gate between the control gate and the channel. The floating gate is surrounded by dielectric on all sides. A write operation involves tunneling charges into the floating gate while an erase involves tunneling charges out of the floating gate. NAND flash uses Fowler-Nordheim (FN) tunneling [23] to tunnel electrons into and out of the floating gate. Programming increases the threshold voltage of the FGT, which is sensed during a read operation. Conversely, erasing reduces the threshold voltage of the cell. The extent of threshold voltage shift due to a program operation is used to encode data. The number of threshold voltage levels to which an FGT can be programmed determines the number of bits that can be stored inside an FGT. When data is programmed into an FGT and no external electric field is applied, the insulators surrounding the floating gate prevent the charges inside the floating gate from tunneling out, thus providing non-volatility.

## 2.1 Flash Failure Mechanisms

The main types of reliability issues in NAND flash memory are: data retention, write endurance, and disturbs. The duration of time for which the data written in flash memory cells can be read reliably is called the *(data) retention period*. The number of Program/Erase (P/E) cycles that can be performed on flash cells while guaranteeing a particular retention period determines the *(write) endurance*. While the retention period and endurance of flash can vary with usage, flash manufacturers guarantee a retention period in the order of years and an endurance of 3000 to 100,000 P/E cycles [16, 34]. Memory cell disturbs occur when a bit flips inadvertently in that cell due to accesses to adjacent cells. There can be read, write, or erase disturbs, depending on the operation in the adjacent cells. Many studies have shown that NAND flash memory has very low vulnerability to disturbs and that erasing the cell (which occurs several times as part of the normal usage of the SSD, since NAND flash does not support in-place writes) reduces the susceptibility of memory cells to bit flips [6, 39]. Moreover, existing Error Correction Codes (ECC) in flash memory are effective in handling errors that occur due to disturbs [6]. Therefore this paper focuses only on endurance and retention time and the tradeoffs between the two. Memory standards organizations, such as Joint Electron Device Engineering Council (JEDEC), provide detailed information about these these failure mechanisms [20].

## 2.2 Factors affecting Flash Reliability

While a comprehensive analysis of flash memory requires examining all the components in a flash chip, this work only focuses on the flash memory array, which occupies nearly 80% of the total chip area and is the most important component of a flash chip [4]. Some key factors affecting flash reliability are: (1) Cycling, (2) Recovery period, and (3) Temperature.

### 2.2.1 Cycling

FN tunneling requires very high electric fields which affects the reliability of the tunnel oxide in flash [23]. Consequently, program and erase operations, which use FN tunneling, are also referred to as stress events [28]. Cycling breaks the atomic bonds in the oxide layer, which increases the probability of charges getting trapped when they tunnel through the oxide layer. When charges are trapped in the tunnel oxide, it increases charge leakage from the floating gate due to a process called Trap Assisted Tunneling (TAT) [27]. This leakage current is called as Stress Induced Leakage Current (SILC) and is exacerbated due to trap assisted tunneling under low electric fields.

As cycling on the flash memory cell increases over a period of time, charge trapping in the tunnel oxide also increases, which increases SILC. As SILC increases, the data retention period and endurance of flash also decreases.

### 2.2.2 Recovery Period

Flash cells experience a period of relative inactivity before they are chosen to be programmed or erased again. This time period between successive stress events is called as the recovery period because flash memory cells experience some recovery from the effect of stresses during this time period [28]. Prior work [25, 39, 41] has shown that as the recovery period increases, some of the charges trapped in the tunnel oxide detrap and thereby increase the strength of the tunnel oxide. A recent study [28] provides a detailed analytical model that examines the effect of cycling and recovery periods on trapping and detrapping and their impact on flash endurance.

### 2.2.3 Temperature

High temperatures can exacerbate several silicon reliability problems and flash memory is no exception. The impact of temperature on the reliability of flash is typically expressed using the Arrhenius equation, where, given a failure rate at stress temperature $T_{stress}$, the failure rate at another temperature $T_{use}$ can be calculated by measuring the acceleration factor ($AF$), which is given by:

$$AccelerationFactor(AF) \quad = \exp^{\frac{E_a}{K}\cdot(\frac{1}{T_{use}}-\frac{1}{T_{stress}})} \tag{1}$$

where $E_a$ is the activation energy of a given type of failure mechanism which is usually determined based on empirical measurements, $K$ is the Boltzmann's constant, $T_{use}$ is the temperature in operating conditions (in kelvins) and $T_{stress}$ is the temperature under stress conditions (in kelvins). JEDEC indicates an activation energy of $1.1eV$ for detrapping, while the activation energy for the data retention failure mechanism is indicated to be $0.3eV$ [20]. As the activation energy increases, the sensitivity of the failure mechanism to temperature significantly increases.

In order to understand how the various factors mentioned in this section affect flash reliability, we now present an analytical model.

## 3 An Analytical model for NAND flash reliability

We estimate the retention period by modeling the SILC due to charge trapping and detrapping and use this estimate to calculate the duration of time before data loss occurs. By estimating the retention period of flash and the number of P/E cycles it takes for the retention to drop below a specific threshold, we estimate the endurance of flash memory. This analytical model is developed by combining information from device physics papers on NAND flash memory cells [22,25,26,41,42]. These papers provide information about the various parameters affecting endurance and retention, their relationship to each other, and values for some fitting constants that are used in the model. It is important to note that all these device physics papers are based on very similar flash memory technology and hence are consistent with each other.

In order to estimate SILC, data retention, and endurance, we need to first determine the effect of cycling and recovery on trapping and detrapping. We use the analytical model developed by Mohan et al. to capture this relationship [28]. They show that the threshold voltage shift due to trapped charges in tunnel oxide has

a power law relationship with cycling while recovery period has a logarithmic effect on detrapping. They also show the relationship between cycling and trapping to be:

$$\delta V_{th,s} = \frac{(A \cdot cycle^{0.62} + B \cdot cycle^{0.30}) \cdot q}{C_{ox}} \tag{2}$$

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 capacitance of the oxide. Mohan et al. show the relationship between detrapping and recovery period to be:

$$\delta V_{th,r} = \begin{cases} \delta V_{th,pr} & \text{if } \delta V_{th,pr} < \text{K} * \delta V_{th,s} \\ K * \delta V_{th,s} & \text{otherwise} \end{cases} \tag{3}$$

In the next section, we use Equations 2 and 3 to develop a reliability model to estimate the effect of trapping and detrapping on the data retention of flash.

### 3.1 Model for Data Retention

The retention model estimates the duration of time taken by the memory cell to leak the charges stored in the floating gate. To determine this time duration, the model estimates SILC based on the number of charges trapped in oxide layer which is a function of the total number of stress events and recovery periods.

According to de Blauwe et al., SILC ($J_{SILC}$) is a sum of two components, (a) a time-dependent transient component ($J_{tr}(t)$) and (b) a time-independent steady state component ($J_{ss}$) [17, 18]. We have,

$$J_{SILC} = J_{tr}(t) + J_{ss} \tag{4}$$

Moazzami et al. observed that for thinner tunnel oxides ($< 13nm$), the steady state component dominates the transient component [27]. As DiMaria et al. show this steady state component predominantly originates from a trap-assisted tunneling mechanism, where presence of interface and bulk traps increase the leakage current [9]. Hence, in order to obtain a first order reliability model of $J_{SILC}$, it is sufficient to model the time independent steady state component, $J_{ss}$. So, Equation (4) can be modified to

$$J_{SILC} \approx J_{ss} \tag{5}$$

Because the tunnel oxide thickness is smaller than 13nm for modern NAND flash generations [16], Equation (5) provides a good estimate of SILC. According to Larcher et al., the steady state component is modeled by using a Fowler-Nordheim (FN) like expression, as given below [22].

$$J_{SILC} = J_{ss} = A_{SILC} \cdot F_{OX}^2 \cdot \exp^{(} -\frac{B_{SILC}}{F_{OX}}) \tag{6}$$

$$A_{SILC} = C \cdot J_{STR}^{\beta} \cdot \exp^{(} -D \cdot Q_{INJ}^{\alpha}) \tag{7}$$

The barrier height used to calculate the exponential factor $B_{SILC}$ is between $0.8 - 1.1eV$. $F_{OX}$ is the electric field across the tunnel oxide during stress events and is considered to be $3.8MV/cm$. The values for constants $C$, $\beta$, $D$, $\alpha$ are available in [22]. The term $J_{STR}$ represents the current density across the tunnel oxide during stress events and is a function of applied program or erase voltage. We assume an operating voltage of 16V for program and erase operations based on data from the ITRS Roadmap [16]. $Q_{INJ}$ is the total amount of charge exchanged across the tunnel oxide and is a function of P/E cycles. Incorporating the

cycling term in $Q_{INJ}$ helps to calculate the leakage current as a device is cycled over a period of time. Based on [22], $Q_{INJ}$ can be defined as,

$$Q_{INJ} = \Delta Q_{INJ} \cdot N_C \tag{8}$$

$$\Delta Q_{INJ} = \Delta V_{th} \cdot C_{CG} \tag{9}$$

where $\Delta V_{th}$ is the threshold voltage difference between programmed and erased state,$N_C$ is the P/E cycle count, and $C_{CG}$ is the capacitance between the control gate and floating gate of a FGT.

Equation (6) represents the SILC due to the presence of trapped charges in the tunnel oxide. Because $J_{SILC}$ is dominated by $J_{ss}$ and $J_{ss}$ remains constant with time [18], the leakage current ($J_{SILC}$) can approximately be considered to be constant with time. Assuming that $\delta Q_{th}$ to be the total charge stored in the floating gate corresponding to a logical bit, $\delta V_{th}$ to be threshold voltage shift due to charge trapping (calculated from Equations 2 and 3), and $J_{SILC}$ to be the leakage current, we can calculate the time taken for the charges to leak from the floating gate ($t_{retention}$) to be,

$$t_{retention} = \frac{(\delta Q_{th} - (\delta V_{th} \cdot C_{CG}))}{J_{SILC}} \tag{10}$$

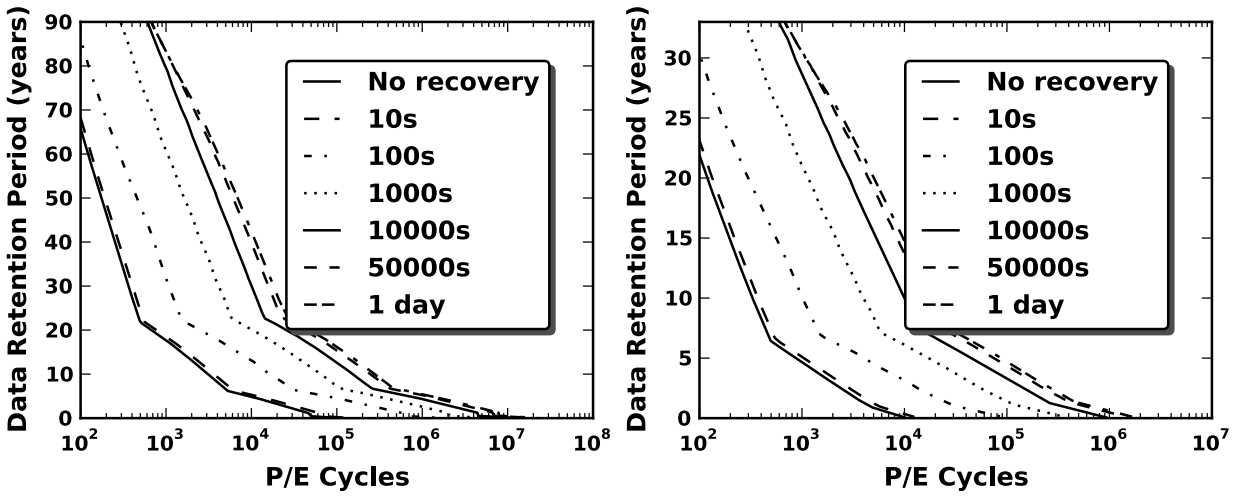
After $t_{retention}$ seconds, most of the charges from the floating gate would have leaked through the tunnel oxide and any read operation has a high probability of returning incorrect data. When this happens, we consider the memory cell to have reached its retention limit and the number of P/E cycles it takes for the cell to reach its retention limit is defined as the endurance limit of the cell. Since $\delta V_{th}$ and $J_{SILC}$ are functions of stress events and recovery period, Equation (10) provides an estimate of data retention period in NAND flash memory after taking stress and recovery into account.

### 3.2 Impact of Detrapping on Data Retention

Using the analytical model that we developed in the previous section, we analyze the impact of recovery and detrapping on data retention. We examine when data retention related failures occur and how recovery periods help in increasing data retention of flash. The results of this analysis are shown in Figure 2a for SLC and Figure 2b for 2-bit MLC flash. These results use a flash feature size of 80$nm$ because the cell-level reliability measurements are available only for this technology node. These estimates are based on an operating temperature of 30°C, the typical ambient temperature of the storage system in datacenters [32].

Typically, OEM datasheets specify a retention period of at least 10 years for NAND flash. This rating is usually very conservative and many supplementary documents provided by OEMs show that the typical retention period is around 100 years for NAND flash memory that has undergone very little cycling [29,30]. Our results show that the data retention period of SLC based flash is about 65 years (nearly 23 years for 2-bit MLC flash) for relatively small P/E cycle counts, which is similar to datasheet estimates.

From Figures 2a and 2b, we can observe that, when there is no recovery between successive cycles, the memory cell encounters retention failure when the total number of P/E cycles is around 100K for SLC flash and 10K for MLC flash, which concurs with the conservative values specified in the datasheets. However, as the recovery period between successive stress events increases, the number of times the cell can be cycled before retention failure occurs increases exponentially. Both SLC and 2-bit MLC flash experience a steep drop in their retention period when they are subjected to a few hundreds to thousands of cycles. For SLC flash, the retention period drops from about 65 years (for no recovery period) to about 20 years in a few hundred cycles, while in case of MLC flash, the retention period drops from nearly 23 years (for no recovery period) to around 5 years in a few thousand cycles. However, after this steep drop, the rate of decrease in the




(a) SLC flash

(b) 2-bit MLC flash

Figure 2: Impact of different recovery periods on the Data Retention for 80nm SLC and 2-bit MLC flash at 30°C

retention period slows down. As the memory cell is cycled, its retention period keeps dropping and when the memory cell is cycled beyond a certain threshold (the write endurance limit), the flash memory cell experiences retention failure, the point at at which the data stored in the memory cell is lost. From Figure 2a and Figure 2b, we can note that even though SLC flash and 2-bit MLC flash exhibit the same trend, the initial retention period of a new MLC flash memory is significantly lower than that of SLC memory. This is because, the $\Delta V_{th}$ for MLC flash is about 2.6 times lower than the $\Delta V_{th}$ for SLC flash.

Another distinct trend that we can observe is that the slope of the curves vary with the amount of cycling the cell has experienced. This behavior is in accordance with the study by Larcher et al. [22]. They show that as cycling increases, the dominance of parameters affecting leakage also varies and hence the rate of change of leakage also varies [22]. For P/E cycles less than $10^3$, the drop in threshold voltage is dominated by $A_{SILC}$, while for P/E cycles greater than $10^5$, the threshold voltage drop saturates because the oxide field across the tunnel oxide ($F_{ox}$) decreases and the leakage current also decreases. Between $10^3$ and $10^4$ cycles, both $A_{SILC}$ and $F_{ox}$ interact with each other creating a different slope. Because the slope of the curve changes, the rate of change in retention period also varies significantly with cycling.

Figures 2a and 2b show that there is clear tradeoff between the data retention and the endurance of flash memory. If the data retention for NAND flash can be relaxed linearly, the total number of P/E cycles possible can be increased exponentially. For example, in the case of 2-bit MLC flash having an average recovery period of about 10,000 seconds, the P/E cycle count can be increased from 10000 to 52,800 when the data retention period requirement is be reduced from 10 years to 5 years. While these results show the tradeoff between endurance and data retention for both SLC and MLC flash, the rest of this paper exploits this tradeoff for MLC SSDs for the reasons described in Section 1.

| Workload | Trace Duration (hours) | Total I/Os (millions) | Read & Write ratio | Request Inter-Arrival average(ms) |
|---|---|---|---|---|
| Display Ads Platform Payload Server (DAPPS) | 24 | 1.09 | 1:1.22 | 79.63 |
| Exchange Server (Exchange) | 24 | 5.50 | 1:2.22 | 15.79 |
| MSN File Server (MSNFS) | 6 | 2.22 | 1:1.24 | 9.78 |
| MSN Metadata Server (MSNCFS) | 6 | 0.52 | 1:0.64 | 41.63 |

Table 1: Properties of Enterprise Workloads used for evaluation. This data corresponds to the Logical Unit (LUN) with the highest write traffic.

# 4 Experimental Methodology

In this section we describe our experimental methodology for analyzing SSD performance and reliability.

## 4.1 SSD Configuration and Simulator Setup

As our objective is to evaluate the reliability of MLC in datacenters, we simulate a 64GB 2-bit MLC based SSD (M-SSD) similar to [15]. M-SSD uses a wear-aware cleaning algorithm and uses free blocks within a memory chip for wear leveling (allocation pool granularity of a chip) as mentioned in [1]. M-SSD has a spare area of 10% and also supports internal plane level copy back operations to leverage plane-level parallelism available inside SSDs. We use Disksim, a widely used trace driven simulator for studying storage systems, augmented with with an SSD model developed by Microsoft [1]. We incorporate our analytical reliability model developed in Section 3 into Disksim.

## 4.2 Workloads

Our workloads consist of block-level I/O traces collected from production systems in Microsoft that use HDD-based storage and are available publicly [35]. These workloads are summarized in Table 1. Each workload trace consists of millions of I/O requests corresponding to many hours of usage, span over several terabytes of I/O address range, and spread across many Logical Units (LUNs). As program and erase operations have the highest impact on flash reliability, we choose the LUN with the highest write traffic for our analysis.

## 4.3 Simulating SSDs with HDD Workload Traces

Since SSD performance is much higher than that of an HDD, a storage system where HDDs are replaced by SSDs will be expected to be more responsive and can absorb higher amounts of I/O traffic. Therefore, analyzing SSD behavior using HDD traces can lead to inaccurate assessments. There are two alternatives to address this problem. One approach is to use closed-loop simulation, where we recreate the entire computing infrastructure on which the traces were collected to assess the I/O behavior when the HDDs are replaced by SSDs. Another alternative approach is to use I/O traces that were collected on SSD-based storage. Unfortunately, neither option is possible for us since we do not have access to datacenter systems on which the traces were collected and there are no publicly available SSD I/O traces.

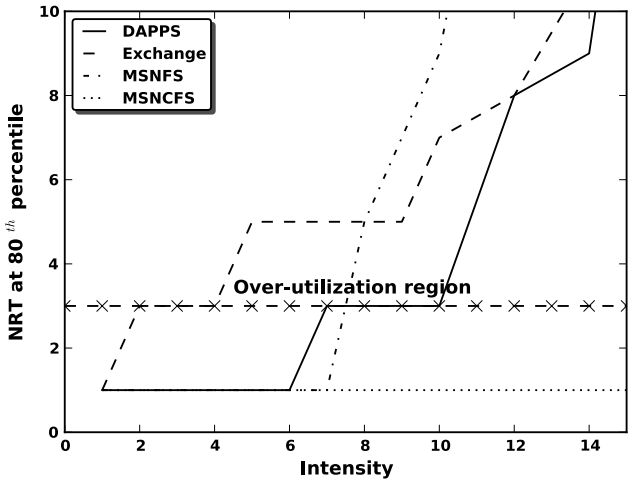



Figure 3: Impact of varying intensity on SSD response time (lower the better). An intensity factor is chosen such that the response time is below the over-utilization region.

In this paper, we approximate the behavior of the existing HDD traces on SSDs using the methodology proposed by Delimitrou et al. [7], who considered the problem of generating SSD traces from a HDD trace for datacenter workloads. They showed that the inter-arrival time between I/O requests is a key parameter that needs to be considered when generating an SSD trace. Since an SSD can handle much higher I/O loads, we shorten the inter-arrival times between the I/O requests, while preserving all the other properties of the workload trace, such as the spatial distribution, the ratio of reads and writes, and the distribution of request sizes. We use a scaling factor for the inter-arrival time such that the SSD can handle the increase in traffic without being over-utilized. To define over-utilization, we first define performance. Performance is defined as the response time of the SSD for the $80^{th}$ percentile of the I/O requests in the higher intensity workload, normalized to the response time of the $80^{th}$ percentile of the requests in the original HDD workload trace. We define over-utilization to be the region where the Normalized Response Time (NRT) at the $80^{th}$ percentile is greater than 3, which we find is the point when the bandwidth of the SSD has been exhausted and consequently the SSD begins to show a clear degradation in performance for our scaled workloads.

Figure 3 plots the variation in the NRT as the intensity of the workload is gradually increased. The area above the horizontal line in the Figure indicates the over-utilization region. From the Figure, we see that each workload reaches over-utilization at a different intensity. For Exchange, DAPPS and MSNFS, M-SSD reaches over-utilization when the intensity increases beyond 4, 10 and 7 respectively. Hence, for Exchange, DAPPS and MSNFS we choose acceleration factors of 4,10, and 7 respectively. For MSNCFS, the NRT does not change even when the intensity increases to 15. As shown in Table 1, the I/O intensity of MSNCFS is so low that even at 15X intensity, the response time at the 80th percentile does not change. Therefore, we choose an acceleration factor of 15 for MSNCFS. For the rest of this paper, we denote these modified higher intensity workloads as SSD-EXCH,SSD-DAPPS, SSD-MSNFS and SSD-MSNCFS while using them for analysis.

## 4.4 Estimating SSD Lifetime Over Long Timescales

Since SSD reliability issues manifest over a period of many years and TCO considerations also span 3-5 year time horizons, our analysis needs to span such time periods. In order to carry out such an analysis,

we can replay the workload traces, which span several hours to a day's worth of I/O activity, repeatedly over a multi-year timescale. However, running a simulation of an I/O intense workload over such a long time period requires several weeks of simulation time and is therefore prohibitively expensive. All our high-intensity workloads except SSD-MSNCFS are very I/O intense and therefore suffer from these excessively long simulation times. In order to tackle this problem, we use a statistical approach to reduce the simulation time for SSD-EXCH, SSD-DAPPS, and SSD-MSNFS, and perform a detailed simulation of SSD-MSNCFS. We now describe our statistical approach.

For each workload, we run a detailed simulation in Disksim using the baseline HDD workload trace as well as some high intensity versions of the trace over a multi-year timescale. We choose this timescale to be 5 years. After simulating every 15 days worth of I/O activity, a snapshot containing the lifetime of all the blocks in the SSD is obtained. Each snapshot contains the $\delta V_{th}$, the P/E cycles, and the retention period of every block in the SSD.

We found that over this 5-year time period, the retention period of blocks in the SSD are normally distributed for MSNFS and DAPPS. Exchange had bimodal normal distribution due to a skewed spatial distribution of requests in flash chips 7 and 8 and hence those chips were wearing out faster than the other chips in the SSD. However, most importantly, irrespective of the intensity and time at which the snapshot was obtained, the type of the distribution remained the same. Because the type of the distribution is found to be independent of intensity and time at which the snapshot is obtained, determining the lifetime of blocks at a higher intensity and at a particular time requires us to estimate the mean and standard deviation of block lifetime distributions for MSNFS and DAPPS, while for Exchange, we need to estimate two means and two standard deviations corresponding to the bimodal normal distribution. (Bimodal normal distribution has a $5^{th}$ parameter, $p$, that indicates the ratio of the area of the two normal distributions. We do not estimate this parameter, because $p$ remains constant across all simulations.)

For each workload, let $S_{t,i}$ denote the snapshot at time $t$ for intensity $i$. As snapshots are stored for every 15 days of simulated I/O activity, $1 <= t <= 120$ (snapshots are taken every 15 days for 60 months - a total of 120 snapshots). Because simulation time increases as intensity increases, we could not run detailed simulations with the same intensity for every workload. For Exchange, snapshots corresponding to detailed simulation were obtained for $i = 1$ and $i = 2$, while for DAPPS, MSNFS, snapshots were obtained for $i = 1$ to $i = 5$. In order to get the distribution of retention period for all blocks in the SSD for SSD-MSNFS ($i = 7$) and SSD-DAPPS ($i = 10$), we need to estimate two parameters, while for SSD-EXCH ($i = 4$), we need to estimate four parameters at various time intervals. For each parameter to be estimated, we use the following approach. We use regression on a subset of data from the snapshots to obtain a curve that fits the data. Since it is possible to fit the data with more than one curve, we choose the curve that has the least root mean square error. Each curve is of the form $z = f(t,i)$ where $z$ is the parameter to be estimated, $t$ corresponds to the snapshot time interval and $i$ is the intensity. For example, in order to estimate the mean lifetime of blocks for SSD-MSNFS ($i = 7$) at various time intervals, we use the mean lifetime of a subset of snapshots $S_{t,i}$, where $1 <= i <= 5$ and $1 <= t <= 48$ as the training data to obtain a curve which fits this data. The accuracy of the fitted curve is cross-validated by comparing the estimated parameter with the measured data which was not part of the training set. Table 2 shows the results of the cross-validation for two data points corresponding to $i = 1$ and $i = 2$ while maintaining $t$ at 72.

From Table 2, we can see that the estimated and the measured parameters are very similar to each other. Specifically, the means and standard deviations of the predicted and estimated parameters are almost the same for MSNFS and DAPPS. For Exchange the means and the standard deviations of the first mode differ by less than 0.5 years. However, for the second mode of the bimodal distribution, the difference in means and standard deviations is a little higher (1.59 to 2.79 years). It should be noted that the second mode

| Workload | Mean | | | Std deviation | | |
|---|---|---|---|---|---|---|
| | M | E | D | M | E | D |
| DAPPS ($i=1$) | 6.51 | 6.52 | 0.01 | 0.34 | 0.27 | 0.07 |
| DAPPS ($i=2$) | 4.49 | 4.52 | 0.03 | 0.33 | 0.23 | 0.10 |
| Exchange ($i=1$) | 2.91 | 3.37 | 0.46 | 0.33 | 0.80 | 0.47 |
| | 3.78 | 5.37 | 1.59 | 0.19 | 2.98 | 2.79 |
| Exchange ($i=2$) | 1.07 | 1.31 | 0.24 | 0.15 | 0.63 | 0.48 |
| | 1.73 | 3.54 | 1.81 | 0.13 | 1.93 | 1.8 |
| MSNFS ($i=1$) | 4.39 | 4.45 | 0.06 | 0.33 | 0.24 | 0.09 |
| MSNFS ($i=2$) | 2.31 | 2.33 | 0.02 | 0.31 | 0.23 | 0.08 |

Table 2: Cross validation results for extrapolated vs measured parameters. All units are in years."M"stands for measured parameter,"E" for estimated parameter, and "D"for absolute difference between measured and estimated parameter. Exchange has 2 means and 2 standard deviations because of bimodal normal distribution.

corresponds to data blocks with higher lifetime (their mean is higher than the mean of the first mode) and hence, the error introduced by the difference in estimation of the second mean would only impact blocks with higher lifetime and hence have less impact on the overall lifetime of the SSD.

We use the regression equations to estimate the age of all the blocks in the SSD for a given time and workload intensity. We then sample the multi-year timescale that we are interested in studying by choosing several points in time over this timescale. We then run a short duration (1 hour) detailed simulation in Disksim from each such instant to determine the impact of aging on performance for each of the workloads that use the extrapolation.

# 5 Impact of SSD Workloads on MLC Flash Reliability

In this section we analyze the impact of SSD enterprise workloads on the reliability of M-SSD. We begin by defining the metrics that we use for evaluation, then analyze the retention time vs. endurance characteristics for the workloads, and then evaluate the refresh policy.

## 5.1 Metrics

To quantify lifetime, we use the *virtual retention period* as the metric. The *virtual retention period* is defined as the minimum time duration for which any data written in the SSD can be read successfully. SSDs can have different virtual retention period requirements depending on the application for which they are used. If the SSDs cannot store data for the specified virtual retention period they are considered to be unreliable. Typical virtual retention period requirement for applications vary from 3 months to 1 year [19]. For the baseline analysis, we consider the SSD to be unusable if the *actual* data retention period of blocks drops below the *virtual* retention period. By using a range of virtual retention period thresholds (1 year, 6 months, 3 months and 1 month), we analyze the reliability of SSD over a variety of usage scenarios.

Since different blocks in the SSD can have different retention periods, we use the retention period at the $10^{th}$ percentile of all blocks as the criteria for our evaluation. We chose the $10^{th}$ percentile because, the over-provisioning capacity of M-SSD is 10% and the SSD is still functional as long as the amount of free space in the drive is more than 10%. Hence, we consider the SSD to be completely unusable when the data retention period of 10% of blocks falls below the virtual retention period threshold. As mentioned in Section 4, we use the normalized response time of the $80^{th}$ percentile of the total number of requests as the performance metric.

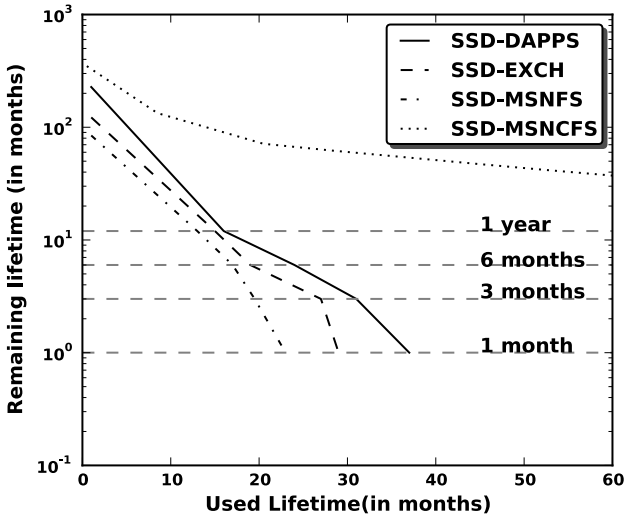



Figure 4: Impact of SSD workload on M-SSD Reliability. The horizontal lines correspond to different virtual retention time thresholds.

## 5.2 Baseline Evaluation

In Figure 4, we show the impact of SSD workloads on MLC SSD reliability. Our objective is to determine if our MLC SSD (M-SSD) is capable of servicing enterprise workloads over the service life. If not, we use our results to determine when the M-SSD needs to be replaced. The x-axis in the figure represents the period of simulation which extends upto 5 years (or 60 months) while the y-axis is the retention period at the $10^{th}$ percentile. The various dotted horizontal lines parallel to the x-axis indicate different virtual retention period thresholds. The curves in the Figure 4 are similar to the curves in Figure 2b, except that the y-axis is in log-scale. While the data in Figure 2b is for a single flash memory cell under controlled stress-recovery conditions, the data in Figure 4 also takes into account the workload behavior, SSD architecture, and the FTL. Hence, the points in the curve where their slopes change are different for different workloads. From Figure 4, we can see that SSD-EXCH, SSD-DAPPS and SSD-MSNFS have a significant impact on the reliability of M-SSD. For these workloads, the lifetime of baseline M-SSD crosses the virtual retention threshold of 1 year within 20 months of actual usage, thereby requiring multiple replacements within the TCO lifetime of the server (3-5 years). With lower virtual retention period thresholds, the usage time of the SSD increases, but this can still be improved further. Our objective is to delay the SSD replacements as long as possible, ideally beyond the TCO lifetime. The only exception to this trend is SSD-MSNCFS, whose impact on the lifetime of M-SSD is modest. Even after 5 years of use, the retention period of blocks in M-SSD is more than 3 years, indicating that the write traffic in the SSD is not sufficient enough to stress the SSD. Hence, we exclude SSD-MSNCFS from further consideration in our experiments.

It should be noted that these results take into account the stress and the recovery patterns that M-SSD experiences over its usage period. While our analysis in Section 3.2 showed that recovery periods increase the retention (and thereby the endurance) of the SSDs, these results indicate that even with the benefit of recovery periods, M-SSD cannot be used for the entire 5-year timescale for some enterprise workloads. These results were obtained after enabling state-of-the-art wear leveling optimizations like hot-cold data swap and rate-limited wearout in M-SSD [1], indicating that these existing optimizations are not sufficient

to extend the usability of the SSD over the timescale. Therefore, we require alternative approaches that can extend the lifetime of the SSDs to the maximum possible extent so as to minimize replacement and service costs and also the service downtime due to failure.

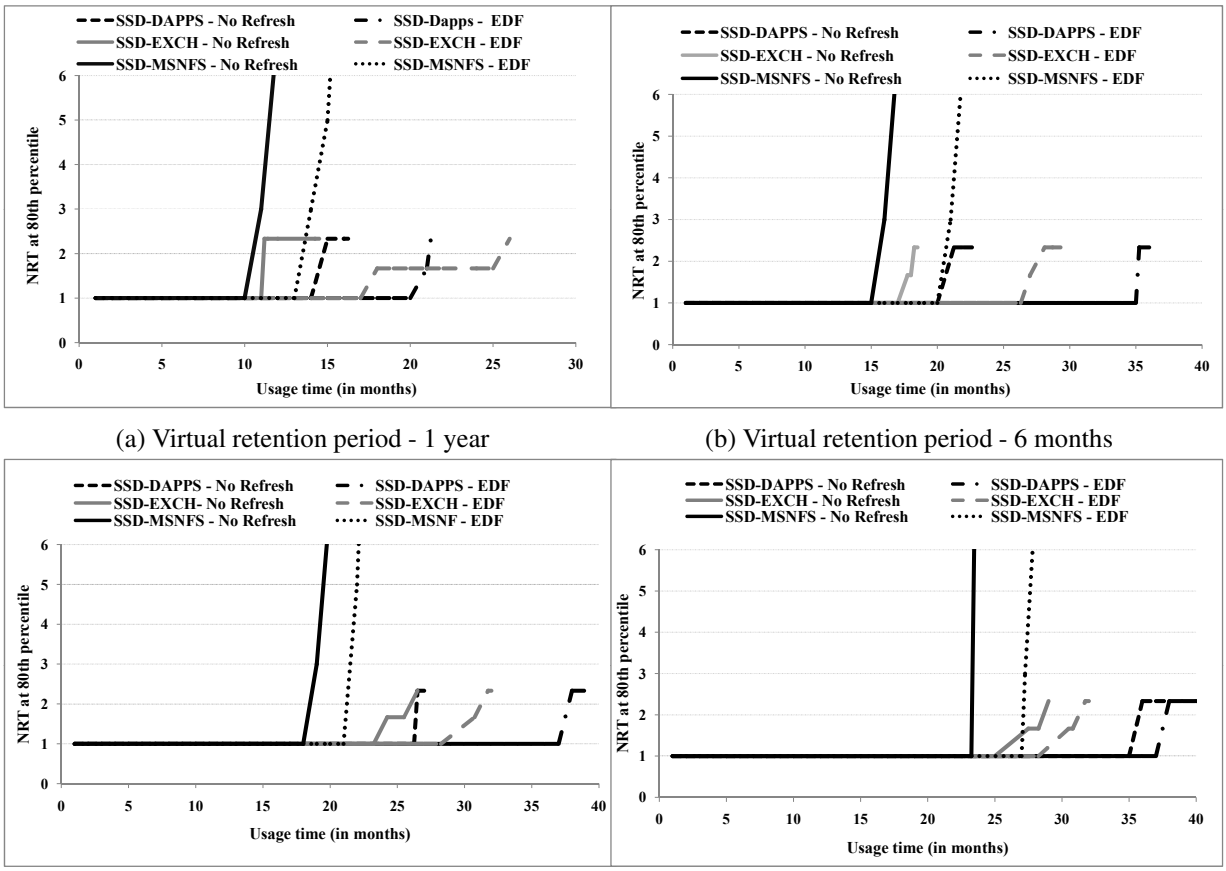
### 5.3 reFresh SSDs - An alternative approach to increase SSD lifetime

While trying to determine how to increase the usable lifetime of M-SSD, we observed one key point in the Figure 4. Figure 4 shows that even though the retention of some of the blocks in M-SSD is below the virtual retention period threshold, they have not yet reached their endurance limit. If we could use the remaining endurance to preserve the virtual retention of these blocks, then the usable lifetime of SSDs can be increased. In essence, what we need is a way to ensure the integrity of data stored in the underlying memory cell when the memory cell itself become unreliable.

We draw inspiration from DRAM, where data is stored in a capacitor whose inherent retention time is very short, and therefore data is preserved in the memory cells using a *refresh* mechanism. The refresh operation periodically restores the data in the memory cell capacitor so that no data is lost as long as the DRAM chip is connected to a power source and an intelligent scheduling of the refresh operations can minimize the performance interference of the refreshes with normal memory access operations. We apply this notion of a refresh to Flash memory, wherein the data in SSD blocks whose lifetime falls below the virtual retention threshold are refreshed, thereby increasing the usable lifetime of blocks. We call our SSD that use these refresh operations as *reFresh-SSD*.

Refreshing an SSD involves periodically checking all blocks in the SSD and moving data from less reliable blocks to more reliable blocks to match the reliability constraints of the application. Thus, a refresh operation involves a read operation of a less reliable block and a program operation to a more reliable block. Since a refresh operation is a combination of read and program operations, the only change required to support them in SSDs is to change the FTL software to initiate these operations whenever required and does not require any change to the hardware of the SSD or the flash memory chips. Refresh operations are different from wear leveling operations. Both move data across SSD blocks. But, refresh operations are triggered by the FTL when it senses that the lifetime of blocks containing data falls below some threshold (based on the application's requirement). Hence, refresh operations are triggered due to an immediate deadline, while wear leveling operations are triggered in order to achieve the long term objective of evening the wearout across blocks. The latency of a refresh operation can be significant, as it involves reading and writing all valid pages in the block. For M-SSD, this means that the latency of a refresh operation is 128 times the sum of read and write latency. Therefore, it is imperative to intelligently schedule the refresh operations so that the performance impact on normal I/O operations is minimized.

Since the need to refresh a block can be viewed as an operation that needs to be completed within a specific deadline to meet the virtual retention time requirement, we evaluate a refresh policy that is based on the *Earliest Deadline First (EDF)* scheduling algorithm. In this approach, the state of those blocks that contain valid data and whose lifetime is less than the virtual retention period threshold is maintained in a queue called the refresh queue. This queue is a priority queue where blocks with the lowest lifetime are at the head. Dequeue operations always begin at the head of the queue. The block with the least retention period is assigned the highest priority for the next refresh operation. Since flash does not support in-place updates, a target block needs to identified to move the contents of the block to be refreshed. To identify the target block, we use the wear leveling algorithm in the FTL to determine a free block with the highest remaining lifetime. At every refresh interval $r$, the FTL checks the refresh queue, picks all those blocks that need to be refreshed (refresh blocks) based on their deadline, identifies a free target block for each refresh block, copies the contents of refresh block to the target block, and finally erases the refresh block. The FTL




(a) Virtual retention period - 1 year

(b) Virtual retention period - 6 months

(c) Virtual retention period - 3 months

(d) Virtual retention period - 1 months

Figure 5: Impact of refresh on the lifetime of M-SSD

invokes garbage collection if the number of free blocks in the SSD becomes low. In this way, even if the lifetime of a block is less than the virtual retention period, the data in the block is periodically refreshed to ensure their integrity.

Unlike traditional approaches that seek to reduce the write traffic into the SSD to increase the lifetime of SSDs, refresh operations actually increase the write traffic due to additional write operations. While this may seem counterintuitive, it should be noted that refresh operations help the FTL exploit the remaining endurance in the SSD blocks instead of ignoring them just because their lifetime is below an application determined threshold. Even though this involves additional write activity, it increases the longevity of the SSD upto a certain lifetime (which is higher than the baseline) until these additional write operations themselves decrease the lifetime of blocks in the SSD. We soon reach a point in time where the SSD spends a majority of the time refreshing its blocks, thereby heavily interfering with normal I/O operations of the workload and causing severe degradation in SSD performance. At this point, the SSD becomes practically unusable.

Figures 5a to 5d show the benefit of using the EDF based refresh policy on the reliability of the SSD for the various workloads at different virtual retention period thresholds. The x-axis in these figures represent the usage time while the y-axis represents the normalized response time over the usage period (lower is better). These results were obtained by running simulations upto the point where the performance of reFresh SSDs is equal to the end-of-life performance of the baseline SSDs (without refresh). Beyond this point, a

significant amount of time is spent by the SSD is maintaining its state, instead of servicing the I/O requests of the workload. From these figures, we can make the following important observations:

1. Without refresh operations, SSDs with virtual retention thresholds of either 1 year or 6 months can be used for a maximum of 23 months (6-month virtual retention period for SSD-DAPPS). With refresh, the usage lifetime of the SSD can be extended to upto 36 months. The increase in lifetime varies from 23% for SSD-MSNFS to 56% to SSD-DAPPS.

2. For SSDs with lower virtual retention thresholds of 3 months and 1 month, the usage lifetime without refresh operations is a maximum of 26 months, while with refresh operations, the usage time increases to 38 months. The increase in lifetime varies from 6% for SSD-EXCH to 42% for SSD-DAPPS.

3. Reducing the virtual retention period requirement diminishes the gain in lifetime because the margin of lifetime for blocks is already becoming lower for lower virtual retention periods and blocks begin to become unusable within a relatively short time period.

4. With refresh operations, performance of the SSD varies with time. Even after the SSD crosses the baseline lifetime, there is a significant time period where refresh operations have very little impact on the performance of SSDs. This is because the lifetime of blocks is relatively high such that the number of refresh operations are performed infrequently. However, as the blocks in the SSD reach their physical reliability limit, the performance starts to degrade. We can observe that the normalized response time remains at 1 for some duration of time before starting to increase sharply and soon reaches the response time of the baseline case.

5. The benefit of refresh operations directly correlate to the intensity of the write traffic in the workloads. Workloads with lower write traffic (e.g. SSD-DAPPS) cause the SSD to fail at a later point in time and the rate at which the blocks fail is also slower compared to the more write-intense workloads. For those workloads where the rate at which blocks fail is low, fewer refresh operations are scheduled over any given interval of time. Therefore, the performance overheads imposed by these refreshes are also lower for those workloads.

In Figures 5a to 5d, we can observe that the duration between when the device is starting to fail (performance starting to degrade) and has completely been rendered unusable is in the order of a couple of months for all the workloads. This is because the wear leveler in the FTL tries to even the wearout of the flash and hence the standard deviation in the lifetime of blocks in the SSD is low. This behavior does not change with the inclusion of refresh operations because the wear leveler does not change its function.

Overall, we find that refresh is an effective way of leveraging the tradeoff between endurance and retention time to increase the usable lifetime of the SSD. Compared to the baseline case (no refresh), refresh operations allow the SSD to operate for extended periods of time with minimal performance impact. Hence, they can help reduce the number SSD replacements over the TCO period of the server infrastructure.

# 6 Related Work

Many architecture and software level techniques have been proposed to improve the reliability of flash. These include the use of hybrid HDD-SSD storage devices [37, 44], FTL optimizations [13], and the use of parity codes across a RAID of SSDs [21]. While there have been transistor-level studies on charge trapping and detrapping of flash [25, 41, 42], Mohan et al. propose leveraging these properties at the architecture

level to boost endurance [28]. Wu et al. propose to accelerate the recovery process using heat [33], although heat-assisted detrapping leads to data loss and therefore requires data to be backed up and reloaded into the SSD. Content aware techniques that examine data patterns and reduce the number of write operations have also been proposed [5,14]. Arpaci-Dusseau et al. propose a new interface called nameless writes that allows the device to control actions like wear-leveling while obviating the need for large tables [3]. A recent paper by Smullen et al. applied the idea of refresh to another non-volatile memory - STT-RAM - where they use memory cells that are designed to provide very low retention times, which provide improved performance and energy-efficiency for that memory technology when used for microprocessor caches [36]. To the best of our knowledge, our paper is the first to leverage the tradeoff between retention time and endurance and propose the use of refreshes to boost flash endurance.

Gaining a deep understanding of flash reliability has recently become a topic of great interest in the architecture and systems communities. There have been recent flash chip characterization studies that have shown that the endurance of flash memory chips is significantly higher than those given in datasheets [8,12]. There has also been a field-data analysis study from Fusion-IO on the endurance, read disturbs, and data retention of SSDs in datacenters [38]. Recent research efforts like [24, 43] have explored the benefits of relaxing flash retention to increase SSD performance. While this paper also examines the tradeoffs involved in relaxing flash retention, we focus our effort on increasing flash endurance instead of flash performance. Our evaluation of flash reliability is more realistic than [24,43] as our model considers the impact of recovery period on data retention and endurance.

# 7 Conclusion

SSDs can provide large performance and power benefits for datacenter storage. However, the limited endurance of flash memory needs to be addressed before SSDs can be practical for datacenter workloads. We have shown that the endurance of flash can be boosted if one can sacrifice retention time and that the reduction in the retention time can be offset by refreshing the data in the flash memory cells. We have shown that such refresh policies can significantly extend the usability of the SSD. Our future work in this area will be to explore other refresh policies, evaluate their implementation overheads within the flash controllers, and also study how refresh can be used in conjunction with other reliability techniques to boost SSD reliability in datacenters.

# References


[1] N. Agrawal and et al. Design Tradeoffs for SSD Performance. In *Proceedings the USENIX Technical Conference (USENIX)*, June 2008.

[2] Amazon.com. `http://www.amazon.com`.

[3] A. C. Arpaci-Dusseau, R. H. Arpaci-Dusseau, and V. Prabhakaran. Removing the costs of indirection in flash-based ssds with nameless writes. In *Proceedings of the 2nd USENIX conference on Hot topics in storage and file systems*, HotStorage'10, pages 1–1, Berkeley, CA, USA, 2010. USENIX Association.

[4] J. Brewer and M. Gill, editors. *Nonvolatile Memory Technologies with Emphasis on Flash*. IEEE Press, 2008.

[5] 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*, FAST'11, pages 6–6, Berkeley, CA, USA, 2011. USENIX Association.

[6] J. Cooke. Introduction to Flash Memory (T1A), 2008. `http://download.micron.com/pdf/presentations/events/FMS08_Intro_to_Flash_Memory_Jim_Cooke.pdf`.

[7] C. Delimitrou, S. Sankar, K. Vaid, and C. Kozyrakis. Storage i/o generation and replay for datacenter applications. In *Performance Analysis of Systems and Software (ISPASS), 2011 IEEE International Symposium on*, pages 123 –124, april 2011.

[8] P. Desnoyers. Empirical Evaluation of NAND Flash Memory Performance. In *Proceedings of the Workshop on Hot Topics in Storage and File Systems (HotStorage)*, October 2009.

[9] D. J. DiMaria and E. Cartier. Mechanism for stress induced leakage currents in thin silicon dioxide films. *Journal of Applied Physics*, 78(6):3883 –3894, sep 1995.

[10] Reducing Data Center Power Consumption Through Efficient Storage, 2009. `http://www.gtsi.com/eblast/corporate/cn/06_2010/PDFs/NetApp%20Reducing%20Datacenter%20Power%20Consumption.pdf`.

[11] The Diverse and Exploding Digital Universe: An Updated Forecast of Worldwide Information Growth Through 2011, 2008. `http://www.emc.com/collateral/analyst-reports/diverse-exploding-digital-universe.pdf`.

[12] L. Grupp and et al. Characterizing flash memory: Anomalies, observations, and applications. In *Microarchitecture, 2009. MICRO-42. 2009 42nd IEEE/ACM International Symposium on*, Nov. 2009.

[13] A. Gupta, Y. Kim, and B. Urgaonkar. DFTL: a flash translation layer employing demand-based selective caching of page-level address mappings. In *ASPLOS '09: Proceeding of the 14th international conference on Architectural support for programming languages and operating systems*, pages 229–240, 2009.

[14] A. Gupta, R. Pisolkar, B. Urgaonkar, and A. Sivasubramaniam. Leveraging value locality in optimizing nand flash-based ssds. In *Proceedings of the 9th USENIX conference on File and stroage technologies*, FAST'11, pages 7–7, Berkeley, CA, USA, 2011. USENIX Association.

[15] Intel X25-M and X18-M Mainstream SATA Solid-State Drives. http://www.intel.com/design/flash/nand/mainstream/index.htm.

[16] Process Integration and Device Structures, ITRS 2009 Edition. `http://www.itrs.net/links/2009ITRS/2009Chapters_2009Tables/2009Tables_FOCUS_C_ITRS.xls`.

[17] J. de Blauwe, J. van Heudt, D. Wellekens, G. Groeseneken, and H.E. Maes. SILC-related effects in flash $E^2$PROM's-Part I: A quantitative model for steady-state SILC. *Electron Devices, IEEE Transactions on*, 45(8):1745 –1750, Aug. 1998.

[18] J. de Blauwe, J. van Heudt, D. Wellekens, G. Groeseneken, and H.E. Maes. SILC-related effects in flash $E^2$PROM's-Part II: Prediction of steady-state SILC-related disturb characteristics. *Electron Devices, IEEE Transactions on*, 45(8):1751 –1760, Aug. 1998.

[19] JEDEC - Solid-State Drive (SSD) Requirements and Endurance Test Method. http://www.jedec.org/sites/default/files/docs/JESD218.pdf.

[20] JEDEC - Failure Mechanisms and Models for Semiconductor Devices. http://www.jedec.org/sites/default/files/docs/JEP122F.pdf.

[21] A. Kadav, M. Balakrishnan, V. Prabhakaran, and D. Malkhi. Differential raid: rethinking raid for ssd reliability. *SIGOPS Oper. Syst. Rev.*, 44:55–59, March 2010.

[22] L. Larcher, S. Bertulu, and P. Pavan. SILC effects on $E^2$PROM memory cell reliability. *Device and Materials Reliability, IEEE Transactions on*, 2(1):13 –18, mar 2002.

[23] M. Lenzlinger and E. Snow. Fowler-Nordheim tunneling into thermally grown Si$O_2$. *Electron Devices, IEEE Transactions on*, 15(9):686–686, Sep 1968.

[24] R.-S. Liu, C.-L. Yang, and W. Wu. Optimizing NAND Flash-Based SSDs via Retention Relaxation.

[25] N. Mielke and et al. Recovery effects in the distributed cycling of flash memories. In *Reliability Physics Symposium Proceedings, 2006. 44th Annual., IEEE International*, pages 29–35, March 2006.

[26] N. Mielke, T. Marquart, N. Wu, J. Kessenich, H. Belgal, E. Schares, F. Trivedi, E. Goodness, and L. Nevill. Bit error rate in nand flash memories. In *Reliability Physics Symposium, 2008. IRPS 2008. IEEE International*, pages 9 –19, May 2008.

[27] Moazzami, R. and Chenming Hu. Stress-induced current in thin silicon dioxide films. In *Electron Devices, IEEE Transactions on*, pages 139 –142, Dec. 1992.

[28] 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*, HotStorage'10, Berkeley, CA, USA, 2010. USENIX Association.

[29] Typical Data Retention for Nonvolatile Memory. http://www.freescale.com/files/microcontrollers/doc/eng_bulletin/EB618.pdf.

[30] Typical Data Retention for Nonvolatile Memory. http://www.spansion.com/Support/AppNotes/EnduranceRetention_AN.pdf.

[31] OCZ Deneva - eMLC based SSD. http://www.oczenterprise.com/downloads/solutions/ocz-deneva2-c-mlc-2.5in_Product_Brief.pdf.

[32] E. Pinheiro, W.-D. Weber, and L. Barroso. Failure Trends in a Large Disk Drive Population. In *Proceedings of the USENIX Conference on File and Storage Technonologies (FAST)*, February 2007.

[33] G. D. Qi Wu and T. Zhang. Exploiting Heat-Accelerated Flash Memory Wear-Out Recovery to Enable Self-Healing SSDs. In *Proceedings of the 3rd USENIX conference on Hot topics in storage and file systems*, HotStorage'11, Berkeley, CA, USA, 2011. USENIX Association.

[34] Quality Comparison of SLC, MLC and eMLC, 2011. http://www.flashmemorysummit.com/English/Collaterals/Proceedings/2011/20110809_F2C_Wu.pdf.

[35] S. Kavalanekar and et al. Characterization of storage workload traces from production Windows servers. In *Proceedings of the IEEE International Symposium on Workload Characterization (IISWC)*, pages 119–128, October 2008.

[36] C. Smullen, V. Mohan, A. Nigam, S. Gurumurthi, and M. Stan. Relaxing non-volatility for fast and energy-efficient stt-ram caches. In *High Performance Computer Architecture (HPCA), 2011 IEEE 17th International Symposium on*, pages 50–61. IEEE, 2011.

[37] 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*, FAST'10, pages 8–8, Berkeley, CA, USA, 2010. USENIX Association.

[38] H. Sun, P. Grayson, and B. Wood. Quantifying reliability of solid-state storage from multiple aspects. 2011.

[39] Application Report: MSP430 Flash Memory Characteristics. http://focus.ti.com/lit/an/slaa334a/slaa334a.pdf.

[40] K. V. Vishwanath and N. Nagappan. Characterizing cloud computing hardware reliability. In *Proceedings of the 1st ACM symposium on Cloud computing*, SoCC '10, pages 193–204, New York, NY, USA, 2010. ACM.

[41] R. Yamada and et al. A novel analysis method of threshold voltage shift due to detrap in a multi-level flash memory. In *Symposium on VLSI Technology, Digest of Technical Papers*, 2001.

[42] H. Yang and et al. Reliability issues and models of sub-90nm NAND flash memory cells. In *International Conference on Solid-State and Integrated Circuit Technology*, 2006.

[43] Q. Yang and J. Ren. Quasi-Nonvolatile SSD: Trading Flash Memory Nonvolatility to Improve Storage System Performance for Enterprise Applications. In *High Performance Computer Architecture (HPCA), 2012 IEEE 17th International Symposium on*.

[44] Q. Yang and J. Ren. I-cash: Intelligently coupled array of ssd and hdd. In *High Performance Computer Architecture (HPCA), 2011 IEEE 17th International Symposium on*, pages 278 –289, feb. 2011.