



White Paper

Longterm Data Endurance (LDE) for Client SSD

Oct. 31, 2008

Abstract

SanDisk, Inc. is submitting the following white paper to the JEDEC 64.8 workgroup. It provides an overview of the issues involved in the endurance of SSDs and endurance testing. It describes SanDisk's views on LDE and how to measure LDE. However, it is not an actual spec, which at a minimum depends on industry consensus on user workloads.

LEGAL DISCLAIMER

This document is subject to any applicable SanDisk Corporation terms of use. No license, express, implied, by estoppel or otherwise, to any intellectual property rights is granted by this document. SanDisk makes no warranties whatsoever, and hereby expressly disclaims any and all express or implied warranties regarding this document and its contents. This document and its contents are subject to change without notice.

Copyright © 2008 by SanDisk Corporation and its applicable affiliates. All rights reserved.

1 megabyte (MB) = 1 million bytes. 1 gigabyte (GB) = 1 billion bytes. Some of the listed capacity is used for formatting and other functions, and thus is not available for data storage.

Table of Contents

1	Overview	4
1.1	Goals.....	4
1.2	Definitions	4
2	Factors that Affect Flash Drive Endurance.....	4
2.1	Storage Activity Categories	4
2.2	Effects of Writes on Endurance.....	5
2.3	Capacity Used.....	7
3	Things the System Can Do to Improve Endurance	7
3.1	SSD.....	7
3.2	Operating System.....	8
4	Previously Proposed Endurance Metrics	9
4.1	Write Amplification.....	9
4.2	Calculated Metrics.....	10
5	SanDisk’s Endurance Metric Proposal - LDE	11
5.1	What Does an LDE Value Represent?	11
5.2	Should LDE be a Warranty?.....	11
5.3	Endurance vs. Capacity	12
5.4	Endurance Grades	12
5.5	Endurance vs. OS	12
5.6	Endurance vs. Command Queuing	13
6	Methodology for Testing and Measuring LDE	13
6.1	Phase 1 – Create a Test Workload	14
Step 1.1	– Selecting typical users and their workload	14
Step 1.2	– Tracing workload of “typical” client users	14
Step 1.3	– Characterization of traced workloads.....	14
Step 1.4	– Creating merged characteristics representing test workload	14
6.2	Phase 2 – Create a Test Workload Generator Program.....	14
6.3	Phase 3 – OEM Testing and Marketing or Labeling.....	15
Step 3.1	– Running the test program.....	15
Step 3.2	– Retention Testing.....	15
7	How to Capture or Represent User Workloads	15
7.1	What’s wrong with current benchmarks?.....	15
7.2	Workload Generation Considerations for LDE	16
7.2.1	Trace Playback vs. Workload Generation	16
7.2.2	Capture File or Interface Traffic?.....	16
7.2.3	Trace Size	17
8	Workload Characteristics	18
8.1	Characteristics That Can Be Ignored for LDE	18
8.2	Characteristics That Are Important for LDE.....	18
8.3	Selection of Workload Parameters	19
8.3.1	Number of user types	19
8.3.2	Smoothing the Distributions	20
9	Playback Program.....	20
9.1	Acceleration Factors.....	20
10	Retention Testing	21
11	Summary.....	22

References.....	22
Appendix A – Idle Time Compression	22
Appendix B – Sample Distributions	23
B.1 Write Size	23
B.2 Sequential Writes.....	25
B.3 Aligned Writes.....	27
B.4 Footprint.....	27
B.5 Flush Intervals.....	31
Appendix C – How OEM Might Arrive at an LDE Value	31

1 Overview

This document describes an approach to SSD endurance testing. As background, factors affecting endurance are first discussed. It next looks into the merits of various endurance metrics and explains where they fall short. Finally, we present our preferred methodology for endurance testing.

The focus of this white paper is on client usage. In this context, client is synonymous with single user.

1.1 Goals

SanDisk's primary goals for an SSD endurance test are that it should:

- Be representative of typical user activity
- Enable accelerated testing (i.e. a test for a 5 year product life could be performed in weeks or a few months)
- Be verifiable by OEM or a 3rd party
- Produce a single value that a typical user can use to compare flash products (this value will be used as a label on the product's box, but will not be the basis for a product warranty)

To do this, we need to first understand what creates the activity on a flash based SSD and then we will discuss ways to measure it.

1.2 Definitions

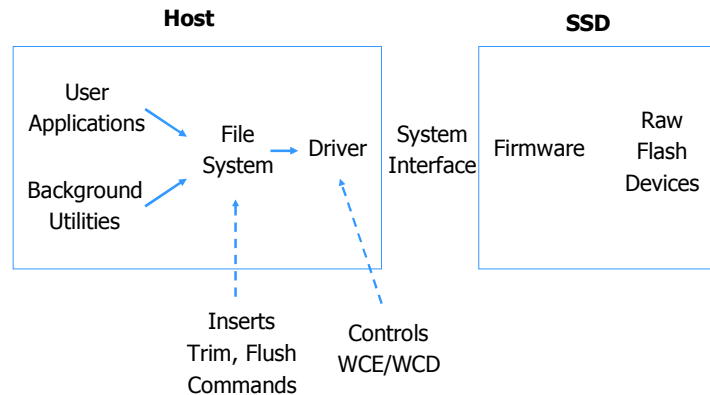
The terms *endurance*, *endurance failure*, *retention* and *retention failure* of flash memory are as defined in Ref. [1]. The same definitions of *endurance*, *retention* and *retention failure* can be used for SSD. However, because an SSD has spare blocks, an SSD does not have endurance failure just because one block has endurance failure. SSD endurance failure occurs only after all spare blocks have been used up and the next block endurance failure occurs. Therefore, spare blocks improve both the reliability and the endurance of an SSD.

2 Factors that Affect Flash Drive Endurance

2.1 Storage Activity Categories

A typical single user PC environment looks like that shown in Figure 1:

Figure 1: A typical client system



Storage activities that impact the endurance of the flash device in this environment come in several categories:

- 1) **User applications:** Programs the user directly requests such as Office applications, email, web browsers, multi-media players, etc.
- 2) **Background applications:** Virus scanning, software version checking, or archiving and backup utilities that a user typically does not explicitly initiate. Many of these are run when the machine is idle or based on a predetermined schedule, e.g., run at 12 noon daily.
- 3) **OS functions:** Paging and, more recently, file caching, prefetching, and defragmenting. Some of these are invoked when the user is running applications, others work more in a background mode when the system is less busy.
- 4) **SSD internal operations:** Garbage collection, etc.

The combination of categories 1-3 create the I/O activity that the SSD sees over the system interface, while category 4 is strictly internal and created by the device's own firmware. Each of the above could represent 20-60% or more of the I/O activity that the underlying flash devices see; the percentages vary from user to user, depend on the software being used (including the OS), vary over time of day and may be only roughly correlated with the amount of user activity.

Furthermore, the workload characteristics (R/W ratio, transfer sizes, inter-arrival time of I/O requests, etc.) of categories 1-3 can vary significantly. Even within a category or an application, different implementations can vary significantly – for example, virus scanning can have several very different workload characteristics, from nearly all long reads to a nearly 50/50 R/W ratio with short writes to file system metadata.

2,2 Effects of Writes on Endurance

The primary activity that influences the endurance of a flash based storage device (or SSD) is the amount of writes. In particular, in a number of cases, writes can create 2-8X the number of actual bytes to be written by the application; this is called Write

Amplification. We describe some Write Amplification examples below, but we first briefly describe the structure of the flash as related to write operations.

(Note: A very large number of repetitive reads of the same data may create a problem known as *read disturbs*. While the threshold for this varies from vendor to vendor and generation to generation, the overall impact on endurance from reads is probably 3 orders of magnitude less than writes, so we ignore it for LDE.)

Externally, the smallest addressable unit of data for flash memory (and hence SSD) is a *sector*, which is typically 512 bytes. However, internally the smallest unit of data that can be written is a *page*, which is some multiple of sectors, typically 8 sectors (4KB) or 16 sectors (8KB). Furthermore, flash memory must first be erased before it can be written, and the unit of memory that is erased at a time is a *block*, which is some large multiple of pages. Lastly, the number of times that a block can be erased is finite, which is the root cause of the endurance issue.

Because of the above, a page can not be updated in place. Instead, it is logically mapped and written to a different physical page in an erased block. Furthermore, when only part of a page is being updated, a read-modify-write (RMW) operation must be done since a page is the granularity for doing a write.

In each of the examples below, the underlying page size in the raw flash is assumed to be 4KB. (Although this size may vary with vendor and time, it generally will not change the overall effect).

- 1) Short writes of less than a page create RMW scenarios. For example, a single 512B sector write will require reading 4KB and writing 4KB page, which is 8X the amount of writing the user requested. Writing 2 to 7 sectors will have a smaller amplification factor for the actual amount of data written if all sectors fall within the same page. For instance, writing 2KB will result in one 4KB page being written (2X) if the 2KB falls within a single page.
- 2) Misaligned file systems. Writes that are not aligned to start on 4KB page boundaries create a similar RMW requirement. This may be a rare problem for certain OS. However, it is a common problem affecting all legacy Windows OS. Up until now Windows has the first 63 sectors of a drive reserved for the master boot record and other boot related info, leading Windows to start the first partition for its file system at LBA 63. The result is that starting addresses of allocation clusters always have LBAs that end in 0x7 or 0xF (4KB cluster). This is known as the Offset 1 problem - and there are similar Offset 1 cases. Since this is quite common and only recently recognized as a problem, a SSD can internally re-align itself by always adding 1 to a user requested LBA to remove this problem. (We encourage file system developers to avoid this problem and ensure that all writes start on 4KB boundaries, which coincidentally happens to also be needed as HDDs transition to 4KB native sectors.)
- 3) Writes that span extra pages. Even if problem 2) is avoided, there may still be some instances where applications write to unaligned starting sectors and a length that results in an extra page. In this case, both the first and last pages will require RMW.

For example, writing 1KB will result in two 4KB pages being written (8X) if the 1KB spans across a page boundary.

- 4) A similar problem is writes that do not end on 4KB page boundaries. Even when file systems allocate space on aligned, reasonable sized clusters (say 4-16KB), the end of a file is usually rounded off to a fixed number of 512B sectors. In the pure random case, 7/8ths of the time, this also triggers a RMW since the SSD does not know if the remaining bytes are to be preserved or can be erased, and so on average nearly doubles the number of bytes written for the last page. (We recommend that file system developers write out the last record of a file in at least a 4KB amount to avoid this problem.)

It is clear from these examples that write patterns can have a huge effect on Write Amplification. We'll discuss how the SSD firmware and the OS can help in Section 3.

2.3 Capacity Used

When the total LBAs (footprint) of an SSD actually used by a user is small, relative to the total size of the SSD, less garbage collection is needed. When the LBAs of a device's stated capacity have all or nearly all been written to, any given page is more likely to be associated with valid data (assuming no Trim command to free some of these LBAs up) and hence needs to be copied when its block is being reclaimed. Conversely, on average, the blocks being reclaimed are more likely to have fewer valid pages (which need to be copied) if a lower percentage of the LBAs of a device has been used. Thus, the percent of a device's capacity used by a user has a direct effect on endurance.

3 Things the System Can Do to Improve Endurance

3.1 SSD

In addition to offsetting LBAs by 1 as described in (2), Section 2.2, there are some more common things that an SSD do to improve its overall data endurance.

- 1) **Wear Leveling.** By ensuring that all blocks in a SSD get re-used evenly when LBAs are remapped, erasures are evenly distributed. This way, no blocks will fail substantially ahead of the others.
- 2) **Sparing.** Spare blocks are provided to replace blocks that have failed, extending the life of a SSD. Obviously, the more spare blocks there are, the longer is the SSD's endurance.
- 3) **Over-provisioning.** To create a new block that can accept new writes, one or more old blocks are selected for erasure. However, before a selected block can be erased, any valid data that it holds must first be read, combined with valid data from other selected blocks, and then written elsewhere (garbage collection). Clearly, the least impact on endurance is achieved if such relocation writes can be kept to a minimum, meaning blocks with the least amount of valid data (ideally none) should

be selected. By having more “free” capacity than its spec’ed capacity, the probability of finding blocks with low percentage of valid data is increased.

- 4) Write caching. By storing write data in non-flash cache memory, many flash writes can be eliminated. This can happen in one of two ways. If the same LBA is written to twice or more times (write cache “dirty” hit), only the last write needs to be written to flash. Writes to different sectors of a page arriving as separate commands can be combined into a single flash write. If the write cache is volatile memory, such as DRAM, then this feature must first be enabled by the OS – see below. If the write cache is some form of non-volatile memory, such as MRAM, then this feature does not need OS support. Of equal importance, many flash devices may also update (write) some amount of metadata on each I/O, but by combining I/Os into longer I/Os and reducing the number of I/Os, it may also reduce the amount of metadata written.

3.2 Operating System

There are several commands or flags that a File System or OS introduces into the I/O workload that may have significant effects on the endurance of a flash device.

- 1) Write Cache Enable (WCE) or its inverse, Write Cache Disabled (WCD) is a parameter from the host to the storage device. The OS (and sometimes the OEM) typically sets this value, but the user can also manually override it. With WCD, writes must be committed to non-volatile storage before they return completion, using either battery or capacitor backed DRAM or writing data directly to flash. With WCE, the data can be written to volatile DRAM and later de-staged to flash, which is generally a faster operation. Of course WCE also carries a small risk that power will go out and the data can be lost. Thus, enterprise systems always have WCD. Some desktops have WCD and some have WCE, while notebooks invariably always have WCE (since they have a system battery and can do an orderly shutdown that flushes any volatile data to flash). The impact of write caching on endurance has been discussed in 4) of the previous section.
- 2) Flush commands – with WCE, the OS typically issues flush commands when a file is closed, which tells the device to “flush” or write out all volatile data to flash before continuing. The frequency of flushes (or equivalently, the interval between flushes, measured in either time and/or KBs transferred) can significantly affect performance as well as the endurance of the flash. Long intervals between flushes allows a write cache to do its thing and reap the benefit of write caching, whereas frequent flushing would cut into the beneficial effect of write caching.
- 3) Trim commands – if the file system knows that certain data has been logically deleted, it can inform the SSD device via a Trim command. Notifying the SSD what sectors are no longer valid will save the SSD from having to relocate those sectors during garbage collection, reducing the amount of writes that would be done and hence improving endurance.

Without Trim commands, a typical user device will be physically full inside the SSD, since all physical sectors will eventually have been written to, even if the user and file system think it is only 20 or 30% logically full. In this case, the SSD device

manufacturer will have to provide for some extra invisible capacity in order to do garbage collection – see 3) on over-provisioning in the previous section 3.1 (the extra capacity ranges from a few percent to as much as 30 or 40% extra flash capacity). Even so, depending on the workload, it may take a significant amount of copying (writing) to clear up blocks that can be erased and then written. With Trim commands, the SSD is only as full as the capacity actually used by the user – see Section 2.3 above. So even with low amounts of extra capacity, an SSD that is only 30-50% logically full will have less data to copy during garbage collection. (The actual effects and trade-offs between extra capacity, Trim commands, and amount of user space are much more complicated and depend on the exact firmware algorithms being used, but the above is a simplification for the non-expert.)

The usage of Trim is virtually non-existent in 2008 but is expected to grow with newer releases of OSes over the next several years. Thus, a benchmark or workload (or user) that uses a newer OS with Trim will see an improved Endurance value.

- 4) Hints – like Trim, other new commands with hints (e.g., sequential access) similar to those being considered in T13 for AHCI and NVMHCI may also have impact on endurance.

4 Previously Proposed Endurance Metrics

In this section, we discuss what some endurance metrics have been proposed by others and why we think they are not suitable. We then present in Section 5 our proposed metric.

4.1 Write Amplification

As discussed earlier in Section 2.2, a user's write request can result in more data being actually written in a SSD than requested. The term *write amplification* (WA) is roughly defined as the number of bytes written to flash divided by the number of user requested bytes written to the SSD over the storage interface. A value near 1 is considered good and higher values are less desirable. Therefore, some in the industry have proposed using WA as the endurance metric.

There are several main problems with using Write Amplification as endurance metric

- 1) WA in itself does not take into account the endurance of the underlying flash memory. An SSD with $\frac{1}{2}$ the WA of another SSD is no better if it uses flash memory that has $\frac{1}{2}$ the endurance.
- 2) WA has no provision to handle an SSD architecture in which mixed flash device types are used. Also, many flash parts incorporate the ability to write in the equivalent of SLC and MLC modes. In both cases, they have different performance and endurance characteristics. Characterizing them with a single WA would fail to address how many writes are to each type and how that affects overall endurance.

- 3) There are various ways to slightly alter the WA value. There may be delays between when the data is transferred over the storage interface and when it is written to the flash (in other words, it may be cached). So over short intervals, the WA may be near zero. While this may offer better performance, and may be observable in many benchmarks, this lower value of WA is clearly not sustainable or relevant to long term endurance measurement. Another medium term effect is to measure a device when it is new (unused) or has extra pre-erased capacity (e.g. garbage collection and/or Trimming have been done). In both cases, the short term effect is to provide better performance and a reduced WA value, and these effects could last days or at least as long as most performance benchmarks today.
- 4) WA is commonly used in conjunction with a SSD's capacity in estimating its usable life – see equation (1) in the next section. An ambiguity arises for SSD designs that have extra invisible capacity – see “over-provisioning” discussion in Section 3.1. If a device reports its capacity as X and it has, say, 30% extra capacity for improved performance and endurance, should the user be using X or 1.3X with the WA in estimating the device's usable life? It would seem 1.3X is the correct number to use. However, the percent of over-provisioning may not be known to the user if it is not published by the vendor.
- 5) Last, but not least, WA is very difficult for an OEM or 3rd party to verify.

4.2 Calculated Metrics

There have been several proposals in the industry to define metrics that are calculated based on some formulas. Commonly, the calculation involves some or all of the following:

- the SSD's capacity (may include extra “invisible” capacity)
- the write cycles (endurance) of the underlying raw flash
- the SSD's write amplification factor
- the SSD's wear leveling factor
- etc.

For example, an SSD endurance metric may be defined as:

$$\text{No. of bytes that can be written} = \text{capacity} \times \text{write cycles of the raw flash} / \text{WA}. \quad (1)$$

However, as we just discussed in Section 4.1, there may be different ways to spec or measure WA and capacity. Furthermore, the number of write cycles in the raw flash may not be disclosed, and the capacity could be made up of different amounts of flash with different endurance values (either slightly different or an order of magnitude or more different). There may be varying numbers of spare blocks in each flash device, and different thresholds for using these spares (e.g. how much ECC is used and when does one pro-actively replace a block that is getting a certain level of ECC error).

The overall issue is that we should not base a total endurance measurement on any mathematical model that relies on component measurements. Equations are too complex to be broadly used, yet too simple to yield meaningful results for an end user.

5 SanDisk's Endurance Metric Proposal - LDE

To the end customer, the SSD is a system. What is needed is an endurance specification for the system as a whole – a single, simple “gas gauge” that everyone can use. Our goal of the endurance test is to find the right mix of workloads and then run it against different devices to measure their overall endurance.

We propose to define *Longterm Data Endurance* (LDE) as based on the total number of GBs written over an SSD's “nominal” lifetime. In other words, X TBs of total user GBs written, and abbreviated as TBW (terabytes written). This is a simple “gas gauge”-like metric that users can easily understand. If a user has some idea of how many GBs/day he writes on the average, he can estimate the actual usable lifetime for him from dividing LDE by his average daily usage. For example, if a drive has a LDE of 40TBW and the user writes 10GB per day, then LDE is not exhausted in the first 10 years.

As part of the LDE definition, not only should an SSD function properly during its LDE period, but it also should be capable of retaining all its data for one year after its LDE has been exhausted.

5.1 What Does an LDE Value Represent?

A device could have several LDE values representing various things. A vendor could have a best case, worst case, or average/typical case LDE.

Best case is probably useless except for outrageous marketing claims, so it will be ignored.

Worst Case is more like a warranty or guarantee, and similar to tire ratings – you should expect N thousand miles of life or it will be replaced on a pro-rata basis. If a device is rated for X TBs of writes over its lifetime and the user exceeds this value, presumably the warranty is expired (the device may continue to operate), and there could be feedback mechanisms to gauge how close it is to end of life. We will discuss why this is not appropriate in Section 5.2.

Average or typical endurance is analogous to car MPG ratings – it is a measurement of MPG over some average set of conditions and your mileage may be different. Compared to the alternatives, this is our recommendation and it will be our goal to find the best estimates of average or typical user behavior.

Below are some things to consider when assessing the meaning of an LDE rating.

5.2 Should LDE be a Warranty?

As the HDD industry discovered in the mid to late 1990's with Smart [6] or PFA, if a spec becomes the basis for a warranty claim, vendors become very cautious. In the HDD situation, HDD vendors were reporting statistics that may forecast catastrophic failure. Thus, users wanted full warranty coverage and the vendors wanted to reduce

“false positives” (as it cost them money). As a result, many of the Smart/PFA settings are so cautious as to be useless.

In the case of SSDs, there is not the analogy of catastrophic failure – a write failure when all spare blocks have been consumed typically renders the SSD as read-only with no loss of data. However, turning LDE into a warranty spec would require different test methods and unnecessary conservatism in setting an LDE value for the end user. Similarly, to use the car analogy, it would be like spec'ing a car for only 3 years or 36,000 miles, based on warranty, when most cars are used long past their warranty periods. So if too conservative a value is used (as would be the case if LDE was the basis for a warranty claim), users would then be more likely to exceed the LDE value, and for example, risk loss of data through retention problems.

Therefore, we do not think LDE should be used as a basis for warranty.

5.3 Endurance vs. Capacity

The total amount of writes that a SSD can write over its lifetime is roughly linearly related to the actual storage capacity of the device. In other words, the LDE or total number of TBs that can be written on a 128GB SSD is approximately 2X the LDE of a 64GB device of the same make and model.

On the other hand, the daily I/O activity is more closely related to the user, their applications and their file sizes. So a given user might produce 1, 2, or 5 GB/day on average of writes, regardless of whether they are using a 64, 128, or 256GB SSD. In other words, the number of GBs written per day by a user may be independent of the capacity of the SSD he is using. This means in general the usable life for a user is longer for a SSD with bigger capacity.

5.4 Endurance Grades

Since the users cannot fully appreciate small changes in Endurance values, we propose to work with the industry to come up with various “levels” of Endurance to avoid gamesmanship. Since the current capacity values are factors of 2, and endurance is related to capacity, we propose basing the Endurance Grades on roughly factors of 2 separations. For example, the total endurance values may be bucketed into 10, 25, 50, 100, 250, 500, 1000, etc. TBs of writes. And if for example, a 64GB SSD has an Endurance value of 50 TBs, then a similar 128GB SSD would likely have an Endurance value of 100 TBs (although this is up to the SSD vendor to decide).

5.5 Endurance vs. OS

Please refer to Section 3.2 for a list of discussions about things in an OS that can affect endurance. It is clear from that discussion that the LDE value is highly dependent on which OS is being used for its test and measurement. We propose using “typical” workloads of the common OSes for measuring the LDE (the industry will have to provide guidance on their preferred weightings of which OSes). Thus if a user upgrades their OS or their workload changes, this may impact their usage of an SSD, but does not affect the underlying Endurance value (“amount of gas available”). In other

words, if they bought an SSD with x TBs of Endurance, it should always provide that level of endurance (“amount of gas”), even if their activity (rate of consumption) changes (“switch from city to highway driving”).

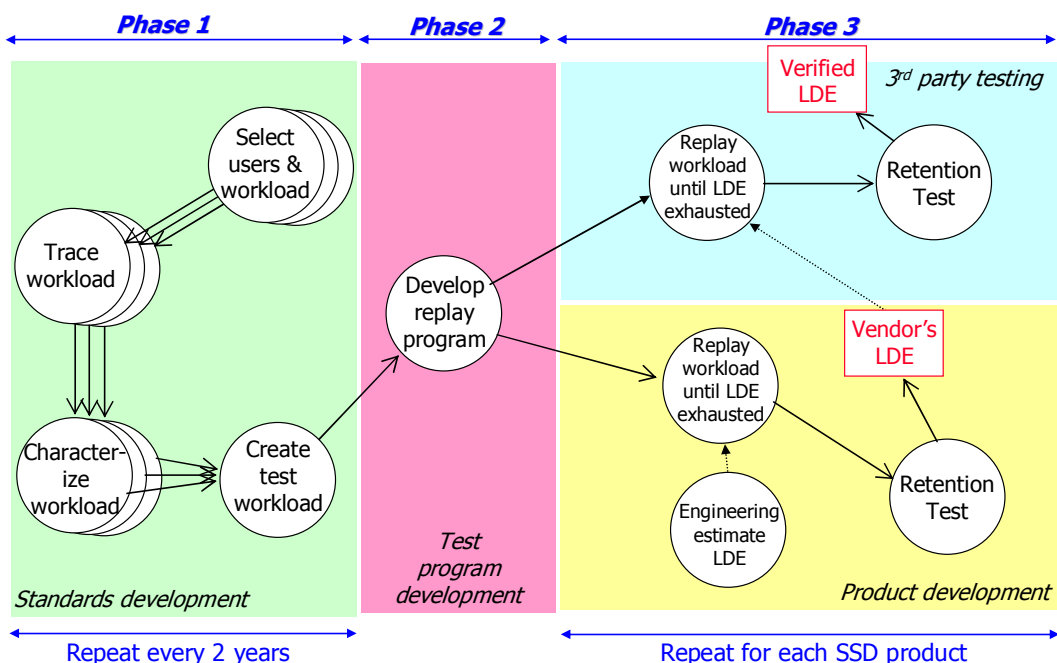
5.6 Endurance vs. Command Queuing

SATA and SCSI both support command queuing. When the command queue is deep, potentially it can be beneficial to endurance. This is because it offers the opportunity for multiple writes to be consolidated, depending on the access pattern, and hence reduces the amount of RMW that would otherwise have to be done (i.e., improving the WA and therefore endurance). However, it has been reported in previous analyses of typical client access patterns that generally the queue depth is often not very deep. Therefore, we suggest that, to keep things simple, command queuing not be considered when testing for endurance.

6 Methodology for Testing and Measuring LDE

We propose the following methodology to the industry for adoption. There are three major phases, namely, creating a test workload, creating a playback program for this test workload, and testing for LDE. Each phase has one or more steps. The various phases and steps of this methodology are illustrated in Figure 2 and explained in the remainder of this section.

Figure 2: Methodology in Developing & Testing LDE



6.1 Phase 1 – Create a Test Workload

As will be discussed in Section 7.1, we believe a new test workload needs to be defined and created specifically for testing endurance. For LDE to be meaningful to the average user, it is important that a workload representing the typical average user be applied in the testing and measurement for LDE.

Step 1.1 – Selecting typical users and their workload

SanDisk wants to collaborate with OEMs and other industry experts to create a set of workloads. Each participant company can select and define its own “typical” users and their workloads, including what mix of applications, which OS, which file system, system setups (e.g., WCE or WCD), etc. The pool of users may represent different segments of computer users running a diverse set of very different applications.

Step 1.2 – Tracing workload of “typical” client users

Each participant will capture detailed traces at the SATA level of the I/O activity for the above workloads for approximately 2 weeks. Section 7 will discuss the various issues concerning the capturing of these traces.

Step 1.3 – Characterization of traced workloads

Each captured trace is reduced to a set of “signatures” that represent distributions of the critical workload parameters, such as: LBA distribution, block sizes; inter-arrival times of write, flush and trim commands; and the interdependencies of these distributions. See Section 8 for a more detailed discussion.

Step 1.4 – Creating merged characteristics representing test workload

The individual sets of characteristics for each one of the selected and traced workloads are then merged together to form a single set of workload characteristics representing that for a “typical”, “average” client user. We will work with participating industry partners to smooth and weight these distributions to best reflect industry consensus of “typical” users.

6.2 Phase 2 – Create a Test Workload Generator Program

The playback program is a program that creates a repeatable artificial workload and sends its I/O commands to the device under test, so it will be kind of like the IOMeter test program that most people are familiar with. It will take the workload characteristics extracted in Phase 1 of our suggested method (Section 6.1) as inputs and will continually generate write commands to the test device in accordance to the input characteristics, and will run continuously until the LDE value of TBs have been written. More details about the workload generator replay program will be discussed in Section 9.

Phases 1 and 2 would be done with industry partners one time (and probably repeated every year or two as workloads change, e.g. to reflect different Operating Systems, different applications, and different file sizes).

6.3 Phase 3 – OEM Testing and Marketing or Labeling

In the last phase, the LDE of an SSD is tested. The steps in this phase can be carried out by a vendor or a third party tester; the steps will essentially be identical for both.

Step 3.1 – Running the test program

A vendor would internally have their engineering specify the X TBs of LDE that they expect their device to achieve. A third party tester would use the LDE number published by a vendor (see Appendix C for how this might be done). Then the vendor or the third party tester would run the standard test program against a device until X LDE TBs had been transferred. For example, say, the test program has an average write rate of 20MB/s, then a device spec'ed at 50 TBs LDE would take 2,500,000 seconds or about 4 weeks to measure. (Of course, if the device fails before X TBs have been written, then the vendor would need to revise its LDE spec downwards.) Afterwards, they would proceed to the next step.

Step 3.2 – Retention Testing

The last step of the playback program is to record a known data pattern on the SSD. Then the user would perform accelerated retention testing per JEDEC standards [1,4, 5] to simulate the retention period. Afterwards, they would use the playback program a final time to scan the device and ensure that the known data pattern has been fully retained. If this step passes successfully, then the LDE number that was tested to in Step 3.1 can be published. More details about retention testing will be discussed in Section 10.

7 How to Capture or Represent User Workloads

In Phase 1 of our suggested methodology, it is necessary to decide on a workload for testing LDE. Endurance testing has at least one key similarity with performance testing – namely, we must figure out what constitutes “typical” user behavior.

7.1 What’s wrong with current benchmarks?

Most current benchmarks are of two types:

- 1) Application workloads (e.g. FutureMark’s PCMark [2]): These typically focus on user applications, but ignore background utilities and most OS functions (see Section 2.1), so they may capture a small (less than 50%) percentage of the typical I/O activity.

Additionally, these programs exercise only a small part of the logical LBA space – the biggest one we’ve seen so far only does a few GBs of writes. While this may appear to offer a way to accelerate testing, the results may be misleading if the same benchmark was simply run over and over again (and it may be easy to “cheat” on such a benchmark).

- 2) Synthetic workloads (e.g. IOMeter): These workloads can span an entire SSD using either random, sequential or some mixed workload. These come closer to

being what we need; however, they currently do not implement Flushes or Trims, so the generated workload is not sufficiently accurate.

We conclude that a new workload needs to be defined specifically for testing SSD endurance.

7.2 Workload Generation Considerations for LDE

There are a few things to consider for creating a test workload for testing LDE.

7.2.1 Trace Playback vs. Workload Generation

There are two approaches for creating the endurance test program:

- 1) Capture user traces and play them back, or
- 2) Capture the characteristics of the user behavior and construct a synthetic workload generator that matches those characteristics.

The challenge with either approach is that we need a representative pool of users and their work styles. We propose forming an alliance with industry partners to jointly define what the representative users and environments should be.

A further problem of approach (1) is the size of the trace – if this is expected to represent the cumulative I/O activity of an extended period of time such that it is long enough to stress test a SSD, the trace will be huge and may prove unwieldy.

We also anticipate that the applications may change over time (which ones, their versions and algorithms) and the user workloads will change over time (frequency of which application is run and file sizes). Thus any selected test workload is probably a best estimate of user behavior at some point in time and we propose revising this periodically, perhaps every 2 years, just as many benchmarks do.

Another issue with strictly playing back a captured trace is that the LBAs are highly dependent on the capacity of the original device with which the trace was taken. Replaying it on a smaller device and some of the LBAs would be out-of-range. Conversely, replaying it on a larger device, then the upper LBAs will never be accessed.

For these reasons, we recommend that approach (2) above be used for our proposed endurance test methodology.

7.2.2 Capture File or Interface Traffic?

We can capture traffic at the:

- 1) file system level, or
- 2) system interface.

If we capture traffic at the file system level, it becomes OS specific (which is true for the user experience and ultimate endurance). In other words, a set of applications run under XP would produce X TBs of endurance, Y TBs of endurance under Vista, and Z

TBs of endurance under Windows 7. (This could be analogous to the difference between city and highway driving and the effect on Miles Per Gallon (MPG). Or perhaps the difference in applications is analogous to city vs. highway driving, and the OS is analogous to the quality of the road or the weather conditions?)

Capturing and/or synthesizing file system workloads is not new and is done in SPECsfs for example. However, to do this accurately is not easy - we would have to measure and then create a distribution of file I/O characteristics, including average file size, access or transfer sizes, write and erase ratios (we can ignore the reads), which directory, fragmentation, and inter-arrival times for these I/Os. This could be done for example with something like the Filemon utility and then a simple application program that generates the desired file operations at the requested distributions. Additionally, some OS generated activities may not get captured if they do not use the file system.

On the other hand, if we capture the traffic at the system interface, it includes the characteristics of Flush, Trim, and WCE/WCD, in addition to all the application and file system level traffic.

Moreover, measuring I/O activity at the system interface is easier. It requires measuring and creating a distribution of the transfer sizes, frequency of writes/trims/flushes, locality, and inter-arrival times, but does not require worrying about file sizes, which directory, or fragmentation; these are all approximated by the distribution of transfer sizes and locality – examples are shown in Appendix B. A normal SATA interface analyzer can capture this trace, and a relatively simple I/O generation program (similar to IOMeter) can be created to recreate the desired distribution and timing.

In the end, we need to agree on specific OSES and device settings as well as user activities (e.g. as is done in cases of PCMark[3] or SYSmark[2]).

[We recommend capturing trace at the system interface level for our proposed endurance test methodology.](#)

7.2.3 Trace Size

We believe when we trace the workload of a “typical” user, the tracing should be long enough such that several times the device capacity of data is written. We realize that the user does not actually write over the entire LBA space, as there are likely static regions (reflecting software and old data) that do not typically get written. For example, a user might have 50% of the capacity as static, with the other 50% used dynamically by the file system.

If we just had a benchmark or workload that writes less than the device capacity, it won't fill the device or test the garbage collection efficiency (see Section 2.3). Running the same workload over and over again may or may not reflect the real life user traffic to the device.

[We also think the trace needs to span enough elapsed real time so that it captures OS background utilities like virus scanning.](#)

7.3 Collecting User Traces

We invite the industry to collect long traces of users they believe to be representative of typical users or various segments of users (e.g. casual, extreme, etc.). SanDisk can work with partners if needed to reduce these traces into the characteristics described in the next section, since only the characteristics are needed, not necessarily the full traces.

8 Workload Characteristics

Many different characteristics can be extracted from a workload. It is next to impossible to enumerate them all. We will discuss some of the more common ones in this section.

8.1 Characteristics That Can Be Ignored for LDE

Here are some of the common workload characteristics (and their distributions) that we think will not be needed in creating a playback program for LDE testing.

1. Read-to-write ratio. While this is an important workload characteristics, because of the low impact of reads on endurance (see Section 2.2), we are recommending skipping all read commands in the playback in order to speed up LDE testing (see discussion in Section 9.1).
2. Drive idle time, defined as the time between the completion of an I/O command (or all outstanding commands in the case of command queuing) and the arrival of the next command. Since we are recommending compressing out all drive idle times in order to speed up LDE testing (see Section 9.1), idle time distribution in a workload can be ignored.
3. Interarrival time, defined as the time between the start of one I/O command to start of the next command, can also be ignored. This is partly because we recommend that command queuing not be used (see Section 5.6), and partly because we are compressing out idle times (there is some relationship between interarrival time and idle time).
4. Queue depth distribution can be ignored because we recommend that command queuing not be used (see Section 5.6).

8.2 Characteristics That Are Important for LDE

Here are some of the common workload characteristics (and their distributions) that we think will be important in creating a playback program for LDE testing. This is not meant to be an exhaustive, all inclusive list. We invite others in the industry to join SanDisk in defining such a list.

1. Write size, in bytes or number of sectors
2. Distribution of write alignments (see Section 2.2)
3. Percent of writes that are sequential
4. Footprint – in particular the percent of disk capacity used (see Section 2.3)
5. Percentage of LBAs that are re-written, particularly the distribution of their frequency
6. Distribution of Flush commands
7. Distribution of Trim commands (when available and used)
8. ...

In addition to the simple attributes above, we may also need to examine and use interdependencies or correlations between such attributes. Examples are relationship between:

1. Sequential writes and write size
2. Footprint of sequential writes
3. Footprint vs. write size
4. ...

See Appendix B for more details about these examples.

8.3 Selection of Workload Parameters

After a variety of user workloads have been traced and reduced to the above distributions and correlations, we invite the industry to participate in two key steps:

- 1) agree on how many user categories there are, and
- 2) merge, “smooth” and “simplify” the distributions that are the inputs for the playback program.

Note that these two steps are inter-related and potentially iterative. At the end of this stage, the industry will need to consolidate around one or a small sets of distributions, which correspond to one or a small number of user types.

8.3.1 Number of user types

Until we have collected traces across a broad set of users, it is difficult to know how much variation exists in the above distributions and correlations between types of users. If many of these are the similar, then we may be able to define a single “average” user with a single value of LDE and produce a single set of parameters for the playback program. (This would be ideal).

However, we may discover that some of the distributions and correlations are significantly different for sizable segments of users. For example, we may find that certain segments of casual users do not use large files and their activity is mostly short random access. As another example, we may discover that most “heavy duty” users work with much larger files and their average transfer sizes and sequentiality is much higher. If these examples are true and it is determined by many SSD vendors that this makes a significant difference in their SSD endurance, we may be forced to define several user profiles, with a corresponding set of workload parameters and subsequent LDE values.

This is the logical equivalent of deciding that there are two workloads for defining fuel economy – city and highway, and not one or 3 or more. Other performance benchmarks frequently decide to break up workloads into many different types of activity, and then frequently they produce some weighted average of these as a single benchmark score. At this time, we do not know enough about the differences between user types and whether these affect endurance.

The latter approach above (which we rejected) would be to assume monolithic or single application users, such as a user that does nothing but web browsing, a user that does nothing but video gaming, a user that does nothing but spreadsheet editing, etc. We could artificially create such single minded users and collect their traces over a period of time and attempt to estimate the endurance characteristics of these workloads and then somehow weight them or average them. Aside from the difficulty in doing this, we do not believe this is realistic of how users work or likely to produce a realistic endurance metric.

8.3.2 Smoothing the Distributions

As we collect traces from different users and reduce these to distributions, we expect many of these to be somewhat similar. In the end, we will need to reduce these different user distributions into 1 or more common distributions. There is a danger here that we could “average” or smooth all of the distributions into a single hypothetical user that reflects no one.

9 Playback Program

SanDisk is volunteering to create a vendor neutral user application program that uses the distributions and correlations of the previous step as inputs to drive a synthesized workload to an SSD device under test. This program will create a synthesized workload up to a specified “Endurance” value at an accelerated rate. We expect the playback program to run on a standard PC platform, issuing SATA commands over a SATA interface. The binary (or object) code will be made freely available to all and the source code will be available under a no-fee license agreement.

JEDEC standards [1] states that there is no one data pattern or set of patterns that is worst-case for all failure mechanisms. Until further discussions decide otherwise, we suggest using alternating patterns of 0x00, 0x55, 0xAA and 0xFF as the data pattern to be used by the test program for writing to the SSD under test.

We propose that the playback program should be run at a case temperature of 40C. This corresponds to a nominal temperature rise inside a notebook from a 25C operating temperature. We pick this temperature point because the LDE measurement is intended to be a typical and not worst case measurement.

9.1 Acceleration Factors

There are two ways that the playback can be accelerated –

- elimination of reads,
- elimination of idle times,

and for simplicity and to offset slightly the elimination of idle times, we also propose to run at a queue depth of one.

Typical client systems are only used a fraction of the time. Even a heavy duty desk bound worker (or game player) probably uses their system less than 50 hours a week, or 1/3 of the time. Many users only use their system 10-20 hours per week due to meetings or just use in the evenings at home, or roughly 1/10th of the time.

Even when a user is busy using their system, there are many long gaps between writes (or any I/O). In fact, aside from batch processing like applications, even a busy interactive user is only likely to be doing I/O activity 1-5% of the time. They may have bursts of 100's of I/Os, but their actual average I/O rate is probably less than 10 IOPS.

We propose eliminating all read commands in the trace (see Section 2.2 for a short discussion about effect of reads on endurance). We also propose compressing out all idle times and running commands single threaded (equivalent to queue depth 1). We believe doing both will typically reduce the running time of a typical user by almost two orders of magnitude; a factor of 60 means simulating 5 years (60 months) of “nominal rate” traffic would take 1 month (see Appendix A for further details).

There are two nearly offsetting side effects of this acceleration factor:

- 1) In an SSD under normal client workloads, idle times may be used to do background garbage collection to improve performance. However, at normal speed, unless they did perfect prediction of future workloads, this may have a slightly worse effect on endurance. So eliminating the opportunity to do background garbage collection may artificially improve the endurance measured by the LDE method.
- 2) Running at queue depth 1 (as proposed) may reduce some potential parallelism, which may also subtly affect caching behavior and endurance in a negative way.

Our estimates indicate that the combination of these factors produces a small single digit percentage effect on the overall endurance of some of our future SSD products. However, other SSD developers must validate these assumptions or this acceleration factor (removing idle times) is invalid for their devices.

10 Retention Testing

We expect that the industry will adopt an SSD data retention requirement for fully cycled devices to be 1 year. We also propose to use a nominal storage temperature of 25°C (non-operating storage temperature), which like the operating temperature, is a typical (but not worst case) value.

Retention testing will be done at an elevated temperature in order to reduce the testing time. These conditions can be tested at various temperatures (stress temperature) following Arrhenius equation with activation energy of, say, 1.0 eV for data retention. For example, 1 year data retention @ 25°C may be tested at:

1. Stress temperature of 85°C require bake time of ~13 hours
2. Stress temperature of 100°C require bake time of ~3.5 hours
3. Stress temperature of 125°C require bake time of ~ 0.5 hours

Please refer to reference [5] for additional details regarding High Temp Data Retention (HTDR) qualification requirements.

JEDEC standards [1] states that there is no one data pattern or set of patterns that is worst-case for all failure mechanisms. Until further discussions decide otherwise, we suggest using alternating patterns of 0x00, 0x55, 0xAA and 0xFF as the data pattern to be used by the test program for writing to the SSD under test.

11 Summary

In this paper, we enumerate the issues and factors affecting the endurance of a SSD. We discuss why Write Amplification as well as metrics that are based on formulas are not suitable as endurance metric. We then propose and define Longterm Data Endurance (LDE) for the SSD industry to adopt. We describe a three-phase procedure for testing/measuring LDE:

- 1) Create a test workload
- 2) Create a playback program
- 3) Endurance and retention testing

References

1. JEDEC Standard No. 22-A117A, “*Electrically Erasable Programmable ROM (EEPROM) Program/Erase Endurance and Data Retention Stress Test*”
2. “*SYSmark 2007, An Overview of SYSmark 2007 Preview*”, BAPCO, May 2008
3. “*PCMark Vantage Whitepaper V1.0*”, FutureMark Corp., Nov. 13, 2007
4. JEDEC Standard No. 74, “*Early Life Failure Rate Calculation Procedure for Electronic Components*”
5. JEDEC Standard No. 47F, “*Stress-Test-Driven Qualification of Integrated Circuits*”
6. T13 Document e05173r0, “*List of Public SMART Attributes*”, Oct. 18, 2005

Appendix A – Idle Time Compression

We analyzed a one-day trace of an engineer. This trace is 30,779 seconds (about 8.5 hours) long. Approximately 6GB of total data is written to the drive. (Various distributions about the writes of this trace are discussed in Appendix B). The average write data rate for the whole trace is therefore

$$6\text{GB} / 30779\text{s} = 0.19\text{MB/s.}$$

Let’s assume that this workload is the basis for the endurance testing program and that an SSD can perform all of these writes in 400 seconds of total I/O time. Thus, by eliminating all the time between writes (this means skipping all reads as well as idle times) and running with queue depth of 1, we get a speedup factor of about

$$30779 / 400 = 77,$$

and the estimated data rate is about

$$6\text{GB} / 400\text{s} = 15\text{MB/s.}$$

So in this example, we have an acceleration factor of 77 over the observed or “nominal” write data rate.

To test a drive with an LDE of 40TBW at this data rate will take approximately

$$40\text{TB} / 15 = 740 \text{ hours}$$

or about 31 days.

In the above example, if we were trying to demonstrate an LDE of 400 TBW, it would take 10 times as long. Likewise, if the average data rate of the SSD to run the “endurance workload” is 1/2 to 1/5 of that shown, it would take 2 or 5 times longer to demonstrate its LDE.

This is because the actual time to run the test is related to the ratio of the LDE value divided by the amount of data that the SSD under tests writes in the average day running the endurance workload. So using our same 40 TBW, if an SSD being tested runs the simulated workload at 10 MB/sec, it will take $40 \text{ TB}/10 \text{ MB/sec} = 1111$ hours (or 46 days). At 20 MB/sec, it will take $40 \text{ TB}/20 \text{ MB/sec} = 556$ hours (or 23 days).

Appendix B – Sample Distributions

We will present examples of some of the distributions that are described in Section 8.2.

The first workload that we use for creating these distributions comes from tracing the daily activities of an engineer’s laptop machine running Windows XP Professional. The machine has 1GB of memory and a 120GB HDD. The machine is set up with disk write caching turned on (WCE), so the drive will be seeing flush commands from the host. NCQ (native command queuing) is not used. The length of the trace is one full working day of roughly 10 hours. There are approximately 800,000 I/Os in the entire trace, of which about 470,000 are writes.

The second workload comes from tracing PCMark Vantage [3]. There are eight subtests in PCMark Vantage. The traces of all eight subtests are concatenated to form one single trace. There are about 13,000 writes in the trace and a total of about 395MB of data written.

The distributions of the final workload chosen for LDE measurement will likely differ from the examples we show here, due to the use of possibly a different OS, a different file system, and different mix of applications. We welcome input from OEMs and others that have experience with a broad cross section of users and want industry consensus on the final distributions that drive the Playback program.

B.1 Write Size

One statistic of interest is the distribution of request size of the writes, which is as shown in Figure 2a for our sample user trace. Note that over 53% of the writes are for 8 sectors (4KB); 8.9% are for 128 sectors (64KB), 5.7% are for 16 sectors (8KB), and 4.9% are for 32 sectors (16KB). However, note that 16.7% are not multiples of 4KB.

Write Size Distribution Sample Client User Trace

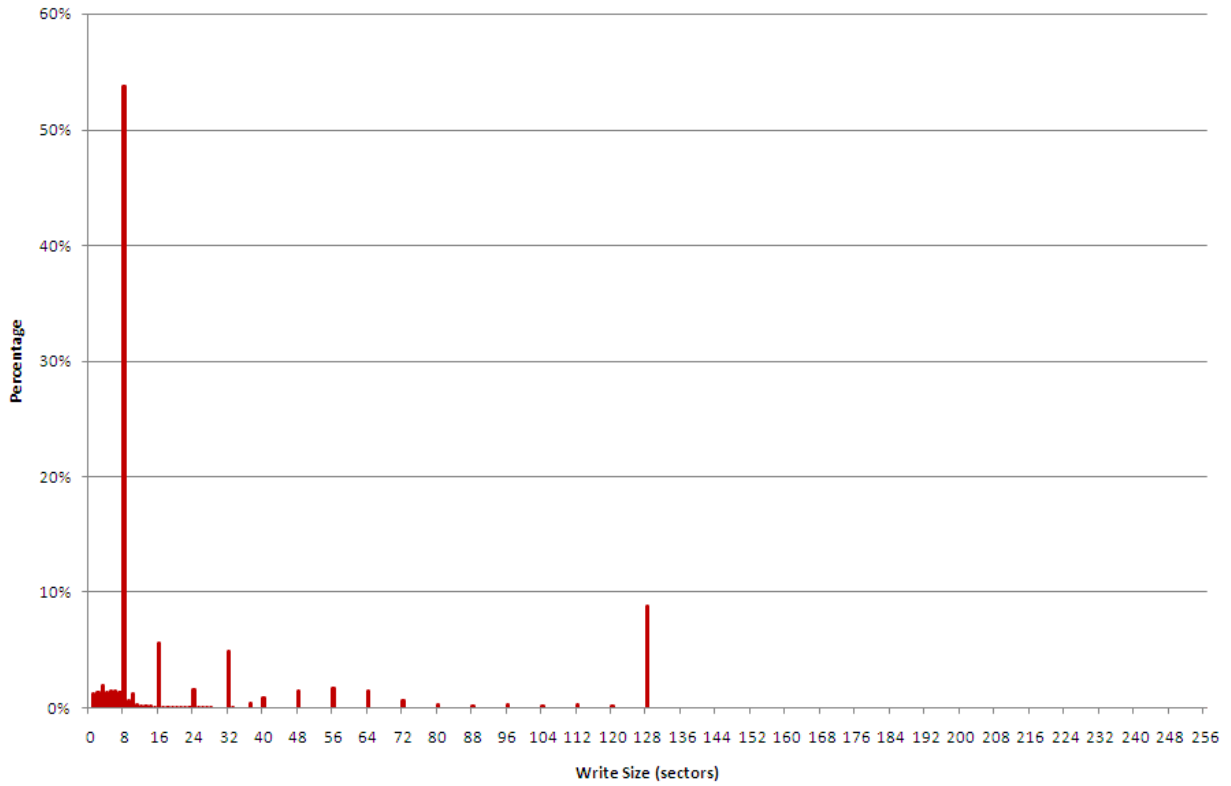


Figure 2a

Figure 2b shows the write size distribution for PCMark. There is much similarity between this and the real user workload. The only major difference is that PCMark uses 256 sectors for large block transfers, while the user workload uses 128 sectors.

Write Size Distribution PCMark

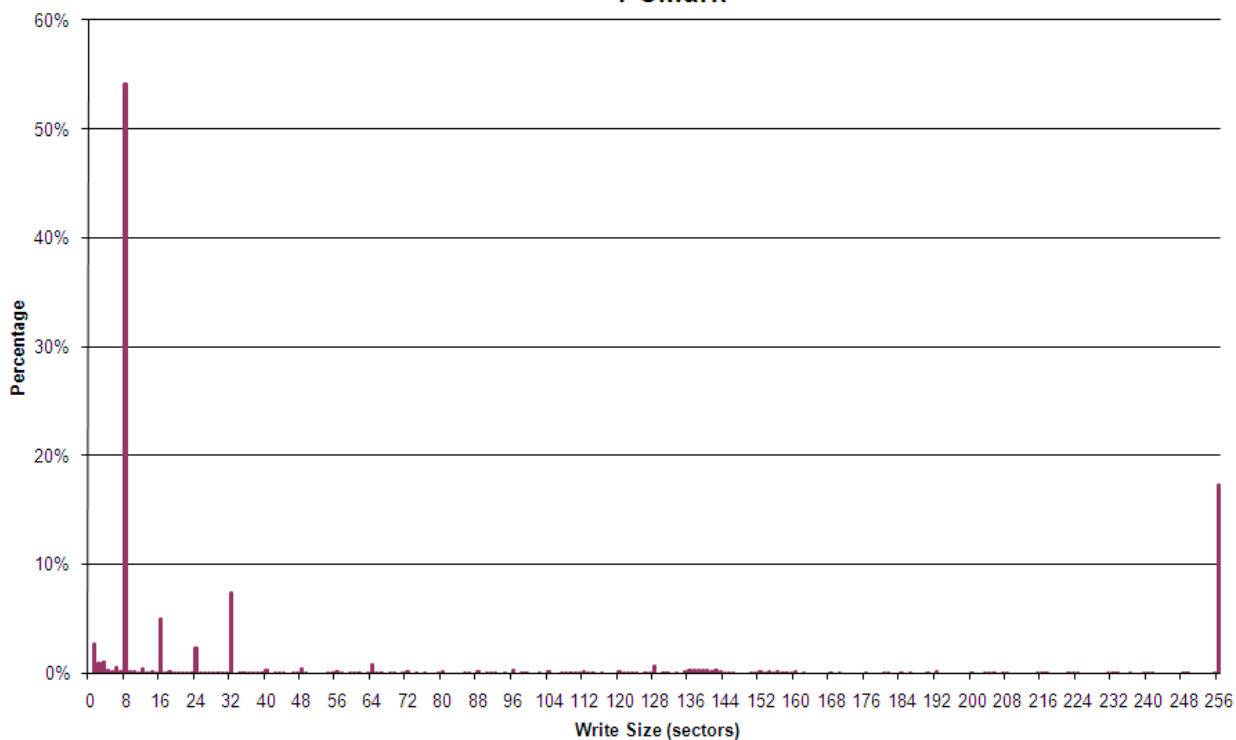


Figure 2b

B.2 Sequential Writes

For our sample user trace, 29% of all write commands are sequential writes. A write is deemed sequential if its starting LBA is contiguous to the ending LBA of the last write command, regardless of how many read commands separate the two write commands. Figure 3a plots the write size distributions of sequential and non-sequential writes separately.

Write Size Distribution for Sequential and Non-sequential Sample Client User Trace

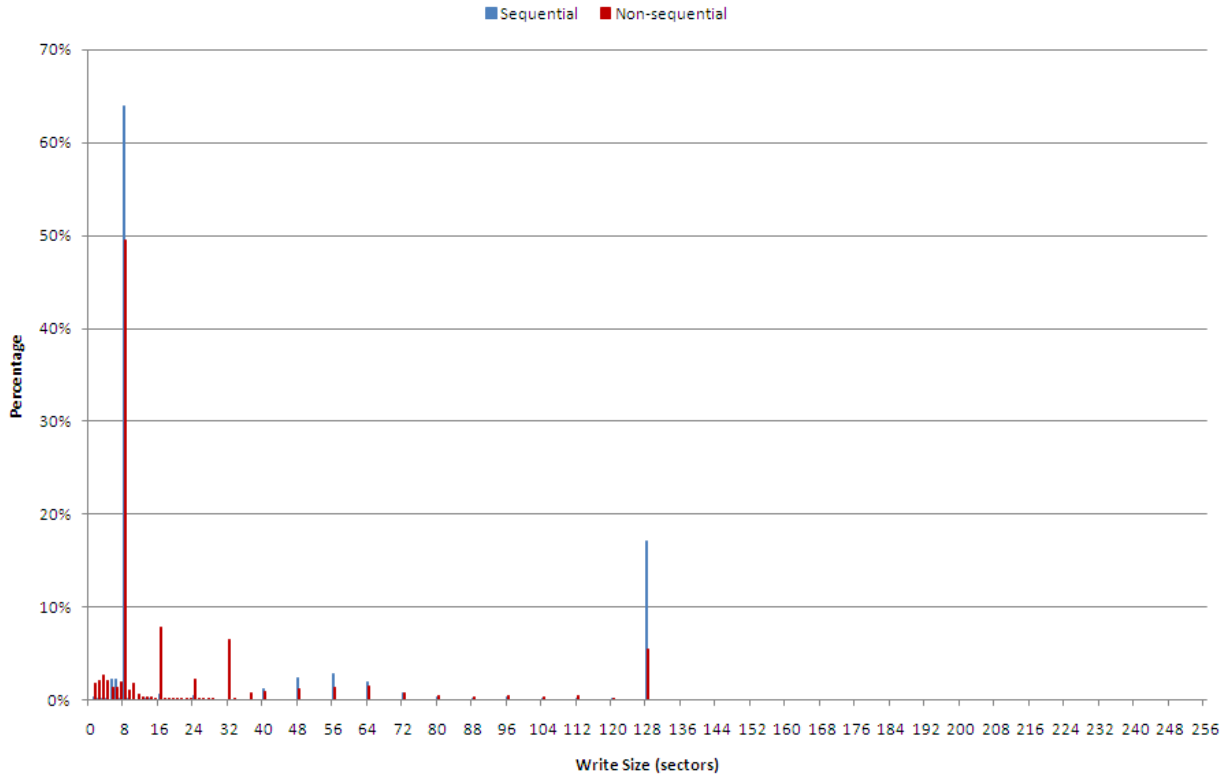


Figure 3a

In the PCMark trace, only 20% of the writes are sequential. The write size distributions of sequential and non-sequential writes for PCMark are plotted in Figure 3b.

Write Size Distribution for Sequential and Non-sequential PCMark

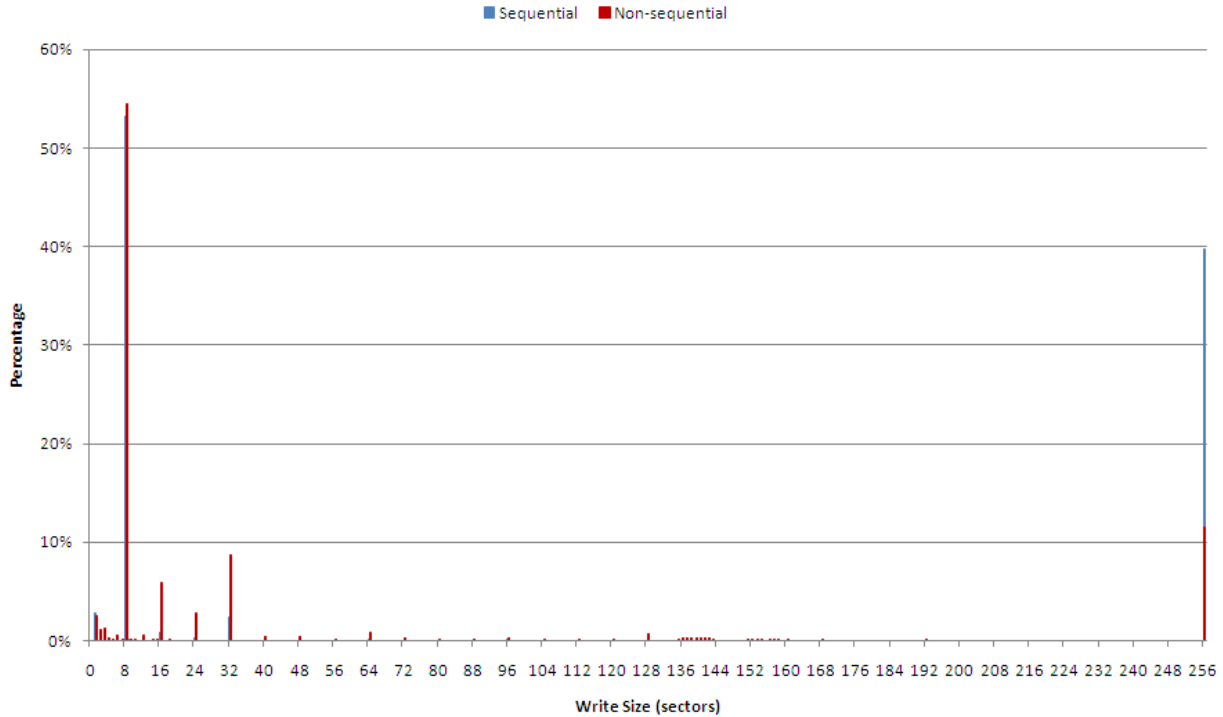


Figure 3b

B.3 Aligned Writes

As expected and pointed out in 2), Section 2.2, because Windows XP is used, none of the writes in the sample user trace are 4KB aligned. A write is aligned if and only if it starts on a 4KB boundary and ends at a 4KB boundary, thus requiring no RMW. However, when 1 is added to the LBA of each command, 82% of all the writes become aligned writes. Of the remaining 18% non-aligned writes, 10% are smaller than 8 sectors (4KB), i.e., they are less than a page.

Because the PCMark trace was taken with Vista which has 4KB support, 81% of all the writes are aligned. The remaining 19% are of sizes that are not multiples of 4KB.

B.4 Footprint

We define the footprint of a workload to be the areas of a drive being accessed by the workload. For this analysis, we arbitrarily divide the drive into 100 equal sized logical regions and count the frequency of write commands going to each region. The result for our sample user trace is as shown in Figure 4a.

Write Footprint Sample Client User Trace

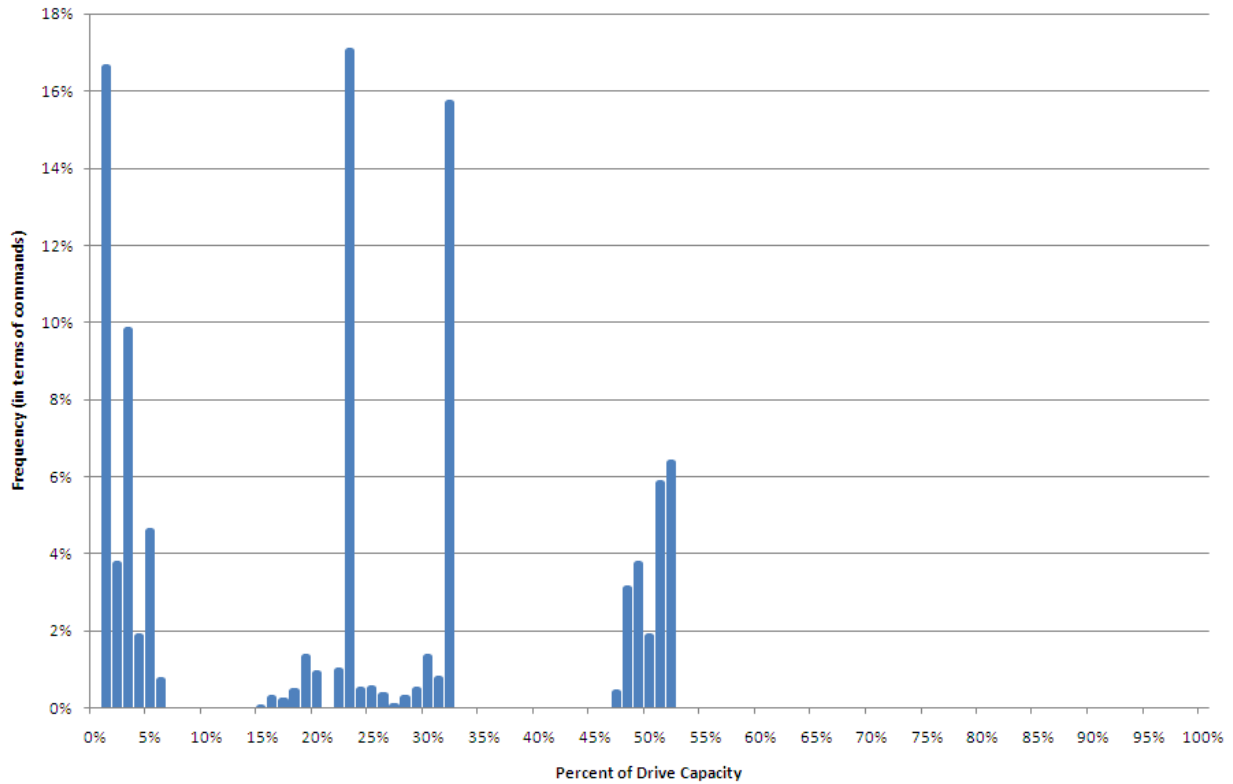


Figure 4a

It can be seen that portions of the drive get accessed much more often than others, and a good fraction of the drive never gets accessed.

In Figure 4b, we show what fraction of each bucket is sequential writes.

Percent of Sequential vs. Footprint Sample Client User Trace

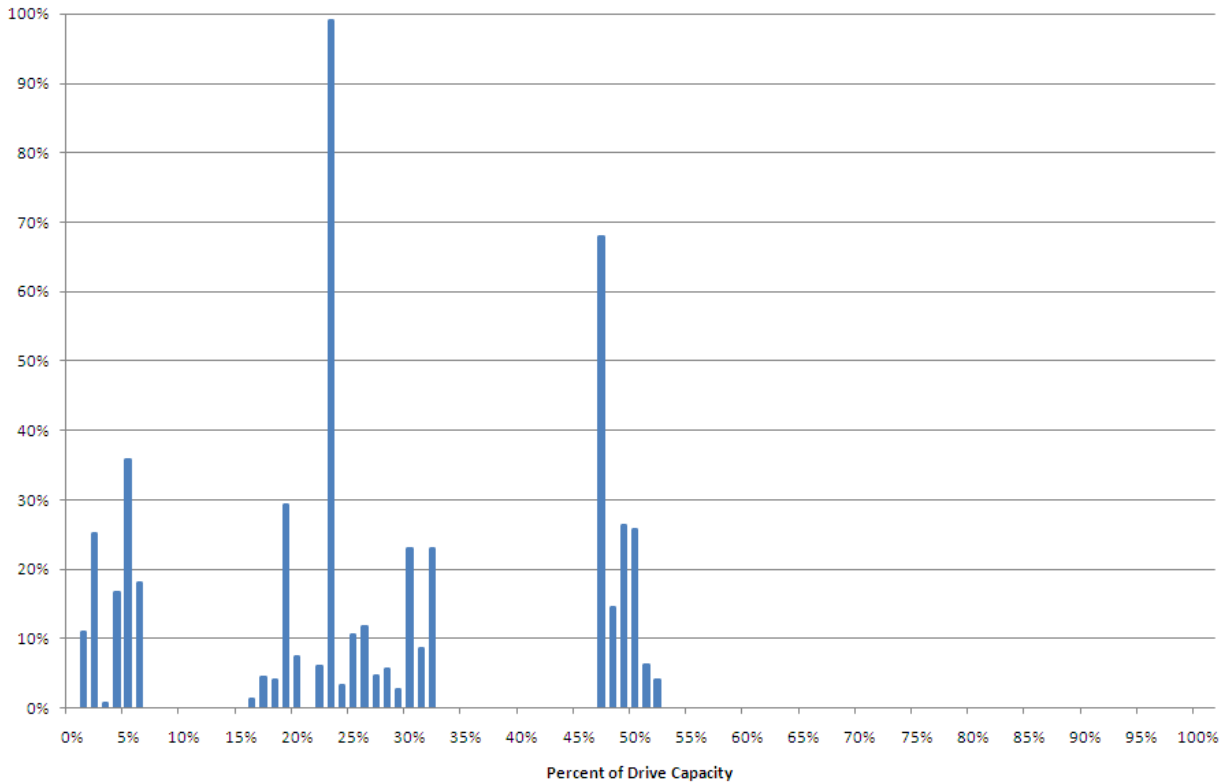


Figure 4b

However, the data in Figures 4a and 4b only represents one day of user activity, and a key factor in overall endurance will be how much of the SSD gets written to (see section 2.3). Clearly, a broader class of users and longer traces (or parametric extrapolation) will be required to accurately represent this parameter.

Figures 5a and 5b show the footprint distributions for PCMark. It can be seen that PCMark writes only two very narrow bands of LBAs.

Write Footprint PCMark

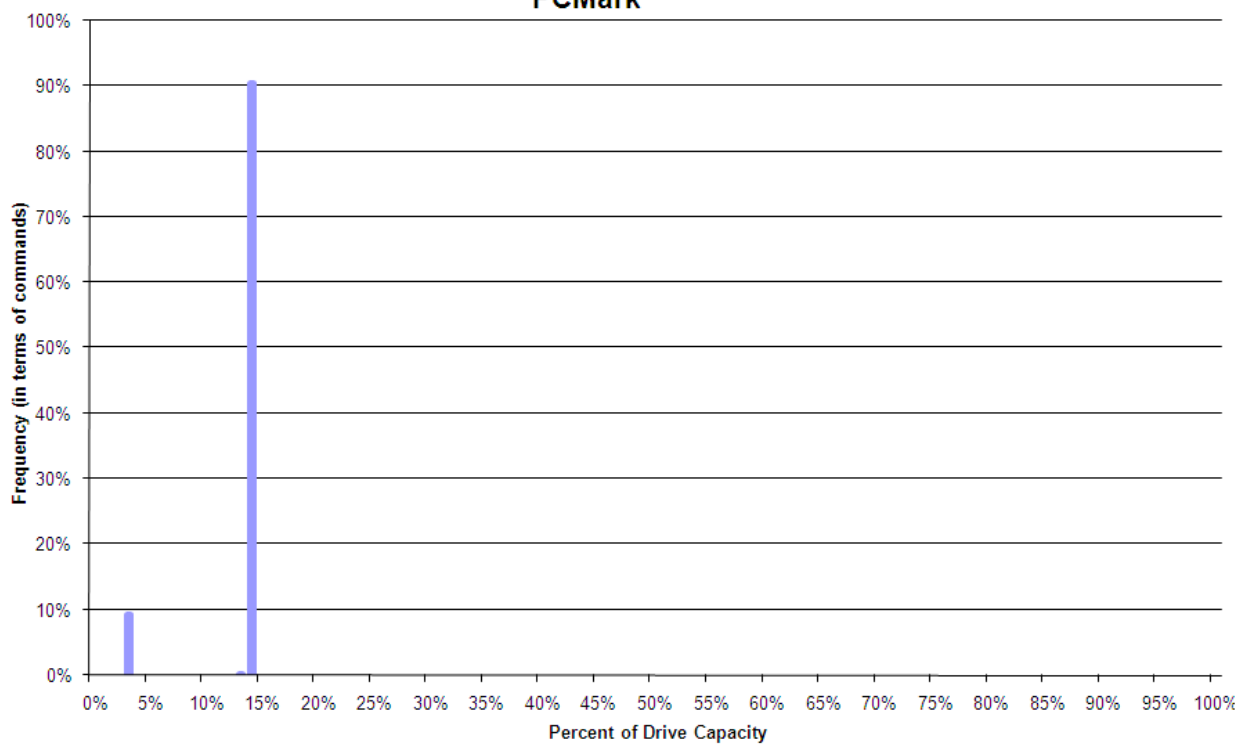


Figure 5a

Percent of Sequential vs. Footprint PCMark

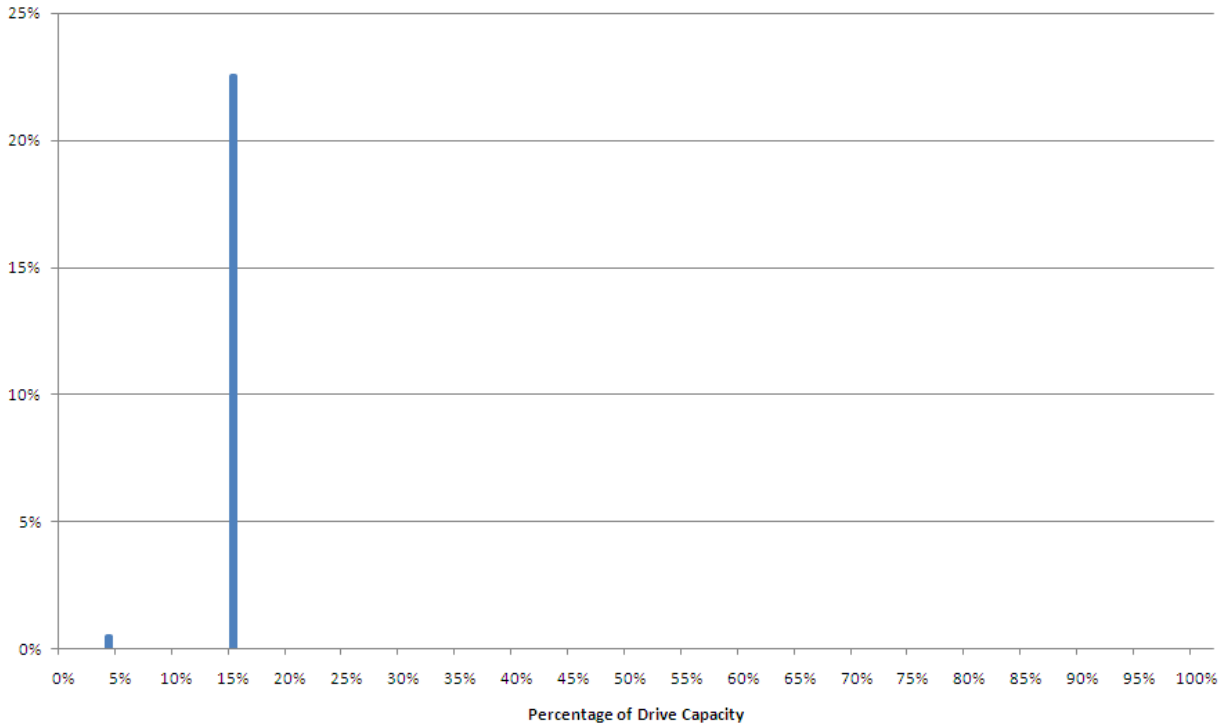


Figure 5b

B.5 Flush Intervals

Flush command reduces the endurance benefit that write caching can provide, as discussed in 2) of Section 3.2. In our user sample trace, there are 60,815 flush commands. There is one flush command at the very end of the trace which comes after about 80,000 writes. Excluding that, the average number of writes between flushes is about 6.4, in which case the flush commands are issued frequently enough that endurance benefit from write cache is kept very low. Of course, write caching still has performance benefit, but not much endurance benefit in this environment.

In the PCMark trace, there are a total of 1018 flush commands. The average number of writes between flushes is about 13, with 360 being the maximum number

Appendix C – How OEM Might Arrive at an LDE Value

This is an outline of a scenario whereby an SSD vendor might arrive at a publicized LDE value.

1. First, the engineering team, which knows intimately the characteristics and behavior of all the components that go inside the SSD, would calculate an estimated LDE value. It can be as simple as Equation 1, but likely will be something a bit more sophisticated. Call this LDE value X.

2. Next, the testing process described in Section 6.3 is performed on a sample population of the SSD (sample size determined by the OEM) at each LDE values of $X-Y$, X , and $X+Y$ (Y determined by the OEM)
3. Results of testing
 - a. If 100% pass X , and $(100-z)\%$ pass $X+Y$, then the estimated LDE value of X is good and can be published
 - b. If only $(100-z)\%$ pass X , then estimated X is too high. Set $X=X-Y$ and go back to step 2 to redo testing.
 - c. If 100% pass $X+Y$, then estimated X is too low. OEM can go with X and publish it, or if it wants it can set $X=X+Y$ and go back to step 2 to redo testing for a higher rating of LDE.