How ESS Reduces Write and Data Amplification and Increases Random Write Performance

A White Paper

Abstract

The patented write mechanisms used by WildFire Enterprise Storage Stack (ESS) to update NAND Flash are significantly different from those used in most consumer and enterprise storage environments. This difference results in significant increases in storage life and random write performance when compared with commonly used update mechanisms. The improvements in storage life and performance are often more than an order of magnitude. This paper discusses the evolution of Flash update mechanisms generally as well as the specific write reduction and data reduction mechanisms used by ESS to reduce write and data amplification and increase write speed, and the practical advantages and consequences of the ESS design. Having discussed methodology, it concludes with a look at the high degree of cost reduction which the ESS design imparts, especially as software which can be wedded with off the shelf computer storage hardware.

Basic NAND Flash Update Mechanisms

When NAND Flash was originally introduced in the 1990's, it entered existence as a simple block device where each erase block could only be written from beginning to end. In this initial period, erase blocks of Flash were very small: typically only 16KB or 32KB long. Each block represented a range in the total logical size of the device. The first step in random writing was the erasure of a free erase block for update. Because in-place updates are not allowed, or have restrictive rules, the first write step was to write any unchanged data preceding the logical start of the new data. This was followed by the drop-in of the changed data. Finally, if any unchanged data followed the just written new data, this was also written.

Erase Block consisting of 8 4KB data blocks		
Copy of Existing Precedent Data, if any	Logical Updates of Data, in sequential order	Copy of Existing Susequent Data, if any

Figure 1 – Primitive Update of Flash Erase Block; Update Flow is Left to Right

While this block update generated write amplification with most writes – for instance, a 4KB write to a 32KB erase block would always result in a 32KB write of the entire erase block – this methodology worked for two reasons. First, many of the writes would have only small write amplification – for instance, a 16KB write would inherently have only 2:1 amplification. Similarly, significant numbers of writes would have no write amplification at all. Second, the Flash erase blocks of the time were highly durable single-cell devices and typically had more than 100,000 erase cycles of life engineered into them, reducing the practical impact of write amplification.

Hardware-Implemented Write Reduction

In order to make Flash less expensive, Flash manufacturers have followed several strategies. In general, these all involve shrinking the amount of silicon required to store an amount of data. Early Flash at 70 nm SLC cells have evolved to 15 nm three bit per cell (TLC) designs and now three dimensional techniques further increase the density of storage. This increased density lowers cost by allowing more storage in the same amount of silicon. At 70 nm SLC vs 15 nm TLC, the amount of "volume" of a cell per bit has decreased by a factor of about 300. This literally translates to 300 times the capacity at the same manufacturing cost.

The increase in flash density has had side effects. First, smaller cells mean a smaller amount of "charge" representing a bit. With original SLC, 30,000 or so electrons represented a bit. This is now under 100 electrons. At these scales, the effects of electrical noise and even quantum physics make signal processing more challenging. Older design only required simple single bit error correction. New designs require layers of error correction that push the limits of todays technology. Smaller geometry has also reduces the number of overwrites that a cell can survive. Endurance of 100,000 is now often below 1000, and 100 endurance designs have been proposed. Finally, the smaller design has required larger and larger logical erase blocks. A current 3D TLC design lists the erase block as 24 MB. But the actual erase block is much larger after error recovery is considered. These new designs often incorporate "internal RAID" (sometimes called RAISE) so the 24MB quickly becomes 360 MB after you multiply in the RAID stripe size.

With these changes, simple FTL designs start to exhibit staggeringly high write amplification.

This amplification problem is reduced by the SSD makers in several ways. One, comparatively simple solution involves incomplete writes instead of full erase block writes. Instead of filling an entire erase block with data in logical and linear order, a percentage of each block was set aside as unwritten at the tail of the erase block. This tail was then filled in sequentially with individual updates for the logical range. When all of the tail was filled with changes, the contents of the tail were reintegrated as part of the logical linear chunk, with a new empty tail. Two diagrams showing the variation possible with this method are shown below.

Erase Block consisting of 128 4KB data blocks	
116 sequential data blocks	12 updates

Figure 2 - Typical Consumer Flash Layout

In Figure 2, above, each of the 12 4KB updates is filled in one after the other, in accordance with Flash PE update rules. When all 12 updates are in place, and a 13th comes along, an update will incorporate the 12+1 changes into the 116 sequential blocks into a new erase block. This will cause a theoretical write amplification for the erase block of 128/13 or 9.84:1 write amplification because 128 blocks (a full erase block) will have to be written for each 13 new blocks incorporated into the logical segment.

Similarly, if we increase free space to approximately 30% of all space, we might end up with an example like the following Figure 3:

Erase Block consisting of 128 4KB data blocks	
90 sequential data blocks	38 updates

Figure 3 - Typical Enterprise Flash Layout

In Figure 3, 38+1 updates must occur sequentially before a PE cycle occurs. In such an example, write amplification will be reduced to 128/39, or 3.28:1. Setting aside more free space obviously reduces wear in a significant way, but increases the cost of each unit of usable flash by 29% compared with the example in Figure 1.

The general advantage of this design is that it is totally deterministic. If one has a given percentage of free (tail) space, then write amplification will be a precisely known result.

Variants of Hardware-Implemented Write Reduction

The empty tail FTL (Flash Translation Layer) model is not the only hardware implemented FTL. There are many variants on this basic empty tail theme. For instance, some designs may use larger individual pools of free space. Others may use a ring buffer design of some sort. But such alternative methods provide only marginal reduction of write amplification when compared with the basic empty tail approach.

How ESS Changes the Random Write Process

ESS is a write process covered by multiple patents. Instead of the commonly used logical segment and tail system described above, the ESS writing system takes clusters of random writes, and builds these into a linear string, which is further surrounded by control data used to rebuild the memory table database, should this be necessary. These atomic clusters are then written sequentially to a large scale write block in sequential order. (A write block can be a single erase block, or multiples, or fractions of erase blocks depending upon circumstantial needs. A write block is also likely to land on RAID-stripe boundaries in order to maximize RAID efficiency.) The advantage of the ESS design is four fold.

First, as ESS does pure and perfect linear writes in all cases, its “random write” speed is the composite linear write speed of all of the SSDs managed by ESS. This is normally vastly faster than is possible with traditional random write methods. In some products, the speed increase can be 500-fold or more.

Second, ESS never overwrites existing data, and always writes all new data in the order received. Conversely, the empty tail approach can overwrite existing data, and lacks certainty of update in order, which can lead to broad data corruption in failure environments.

Third, ESS fills full write blocks, but stores all data in time-received order rather than storing it in logical sequential order, as is the case of the logical segment and tail discussed above. Because storage is in time order, each data element in the write block expires when it is superseded by newer data referencing the same logical block in another, newer, write block. Obsoleted data is not immediately deleted. Instead, each write block is retained in its original state until the write block in question becomes the most empty of all the write blocks in the data set. At such a point, the still-valid contents of the write block are scraped into a new, empty write block and the old block is flagged for erasure and use in storing new data changes.

Finally, ESS can run on top of existing block devices. The write nature of ESS does not have to be implemented “in the SSD”. It can live on the host feeding writes to stock SSDs that use other methods. This “host side” implementation has further benefits in terms of increasing performance, lowering wear, and protecting data.

For various reasons which will be explained in detail in the rest of this paper, the most empty erase block tends to become extremely empty. Typically the block will be 80% to 90% empty, and thus will have only 10% to 20% of its contents being recycled. This will result in write amplification typically less than 1.5:1 whether the SSD set being used is a Consumer, Enterprise, or Read Intensive SSD.

This extremely low write amplification number, when combined with ESS’s ability to reduce the data amplification of RAID-sets, and to also reduce data size through compression, results in extremely long-lived media that can accept 15x to 30x more original data than was possible with the bare SSD set itself. How this is achieved will be discussed in detail as this document progresses.

The ESS Data Structure

Instead of associating a particular range of logical data blocks with a particular Program/Erase (PE) block or block-set, and preparing that group for the acceptance without PE cycle of further random writes (i.e. Figures 2 and 3), ESS writes all incoming and changing data in time-order received to an empty write block which consists of one PE block or a group of related PE blocks.

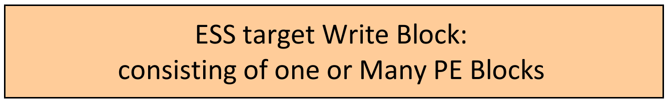


Figure 4 – Data Targets

This target write block can be the length of a single PE block, or it can be multiple - a linear set of PE blocks in a single storage device. It can also be a further multiple targeting a particular RAID format such as a set of SSDs configured RAID-0, RAID-5, or RAID-6. Use of multiple PE blocks, rather than single PE blocks can be advantageous in reducing mount times and in reducing wear.

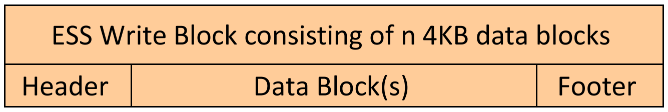


Figure 5 - Basic ESS Data Structure

As shown in Figure 5, all writes to the target write block are atomic, in that each write chunk must consist of one or multiple data blocks, together with a header and footer which collectively hold all the control information including time stamps, and the data-related metadata defining the datum's exact physical and logical position, as well as state references of each data element present in the write chunk.

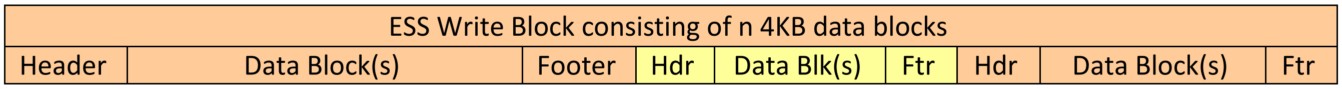


Figure 6 - ESS Data Structure with Multiple Segments

As shown in Figure 6, the actual size of a chunk written can be less than a whole write block, and the chunks can be of variable length, provided that each chunk is written sequentially and provided the sum of lengths of all chunks matches the size of the write block. For instance, in enterprise usage with multiple SSDs, it is common to write chunks which are exactly one RAID stripe in size even though the target write block may be many megabytes long. As write activity increases, chunk size will increase to improve write efficiency, up to the point where chunks are the same size as write blocks.

Write activity is mirrored in the controller device and memory, where RAM tables keep track of the present physical location and state of each individual logical 4KB block addressable by the system.

The ESS structure has two primary advantages. First, because all data and control information is written in a sequential manner from beginning to end of PE blocks, and sequential writing is normally faster, ESS writes very quickly, at speeds that are often 10 times faster than the random write speed of the target device. Second, because ESS writes all change in a single linear stream it inherently reduces wear and has ways of reducing wear which are not available to standard structures.

A Graphical Exploration of ESS Usable Free Space and Wear

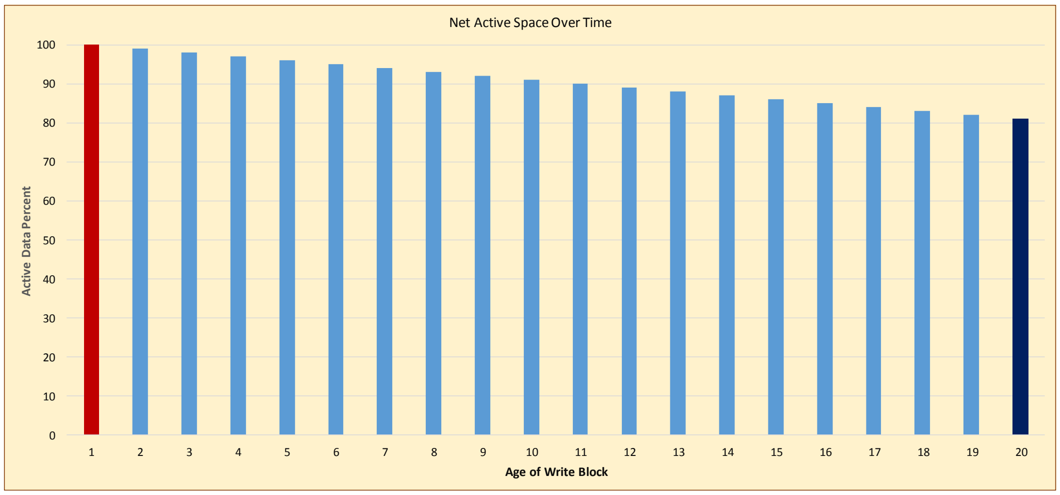


Table 1 – How Random Expiration of Data Impacts Free Space In the Oldest Write Blocks

Traditional FTLs with 10% free space generally write updated PE blocks which are 90% full of unchanged data and have 10% free for collection of changes. Thus, we can say that for all PE blocks in such a device, each erase block will have a write amplification of about 10:1 and that all will be similar to block 11 in Table 1.

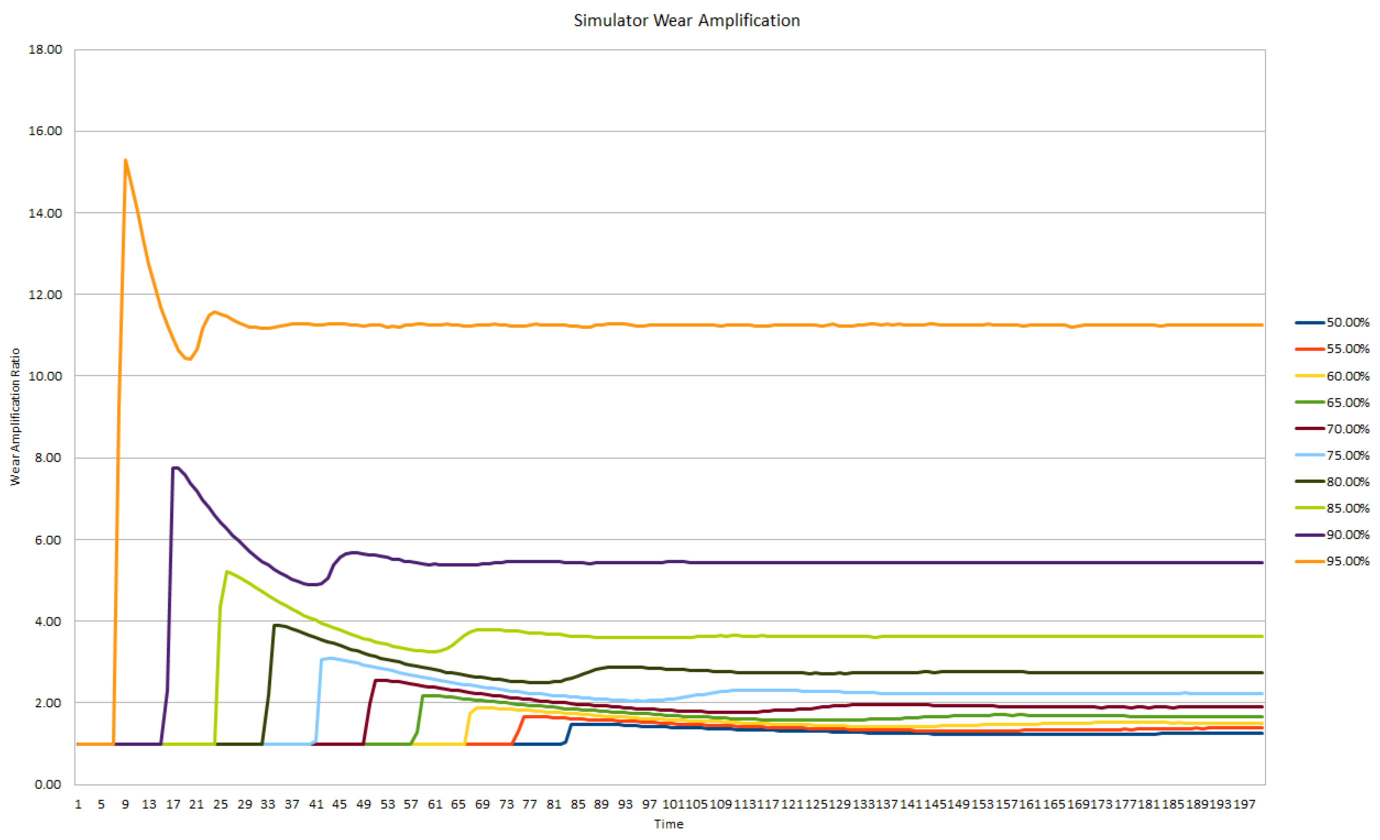
ESS inverts this and writes PE blocks that are 100% full of active data, as represented by the red line in column 1 of Table 1. As additional blocks are written with new active data, they deactivate individual segments of data in previously written write blocks. In a purely random write environment, these deactivations will evenly distribute across all existing elements. Over time, the total size of each previously written block will shrink consistent with the sum of all free space (i.e. 10%) but the oldest (20 - purple) will have shrunk 19 times more than the most newly aged element (2) because it has had longer to age. The most empty is therefore harvested at a point that is approximately twice as empty as the average block (i.e. 11).

This condition is repetitive. Once block 20 is harvested and a new block 1 is added, what is currently block 19 will have approximately the emptiness of what is currently block 20.

Pure random update is the most stressful to an SSD. Non-random events such as the impact of static data or time-based data replacement (which will be discussed later) will create a less random environment, and hence reduce write amplification.

Baseline Write Amplification in ESS Is Half the Norm

We can model ESS write amplification over time by using a Monte Carlo simulation. Graph 1 shows average ESS write amplification over time based upon the allocation of free space between 5% and 95% of all storage in the system. As we can see, at 10% free space (the purple line), over time write amplification will average out to about 5.6 to 1, including the metadata allowances described above. This graph also shows the peculiar write amplification double-hop of the ESS environment. These hops occur because early on in the update process, the system moves from perfect, no-amplification, equilibrium to a point where all possible write blocks are near-full and little difference in fullness exists. Under such circumstances where the emptiness of the most-empty write blocks is much lower than the long term average, we end up with the hop effect, though this effect is generally less than the 10:1 write amplification shown in Figure 2.



Graph 1 – Free-Space Impact of ESS on Write Amplification Over Time

ESS Amplification Reductions Caused by Inactive Storage

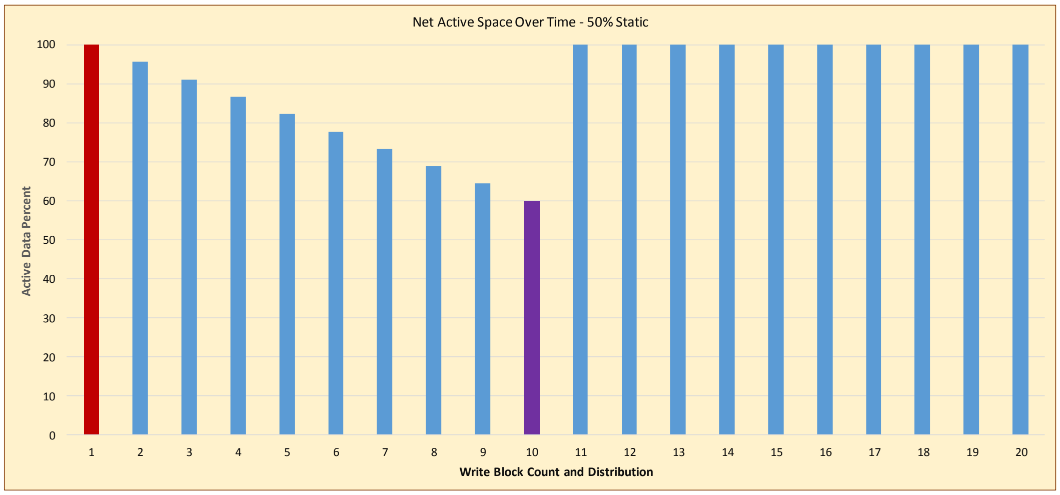


Table 2 – ESS Increases Usable Free Space In the Presence of Static Data

Some storage is written once but never changes. Examples of such data would include programming and operating system files, configuration files, and image files such as sound, picture and video files. In most environments where these might be stored in flash storage, such static images can easily consume 20% to 50% of the data stored in the environment. Even the primary 128GB JEDEC trace for testing consumer SSDs – a trace actually generated from 8 months of activity by a single individual – actually contains more than 60% static data.

In an environment where some of the space is static, the available free space is applied only to the dynamic space. Accordingly, as exampled in Table 2, if 50% of the space is static, all the free space will be distributed into the dynamic space, forcing a shift in the slope of the rate of emptying. Accordingly, in such an example, where the most-stale write blocks previously contained 20% stale data, the same block will now have approximately 20% average free space and 40% stale data at the point of garbage collection. A practical level of 20% free space, as can be seen in the evergreen line of Graph 1, which would have an inherent write amplification rate of 2.6:1. Conversely, as Figures 2 and 3 establish, there is no advantage from static space in the regular logical chunk plus tail method, or its variants.

How ESS Enables Trim() and Dynamic Free Space

ESS stores any space that has null data content, or that has content which is all HEX00 or all HEXFF as virtual space instead of real space. Here, both primary memory and the metadata (see Figures 5 and 6) contain references to the logical block, including its state as a virtualized block, but no data from the block is stored in the data section. Accordingly, the individual write block describes more logical blocks than it otherwise could, while consuming no more space in practical terms. This design approach results in gains in three areas:

First, ESS fully supports trim(). This is important in Consumer products as most client devices, including Windows and Linux client devices, support trim() at the client level. It is obviously less important at the Enterprise level. While trim() supported in file systems such as XSF and EXT4, as many enterprise usages such as SANs and iSCSI do not yet support trim(). Trimmed space becomes part of the dynamic free space pool and thus reduces average wear. Trim(), where supported, will often halve wear amplification by itself, when managed by ESS.

Second, to the extent that the storage media stores less data than the physical storage, the unused storage also becomes dynamic free space, reducing wear also. In a general sense, many systems have lots of unused space early on, and most systems have at least some excess to needs space for the life of the system. Space which is consumed and then deleted at the client level without notice to the storage device can also be recovered by creating a series of files which contain all-zeros (HEX00) and then deleting the same. All this, as well as trimmed space will automatically be employed by ESS to reduce write amplification.

Third, to the extent that functions such as compression are used without expansion of the logical space, any compression which occurs also results in increased free space and reduced defragmentation, as will be discussed in the compression section, below.

Even small gains in free space can make major differences in wear. For instance, if we are capable of maintaining 10% mandatory free space together with 5% dynamic free space through all means, then (as indicated by the "Simulator Wear Amplification" graph (Graph 1) above, we would have total free space equal to about 15%, and wear amplification would fall from 5.65:1 to about 3.8:1. Similarly, as will be discussed in the compression section, a compression level of only 5% with no increase in logical space would have the same impact on total free space.

How ESS Produces Time-based Wear Avoidance

If you are managing your system IO through a journaled file system such as XFS, EXT4, or NTFS (almost everyone does), then most of the data you are writing to a storage system is either being deleted or overwritten just a few minutes or hours after it is written.

A typical write to any of these file systems requires 32K to 80K of journal and metadata writes, whether the data write in question is 4KB or 1MB. Accordingly, in an ESS environment, 50% to 60% of all data written to a storage system will inherently be flagged as expired within almost no time, assuming a primary data average write size of <=32KB.

These metadata and journal entries have no impact on traditional Flash systems as described in Figures 2 and 3. But because each ESS write block works backward from full state to most-empty state, individual write blocks can accept data expiration up to the point of total emptiness, the rapid expiry of metadata and journal data results in write blocks which tend to be very empty and thus low impact at defragmentation time.

In addition to journal data and metadata, most systems have a significant quantity of regular data which is updated either with high frequency or with high frequency for a limited period of time. Counters, as a simple example, are items which update with high frequency all the time. Similarly, individual files or database blocks used in an order processing environment may be updated 20 or 30 times over several days as they go through "production" steps, and then sit inactive for months or years. Typically, such high-change will amount to more than 50% of the data writes to the system even though they may only represent a percent or two of all the data write targets in the system.

The key to understanding these types of data deletions is that they are applied only to the newest data: data posted within the last few hours, which is typically only 2% or so of the total data. As ESS stores data on a time-received basis, rather than in logical order as most storage does, and as ESS always writes new data to new open space, the newest write blocks written are those which have the highest rates of data expiry.

When write blocks can regularly become 80% empty or even more, there is very little which has to be scraped from old write blocks into new write blocks. Accordingly, it becomes very easy to achieve base write amplification that is sub-1.5:1 rather than the 3:1 to 10:1 common in Enterprise and Consumer SSDs managed by traditional means.

The next section, which is the summary of some test results shows that radical reduction of write amplification is not only theoretically possible but to be expected in most Consumer and Enterprise environments.

ESS Test Report – JEDEC 128GB Client Trace Test

On 9/15/2015, Doug Dumitru, the CTO of EasyCo LLC performed a number of tests to determine how ESS impacted the durability of SSDs. Dumitru designed and executed these tests by using the standards which JEDEC promulgates in its standard testing procedure: JESD219A, for testing the durability of SSDs. He then prepared the summary and report below:

> As part of the JESD219A procedures, JEDEC publishes a trace of a "standard" client workload for a general use desktop system. This workload was recorded over 8 months and represents actual blocks written on a live system. The trace includes writes, trim, and sync operations. Reads are not included.
>
> This workload is part of a standard JEDEC wear amplification test procedure for SSDs: JESD219A. The test procedure is to 100% fill a target device with linear writes, and then play the trace back to the device. By comparing device wear induced by the trace writes with source data, a wear amplification factor can be calculated.
>
> ESS is a little different from most FTLs in that it is external to the block device. Regardless, the same calculations can work with ESS.
>
> Trim/discard operations are fully implemented in ESS. ESS also implemented writes of zero blocks as synonymous to a trim operation.

Trace sync operations are unnecessary with ESS. ESS always maintains full write chronology for every write and syncs are not necessary to maintain data integrity. Trace syncs are treated as no-ops in this test.

Test Setup: The device was setup with 128 GB of available space (128,002,187,264 bytes). Overprovisioning was set at 10%, so the physical device was 142 GB (142,222,557,184 bytes). This 10% over-provisioning is tunable. 10% is the default that EasyCo recommends for most workloads.

Test Parameters:

Pass 1: Initial Linear fill.

- 128.002 GB written by the linear fill
- 128.501 GB written to the SSD
- 0.39% ESS Meta Block Overhead

Pass 2: Playback of client workload trace

- 35,391,419 total trace write operations, 781.33 GB written, 21.56 KB/write average
- 2,498,963 total trace trim operations, 821.33 GB trimmed, 320.96 KB/trim average

- 781.33 GB written by the trace
- 821.33 GB trimmed by the trace

Test Results:

- 950.27 GB written to the SSD during the trace
- ESS Client Workload Wear Amplification **1.216:1**

Results with other over-provisioning ratios:

• 5% free space	**1.363:1**
• 7% free space	**1.295:1**
• 10% free space	**1.216:1**
• 20% free space	**1.111:1**

Results without trim:

• 10% free	**1.437:1**

This shows that trim is effective, but even without trim ESS still produces compelling wear results.

Comparison among Consumer SSDs: Not all SSDs have adequate SMART reporting to see how their FTLs compare. Of the SSDs in EasyCo's LAB, only one model is both small and has this level of SMART reporting. This is a Crucial M500 240GB SSD. Because the capacity is larger than the 128GB test trace, this is not a 100% valid comparison (it should actually advantage the Crucial M500), but the results should be interesting never the less.

The SSD was secure erased and then the SMART attributes "Host Program Page Count" and "Bckgnd Program Page Count" were used to calculate a wear amplification ratio. The overprovisioning ratio of this SSD is not known. Trim operations were implemented by shelling the 'hdparm –trim-sector-ranges …' command. Actual device writes were done from a C application with the device opened in O_DIRECT mode.

240GB M500 Client Trace Write Amplification of the Bare SSD: The trace took a very long time to run. Where the other tests were performed in about 40 minutes each, this test required more than 24 hours. This is partially due to the overhead of hdparm, but largely due to a very high wear amplification ratio:

- M500 Client Workload Wear Amplification without ESS **9.652:1**

There is nothing "inferior" about the result. Virtually all Consumer grade SSDs have write amplification of around 10:1. One of the things which makes the M500 superior among this class of SSD is the presence of a precise and informative SMART system of registers.

240GB M500 w/ ESS: Just as a sanity check, the test procedure was used with the M500 SSD as the backing SSD. In this case, the M500 reported, via SMART, wear amplification of about 1.01:1. This confirms that ESS's drive-level workload is basically a perfect workload for SSDs.

A Summary of Write Amplification Expectations Created by ESS

In a client or consumer use of ESS, write amplification will nearly always be reduced below 1.5:1 without compression being employed. Typical write amplification for these settings is around 1.2:1 without compression and 0.7:1 with compression. This conclusion is based both upon use of a JEDEC 128GB Windows Client trace run against both actual media and the Monte Carlo simulation previously referenced with 10% mandatory free space. In the following list, we analyze the impact of each function in the JEDEC 128GB trace by working back from hard tested numbers: trim() support with write amplification of 1.2:1 and write amplification without trim() at 1.4:1. We similarly determined static space to be 62% through a bitmap of all writes and determined the impact of wear avoidance by computing the difference. (In a Journal file system, wear reduction will almost always exceed 50% due to the journal effects. Then the question becomes the degree of repetitive data hits.)

Function	Factor	Wear Mult
Baseline wear, based upon use of 1MB write block	5.65	5.65:1
62% set aside for static space	.43	3.00:1
80% journal, metadata and data time-based wear avoidance	.20	1.40:1
10% dynamic free space created through trim() increases total free space by 50%.	.50	1.20:1

Table 3 – Analysis of Write Amplification Reduction Components in ESS

The same conclusions will apply to Android devices. Such devices will have lots of static space, though perhaps less than Windows due to typically smaller total storage size. Wear avoidance through repetitive writes both due to the behavior of journals and the inherent function of databases will be common. And trim() is supported in Android.

Predicting benefits in enterprise environments is harder. We have seen many ESS-managed Enterprise devices over the years with write amplification levels that are around 1.3:1. Clearly, these systems will have baseline wear of around 5.3:1 due to the large write block size used in multi-SSD servers. Also clearly, the elements of static space, repeated rapid overwrites of data, and dynamic free space will have a significant impact even if trim() support is less than universal. But other enterprise environments will have higher wear amplification levels because the way they manage data cannot take advantage of trim(), dynamic free space, wear avoidance, or static space. It is safe to say that all storage systems used for VDI will have write amplification of about 1.2:1 because windows environments support trim(), and virtual systems will have write amplification of 1.4:1 because trim() support will be limited. Most other storage uses using a broad mix of files will have write amplification of about 1.4:1 as well.

The only obvious exception to this is of high speed storage used for video processing or rendering. Here, we deal with data not normally compressible (see data reduction, below) and which has metadata and journal data which are very small. Accordingly, time-based wear avoidance as used in other environments is not applicable. On the other hand, there are other techniques which will contextually reduce write amplification near to 1:1. .

How ESS Deals with Large Data Elements

There is one exception to the methods described above: situations where the storage of very large scale records is the predominant purpose of the storage server. Complex formatted documents, sound streams, and simple digital pictures can be stored on media as kilobyte to megabyte elements. But video streams can require huge amounts of data. Individual uncompressed 4K video frames can consume 50 megabytes of storage, and a second of uncompressed video requires more than a gigabyte of data space.

On the one hand, newer SSDs with ring buffers or consolidated update tails have relatively low write amplification. Similarly, they do not encounter all of the parity RAID penalties (see RAID, below) because they generally span RAID stripes. As a result, write and data amplification even of Consumer or Read Intensive SSDs is limited to somewhere in the range between 2.5:1 and 3.5:1 for just about any kind of media, including Consumer and Read Intensive media.

On the other hand, ESS based systems writing mostly ultra-large writes quickly lose the ability to apply some of their methods. For instance, time based wear avoidance does not apply because metadata and journals are quite small relative to the size of the data written. In addition, the data itself is unlikely to expire or be overwritten rapidly. In like manner, standard data compression may not be applicable to some types of large records.

There are significant areas where ESS does improve speed and durability of large scale writes. ESS produces a pure, linear, time based stream which maps one-to-one against logical space. In this stream, there is careful focus on RAID and erase block boundaries. In addition, modifications are made to the basic Linux RAID-5 and RAID-6 code to avoid premature execution which causes de-facto random writing of RAID stripe content. Ordinary systems lack this continuity, and as such are generally mapped to free space and then later are integrated into a pure logical space stream. The general effect of all these changes is to produce a linear stream which runs with wear amplification of 1:1 on the basic set, together with a further +/- 1.1:1 to cover the data amplification of RAID stripes. This leads to the conclusion that ESS is approximately 2:1 to 3:1 more efficient than stand-alone sets of SSDs when both are accepting large scale data such as video streams.

The above consideration of both ESS and native large scale writes is based upon the assumption of a single stream of large scale writes. Mixing multiple update streams together has negative impacts for both native and ESS operations. On the native side, significantly more write amplification will be needed to map multiple streams through free space and into logical segments. On the ESS side, multiple write streams do not negatively impact speed or durability as long as the device is merely a consolidator of streams without much update. However, if the system is used for rendering, a single stream and single operator will result in significantly higher operating speeds and reduced write amplification. Arguably, one storage unit per operator may also result in lower construction and operating costs as well.

The Power of Data Reduction In ESS Environments

Data reduction is the process of reducing the amount of data written to a Flash array by reducing the size or quantity of data written. At this point, there are two ways in which ESS can reduce the data written. The first is to compress the data using standard compression tools. The second is to reduce the amount of data written to RAID-5 and RAID-6 drive sets.

How Compression Is Used by ESS

Normally, compression such as the LZ4 used by ESS, is only used to compress data being stored. While data compression is effective, most servers only achieve 20% to 30% compression of such data.

Surprisingly, however, compression of the journaled data and metadata of file systems is highly compressible: when primary file systems such as NTFS, EXT4, and XFS are studied, one discovers that this data, which represents 50% to 60% of writes when source writes average 32KB to 64KB in size, and which is rapidly deleted or replaced (see the section on time-based wear avoidance, above), also has extreme compressibility ranging from 80% to 90% depending upon the file system used.

When we use the mean value for such file systems, we realize that while 45% of all writes may be only 25% compressible, the remaining 55% will be 85% compressible, such that average compression will rise to (45% * 25%) + (55% * 85%) or (11.25% + 46.75%) for a total typical compressibility of 58%.

Such a level of compressibility can be used to create additional free space, which is useful but has administrative and monitoring requirements, or it can be used passively to significantly reduce the quantity of data actually written to a system, consequently increasing the life of Flash arrays when measured from the perspective of source data size. ESS installs default to this second path due to its clear advantages.

ESS-Managed RAID-based Write Reduction

To the extent that data written to a RAID-5 or RAID-6 drive set is written as a full stripe, instead of being written randomly, the amount of data is reduced from a 2x multiple in the case of RAID-5 (and 3x in the case of RAID-6), to the ratio between the parity drive(s) and the sum of data drives present. For instance, a 24 drive set written to RAID-6 as full stripes will have data amplification of only 24/22 or 1.09:1. ESS supports such data reduction due to its inherent linear writing method.

ESS and Deduplication

As of the writing of this white paper, ESS does not support deduplication. While we intend to do so reasonably soon, there are a number of issues that impact the viability of deduplication:

1. Deduplication typically only works well in virtual systems environments where large amounts of programs and stock data are being stored on a single drive set. By comparison, storage of video, compressed, encrypted, or single instance data gains virtually no advantage from deduplication.

2. Deduplication normally requires a great deal of RAM memory for mapping purposes, and related system expense, unless one can develop a program which can use flash media as a memory store. WildFire has such a design.

3. Deduplication requires the creation of logical space in excess of the physical space available on the system. Accordingly, people must monitor the storage environment to make sure they don't run out of physical space. In addition, clean up of space-critical systems can be difficult because duplicate items have virtually no impact on free space when removed. Only substantial non-duplicate items do.

4. Because WildFire permits the use of less, and less expensive SSDs than do other systems, the economic advantage of deduplication is limited in an ESS framework. Deduplication also is subject to diminishing returns, and returns negligible economic advantage in a low cost SSD environment after the first or second expansion of logical space. Deduplication can only offer significant value when premium SSDs are used.

Peak Write Performance of ESS

Peak random and linear write performance in an ESS environment is typically within a few percent of the sum of the published linear write speed for all devices present. Accordingly, a device with 24 SSDs, each running at 500MB/sec, is accordingly about 12GB/second. Similarly, peak random write performance will be close to 3 million IOPS if the system is engineered and configured for this purpose. This peak performance level will fall if the number of SSDs is decreased, if RAID-6 is used in lieu of RAID-5, or if data reducing features such as compression are active. Even with all features active, peak performance remains significant. With adequate compute performance, a 24 SSD system with enhanced data reduction and parity RAID will typically have peak performance in the 1.5 to 2 million 4KB random write IOPS range, and an 8 SSSD product will typically have a peak rate in the 0.5 to 0.75 million IOPS range.

By comparison, the peak performance of industry standard functionality is much lower. For instance, Flash managed through a smart disk controller, RAID-5, typically has a random write rate of only 10,000 IOPS. Similarly, the base product of a major SAN/NAS provider using RAID-5/6, achieves a combined peak read/write rate with a 70% read level of only 35,000 IOPS, which computes to a random write rate of about 20,000 IOPS. Some variants of parity RAID can achieve 60,000 IOPS, and RAID-10 will do a little better than this, but typically peaks out at 150,000 or so, when used with Enterprise media.

Peak write performance is most directly important when ESS is used locally without networking. With a network, ESS IOPS rates are significantly influenced by packet transfer times. For instance, a single ten gigabit Ethernet interface will process about 40,000 4KB random reads, and 75,000 random writes. However, both read and write channels will saturate at higher block sizes. Clearly, there is room for significant growth in total throughput either through the use of additional ports, or through the use of 40 gigabit, and faster, Ethernet ports. When supported, Fiber Channel can also be used to increase IOPS. By comparison standard RAID-5 through a 10 gigabit port, is limited to about 11,000 IOPS and multiple interfaces do not increase throughput by much.

Worst Case Write Performance with and without ESS

Many flash devices go into the weeds when they are forced into garbage collection mode. ESS managed devices generally do not. The reason is simple. Lower write and data amplification means that there are fewer internal reads and writes which need to be performed to collect clean free space write blocks. Reduction of internal reads and writes results in significantly higher worst case random and linear write rates.

Read Performance in ESS

Individuals looking at wear reduction may be impressed with wear gains but be concerned about read performance. Here, there are two answers: minor and major.

Based upon testing, absolute read performance with ESS will probably decline 2% to 4% versus the devices run with native IO software. On the other hand, ESS will read faster in some cases because of its ability to queue multiple read processes. Flash drives read far more efficiently when there are a number of read requests stacked up than when there is only a single linear thread.

The more important fact is that ESS radically improves effective read performance through enhancement of write performance. Consider the performance of a device that is capable of performing 10,000 4KB random reads a second, but only 1,000 random writes. Given a 70% read, 30% write mix, the device would only deliver about 1,900 combined IOPS, and only 1,330 reads per second because 867 milliseconds per second are needed to perform the 30% random writes required.

Then consider the same read performance matched with write performance capable of doing 20,000 random 4KB writes a second. Typically, ESS managed systems random write several times faster than they random read. In the second example, overall performance would be 11,765 IOPS, and random read IOPS would actually exceed 8,230. Overall, practical read performance would increase 6.2 times. High performance reading is useless if write performance is so poor that little time is available to perform the random reads requested.

The above ratios understate the importance of random write speed because RAID-amplification as well as the data amplification of journaled file systems lead to a situation where a system accepting 30% random writes is actually executing 70% within the RAID and journaling structures.

A Recapitulation of ESS Performance with Observations

Table 4 looks at the comparative durability of different classes of SSDs (i.e. Consumer, Read Intensive, Data Center and Enterprise), and then re-calculates the durability of the same classes of SSDs when these are managed as a set by ESS software.

The Flash SSD industry over time has developed durability standards for most of its media classes. These are generally expressed as over-writes per day of the media. As a general rule, heavy duty Enterprise drives must be able to accept at least five overwrites a day and still have a useful life of at least five years. Data Center drives, suitable for general purpose Enterprise use, are expected to accept approximately one over-write per day for five years. Read Intensive SSDs are supposed to be capable of accepting 2/10ths of an over-write per day for five years. Consumer SSDs are expected to accept approximately 1/10th over-write per day while lasting five years.

The means by which write life is obtained will vary from manufacturer to manufacturer. Our table shows some older two dimensional technologies and adjusts media type, design erase cycle life, and design write amplification to achieve the durability goals specified above.

There are a number of different construction methods. For instance, Samsung builds both its consumer and Read Intensive SSDs out of 3D TLC memory, but uses MLC for Data Center devices. Similarly, the Micron 5100 series delivers Read Intensive, data center, and enterprise durability levels all in 3D TLC, while adjusting free space to reach the durability goals.

8 SSD Set, RAID-5

	Native RAID Set without ESS				**RAID Set Managed by ESS**			
SSD Type	Consumer	Read Intensive	Data Center	Enterprise	Consumer	Read Intensive	Data Center	Enterprise
RAID Set Drive Count	8	8	8	8	8	8	8	8
Media Type	TLC	MLC	MLC	pSLC	TLC	TLC	MLC	pSLC
Useful Life Erase Cycles	2,000	3,000	5,000	30,000	2,000	3,000	5,000	30,000
Write Amplification	10.00	9.40	3.00	3.00	1.43	1.43	1.43	1.43
5-year Overwrites per Day	0.110	0.175	0.913	5.479	0.766	1.150	1.916	11.495
RAID set type	RAID-5	RAID-5	RAID-5	RAID-5	RAID-5	RAID-5	RAID-5	RAID-5
RAID-5 Amplification	2.00	2.00	2.00	2.00	1.14	1.14	1.14	1.14
RAID-5 Overwrites per day	0.055	0.087	0.457	2.740	0.671	1.006	1.676	10.058
Compression rate	0.00	0.00	0.00	0.00	0.58	0.58	0.58	0.58
Overwrites per day with compression	0.055	0.087	0.457	2.740	1.597	2.395	3.991	23.949

Table 4 - Summary of Amplification and Overwrites

The calculation method for calculating overwrites per day is very simple. Take the useful life (2,000 erase cycles for TLC Consumer grade memory), and then divide it by the write amplification, days in a year, and five years. This gives overwrites per day. If the set is then managed as a RAID set, one next divides by the RAID Amplification of 2 for RAID-5 and 3 for RAID-6. RAID amplification levels are lower for ESS managed hardware.

In the case of ESS values, a write amplification of 1.43 is chosen as this is a tested value for VDI and Virtual Systems, which normally do not have trim() support. Similarly, because ESS writes in RAID stripes and thus writes parity as a proportion of all writes rather than as a supplement, the amplification is all drives divided by data drives.

Regular Linux does not support compression of data though a few file systems such as ZFS do. The ESS compression value is applicable for VDI, Virtual Systems, and similar systems but is not suitable for systems managing large individual files such as video and voice files. Many such files are not compressible. Because of their large size, the proportion of journal and metadata files is extremely small. In such cases, compressibility is probably between 0% and 10%.

While traditional technology varies in its result, ESS technology is equally applicable to Consumer and Enterprise SSDs, and everything in between. Even in the primary analysis, Consumer and Read Intensive SSDs managed by ESS are as durable as Data Center SSDs without ESS, and become significantly more durable when data reduction techniques such as linear writing of RAID stripes and data compression are employed.

For convenience, the same values are shown for a 24 SSD drive set, RAID-6 in Table 5.

24 SSD Set, RAID-6								
	Native RAID Set without ESS				**RAID Set Managed by ESS**			
SSD Type	Consumer	Read Intensive	Data Center	Enterprise	Consumer	Read Intensive	Data Center	Enterprise
RAID Set Drive Count	24	24	24	24	24	24	24	24
Media Type	TLC	MLC	MLC	pSLC	TLC	TLC	MLC	pSLC
Useful Life Erase Cycles	2,000	3,000	5,000	30,000	2,000	3,000	5,000	30,000
Write Amplification	10.00	9.40	3.00	3.00	1.43	1.43	1.43	1.43
5-year Overwrites per Day	0.110	0.175	0.913	5.479	0.766	1.150	1.916	11.495
RAID set type	RAID-6	RAID-6	RAID-6	RAID-6	RAID-6	RAID-6	RAID-6	RAID-6
RAID-6 Amplification	3.00	3.00	3.00	3.00	1.09	1.09	1.09	1.09
RAID-6 Overwrites per day	0.037	0.058	0.304	1.826	0.702	1.054	1.756	10.537
Compression rate	0.00	0.00	0.00	0.00	0.58	0.58	0.58	0.58
Overwrites per day with compression	0.037	0.058	0.304	1.826	1.673	2.509	4.182	25.089

Table 5 - Summary of Amplification and Overwrites

In a RAID-6 environment, with or without more SSDs, ESS management is relatively more beneficial.

How ESS Lets You Radically Reduce the Cost of Flash SSD Mass Storage

ESS lets you build high performance all-Flash storage using a RAID-5 or RAID-6 topology that has write speeds superior to RAID-10. This almost halves the number of SSDs you need for a given unit of storage and performance. As important, ESS reduces write amplification and data amplification in your system. The reduction is more than a factor of ten in some machines and environments. This means that you can buy a lower grade of SSD than you might otherwise have done. The saving may be substantial, and can halve again the money you need to spend on high speed mass storage. But these solutions are only possible with ESS.

Table 6 concurrently looks at all three issues discussed in this paper: the cost reduction, Flash life extension, and random write speed enhancement of systems built using ESS technology, and comparison with SSD arrays using traditional RAID technologies. The presentation looks at various models of consumer and read intensive SSDs, and compares the performance of these with data center SSDs managed using traditional mirror or parity configurations.

Information and results are broken down into logical segments. Pink covers general information. Yellow looks at comparative costs. Here, for instance, we see that the least expensive consumer SSDs lead to costs that are less than a quarter of RAID-10 arrays of data center SSDs. But in green, we see that when durability is modified by the various techniques previously discussed, we see that the least durable consumer SSDs can be expected to last longer than data center media, RAID-10, and much longer than parity RAID-6. Finally, when we look at comparative write IOPS in the blue segment, we see that with appropriate media, we can expect ESS-managed RAID-6 to write as much as eight times faster than traditional RAID-10, and eighty times faster than traditional RAID-6.

All of this information is extractable either from public sources or by calculation. The only lines that are questionable are the broad spread of IOPS rates (in blue) and the write amplification rate (green) selected in each case.

Summary of Performance, Durability, and Manufacturing Cost Impacts to an Approximately 20 Terabyte (usable space) Storage Array Both When Managed and Not Managed By ESS

SSD Manufacturer and Model	SSD Class	Managed by ESS?	RAID Type	SSDs Reqd	SSD Capacity	Cost per SSD	Total SSD Expense	Usable Capacity (GB)	Cost per Gigabyte	Erase Cycles	Write Amplif	Over-writes per Day	Internal Random Write IOPS
Crucial MX-300	Consumer	YES	RAID-6	24	1,050GB	270	6,480	20,790	0.31	1,500	0.8	1.03	1,200,000
Samsung 850 EVO	Consumer	YES	RAID-6	24	1,024GB	370	8,880	20,275	0.44	3,000	0.8	2.05	2,000,000
Samsung PM863	Read Intense	YES	RAID-6	24	960GB	480	11,520	19,008	0.61	3,000	0.8	2.05	2,000,000
Samsung SM863	Data Center	NO	RD-10	20	1,920GB	1,254	25,080	19,200	1.31	5,000	2.5	1.10	250,000
Samsung SM863	Data Center	NO	RAID-6	24	960GB	680	16,320	21,120	0.77	5,000	7.5	0.37	25,000

Table 6 - Summary of performance, durability and cost changes

The blue line for ESS managed SSDs is ultimately dependent upon the computational horse power available and the published linear write speed of the tested SSDs. By turning off data compression, WildFire actually has achieved random write performance of > 2.7 million 4KB writes a second with 24 SSDs. This was 98% of the composite linear write rate of the 24 Samsung 850 PROs tested. >2 million has been achieved with the same SSDs with compression on and RAID-5. Marginally less than 2 million has been achieved with RAID-6. The 1.2 million rate for the Crucial MX-300 is based upon its design which practically limits write rates to about 290MB/sec. By comparison, most SSDs, including all of the Samsungs listed, have linear write rates of approximately 500MB/second.

Conversely, traditional SSD RAID sets are limited by their random write speed. The SM863 has a published peak random write rate of 25,000 IOPS per second. Given that a RAID-10 set of 20 SSDs will be able to update any of 10 SSDs on each side of the pair simultaneously, the composite performance will be 10 * 25,000. Similarly, random writing to a parity RAID set requires that all elements of a set be either read from or written to before a write is complete. Accordingly the Random write speed in a RAID-5 or RAID-6 environment will be limited to the random write speed of one SSD. While this seems quite slow, use of an HBA instead of a Smart disk controller is desirable – many Smart disk controllers random write at only half the rate of HBAs.

It needs to be remembered that the write rates shown have no drag from network losses. However, this will have some sort of proportionality. With the right networking, performance will fall to around 400,000 to 500,000 IOPS. But traditional methods will lead to a RAID-10 environment that does not much exceed 100,000 IOPS, while traditional partity RAID-6 will fall to near 10,000 IOPS. There is still a broad sweep between the different performance levels.

The elected write amplification rate (see the green segment) is crucial in estimating the usable life of an SSD. While write amplification of 0.8:1 seems overly low, this paper has demonstrated that basic rates in the 1.2:1 range can be achieved through write reduction only. Similarly, we have noted that RAID amplification is reduced to nominal levels. Finally, we have observed that the compression of metadata can create profound data reductions, and that the whole can often be brought down to a value approaching 0.6:1.

Conversely, we see that the data center SM863 has two different values. Data center SSDs are generally designed for wear amplification of around 2.5:1, which requires approximately 35% to 40% free space. The parity value of 7.5:1 is much higher but accurately reflects the realities of random updated RAID-6, where three writes will be required for each source write.

Conclusion

The architecture of ESS results in dramatic manufacturing cost reductions and performance enhancements due to software. This is not a normal condition. Most software improves one category of performance at the expense of another. But ESS began from the recognition that reducing wear amplification and achieving speed increases were two sides of the same coin: a reduction in wear amplification left more time and more resource to focus on writing, thus increasing linear write speeds.

WildFire-Storage™
A division of EasyCo LLC
220 Stanford Drive
Wallingford PA 19086 USA
Tel: (+1) 610-237-2000
888-473-7866
Email: sales@WildFire-Storage.com
Web: http://WildFire-Storage.com

19 June 2017

By Sam Anderson
President of EasyCo LLC