GALLAGHER PR

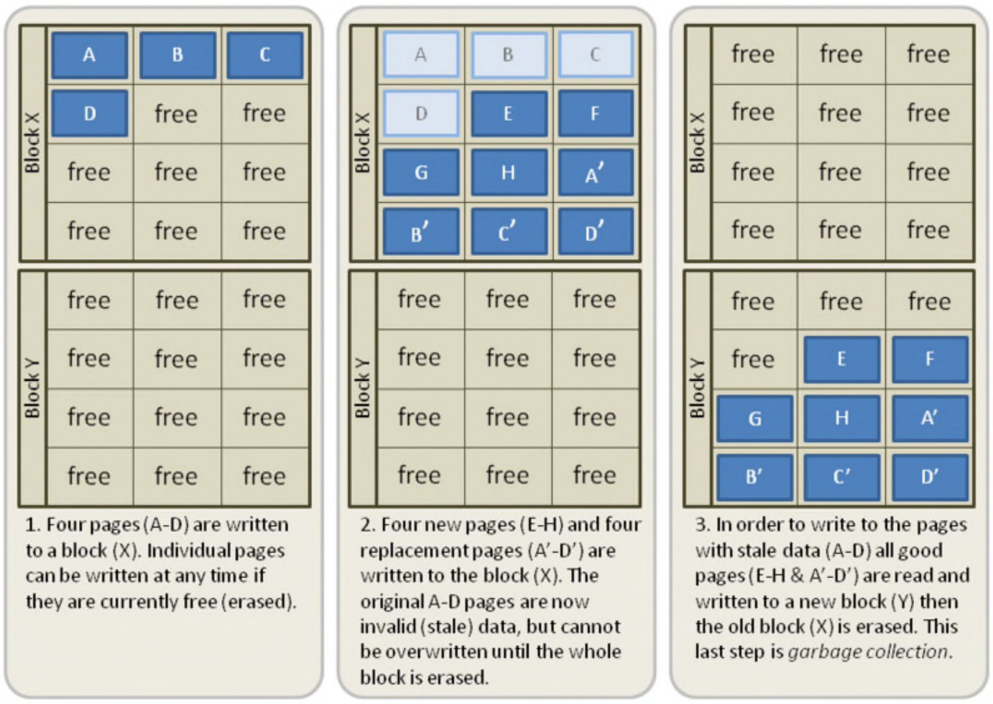
Garbage Collection in SSDs

Written by Charlie Rubin for LSI
Exclusive to *SSD Review*

Garbage collection is a fundamental process with all SSD drives, but it can be implemented in different ways that can impact overall SSD performance and endurance. In this article, we'll look at how garbage collection works, how it can be implemented, and how it relates to the TRIM command and overprovisioning.

With flash memory, garbage collection is the name for the process of relocating existing data to new locations. Unlike hard disk drives, SSDs cannot overwrite existing data – they must first erase old data before writing new data to the same location. Flash memory is divided into pages, which are grouped in blocks. You can write data to a page, but you can only erase a block of pages. Therefore, to reclaim space from data that is no longer valid, you must copy the valid data from one block and write it in a new block before erasing the block that contained the invalid data.

The process is illustrated in Figure 1. Pages A-D are written to Block X, and then later that data is changed, so Pages A' – D' are written, making the original A-D invalid. Column two shows the updated A-D being written (A'-D'), and column three shows data A'-D' as well as E, F, G, and H being written to a new block, Block Y, so the space from Block X can be reclaimed by erasing it.


Block X
A
B
C
D
free
Block Y
1. Four pages (A-D) are written to a block (X). Individual pages can be written at any time if they are currently free (erased).
E
F
G
H
A'
B'
C'
D'
2. Four new pages (E-H) and four replacement pages (A'-D') are written to the block (X). The original A-D pages are now invalid (stale) data, but cannot be overwritten until the whole block is erased.
3. In order to write to the pages with stale data (A-D) all good pages (E-H & A'-D') are read and written to a new block (Y) then the old block (X) is erased. This last step is *garbage collection*.

Source: *Wikipedia*

Figure 1: The garbage collection process.

Note that wear leveling typically occurs during garbage collection, as data is written to a variety of new blocks in order to spread wear around over the breadth of the SSD. Since there a limited number of writes you can do to the Flash, if you always wrote to one block, you would exhaust its life of write cycles, and if you continued wearing out one block at a time, you would reduce the number of blocks available for moving data around, thereby slowing down the SSD. Ideally, you want all of the blocks of the SSD to wear at the same rate and be available throughout the life of the drive.

OS Awareness vs. Drive Awareness

In an HDD system, the OS can simply request that new data be written to the location where old, invalid data is stored, and the HDD will overwrite it. In an SSD, however, the page must first be empty before it can be written to – the SSD cannot overwrite existing data.

The OS only tracks the logical blocks that are holding the files. It understands files and the logical locations where they are stored. In any storage system, the storage device doesn't know the file structure – it simply knows that there is certain data written in specific pages and blocks. The system, whether SSD or HDD, returns the data from those locations when asked to do so by the OS. When the OS deletes the file, it simply marks the space used for that data as free in its logical data table.

When it deletes a file from its logical data table, the OS does not tell the SSD that the space is now available. The SSD only becomes aware that the data is invalid when the OS tries to write to that location again, at which time the SSD does garbage collection and then writes the new data.

The TRIM Command

In newer operating systems (Windows 7, Windows Server 2008 R2, Linux 2.6.33, FreeBSD 8.2, Open Solaris, Mac OS X Lion, for example), the TRIM command allows the OS to notify the SSD that data is no longer valid at the time it deletes the logical block addresses from its logical table. The advantage of the TRIM command is that it enables the garbage collection to skip the invalid data rather than moving it. Invalid data is not rewritten, so it consumes less time during garbage collection. The SSD doesn't immediately delete the locations – it just marks them as no longer valid. This is a huge advantage because during garbage collection it can ignore the invalid files instead of moving them to a new location.

The difference is illustrated in Figures 2 and 3.

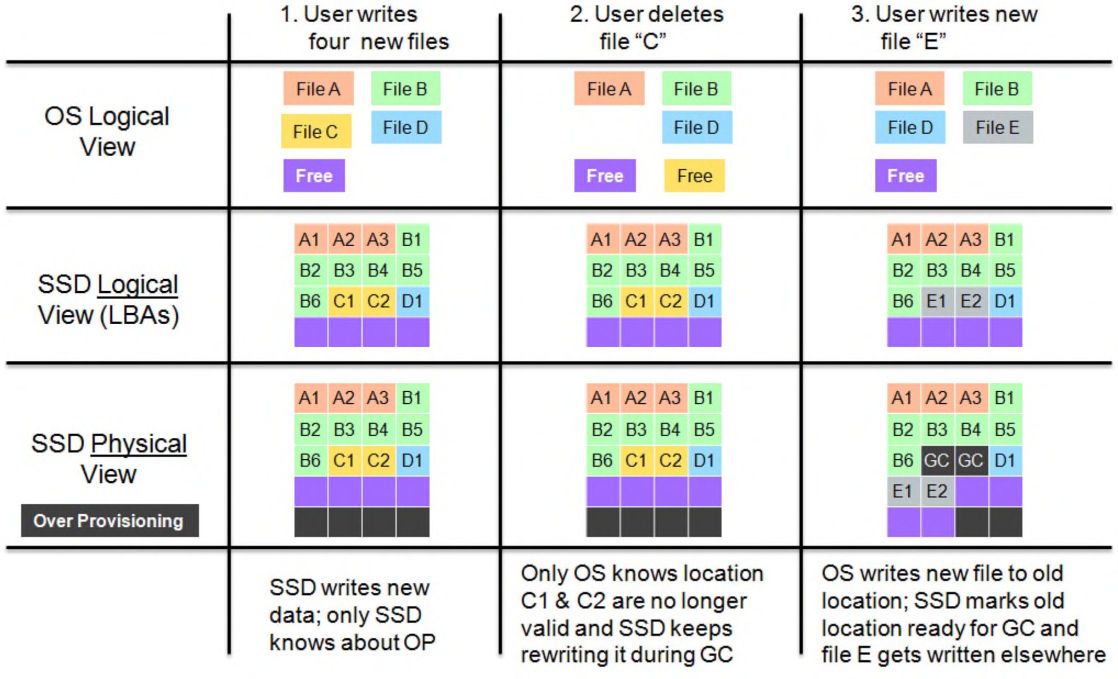

1. User writes four new files
2. User deletes file "C"
3. User writes new file "E"
OS Logical View
SSD Logical View (LBAs)
SSD Physical View
Over Provisioning
File A
File B
File C
File D
File E
Free
A1 A2 A3 B1
B2 B3 B4 B5
B6 C1 C2 D1
B6 E1 E2 D1
B6 GC GC D1
E1 E2
SSD writes new data; only SSD knows about OP
Only OS knows location C1 & C2 are no longer valid and SSD keeps rewriting it during GC
OS writes new file to old location; SSD marks old location ready for GC and file E gets written elsewhere

Figure 2: Garbage collection without the TRIM command.

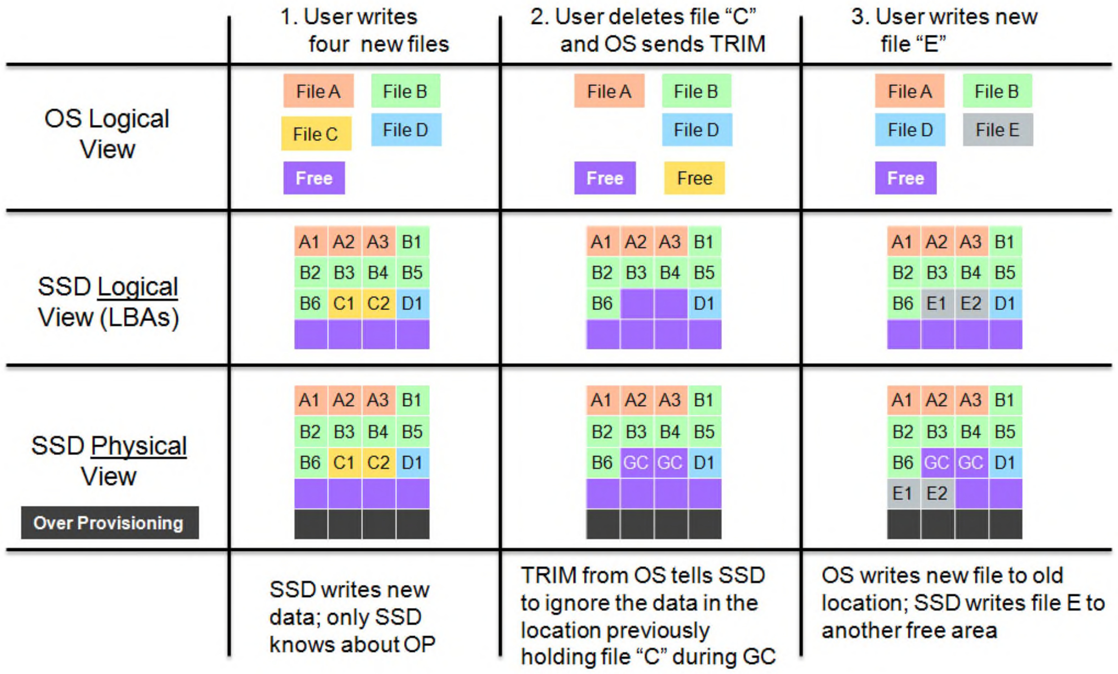

1. User writes four new files
2. User deletes file "C" and OS sends TRIM
3. User writes new file "E"
OS Logical View
File A
File B
File C
File D
File E
Free
SSD Logical View (LBAs)
A1 A2 A3 B1
B2 B3 B4 B5
B6 C1 C2 D1
B6 E1 E2 D1
SSD Physical View
Over Provisioning
B6 GC GC D1
E1 E2
SSD writes new data; only SSD knows about OP
TRIM from OS tells SSD to ignore the data in the location previously holding file "C" during GC
OS writes new file to old location; SSD writes file E to another free area

Figure 3: Garbage collection with the TRIM command.
<Insert JPEG of Slide 6>

Figure 2 shows the process without the TRIM command in use. In Column 1 of Figure 2, the SSD user writes four new files to the drive. The OS sees the files in its logical table, and the SSD has both a logical and physical view of the space available, which in the physical view includes overprovisioning space that is not part of the drive's 'stated' capacity as known to the OS.

In Column 2, the user deletes File C, but the SSD does not know the file has been deleted because the TRIM command is not in use. If it does garbage collection at this point, it will move the invalid data in File C because it doesn't know it's invalid. In Column 3, the user writes a new File E, the OS tells the SSD to use the same space it used from the old File C, and the SSD writes the new file to free space while marking the old space from file C as available for garbage collection.

Figure 3 shows the difference with the TRIM command. In this case, the process is the same when the user writes the four original files. When the user erases file C in Column 2, however, the old File C space is immediately marked as invalid in preparation for garbage collection because the OS uses the TRIM command. Then, when the user writes file E in column 3, the garbage-collected location of File C now becomes free space.

The net result is that thanks to the TRIM command, the SSD knows that data is invalid and it can be considered free space during garbage collection without having to move the invalid data to another block. The TRIM command saves the SSD from having to move some data that is actually invalid and makes more free space available during garbage collection..

In summary, TRIM prevents garbage collection on invalid data and increases the free space known to the SSD controller. This produces three key benefits:

1. Lower write amplification. Less data is re-written and more free space is available during garbage collection (more space to write equals fewer writes needed).

2. Higher throughput. With the TRIM command, there is less data to move during garbage collection and the drive runs faster. Throughput is bottlenecked at the flash – an SSD is only as fast as it can write to the flash memory. During the time it is doing garbage collection, the drive has to stop some of the data transfer from the host while it moves data around. This is why it's beneficial for the SSD to know which data is invalid – so it doesn't have to be moved during garbage collection.

3. Improved endurance, because the drive is writing less to the flash (because it's not rewriting invalid data).

Note: Today TRIM doesn't work in most RAID environments because current RAID drivers don't yet support it. Once more RAID manufacturers can make changes in their drivers, the hope is that TRIM will eventually work in all RAID environments.

DuraWrite and the TRIM command

DuraWrite, found in the SandForce Flash storage processors by LSI, produces similar benefits to the TRIM command whether or not TRIM is present. And when TRIM is present, DuraWrite creates more free space on the SSD than would be possible otherwise. Figure 4 illustrates.

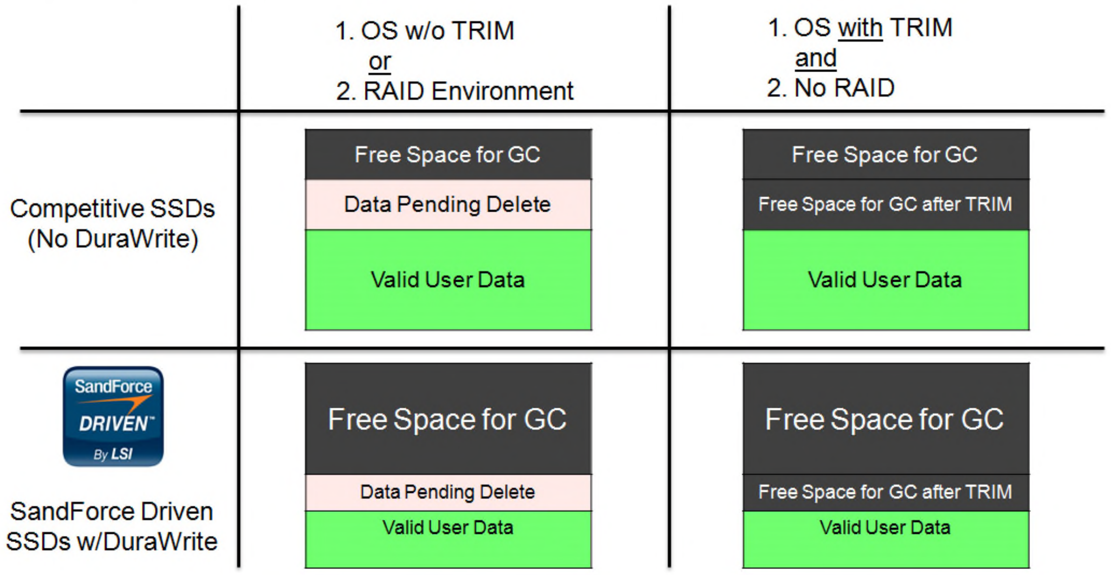


Figure 4: DuraWrite versus non-DuraWrite SSD Controllers and the TRIM command.

In a standard SSD without DuraWrite (Row 1 left), data pending deletion and the free space share the same space. There is less space available for garbage collection, so the drive has to spend more time doing garbage collection to efficiently use the available space. In a conventional drive that supports TRIM, the data pending deletion becomes free space (Row 1 right), and there is much more space available for garbage collection. The more space available, the faster the drive works, so TRIM has a big impact on the drive's overall performance.

In a DuraWrite SSD, less data is written to the drive in the first place, so there is more free space available for garbage collection. In fact, the amount of space available in a non-TRIM environment is comparable to the space available in a TRIM environment on a conventional SSD. (Row 2 left) Writing less data frees up more garbage collection space and improves the performance of the SSD. In the DuraWrite drive with the TRIM command (Row 2 right), there is a massive amount of space available for garbage collection. Having significant amounts of free space does not linearly improve the performance of the drive, however, because there are other bottleneck points in place.

There are many techniques used in the storage industry, such as data de-duplication, data compression, and data differencing. DuraWrite combines elements of these and similar technologies in order to reduce the amount of writes that must be passed through to the Flash.

Therefore, an SSD with DuraWrite in a system that doesn't support TRIM behaves much like a regular SSD that does support TRIM. It is also possible to use DuraWrite in a RAID environment and get the benefits of having the TRIM command without the RAID drivers having to support TRIM.

Background vs. Foreground Garbage Collection

The final question is when to do garbage collection. There is debate in the industry between doing background (or idle-time) garbage collection, and foreground (or standard) garbage collection.

It seems like a good idea to do garbage collection in the background, when the host isn't demanding something of the SSD. It seems reasonable to perform housekeeping like garbage collection when the SSD isn't otherwise busy, so it can be ready with optimized blocks when the OS writes a new file or changes a file.

However, there are two problems with this. First, background garbage collection collects all of the data currently on the SSD, when in fact some of that data may be in temporary files, or may otherwise soon be deleted by the OS. If you garbage-collect data in the background, you may be rewriting a lot of data that won't be needed in a short time, which causes unnecessary wear on the SSD and reduces endurance.

Background garbage collection is used primarily with SSDs whose write speed is slow, because it reduces the amount of garbage collection that must be done in the foreground, and therefore allows the SSD to speed up a bit.

Foreground garbage collection, on the other hand, results in data being rewritten only when it will be needed. Foreground garbage collection thus avoids performing garbage collection on data that soon won't be needed, and it reduces overall wear on the drive. This requires an SSD that has very high write speeds. SandForce Flash storage processors have an architecture that enables the fastest possible foreground garbage collection.

The difference between background and foreground garbage collection is less obvious in consumer-level drives where there tends to be a lot of idle time and endurance isn't as crucial, but it is more obvious in enterprise-class drives, where writes occur all the time and duty cycles are longer.

LSI SandForce Flash storage processors use optimized garbage collection techniques, so they can do foreground garbage collection better than the other SSD controllers. If the SSD also writes less data to begin with, there is less data to rewrite during garbage collection, and the SSD performs better.

As we have seen, garbage collection is a necessary function of all SSDs. The decisions to use foreground or background garbage collection as well as the use of the TRIM command impact an SSD's performance, write amplification, and endurance. These factors should be carefully considered when designing SSDs.