Data Integrity Challenges in SSD Design

Solid state drives (SSDs) are similar to hard disk drives (HDDs) because both are storage devices, but due to differences in the way data is stored, managed and retrieved on each type of device, there must be a different approach to how data integrity is maintained on SSDs. While the need for a different approach to data integrity began appearing with the first memory-based SSDs, the impending release of commodity flash-based drives makes it more important than ever to implement new methods to ensure data integrity.

In this paper, we will examine the types of data integrity used in SSDs, and how the requirements for data integrity change as commodity flash memory is used in broader and more demanding enterprise storage applications.

HDDs versus SDDs

HDDs write data to free sectors and then read it from the same sector as needed. If the data is changed, the HDD controller simply overwrites the original sector, assigning a new CRC value as it is done. SSDs, on the other hand, must always erase blocks before storing new information, and they always use a newly erased block to store changed data, leaving the old data untouched in the old block. Because of this, address translation and data versioning techniques are essential to prevent the drive controller from returning stale data, e.g. data that was correct originally but has since been updated.

Given these differences in the way data is stored and changed, the designer must adapt data integrity techniques that are properly suited to each type of device. In the case of the SSD you need a much more encompassing form of protection.

Data Integrity Techniques for SSDs

The four methods that can be used to maintain data integrity in enterprise-class SSDs:

- Error Correction Code (ECC), which protects against read errors as a result of hardware errors in the flash memory. The drive controller monitors the read process and is able to correct hardware read errors up to a certain level. When successful the ECC will enable the drive to provide the correct data back to the user.

- Cyclic Redundancy Check (CRC), which provides “end-to-end” protection by ensuring that the data written is the same data returned when read. As data comes in over the interface, the drive controller generates a CRC value and embeds it with the file’s other metadata. When data is retrieved, the controller checks to ensure that the proper CRC value is present. However if the data does not match, CRC can only identify that an error has occurred. It cannot correct it, but it does prevent “silent data corruption.”

- Correct Address Translation via Logical Block Address (LBA), which ensures that the data is retrieved from the correct location. On an HDD this is the physical sectors on the rotating media. On an SSD this is the logical mapping of the flash memory blocks.

- Correct Version of Data, which is used in SSDs to ensure that the current version of data is returned, rather than the stale version. This is not a problem for HDDs since they can directly overwrite the older data.

ECC

Uncorrectable Bit Error Rates (UBER) occur when a sector (HDD) or flash block or page (SSD) goes bad or there is an error reading back the data from the media. In high-end enterprise hard drives, there is less than one sector error in 10^{16} bits read. These rates are acceptable for hard drives because their peak data transfer performance is generally 50-100 Mbps. Even if the drive runs with a typical duty cycle, this means there will be an uncorrectable sector error only once every year or so. But in an SSD, the data transfer performance is much higher at up to 250 Mbps, so the UBER must be even higher – 10^{17} – to ensure comparable reliability.

ECC techniques are well-defined (typically using BCH or Reed Solomon algorithms), but SSDs need more sophisticated ECC protection. Each flash memory maker requires a different amount of correction depending upon the type of Flash memory and the process technology involved. As the process technology shrinks, the potential for errors increases. With SSDs, therefore, ECC must become stronger to support the increased transfer rate over HDDs.

In addition, designers must consider that ECC uses processing time and processing power on the drive controller. To optimize SSD performance, the designer must use an ECC scheme that is appropriate to the Flash memory used (sometimes spanning 4-bit to 16-bit protection). MLC flash-based SSDs will need a stronger ECC technique than SLC flash-based SSDs. The SSD controller must be also flexible enough to handle more complex ECC algorithms.

CRC

The rotating magnetic media (HDD) or flash memory (SSD) itself can introduce errors that are protected by ECC. But data passes through on-board RAM, firmware, and other elements of the drive, where a hardware or software error may cause the drive to modify the user data before the drive writes it to memory or after it reads it and sends it to the user. This is very important because these silent errors change data without the user's knowledge. Imagine a silent error occurs in a financial firm's history log of a large stock trade. This could cost the company billions of dollars and they wouldn't be aware of any data corruption until an audit occurred, often weeks or months after the fact. This would be unacceptable, so the drive must be made aware of these "silent corruption" errors. CRC helps prevent these occurrences from happening. If the CRC is generated and stored with the user data, a simple CRC check can determine if the data about to be provided to the user matches what was originally stored. If the CRC test fails, the drive can report there was an error. It will not be able to correct the error, but it will prevent

sending back wrong data without telling the user of the problem. The only way to correct such an error would be to incorporate the drive in a RAID array where the other drives would be used to regenerate the corrupted data.

LBA and Correct Data Version

Just using ECC and CRC is a particularly effective technique with HDDs because it is seldom necessary to ensure that the data address is accurate or that the data address has been translated properly. However an SSD cannot directly overwrite data in the flash memory. A process is required where the block to be modified is read by the controller, the data to be changed is modified in the buffer, and finally the entire new block is written to a new location in the flash memory. The old "stale" data is left in place.

This process requires the SSD to have a way to translate the request from the host for a particular Logical Block Address (LBA) into the correct physical location in the flash memory. This is not an issue on a hard drive (except for the few mapped out bad sectors), but it is a continuing problem with SSDs because of the need for SSDs to erase a block before writing to it. Multiple versions of data are invariably created on an SSD.

The SSD's LBA table must be continually updated to properly identify the current location of the freshest data. The stale data doesn't contain a record of its being rewritten to a new block, so if the drive controller points to the old location by mistake, the drive will return stale data. The ECC will work properly on the stale data. The CRC will decode properly from what was originally saved. In the end, the end user will get data that the drive assumes is correct. Unfortunately that data is old and was likely updated at some point, so the user is not aware of this "silent corruption."

Currently the only way to validate that the drive does not have this silent corruption problem is with explicit testing. One method would be to run a test where the LBA itself and an incremental value are stored in that LBA. The host test system would ensure that the data returned contains the correct LBA and incremental value. If the wrong value is returned the host would know the drive failed the test.

This problem can also occur if there is a power failure while the LBA is being updated, the table may contain only information for the data that is now stale. To prevent errors like this, designers should explore using on-board capacitors (SuperCaps) to provide temporary power during a failure so that the buffers can be flushed and the LBA table properly updated.

Lastly, there is the issue of silent corruption due to firmware bugs. The only way to guard against firmware bugs is through extensive directed testing so that products are known to be bug-free before being put into production.

Conclusion

The advantages of SSDs outweigh any challenges that impact their practical implementation in enterprise storage systems. HDD manufacturers and enterprise storage system providers have already begun to offer SSD-based storage systems, but now they are beginning to deliver systems that use commodity flash, which is denser, and has higher error rates. To ensure that MLC flash-based enterprise SSDs deliver the same or better data integrity protection as HDDs, designers must take into account the differences in the way data is written, changed, and retrieved from these devices, and they must ensure that the SSD controller has appropriate data integrity protection techniques in place.