

# Silent data corruption in SATA arrays: A solution

Josh Eddy  
August 2008

## *Abstract*

Recent large academic studies have identified the surprising frequency of silent read failures that are not identified or resolved in enterprise SATA disk arrays despite the typical integrity functions. Such errors result in corrupt data being provided by the disk array to the application without any notification or warning, potentially leading to erroneous operations and results. This “silent data corruption” includes misdirected writes, partial writes, and data path corruption with evidence pointing to the errors being uncaught in 1 in 90 drives. By using an extended data integrity feature, silent data corruption can be identified and handled so that corrupt data is not sent to the application. Furthermore, data that otherwise would be lost can be recovered.

<b>INTRODUCTION .....</b>	<b>1</b>
<b>BACKGROUND: STORAGE SYSTEM ARCHITECTURE .....</b>	<b>1</b>
SECTORS AND ECC .....	1
DISK ARRAYS AND RAID.....	1
<b>SILENT DATA CORRUPTION.....</b>	<b>2</b>
INTRODUCTION .....	2
MISDIRECTED WRITE .....	2
TORN WRITE.....	2
DATA PATH CORRUPTION .....	2
PARITY POLLUTION.....	3
SILENT CORRUPTION FREQUENCY.....	4
<b>EXTENDED DATA INTEGRITY FEATURE.....</b>	<b>5</b>
WHAT IT IS .....	5
HOW IT IS STORED .....	5
HOW EDIF WORKS .....	6
REAL WORLD EXPERIENCE .....	7
<b>SUMMARY AND CONCLUSIONS.....</b>	<b>7</b>
<b>REFERENCES AND FURTHER READING .....</b>	<b>7</b>

## Introduction

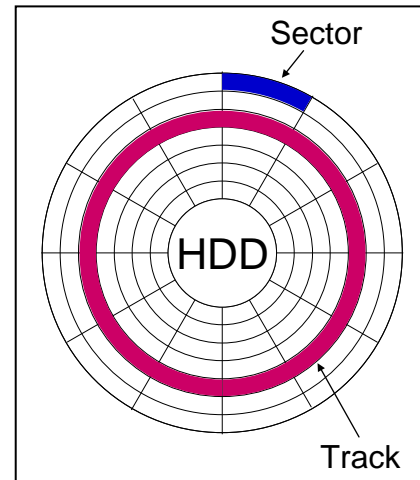
Data integrity is the core of what enterprise systems rely upon. When applications and other systems receive data from their storage system that is malformed or corrupted in some way, it leads to wrong results or downed systems. Even though storage systems contain many layers of data integrity checking and correction, users still encounter corrupt data and its detrimental effects.

## Background: Storage system architecture

### **Sectors and ECC**

The smallest externally addressable unit on a hard disk drive (HDD) is a sector, which typically contains 512 bytes of data. When data is written to a hard drive, it is written in a “block” of data, which is just a set of contiguous sectors.

Hard drives contain error detecting and correcting mechanisms to recover from mis-read or mis-written data. For each sector, extra space is set aside to store an error correcting code (ECC) that can be used to detect and correct mis-read or corrupted data. This extra space accounts for much of the difference between an unformatted hard drive’s capacity and the actual usable capacity. But, as we will see later, the hard drive sometimes does not detect that data has been mis-written.



**Figure 1: Hard Disk Drive Structures**

Disk drives commonly experience errors that are recoverable via the ECC. However, various factors lead to degeneration of data integrity on the hard drive, making it harder to recover from errors. At some point, data starts to become unreadable leading to “read failures,” which can cause systems to crash, hang, or become unstable. For greater reliability, people use a RAID system in which read failures will cause the array controller to mark the disk drive as “failed” but maintain data availability using the other drives in the RAID group.

### **Disk Arrays and RAID**

RAID technology uses multiple disks to achieve greater performance and data availability. By spreading data across multiple disks and storing a parity calculation of the data, RAID can reconstruct what would have been lost data in the event of a single disk failure. Note that the reconstruction period of a failed disk can take hours to complete and performance is negatively affected.

RAID can also catch and address errors that are flagged by hard drives. However, there is a set of disk errors that are not caught by the hard drive and thus not by RAID, which can lead to incorrect data being returned to the server.

# Silent data corruption

## Introduction

There are types of storage errors that go completely unreported and undetected in other storage systems. They result in corrupt data being provided to applications with no warning, logging, error messages or notification of any kind. Though the problem is frequently identified as a silent read failure, the root cause can be that the write failed, thus we refer to this class of errors as “silent data corruption.” These errors are difficult to detect and diagnose, and what’s worse is that they are actually fairly common in systems without an extended data integrity feature.

## Misdirected write

In some instances, when writing to a hard disk, data that is supposed to write to one location actually ends up being written in another location. Through some fault, the disk doesn’t recognize this as an error and will return a success code. As a result, the “misdirected write” is not detected by the RAID system because it only takes action when the hard disk signals an error.

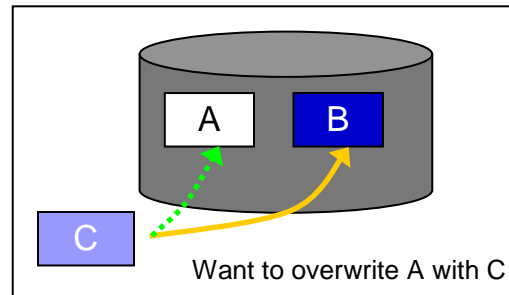


Figure 2: Misdirected Write

Thus, not only has an undetected error occurred, but there has been data loss as well. In Figure 2, data block C was supposed to overwrite data block A but instead overwrote data block B. So data block B is lost and data block A still contains the wrong data!

As a result, data has been written to the wrong location; one area has old, wrong data; another area has lost data, and this error has not been detected by the RAID system nor the HDD. Accesses to retrieve B or C will result in incorrect data being returned without any warning.

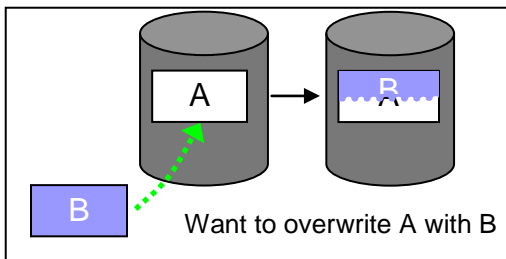


Figure 3: Torn write

## Torn write

In other instances, only some of the sectors that are supposed to be written together end up on the disk. This is called a “torn write” which results in a block of data that contains part of the original data and part of the new data. Some of the new data has been lost, and some reads would return the old data. Again, the hard disk is not aware of this error and returns a success code, so it goes undetected by RAID. Accesses to retrieve B would return partly incorrect data, which is completely unacceptable.

## Data path corruption

The data path from the controller to the hard drive consists of several components: the controller itself, the initiator adapter, the cable, the target adapter, the disk enclosure, and

the hard drive. Each of these components is made up of smaller components and modules. Though data is sometimes protected with error detecting (e.g. Cyclic Redundancy Check, CRC) or error correcting code (ECC) while it is being transported from component to component, often the data is exposed to corruption while still within the hardware modules and chips. For example, the RAID module could change the data while calculating the parity, or some bits may be erroneously flipped while stored in the disk controller cache.

There are thousands (if not hundreds of thousands) of lines of low-level firmware code running on most pieces of hardware – when communicating data, it is called the “protocol stack.” And as much as we want it to be, that code is not bug-free. So a logic error in the firmware can change data in a way that error checking is circumvented when in fact it is not a proper change. Because these can be logical errors rather than a corruption caused by a malfunction, the likelihood that the error would be detected elsewhere is reduced.

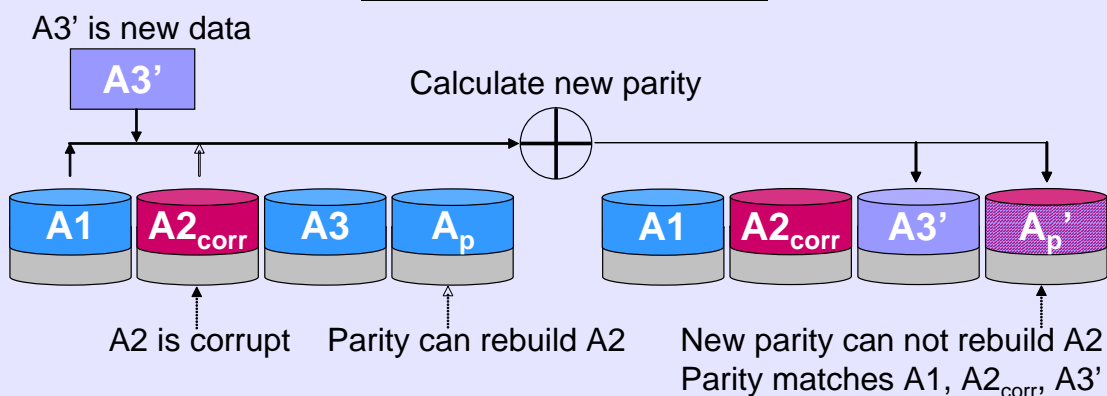
With the combination of firmware code bugs and potential hardware malfunction, there is a possibility that during the round trip from controller to disk and back, data might be corrupted without the application’s knowledge.

### ***Parity pollution***

Once a silent corruption has occurred, the error can be compounded to a point where the original data can no longer be retrieved or detected even in a RAID system.

When new data is written in a system that uses a RAID level with parity, a new parity is calculated before the data is written to disk. The RAID system reads the data across the stripe and combines it with the new data to calculate the new parity. In a system that has silent data corruption, this situation leads to unrecoverable and undetectable data loss. When corrupt data is used to calculate the new parity, the RAID system can no longer use the parity to restore the non-corrupt data. This scenario is called “parity pollution.”

### Parity Pollution: An Example



In the diagram, the disk array (bottom left) originally contained A1, A2, A3 and A<sub>p</sub> which is the parity calculation for A1, A2, and A3. There was silent data corruption in A2 so that disk now holds A2<sub>corr</sub>. Some new data comes in to be written – it is called A3'. The RAID system reads A1 and A2<sub>corr</sub> in order to calculate the new parity before A3' is written to disk. However, the parity of A1 and A2<sub>corr</sub> are usually not checked at this time, so the corruption goes undetected. The RAID system calculates the new parity, called A<sub>p</sub>' and writes it to disk along with A3'. Because the new parity was calculated with A2<sub>corr</sub>, it is no longer possible to rebuild the original A2. Furthermore, because the new parity matches all of the data on disk, the original corruption is undetectable.

### **Silent corruption frequency**

A recent academic study [1] of 1.5 million HDDs in the NetApp database over a 32 month period found that 8.5% of SATA disks develop silent corruption. Some disk arrays run a background process that verifies that the data and RAID parity match, and can catch these kinds of errors. However, the study also found that 13% of the errors are missed by the background verification process.

When you put those statistics together, you find that on average 1 in 90 SATA drives will have silent data corruption that is not caught by the background verification process. So when those data blocks are read, the data returned to the application would be corrupt, but nobody would know. For a RAID-5 (4+P) configuration at 930 GB usable per 1 TB SATA drive, that calculates to an undetected error for every 67 TB of data, or 15 errors for every petabyte of data. If a system were constantly reading all that data at 200 MB/sec, it would encounter the error in less than 100 hours.

Another very large academic study [2] looked at failure characteristics for entire storage systems, not just the disks. In the 39,000 storage systems that were analyzed, the protocol stack (firmware) accounted for between 5% and 10% of storage failures. These are the kinds of failures brought on by faulty code in the firmware.

Thus firmware is not error-free and may introduce silent corruption. It's clear that the introduction of data corruption in the data path is a very real scenario.

## Extended Data Integrity Feature

### ***What it is***

Xanadu arrays include a standard feature called “Extended Data Integrity Feature,” or EDIF. It protects against misdirected writes, torn writes, and data path corruption.

Before handling data, the Xanadu controller calculates a Data Integrity Field (or DIF) for each sector. The DIF is like a super-checksum CRC and is stored on disk. The DIF is checked on read and write of every sector and this is how corrupted data is identified. Once corrupted data is identified it can be fixed. EDIF also avoids the problem of “parity pollution” which prevents silently corrupted data from being detected.

In return for protection against silent data corruption, there is a 1.5% capacity tradeoff, accounting for the DIF storage.

### ***How it is stored***

Typical Enterprise-class SCSI-based drives (i.e. Fibre Channel, SAS, and FCoE) in disk arrays use a 520 byte sector size. So for every 32KB block, there are 64 sectors. Each sector has 512 bytes of data and 8 additional bytes called the DIF (Data Integrity Field), for checksumming and other uses.

Conversely, ATA-based drives (including SATA) typically fit 65 sectors into a 32KB block. These sectors are 512 bytes and are used only for data storage. There is no similar checksum functionality in SATA at this level in the storage technology stack.

However, for every 32KB block of disk, Xanadu SATA includes 64 data sectors of 512 bytes and one DIF sector of 512 bytes.

Inside the DIF sector are 64 Data Integrity Fields that correspond to the 64 data sectors in the block. Each DIF is 8 bytes. So logically, Xanadu SATA uses 520 bytes per data sector – 512 for data, 8 for the DIF – like in Fibre Channel and SAS drives.

The DIF includes a checksum of the data as well as the location of the data and other information, which it uses to protect against silent data corruption.

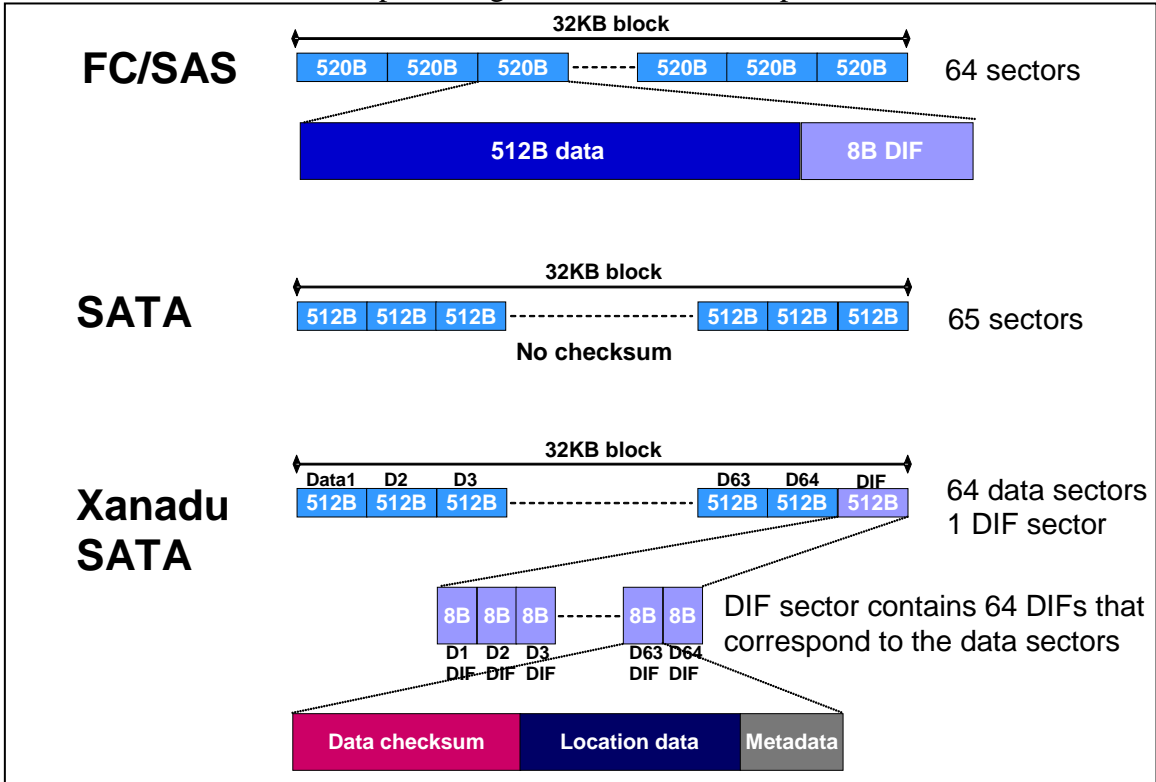


Figure 4: How DIF is Stored

### How EDIF works

When performing a read, write, or parity calculation, the controller calculates a DIF and compares it to the source multiple times. It checks for corrupt data coming from disk and in the data path, both external to the controller and inside the controller. It is calculated and compared using a high-performance hardware module, rather than requiring the overhead of software or an operating system to perform the calculation.

When a disk sector has an error detected repeatedly, it indicates a possible bad sector. So the array controller tells the hard drive to reassign the sector to another location. Further requests to or from the original sector are automatically redirected to the new sector.

When multiple sectors have errors repeatedly, it indicates a whole drive problem. The ability of the drive to function properly is called into question, and the data it contains is suspect. Depending on the circumstances, the drive may be removed from use and the parity information would be used to provide data and rebuild a spare drive.

If there is a corruption on the data path, it is detected and logged, and the corrupt data is not sent to the application or written to disk. For a read, the correct data is still on disk, but there is a malfunctioning component on the path that prevents access to the data. With the Xanadu, such a problem is no longer a silent error, so the problem can be recognized, diagnosed, and addressed without putting data at risk.

## ***Real World Experience***

Many storage administrators can tell you that they have witnessed the presence of undetected data corruption, though there is often no proof of the problem. That prevents many organizations from spending further resources on investigating the problem.

For scientific work, data integrity must be at its highest so that performing the same analysis on the same data leads to the same results. However, when the same results are not achieved, it makes it easier to identify data corruption as the culprit. For example, CERN, Fermilab, and Desy all experienced data corruption cases, and the investigation at CERN Computer Center pointed the finger at various causes, including silent data corruption [3].

It was a similar investigation that led RAID Inc. to select the Xanadu to offer its customers in the High Performance Computing (HPC), government, and research markets where data integrity can be readily apparent.

## **Summary and Conclusions**

The evidence presented in academic studies and customer experience should prove the existence and regularity of errors that are not caught by typical RAID systems. Silent data corruption is an unrecognized threat that may affect as many as 1 in 90 SATA drives. The Xanadu Extended Data Integrity Feature detects and addresses silent data corruption, while also preventing parity pollution.

## **References and Further Reading**

[1] Bairavasundaram et al. "An Analysis of Latent Sector Errors in Disk Drives." *Proceedings of the International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS'07)*. June 2007.

Bairavasundaram et al. "An Analysis of Data Corruption in the Storage Stack." *Proceedings of the 6th USENIX conference on File and Storage Technologies (FAST'08)*. February 2008

[2] Jiang et al. "Are Disks the Dominant Contributor for Storage Failures?" *Proceedings of the 6th USENIX conference on File and Storage Technologies (FAST'08)*. February 2008

Krioukov et al. "Parity Lost and Parity Regained." *Proceedings of the 6th USENIX conference on File and Storage Technologies (FAST'08)*. February 2008

[3] Panzer-Steindel. "Data Integrity." *Internal CERN/IT study*. 8 April 2007  
(<http://indico.cern.ch/getFile.py/access?contribId=3&sessionId=0&resId=1&materialId=paper&confId=13797>)

Note: Figure 2 and Figure 3 are derived from concepts in the presentation given on "Parity Lost and Parity Regained."