Understanding Bad Sectors: Causes, Effects, and Management

By Joseph Chen (2024-4-10)

I. Overview

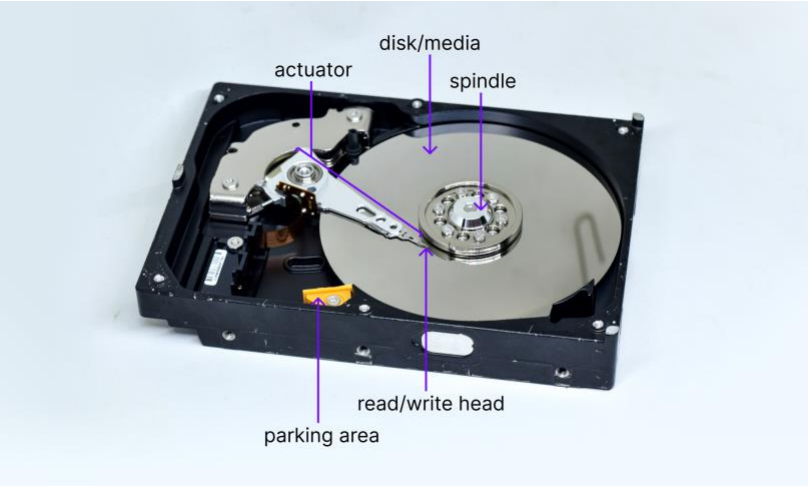
One of the leading causes of damage and failure in disk drives is bad sectors. In simple terms, bad sectors are unreadable locations on a disk drive's media. The purpose of this article is to help the reader understand what bad sectors are, how they occur in the context of hard disk design and operation, what negative effects they may create, and how they can be managed by the drive, the system, as well as the user. We will pay special attention to how bad sectors can be managed within NAS devices.

Disk drives are also known as Hard Disk Drives (HDDs), where the term "disk" describes the use of disk plates in the drive as storage media. While popular storage devices called Solid State Drives (SSDs) exist and serve a similar purpose to HDDs in computer storage, albeit with a completely different underlying storage mechanism, this article will focus on HDDs.

II. Introduction to Disk Drive Design

This section introduces the basic building blocks of a disk drive, which provides the context for what bad sectors are, how they can occur, and how they are handled by the disk drive. As it has evolved over half a century, the way that a disk drive stores data has remained the same: it is a magnetic recording device that stores digital data for a computer. Disk drives are built with spinning magnetic disks and one or more heads that write and read data from the surface of the disks. Over the years, the physical size of the device shrank from the size of large refrigerators to having an area of about 20% the size of copy paper. Not only has the physical size been compressed, but the amount of data that can be stored has increased drastically, from megabytes (MBs) to terabytes (TBs), effectively making the memory of modern disk drives a million times denser than their predecessors.

Figure 1: A labeled disk drive



A. Head and Media

The rotating disk is the basic recording medium in a disk drive. During a write operation, a head records information on a disk by magnetizing it with an electric signal.

The head is mounted on an actuator that moves along the surface of the medium. There are two components of the head, one for reading and another for writing. The writing component, or write head, is responsible for recording the data on the medium. The reading component, or read head, is responsible for picking up the signals previously recorded on the medium.

In addition, read heads are also responsible for detecting servo tracks or servo marks on the media. Servo tracks or servo marks guide each head to a precise location on the disk for each read and write operation.

Typically, for drives made after the year 2000, when the drive is stopped, or power is turned off, the head will be placed in a location where it does not touch the media, called the parking area. When not in use, the head is latched securely within the parking area, where it is not moving, to avoid head and media contact. Contact between the head and media may cause scratches on the media, which in turn may create particles that may cause additional scratches.

When the disk starts spinning, the media will reach a fixed speed of either 5400 RPM, or 7200 RPM, or for enterprise-class drives, 10000 RPM. The faster the rotation speed, the faster the response the drive can provide. The disk drive can perform reads/writes only after the media has reached a designed speed, such as 7200 RPM. After this speed is reached, the head will be moved to a ramping location to engage the media surface. This movement is called "taking off the head." The head flies over the media surface at a small distance, such as a few microns. The air bearing between the head and the media allows the head to fly over the media at high speeds without touching the media.

Under normal conditions, this flight is very safe, and the drive can operate for years without any issues. However, there are external influences that can make the flight abnormal. For example, external shock and vibration may cause the head to bump into the media, causing bad spots, or causing bad read or write conditions. Other potential issues are related to the particles generated during and after the drive is manufactured. These particles may come under the head and create scratches.

When the drive stops spinning, the head is designed to automatically return to the parking area. The head actuator is designed to automatically move the head smoothly onto the ramp and then park and securely latch the head.

B. Sectors and Tracks

The heads are ready to read or write data to the media once the disk drive spins to the correct speed. On the media, there are many invisible concentric circular lines, where each circle is a single track. These circles are packed as closely as possible together to maximize recording capacity. The width of each recording mirrors the width of the magnetic spot on each head. Modern disk drives are usually constructed with tens of thousands of tracks on each surface. To read or write data, a head likely needs to move from one track to another. This movement includes the time it takes the head to move, settle, prepare the electronic element of the head, and wait for the target location on the track to arrive.

Due to the rotation of the media, the time it takes to access any location on the disk is equal to the time it takes for the head to move to its target track, plus the time it takes for the media to rotate to its target sector. The time it takes for the head to move to its target track, while guided by servo tracks, is called seek time, and typically takes 5 to 10 milliseconds (ms). The media rotation speed is typically 7200 RPM or about 8 ms per revolution, so the average time it takes for the media to move to its target sector, or rotational latency, is about 4 ms. Therefore, the total time it takes to access any location on the disk, or total random latency, is typically between 9 to 14 ms.

Servo tracks or servo marks are markings on the media that define the precise location of each track, guide the head to its target track, and help the head to stay on its target track while the media is spinning. Even if the media shifts away from its center of rotation, servo tracks allow the head to continue to dynamically follow its target track.

Each disk surface has a single head assigned to read and write to it. In most disk drives, all heads are mounted on a single actuator powered by a single servo motor. Therefore, whenever a single head seeks a single track on a single disk, the disk drive's actuator typically ends up moving all the heads simultaneously.

However, newer disk drives designed for higher performance occasionally employ multiple actuators that allow for simultaneous multi-channel operation. But due to the added mechanical and electrical complexity and cost of this design, it is usually limited to specialty

drives. As of early 2023, the only multi-actuator drives deployed in the market are dual-actuator drives, and only a small number of disk drives (perhaps less than 3% of drives shipped) have this type of construction.

A sector is a recording unit physically laid out on the track of a disk, normally consisting of 512 bytes or 4096 (4K) bytes. In a disk drive, sectors are placed in a contiguous fashion on each track. In early disk drive designs, each sector had a dedicated ID area to distinguish it from other sectors on the same track. However, newer disk drives simply utilize the starting point on each track to locate each sector. By removing the ID area from each sector, which results in what the drive manufacturing industry calls "ID-less" sectors, newer disk drives can increase their capacity by 5% over traditional disk drives.

In error reports, two sector-related conditions that can appear are ‘ID Not Found’ and ‘Address Mark Not Found’. These errors describe when either the sector ID or Address Mark is missing, respectively. The Address Mark in the latter error refers to the starting point of a sector. An Address Mark Not Found error is due to a data recording issue. ID Not Found errors, on the other hand, are caused by an issue with the servo mark.

There are many sectors on a track, usually hundreds to thousands. The number of sectors per track depends on the distance from the center of the disks. That is, there are more sectors per track the further away you get from the center of each disk. The recording density remains the same between tracks near and far from the center. However, the frequency of data reads and writes is higher for tracks further away from the center.

Disks are usually physically divided into areas called "zones." Each zone has a distinct read/write frequency, whose value depends on the zone's distance from the disk center. Dividing disks into zones allows them to be used more efficiently. The number of sectors per track in each recording zone is the same. The outer zones have more sectors per track than the inner zones. This practice of dividing disks into zones is called “zone bit recording.”

Each track starts at the sector that gets written to first. At the end of each track, there may be some sectors that are not used during regular reading and writing. Such sectors are called "spare sectors." Spare sectors are reserved for use when a defective sector is found on the track and needs to be reassigned. We will cover spare sectors in more detail in the “Spare Sectors” section below.

The starting points of adjacent tracks are staggered to accommodate the time it takes for the head to move from one track to the next. When the drive finishes reading or writing one track, it may continue to the next track. To prevent the disk from having to make nearly a full rotation before the head can read the first sector of the next track, the starting point of the next track is offset by an amount determined by the time it takes for the head to move from one track and settle on another. This offsetting of track starting points is called "skewing."

The drive's media is comprised of physical sectors. However, the drive's host system views and accesses the drive's sectors as individual "logical" blocks, where each block may consist of more than one sector. Therefore, each sector on a disk is assigned a logical number called a logical block address (LBA), where the LBA is the address of the sector from the point of view of the drive's host system, and where multiple sectors may be assigned to each LBA. Normally, the total available Logical Block Number does not change for a drive after it is manufactured. Some special commands may reduce this number, but this is not typically done.

The firmware inside the drive will map each LBA onto a physical drive location in terms of the platter, track, and sector. There is no hard rule for how this mapping will occur. However, traditionally, the mapping tries to optimize drive performance. And generally, the ordering of LBAs typically proceeds from the outer cylinder (outer tracks) towards the inner cylinder (inner tracks), and continues onto the next disk surface, and onto the next platter. LBA mapping typically also considers spare sectors on each track, spare sectors on each group of tracks, and spare tracks. Some spare sectors and spare tracks may be located within small areas of the drive that are also reserved for internal tables and firmware.

C. Spare Sectors

Spare sectors are additional sectors that are not normally counted towards the drive capacity. Spare sectors serve to mitigate the effects of drive manufacturing defects, which occur naturally during manufacturing. Spare sectors also serve to mitigate the effects of drive defects, which occur after the drive leaves the factory and over the drive's life. Spare sectors may be located at the end of a track, the end of a group of sectors (such as a drive zone), or the end of a disk. The closer a spare sector is located to a defect, the better the drive will perform. Typically, the number of spare sectors available is sufficient to cover sector reallocation needs.

Whenever the host issues a read or write command to the drive, the drive must determine whether the target sector has been reallocated and whether the target sector is pending reallocation. This information is stored in two tables on the drive, a **Reallocated Sector Table**, and a **Pending Reallocation Table**. For convenience, we will refer to these tables going forward collectively as "**drive reallocation tables**." The size of these drive reallocation tables and the speed with which they can be searched places a practical limit on the number of sectors that can be reallocated. When this practical limit is exceeded, S.M.A.R.T. failures can result, particularly S.M.A.R.T. Attributes 5 (i.e., Reallocation Event Count) and 197 (i.e., Current Pending Sector Count).

There are two potential constraints on the number of available spare sectors. The first potential constraint is simply the number of available spare sectors that the drive has at any given moment. This number is determined during the drive's manufacturing and decreases over the life of the drive as spare sectors are used up by the sector reallocation process to mitigate the effects of drive damage due to wear and tear. The second potential constraint is the amount of free space available in the Reallocated Sector Table.

For performance reasons, the sizes of the two drive reallocation tables are limited by design, because larger drive reallocation tables result in longer search times per command and, consequently, a greater overall overhead on drive performance.

If the number of available spare sectors is constrained by the number of available spare sectors rather than the amount of free space on the Reallocated Sector Table, then the value of SMART Attribute 5 will be based on the number of spare sectors available. When a drive has all its spare sectors available, the normalized value of SMART Attribute 5 (aka Reallocated Sector Count) will be 100. When a drive has 10% of its spare sectors available, the normalized value of SMART Attribute 5 will be 10. In the latter case, if the drive's SMART 5 threshold is also 10, then a SMART 5 trip will be raised. When a drive has no spare sectors available, the normalized value of SMART 5 will be 1.

However, the size of the Reallocated Sector Table is typically smaller than the number of available spare sectors and therefore, typically, constrains the number of sectors that can be reallocated. When such is the case, the value of SMART Attribute 5 will be based on the amount of free space in the Reallocated Sector Table. The Reallocated Sector Table is used by the drive to check if the logical address of the target sector that is to be read from or written to points to a spare sector. This table is searched during every single read and write command. The size of this table cannot be too large, or else drive performance will be impacted negatively. The response time for reads and writes is critical, and the larger this table is, the longer the drive's response time will be. In the past, this table ranged from 1000 to 5000 sectors. The size of this table is ultimately determined by the drive vendor.

Although the number of available spare sectors can be constrained by either the number of available spare sectors or the size of the Reallocated Sector Table, most recent drives are designed to have the number of available spare sectors constrained by the size of their Reallocated Sector Table. This is because recent drives tend to have abundant spare sectors, which causes the Reallocated Sector Table size to be the constraining factor.

The value of SMART Attribute 197, especially its normalized value, depends on the amount of free space in the Pending Reallocation Table. When this table is empty, the normalized value of SMART Attribute 197 will be 100. When this table is full, the normalized value of SMART Attribute 197 will be 1. Any write command will first search this table, and if the sector being written to is found in this table, then reallocation will be performed before the data is written. In some drives, the Pending Reallocation Table determines the values of *both* SMART Attribute 197 and 198 (SMART 198 is also known as Offline Scan Uncorrectable Count), because in practice both SMART attributes refer to sectors that are pending reallocation.

When a drive runs out of spare sectors or space in its Reallocated Sector Table and, correspondingly, the normalized value of SMART Attribute 5 reaches 1, the drive will no longer be functional. In such a scenario, no spare sectors are available for reallocation. In such a scenario, to prevent data loss, the drive will go into a "Read Only" mode where the drive can accept read commands but cannot accept write commands. This "Read Only" mode indicates

that the user must not write any more data to the drive, and that the data on the drive should be transferred to another drive for data recovery.

Similarly, when the Pending Reallocation Table is full, SMART Attribute 197 will drop down to 1. And the drive will not function properly, but instead, enter a "Read Only" mode.

D. Bad Sectors

Unreadable sectors, which are also known as bad sectors, have the potential to occur in any disk drive. Bad sectors can be formed at any point in the drive's lifespan, from the manufacturing process to the end of the drive's life. Bad sectors formed during manufacturing are called "primary" defects. Primary defects are usually handled before the drive leaves the factory. Bad sectors that form after a drive has been shipped from the factory are called "grown" defects. Grown defects can be handled to an extent by defect handling methods that will be described later.

Bad sectors are an inherent part of storage drives. Such drive imperfections are expected due to the complex construction of drive heads and media. For this reason, spare sectors are designed to provide backup for defective sectors.

Bad sectors are caused by factors that impair the ability of the drive to retrieve data from the media. Bad sectors may result from electrical issues, mechanical issues, or a combination of the two.

A list of potential reasons for bad sectors may include:

a. The head may have deteriorated due to age and become less sensitive.
b. Imperfections may have been introduced into the media during manufacturing.
c. The head may have hit disturbed particles in the drive, causing scratches on the media. Such particles may have been created during the drive's shipping process, by shock and vibration, by aging, or by a host of other reasons.
d. The head may have scratched the media due to an external force, such as a shock or vibration, or due to the mechanical effect of hot or cold temperatures.
e. The height of the head may have changed, either becoming higher or lower than typical, resulting in a weak write. This often results from shock and vibration.
f. Off-track writes due to either servo tracking issues or shock and vibration events may have caused the data to be written onto an adjacent track. This may cause bad sectors on the track that was supposed to be written to as well as on the track that was accidentally infringed upon.
g. The power may have been shut off during a write operation.

The latter three causes of bad sectors are not permanent. Most bad sectors caused by these reasons can be fixed by writing over them.

Drive electronics implement error detection and correction methods, such as Error Correction Code (ECC) and Partial-Response Maximum-Likelihood (PRML), to detect and repair bad sectors. Error correction techniques can recover a small number of bad sectors.

Other methods of recovering data from a bad sector include reading the data multiple times, moving the read head slightly off track, reinitializing the head, readjusting the servo systems, etcetera. However, these processes take time and slow down the command response.

Data recovery that takes too long may make the system feel unresponsive and lead to user frustration. When bad sectors are caused by a group of scratches, the user may feel that the system is not running due to the several read retries that take place, and the user may reset the system to make the system run faster.

If such a system is on a server, the administrator will need to choose between providing slow service or retrieving the data from other drives of the RAID system without further retries by the system. The desire for faster system responses is the reason why some systems have their command time-out (CTO) values set relatively short.

E. Bad Sectors Detected in the Factory (Primary Defects)

Hard drive media are manufactured to have smooth surfaces. However, imperfections on the media created during the manufacturing process are unavoidable, and they cause bad sectors. Therefore, the manufacturing process includes the detection and handling of such media imperfections.

One way that factories detect media imperfections is by executing special software to validate the drive's data. The detection of media imperfections is a time-consuming step in the factory. Media imperfections are detected by scanning the disk from beginning to end for bad spots. Small bad spots can be corrected by the ECC and may be permissible. Large bad spots may result in the drive retiring affected sectors or tracks. Extremely large bad spots may result in an entire disk or drive being marked as failed. The extent of bad spots in a drive may be used to define the "class" of the drive, such as class "A", "B", or "C," which represents the quality of the drive. Higher-classed drives may be sold at a premium.

There are two methods to handle bad sectors in the factory. The first method is called sector slipping, and the second method is sector reallocation.

Sector slipping relies on the fact that there are a few unused sectors at the end of each track. Sector slipping marks a bad sector and replaces it with the following sector. All sectors after the bad sector are shifted forward by one. For the sake of illustration, let us say that we have a track with five physical sectors: four logical sectors and one spare sector: 1, 2, 3, 4, s. If sector 3 is a bad sector, sector slipping would simply treat the track as having four logical sectors and label its sectors as 1, 2, x, 3, 4. This method minimizes impact on drive performance because

logical sectors are simply moved back by one physical sector. The drive does not need to wait for a full rotation to read upcoming sectors.

Sector reallocation is used when the number of bad sectors on the defective spot exceeds the number of available unused sectors on the track. In sector reallocation, the bad sectors are reallocated to another location on the drive. If the track has many errors, the whole track will be reallocated to a different track.

In SCSI drives, bad sectors detected in the factory are recorded on the Primary Defect List (PLIST). On SATA and NVMe drives, bad sectors detected in the factory are not visible to the host.

F. Bad Sectors Detected After Drives Have Shipped (Grown Defects)

When a drive writes a piece of data for the first time, it always assumes the written data is good. That is, drives do not check the validity of data after their initial writing. Bad sectors are only detected after written data is later read.

It is important to understand that from a disk drive's point of view, bad sectors are defined as locations on the media where data are unable to be retrieved after they have been written. Many bad sectors are recoverable via the drive's ECC data recovery mechanism or other methods. For bad sectors that cannot be recovered (i.e., UNC or Uncorrectable Errors), some are unrecoverable due to permanent conditions, such as scratches on the media, and some are due to temporary conditions, such as the data being written off-track, being overwritten by adjacent tracks, or being written weakly. Recoverable bad sectors are also known as "soft errors", and unrecoverable bad sectors are also known as "hard errors." These two types of bad sectors will be further elaborated upon in the “Soft Errors and Hard Errors” section.

Unrecoverable bad sectors that result from permanent damage need to be reallocated to a spare sector. Unrecoverable bad sectors that result from temporary conditions can be fixed by writing over them.

When a drive detects an unrecoverable bad sector, the first thing it does is log the defect in the Pending Reallocation Table. Uncorrectable bad sectors logged in the Pending Reallocation Table await the next time that they are written to by the host. Uncorrectable bad sectors in the Pending Reallocation Table are not immediately written over or reassigned because there may still be a chance that the data on the bad sector can be recovered if environmental conditions, such as the temperature, change.

After a drive encounters an unrecoverable bad sector, it will also send a UNC Command Status report to the hard disk driver. We will elaborate on the hard disk driver in Section 3.

Whenever the host writes to a drive (which includes information about the target LBA, the number of applicable sectors, and the data to be written) the drive will always check if the LBA

of the write command refers to any sector in the Pending Reallocation Table. If not, the write proceeds normally. If yes, the drive will write to the LBA and then check if it can read what it just wrote. If not, the drive will assign the LBA to a spare sector, and all future writes to this LBA will go to this spare sector, which involves: a) the drive creating an entry in the Reallocated Sector Table containing the bad LBA and the spare sector that the LBA has been reallocated to, b) the drive incrementing SMART Attribute 5 by 1 and set the corresponding SMART trip to "true" if the trip condition is met (e.g., if SMART Attribute 5 Normalized drops to 10), c) the drive removing the LBA from the Pending Reallocation Table, and d) the drive decrementing SMART Attribute 197 or 198 by 1. If yes, the drive will remove the LBA from the Pending Reallocation Table, decrement SMART Attribute 197 or 198 by 1, and not reallocate the LBA.

The LBA of an uncorrectable bad sector may be reallocated to spare sectors in one of several locations. However, to minimize performance impact, spare sectors that are closer to the uncorrectable bad sector will be given higher priority for reallocation than spare sectors further away. Spare sectors located at the end of the affected track usually have the highest priority. For drives implementing zone bit recording, spare sectors located at the end of the affected zone usually have the next highest priority. And spare sectors at the end of the media usually have the lowest priority.

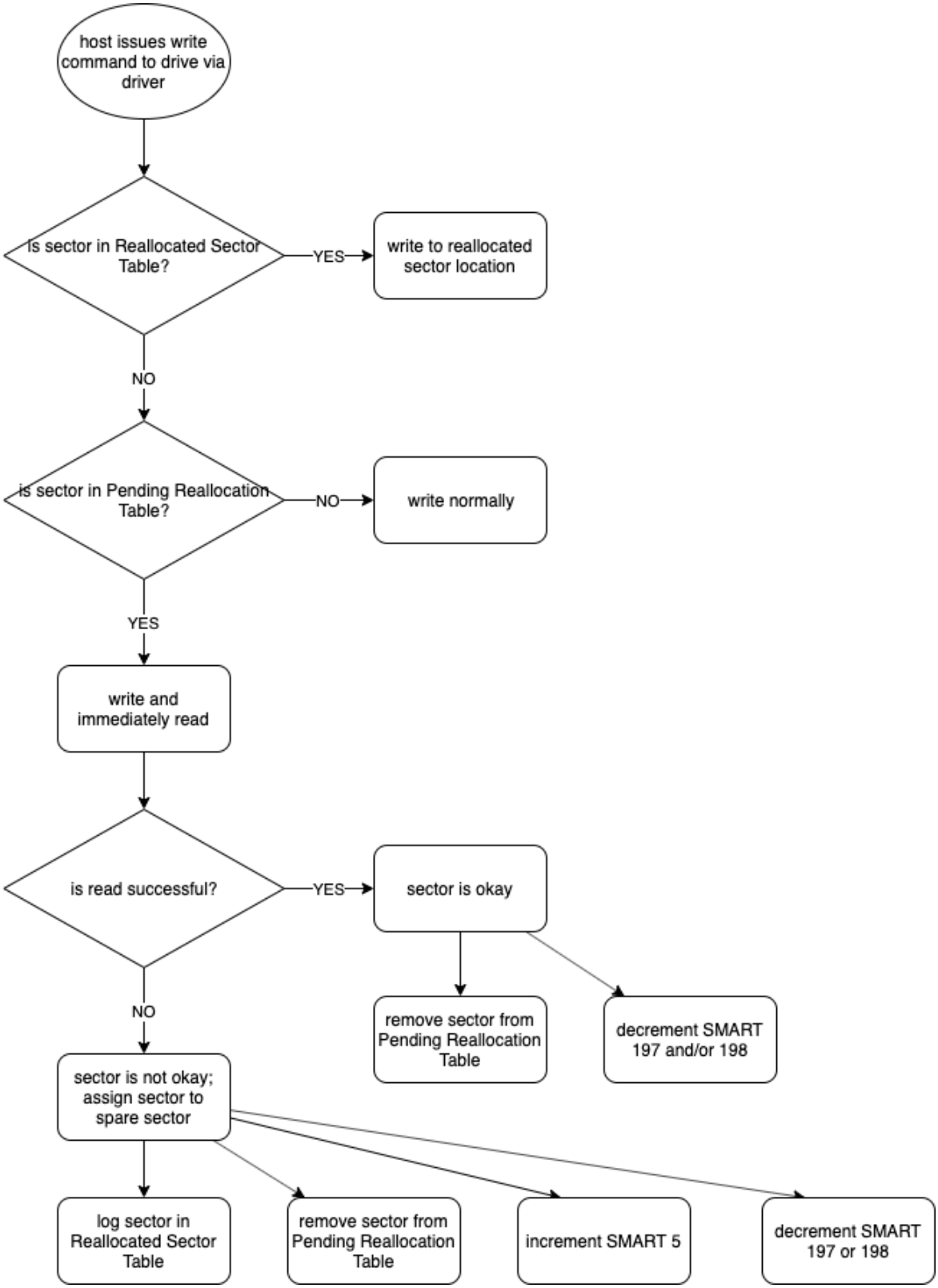
It is normal for a write command to have multiple sectors that are pending reallocation. The drive may implement a policy to reassign an entire track if many of its sectors need to be reallocated. However, whether this policy is in place is usually known only by the drive's maker.

In some vendor-specific cases, a drive may perform a more discreet type of sector reallocation called "automatic reallocation." Two versions of automatic reallocation are described as follows:

a) When a drive that implements automatic reallocation successfully reads data from a sector after a certain number of failed retries, the drive may reallocate the sector without letting the host know. In such a scenario, SMART Attribute 5 would reflect the reallocation, but SMART Attribute 197 would not because the bad sector is reallocated immediately after it is detected. To illustrate this further, let us consider an example where a drive must unsuccessfully read a sector 10 times before listing it in its Pending Reallocation Table. If this drive implements automatic reallocation, it will also have a lower threshold of 3 retries, where if the data can be retrieved from the sector within 3 read retries, then no action other than data retrieval is taken; but if the data is retrieved anywhere between 3 and 10 retries, then the data is sent to the host while the drive discreetly reallocates the compromised sector.
b) Some implementations of automatic reallocation allow the drive to reallocate bad sectors during the drive's self-test. That is, if the drive's self-test detects a bad sector, and the data can be retrieved within some limited number of retries, then the drive may perform automatic reallocation as described in a).

Figure 2: A typical flow for bad sector identification

Figure 3: A typical flow for bad sector handling



G. Soft Errors and Hard Errors

A soft error is a bad sector that occurs while the drive is reading data, but the error is recovered by the drive's data recovery methods. Recovery methods include the ECC method and the PRML method, re-reading, offtrack reading, modified electrical signal sensitivity, and algorithms that are implemented inside the drive's electrical and mechanical systems.

The exact definition of soft errors has not been standardized. However, the common understanding is that errors only count as soft errors if a retry is needed. That is, if an erroneous piece of data can be corrected "on the fly" without impacting drive performance, it does not count towards a soft error. For example, if an erroneous piece of data is retrieved but corrected by ECC, and the next sector is read without delay, then that erroneous piece of data does not count towards a soft error.

Some definitions of soft errors *do* count sectors that have been corrected. However, this type of definition arguably exaggerates the frequency of soft errors because the chance of data correction can be as high as 1 in 10,000 bits.

In contrast with soft errors are hard errors, which are also known as Uncorrectable Errors (UNC). The rate of hard errors is referred to as the Uncorrectable Error Rate (UER) or Uncorrectable Bit Error Rate (UBER). Typical desktop drives have a UER of less than 1 in 10^15 bits, or 125 terabytes of data. Enterprise-grade drives designed for use in RAID and have time-limited error recovery (TLER) have a UER of less than 1 in 10^17 bits, or 12.5 petabytes of data. This requirement is stated in the Open Compute Project (OCP) storage specification.

In addition to soft and hard errors, there is another type of error called an Un-Detectable Error (UDE). UDEs are caused by the limitations of the drive's Error Detection Code (EDC). EDCs are pieces of additional data attached to the main data for the purpose of detecting errors. Therefore, UDEs are the data errors that escape the drive's error detection logic. The chance of encountering a UDE is very low -less than 1 in 10^22 to 10^23 bits (1.25 zettabytes to 12.5 zettabytes of data), depending on whether the drive is desktop or enterprise-grade.

H. Error Reporting by the Drive

The exact information contained within error reports generated by the drive is interface-dependent. For SATA drives, drive error reports will return an error status and error code. The error status is a byte of data that indicates whether there is an error, and the error code is a byte of data that indicates the type of error. For example, a SATA command with no errors might return an error status of 50h to indicate that no errors have occurred, and an error code of 00h to indicate a null error type. On the other hand, a SATA command that results in a UNC would return an error status of 51h to indicate the presence of an error, and an error code of 40h to indicate that the error is a UNC. Along with the error status and error code, SATA drives will also report the error-causing LBA in the response package.

SATA drives can have the following error types: Interface Cyclic Redundancy Check (CRC) Error, UNC, ID Not Found Error, Abort Error, and Command Completion Time Out Error. SCSI and NVMe drives have similar error types, but SCSI drives have more detailed error information.

Even though the exact information contained within error reports generated by the drive is interface dependent, all drive error reports will contain some similar information. For example, all drive error reports will indicate the type of error, and the location of the first LBA that encountered the error.

I. SMART Attributes

Bad sectors, among other drive health issues, can be reported to the user through SMART. SMART stands for Self-Monitoring, Analysis, and Reporting Technology. It is a set of drive health indicators designed to monitor the quality and performance of disk operations, analyze drive issues, and report the condition of the disk drive to the host.

SMART was initially defined for the SATA interface. The following descriptions are primarily defined for the SATA specification. The SCSI and NVMe interface also have similar specifications.

Because SMART information is not always standardized, its values and interpretations may be different between manufacturers. However, the storage industry, led by the major HDD companies, is *trying* to standardize the meaning of each attribute. Therefore, the value of SMART Attributes can be used as a reference.

SMART calls each of its indicators “attributes.” Attributes include information observed by the drive over the course of its operation, such as temperature, power-on hours, and reallocated sector count. SMART attributes are monitored and recorded by the disk drive and are available for the host to examine.

Each attribute comes with two pieces of metadata: an attribute ID and an attribute name.

Each SMART attribute also comes with five pieces of data: a raw value, a normalized value, a threshold, a worst value, and a pre-fail/advisory flag.

The raw value is simply the raw value of the SMART attribute being measured, such as the actual number of hours that the drive has been powered on.

The normalized value is the SMART attribute scaled to some value between 1 and 253, where values greater than 100 are typically good, and higher values are typically better.

The threshold is the normalized value below which a SMART attribute will indicate that it has been “tripped.” A SMART trip can be thought of as an alarm sent from the drive to the host that

requires an immediate response. When a SMART trip occurs, users are advised to replace their drive.

The worst value is the worst value the SMART attribute has ever attained.

The pre-fail/advisory flag indicates whether a trip of the attribute signals impending drive failure according to the manufacturer.

III. Reading and Writing to a Disk Drive

This section describes the steps that occur when a host issues a read or write command to a disk drive. Knowing this process will provide us with a deeper understanding of how bad sectors are detected and managed by the drive's host, and how they affect the user's experience. As an overview, during a read or write command: [A] the command will be issued by the host to the driver, the driver will translate the command and issue the command to the drive, [B,C] the drive will interact with data, the drive will generate a status report, [D] and the driver will react to the outcome of the command if the driver detects an error reported by the drive.

A. The Driver Issues a Read/Write Command to the Drive

The host accesses connected devices, such as hard disks, using pieces of software in the host OS known as drivers. A hard disk's driver translates the host's requests, such as read and write commands, into drive-recognizable commands. However, in communicating with the drive, the driver must also account for the different interfaces between the host and disk drive, which include SATA, NVMe, and SCSI. When the driver sends a command, it will follow the command specification unique to the applicable interface. However, most interfaces accept commands based on a similar format. That is, all read and write commands contain three major components: the Op code, LBA, and length.

The Op code is a technical term for the specific command. For example, a read command will have a unique Op code. Other commands will each have their own Op codes. Typically, there will be tens to hundreds of Op codes defined for a given interface. Each Op code refers to a different command as defined in the specification of the applicable interface. Many of the Op codes are for drive maintenance purposes, while some commands are for reading and writing data.

The LBA is the address of the data being requested or read. The drive firmware will translate the LBA into a physical location on the drive.

The length is the size of the data in terms of blocks. The block length defines how much data will be read or written. For example, if the block size for a drive is 512 bytes (a common block size), and a read command has a length of 4, then the read command will return 2048 bytes of

data. Because a disk drive stores data in sectors, there is another translation from the host blocks to the physical sectors.

Initially, drives were designed with 512-byte sectors. A sector is the minimum unit of data the drive can read or write from the drive. Each sector has its own starting and ending marks, as well as an error protection mechanism. To improve the recording efficiency on disk drives, many newer designs now have 4K-byte sectors. Larger sectors reduce sector overhead. It is estimated that the space savings of a 4K sector can be up to 8%.

For years, the established block size for disk drives has been 512 bytes. Most drivers have been written with this standard. To accommodate the 4K-byte sector size on disk drives, some hosts have also increased their block size to 4K bytes. However, the 4K-byte host block size is not very popular and is mostly implemented in server environments with special drives.

It is important to remember that, in most cases, the driver will read or write to *multiple* sectors on a drive in a single command. And it is considered more efficient to include a larger number of sectors in a single read or write command. For example, it is more efficient for a host to issue one command targeting 100 sectors rather than issuing 100 commands targeting one sector each. This is because more commands require more processing and cause performance to slow down as a result.

However, even though drivers prefer to include more sectors in a single command, the actual number of sectors that can be included per command is constrained by the command structure, buffer size, and other factors. The command structure specifies the maximum length of each command. Buffer size refers to the size of the read and write buffer in the host system and in the disk drive. Most frequently, commands are issued to 128 or 256 sectors at a time. However, depending on the driver and system, other sector counts per command may be possible.

When commands are issued one after another, it is called "sequential command execution." When commands are issued in parallel to a drive that can accept more than one command at a time, it is called "command queueing." Parallel command execution may improve performance. All major interfaces (i.e., SATA, SAS, and NVMe) support command queueing.

B. The Drive May Encounter Errors After Receiving a Read or Write Command

Drives are designed to perform read and write operations. However, the drive may encounter errors while performing these tasks. Errors due to read commands are more common than errors due to write commands because reads occur more frequently than writes.

Write errors are usually related to servomechanical issues. Examples include the drive being unable to put the recording head on the right track, writing while the head is flying too high or low, or the electrical signal being incorrect. These errors can be serious but occur rarely.

Read errors (i.e., bad sectors) are related to the media. This can manifest in the following ways at the drive level:

i) The drive cannot find a sector location's identification mark (address mark).
ii) The drive cannot detect any signal from the media.
iii) The requested sector's EDC detects an error.
iv) The number of retries surpasses the drive's retry limit.
v) The drive's read command timeout has been reached. For example, if the read command timeout is 5 seconds, and the read takes longer than 5 seconds, the drive will generate a read error. Please note that the drive's read command timeout limit applies to all sector reads, and the timeout occurs when the read takes longer than this timeout limit.

C. The Drive May Detect and Repair Bad Sectors

When a drive issues a read command, it may attempt to read sectors that belong to damaged media (commands usually target multiple sectors). As soon as the drive (specifically, the drive's firmware) encounters the *first bad sector in the command*, it will mark that bad sector as having a UNC error, record that bad sector in the Pending Reallocation Table, increment SMART 197 for that bad sector, report that bad sector to the driver, and discard whatever data was read from this sector. This sector will be reallocated when the host writes to it later.

However, potential subsequent bad sectors in the command will not be identified nor reallocated on subsequent writes. This happens because, during the read command, as soon as the drive encounters the first bad sector, it terminates the read operation and does not identify other potential bad sectors covered by the command. Drive Self-Tests can be used to overcome this limitation.

D. The Driver Responds to the Outcome of the Command

In addition to sending commands to hard disks, the driver must determine the outcome of the command, handle the outcome, and report the outcome to the host.

Typically, the command will succeed, the drive will report to the driver that the command succeeded, and in turn, the driver will report to the host that the command status is Good or OK.

However, if a read command fails, the drive will report a UNC Command Status Report to the driver, and the driver will normally retry the command 3-5 times, with the exact number of retries depending on the specific driver or the policy set by the system. If any of the retries succeed, the driver will report a good result to the host and move on to the next command. If all the retries are unsuccessful, the command will fail, and the driver will report to the host that the command status resulted in an Error. There is a limitation on the number of retries

attempted by the driver because the system must meet certain standards of performance and responsiveness.

And if the drive reports a Command Timeout (CTO) to the driver, or the driver detects that the command is taking too long according to its own CTO policy, the driver will jam the issued command with a reset or a new command, telling the drive to give up its internal recovery process and resetting the internal state of the drive. This jamming will cause the drive to stop its current read or write operation, and consequently, no bad sectors will be recorded for the unrecoverable block; nor will any sectors be marked as pending reallocation, so the offending bad sector will not be reallocated when the host writes to it the next time.

If the driver times out, the system may use RAID information to reconstruct data rather than spending more time performing command retries. In the extreme case, the driver may have such a short timeout that no retries are ever performed. Upon encountering a problem with a read command, such a driver will immediately give up the read command, construct the data from RAID drives, and send the data to the host. The system makes this choice by considering the trade-off in performance. If getting the RAID-reconstructed data is faster than a retry, a retry may be deemed unnecessary.

It should be noted that hard disks and their drivers each have their own timer for determining when a command is taking too long. If the driver's command timeout limit is shorter (e.g., 3 seconds) than the drive's command timeout limit (e.g., 5 seconds), the driver may terminate a problematic command before the drive does. In such a scenario, the drive will not report any errors because the command was reset by the host.

The CTO limit of the driver is determined by the system, which is typically not adjustable by the user. However, a few systems do allow users to set command timers. For example, QNAP Systems allows their users to set a time limit for read and write commands by specifying the number of seconds to allocate for time-limited error recovery (TLER), error recovery control (ERC), and command completion time limit (CCTL).

In general, the TLER/ERC has higher priority than the number of retries, because the time it takes to retry a command is more important than the number of retries. During error recovery, the retry time determines the system's performance. Depending on how long a read retry takes, a system may give up on the retry and return data reconstructed from redundant data on other drives instead. Getting data reconstructed from redundant sources is sometimes faster than waiting for a long retry and can provide a better quality of service. In some cases, if there is no RAID available, the system may ask the user if they wish to continue retrying a read command. And in some cases, such as during video playback, a video glitch may be more acceptable than paused playback.

IV. Disk Drive Self-Tests

The disk drive self-test is a way for the disk drive to examine its own health, and to look for newly formed bad sectors. A disk drive comes with its own firmware for performing a self-test. Self-tests are included as part of the disk drive's design for the purposes of improving the quality of the disk operation, preventing drive problems, and repairing damage to the drive. This section describes popular self-test methods used in disk drives.

A. Summary of the Drive Self-Test

Drives perform self-tests to identify media, read head, write head, servomechanism, spindle, and other electrical and mechanical issues. Self-tests can detect drive issues before drive failure. Because a drive is an intelligent computing subsystem with a built-in processor and its own software, a drive can detect and record its own problems.

When drives are built, they undergo a special testing program to ensure their quality before they leave the factory. This type of testing program is generally called a “burn-in test.” The testing program evaluates the mechanical characteristics, media characteristics, and servomechanism of the drive. It also detects defective spots and maps out defective spots to other locations on the drive. As drive capacity increased over time, the length of the factory testing programs increased from hours to days.

After a drive is commissioned to operate in the field, the drive can periodically perform self-tests to detect drive issues. Unlike the factory test, a self-test is not used to determine the class of the drive, but rather is used to detect and protect the drive against failures.

However, a drive self-test may reduce the performance of a drive during its normal operation because the self-test occupies some of the resources the drive uses to service the host's requests. Therefore, a self-test can be viewed as a trade-off between the performance and reliability of a drive. There is also an option to perform self-tests during off-peak hours.

The result of a self-test is stored in the self-test error log and self-test error list, which reside on the drive in an area that also contains drive metadata. In addition, the count of offline scan errors detected during a self-test is also recorded in SMART Attribute 198, which has a similar meaning to SMART Attribute 197.

B. Short and Extended Self-Test

Self-tests can be categorized into short and Extended Self-Tests. Short self-tests usually take 2 minutes and perform basic read/write, electrical, mechanical, and servomechanism testing. Extended Self-Tests usually perform a complete media scan. Because the capacity of disk drives has increased to many TBs, the scanning of the media can take many hours or days to complete.

Short self-tests can also scan the media if the media scan can complete within 120 seconds. During a short self-test, some drives will just scan important sectors, such as the beginning of the drive or vulnerable tracks, such as those near the landing area, to validate that those sectors are good.

C. Online and Offline Self-Test

Another way to categorize self-tests is into online and offline tests. During an online self-test, the drive will continue the test and only report to the host after the test is complete. During the test, no other command can be accepted and/or performed by the drive. If an online short self-test is run, the drive will be unresponsive for 2 minutes, and if an online Extended Self-Test is performed, the drive may be unresponsive for several hours. Online self-tests are seldom used except during drive manufacturing or in a controlled environment.

Offline self-tests are performed in the background. The test progress (shown in terms of the percentage of completion) can be polled by the host, and the estimated remaining test time is also reported by the drive.

When a drive is performing an offline self-test, it can receive other commands, such as read/write commands. As soon as one or more read/write commands are received, the drive will suspend its current offline self-test and perform the read/write requests. To accommodate and prioritize the foreground read/write tasks, there will be a vendor-defined timer to resume the offline self-test in the background once the foreground task is complete. For example, if the "resume" timer is set for 3 seconds, the drive will wait for at least 3 seconds with no read/write requests before it resumes its offline self-test. Offline self-tests are the most common form of self-test because the drive can continue to accept read/write operations while the test is running. However, with offline self-tests, there is an impact on drive performance due to the time required for the drive to transition from self-test operations to regular read/write operations. But due to the implementation of the "resume" timer, the impact is perhaps minimized.

Both online and offline self-tests will affect drive performance by slowing down the host's read/write requests. It is recommended to perform most self-tests during off-peak hours to avoid impacting performance.

D. Auto Self-Test

Drives may also implement automatic self-testing based on a vendor-defined schedule. The auto self-test is a feature implemented by some drives during manufacturing, sometimes at the request of original equipment manufacturers (OEMs). For example, a drive can be configured to run an automatic self-test every week or month.

Automatic self-tests are initiated inside the drive without a host command. When the drive's internal timer is up, the drive will start to perform a self-test. Automatic self-tests are performed in the same way as an offline self-test. In other words, during an automatic self-test, the drive can accept commands by suspending the self-test, and once the timer has expired, the drive will resume its automatic self-test.

Most of the time, automatic self-testing is enabled and configured at the request of OEM customers. Drives shipped to the general market do not have this feature turned on for fear that the performance impact may be unacceptable. OEM customers who desire rigorous drive testing may ask drive manufacturers to enable automatic self-testing to improve the drive's quality.

V. Bad Sector Handling by NAS Systems

This section describes some potential ways that network-attached storage (NAS) systems handle bad sectors. This includes the ways that NAS Systems identify and fix errors.

A. RAID in the Degraded Mode

A NAS device is a computer specialized for storing data and providing that data over the internet or a network. When a NAS device encounters an uncorrectable read error, it will issue an io error event, set a Faulty condition, and put RAID into a "degraded" mode.

If the NAS supports RAID, the NAS will reconstruct the information on the bad sector from redundant information on other drives. This reconstructed data should be usable by the system.

i. However, due to this UNC, RAID will enter a degraded mode. This lets the NAS user know that there may be no protection from data loss if other drives within the RAID fail catastrophically (e.g., are unable to spin up).
ii. One rationale for having a RAID is to ensure that if a drive breaks, the data that was on that drive is still available. If one drive is removed from a RAID that is set up for redundancy and a new one replaces it, the RAID array can be rebuilt.
iii. When one UNC exists, the RAID usually cannot tolerate another drive's removal or failure, because if you wish to rebuild the drive having the UNC (i.e., replace the drive having the UNC with a new drive and rebuilding the information from the old drive on the new drive via the redundancy provided by the other drives in your RAID) and another drive is removed or fails, then the sector of the UNC area cannot be salvaged (unless the RAID is set up to tolerate more than one failure, such as RAID 6). This is the reason the NAS placed the RAID in a degraded mode.
iv. When the RAID is in degraded mode, the user should take action to prevent further issues. These actions may include:
    (1) Replacing the drive with the UNC with a new drive and rebuilding the RAID.
    (2) Removing the drive with the UNC, reinserting the drive, and rebuilding the RAID.

(3) Performing a self-test with the drive with the UNC, reinserting the drive, and rebuilding the RAID.
(4) Or repairing the UNC-causing bad sector by RAID Scrubbing. Raid Scrubbing can scan the RAID and repair the UNC-causing bad sector with data calculated from other drives in the RAID. When the UNC-causing bad sector is repaired in this way, the drive will reallocate the bad sector.

B. Detecting Bad Sectors in a RAID

There are several methods to find bad sectors in a RAID:

1. Through read commands
2. By scanning for bad blocks
3. By scrubbing the RAID
4. By performing a

Methods 1 to 3 have a drawback. As mentioned previously, when the host issues a single read command, it usually does so to multiple sectors (e.g., 32 or 128) for efficiency reasons. When there is an uncorrectable error on the read command, the error will only report the LBA of the first occurrence of the UNC. In other words, if there are multiple bad sectors in the command, the drive will only report the first bad sector. Subsequent bad sectors in the command will be ignored by the host and drive. The drive may only list the first bad sector in the Pending Reallocation Table.

Methods 1 to 3 are also affected by the host's command timeout limit. When the drive's internal command timeout limit is longer than the host's, the host may terminate a command with a reset before the drive finishes the command. In such a scenario, the drive will not record any bad sectors detected. SMART Attribute 197 (Current Pending Sector Count) will not be incremented. And bad sectors will not be reallocated when new data is written to them.

These two drawbacks are not present when performing a Self-Test. When a drive performs a Self-Test, it will typically record every error location. (Note: this may vary depending on vendor implementation). Because there is no limit to the number of errors reported per command, the drive can continue to detect one error after another without missing any bad sectors. This makes the Self-Tests an optimal way of finding entire clusters of bad sectors.

C. Scan for Bad Blocks

NAS companies recommend that users scan for bad blocks to detect any errors on a drive. Scanning for bad blocks uses the driver to read a drive from the beginning of the drive to the end of the drive while detecting bad sectors. However, scanning for bad blocks may not detect all bad sectors if the detection process results in a CTO error or if the error affects multiple bad

sectors in close proximity. Furthermore, the scan for bad blocks does not automatically repair those bad sectors.

D. RAID Scrubbing

The NAS manufacturer, QNAP Systems, defines RAID Scrubbing as follows: “RAID Scrubbing is used to verify the data integrity of disk groups with RAID 5 and RAID 6 configurations. It works by running a redundancy check to detect and correct inconsistencies that are undetectable during routine usage. Periodically running RAID Scrubbing can detect potential corrupted data or disks at an early stage, giving your NAS the chance to attempt automatic repairs or to report disk-related issues, helping to ensure the integrity of user data and disk groups.”

RAID Scrubbing performs a “data scan” over the RAID data area. In some cases, this may be limited to areas containing the user's data instead of the whole drive. RAID Scrubbing detects any “bad” locations in the data area by scanning over the data area and performing a data consistency check to detect any inconsistent data. For example, if the data is readable, but inconsistent with a parity check, then the data on the RAID is bad. Data consistency issues cannot be resolved by the RAID alone. It needs an extra mechanism such as a Btrfs system to correct the error.

The principle behind RAID Scrubbing is to detect bad sectors and use redundant RAID data from other drives to "repair" those bad sectors. The hope is that when data is written to the bad sectors, it can overwrite the previously bad data on those sectors. However, writing data to a scratched sector will not repair the sector. It is still a bad sector unless the drive reallocates the bad sector. The important thing is to let the drive perform the reallocation process to repair the bad sector.

Depending on the RAID Scrubbing implementation, the repair may be partial. For example, if the RAID Scrubbing detects just one UNC out of multiple UNC sectors, it may only mark the first UNC as pending reallocation and repair the first UNC, leaving the rest of the UNCs unrepaired. In the case of host CTOs, the "repair" might only involve overwriting the bad sectors without reallocating them. Therefore, whether RAID Scrubbing actually repairs all bad sectors depends on the implementation.

A more reliable way to repair bad sectors is to perform a Self-Test first, to identify all potential sectors pending reallocation before RAID Scrubbing. When RAID Scrubbing is performed after a SMART Self-Test, all bad sectors will have been added to the Pending Reallocation Table, and reallocation can be correctly performed on all bad sectors.

VI. Recommended Bad Sector Management Practices

This section recommends practices for minimizing the effects of bad sectors, especially for NAS systems. The section is divided into RAID and non-RAID configurations.

A. For RAID NAS Systems

If a drive is in a RAID configuration, such as RAID 1 or RAID 5, and a low-level read or write command fails, then the system will retry the command. If the retry fails, the NAS will increment the IO_Error_Count. If the IO_Error_Count is greater than the IO_Error_Limit (e.g., 512), the NAS will mark the drive as having a "FAULT" error. (Note: the FAULT error is different from the RAID Degraded error.)

When a "FAULT" error is encountered, the following steps are recommended:

1. Perform a Self-Test (SST) with the Extended SMART Test option to identify bad sector locations.
2. If the Self-Test with Extended SMART Test option detects bad sectors, perform RAID Scrubbing to fix the bad sectors.

The Self-Test (with the Extended Option) can scan the whole drive, detect all bad sectors, and mark the problematic sectors as pending reallocation, even if the bad sectors exist in close proximity. Future data written to these pending sectors will initiate the reallocation process and fix the bad sectors. Self-Tests solve the issues with CTOs and multiple bad sectors in close proximity by marking all potential bad sectors for reassignment.

RAID Scrubbing alone may not fix all the bad sectors if the errors are detected by CTOs or caused by multiple bad sectors targeted by a single command. When there is a CTO error, RAID Scrubbing simply finds the location of the CTO error, reads the data that is supposed to be there from redundant RAID data, and writes the data to the bad sector. Because the bad sector is detected by a CTO, no pending reallocation mark is set, and data written to the bad sector will still be bad. For multiple bad sectors targeted by a single command, only the first bad sector is marked as pending reallocation, and the remaining bad sectors are not marked. Writing to the remaining bad sectors will not fix them.

Therefore, the recommendation is to perform a Self-Test with the Extended SMART Test option first, followed by RAID Scrubbing. This process can detect all bad sectors and repair them.

B. For Non-RAID NAS Systems

If drives are not in a RAID configuration (i.e., there is no data redundancy), data may simply be lost when a read or write command fails. On non-RAID systems, such as Just a Bunch of Disks (JBOD) systems, a bad sector can cause data loss.

Recommendations when a read or write command fails:

If there is backup data available, the recommended steps are:

1. To perform a Self-Test with the Extended SMART Test option. This identifies all bad sectors and marks them for reallocation. When new data is written to them, they will be reallocated.
2. To restore the affected file from a backup.

If there is no backup available, the options are:

1. To perform a Self-Test with the Extended SMART Test option. This identifies all bad sectors and marks them for reallocation. When new data is written to them, they will be reallocated. After this step, one can take the following options:

   Option 1: Delete the affected file. 99% of the time the bad sector is on the data field instead of the system metadata. By removing the bad file, the bad sector becomes available for later writes.
   Option 2: Clone the drive to a new drive and ignore the error that might appear when cloning the drive.

Non-redundant systems, such as JBOD, are meant for use in environments that can accept data loss. Such systems may be appropriate when the data is non-essential, or if the user is relying on backups or copies of the data stored in multiple locations to recover the data. Many modern PC users are in this situation.

Because there is no data redundancy within the system, the data cannot be recovered from the system itself. In many cases, the user will have data loss. Though a few options for data recovery may be available, it will often remain uncertain whether the condition of a drive warrants drive replacement. Bad sectors can occur in any drive, healthy or not. If the condition occurs many times, it is better to replace the drive.

VII. Summary

Disk drives are precision instruments that form an essential component of modern computers. Most of the time they work as intended. However, due to their environment or operating conditions, the sheer volume of operations they undergo on a daily basis, and the tight tolerances under which they must operate, occasional bad sectors are expected to occur. Given that they are expected, drives, drivers, systems, and RAID all have built-in mechanisms for handling them. Their occurrence, while not always catastrophic to the drive, can result in data loss if the user is not prepared. This type of data loss can be minimized through best practices, such as backing up data, setting up RAID, periodically performing Self-Tests with the Extended SMART Test option followed by RAID scrubbing, rebuilding a volume from backup or RAID when necessary, and monitoring of drive health indicators to check whether bad sectors are on the rise.

VIII. References

1. https://docs.qnap.com/nas-outdated/QTS4.3.4/en/GUID-0BE980D3-1190-486B-A1E8-228DB71A9C93.html
2. https://www.qnap.com/en/news/2017/automatically-enabled-raid-scrubbing-with-qts-4-3-30210-build-20170606#:~:text=RAID%20Scrubbing%20is%20used%20to,are%20undetectable%20during%20routine%20usage.
3. https://www.qnap.com/en/how-to/tutorial/article/how-to-use-data-scrubbing-to-prevent-data-corruption
4. https://en.wikipedia.org/wiki/Data_scrubbing
5. https://louwrentius.com/scrub-your-nas-hard-drives-regularly-if-you-care-about-your-data.html
6. https://www.qnap.com/en/how-to/faq/article/my-raid-is-in-degraded-mode-what-should-i-do