The Old New Thing

Where is this CRC that is allegedly invalid on my hard drive?

11 Jul 2013 7:00 AM | 29

If you're unlucky, your I/O operation will fail with ERROR_CRC, whose description is "Data error (cyclic redundancy check)." Where does NTFS keep this CRC, what is it checking, and how can you access the value to try to repair the data?

Actually, NTFS does none of that stuff. The CRC error you're getting is coming from the hard drive itself. Hard drives nowadays are pretty complicated beasts. They don't just plop data down and suck it back. They have error-checking codes, silent block remapping, on-board caching, sector size virtualization, all sorts of craziness.

What's actually happening is that the file system asks the hard drive to read some data, and instead of handing data back, the hard drive reports, "Sorry, I couldn't read it back because of a CRC error." NTFS itself doesn't do any CRC checking.

"Well, that's awfully misleading. If NTFS is reporting a CRC error, then that makes the user think that NTFS is maintaining CRCs. Shouldn't it just report 'general I/O error' instead of a more specific error?"

NTFS is just bubbling up the BIOS reported hard drive error codes, and those error codes were returned all the way back to the application. Who knows, maybe the end-user knows enough about drive technology that they can tell the difference between a CRC error and a seek error. (For example, a seek error may be fixed by removing the floppy disk and reinserting it, or by recalibrating.)

What about the converse? If an I/O operation completes successfully, does that provide metaphysical certitude that the data read back exactly matches the data that was originally written?

No. It only provides metaphysical certitude that *the hard drive reported that* the data read back exactly matches the data that was originally written, *as far as it could tell*.

Generally speaking, upper layers of a system trust that a lower layer is functioning properly (and often they have no way of detecting a malfunction in the lower layer, anyway). If the hard drive says that it read the data successfully, well, the hard drive is the expert at this sort of thing, so who are we to say, "Nuh uh, I think you're wrong"?

Blog - Comment List MSDN TechNet

Comments

alegr1
11 Jul 2013 7:35 AM
#

Another factoid:

The modern HDD gives a lot of effort to recover a faulty block. If it was possible, the data is then written to one of spare locations, and the sector replacement table is updated. If the data recovery was not possible, the block is marked for replacement anyway, and the next write to it will go to a replacement location.

When the data is written, a pretty complex error correction code is added to it. It allows to correct quite long error bursts.

NTFS gets error codes returned by disk.sys, which are derived from the sense data returned in the storage port SRB. BIOS is not involved anymore.

Steve Wolf
11 Jul 2013 8:05 AM
#

Whomever is arguing "shouldn't it give us LESS information" should be summarily shot. As an example to others. Stupidity!

alegr1
11 Jul 2013 8:39 AM
#

And when the I/O error happens, disk.sys will log an error in the system log. But that log record is absolutely useless, because it refers to the device object name, which is: 1) ephemeral and can change on every boot, 2) there is no way to map it to the actual disk.

dave
11 Jul 2013 8:54 AM
#

>It only provides metaphysical certitude that the hard drive reported

>that the data read back exactly matches the data that was originally

>written, as far as it could tell.

The drive should actually compare the data it reads back to the data it originally wrote....

Joshua
11 Jul 2013 9:16 AM
#

> The drive should actually compare the data it reads back to the data it originally wrote....

Believe it or not, there's a op command to do just that. Nothing seems to use it. There is copy /v but people are starting to wonder if it even sends the right op command to the drive.

AsmGuru62
11 Jul 2013 9:44 AM
#

CRC is stored on the drive and data is stored there.

So, CRC Error (data is not matching CRC) may mean two things:

1. CRC value is OK and data is corrupted

2. Data is OK, but CRC value is corrupted

Does it sound right?

Or I am missing something?

Falcon
11 Jul 2013 9:47 AM
#

@AsmGuru62:

3. Both data and CRC are corrupted

jeff
11 Jul 2013 10:21 AM
#

raymond> >It only provides metaphysical certitude that the hard drive reported that the data read back exactly matches the data that was originally>written, as far as it could tell.

dave> The drive should actually compare the data it reads back to the data it originally wrote....

Dave, how does that work? I wrote my file to disk 2 years ago, and haven't touched it since? How does the drive remember what it originally wrote two years ago? The drive is most likely reporting a CRC mismatch, hence the disclaimer "as far as it could tell"

rlaager
11 Jul 2013 10:38 AM
#

> who are we to say, "Nuh uh, I think you're wrong"

ZFS or ReFS.

alegr1
11 Jul 2013 10:41 AM
#

dave> The drive should actually compare the data it reads back to the data it originally wrote....

Some HDD brands actually did write-read-verify for a few first power-on cycles. Not sure if they do that anymore.

Maurits [MSFT]
11 Jul 2013 11:31 AM
#

NTFS is a transport layer here. Fine.

The questions all still stand. Just apply them to the layer below NTFS instead.

> Where does the HDD keep this CRC, what is it checking, and how can you access the value to try to repair the data?

The answers, I suspect, are:

> The HDD gets to decide where to store the CRC; it's checking the CRC of the data against a previously computed CRC; and NTFS does not define a transport mechanism on its lower boundary to query the either the previously computed CRC, or the new CRC, though you could presumably calculate the new CRC yourself if you knew exactly what algorithm to use.

So there is a valid criticism of NTFS here, namely that it is not as transparent a transport layer as it could be.

Someone
11 Jul 2013 11:58 AM
#

@Mauritis:

I understand your point, but at the same time I disagree. NTFS is about the file system, whereas the disk's CRC stuff is at a block level or some other implementation-defined region. Disk errors should probably be in the purview of a different layer and different tool. (Though maybe NTFS should provide a mechanism like "what blocks am I looking at when accessing foo.txt" so you can find out what question to ask of said tool.)

ms
11 Jul 2013 12:28 PM
#

Rather than dumbing the message down to a generic I/O error, how about going in the other direction and differentiate between CRC error coming from the disk versus CRC errors from elsewhere? That'll take care of questions like "where is this CRC stored and what can I do about it".

Nicholas
11 Jul 2013 12:43 PM
#

I recall reading an article a couple of years ago that said modern hard disk drives (2TB at the time) were completely reliant upon their internal error correction routines for normal operation. That is to say, it was a very rare case that data requested from the disk was read and returned without any problems. The normal case was that data is always a "little bit wrong" and requires being run through the correction algorithms before it is "correct".

That gave me pause looking at my 2TB disk in 2011. Now I look at this 4TB disk for sale and say "Hmmmmm".

It seems like some mean-spirited joke of fate that the devices we use to store everything digital we care about have increased in capacity by several orders of magnitude but largely gotten worse in terms of lifespan and recoverability. And it makes me sad.

hacksoncode
11 Jul 2013 2:07 PM
#

>It only provides metaphysical certitude...

911 operator: "I'm sorry, Search and Rescue can't extract you from Plato's Cave".

Ted
11 Jul 2013 4:46 PM
#

Interestingly, the latest file systems for large arrays do store checksums for all data. See ZFS on Oracle systems, and btrfs on Linux.

alegr1
11 Jul 2013 5:48 PM
#

> modern hard disk drives (2TB at the time) were completely reliant upon their internal error correction routines for normal operation.

PRML basically relies on choosing what bit pattern is the "least wrong" (Maximum Likelihood) for the read signal.

> Where does the HDD keep this CRC, what is it checking, and how can you access the

value to try to repair the data?

If the drive reports CRC error, it already made its best effort to repair the data. Because of using the Reed-Solomon codes, unrepaired data doesn't make any sense, anyway.

>(Though maybe NTFS should provide a mechanism like "what blocks am I looking at when accessing foo.txt" so you can find out what question to ask of said tool.)

Use IOCTL code FSCTL_GET_RETRIEVAL_POINTERS.

Jon
11 Jul 2013 10:25 PM
#

As people have mentioned, the ECC code is handled internally by the drive. For a long time, there have been enterprise systems which go around this transparency issue with a system called end-to-end check or Protection Information. I think IBM has been doing this for decades and I've heard NetApp and EMC do too, but Windows, as far as I know cannot.

It works by using enterprise hard drives which can be reformatted to 520 or 528 bytes per sector. The OS does CRC itself and stores it in the additional bytes, which are passed through and written to disk. That way, data is continuously protected end-to-end, guarding against things like drive firmware bugs, and CRC is visible to the OS. This used to require pricey Fibre Channel disks, but now has made its way into less pricey SAS drives.

Joker_vD
11 Jul 2013 11:24 PM
#

@alegr: "Because of using the Reed-Solomon codes, unrepaired data doesn't make any sense, anyway."

That's not entirely true. There are systematic Reed-Solomon codes, which encode data simply by appending control bits to it. So if you don't care about error detection (and correction), you may simply drop those control bits and get the original data.

Simon Farnsworth
12 Jul 2013 12:59 AM
#

@Jon

You might want to look at the T.10 (SCSI) Data Integrity Feature specification. In DIF, you transfer 512 data bytes and a standardised 8 byte DIF tuple for each sector written or read. Everything in the chain, from the HDD up to the OS knows how to verify that the DIF tuple matches the data bytes - as a result, the HBA can fail a transfer (and enter retry mechanisms) because the DIF field is clearly wrong, the disk can fail a transfer (again, retry mechanisms kick in) because it can't validate the DIF, and the OS can fail a transfer for DIF problems. This catches both bad resting points (HBA, disk, RAM etc) and bad cabling.

Where this really shines (just like btrfs and ZFS) is with a RAID system - the OS can quickly rewrite the bad data, having determined exactly which disks have trouble.

Neil
12 Jul 2013 3:52 AM
#

Does anyone know whether RAID implementations attempt to trigger sector remapping on read errors or do they just fail the drive? I tried to search for something relevant but was only able to find an enhancement request for the "feature" (if you can call it that - you might think that it's safer to fast fail the drive).

jeff
12 Jul 2013 8:03 AM
#

Despite the riveting reportage of the minutiae of CRC handling on various drive types and configurations, the larger point is when you're implementing a particular level of a layered system, don't report stuff that's not helpful to the consumer of your API. The hint is:

raymond> Generally speaking, upper layers of a system trust that a lower layer is functioning properly (and often they have no way of detecting a malfunction in the lower layer, anyway). If the hard drive says that it read the data successfully, well, the hard drive is the expert at this sort of thing, so who are we to say, "Nuh uh, I think you're wrong"?

ErikF
12 Jul 2013 9:29 AM
#

The vast majority of programs that I can think of couldn't care less about why a hard drive failed to read/write something, just that it *did*. That would be an argument in favour of limiting the number of failure cases in my opinion.

If you want to get more information about the error, you're probably some sort of system utility and will by necessity be tightly coupled to the hardware. For example, the T.10 SCSI DIF is great, but assuming that all SCSI devices (let alone all devices!) provide it is probably assuming too much. Personally, I'd punt stuff like ERROR_CRC into extended error and simply return a generic read error for most I/O.

alegr1
12 Jul 2013 10:43 AM
#

I had to read on T.10 DIF/DIX some time ago and so far my opinion is that it is for peace of mind ("reliability theater") and a bullet point in the specs, not for actual reliability.

Everything it provides can be done by other transparent means without having to have it end to end.

Gechurch
12 Jul 2013 6:12 PM
#

@Neil "Does anyone know whether RAID implementations attempt to trigger sector remapping on read errors or do they just fail the drive?"

The firmware of the drive itself handles sector remapping internally. The layers above (including the RAID controller) don't even know it's happened.

Once the drive has run out of spare sectors to remap to it then reports the error to the upper layers. The file system can then handle it (for example, NTFS makes its own note about which sector/s are bad and knows not to try to use them. It also reports bad sectors at this stage when you run a chkdsk, which is a sign to replace the drive). RAID controllers will also see this error and report the drive as being bad. I'm not sure if RAID controllers also keep track of the bad sectors to make sure it doesn't re-use them. It wouldn't surprise me if they do.

Neil
13 Jul 2013 3:46 AM
#

@Gechurch Drives can't remap read errors.

Gechurch
13 Jul 2013 6:53 AM
#

@Neil

I assumed you meant for write errors. Since you did mean read, I either don't understand the question, or the answer is obvious.

If you truly mean 'remap' then I have no idea what you're asking. A write can be remapped because you have the data in memory. If the drive says it can't saved to the sector you're trying to write to then you still have that data in memory and can write it to a different sector. What is the equivalent of this for read? If you ask for the data at a given sector and the drive says "sorry, I can't read that" then how can you remap that read to a different sector? Other sectors of the hard drive contain different data. Asking to have that read 'remapped' to another sector is another way of saying "please return me some garbage data that I don't want".

If you simply mean "if one hard drive in a RAID array returns a read error, will the RAID controller read that data from a different disk in the array" then the answer is always* yes - that's the whole point of RAID.

*Ok, well except for RAID0, but we all know RAID0 isn't really RAID at all.

alegr1
13 Jul 2013 1:56 PM
#

>Drives can't remap read errors

If a transient (recoverable) read error happens, the block is remapped and copied over to the new location.

If a non-recoverable error happens, the block is internally marked as defective and is remapped at the next write to it.

Neil
18 Jul 2013 3:46 AM
#

@Gechurch Sure, I know that at a high level the data is still readable, but will the controller simply fail the disk or will it attempt to rewrite the block so that the disk will remap it?

@alegr1 But does the disk even report a recoverable read error?