http://www.sqlservercentral.com/articles/Memory+Corruption/93424/
Printed 2013/07/04 04:33PM

Memory Corruptions, or Why You Need DBCC CHECKDB

By Paul Randal, 2012/09/12

Every so often, I see someone posting advice on Twitter or a blog about how to avoid running expensive consistency checks on production SQL Server instances. This is a noble goal, as `DBCC CHECKDB` is a very resource-intensive process and can cause performance problems when run concurrently with a workload. There is a problem, however, if the posted advice is to stop running consistency checks altogether, and instead rely on the `WITH CHECKSUM` option when performing a full database backup. Don't get me wrong, this option is great when used for the right reasons, but it's not by any means a replacement for consistency checking.

How Page Checksums Work

SQL Server offers some *page protection* options, although a better name would be *page corruption detection*, because setting a page protection option does not protect data file pages from being corrupted. What it does is allow SQL Server to detect when a data file page is corrupt.

The default page protection option is `CHECKSUM`, which enables page checksums, and all new databases created on SQL Server 2005 onwards will have page checksums enabled (unless someone has changed this option in the `model` database). This means that when a data file page is written to disk (usually by a periodic `CHECKPOINT` operation), the last thing that happens is that a simple checksum is calculated for the 8KB page contents and written into the page's header structure.

When, subsequently, SQL Server reads a data file page into memory from disk, the first thing that happens is that it recalculates the page checksum and checks it against the value stored in the page header. If the checksums do not match, SQL Server knows that something in the I/O subsystem (essentially all the software and hardware between SQL Server and the disks) corrupted the page, and reports the problem by raising the 824 error.

Therefore, SQL Server checks the page checksums whenever it reads into memory a data file page, whether because of a query, index rebuild operation, consistency check, or any other operation that SQL Server performs that requires it to read the data pages.

SQL Server will also verify the page checksums during backup operations, but only if we specify the `WITH CHECKSUM` option on the `BACKUP` statement. If the backup operation finds an invalid page checksum, by default the backup will fail and report the corrupt data file page.

So why isn't using `WITH CHECKSUM` with backups a proper substitute for consistency checking using `DBCC CHECKDB`, as surely both operations will test page checksums?

Memory Corruption and Page Checksums

Page checksums help detect page corruption within the I/O subsystem; if when a page enters memory its checksum value does not match what it was when SQL Server wrote it to disk, then it knows that page corruption occurred within the I/O subsystem.

What happens, however, if there is *in-memory* corruption of data file pages? Consider the following scenario: SQL Server reads a data file page into memory and then modifies it, because an update statement changes a column value of a table row that is stored on the data file page. Unfortunately, after the page is modified, but before the next checkpoint operation, a faulty memory chip in the physical server causes a corruption in the 8KB block of memory that is holding the modified data file page. When the checkpoint occurs, it calculates a page checksum over the data file page contents, including the portion corrupted by the faulty memory chip, writes the page checksum into the page header, and the page is written out to disk.

Later, a query causes SQL Server to read this data file page from disk and so it validates the page checksum and, assuming nothing went wrong at the I/O-subsystem level, the checksum values will match, and it will not detect any problem. The page *is* corrupt, but the page checksum algorithm cannot detect it, as the page contents have not changed since SQL Server first wrote the page to disk.

Similarly, a backup operation that validates page checksums will not detect in-memory corruption; the backup operation simply reads the data pages, calculates the checksum, compares the value to what it was when the page was last written to disk, finds the values match, and writes the in-memory corrupted page into the backup file.

Page checksums, by design, only detect corruption that occurred in the I/O subsystem, and will not catch in-memory page corruptions. As such, we cannot rely on a valid page checksum to show, categorically, that a data file page is free of corruption. That is the purpose of `DBCC CHECKDB`.

Running regular Consistency checks with DBCC CHECKDB

`DBCC CHECKDB` *interprets* the page contents (for example, validating row structures, validating column values, checking linkages in indexes), and so should always be able to detect a corrupt page, even if the corruption happened while in memory.

This means that, in addition to using `WITH CHECKSUM` when performing backups, we need to run regular consistency checks, but not necessarily on the production server. A good alternative is to restore backups on another server and then run consistency checks on them, or to use a virtual restore tool and then run consistency checks on the virtually mounted database. I discussed these alternatives in an article on the importance of validating backups (see here for more details).

Other Features for detecting in-memory Corruption

In addition to `DBCC CHECKDB`, SQL Server actually has two other features that can aid with finding memory corruptions, but which only apply if SQL Server has not yet changed the data file page in memory.

Since SQL Server 2005, the buffer pool has conducted page checksum checks for data file pages that are already in memory. If a page resides in memory, it has a page checksum, and it has not changed since SQL Server read it from disk, then the buffer pool will revalidate the page checksum, occasionally, to make sure that no in-memory corruption has occurred. If a corruption is found, an 832 error is reported – which I blogged about on my personal blog here. This can help to give an early warning of faulty memory chips, buggy Windows drivers, or other causes of memory corruption.

SQL Server 2012 running on Windows Server 2012 has a new feature that will detect and fix memory corruptions by re-reading data file pages from disk. Again, this only applies to a page that remains unchanged since SQL Server read it in from disk. This can help prevent SQL Server writing corrupted pages to disk and Glenn Berry blogged about the feature here.

Summary

To summarize, page checksums are fallible and should not be trusted to find all possible corruptions that can befall a data file page. DBCC consistency checks should always be run regularly, using whatever method allows you to run them that does not impact, or at least minimizes the impact on, your production workload.