3 Ways to Run DBCC CHECKDB Faster

August 27, 2020

[Brent Ozar](https://www.brentozar.com/archive/author/brento/)

[CHECKDB and Corruption](https://www.brentozar.com/archive/category/backup-and-recovery/checkdb-and-corruption/)

[8 Comments](https://www.brentozar.com/archive/2020/08/3-ways-to-run-dbcc-checkdb-faster/#comments)

FROM: <https://www.brentozar.com/archive/2020/08/3-ways-to-run-dbcc-checkdb-faster/>

In my last post, I talked about [how you can get 43%-67% faster backups](https://www.brentozar.com/archive/2020/08/back-up-sql-server-43-67-faster-by-writing-to-multiple-files/). Today, I’m using that same SQL Server setup to discuss how throwing more CPU cores might help you check for corruption faster if you’ve already got modern fast storage. I don’t expect everyone to go to this level of detail by any means, but it’s the kind of tuning you can do when you’re facing multiple terabytes of production data per server and shrinking maintenance windows.

You need to check for corruption regularly because SQL Server isn’t doing it for you (although [I’ve asked Microsoft to do that, and you can vote for it](https://feedback.azure.com/forums/908035-sql-server/suggestions/34895971-run-dbcc-checkdb-with-physical-only-automatically).) All kinds of things can cause corruption: storage issues, storage networking issues, and even [SQL Server bugs like this](https://support.microsoft.com/en-us/help/4526315/fix-database-cannot-recover-and-reports-error-5243-in-sql-server) or [this](https://support.microsoft.com/en-us/help/4489150/fix-filtered-index-may-be-corrupted-when-you-rebuild-index-in-parallel) or [this](https://support.microsoft.com/en-us/help/4101502/tde-enabled-database-backup-with-compression-causes-corruption) or [this](https://support.microsoft.com/en-us/help/2969896/fix-data-corruption-occurs-in-clustered-index-when-you-run-online-inde) or [this](https://support.microsoft.com/en-us/help/4019701/fix-database-schema-is-corrupted-when-you-restore-a-database-from-a-sn).

However, folks are often hesitant to run DBCC CHECKDB because it has a reputation for being resource-intensive. Good news, though: there are a few easy ways to influence CHECKDB’s impact.

Performance tweak #1:  
run it across more cores.

In Standard Edition, CHECKDB is single-threaded. That’s because you don’t need fast corruption checking in Standard Edition because it doesn’t have [any](https://support.microsoft.com/en-us/help/4490138/fix-filtered-nci-over-a-cci-may-not-be-maintained-when-the-table-is-up) [corruption](https://support.microsoft.com/en-us/help/4531238/fix-heap-corruption-occurs-when-adr-is-disabled-and-aborted-transactio) [bugs](https://support.microsoft.com/en-us/help/4531238/fix-heap-corruption-occurs-when-adr-is-disabled-and-aborted-transactio) [at](https://support.microsoft.com/en-us/help/4531238/fix-heap-corruption-occurs-when-adr-is-disabled-and-aborted-transactio) [all](https://support.microsoft.com/en-us/help/4466793/assertion-error-occurs-when-you-restart-the-sql-server-2016-database). [Nope](https://support.microsoft.com/en-us/help/2878968/fix-nonclustered-index-corruption-may-occur-when-you-run-a-complex-upd), [none](https://support.microsoft.com/en-us/help/3006648/fix-data-corruption-in-a-target-table-when-you-use-a-sql-server-destin) [whatsoever](https://support.microsoft.com/en-ca/help/3213240/fix-changing-the-data-type-and-then-updating-the-table-with-more-than). Those of you on Standard Edition can just ignore this tweak and move right on to the next one.

Enterprise Edition has parallel corruption checking because it’s so much more vulnerable to corruption bugs (is that right? that must be what it is – surely it can’t be a money grab) and since SQL Server 2016, CHECKDB has accepted a MAXDOP parameter that lets you set the number of CPU cores involved. You can also hint it upwards, higher than the server-level MAXDOP, so that it can use more threads even on servers hosting single-threaded apps.

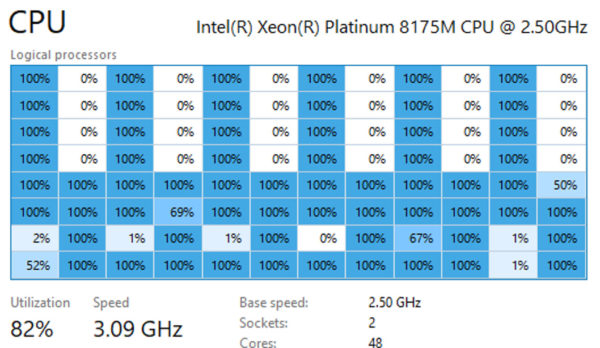
The more you use, the more you save, but the story here is a little trickier than the multi-file backup post. Here, I not only examined CHECKDB’s runtime, but also the amount of wait time generated by the command, and the wait time ratio. Wait time ratio means for every minute on the clock, how many minutes did SQL Server spend waiting? The higher that number goes, the uglier your performance is.

|  |  |  |  |
| --- | --- | --- | --- |
| **MAXDOP** | **Minutes** | **Wait Time, Minutes** | **Wait Time Ratio** |
| 1 | 53 | 4 | 0 |
| 2 | 37 | 121 | 3 |
| 4 | 26 | 155 | 6 |
| 8 | 18 | 212 | 12 |
| 12 | 14 | 234 | 16 |
| 16 | 14 | 332 | 23 |
| 24 | 12 | 354 | 30 |
| 32 | 11 | 462 | 41 |
| 48 | 9 | 494 | 56 |
| 56 | 8 | 564 | 65 |
| **64** | **7** | **579** | **75** |
| 72 | 13 | 1,052 | 79 |
| 80 | 17 | 1,409 | 84 |
| 96 | 19 | 1,879 | 99 |

Keep in mind that this particular SQL Server has 48 physical cores, 96 with hyperthreading turned on.

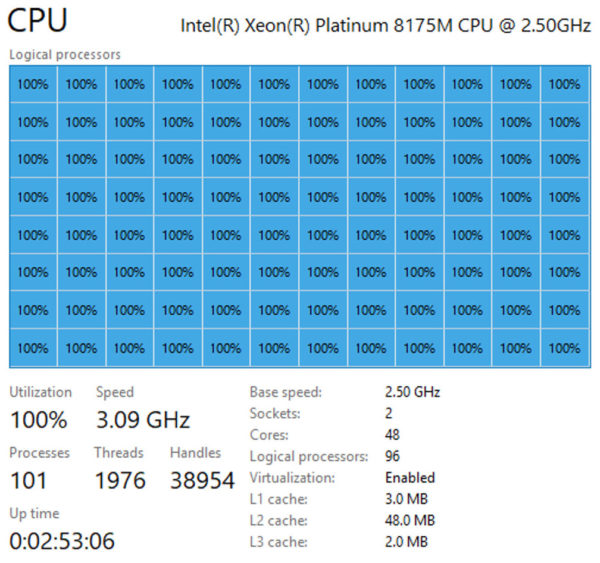
At MAXDOP 64, we’re checking 56GB of data per minute, and:

* CHECKDB runs 86% faster than when it’s single-threaded
* 59% faster than MAXDOP 8 (which is what a lot of folks would be at if they set server-level MAXDOP here)
* 48% faster than MAXDOP 16
* Wait time ratio is 75, meaning for every minute on the clock, we’re generating 75 minutes of wait time, most of which is CXPACKET and CXCONSUMER
* The 64 CPU cores involved stick at 100% for most of the duration (thanks to our awesome hardware provisioning)

[](https://www.brentozar.com/wp-content/uploads/2020/08/dbcc_checkdb_maxdop_64.jpg)

I’m certainly not saying MAXDOP 64 makes the most sense for all hardware, but in this scenario, it would let us keep the maintenance window as short as possible, assuming we have 7-10 minutes each night where we can sustain heavy loads.

MAXDOP 96 runs slower, not faster, and the CPU usage makes the box feel pretty unusable:

[](https://www.brentozar.com/wp-content/uploads/2020/09/checkdb_100_percent_cpu-1.jpg)

Performance tweak #2:  
check for less (or more) corruption problems.

By default, CHECKDB checks for the most common issues, but you can make it go way faster if you just ask it to check the checksums on each page. This doesn’t catch logical issues, like [corrupt statistics](https://support.microsoft.com/en-us/help/4530907/improvement-corrupt-statistics-can-be-detected-by-using-extended-logic), but it at least gets you the earliest warning signs of storage corruption.

To do this, use the WITH PHYSICAL\_ONLY switch – which doesn’t do nearly as much CPU work, so we also see different performance results here:

|  |  |  |  |
| --- | --- | --- | --- |
| **MAXDOP** | **Minutes** | **Wait Time, Minutes** | **Wait Time Ratio** |
| 1 | 17 | 1 | 0 |
| 2 | 12 | 35 | 3 |
| 4 | 8 | 41 | 5 |
| 8 | 6 | 47 | 8 |
| 12 | 5 | 61 | 12 |
| 16 | 5 | 74 | 15 |
| 24 | 5 | 113 | 24 |

For this particular server, as I threw more cores at it, the only things that went up were my parallelism waits. The sweet spot here was around 8-12 cores.

But this leads to an interesting comparison:

* PHYSICAL\_ONLY with 12 cores: takes 5 minutes, only checks page checksums
* FULL CHECKDB with 64 cores: takes 7 minutes, checks everything

On this particular server, as long as I’m running Expensive Edition, I just don’t think it would make sense to use the PHYSICAL\_ONLY setting because as long as I’m taking a 7-minute slowdown, if I’ve got the CPU cores available (and they’re not working otherwise), then might as well just check everything. If this was truly a 24/7 environment where I couldn’t handle a 7-minute slowdown, then…I’m probably offloading CHECKDB to another AG replica anyway, especially given the [wonderful 2019 licensing changes that make it free](https://cloudblogs.microsoft.com/sqlserver/2019/10/30/new-high-availability-and-disaster-recovery-benefits-for-sql-server/).

Furthermore, I might even want to use the EXTENDED\_LOGICAL\_CHECKS parameter. It catches even more corruption culprits, albeit at the cost of higher runtimes:

|  |  |  |  |
| --- | --- | --- | --- |
| **MAXDOP** | **Minutes** | **Wait Time, Minutes** | **Wait Time Ratio** |
| 1 | 53 | 4 | 0 |
| 2 | 40 | 130 | 3 |
| 4 | 29 | 175 | 6 |
| 8 | 17 | 198 | 11 |
| 12 | 16 | 292 | 17 |
| 16 | 14 | 321 | 22 |
| 24 | 10 | 327 | 31 |
| 48 | 9 | 492 | 57 |
| 56 | 11 | 731 | 60 |
| 64 | 15 | 1,047 | 69 |
| 96 | 19 | 1,880 | 99 |

It’s interesting to note that the CPU sweet spot for this particular box was 64 cores for the regular CHECKDB, but around 24 cores for EXTENDED\_LOGICAL\_CHECKS, producing the nicest balance between short runtimes and an overloaded, unresponsive server. (I’m abbreviating some of the results here.) That sweet spot is going to depend on not just your hardware, but also the contents of your database and which SQL Server features you’re using, which brings me to…

Performance tweak #3:  
make your database smaller.

The number of tables you have AND the number of indexes on ’em both affect CHECKDB’s speed. All of the tests above involved the [390GB 2020-06 Stack Overflow database](https://www.brentozar.com/archive/2015/10/how-to-download-the-stack-overflow-database-via-bittorrent/), which ships with no nonclustered indexes whatsoever.

To make the database more complex, I added:

* 80 3-column indexes
* 2 indexed views
* 2 nonclustered columnstore indexes
* Bringing the database from 390GB to 560GB (up about 44%)

And then ran a few of the tests again:

* **Physical\_only, 16 cores: 7.2 minutes – up ~60% from 4.5 minutes**, bigger than the 44% database size growth, but I wouldn’t read too much into one-off tests – I wouldn’t use these results to say definitively that physical\_only performance doesn’t scale linearly with database size.
* **Regular CHECKDB, 64 cores: 30.4 minutes – up dramatically from 7.5 minutes** with no indexes, and wait time ratio remained around 75, so the server was really hustling the whole time.
* **Extended logical checks, 64 cores: 35.3 minutes – up from 14.9 minutes**, but the extended checks runtime penalty (compared to regular CHECKDB at 30.4 minutes) wasn’t as big here as it was for the test with no nonclustered indexes at all. This lines up with what I see out in the field: extended logical checks aren’t that big of a deal on real-life databases with indexes, and they’re totally worth it if you’re taking the gamble on indexed views.

These timings are why it often makes sense to run CHECKDB with the PHYSICAL\_ONLY switch during the week in order to get that blazing fast runtime with hardly any CPU usage, and then upgrade to the resource-intensive full CHECKDB with EXTENDED\_LOGICAL\_CHECKS on the weekend, throwing lots of CPU cores at it to get the work over as quickly as possible.

Didn’t know about this? You might have missed out on some of the other things we talk about in [our Fundamentals of Database Administration class.](https://www.brentozar.com/product/fundamentals-of-database-administration/)

[Previous Post](https://www.brentozar.com/archive/2020/08/back-up-sql-server-43-67-faster-by-writing-to-multiple-files/" \o "Back Up SQL Server 43%-67% Faster by Writing to Multiple Files.)

[Back Up SQL Server 43%-67% Faster by Writing to Multiple Files.](https://www.brentozar.com/archive/2020/08/back-up-sql-server-43-67-faster-by-writing-to-multiple-files/" \o "Back Up SQL Server 43%-67% Faster by Writing to Multiple Files.)

[Next Post](https://www.brentozar.com/archive/2020/08/watch-brent-tune-queries-at-sqlsaturday-oslo/" \o "Watch Brent Tune Queries at SQLSaturday Oslo)

[Watch Brent Tune Queries at SQLSaturday Oslo](https://www.brentozar.com/archive/2020/08/watch-brent-tune-queries-at-sqlsaturday-oslo/" \o "Watch Brent Tune Queries at SQLSaturday Oslo)

8 Comments. [Leave new](https://www.brentozar.com/archive/2020/08/3-ways-to-run-dbcc-checkdb-faster/#respond)

* https://secure.gravatar.com/avatar/3e528b600742499948aecb1f1f2bf99c?s=50&d=wavatar&r=r

**Garry Bargsley**

[August 27, 2020 8:35 am](https://www.brentozar.com/archive/2020/08/3-ways-to-run-dbcc-checkdb-faster/#comment-3239464)

This topic is well timed. Based on one of your demos last week it caused the light bulb to go off in my mind. Why am I not pumping up resources when I run my CheckDB process on my admin server then dropping them down when done. It takes 2 days to test restores and run CheckDB on all my production databases. So my plan is to build a new azure server with 96 cores and 128MB. And I will start the server as part of my workflow, run all my restores and CheckDB and even try to do some in parallel, then power off and deallocate my server.

However, when I asked what type of specs you should look for in doing this on a Twitter post I got an interesting response from Mr. Randal. His order of priority is the IO speed, TempDB drive performance CPU and then Memory.

I am still working on approvals from management on this concept. Then I will take your info and Mr. Randal’s and do some testing and see where the sweet spot configuration is for this setup.

Thanks as always for the thought provoking posts.

[Reply](https://www.brentozar.com/archive/2020/08/3-ways-to-run-dbcc-checkdb-faster/#comment-3239464)

[[https://secure.gravatar.com/avatar/77f776c2eaf0cc691e8a0880bb8a191f?s=50&d=wavatar&r=r](https://www.brentozar.com/)](https://www.brentozar.com/" \t "_blank)

**[Brent Ozar](https://www.brentozar.com/" \t "_blank)**

[August 27, 2020 8:38 am](https://www.brentozar.com/archive/2020/08/3-ways-to-run-dbcc-checkdb-faster/#comment-3239465)

Yeah, Paul’s right – you want IO throughput first since it involves reading the entire database as quickly as possible. Thankfully in AWS, the i3 series comes with plenty of free insanely fast NVMe SSD space, so it’s great for these kinds of operations.

[Reply](https://www.brentozar.com/archive/2020/08/3-ways-to-run-dbcc-checkdb-faster/#comment-3239465)

* https://secure.gravatar.com/avatar/9bce36ef31f8fc4bf14348c9728f5a62?s=50&d=wavatar&r=r

**TechnoCaveman**

[August 27, 2020 9:32 am](https://www.brentozar.com/archive/2020/08/3-ways-to-run-dbcc-checkdb-faster/#comment-3239468)

2 sockets, 48 cores. What would 4 socket – 48 core (4×12 cores) with similar L2 and L3 on die cache do?  
(yes they do make 4 socket DDR4 mother boards for under $1200 [unpopulated]  
1) Win lottery  
2) Get 4 socket motherboard to test.  
3) Populate mother board with dual 1KW power supplies  
Test to see if L2 and L3 cache should be over run with so many reads. Yes MaxDOP maxed out at 64 – but inside SQL engine there are many other processes wanting a quantum time slice.  
Guessing test #1 woule max out at 80 cores Why? lower core to L2 cache ratio resulting in more hits and fewer pipeline flushes.  
4) Tell spouse and friends I won lottery – They would never understand buying \*another\* computer.

[Reply](https://www.brentozar.com/archive/2020/08/3-ways-to-run-dbcc-checkdb-faster/#comment-3239468)

* https://secure.gravatar.com/avatar/7315ab21067958756b53a84e7a41a68c?s=50&d=wavatar&r=r

**Steve**

[August 27, 2020 10:48 am](https://www.brentozar.com/archive/2020/08/3-ways-to-run-dbcc-checkdb-faster/#comment-3239490)

So, if I had a need to improve performance of checkDB on a non-prod instance, is it suitable to drop all non-clustered indexes first and then run checkDB?

Then run a full checkDB periodically ?

If any corruptions are then identified in the non-clustered indexes, then they can just be rebuilt.

[Reply](https://www.brentozar.com/archive/2020/08/3-ways-to-run-dbcc-checkdb-faster/#comment-3239490)

[[https://secure.gravatar.com/avatar/77f776c2eaf0cc691e8a0880bb8a191f?s=50&d=wavatar&r=r](https://www.brentozar.com/)](https://www.brentozar.com/" \t "_blank)

**[Brent Ozar](https://www.brentozar.com/" \t "_blank)**

[August 27, 2020 10:53 am](https://www.brentozar.com/archive/2020/08/3-ways-to-run-dbcc-checkdb-faster/#comment-3239492)

Steve – so if storage corruption just happens to strike a nonclustered index…

Do you think it’s always only going to hit them?

Or do you maybe wanna find out as soon as possible and fail outta there before it hits a clustered index and really ruins your day?

[Reply](https://www.brentozar.com/archive/2020/08/3-ways-to-run-dbcc-checkdb-faster/#comment-3239492)

* + - https://secure.gravatar.com/avatar/7315ab21067958756b53a84e7a41a68c?s=50&d=wavatar&r=r

**Steve**

[August 27, 2020 11:09 am](https://www.brentozar.com/archive/2020/08/3-ways-to-run-dbcc-checkdb-faster/#comment-3239502)

That’s a pretty fair point.

I knew it smelt like a bad idea, but wasn’t sure why

[Reply](https://www.brentozar.com/archive/2020/08/3-ways-to-run-dbcc-checkdb-faster/#comment-3239502)

* https://secure.gravatar.com/avatar/9d6f955da44ba7815e500b56ffc8491a?s=50&d=wavatar&r=r

**Klaus Ladegaard**

[August 28, 2020 1:24 am](https://www.brentozar.com/archive/2020/08/3-ways-to-run-dbcc-checkdb-faster/#comment-3239689)

Hi Brent

Think you forgot a point  
CHECKDB makes a internal snap of DB per table, meaning having to walk buffer each table n times

having huge servers with a lot memory and db on 15+ TB with 1000+ tables, this takes time  
In this case doing a snap ourself saves a lot of time

Any recommendation in this area.

[Reply](https://www.brentozar.com/archive/2020/08/3-ways-to-run-dbcc-checkdb-faster/#comment-3239689)

[[https://secure.gravatar.com/avatar/77f776c2eaf0cc691e8a0880bb8a191f?s=50&d=wavatar&r=r](https://www.brentozar.com/)](https://www.brentozar.com/" \t "_blank)

**[Brent Ozar](https://www.brentozar.com/" \t "_blank)**

[August 28, 2020 3:21 am](https://www.brentozar.com/archive/2020/08/3-ways-to-run-dbcc-checkdb-faster/#comment-3239703)

Klaus – yep, that too! I didn’t forget that, but just only so many things I can do per blog post. By all means, sounds like you’re qualified to write posts too – start blogging! It’s a great way to share what you know and advance your career.