Performance Regression

Last Modified on 11/05/2018 11:54 am CST

```
Source for this KB:
https://littlekendra.com/2016/07/21/why-can-an-upgrade-cause-performance-regression-dear-sql-dba-episode-
```

Most performance problems in TempWorks can be attributed to SQL Server performance. The goal of this article is to provide guidance on how to troubleshoot performance degradation after a SQL Server version or hardware upgrade.

> Note: It's also a good idea to perform a lot of the diagnostics discussed here once a year to make sure your current system is performing optimally.

Review your Max Degree of Parallelism and Cost Threshold for Parallelism settings (and other things from sys.configurations)

You've already done this one. I'm listing it for others who might hit the same situation, because it's an important step.

SQL Server defaults "max degree of parallelism" to 0, which means when a query goes parallel, it can use all the cores. If you accidentally left that set at 0 on the new instance, your workload could have queries fighting over all CPUs all the time.

For more information on how to determine these two settings, check out my last episde, "Max Degree of Confusion ."

While you're at it, compare all the settings in sys.configurations between the instances to check for differences, and look into each one. I've seen weird accidents cause performance issues, like people mistakenly setting minimum memory for a query on one instance.

The new hardware may have a bottleneck that only shows up under load

I've run into a couple of issues with hardware configuration that weren't caught until after migration.

In one case, the server power wasn't completely plugged in. This led to CPUs that operated at lower power, and actually showed *higher* percent utilization when examined in Windows task manager. They were just capable of doing less!

You can detect this by doing two things:

- Check the Windows System log for problematic messages — in this case there were messages about power sources not being fully plugged in, but I look at all messages in the system log (some things are "informational" that are really important)
- Run a free tool like CPU-Z to measure the clock speed that your processors are running at

In another case, following a migration we had one single core that was always at 100%! It turns out that our new

NIC was pegging one core under load.

- We saw SOS_SCHEDULER_YIELD in wait stats – when SQL Server was trying to use that one core, it had to wait a LOT, because it was being hogged by the NIC
- This skewed overall utilization up
- We were able to fix this by tweaking settings on the NIC so that it used multiple cores – required planned maintenance
- This hadn't happened on the old server, and our load testing pre migration just wasn't chatty enough over the network to surface the issue

You may have power savings enabled on your processors

I wouldn't expect this to make the entire difference between 25% and 75% usage, but it could be one of a few things contributing to the problem. It's easy to fix, so it's worth checking for.

Look at the BIOS on the new servers and make sure that power savings has been disabled.

Just changing the power savings settings in Windows Server does not always disable power savings, you need to look at a server management tool like Dell OpenManage or HP System Insight Manager.

Use your server model and look up the manufacturer's guide to configuration for performance. These guides will list the power settings needed to get "high performance."

Most servers these days ship with default settings that clock the processors down, and they can be very sluggish to clock up.

This typically takes a planned outage to fix: you need to make a change in the BIOS and then restart the server.

Your fresh new SQL Server version could be suffering from stack dumps or "15 second" storage errors

Stack dumps typically causes periodic "freezeups" that are followed by high load. 15 second storage errors mean the storage literally isn't responding for 15 seconds.

Look in the SQL Server error log. Look pretty closely at everything for a few days for errors, then filter for the terms:

- "Stack Dump"
- "15 second"

For more on 15 second errors, check out my previous episode, "Outside the big SAN Box: Identifying Storage Latency in SQL Server ."

Licensing may not be allowing you to use all the CPUs / memory banks

If SQL Server can't use all your CPUs, it can cause weird high CPU usage on some of them– and in some cases not allow SQL Server to use all the memory on the server, which can cause unexpected slow performance patterns.

I've seen this happen in a couple of different ways:

- SQL Server 2012 has a weird Enterprise Install limited to 20 cores
    - There was an upgrade path for people using "seat based" or CAL licensing to use SQL Server 2012 Enterprise Edition, but the installation limited them to using 20 cores or less.
    - Some people didn't understand the upgrade licensing completely, and just didn't realize they hadn't licensed all the cores. In these cases CPU affinity masking can at least be used to spread the cores in use evenly across all server sockets (be very careful, affinity masking can go wrong)
    - Once I found a case where someone had simply downloaded the incorrect install media from MSDN and it was used on all the instances at their company by accident– and these servers had a lot more than 20 cores. The name of the download appeared very innocent
    - Aaron Bertrand wrote a post with lots of detail on this on SQLBlog
- I've also seen this happen when people are running SQL Server Standard Edition with a VM, and they configure the VM to look like it has more than 4 CPU sockets with a single core by accident

Look in the SQL Server Error log after the last startup for the message telling you how many sockets and cores SQL Server detected and make sure it's using all the cores it should be.

Something outside SQL Server may be using CPU

You've probably checked for this.

It never hurts to doublecheck, because I've found surprises on servers that I manage. Stuff just creeps in. Someone decides to use the server for testing before you go live and forgets to tell you, and also forgets to turns stuff off.

Or if you're me, you did something in the middle of the night before the instance went live that you forgot about.

Check the Windows task manager for processes using CPU and memory, and make sure there's not scheduled tasks or agent jobs that are running things that didn't used to run on the old server.

You could be using a different version of the SQL Server Cardinality Estimator on the new instance

This is specifically for folks who have upgraded to SQL Server 2014 or higher.

Check your database compatibility level and related settings.

- Database compatibility level 120 uses a new version of the "cardinality estimator"
- This can lead to very different plans being generated
- On SQL Server 2016, you can raise the database compatibility level to 120 or 130 and set LEGACY_CARDINALITY_ESTIMATION=ON for the database to use the old level

The new cardinality estimator is a good thing, but some folks have found it causing performance regressions, so it's worth checking if this is part of your issue.

SQL Server may be optimizing the top CPU burning queries differently

Instead of moving the files and attaching, I prefer backup and restore (or log shipping for a fast cutover).

- I like to keep the old copy of the database on the old hardware for at least a month
- This means if a "slow query" issue comes up, I can run it against the old hardware and compare execution times, to see if it's really slower or burns more resources
- I can also compare query execution plans to see if SQL Server is making a different decision about how to run the query

You should be taking a final backup pre-migration no matter what, so you can restore that last backup and compare, even if you moved the database files to do the migration.

In any case, pull the top 10 queries by total CPU usage. Look for:

- large amounts of reads / indexing opportunities
- huge amounts of executions per minute
- plan warnings of any sort (these show up with a little yellow warning sign, like for roadwork)

I like to pull this list prior to migration as well, so I can tell if the top queries afterward are the same or different.

Free tools to pull top queries:

- sp_BlitzCache from Brent Ozar Unlimited (registration required)
- Glenn Berry's SQL Server DMV queries from SQLSkills

A new bottleneck may have emerged simply because the new hardware is faster

Even if you're matching things like the number of CPUs, the CPUs themselves are hopefully faster. Because hardware is just faster.

And since memory is cheaper, we often have a bit more memory in the new SQL Server.

When you make one thing faster, sometimes that can lead to a new bottleneck. I've hit this situation:

- Migrate to a server with faster CPUs and more memory
- Amount of physical reads goes down, because SQL Server can cache more, and queries are faster
- Suddenly, blocking problems get worse because query patterns changed

For migrations, I like to capture wait statistics samples for a couple of weeks prior to the migration. That way, after the migration I can compare waits and see if anything new pops up, and track it from there.

Two free tools to sample waits:

- Paul Randal's wait stats script (SQLSkills)
- sp_BlitzFirst from Brent Ozar Unlimited (registration required)

You could just be hitting a weird wait on the new version

Good news, the wait stats check in the previous section will find this, too!

Sometimes you just get unlucky, and your code hits a weird issue on a new version of SQL Server that you didn't have on the old one. In once case I had very high CMEMTHREAD waits under load on one specific version of SQL Server.

This is a pretty rare wait type and I had to do a bunch of research on it, and we eventually found a fix for our version.

I always look for weird wait types, and want to compare waits pre and post migration when possible.

You could be getting more load

Sometimes people migrate right before the busy season hits. They know some load is coming, so they buy new hardware.

And sometimes the person doing the DBA work doesn't get to talk to the people who know when it's a high or low period for load.

You might be able to see this looking at top queries by execution, but another simple way to check it is by collecting a few performance counters before and after migration

The "SQL Statistics" Performance Object has three counters that can help measure how much work your instance is doing:

- Batch requests/sec
- Compilations/sec
- Recompilations/sec

Compilations particularly burn CPU, so whenever CPU goes up I'm always interested to see if compilation rates have risen.

Migration planning and troubleshooting checklists

Some planning notes (to make troubleshooting easier)

- Keep the original server and database around as part of your plan. Do the migration with backup /restore (or possibly logshipping or mirroring to reduce downtime).
    - Lock accounts out of the old database and even shut of the instance if needed
    - Keep it around for a month in case it's needed to troubleshoot performance / compare execution plans
- Collect samples of wait stats during important periods for several weeks prior to migration, save them off
- Collect top sql server queries with plans by CPU, logical reads, and execution count prior to migration
- Collect performance counters prior to migration:
    - SQL Statistics – Batch requests/sec, Compilations/sec, Recompilations/sec
    - % Processor Time

Troubleshooting post migration

- Review your Max Degree of Parallelism and Cost Threshold for Parallelism settings
- Compare all the settings in sys.configurations between the instances to check for differences, and look into each one
- Check the Windows System log for problematic messages — in this case there were messages about power sources not being fully plugged in, but I look at all messages in the system log (some things are "informational" that are really important)
- Run a free tool like CPU-Z to measure the clock speed that your processors are actually getting
- Look at the BIOS on the new servers and make sure that power savings has been disabled
- Look in the SQL Server Error log after the last startup for the message telling you how many sockets and cores SQL Server detected and make sure it's using all the cores it should be
- Check task manager for processes using CPU and memory, and make sure there's not scheduled tasks or agent jobs that are running things that didn't used to run on the old server.
- Review SQL Server Error log for stack dumps, 15 second IO latency warnings, or other errors
- Check the Windows task manager for processes using CPU and memory, and make sure there's not scheduled tasks or agent jobs that are running things that didn't used to run on the old server
- Check your database compatibility level and related settings to see if you're using the new cardinality estimator (and weren't before) – SQL Server 2014+
- Identify top 10 CPU burning queries. Compare query plans against older instance. Look for large amounts of reads / indexing opportunities, or huge amounts of executions per minute.
- Sample SQL Server wait stats when performance is poor to see what the top waits are — and if they're surprising

Related Articles