**­­**

**Discovery and Assessment**

# Performance Findings and Recommendations

After analysis of the environment, we have found several issues which impact performance. Below you will find a description of the findings as well as the steps to correct each.

* **Sever Sizing:** The Server currently has 64GB of memory along with 8 CPU, 50GB of memory is allocated to SQL Server. Ideally in a SQL Server environment you want to have more memory and lower CPU, the more pages you can have left in memory the better performance would be because it will limit the amount it needs to go to disk and pull the pages into the buffer pool. Microsoft recommends that a server with 8 or less CPU is to keep the MAXDOP at or below the number of processors.
  + **Recommendation**: Based on the wait types that I have pulled and how SQL seems to be responding, I would say you could benefit from adding additional memory; the PLE is a lot lower than the recommended minimum MAXDOP is currently at 8.
  + **This will require restarting sql after changing a few config settings. Total time <30 mins.**
* **DBCC CHECKDB**: Database Console Command CHECKDB (DBCC CHECKDB) is used to check the integrity (physical & logical) of objects in a SQL Server database. The command is supported in databases that contain memory-optimized tables but the validation is only supported in disk-based tables. The DBCC repair option is unavailable in memory-optimized tables and therefore, leads to the need of regular database backup. In case an issue arises in a memory-optimized table, the data can be restored from the last backup done.
  + **Recommendation**: Based on my findings Database integrity checks are either not happening at all, or are happening greater than two weeks apart, I would suggest having a Database Integrity check at least weekly.
  + This will take roughly 2-3 hours to complete, with another 2-3 total hours in reassessments.
* **TempDB**: The TempDB database is one of the most important SQL Server system databases, that is used to store temporary user objects, such as the temporary tables that are defined by the user or returned from table-valued function execution, temporary stored procedures, table variables or indexes. In addition to the user objects, TempDB will be used to store internal objects that are created by the SQL Server Database Engine during the different internal operations, such as intermediate sorting, spooling, aggregate or cursor operations. The TempDB system database is used also to store the rows versions to support the features that require tracking the changes that are performed on the table rows, such as the snapshot isolation level of the transactions, Online Index rebuilds or the Multiple Active Result Sets feature. Tempdb is currently entirely on the C drive.
  + **Recommendation**: Move Tempdb to its own dedicated drive, as SQL Server will use Tempdb as much as it needs and the files will grow larger than the that is left on the C drive, performance is also a big factor in that database files should not be on the C drive as this should be reserved for the OS only.
  + This will require moving tempdb to its own dedicated drive, which will require an out of 1 hour.
* **Instant File Initialization**: Instant file initialization (IFI) allows SQL Server to skip the zero-writing step and begin using the allocated space immediately for data files. It doesn’t impact growths of your transaction log files; those still need all the zeroes. The larger the growth operation, the more noticeable the performance improvement is with IFI enabled. For instance, a data file growing by 20 GB can take minutes to initialize without IFI. Read more about waits for file growths here. This can make a huge difference whenever you are proactively growing out data files.
  + **Recommendation**: Enable IFI; this will help improve the time and overhead it takes for SQL Server to extend a database file, a log file, Tempdb, or any other operations it needs to use the files for. Plus, this will reduce the time it takes to restore databases from backups, and in the event of an outage reducing downtime.
  + This will require changing a local secuirty policy and rebooting, down time should be <30 mins.
* **Maintenance Plans**: Microsoft recommends coming up with Maintenance Plans that run on a regular basis, these usually include, updating statistics, reorganizing indexes, or rebuilding indexes. If these are out of date it can cause major performance issues. There does not seem to be any sort of Maintenance plans or Agent Jobs in place that run these tasks. Looking into the Statistics most of them have never been updated, and most indexes are severely fragmented. The more an index is fragmented the hard SQL must work harder to make data modifications, and out of date statistics will cause the optimizer to choose old/inefficient query plans.
  + **Recommendation**: Create some sort of Daily/Weekly Maintenance, ideally you would want to have the Statistics be updated daily to make sure SQL is uses the best plan possible.
  + Either rebuild indexes or reorganize depending on fragmentation levels. Any index that is over < 30% fragmented would need to be reorganized, and any index >30% fragmented needs to be rebuilt.
  + Run Database Integrity checks to ensure there is no sort of corruption going on, these can be put into an Agent Job and issue a DBCC.
  + This is dependent on a few factors, but in general should take 5-10 hours to complete with another 2 hours of monitoring and period assessment to ensure no changes are made. No downtime.
* **Security**: Microsoft recommends keeping up on OS and SQL Server patches as often as possible, along with rotating any SA level users within both the OS and SQL Server. Microsoft also recommends having the least amount of SA users as possible. It seems there are a lot of Agent jobs that are owned by an individual user, this will be an issue if the user is removed, or if there is an issue with the login/AD; Those agent Jobs will not run.

**Recommendations**: Current version of SQL Server is 15.0.2000.5 and the most recent version is 15.0.4405.4 Patch as soon as possible would be ideal, the instance is currently 29 patches behind.

* + Complete an audit of all SA or elevated privileges within SQL Server such as db\_owner, and verify all are needed and still valid, also rotate the passwords if has not been done so recently.
  + Change both the Database owners and the Agent Job owners to a non-user account such as SA, or a dedicated account to avoid any issues with users logins being disabled, removed, etc. that would cause problems with both Database access as well as Agent Jobs being able to continue to run without issues.
  + This is a longer process that will involve testing of each app that has elevated access. No true downtime is required but would recommend a maintenance window. 1 hour weekly for a month.
* **Performance**: There are a decent number of unused indexes (Scans, seeks) this can be because they are truly no longer needed or that the indexes themselves are not being used due to the index not being correct and the query is bypassing it and doing a table scan instead. There also seems to be a good amount of High Value Indexes that are missing (Heaps) and could be limiting performance on several tables causing the optimizer to work harder to find the data the query is looking for. There are also over 10k instances of queries forcing a certain order and not letting the optimizer do it, which can hinder performance if it is not 100% correct. More than 50 plans are present for a single query in the plan cache - meaning there are probably have parameterization issues.
  + **Recommendations**: Review all indexes within all tables to verify that they are still being actively used and if not drop the unused ones, this will help the optimizer finding the data it needs. There are also some missing “High Value” indexes meaning that these will have a high impact on improving performance, having greater than 1000 table scans since the last restart of SQL Server
  + Double check all user created stats, these might have been something that was once a benefit and now has turned into a bottleneck since SQL Server will not manage those stats directly.
  + Consider using the Forced Parameterization option in Databases that have multiple plans per query, if fixing the certain queries parameters is not an option. This will reduce the number of query plans that gets created and the query would run faster with an existing plan vs a newly made plan.
  + This is getting into query performance tuning, these types of actions are not easy to estimate the timing on. For the size of the DBs and what i have seen i would say 10-30 hours. No real downtime.
* **Disk Configuration:** Microsoft recommends that all Data disks (Data .mdf, Log .ldf, Tempdb) to be on their own drives and formatted to 64k bytes, currently all of the Data files, Log files, and system DBs all share the same drive. This causes disk contention when multiple files are trying to be read/written to at the same time.
  + **Recommendations:** Spread out all of the DB files into three drives. Data Files (.mdf), Log Files (.ldf), and Tempdb should be isolated to their own drives to avoid any sort of contention among the different files.
  + Adding additional Tempdb files to match the CPU would help reduce some of the pressure and waits on Tempdb.
  + There are multiple database files that are set to 1MB auto growth, this is extremely inefficient because every Mb that it grows by SQL has to use extra resources to extend the file which can slow down Inserts and other data modifications because the space is not there and SQL has to extend the file each MB it grows by.
  + This will require adding additional disks to SQL (3), then configure the disks to meet best practices, and migrate the data to the new disks. This can be anywhere between 2-8 hours depending on speed and size. There would need to be an outage the entire time this is being done.
* **Server waits**: The biggest wait that SQL is running into is ASYNC\_IO\_COMPLETION since the last restart; there has been 192.1 hours of waits, 11.66 minutes average wait time per hour, 6.0% signal wait, 743660 waiting tasks, 12.1 ms average wait time. This wait type is directly tied to tasks waiting for I/O to complete.
  + **Recommendation**: Being that all the disks are formatted at 4K bytes per cluster vs the recommended 64K bytes per cluster for SQL Server, the recommendation would be to reformat the drives to 64K bytes per cluster to ensure optimal Read/Write capabilities. This should directly reduce the amount the this wait type and boost performance.
  + This should get resolved by moving to new storage for SQL.
* **High VLF Count**: VLF stands for Virtual Log File and the transaction log file is made up of one or more virtual log files. The number of virtual log files can grow based on the auto growth settings for the log file and how often the active transactions are written to disk.  Too many virtual log files can cause transaction log backups to slow down and can also slow down database recovery, and in extreme cases, even affect insert/update/delete performance.
  + **Recommendation**: Log files should be sized properly so they do not need to grow/extend too often. To reduce the amount of VLFs we will need to increase the size of the Log file and then shrink the log file until it is completely empty and let it grow naturally to the size it should be. Usually this cannot be done during peak hours as shrinking will lock the database.
  + This will also get resolved moving to new storage for SQL.
* **Query Store:** It is basically a SQL Server “flight recorder” or “black box”, capturing a history of executed queries, query runtime execution statistics, execution plans etc. against a specific database. This information helps in identifying performance problems caused by query plan changes and troubleshooting by quickly finding performance differences, even after SQL Server restart or upgrade. All data that SQL Server Query Store capture are stored on disk.
  + **Recommendation:** Enable Query Store, this is done at the Database level, so it can be done for all Databases or only certain ones that would be important. There is not much risk associated with enabling this feature as the performance hit is only 3-5% at the most.
  + This needs to be done for each DB and can become time consuming, no outage required. Will take 3-8 Hours to configure fully.
* **Backups:** A copy of SQL Server data that can be used to restore and recover the data after a failure. A backup of SQL Server data is created at the level of a database or one or more of its files or filegroups. Table-level backups cannot be created. In addition to data backups, the full recovery model requires creating backups of the transaction log. Currently there are FULL backups every hour with a retention of only 3 hours, meaning that if something were to happen you would only be able to recover 1 of the last three hours and lose everything prior and after that hour.
  + **Recommendation:** Need to establish a well-defined backup schedule that fits RPO/RTO requirements for the business and provides reliable backups, recoverability, and keeping the LSN chain intact. Ideally doing a daily FULL backup and Hourly Transaction Logs would be sufficient and not having any other backups running such as snapshot that would break the chain. We could also use Differential backups to reduce the number of FULL backups if space is a concern.
  + This does not require any outage, it should take 3-8 Hours of initial setup. Plus another 3 hours of maintenance and overview to ensure everything is fully compliant.
* **Items Reviewed without immediate recommendations:**
* Database Collation is not set to default
* Uneven file growth settings within Database Files.
  + The growth rate being different can cause performance issues at the file level if 1 database file is being use more than others, ideally you want all files to be written to evenly to avoid potential performance issues.
* Snapshots Occurring
  + There appears to be some sort of snapshot running hourly, this is fine if they are set to crash consistent, however if they are application consistent then it is freezing the I/O and breaking the LSN chain and in the event of an outage you would need to restore back to the backup prior to the snapshot and with a three-hour retention period; the database could be unrecoverable.