

SQLintersection

Session: Wednesday, 11:45am-1:00pm

Practical Guidance to Make Your Tier-1 SQL Server ROAR!

Pam Lahoud

Pedro Lopes



SQL
intersection



Speakers: Pam Lahoud & Pedro Lopes



@SQLGoddess



@SQLPedro



© SQLIntersection. All rights reserved.
<http://www.SQLIntersection.com>

Reminder: Intersect with Speakers and Attendees

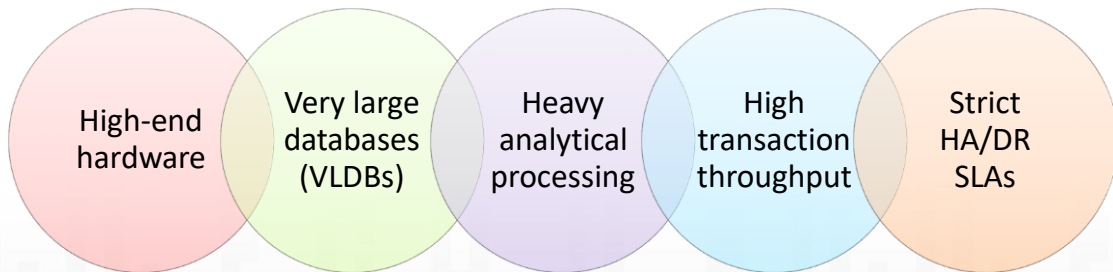
- **Tweet tips and tricks that you learn and follow tweets posted by your peers!**
 - Follow: #SQLIntersection and/or #DEVIntersection
- **Join us – Wednesday Evening – for SQLafterDark**
 - Doors open at **7:00 pm**
 - Trivia game starts at **7:30 pm**
 - Winning team receives something fun!*
 - Raffle at the end of the night
 - Lots of great items to win including a seat in a SQLskills Immersion Event!*
 - The first round of drinks is sponsored by SentryOne and SQLskills



© SQLIntersection. All rights reserved.
<http://www.SQLIntersection.com>

What's a tier-1 SQL Server?

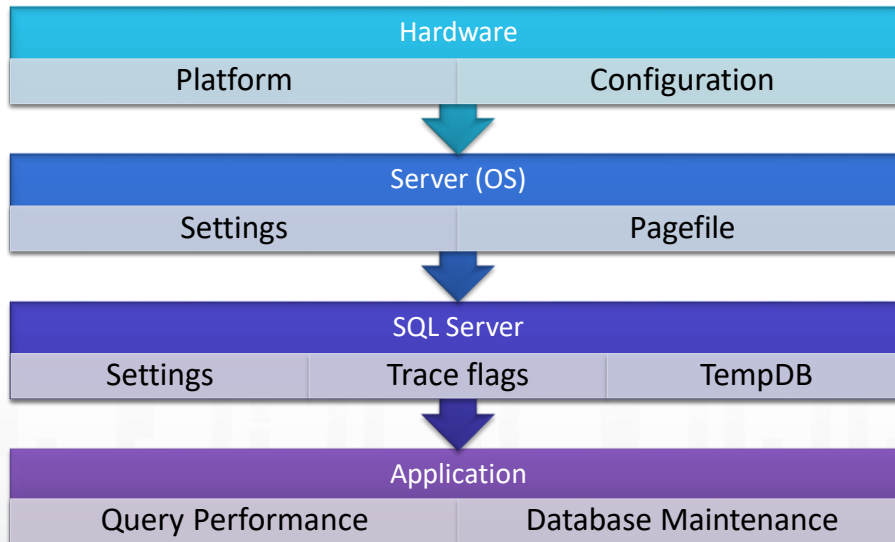
In short – a server where performance really counts



A tier-1 SQL Server is one where performance is critical. It may consist of one or more of the following:

1. High-end hardware – hundreds of cores and over 1TB of RAM
2. Very large databases – terabytes of data
3. Large analytic queries that sort and aggregate gigabytes of data at a time
4. High-throughput transaction processing systems that generate thousands of transactions per second
5. Business critical data that has strict high availability and disaster recovery requirements – minutes of downtime per year, RTO in the minutes, RPO near zero data loss

Performance considerations at every level

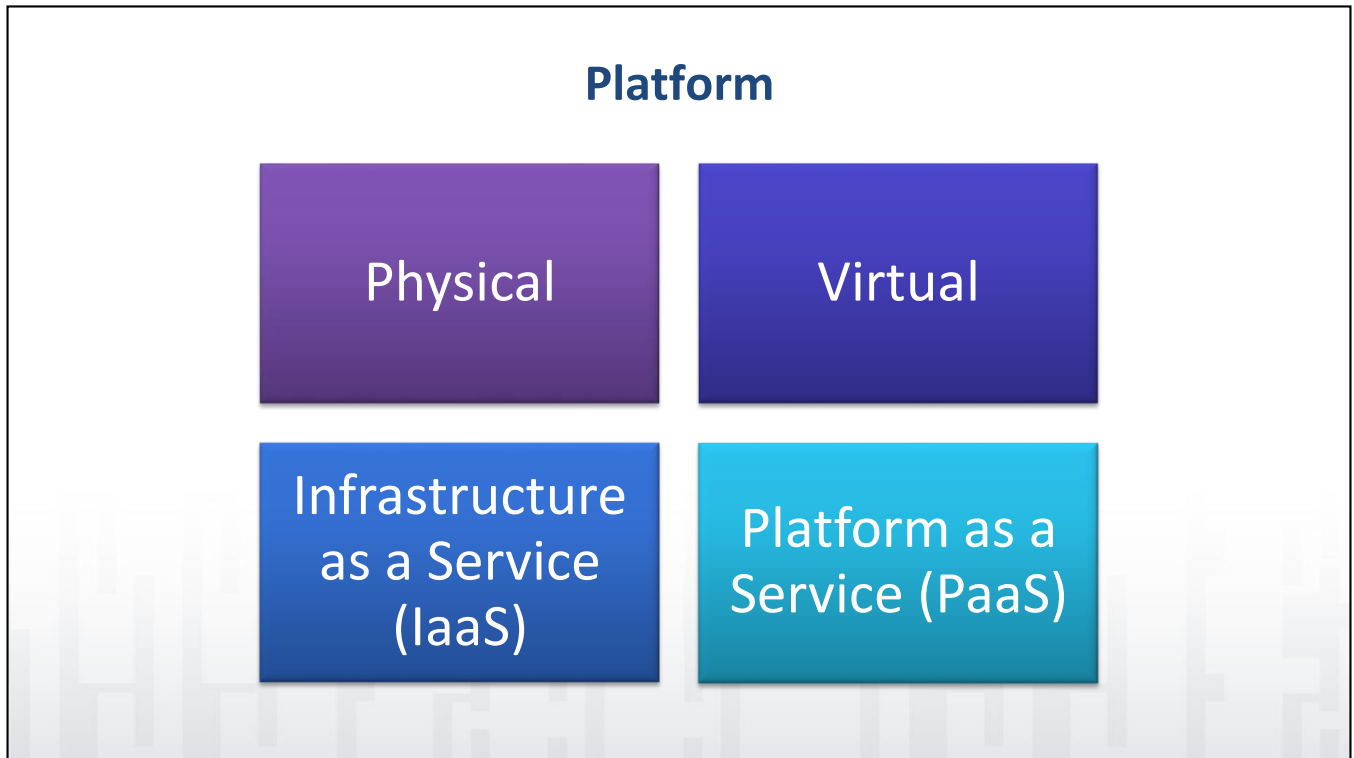


When designing a tier-1 system, performance must be considered at every level. The overall performance of the server depends on the performance at each level. Any weak spot in the stack can generate bottlenecks that resonate throughout the system.

Hardware



© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>



When choosing a platform to host a tier-1 SQL Server, any platform should be able to provide the performance and scalability required, provided that it is configured correctly. Each platform offers different capabilities and strengths around HA/DR, manageability and configuration options. In this presentation we will be focusing on on-premises SQL Server, so we will not cover the PaaS offering (Azure SQL DB), but keep in mind that hardware and software innovations in the PaaS space such as the new Hyperscale service tier can enable even the most intense Tier-1 workloads in a PaaS environment.

Physical Server

Know your workload

- OLTP: Large number of fast cores, large amount of memory, disk may not be as critical
- OLAP: Individual core clock speed not as critical, large amount of memory, high-end disk subsystem

Disk considerations

- Type of storage not as important as the specifications
- Size the storage subsystem based on number of IOPS and latency requirements, not just data size
- For very high-end systems, consider innovations in I/O such as persistent memory
- Keep in mind SQL Server does not yet support 8K sector size drives, support will be coming in a future release

With any platform, the choice for the number of cores, amount of memory and disk configuration will depend on the workload that will be running on the system. Running SQL Server on bare metal guarantees that all the resources on the system are dedicated to the workload on the server, but the trade off is less flexibility for scalability, portability and HA/DR.

In order to properly size the server, it is critical to understand the workload that will be running. For highly-concurrent transaction processing (thousands of short, fast transactions running concurrently), speed of individual CPUs and number of cores is important to maximize concurrent processing, particularly if sub-millisecond latency is required. It is also important to ensure there is enough memory to support the active portion of the database to prevent I/O delays during queries. If there is enough memory to host the entire database, or at least the working portion, disk performance for the data file drives may not be critical. Having fast write times against the transaction log disk is crucial, however, so care should be taken to configure these disks for minimal write latency.

For analytical processing, there are typically fewer concurrent queries, but they are long-running and query large amounts of data. For this, disk performance is critical as most of these workloads are against VLDBs where holding the data in memory may not be practical. In these queries, individual core speed is less important, but having a large number of cores to sustain parallel processing is key. Analytical queries require large amounts of memory, so a large amount of RAM in the system is also important. For these systems, transaction log performance is less critical so storage should be optimized for high-performance reads.

When it comes to sizing storage systems, understanding the I/O patterns of the system is critical. It is not enough to request storage based on the size of the volume required to host the database, the system needs to be configured to deliver the number of IOPS required at a latency level that supports the needs of the application. The choice of hardware here is less important than

understanding the latency and throughput capabilities of the system and matching them to the workload requirements.

Virtual Server Considerations

Same recommendations as physical servers for resource allocation

Minimal performance penalty, if any

More flexibility for HA/DR solutions

Configuration of host server is key

- Don't overcommit resources
- Ensure Power Management is set to High Performance
- Take care to configure vNUMA appropriately

Sizing virtual systems is largely the same as physical systems with respect to number of cores, amount of memory and storage system configuration. Virtualization allows for more flexibility in scalability of the system and HA/DR solutions. There is some overhead involved in the virtualization layer, but the performance hit is generally minimal, given that the host server is configured correctly. For performance critical workloads such as a tier-1 SQL Server, it's important to ensure that the resources configured in the VM are actually available. Avoid overcommitting resources on the host and using dynamic resource allocation. It is also important to pay attention to host server configuration such as power management and NUMA settings. Emulating NUMA configuration in the guest OS is important for SQL Server performance, particularly on systems with a large number of cores and large amount of RAM. Pay close attention to the NUMA settings within the guest OS to ensure that NUMA nodes are balanced.

Infrastructure as a Service Considerations

Know your workload, and tune before you go

- Throwing hardware at the problem no longer cost effective
- Streamline application before migration if possible

Sizing recommendations

- DS3_v2 or higher for SQL Enterprise Edition
- DS2_v2 or higher for SQL Standard Edition

I/O will likely be the biggest challenge

- Use Premium Storage
- Know your IOPS and stripe across multiple disks to achieve higher throughput
- Implement DB I/O optimizations such as page compression and instant file initialization
- Other best practice recommendations can be found here:
<https://aka.ms/SQLIaaSPerf>

IaaS servers can be considered similar to virtualization, the difference here is that there is no control over the host system. In these environments, you are paying for what you configure, so be sure to carefully analyze the workload and right-size the IaaS VM based on workload patterns. In Azure, I/O is typically the bottleneck because throughput limits are hard enforced. Be sure to determine the throughput needs of the workload and configure the appropriate number of Premium disks. Typically more than one disk will be required to get the IOPS required, so leverage Storage Spaces to stripe volumes across disks for maximum performance.

Server (OS)



© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Server (OS) Configuration

Apply the latest version of the OS if possible

Keep up to date with OS updates

Power plan must be set to High Performance

- Sets CPU clock at full-speed (if not restricted in BIOS)
- On Intel CPUs enables temporary overclock if needed (a.k.a. Turbo Boost)

Windows Paging File

- Don't waste disk space, huge page file is not required
- If SQL Server memory is being paged out (Error 17890: A significant part of SQL server process memory has been paged out), add more memory

NTFS configuration

- Format data and log drives in NTFS 64-KB blocks
- Alignment is automatic starting with Windows 2008, review if partitions were upgraded
- Turn off NTFS file encryption and compression

Also evaluate update modules of hardware drivers and firmwares and install if necessary, as most of these are designed to either correct bugs or increment performance.

Windows Operating system introduced energy saving settings back in 2008, by means of new power saving settings that allow saving the power consumption whenever possible.

By default, the Windows sets the power saving mode to “Balanced”, which means that when the system is not “busy” various components, such as the CPU and storage, are scaled back for saving the power consumption.

In some cases, you may experience degraded overall performance on a Windows Server machine when running with the default (Balanced) power plan.

On Intel X5500 and other last-generation CPUs, the clock is throttled down to save power (Processor P-state), and only increases when CPU utilization reaches a certain point.

By default, Windows Server sets the Balanced power plan, which enables energy conservation by scaling the processor performance based on current CPU utilization.

The Minimum and Maximum Processor Performance State parameters are expressed as a percentage of maximum processor frequency, with a value in the range 0 – 100.

If a server requires ultra-low latency, invariant CPU frequency, or the very highest performance levels, such as a database server, it might not be helpful that the processors keep switching to lower-performance states. As such, the High Performance power plan caps the minimum processor performance state at 100 percent.

On modern high end machines, you may need more than 8GB (16GB is a good rule of thumb). On

x64, the VAS is quite large and the kernel memory on those machines could *potentially* grow to the size of the physical memory. But from past CSS experience with kernel memory dumps on 64-bit environments the below guidelines are acceptable:

Server (OS) Configuration (Cont)

Apply the latest version of the OS if possible

Keep up to date with OS updates

Power plan must be set to High Performance

- Sets CPU clock at full-speed (if not restricted in BIOS)
- On Intel CPUs enables temporary overclock if needed (a.k.a. Turbo Boost)

Windows Paging File

- Don't waste disk space, huge page file is not required
- If SQL Server memory is being paged out (Error 17890: A significant part of SQL server process memory has been paged out), add more memory

NTFS configuration

- Format data and log drives in NTFS 64-KB blocks
- Alignment is automatic starting with Windows 2008, review if partitions were upgraded
- Turn off NTFS file encryption and compression

- A kernel only dump on midrange machines would grow up 8 GB and recommend paging file size of at least 8GB.
- A kernel only dump on a high end machine would grow up to 16 GB and paging file of the size around 16 GB.

Note that the size of a kernel memory dump will vary based on the amount of kernel mode memory allocated by the operating system and the drivers that are present on the system.

However, the most accurate way to set page file size is using the following rule of thumb: determine the minimum and maximum page file size, monitoring the memory used by processes over a relevant time, namely by collecting the performance counter Memory\Private Bytes, which is the current size, in bytes, of memory that this process has allocated that cannot be shared with other processes. Then set it according to the below:

- Minimum page file = sum of peak private bytes that are used by each process on the system. Then, subtract the amount of memory on the system.
- Maximum page file = sum of peak private bytes that are used by each process on the system. Then, add a margin of additional space. Do not subtract the amount of memory on the system. The size of the additional margin can be adjusted based on your confidence in the snapshot data that is used to estimate page file requirements.

Do the above exercise in a top-tier machine, and then in a medium-tier, etc. Whatever these are for your estate. And then use those conclusions as your standard for those tiers.

Server (OS) Configuration (Cont)

Apply the latest version of the OS if possible

Keep up to date with OS updates

Power plan must be set to High Performance

- Sets CPU clock at full-speed (if not restricted in BIOS)
- On Intel CPUs enables temporary overclock if needed (a.k.a. Turbo Boost)

Windows Paging File

- Don't waste disk space, huge page file is not required
- If SQL Server memory is being paged out (Error 17890: A significant part of SQL server process memory has been paged out), add more memory

NTFS configuration

- Format data and log drives in NTFS 64-KB blocks
- Alignment is automatic starting with Windows 2008, review if partitions were upgraded
- Turn off NTFS file encryption and compression

SQL Server data pages are 8192 bytes, so the default NTFS block size (4096 bytes) reads only half a page and effectively doubles the number of I/O operations. Consider formatting the DATA drives in 64-KB blocks, because SQL Server commonly does an eight page read-ahead to improve performance during range and table scans. If using the FILESTREAM feature, also format the volume that supports it with a 64-KB block size. Engage your hardware vendor to optimally configure these values as they could diverge on a specific SAN or Vendor.

Turn off NTFS file encryption and compression, especially on directory containers that contain any database files, as these introduce a degree of overhead when accessing these files.

SQL Server



© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Pedro starts

Core Licensing vs. CAL Licensing

Make sure you install the correct version of the binaries

For core based licensing, install Enterprise Core to use all the cores on the server

Watch for the following message in the error log:

SQL Server detected 2 sockets with 12 cores per socket and 24 logical processors per socket, 48 total logical processors; using 20 logical processors based on SQL Server licensing.

SQL Server 2012 introduced two different forms of licensing: core-based and CAL-based. These different licensing models are built into the SQL Server binaries directly, so you need to take care to download the appropriate installation media for the licensing model you are using. The most common licensing model is core-based licensing which allows SQL Server to leverage all the cores in the system. With CAL-based licensing, the server is restricted to 20 physical processors or 40 logical processors if hyper-threading is enabled. Because of this, if you accidentally install the CAL-licensing binaries, even if you have hundreds of cores in the system SQL Server will only make use of at most 40 cores. Unfortunately, it is not always easy to tell which version of the binaries you have. Once they have been downloaded, the file names are all the same and there is no visible indicator of the license model. If you have core-based licensing, be sure to download the media labeled “Enterprise Core” to get the core-based licensing binaries. If you’re not sure what you have installed on a server, examine the startup messages in the SQL Server error log and look for a message about cores in use.

This message indicates that CAL-based licensing has been installed:

SQL Server detected 4 sockets with 6 cores per socket and 6 logical processors per socket, 24 total logical processors; using 20 logical processors based on SQL Server licensing.

Core vs. CAL Licensing in Setup :: SQL Server 2019

Product Key
License Terms
Global Rules
Product Updates
Install Setup Files
Install Rules
Feature Selection
Feature Rules
Feature Configuration Rules
Ready to Install
Installation Progress
Complete

MICROSOFT PRE-RELEASE SOFTWARE LICENSE TERMS
MICROSOFT SQL SERVER 2019 COMMUNITY PREVIEW 3 (CTP3)

These license terms are an agreement between Microsoft Corporation (or based on where you live, one of its affiliates) and you. Please read them. They apply to the pre-release software named above, which includes the media on which you received it, if any. The terms also apply to any Microsoft

☐ I accept the license terms and [Privacy Statement](#)

Notice: A paid SQL Server edition product key has been provided for the current action - Enterprise. Please ensure you are entitled to this SQL Server edition with proper licensing in place for the product key (edition) supplied.

Warning: Enterprise Server/CAL license Product Key supplied. This edition limits SQL Engine CPU utilization to 20 physical cores, or 40 logical cores with Hyper-threading enabled. SQL Server detected 2 sockets with 12 cores per socket and 24 logical processors per socket. 48 total logical processors are available

☐ Check this box to acknowledge this limitation to continue or click Back/Cancel to enter an Enterprise Core product license that supports up to the operating system maximum.

Copy Print

Only if HW above CAL limit is detected

NUMA

Important for SQL Server memory object management and thread scheduling

If running in virtualized environment, emulate host NUMA settings

- Avoid uneven and/or partially overlapping NUMA nodes

Auto Soft-NUMA

- Enabled by default on SQL Server 2016 and later, for servers with more than 8 processors in a single NUMA node
- SQL Server 2014 SP2 needs trace flag 8079
- Important to consider Soft-NUMA when calculating MAXDOP

Newer hardware implements more than one system bus, each serving a small set of processors, with each group of processors having its own memory and possibly I/O channels. Each of these groups has its own memory, Scheduler Monitor task, Resource Monitor task, IOCP (I/O Completion Port) and Lazywriter, which can help reduce I/O and Lazywriter bottlenecks .

However, each CPU can access memory associated with the other groups in a coherent way. Each of these groups is called a NUMA node, and the number of CPUs within a hardware-NUMA node depends on the hardware vendor.

Because it is faster to access local memory than the memory associated with other NUMA nodes (hence the reason for the name, **non-uniform memory access**), this architecture provides better performance for NUMA aware applications such as SQL Server, by allowing greater memory locality for a group of CPUs (schedulers) when tasks are processed on the node.

On NUMA configurations, if it is necessary to change CPU affinity from the default setting, **set the CPU Affinity mask equally spread between NUMA nodes**. Consider the scenario of a multiple instance Failover Cluster, where it is necessary to limit the number of CPUs for a given instance, and there are more than two (2) CPUs per instance.

- The guideline is to distribute the CPU affinity evenly between NUMA nodes, in multiples of 2.
- In this scenario, also consider that you must not enable the same CPU in both the CPU

Affinity mask and the Affinity I/O mask options.

- Be aware that if one CPU is on its own in a NUMA node, it will not be possible to leverage parallelism on those CPUs.

Parallelism

MAXDOP

- Default of 0 is rarely the correct setting
- Workload dependent – testing is the only way to know the right setting
- Look for recommendations from vendor if running a third-party application
- No testing?
 - If not NUMA, set to number of logical processors, no higher than 8
 - If NUMA, set to number of logical processors in a single NUMA node, no higher than 16
- Starting with SQL Server 2016, can set this at the database level

Cost threshold for parallelism

- Workload dependent – testing is the only way to know the right setting
- No magic number so default value of 5 is fine. If you detect problems, test – no blind changes
- Consider making a larger number if you want to favor OLTP over OLAP workloads on the same server

The option, max degree of parallelism, controls the number of processors that can be used to run a single SQL Server statement using a parallel execution plan. The default value for this configuration is 0 (zero), indicating that all available processors will be used.

Parallelism is often beneficial for longer running queries or for queries that have complicated execution plans. However, OLTP-centric application performance could sometimes suffer when parallel plans use more threads than the current number of physical processors can handle. Keeping in mind that multi-core CPUs are usually grouped and acting as NUMA nodes in SQL Server, use the guidelines in <http://docs.microsoft.com/sql/database-engine/configure-windows/configure-the-max-degree-of-parallelism-server-configuration-option#Guidelines> for more information.

Note that for a database scope config, 0 is the default value and indicates that the server configuration will be used instead. The MAXDOP at the database scope overrides (unless it is set to 0) the max degree of parallelism set at the server level.

In most cases, the default “cost threshold for parallelism” option should not be changed from the default value of 5, unless a performance benefit was confirmed in a test environment with similar hardware and SQL instance configurations.

This option is currently an abstracted unit of cost (not expressed in seconds), based on the cost of executing a serial plan, and specifies the threshold at which the query engine creates and runs parallel plans for queries instead.

SQL Server ignores the “cost threshold for parallelism” value under the following conditions:

- Your computer has only one processor.
- Only a single CPU is available to SQL Server because of the affinity mask configuration option.

- The max degree of parallelism option is set to 1.

Note that if MAXDOP query option is used, then SQL Server will not ignore the cost threshold for parallelism value.

Parallelism in Setup :: SQL Server 2019

Database Engine Configuration

Specify Database Engine authentication security mode, administrators, data directories and TempDB settings.

Product Key
License Terms
Global Rules
Product Updates
Install Setup Files
Install Rules
Feature Selection
Feature Rules
Instance Configuration
PolyBase Configuration
Server Configuration
Database Engine Configuration
Consent to install Microsoft R ...
Consent to install Python
Feature Configuration Rules
Ready to Install
Installation Progress
Complete

Server Configuration | Data Directories | TempDB | MaxDOP | FILESTREAM

When an instance of SQL Server runs on a computer that has more than one CPU logical core, it detects the best degree of parallelism, that is, the number of processors employed to run a single statement, for each parallel plan execution. MAXDOP specifies the maximum number of cores to utilize for this instance.

Detected logical CPU cores on this computer: 24

Maximum degree of parallelism (MaxDOP)*:

** The displayed default value was either calculated by Setup, or was explicitly specified on the Setup command line with the /SQLMAXDOP parameter.*

You can modify the MaxDOP here to be used as the default in all query executions for this instance, unless overridden at the query level. To suppress parallel query plan generation, set MaxDOP to 1.

See [Configure the max degree of parallelism Server Configuration Option](#) for more information.

< Back

Next >

Cancel

Memory

Max server memory

- Common rule of thumb is 85%, but this is not appropriate for all workloads
- Not all SQL Server memory falls within max server memory limit
- If running things like SSIS or SSAS on the same server, adjust accordingly to prevent SQL Server from stepping on other processes and vice-versa

For an x64 server with 32GB RAM and 32 logical CPUs:

2 MB stack size * 960 default worker threads on 32 CPU machine
 + 4 GB for OS
 = roughly 6 GB
 Max server memory = 26 GB

Keep in mind that there are some memory allocations in SQL Server that happen directly through windows (Virtual Alloc) rather than going through the memory manager and thus do not adhere to the max server memory limit. These would be primarily the thread stacks (memory associated with working threads) and DLLs (the SQL Server engine DLLs themselves, plus any CLR DLLs and third-party DLLs).

Consider capping the max server memory configuration for each running SQL instance in a multi-instance server. Monitor overall consumption of the SQL Server process in order to determine memory requirements.

To be more accurate with these calculations for a single instance:

- From the total OS memory, reserve 1GB-4GB to the OS itself.
- Then subtract the equivalent of potential SQL Server memory allocations outside the **max server memory** control, which is comprised of **stack size * calculated max worker threads + -g startup parameter** (or 256MB by default if -g is not set). What remains should be the max_server_memory setting for a single instance setup.

See <http://docs.microsoft.com/sql/database-engine/configure-windows/server-memory-server-configuration-options#setting-the-memory-options-manually> for more information on stack sizes and calculated worker threads.

For example, on a x64 machine with 32GB RAM and 32 logical CPUs, max server memory should be 26GB. This is the result of taking the overall 32GB memory, and subtracting roughly 6GB (2MB x64 worker thread size * 960 threads default on 32 CPU machine + 4GB for OS).

Memory

Max server memory

- Common rule of thumb is 85%, but this is not appropriate for all workloads
- Not all SQL Server memory falls within max server memory limit
- If running things like SSIS or SSAS on the same server, adjust accordingly to prevent SQL Server from stepping on other processes and vice-versa

Min server memory

- Default is fine for most workloads
- Consider setting higher for the following conditions
 - Running on a virtual server with an overcommitted host
 - Running with other workloads on the same server
 - Running multiple instances of SQL Server on the same server (FCI for example)

In a failover cluster configuration, when several instances can exist concurrently in the same node, set the `min_server_memory` parameter instead of `max_server_memory` for the purpose of reserving memory for an instance. Use this recommendation together with what was said for `max_server_memory`, where in this case it's used mainly to prevent memory exhaustion in the OS.

Memory in Setup :: SQL Server 2019

Database Engine Configuration

Specify Database Engine authentication security mode, administrators, data directories, TempDB, Max degree of parallelism, Memory limits, and Filestream settings.

Product Key

License Terms

Global Rules

Product Updates

Install Setup Files

Install Rules

Feature Selection

Feature Rules

Instance Configuration

Server Configuration

Database Engine Configuration

Feature Configuration Rules

Ready to Install

Installation Progress

Complete

Server Configuration Data Directories TempDB MaxDOP **Memory** FILESTREAM

SQL Server can change its memory requirements dynamically based on available system resources. However, in some scenarios you can configure the range of memory (in MB) that is managed by the SQL Server Memory Manager for this instance, by specifying min server memory and/or max server memory.

☐ Recommended ☒ Default

Min Server Memory (MB):

Max Server Memory (MB):

** The displayed recommended values were calculated by Setup based on your system configuration and edition, unless these were explicitly specified in the Setup command line using the /SQLMINMEMORY and /SQLMAXMEMORY parameters.*

For more information see: [Server Memory Server Configuration Options](#).

☐ [Click here to accept the recommended memory configurations for the SQL Server Database Engine](#)

< Back

Next >

Cancel

Memory – Large Pages

Lock pages in memory (LPIM)

- Good idea to set on most servers (and another reason to set Max Server Memory)
- Needed for large pages

Large page allocations

- Enable with TF 834 to set large-page allocations for the buffer pool, columnstore and in-memory OLTP
- Good for large memory machines with analytical workloads
- Not supported with columnstore indexes – in SQL Server 2019, use TF 876 instead (preview) to set large-page allocations for columnstore only, without SQL permanently hanging on to the memory

Enabling the "Lock Pages in Memory" user right ensures that memory in use by SQL Server is not paged out when other applications or the OS demands more memory.

This does not affect SQL Server's dynamic memory management, allowing it to expand or shrink at the request of other memory clerks. This permission will allow for a copy of the buffer cache to stay resident in physical memory, preventing the system from paging the data to virtual memory on disk.

Note that when using the Lock Pages in Memory user right it is recommended to set an upper limit for max server memory.

TF 845 as a start-up parameter is needed in SQL Server 2008 SP1 CU2 Standard editions to use the "Lock pages in memory" user right. Starting with SQL Server 2012, TF 845 is no longer needed.

TF 834 - Used to set large-page allocations for the buffer pool, columnstore and in-memory OLTP (hectaton) by enabling the large-page memory model. Also controls the pre-faulting of SQLOS/SQLPAL memory at startup, and that memory is not returned to the OS.

TF 876 (SQL 2019) - Used to set large-page allocations for columnstore, without SQL permanently hanging on to the memory, and without enabling the large-page memory model.

TF 877 (SQL 2019) - Used to enable the large-page memory model for the buffer pool and in-memory OLTP (hekaton). Also controls the pre-faulting of SQLOS/SQLPAL memory at startup, and that memory is not returned to the OS.

I/O Considerations

Turn on instant file initialization

- Starting with SQL Server 2016, leverage SQL Server Setup to enable
- Starting with SQL Server 2019, leverage SQL Server Configuration Manager to enable (for all versions)

File placement

- Still recommend separate logical drives for data, transaction log and tempdb, even on a SAN
- No need for multiple data files in the same filegroup unless spreading across multiple VHDs, or hitting allocation contention on a user database (similar to TempDB)
- If using multiple files per filegroup, ensure all are of equal size – proportional fill algorithm
- For IaaS, use Storage Spaces to create logical partitions across multiple disks rather than database files

Using the Instant File Initialization feature will allow a boost in performance, because when extending or creating a data file (does not apply to log files), it will not be zeroed right away, saving time and I/O cycles.

This permission does come with a small security risk, because by not zeroing out the existing space, when deleting data for example, there is a possibility that data could still be read, even though it has been “deleted”, until some other data writes on that specific area of the data file.

However, the performance benefits outweighs the security risk and hence the reason for this recommendation.

This can be accomplished by granting the "Perform volume maintenance tasks" permission to the SQL Server account, which is set in the security policy console (secpol.msc). Starting with SQL Server 2016, leverage setup to enable this permission at install time.

Use multiple LUNs for data, transaction log and tempdb files, and ensure that an adequate number of spindles for each LUN exist to support the I/O requirements with an acceptable latency. You may choose to use multiple filegroups for administration requirements such as backup / restore, partial database availability, physical separation of I/O such as for certain tables, indexes, etc. Within each filegroup, use data files to “stripe” the database across your specific I/O configuration (physical disks, LUNs, etc.). Remember to perform basic throughput testing of the I/O subsystem prior to deploying SQL Server. Make sure these tests are able to achieve the design specs for I/O requirements with an acceptable latency. When testing your I/O, keep in mind the SQLIO Benchmark Tool has been retired and replaced by DiskSpd.exe. You can download the tool and documentation from <http://aka.ms/diskspd>.

If using multiple data files per filegroup, then these should be of equal size within each Filegroup, as SQL Server uses a proportional fill algorithm that favors allocations in files with more free space. You

may recognize this recommendation as directed at the TempDB, but it applies to all user databases also.

Setting LPIM and IFI in Config Manager

SQL Server 2017 Configuration Manager

SQL Server (SQL2017) Properties

Log On Service FILESTREAM
AlwaysOn High Availability Startup Parameters Advanced

Advanced

Clustered	No
Customer Feedback Reporting	Yes
Data Path	D:\Program Files\Microsoft SQL S
Dump Directory	D:\Program Files\Microsoft SQL S
Error Reporting	Yes
File Version	2017.140.3103.1
Install Path	d:\Program Files\Microsoft SQL S
Instance ID	MSSQL14.SQLE2017
Language	1033
Registry Root	Software\Microsoft\Microsoft SQ
Running under 64 bit OS	No
Service Pack Level	0
SQL States	4194821
Startup Parameters	-dD:\Program Files\Microsoft SQL
Stock Keeping Unit ID	1804890536
Stock Keeping Unit Name	Enterprise Edition
Version	14.0.1000.169

Clustered
Indicates whether SQL Server is part of a cluster or not.

OK Cancel Apply Help

SQL Server 2019 Configuration Manager

SQL Server (SQL2017) Properties

Log On Service FILESTREAM
Always On Availability Groups Startup Parameters Advanced

Advanced

Clustered	No
Customer Feedback Reporting	Yes
Data Path	D:\Program Files\Microsoft SQL S
Dump Directory	D:\Program Files\Microsoft SQL S
Error Reporting	Yes
File Version	2017.140.3103.1
Install Path	d:\Program Files\Microsoft SQL S
Instance ID	MSSQL14.SQLE2017
Instant File Initialization	Yes
Language	1033
Lock Pages In Memory	Yes
Registry Root	Software\Microsoft\Microsoft SQ
Running under 64 bit OS	No
Service Pack Level	0
SQL States	4194821
Startup Parameters	-dD:\Program Files\Microsoft SQL
Stock Keeping Unit ID	1804890536

Instant File Initialization
Grants Perform Volume Maintenance Task privilege to SQL Server Database Engine Service Account.

OK Cancel Apply Help

TempDB

Fast storage

- No need for redundancy, optimize for performance (avoid RAID 5 and 6)
- Consider flash, SSD or even PMEM

Multiple data files

- Start with one data file per logical processor up to 8
- Add more in multiples of 4 files until you no longer see contention
- Files must be equally sized
- If adding new files, be sure to restart SQL Server to rebalance across all the files
- Starting with SQL Server 2016, leverage SQL Server Setup to configure

Upgrade to the latest version to get all the new fixes and enhancements

- Latest CU for SQL Server 2016 and 2017
- SQL Server 2019 CTP 3.0 to try memory-optimized metadata and cache improvements

Pam starts

Optimize TempDB file layout by creating one data file for each logical processor, and put them on separate LUNs if possible (although this is not a requisite), up to 8 files (mileage may vary). If there is a need to increase above the initial 8 files, do so in multiples of 4 files.

The TempDB system database is a global resource available to all users connected to the instance of SQL Server and holds all temporary tables and temporary stored procedures. It also fills any other temporary storage requirements such as worktables and workfiles generated by SQL Server. The configuration of TempDB can affect your SQL Server performance.

Having multiple TempDB data files can reduce contention and improve performance on active systems. This is because there will be one or more SGAM pages for each file, the main point of contention for mixed allocations.

Dividing TempDB into multiple data files of equal size provides a high degree of parallel efficiency in operations that use TempDB. These multiple files do not necessarily need to be on different disks or spindles unless you are also encountering I/O bottlenecks as well.

One disadvantage of having too many TempDB files is that every object in TempDB will have multiple IAM pages. In addition, there will be more switching costs as objects are accessed as well as more managing overhead. On very large systems, 8 TempDB data files may be sufficient, but reconsider this based on the workload. Be aware that if one of the data files is larger than the others, this mechanism will not be effective, so check if growth via AUTOGROW has happened on one file and broke the proportional fill.

Consider enabling the trace flag 1118 if the TempDB is in high demand and latch contention is observed on wait resources like 2:1:1 (DB 2, file 1, page 1 - the first PFS page) and 2:1:3 (the first SGAM page). Otherwise, test the trace flag impact on your workload. This trace flag will allow

uniform extent allocations (or full-mixed) instead of mixed extent allocations on all databases. In SQL Server 2016, this is the default behavior without the trace flag, and for TempDB only.

Trace Flags

For SQL Server 2016, 2017 and 2019

- 174: SOS_CACHESTORE spinlock contention, KB 3026083
- 3468: Disable indirect checkpoints on tempdb, KB 4040276
- 7471: Parallel statistics update jobs, KB 3156157
- 2566: Faster CHECKDB command, KB 945770
- 7412: Enabled lightweight profiling, KB 3170113 (not needed in SQL Server 2019)
- 4199: Optimizer fixes (Optional, test workload first, can also be set via a database-level configuration)

For SQL Server 2012 and 2014, all of the above plus

- 1117: Autogrow all files in the same filegroup at the same time (tempdb, KB 2154845)
- 1118: Turn off mixed page allocations (tempdb, KB 2154845)
- 2371: Enable linear recompilation threshold for large tables, KB 2754171
- 6498: Avoid compilation waits for concurrent large queries, KB 3024815
- 6532,6533,6534: Spatial query performance, KB 3107399
- 8048,8079: CMEMTHREAD waits for systems with more than 8 cores per NUMA node
- 8075: VAS fragmentation, KB 3074434

Refer to the page <http://aka.ms/traceflags> for more information.

Application



© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Partitioning?

A perfect fit for VLDBs

- Sliding windows scenarios
- Fast archiving
- Allows partition switching and optimal filegroup backup/restore (read-only data)

Find an optimal partitioning scheme based on the existing business requirements

- Examine the partitioning function, the data in the table, and the business uses of the data
- When partitioning any index (including clustered), the index key must contain the partitioning column
- If a natural partitioning key is available, use it (date, incremental ID)

Leverage partition-aligned indexes

- Partitioned indexes, functions, and schemes should have the same partitioning key and ranges
- To enable partition switching, all indexes on the table must be aligned
- Use the partitioning key in your query predicates (WHERE / JOIN) to encourage partition elimination
- Beware of queries with TOP, MIN/MAX as these must scan all partitions

Although partitioned indexes can be implemented independently from their base tables, it generally makes sense to design a partitioned table and then create an index on the table. When you do this, SQL Server automatically partitions the index by using the same partition scheme and partitioning column as the table. As a result, the index is partitioned in essentially the same manner as the table. This makes the index aligned with the table.

SQL Server does not align the index with the table if you specify a different partition scheme or a separate filegroup on which to put the index at creation time.

Aligning an index with a partitioned table is particularly important if you anticipate that it will expand by taking on additional partitions, or that it will be involved in frequent partition switches. For more information, see [Designing Partitions to Manage Subsets of Data](#). When a table and its indexes are in alignment, SQL Server can switch partitions quickly and efficiently while maintaining the partition structure of both the table and its indexes.

The **Tuning Options** tab of the Database Engine Tuning Advisor provides an **Aligned partitioning** setting to specify that new recommended indexes be aligned with their base tables. The **Keep aligned partitioning** setting can be used for the same purpose and can also be used to drop existing nonaligned indexes. For more information, see [Database Engine Tuning Advisor \(Tuning Options Tab\)](#). Generally, the Database Engine Tuning Advisor can be used to recommend indexes for performance, and this can be a mix of aligned and nonaligned indexes. For more information, see [Database Engine Tuning Advisor Overview](#).

Designing a partitioned index independently (unaligned) of the base table can be useful in the following cases:

The base table has not been partitioned.

The index key is unique and it does not contain the partitioning column of the table.

You want the base table to participate in collocated joins with more tables using different join

columns.

Collations

Beware of user databases with collation different from system databases

- Can lead to comparison issues such as “Cannot resolve the collation conflict between SQL_Latin1_General_CP1_CI_AS and Latin1_General_CI_AI in the equal to operation”

In SQL Server 2019, leverage the new UTF-8 collations

- UTF-8 is enabled when creating or changing an object’s collation to a collation with the “_UTF8” suffix, such as LATIN1_GENERAL_100_CI_AS_SC to LATIN1_GENERAL_100_CI_AS_SC_UTF8
- UTF-8 is only available to Windows collations that support supplementary characters (_SC), as introduced in SQL Server 2012
- CHAR and VARCHAR datatypes store UTF-8, NCHAR and NVARCHAR continues to use UCS-2 or UTF-16 encoding only and remain unchanged
- If using ASCII, significant storage savings may also be achieved.
For example, changing an existing column data type from NCHAR(10) to CHAR(10) using an UTF-8 enabled collation, translates into nearly 50% reduction in storage requirements. This is because NCHAR(10) requires 22 bytes for storage, whereas CHAR(10) requires 12 bytes for the same Unicode string

Pedro starts

Beware of user databases with collation different from system databases. If a database is defined using a collation that is different from the master and model databases, this could cause collation conflicts that prevent code from executing, namely because the TempDB database is created from the settings of the model database.

Some products (Sharepoint Server and SCOM for example) require a specific collation setting. If so, consider the above and create the products databases on a SQL Server that has that required collation.

SQL Server 2019 includes full support for the widely used UTF-8 character encoding as an import or export encoding, or as database-level or column-level collation for text data.

UTF-8 is allowed in the CHAR and VARCHAR datatypes, and is enabled when creating or changing an object’s collation to a collation with the “UTF8” suffix, such as LATIN1_GENERAL_100_CI_AS_SC to LATIN1_GENERAL_100_CI_AS_SC_UTF8. UTF-8 is only available to windows collations that support supplementary characters, as introduced in SQL Server 2012. Note that NCHAR and NVARCHAR allow UTF-16 encoding only, and remain unchanged.

Significant storage savings may also be achieved, depending on the character set in use. For example, changing an existing column data type from NCHAR(10) to CHAR(10) using an UTF-8 enabled collation, translates into nearly 50% reduction in storage requirements. This is because NCHAR(10) requires 22 bytes for storage, whereas CHAR(10) requires 12 bytes for the same Unicode string.

Database Maintenance

Maintain your indexes

- This may mean Rebuild or Reorganize depending on current level of fragmentation
- Workloads that scan are more vulnerable to performance issues with fragmented indexes
- Leverage intelligent management. Ex: Adaptive Index Defrag (<http://aka.ms/AID>)

Update statistics

- As needed, so leverage intelligent management. Ex: Adaptive Index Defrag (<http://aka.ms/AID>)
- Even more relevant if not using AUTO_UPDATE_STATISTICS

Implement Integrity checking

- DBCC CHECKDB runs depends on the individual business needs and the importance of the information in the database
- At minimum, run against all production databases at least once a week and review results
- Not running means not knowing of early signs of corruption, and can lead to data loss

Implement maintenance plans for clean-up tasks, rebuilding or reorganizing indexes, updating statistics and integrity checking. We recommend you review the cycle as set in the TigerToolbox at <http://aka.ms/MaintenanceSolution>. To implement only the index and statistics automatic grooming, refer to <http://aka.ms/AID>.

Also implement maintenance tasks that perform Transaction Log backups and Database backups as well.

Execute database integrity checks with DBCC CHECKDB regularly. The frequency in which DBCC CHECKDB is run against any particular database depends largely on the individual business needs and the importance of the information in the database. However, at a minimum, DBCC CHECKDB should be run against all production databases at least once a week. In addition, the results should be reviewed as soon as possible after execution completes in order to identify and resolve any errors before they become critical.

If DBCC CHECKDB uncovers any errors, it will specify the minimum repair level that should be used to fix the problem. Note that running DBCC CHECKDB with the repair option might not resolve the issue, depending on the type of damage uncovered. This could potentially result in a significant loss of data.

DBCC CHECKDB in SQL Server 2005 and later versions use an internal database snapshot to perform the checks in order to avoid blocking and concurrency problems.

Query Optimizer / Cardinality Estimation

Every new version introduces functional changes to the QO under the latest Database Compatibility Level

- Upgrading and staying in source Database Compatibility level is fully supported
- Still plan to certify ASAP on latest; New work should start on newer Database Compatibility
- Evaluate use of TF 4199 / DB scoped config “Query Optimizer Fixes”

A new version introduces a new CE version

- Mechanism for estimating required number of rows to satisfy query, using statistical models and heuristics
- No more “Legacy CE” vs “New CE” only
 - “Legacy CE” = CE 70, default for SQL Server 7.0 to 2012
 - “New CE” = CE 120 (SQL Server 2014) through CE 150 (SQL Server 2019)
- It's recommended to follow documented upgrade process to avoid increased risk of regression if upgrading Database Compatibility! Use Query Tuning Assistant in this case

Database compatibility level is a valuable tool to assist in database modernization, by allowing the SQL Server Database Engine to be upgraded, while keeping connecting applications functional status by maintaining the same pre-upgrade database compatibility level. As long as the application does not need to leverage enhancements that are only available in a higher database compatibility level, it is a valid approach to upgrade the SQL Server Database Engine and maintain the previous database compatibility level. For more information on using compatibility level for backward compatibility, see the [Using Compatibility Level for Backward Compatibility](#).

For new development work, or when an existing application requires use of new features, as well as performance improvements done in the query optimizer space, plan to upgrade the database compatibility level to the latest available in SQL Server, and certify your application to work with that compatibility level. For more details on upgrading the database compatibility level, see the [Best Practices for upgrading Database Compatibility Level](#).

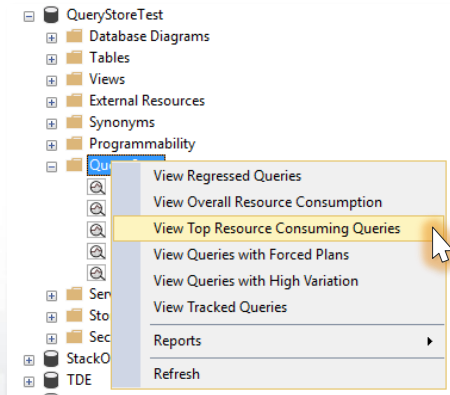
Starting with compatibility level 130, any new query plan affecting features have been intentionally added only to the new compatibility level. This has been done in order to minimize the risk during upgrades that arise from performance degradation due to query plan changes.

From an application perspective, the goal should still be to upgrade to the latest compatibility level at some point in time, in order to inherit some of the new features, as well as performance improvements done in the query optimizer space, but to do so in a controlled way. Use the lower compatibility level as a safer migration aid to work around version differences, in the behaviors that are controlled by the relevant compatibility level setting. For more details, including the recommended workflow for upgrading database compatibility level, see the [Best Practices for upgrading Database Compatibility Level](#). When upgrading Database Compatibility, use the Query Tuning Assistant (<https://docs.microsoft.com/en-us/sql/relational-databases/performance/upgrade->

[dbcompat-using-gta](#)) that guides you through the upgrade process and assists in finding and addressing regressions.

Query Store

Comprehensive query-performance information when you need it most!



Tiger Toolbox

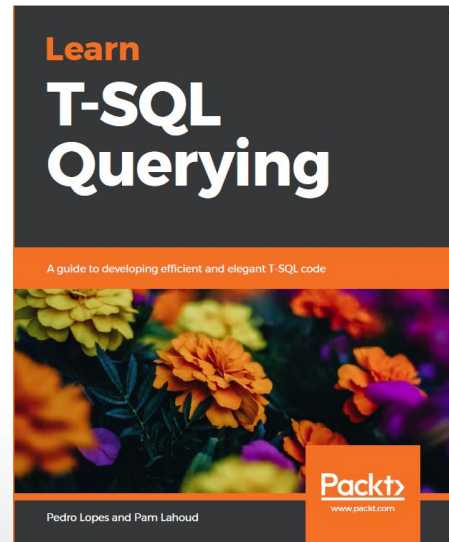
The SQL Server team shares a number of scripts and tools you can (arguably should) use every day in your SQL Server estate:

- Adaptive Index Defrag
- BPCheck (SQL Best Practices and Performance checks)
- Fixing transaction log VLF count
- Maintenance solution that includes integrity checking
- Waits and Latches information
- Usp_WhatsUp (Running and Blocked processes, Execution stats)
- System Health Session PowerBI
- Various other support scripts

Find these and more at <http://aka.ms/TigerToolbox>

References

- One bookmark to rule them all
<https://aka.ms/sqlshortcuts>
- @mssqltiger
- Continue learning with our new book!
<https://aka.ms/LearnTSQLQuerying>



© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

Thank you!



@SQLGoddess

THANKS



@SQLPedro

Questions?

Don't forget to complete an online evaluation!

Practical Guidance to Make Your Tier-1 SQL Server ROAR!

Your evaluation helps organizers build better conferences
and helps speakers improve their sessions.



SQL
intersection

Thank you!

Save the Date

Week of November 18, 2019
We're back in Vegas baby!

www.SQLintersection.com

