DBA Services

1. Performance Tuning Overview

Click here to expand...

A. Most Common Performance Issues (64% of all performance problems)

1. Insufficient or poor indexes
    a. Table scans impact disk performance and memory use, as well as lead to blocking
    b. It's possibly to have too many indexes, which lead to performance hits on data modification queries (INSERT, DELETE, or UPDATE operation)
2. Inaccurate or missing statistics
    a. The query optimizer makes choices based on row estimates that come from these statistics
3. Bad T-SQL
    a. Moving too much data, writing overcomplicated code, using wrong object types, etc.
4. Problematic execution plans
    a. Most of times, these are fixed through code changes, statistics updates, or new indexes
    b. Other times, this occurs due to parameter sniffing gone wrong
5. Excessive blocking
    a. A lack of resources, not enough memory, CPU, or fast enough disks can lead to additional blocking
6. Deadlocks
    a. Caused by blocking, but is something separate
    b. If all your queries complete fast enough, the chances of a deadlock are very slim
7. Non-set-based operations
    a. Caused by cursors and other types of loop operations to force a row-by-row style processing
8. Incorrect database design
    a. Ensuring that your database is properly normalized and data is stored properly (ex. dates go into a datetime column)
9. Poor execution plan reuse
    a. Caused by things like dynamic T-SQL or inappropriate parameters, preventing plan reuse or parameterization
10. Frequent recompilation of queries
    a. While recompilation is generally desirable, there can be too much due to volatile data or poor code

B. Overview of (Recursive) Query Performance Tuning Process

1. Set performance target for application
2. Analyze application performance
    a. Ensure servers are not overwhelmed
        i. Process of capturing performance metrics varies depending on if the server is on VMware, Hyper-V, Docker, AWS, Azure, etc.
        ii. In general, focus on collecting metrics on waits and queues, especially around disk I/O, memory, and CPU
        iii. Network (health of the routers, cables, Wi-Fi repeaters, etc.) can also affect performance
3. Identify resource bottlenecks
4. Ensure proper configuration for hardware, OS, platform, SQL Server, database, and applications
5. Identify costliest query associated with bottleneck
6. Optimize query

C. Identifying resource bottlenecks

This is a repetitive process that goes as follows:

1. Identify the bottleneck
2. Fix it
3. Validate the fix
4. Measure the impact and current performance
5. Start again with the next bottleneck

This process should be done for one bottleneck at a time, making one change at a time and validating that one change at a time.

D. In-depth Query Tuning Process Overview

1. Baseline performance and resource use of costliest query
2. Set performance target for query (ex. every query has to meet a three-second minimum operation, with a few exceptions)
3. Analyze and optimize factors (such as statistics) that influence query execution
4. Analyze query for common problems
5. Analyze query execution plan
6. Analyze and prioritize operators to identify bottlenecks
7. If warranted, modify query and/or index. Afterwards:
    a. Measure performance and resource use again
    b. Determine if query performance improved
        i. If not, undo changes!
8. Determine if query performance is acceptable
    a. If not, return to step 4

2. Creating a Baseline

A. Overview

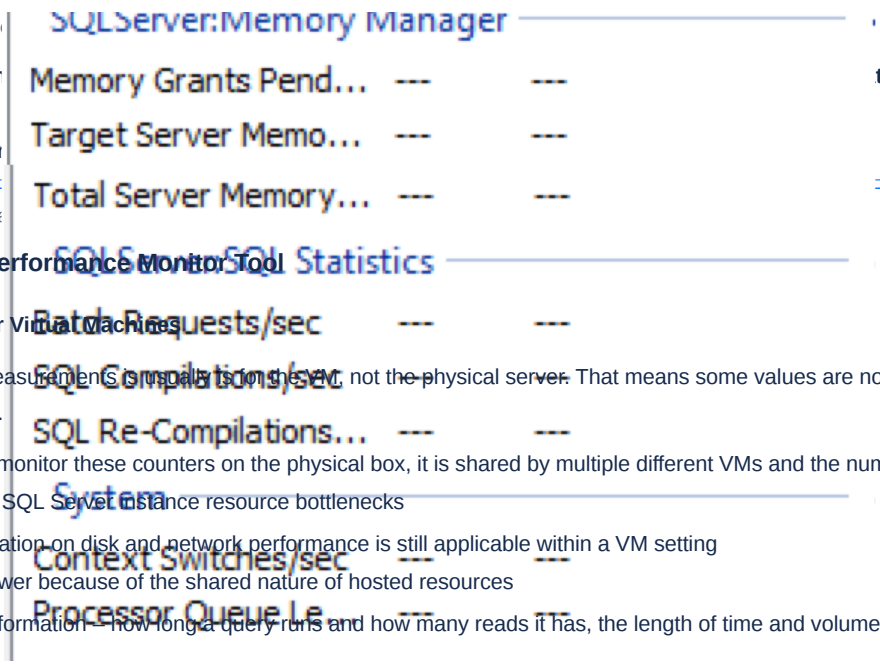
A. Overview

I. Create a reusable list of performance counters

Current Disk Queue ...	---	*
Disk Bytes/sec	---	*
Disk Reads/sec	---	*
Disk Transfers/sec	---	*
Disk Writes/sec	---	*
Process		
Private Bytes	---	*
Processor		
% Processor Time	---	*
% Processor Time	---	_Total
% Privileged Time	---	_Total
SQLServer:Access Methods		
FreeSpace Scans/sec	---	---
Full Scans/sec	---	---
Table Lock Escalatio...	---	---
Worktables Created/...	---	---
SQLServer:Buffer Manager		
Buffer cache hit ratio	---	---
Checkpoint pages/sec	---	---
Lazy writes/sec	---	---
Page life expectancy	---	---
Page reads/sec	---	---
Page writes/sec	---	---
Target pages	---	---
SQLServer:General Statistics		
User Connections	---	---
SQLServer:Latches		
Total Latch Wait Tim...	---	---
SQLServer:Locks		
Lock Timeouts/sec	---	---
Lock Wait Time (ms)	---	---
Number of Deadlock...	---	---

II. Create a counter l

1. **Because we are r** **ta sampling interval can be increased**
   a. **Scott Whighar**



B. Notes on the Performa

B. Notes on the Performance Monitor Tool

I. Considerations for Virtual Machines

On a VM, counter measurements usually is on the VM, not the physical server. That means some values are not going to accurately reflect physical reality.

1. Even if you could monitor these counters on the physical box, it is shared by multiple different VMs and the numbers can't be used to identify specific SQL Server instance resource bottlenecks
2. Most of the information on disk and network performance is still applicable within a VM setting
   a. I/O may be slower because of the shared nature of hosted resources
3. All query metric information – how long a query runs and how many reads it has, the length of time and volume of reads – will be accurate
4. Primarily, you'll find the memory and CPU metrics that are completely different and quite unreliable, since CPU and memory are shared between machines within a VM.
   a. A process may start on one CPU and finish it on another one entirely.
      i. An exception is the queues counters, such as processor queue length
   b. The memory allocated can change as the machine's demands for memory go up and down
   c. In general, both CPU and memory are going to be potentially slower because the management of the VM is getting in the way of the system resources
5. For major VM vendors, reliable documentation is available on how to monitor their systems and how to use SQL Server within their systems:
   a. VMware < VMware Workstation Pro Documentation >
   b. HyperV < Checklist: Optimizing Performance on Hyper-V - BizTalk Server >
6. For Azure SQL Database and SQL Server 2016+, the Query Store (Chapter 11) can be used for baselining
7. DMVs (Chapter 6) are not as reliable for baselining since they change depending the cache, reboots, failovers, etc. That said, they do provide a way to see an aggregated view of query performance.

II. Considerations for Capturing and Viewing Data

1. While real time graphs are available, it's recommended to captured data into a file (called a *data collector set*) for offline analysis.
   a. Performance Monitor graphing is more costly in terms of overhead. Only use them for short-term viewing of data, troubleshooting, and diagnosis.
   b. Counter logs are *sampled* (collected periodically), whereas the Performance Monitor graph is updated in real time.
2. Monitoring multiple performance objects with small sampling intervals could incur overhead on the system. (The number of counters for the selected objects does not add much overhead because it gives only an attribute of the object itself.)
   a. Consider baselining in phases, focusing on one performance object (along with its DMVs, QS, EE, various analyses, etc.) at a time

C. Memory Performance Analysis

C. Memory Performance Analysis

Memory bottlenecks are a priority because they also affect CPU. More CPU cycles are needed to write memory pages to disk so SQL Server can maintain enough free internal memory (via a process called *lazy writer*).

I. Memory Settings

The following script can be used to determine the *max server memory* setting for SQL Server:

scripts/max_server_memory.sql at main · bornsql/scripts

At least 2GB should be available for the OS and another 2GB if MedInformatix is hosted on the same server.

Microsoft recommends that *min server memory* is 0.

II. Memory Perfmon Counters

1. Memory - Available MBytes
    a. Extended periods of time with this value very low and SQL Server memory not changing indicates that the server is under severe memory stress.
        i. **Low memory is defined as less than 1GB by Brent Ozar and 10% of the installed memory by PAL**
    b. Can also look at Available Bytes/Kbytes for more granularity
2. Memory - Pages/Sec and Memory - Page Faults/Sec
    a. Pages/Sec measures the rate of hard page faults (where data must be retrieved from disk), whereas Page Faults/Sec measures the rate of total page faults, including soft faults (where the data can be retrieved from memory).
    b. **A baseline is essential to determine the expected normal behavior, which can range from 0 to 1,000 per second for Page Faults/Sec.**
        i. These numbers vary widely based on the amount and type of memory as well as the speed of disk access on the system
    c. Per Scott Whigham, some page faults are normal; look for consistently high numbers
    d. For more granularity, Pages/sec can be broken up into
        i. Pages Input/sec for page reads. This is what causes wait times in application.
        ii. Pages Output/sec for page writes. This can be ignored unless disk load is an issue.
    e. Process:Page Faults/sec can also be used to see specifically which process is causing excessive paging
3. Paging File - %Usage
    a. Paging is used by virtual memory to allows users to execute programs larger than the actual physical memory.
    b. **Ideally we shouldn't have to use the Paging File. Brent Ozar recommends a 0 or 1% usage at most, while PAL is fine with up to 70%.**
    c. Note that paging may be caused by problems external to SQL Server
    d. Paging File - %Usage Peak can also be used to see peak values
4. SQLServer:Buffer Manager - Buffer Cache Hit Ratio
    a. **Per Brent Ozar, it used to be recommended to keep this above 90%, which means the majority of reads are coming out of the buffer cache**
        i. This may not be true anymore depending on how reads are done in the system (ex. for a reporting workloads with lots of ad hoc queries)
        ii. A baseline value is needed to determine what's normal
5. SQLServer:Buffer Manager - Page Life Expectancy
    a. This indicates how long a page will stay in the buffer pool without being referenced
    b. **Brent Ozar recommends at least 180 seconds, while SQLwatch.io recommends > 300 seconds**
    c. A baseline needs to be established and monitored over time
        i. Reporting systems, as opposed to OLTP systems, are expected to have a lower value
        ii. This number is also expected to dip to very low levels during nightly loads
6. SQLServer:Buffer Manager - Checkpoint Pages/Sec
    a. This represents the number of pages moved to disk by a checkpoint operation
    b. **Should be relatively low: less than 30 per second for most systems**
    c. Higher values indicate a large number of writes occurring within the system, possibly indicative of I/O problems
7. SQLServer:Buffer Manager - Lazy Writes/Sec

a. This records how much dirty, aged buffers are removed from the buffer by the lazy write process, a system process that frees up the memory for other uses

b. **Should consistently be less than 20 for the average system**

c. Higher values possibly indicate I/O issues or even memory problems

8. SQLServer:Memory Manager - Memory Grants Pending
   a. Under normal conditions, this should be consistently 0 for most production servers
   b. To retrieve this value on the fly, run queries against the DMV `sys.dm_exec_query_memory_grants` . A `null` value in the column `grant_time` indicates that the process is still waiting for a memory grant.
9. SQLServer:Memory Manager - Target Server Memory (KB) and SQLServer:Memory Manager - Total Server Memory (KB)
   a. Target Server Memory (KB) indicates the total amount of dynamic memory SQL Server is willing to consume, while Total Server Memory (KB) indicates the amount of memory currently assigned to SQL Server
   b. **Target Server Memory (KB) should be close to the size of physical memory available, especially if the system is dedicated to SQL Server**
   c. **Total Server Memory (KB) should be close to Target Server Memory (KB).** If it's much less, make sure (1) max server memory is configured properly and (2) the system did not just start up (since there is a *warm-up phase* to expand memory allocation and bring data pages into memory). Ruling those two out, we can conclude SQL Server simply has a low memory requirement.
   d. **Systems with a low memory requirement usually have 5,000 free pages or more.** The status of memory allocation can also be queried via the DMV `sys.dm_os_ring_buffers` .
10. Process:Private Bytes and SQL Server:Buffer Manager - Total pages (Target pages?)
    a. Most memory used by SQL Server goes into the buffer pool. However, there are allocations beyond the buffer pool known as *private bytes* that can cause memory pressure
    b. If you suspect this issue, compare Process:Private Bytes with SQL Server:Buffer Manager - Total pages (Target pages?)

III. Additional Memory Monitoring Tools

1. DBCC MEMORYSTATUS
   a. Returns two result sets, one showing allocations of memory and counts of occurrences, another showing different memory managers and the amount of memory they have consumed at the moment.
   b. If Memory Manager "Target Committed" (the system required memory) is higher than "Current Committed" (the memory currently provided), there may be buffer cache problems.
      i. Figure out which process is currently using the most memory using `sys.dm_os_performance_counters`
2. Dynamic Management Views
   a. Sys.dm_os_memory_brokers
      i. While most memory is allocated to the buffer cache, this DMV exposes other processes that also consume memory
   b. Sys.dm_os_memory_clerks
      i. If perfmon counter Process:Private Bytes is high, use this to determine how memory is being allocated
      ii. For in-memory OLTP storage, use `sys.dm_db_xtp_table_memory_stats` instead
   c. Sys.dm_os_ring_buffers
      i. Not documented in Books Online, so it's subject to change or removal
      ii. Outputs as XML, so XQuery may be needed to digest
      iii. A ring buffer is simply a recorded response to a notification. We can use it to see changes in memory usage
      iv. Main ring buffers associated with memory:
         1. RING_BUFFER_RESOURCE_MONITOR
            a. Records changes in memory allocation
            b. Useful for identifying external memory pressure
         2. RING_BUFFER_OOM (out of memory)
            a. Records out-of-memory issues
            b. Can see what kind of memory action failed

3. RING_BUFFER_MEMORY_BROKER
    a. Identifies internal memory pressure, which forces processes to release memory for the buffer
4. RING_BUFFER_BUFFER_POOL
    a. Records when the buffer pool itself is running out of memory
        i. This is a general indication of memory pressure

d. Sys.dm_db_xtp_table_memory_stats
    i. Shows memory used by tables and indexes created in-memory
    ii. Will need to query `sys.objects` to get the actual names of tables or indexes
e. Sys.dm_xtp_system_memory_consumers
    i. Shows system structures used to manage the internals of the memory engine (as opposed to the amount of data loaded into memory)
    ii. Look for the allocated and used bytes shown for each of the management structures

IV. Memory Bottleneck Resolutions

1. When there is high memory stress, indicated by a large number of hard page faults, this general flowchart can be used to resolve it:
    a. Check if Memory:Available Mbytes is low
        i. If yes, it's possible there are external memory problems. Troubleshoot in operating system.
    b. In `DBCC MEMORYSTATUS`, check if COMMITTED is above TARGET
        i. If yes, there is internal memory pressure. Identify large consumers using `sys.dm_os_memory_brokers`
    c. Check if Process:Private Bytes is high
        i. If yes, there is internal memory pressure other than the buffer. Identify with `sys.dm_os_memory_clerks`
    d. Finally, check for memory errors in the OS log and SQL Server log
2. Optimizing Application Workload
    a. Capture all SQL queries using Extended Events (Chapter 6) or Query Store (Chapter 11), then group the output on the `Reads` column
        i. Queries with the highest number of logical reads contribute most often to memory stress
    b. `sys.dm_exec_query_stats` can also be used for a quick and easy analysis, but since it's based on cache it may not be as accurate as EE
3. Allocating More Memory to SQL Server
    a. Hard page faults mean that the memory requirement of SQL Server is more than the max server memory value
    b. In this case, the max server memory configuration should be increased accordingly (assuming enough physical memory is available in the system)
    c. If using in-memory OLTP storage, the memory percentages allocated to the resource pools defined may need to be adjusted
4. Moving In-Memory Tables Back to Standard Storage
    a. Keep an eye on general query performance metrics for in-memory tables since not all tables or workloads benefit from them
    b. See Chapter 24 for more details
5. Increasing System Memory
    a. To identify which queries are using more memory, query `sys.dm_exec_query_memory_grants` to see I/O use
        i. Be careful running this query on systems already under memory stress
    b. If no queries stand out, the easiest and quickest resolution may be to simply increase system memory by purchasing and installing more
6. Changing from a 32-Bit to a 64-Bit Processor
    a. Since SQL Server 2012, a 32-bit instance of SQL Server is limited to accessing only 3GB of memory
        i. Because of this limitation, only small systems should be running 32-bit versions
    b. 64-bit processors can access up to 24TB depending on the version of the OS and the specific processor type
    c. SQL Server 2017 does not support the x86 chip set and must be run on a 64-bit processor
7. Compressing Data
    a. Less system memory is used because compressed data remains compressed in memory

b. There is a CPU cost, so keep an eye on that to be sure you're not just transferring stress

c. Depending on the nature of your data, there may not be much compression

8. Enabling 3GB of Process Address Space
   a. For 32-bit OS systems with 16GB of physical memory or less, less memory can be allocated to the OS and more to applications. This is done by specifying a /3GB switch in the `boot.ini` file
      i. Not applicable to SQL Server 2017+
9. Addressing Fragmentation
   a. While fragmentation may affect storage, it also can affect memory
      i. Fragmented pages have lots of empty space, all of which are retrieved from disk and into memory
   b. See Chapter 17 for details

- https://learning.oreilly.com/library/view/sql-server-2017/9781484238882/html/323849_5_En_5_Chapter.xhtml

D. Disk Performance Analysis

D. Disk Performance Analysis

Although memory and CPUs have become faster and faster, disks and the disk subsystem remains one of the slowest parts of any computing system. Additionally, I/O is affected by many factors such as:

- Virtual environments with shared disks
- Antivirus programs
- Filter drivers acting as bottlenecks in I/O paths

I. Disk Counters

1. PhysicalDisk - Disk Transfers/sec
   a. Although disc activity can be divided into LogicalDisk partitions, bottlenecks ultimately occur on the physical disk
   b. Note that RAID (redundant array of independent disks) and SAN (storage area network) systems are treated as a single physical disk
   c. Numbers are more in line with platter-style disk drives that are fast becoming obsolete
      i. Modern setups will use solid-state drives, SSD arrays, or iSCSI interfaces
   d. Disk Transfers/sec monitors the rate of read and write operations on the disk
      i. **A typical hard disk drive can do about 180 disk transfers per second for sequential I/O (IOPS) and 100 disk transfers per second for random I/O**
         1. Random I/O is lower because more disk arm and head movements are involved
         2. OLTP workloads rely heavily on random access for its singleton/small operations and thus is typically constrained more by disk transfers/sec than disk bytes/sec
         3. **For comparison, an SSD can be anywhere from around 5,000 IOPS to as much as 500,000 IOPS**
2. PhysicalDisk - Disk Bytes/Sec
   a. Monitors the rate at which bytes are transferred to or from the disk during read or write operations
      i. **A typical disk spinning at 7200RPM can transfer about 1000MB per second**
      ii. If the amount of data transfer exceeds capacity, there will be a backlog shown by the Disk Queue Length counters
      iii. This is usually a nonissue for OLTP applications, which access small amounts of data in individual database requests
3. PhysicalDisk - Avg. Disk Sec/Read and Avg. Disk Sec/Write
   a. These track the average amount of time it takes in milliseconds to read from or write to a disk
   b. **Brent Ozar recommends these be less than 100 ms, whereas Grant Fritchey recommends less than 10 ms**
   c. In terms of measuring performance of the I/O system, these are the single best measure
      i. They will not tell which query or queries are causing problems, but rather how the I/O system is behaving as a whole
4. SQLServer:Buffer Manager - Page reads/sec and Page writes/sec
   a. Knowing the pages being moved into and out of the buffer manager indicates whether the I/O (shown in Avg. Disk Sec/Read and Avg. Disk Sec/Write) is within SQL Server

i. This measure is needed to see the complete picture

II. Dynamic Management Objects (DMOs)

1. Sys.dm_io_virtual_file_stats
   a. Returns information about the files that make up a database
      i. Can see which file (log or data) is being pegged
   b. Called like so: `SELECT * FROM sys.dm_io_virtual_file_stats(DB_ID('AdventureWorks2017'), 2) AS divfs;`
   c. Stall data is the time that users are waiting on different I/O operations
      i. `io_stall_read_ms` represents the amount of time in milliseconds that users are waiting for reads
      ii. `io_stall_write_ms` shows the amount of time that write operations have had to wait on this file within the database
      iii. `io_stall` represents all waits on I/O for the file
   d. To put stall data in perspective, compare it with `sample_ms`, which shows the amount of time measured
   e. Once confirmed a file is pegged, further investigate using Paul Randal's wait statistics query and Perfmon metrics
2. Sys.dm_os_wait_stats
   a. Shows
      i. Aggregate information about waits on the system (total wait time)
      ii. A count of the waits that have occurred
      iii. A max value for these waits
         - Useful since it's possible that a single wait could have caused the majority of the wait time
   b. Queried like so: `SELECT * FROM sys.dm_os_wait_stats AS dows WHERE wait_type LIKE 'PAGEIOLATCH%';`
      i. I/O latch operations cause waits to occur and indicates a bottleneck in I/O
      ii. Aside from PAGEIOLATCH, other processes of interest are
         - WRITELOG
         - LOGBUFFER
         - ASYNC_IO_COMPLETION
   c. Values must be compared to a baseline
   d. Can also be seen in the Query Store (Chapter 11)

III. Disc Bottleneck Resolutions

1. Optimizing application workload
   a. Capture all SQL queries using Extended Events (Chapter 6) or Query Store (Chapter 11), then group the output on the `Reads` or `Writes` column
      i. Queries with the highest number of logical reads or writes will be the ones that cause a great deal of disk I/O
2. Using a faster I/O path
   a. Upgrade disk drives
3. Using a RAID array
   a. RAID 0: Striping with no fault tolerance
      i. The failure of any disk in the array will cause complete data loss in the disk subsystem. Therefore, this should not be used for any database files, except perhaps `tempdb`.
      ii. `I/Os per disk = (Reads + Writes) / Number of disks in the array`
   b. RAID 1: Mirroring
      i. Can be used where the complete data can be accommodated in one disk only
      ii. `master` and `msdb` are usually small enough to use RAID 1
      iii. `I/Os per disk = (Reads + 2 X Writes) / 2`
   c. RAID 5: Striping with parity
      i. An acceptable option in many cases. It provides reasonable fault tolerance by using only one extra disk to save the computed parity of the data in other disks

ii. `I/Os per disk = (Reads + 4 X Writes) / Number of disks in the array`
- Four I/Os are needed for each write request since read and write I/Os are needed for both the data and parity
  - Due to the magnification of write operations, use RAID 5 on read-only volumes or volumes with a low percentage of disk writes (10 percent or less)
- This means, where possible, transactional log files should not be placed on a RAID 5 array
- Data files should be fine since write operations are intermittent and batched together to improve efficiency

d. RAID 6: RAID 5 with an extra parity block
   i. Same performance as RAID 5 with a little overhead for writes

e. RAID 1+0 (RAID 10): Striping with mirroring
   i. Much more expensive than RAID 5
   ii. Should be used where a large volume is required to save data and more than 10 percent of disk requests are writes
   iii. Read performance is also very good since it supports *split seeks* (which distributes the read operations between disks)
   iv. Use RAID 1+0 wherever performance is critical
   v. `I/Os per disk = (Read + 2 X Writes) / Number of disks in the array`

4. Using (or removing) a SAN system
   a. Because of their size, complexity, and cost, SANs are not necessarily a good solution in all cases
      i. In some cases the solution may be going to local disks and getting rid of the SAN
   b. The principal strength of SAN systems is not reflected in performance but rather in the areas of scalability, availability, and maintenance
      i. Depending on the amount of data, direct-attached storage (DAS) can run faster
   c. SAN devices can also use Internet Small Computing System Interface (iSCSI) to connect a device to a network device, which acts as locally attached storage (and works nearly as fast)
5. Using solid-state drives
   a. These drives use memory instead of spinning disks to store information
   b. They’re quiet, lower power, and supremely fast, but also quite expensive
   c. You can also put SSDs into arrays through SAN or RAID, further increasing performance
   d. For a hardware-only solution, implementing SSDs is probably the best operation you can do for a system that is I/O bound
6. Aligning disks properly
   a. Windows Server 2016 aligns disks as part of the install process, so modern servers should not be running into this issue
   b. When moving volumes from a pre-Windows Server 2008 system, they need to be reformatted to get the alignment set appropriately
   c. Data on disk is stored as a series of *sectors* (or *blocks*), which in turn as stored on tracks. Partitions should be aligned so you’re storing the correct number of sectors for each track.
7. Adding system memory
   a. The less memory the system has, the more the disk subsystem is used
8. Creating multiple files and filegroups (for data files)
   a. By separating tables that are frequently joined into separate filegroups and then putting files within the filegroups on separate disks or LUNS, the separated I/O paths can result in improved performance
      i. Be aware that this can also lead to worse performance if the disks are not properly configured or get overloaded
         - Having more disks does not automatically mean more I/O
   b. If multiple filegroups are to be used, it is recommended that the primary filegroup be used only for system objects, and secondary filegroups be used only for user objects
      i. This approach improves the ability to recover from corruption
9. Moving the log files to a separate physical drive
   a. Transaction log files should always, when possible, be located on a separate hard disk drive from all other database files
      i. Transaction log activity primarily consists of sequential write I/O
         - Data files, on the other hand, leans more towards nonsequential (or random) I/O

ii. Having a single transaction log file on a dedicated hard disk can result in I/O performance improvements because the disk can concentrate on sequential I/O
- Multiple log files on a single disk brings you back to random I/O again
- The idea is having the physical disk spindle head move as little as possible
  - While this doesn't apply to SSD disk, distribution the work to multiple locations will still improve performance

b. An exception to this rule is for read-only databases, where no write operations are performed on the log file

c. As a general rule of them, try to isolate files with high I/O from other files with high I/O
   i. To identify these files, use `sys.dm_io_virtual_file_stats`

10. Using partitioned tables
    a. While partitioning is primarily a tool for making data management easier, it can increase speed because queries against well-defined partitions only access files with the partitions of interest
       i. Ex. If data is partitioned by month, months in the past can be set to read-only and compressed
    b. This should not be relied as the primary method of enhancing performance

E. CPU Performance Analysis

E. CPU Performance Analysis

The measures covered here are focused on the operating systems and SQL Server. In a virtualized environment, the measures we're looking at for CPU are much less likely to reflect reality. There may be external pressure or even external throttling, none of which will be visible with the counters outlined here. To understand how the measures are reflecting reality, investigation must be made on the hypervisor or virtualization system used.

Remember that CPU is affected by other resources since it's the thing managing those resources, so some situations that look like a CPU problem are better explained as a disk or memory issue (although networks are seldom a major bottleneck).

I. CPU Perfmon Counters

1. Processor(_Total)% - Processor Time
   a. **% Processor Time should not be consistently high, or greater than 80 percent (for Scott Whigham it's 75 percent). Any sustained time greater than 90 percent is effectively the same as 100 percent.**
   b. This counter should be used as a starting point for further investigation. For example, if there is excessive disk use shown in the I/O counters, a major part of the processor may be spent on managing the disk activities (can confirm by looking at the `% Privileged Time` counter of the processor). In that case, the disk bottleneck should be optimized first. If the disk bottleneck, in turn, is caused by a memory bottleneck, that should be the priority instead.
   c. You can track processor time as an aggregate of all the processors on the machine, or you can track the percent utilization individually to particular processors. Use the average value as just an indicator and the individual values as more of a measure of actual load and processing on the system.
   d. In a virtualized environment, the CPUs may be virtualized so what you're seeing isn't accurate. Ex:
      i. In a VMware system, if you install VMware Tools, you'll be able to look at a VM Processor counter to see the processor usage of the host machine. Using this measure you can tell whether the CPU usage you're seeing in your SO is reflected in the hosting machine or whether you're just maxing out your virtual CPUs
      ii. Running HyperV, you'd need to look to `\Hyper-V Hypervisor Logical Processor(_Total)\% Total Run Time` for the same measure
2. Processor(_Total)% - % Privileged
   a. *Privileged* (or *kernel*) mode (as opponed to *user mode*) reflects the time spent of system-level activities, including managing disk access and other external processes.
   b. **The average value is 5 to 10 percent. If 20 to 25 percent or more, there could be I/O issues (ex. defective components), a filter driver such as encryption services, or even out-of-date drivers.**
3. System - Processor Queue Length
   a. This is the number of threads ready to run (or requests outstanding) in the processor queue. There is only a single processor queue, even on computers with multiple processors.

b. **On systems with lower CPU utilization, this counter is typically 0 or 1**
    i. **In general, it shouldn't be more than two times the number of schedulers (usually 1:1 with processors)**
        1. **Assuming a system with 1 processor and 1 scheduler, a sustained counter of greater than 2 generally indicates processor congestion or bottleneck**
    ii. **PAL is fine with anything less than 10**
c. This is a more certain indicator of a busy processor than `% Processor Time`

4. System - Context Switches/sec
    a. This monitors the combined rate at which all processors are switched from one thread to another (a.k.a. the sum of `Thread:Context Switches/sec` for all threads running on all processors). Context switches occur when a thread:
        i. Voluntarily relinquishes the processor
        ii. Is preempted by a higher priority thread
        iii. Switches between user mode and privileged mode to use an executive or subsystem service
    b. **On average this should be < 5,000, but compare to the baseline since it depends on CPU speed**
5. SQL Server:SQL Statistics - Batch Requests/sec
    a. Measures SQL command batches received per second, which is a good indicator of how much load is being placed on the processor.
    b. **What's normal differs based on your standard workload, although 10,000 requests in a second would be definitely considered a busy system.**
6. SQL Server:SQL Statistics - SQL Compilations/sec
    a. Shows both the number of batch compiles and statement recompiles, which are expensive operations.
    b. Must compare with baseline to detect significant or sustained spikes in compilations
    c. Expect this number to be extremely high…
        i. When a server is first turned on, or after a failover or any other startup type event
        ii. If some type of object-relational mapping engine, such as nHibernate or Entity Framework is used
    d. See Chapter 14 on how to optimize this
7. SQL Server:SQL Statistics - SQL Recompilations/sec
    a. Measures the recompiles of both batches and statements, which can lead to processor stress
    b. **SQLwatch.io recommends that the ratio percentage of SQL Re-Compilations to SQL Compilations be less than 10%**
    c. See Chapter 17 for details on how to optimize this

II. Other Tools for Measuring CPU Performance

1. Dynamic Management Objects (DMOs)
    a. Sys.dm_os_wait_stats
        i. Metrics must be gathered over time to understand what's normal on your system. An increasing signal wait time can indicate CPU bottlenecks.
        ii. Starting with SQL Server 2016, CXPACKET waits have been split into CXPACKET and CXCONSUMER waits to differentiate between the producers vs consumers of parallelism, respectively. In these systems, CXPACKET is a wait that indicates real load on the system that affects CPU.
    b. Sys.dm_os_workers and Sys.dm_os_schedulers
        i. Worker and scheduler threads within the Windows operating system show the number of processes that are in a runnable state.
        ii. Running these regularly will indicate how processor load is changing.
2. Query Store
    a. The Query Store can be used to capture
        i. The aggregated CPU usage of the query in question
        ii. The wait statistics for queries, including any related to CPU

III. Processor Bottleneck Resolutions

1. Optimizing application workload
    a. First, capture processor-intensive queries via
        i. Extended Events (Chapter 6) with output grouped on the CPU column
        ii. `sys.query_store_runtime_stats` in the Query Store (Chapter 11), which shows multiple, aggregated, CPU metrics on a per-query basis
        iii. Querying the `sys.dm_exec_query_stats` or `sys.dm_exec_procedure_stats` DMV
    b. Next, analyze and optimize those queries
        i. Frequently, the cause for CPU stress is not extensive calculations but contention with logical I/O
        ii. Common queries and common execution plans can be tuned using a query hash and query plan hash (Chapter 14)
2. Eliminating or reducing excessive compiles/recompiles
    a. Methods for addressing recompiles are covered in Chapter 17
    b. If there are a lot more compiles compared to recompiles, it means few queries are being reused in the system. See Chapter 9 for tips on query reuse.
3. Using more or faster processors
    a. This is an easy but potentially costly solution.
    b. We can either
        i. Increase the power of individual processors
            1. Recommended when you have a high % Processor Time counter and low Processor Queue Length
        ii. Add more processors (which helps the system execute more requests simultaneously)
            1. Recommended when you have both a high % Processor Time counter and high Processor Queue Length
4. Not running unnecessary software
    a. Exterior applications that have nothing to do with maintaining the Windows server or SQL Server are best placed on a different machine

F. Network Bottleneck Analysis

F. Network Bottleneck Analysis

In OLTP environments, few performance issues are because of the network. Most of the network issues are in fact hardware or driver limitations or issues with switches or routers. Most of these can be best diagnosed with the Network Monitor tool. That said, there Performance Monitor does provide objects that collect data on network activity.

I. Network Perfmon Counters

1. Network Interface(Network card) - Bytes Total/sec & Network Interface\Current Bandwidth
    a. Bytes Total/sec shows how the network interface card (NIC) or network adapter is performing.
    b. Compare this value with that reported by the Network Interface\Current Bandwidth performance counter, which reflects each adapter's bandwidth
        i. **To allow headroom for spikes in traffic, you should usually average no more than 50 percent of capacity**
2. Network Segment - % Net Utilization
    a. Represents the percentage of network bandwidth in use on a network segment
    b. **In general, the average value should be < 80% of network bandwidth to keep room for load spikes**
        i. This is true even for dedicated full-duplex networks, which can account for near 100 percent network usage
        ii. **For Ethernet networks, 30 percent is the recommended threshold when SQL Server is on a shared network hub**
    c. Note, the Network Monitor Driver must be installed to collect performance data using the Network Segment object counters
        i. An alternative is looking at the wait statistics in `sys.dm_os_wait_stats` . This isn't ideal however since it often returns ASYNC_NETWORK_IO, which is more commonly associated with poor programming code that does not consume a result set efficiently.

II. Network Bottleneck Resolutions

1. Optimizing application workload
    a. Used stored procedures instead of sending a long SQL string over the network
        i. Multiple database requests can also be grouped into one stored procedure (important for Azure SQL Databases)
    b. Trim down the data set to only table columns that are used by the application
    c. Move data-intensive business logic into the database as stored procedures or database triggers to reduce network round-trips
    d. If data doesn't change frequently, try caching the information on the application
    e. Suppress things like row count using SET NOCOUNT ON at the top of queries
2. Adding network adapters
3. Moderating and avoiding interruptions

G. SQL Server Overall Performance

G. SQL Server Overall Performance

To analyze overall performance, besides examining hardware resource utilization, general aspects of SQL Server should also be examined.

I. Perfmon Counters for Generic SQL Pressure

1. SQLServer:Access Methods - Full Scans/sec
    a. Analyzes the possibility of missing indexes causing table scans or large data set retrievals
        i. Note that scans can also be caused when
            1. Too many rows are requested
            2. The predicate is not selective enough
            3. There are syntax errors (improper T-SQL)
            4. The data distribution or quantity doesn't support a seek
            5. Temporary tables are used in a stored procedure
        ii. Use Extended Events (Chapter 5) or the Query Store (Chapter 11) to identify these queries, which will have a large number of logical reads and an increased CPU time
    b. See also counters FreeSpace Scans/sec, Table Lock Escalations/sec, & Worktables Created/sec
2. SQLServer:Latches - Total Latch Wait Time (ms) & SQLServer:Locks(_Total) - Lock Timeouts/sec, Lock Wait Time (ms), & Number of Deadlocks/sec
    a. Analyzes the impact of database blocking (database concurrency) on the performance of SQL Server
        i. Total Latch Wait Time (ms) monitors latch requests that had to wait in the last second.
            1. Latches are used internally by SQL Server (not directly controlled by users) to protect the integrity of internal structures, such as a table row
            2. A high value here suggests SQL Server is spending too much time waiting on its internal synchronization mechanism
        ii. **Expect Lock Timeouts/sec to be 0 and Lock Wait Time (ms) to be very low**
            1. A nonzero value for Lock Timeouts/sec and a high value for Lock Wait Time (ms) indicate that excessive blocking is occurring in the database
            2. Remember that some degree of locks is a necessary part of the system. Establish a baseline to track thoroughly whether a given value is cause for concern.
        iii. **Number of Deadlocks/sec should also be 0**
            1. If nonzero, identify the victimized request and either resubmit the database request automatically or suggest that the user do so
            2. Troubleshoot and resolve the deadlock using techniques from Chapter 21
3. SQLServer:SQL Statistics - SQL Re-Compilations/sec
    a. Analyzes the number of stored procedures that are recompiling

i. Non-reusable execution plans stress the CPU because generating an execution plan for a stored procedure requires CPU cycles

ii. **This will never be 0 but should be as close to 0 as possible**

1. If there are consistent spikes:

a. Use Extended Events to identify the relevant stored procedures

b. Attempt to analyze and resolve the cause of recompilations (Chapter 17)

4. SQLServer:General Statistics - User Connections & SQLServer:SQL Statistics - Batch Requests/sec

a. Analyzes the volume of incoming requests and general behavior

i. User Connections is critical for ensuring the distribution of user connections in a load-balancing environment, where SQL Server is spread over several machines

1. This number can range all over the spectrum with normal application behavior. Compare with a baseline to determine expected behavior.

ii. Batch Requests/sec allow us to estimate the number of users SQL Server can take without developing resource bottlenecks

1. Helps understand SQL Server's relationship with the number of database connections
2. Helps understand SQL Server's relationship with Web Requests/sec (or Active Server Pages.Request/sec for web applications using IIS & ASP)
3. The value of this counter can range over a wide spectrum with normal application behavior. A normal baseline is essential to determine the expected behavior.

II. Dynamic Management Objects

1. sys.dm_db_missing_index details
   a. Part of a series of DMVs collectively referred to as the *missing indexes feature*, used to check for missing indexes based on execution plans stored in the cache. Since they can't be linked to a particular query, however, it's better off to use the XML execution plan for a given query (Chapter 5) or the Database Tuning Advisor (Chapter 10).
2. sys.dm_db_index_usage_stats
   a. Shows which indexes have been used since the last restart of the SQL Server instance. Not completely reliable since there are a number of ways that counters within this DMV get reset or removed.
   b. As an alternative, a lower-level DMV `sys.dm_db_index_operational_stats` can be used. It helps show where indexes are slowing down because of contention or I/O. More on this in Chapter 20.

III. Blocking/Concurrency Resolutions

1. Identify costly queries currently in cache using SQL Profiler or by querying `sys.dm_exec_query_stats`
   a. Optimize these queries appropriately
2. After optimizing costly queries, analyze and resolve blocking using techniques from Chapter 20
3. Enable the `blocked_process_report` blocking event (Chapter 20) in Extended Events (Chapter 6) to set a threshold to capture blocking information.

3. SQL Server Health Check Tools

Click here to expand...

A. Brent Ozar's First Responder Kit

What_to_Do_W… zar.pdf
01 Feb 2023, 01:01 AM

B. Paul Randal's Wait Statistics Query

```
-- Last updated October 1, 2021
WITH [Waits] AS
    (SELECT
        [wait_type],
        [wait_time_ms] / 1000.0 AS [WaitS],
        ([wait_time_ms] - [signal_wait_time_ms]) / 1000.0 AS [ResourceS],
        [signal_wait_time_ms] / 1000.0 AS [SignalS],
        [waiting_tasks_count] AS [WaitCount],
        100.0 * [wait_time_ms] / SUM ([wait_time_ms]) OVER() AS [Percentage],
        ROW_NUMBER() OVER(ORDER BY [wait_time_ms] DESC) AS [RowNum]
    FROM sys.dm_os_wait_stats
    WHERE [wait_type] NOT IN (
        -- These wait types are almost 100% never a problem and so they are
        -- filtered out to avoid them skewing the results. Click on the URL
        -- for more information.
        N'BROKER_EVENTHANDLER', -- https://www.sqlskills.com/help/waits/BROKER_EVENTHANDLER
        N'BROKER_RECEIVE_WAITFOR', -- https://www.sqlskills.com/help/waits/BROKER_RECEIVE_WAITFOR
        N'BROKER_TASK_STOP', -- https://www.sqlskills.com/help/waits/BROKER_TASK_STOP
        N'BROKER_TO_FLUSH', -- https://www.sqlskills.com/help/waits/BROKER_TO_FLUSH
        N'BROKER_TRANSMITTER', -- https://www.sqlskills.com/help/waits/BROKER_TRANSMITTER
        N'CHECKPOINT_QUEUE', -- https://www.sqlskills.com/help/waits/CHECKPOINT_QUEUE
        N'CHKPT', -- https://www.sqlskills.com/help/waits/CHKPT
        N'CLR_AUTO_EVENT', -- https://www.sqlskills.com/help/waits/CLR_AUTO_EVENT
        N'CLR_MANUAL_EVENT', -- https://www.sqlskills.com/help/waits/CLR_MANUAL_EVENT
        N'CLR_SEMAPHORE', -- https://www.sqlskills.com/help/waits/CLR_SEMAPHORE

        -- Maybe comment this out if you have parallelism issues
        N'CXCONSUMER', -- https://www.sqlskills.com/help/waits/CXCONSUMER

        -- Maybe comment these four out if you have mirroring issues
        N'DBMIRROR_DBM_EVENT', -- https://www.sqlskills.com/help/waits/DBMIRROR_DBM_EVENT
        N'DBMIRROR_EVENTS_QUEUE', -- https://www.sqlskills.com/help/waits/DBMIRROR_EVENTS_QUEUE
        N'DBMIRROR_WORKER_QUEUE', -- https://www.sqlskills.com/help/waits/DBMIRROR_WORKER_QUEUE
        N'DBMIRRORING_CMD', -- https://www.sqlskills.com/help/waits/DBMIRRORING_CMD
        N'DIRTY_PAGE_POLL', -- https://www.sqlskills.com/help/waits/DIRTY_PAGE_POLL
        N'DISPATCHER_QUEUE_SEMAPHORE', -- https://www.sqlskills.com/help/waits/DISPATCHER_QUEUE_SEMAPHORE
        N'EXECSYNC', -- https://www.sqlskills.com/help/waits/EXECSYNC
        N'FSAGENT', -- https://www.sqlskills.com/help/waits/FSAGENT
        N'FT_IFTS_SCHEDULER_IDLE_WAIT', -- https://www.sqlskills.com/help/waits/FT_IFTS_SCHEDULER_IDLE_WAIT
        N'FT_IFTSHC_MUTEX', -- https://www.sqlskills.com/help/waits/FT_IFTSHC_MUTEX

       -- Maybe comment these six out if you have AG issues
        N'HADR_CLUSAPI_CALL', -- https://www.sqlskills.com/help/waits/HADR_CLUSAPI_CALL
        N'HADR_FILESTREAM_IOMGR_IOCOMPLETION', -- https://www.sqlskills.com/help/waits/HADR_FILESTREAM_IOMG
        N'HADR_LOGCAPTURE_WAIT', -- https://www.sqlskills.com/help/waits/HADR_LOGCAPTURE_WAIT
        N'HADR_NOTIFICATION_DEQUEUE', -- https://www.sqlskills.com/help/waits/HADR_NOTIFICATION_DEQUEUE
        N'HADR_TIMER_TASK', -- https://www.sqlskills.com/help/waits/HADR_TIMER_TASK
        N'HADR_WORK_QUEUE', -- https://www.sqlskills.com/help/waits/HADR_WORK_QUEUE

        N'KSOURCE_WAKEUP', -- https://www.sqlskills.com/help/waits/KSOURCE_WAKEUP
        N'LAZYWRITER_SLEEP', -- https://www.sqlskills.com/help/waits/LAZYWRITER_SLEEP
        N'LOGMGR_QUEUE', -- https://www.sqlskills.com/help/waits/LOGMGR_QUEUE
        N'MEMORY_ALLOCATION_EXT', -- https://www.sqlskills.com/help/waits/MEMORY_ALLOCATION_EXT
        N'ONDEMAND_TASK_QUEUE', -- https://www.sqlskills.com/help/waits/ONDEMAND_TASK_QUEUE
        N'PARALLEL_REDO_DRAIN_WORKER', -- https://www.sqlskills.com/help/waits/PARALLEL_REDO_DRAIN_WORKER
        N'PARALLEL_REDO_LOG_CACHE', -- https://www.sqlskills.com/help/waits/PARALLEL_REDO_LOG_CACHE
```

```
        N'PARALLEL_REDO_TRAN_LIST', -- https://www.sqlskills.com/help/waits/PARALLEL_REDO_TRAN_LIST
        N'PARALLEL_REDO_WORKER_SYNC', -- https://www.sqlskills.com/help/waits/PARALLEL_REDO_WORKER_SYNC
        N'PARALLEL_REDO_WORKER_WAIT_WORK', -- https://www.sqlskills.com/help/waits/PARALLEL_REDO_WORKER_WAI
        N'PREEMPTIVE_OS_FLUSHFILEBUFFERS', -- https://www.sqlskills.com/help/waits/PREEMPTIVE_OS_FLUSHFILEB
        N'PREEMPTIVE_XE_GETTARGETSTATE', -- https://www.sqlskills.com/help/waits/PREEMPTIVE_XE_GETTARGETSTA
        N'PVS_PREALLOCATE', -- https://www.sqlskills.com/help/waits/PVS_PREALLOCATE
        N'PWAIT_ALL_COMPONENTS_INITIALIZED', -- https://www.sqlskills.com/help/waits/PWAIT_ALL_COMPONENTS_I
        N'PWAIT_DIRECTLOGCONSUMER_GETNEXT', -- https://www.sqlskills.com/help/waits/PWAIT_DIRECTLOGCONSUMER
        N'PWAIT_EXTENSIBILITY_CLEANUP_TASK', -- https://www.sqlskills.com/help/waits/PWAIT_EXTENSIBILITY_CL
        N'QDS_PERSIST_TASK_MAIN_LOOP_SLEEP', -- https://www.sqlskills.com/help/waits/QDS_PERSIST_TASK_MAIN_
        N'QDS_ASYNC_QUEUE', -- https://www.sqlskills.com/help/waits/QDS_ASYNC_QUEUE
        N'QDS_CLEANUP_STALE_QUERIES_TASK_MAIN_LOOP_SLEEP',
            -- https://www.sqlskills.com/help/waits/QDS_CLEANUP_STALE_QUERIES_TASK_MAIN_LOOP_SLEEP
        N'QDS_SHUTDOWN_QUEUE', -- https://www.sqlskills.com/help/waits/QDS_SHUTDOWN_QUEUE
        N'REDO_THREAD_PENDING_WORK', -- https://www.sqlskills.com/help/waits/REDO_THREAD_PENDING_WORK
        N'REQUEST_FOR_DEADLOCK_SEARCH', -- https://www.sqlskills.com/help/waits/REQUEST_FOR_DEADLOCK_SEARCH
        N'RESOURCE_QUEUE', -- https://www.sqlskills.com/help/waits/RESOURCE_QUEUE
        N'SERVER_IDLE_CHECK', -- https://www.sqlskills.com/help/waits/SERVER_IDLE_CHECK
        N'SLEEP_BPOOL_FLUSH', -- https://www.sqlskills.com/help/waits/SLEEP_BPOOL_FLUSH
        N'SLEEP_DBSTARTUP', -- https://www.sqlskills.com/help/waits/SLEEP_DBSTARTUP
        N'SLEEP_DCOMSTARTUP', -- https://www.sqlskills.com/help/waits/SLEEP_DCOMSTARTUP
        N'SLEEP_MASTERDBREADY', -- https://www.sqlskills.com/help/waits/SLEEP_MASTERDBREADY
        N'SLEEP_MASTERMDREADY', -- https://www.sqlskills.com/help/waits/SLEEP_MASTERMDREADY
        N'SLEEP_MASTERUPGRADED', -- https://www.sqlskills.com/help/waits/SLEEP_MASTERUPGRADED
        N'SLEEP_MSDBSTARTUP', -- https://www.sqlskills.com/help/waits/SLEEP_MSDBSTARTUP
        N'SLEEP_SYSTEMTASK', -- https://www.sqlskills.com/help/waits/SLEEP_SYSTEMTASK
        N'SLEEP_TASK', -- https://www.sqlskills.com/help/waits/SLEEP_TASK
        N'SLEEP_TEMPDBSTARTUP', -- https://www.sqlskills.com/help/waits/SLEEP_TEMPDBSTARTUP
        N'SNI_HTTP_ACCEPT', -- https://www.sqlskills.com/help/waits/SNI_HTTP_ACCEPT
        N'SOS_WORK_DISPATCHER', -- https://www.sqlskills.com/help/waits/SOS_WORK_DISPATCHER
        N'SP_SERVER_DIAGNOSTICS_SLEEP', -- https://www.sqlskills.com/help/waits/SP_SERVER_DIAGNOSTICS_SLEEP
        N'SQLTRACE_BUFFER_FLUSH', -- https://www.sqlskills.com/help/waits/SQLTRACE_BUFFER_FLUSH
        N'SQLTRACE_INCREMENTAL_FLUSH_SLEEP', -- https://www.sqlskills.com/help/waits/SQLTRACE_INCREMENTAL_F
        N'SQLTRACE_WAIT_ENTRIES', -- https://www.sqlskills.com/help/waits/SQLTRACE_WAIT_ENTRIES
        N'VDI_CLIENT_OTHER', -- https://www.sqlskills.com/help/waits/VDI_CLIENT_OTHER
        N'WAIT_FOR_RESULTS', -- https://www.sqlskills.com/help/waits/WAIT_FOR_RESULTS
        N'WAITFOR', -- https://www.sqlskills.com/help/waits/WAITFOR
        N'WAITFOR_TASKSHUTDOWN', -- https://www.sqlskills.com/help/waits/WAITFOR_TASKSHUTDOWN
        N'WAIT_XTP_RECOVERY', -- https://www.sqlskills.com/help/waits/WAIT_XTP_RECOVERY
        N'WAIT_XTP_HOST_WAIT', -- https://www.sqlskills.com/help/waits/WAIT_XTP_HOST_WAIT
        N'WAIT_XTP_OFFLINE_CKPT_NEW_LOG', -- https://www.sqlskills.com/help/waits/WAIT_XTP_OFFLINE_CKPT_NEW
        N'WAIT_XTP_CKPT_CLOSE', -- https://www.sqlskills.com/help/waits/WAIT_XTP_CKPT_CLOSE
        N'XE_DISPATCHER_JOIN', -- https://www.sqlskills.com/help/waits/XE_DISPATCHER_JOIN
        N'XE_DISPATCHER_WAIT', -- https://www.sqlskills.com/help/waits/XE_DISPATCHER_WAIT
        N'XE_TIMER_EVENT' -- https://www.sqlskills.com/help/waits/XE_TIMER_EVENT
        )
    AND [waiting_tasks_count] > 0
    )
SELECT
    MAX ([W1].[wait_type]) AS [WaitType],
    CAST (MAX ([W1].[WaitS]) AS DECIMAL (16,2)) AS [Wait_S],
    CAST (MAX ([W1].[ResourceS]) AS DECIMAL (16,2)) AS [Resource_S],
    CAST (MAX ([W1].[SignalS]) AS DECIMAL (16,2)) AS [Signal_S],
    MAX ([W1].[WaitCount]) AS [WaitCount],
    CAST (MAX ([W1].[Percentage]) AS DECIMAL (5,2)) AS [Percentage],
    CAST ((MAX ([W1].[WaitS]) / MAX ([W1].[WaitCount])) AS DECIMAL (16,4)) AS [AvgWait_S],
    CAST ((MAX ([W1].[ResourceS]) / MAX ([W1].[WaitCount])) AS DECIMAL (16,4)) AS [AvgRes_S],
    CAST ((MAX ([W1].[SignalS]) / MAX ([W1].[WaitCount])) AS DECIMAL (16,4)) AS [AvgSig_S],
```

```
    CAST ('https://www.sqlskills.com/help/waits/' + MAX ([W1].[wait_type]) as XML) AS [Help/Info URL]
FROM [Waits] AS [W1]
INNER JOIN [Waits] AS [W2] ON [W2].[RowNum] <= [W1].[RowNum]
GROUP BY [W1].[RowNum]
HAVING SUM ([W2].[Percentage]) - MAX( [W1].[Percentage] ) < 95; -- percentage threshold
GO
```

4. In-depth Query Tuning

T-SQL Querying