Module 1

Architecture, scheduling, and wait statistics

Module Overview

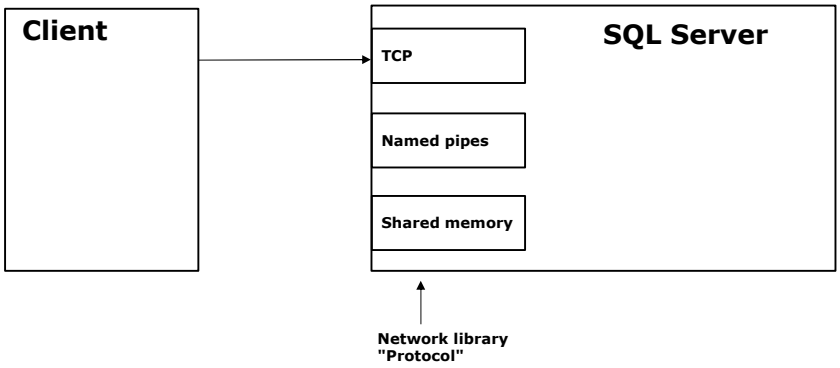
- SQL Server components and SQLOS
- SQL Server scheduling
- Wait statistics

Lesson: SQL Server components and SQLOS

- Connection protocols
- Database engine
- SQLOS
- Multiple CPUs—SMP and NUMA
- The query lifecycle
- Sources for (performance) information and monitoring

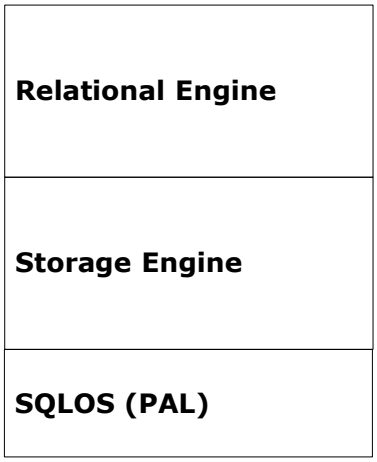
Connection protocols

- Network protocols, also known as Network Libraries (Netlib)
  - Shared memory
  - Named Pipes
  - TCP/IP
- Packets submitted through the Netlibs uses the Tabular Data Stream (TDS) protocol



Network Protocols and Network Libraries
https://docs.microsoft.com/en-us/sql/sql-server/install/network-protocols-and-network-libraries

Database engine



- Relational engine
  - Parses and optimizes the queries
  - Manages the caching and execution of execution plans
- Storage engine
  - Manages buffer pages
  - I/O to the physical files
  - Transactions
  - Locking
- SQLOS
  - Abstraction layer over common operating system functions, providing task and memory management
  - Also known as Platform Abstraction Layer

SQL Server Architecture Explained: Named Pipes, Optimizer, Buffer Manager
https://www.guru99.com/sql-server-architecture.html

SQLOS

- SQL Server requirements for low-level resource (memory, schedulers, synchronization objects, and so on) are complex
- Many services inside the engine need access to these low-level resources
- Components to provide this access are grouped together into a single functional unit called the SQLOS
- SQL Server components make calls to the SQLOS
- SQLOS provides many functions, including memory management, scheduling, I/O management, locking framework, transaction management, deadlock detection, and exception handling

SQL SERVER – What is SQL Server Operating System?
https://blog.sqlauthority.com/2015/11/11/sql-server-what-is-sql-server-operating-system/

The query lifecycle

- SQL Server Network Interface
  - Receive query from client
- Relational engine
  - Parse query and bind to database objects
  - Compile query plan
  - Execute query plan
  - Generate results
- Storage Engine
  - Used by relational engine to access data and metadata
- SQLOS
  - Provides low-level functions to storage and relational engines

Query Processing Architecture Guide
https://docs.microsoft.com/en-us/sql/relational-databases/query-processing-architecture-guide

Sources for (performance) information and monitoring

- Dynamic Management Views, DMVs
  - Will be used throughout the course
  - Examples
    - sys.dm_**exec**_sessions
    - sys.dm_**os**_schedulers
    - sys.dm_**tran**_locks
    - sys.dm_**io**_virtual_file_stats
    - sys.dm_**db_index**_physical_stats
- Extended Events
  - Discussed in a later module
- Performance Monitor
  - Discussed in a later module
- Activity Monitor

System Dynamic Management Views
https://docs.microsoft.com/en-us/sql/relational-databases/system-dynamic-management-views/system-dynamic-management-views

Glenn Berry's SQL Server Diagnostic Queries
https://glennsqlperformance.com/resources/

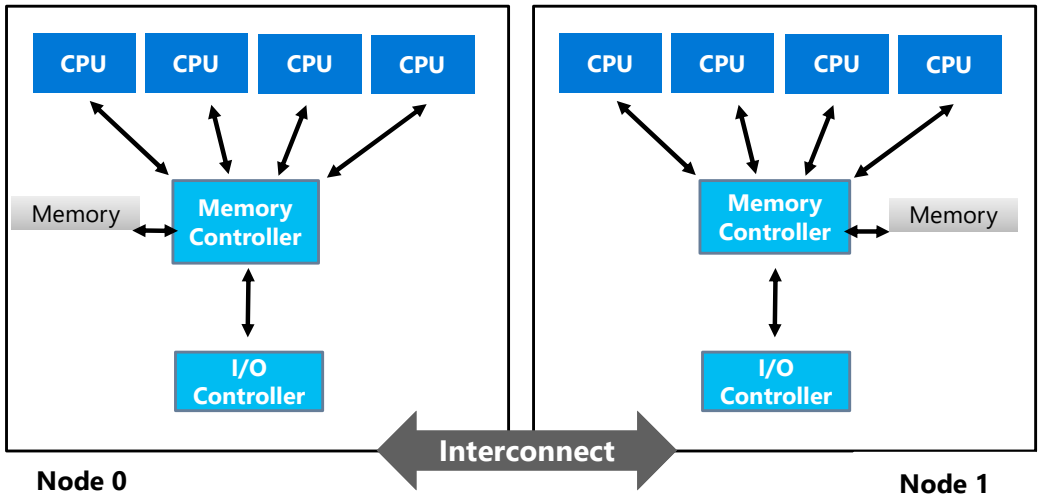
Activity Monitor
https://docs.microsoft.com/en-us/sql/relational-databases/performance-monitor/activity-monitor

Explaining Activity Monitor
https://sqlblog.karaszi.com/explaining-activity-monitor/

Lesson: Windows scheduling vs. SQL Server scheduling

- Preemptive vs. non-preemptive scheduling
- SQL Server on OS Scheduler and User Mode Scheduler
- SQL Server execution model
- User request lifecycle

Multiple CPUs—SMP and NUMA



Logical diagram of a two-node NUMA system

Demo Parallel

Parallelism in SQL Server Execution Plan
https://www.mssqltips.com/sqlservertip/5404/parallelism-in-sql-server-execution-plan/

I SQL Server används flera CPU
eller kärnor genom två huvudsakliga arkitekturer: **SMP (Symmetric Multi-Processing)** och **NUMA (Non-Uniform Memory Access)**.**SMP:**
I en **SMP-arkitektur** delar alla processorer ett gemensamt minne och har lika tillgång till resurser. SQL Server använder **parallellism** för att distribuera arbetsbelastningen över flera CPU-kärnor, vilket gör att flera frågor kan köras parallellt. Detta är bra för allmänt hög prestanda, men det kan uppstå **minnesbandbreddsbegränsningar** när CPU
konkurrerar om gemensamma minnesresurser.**NUMA:**
NUMA är en mer avancerad arkitektur där CPU-kärnor är organiserade i **noder**, och varje nod har sitt eget dedikerade minne. SQL Server drar nytta av detta genom att optimera hur den hanterar processer och minne, vilket minskar latens och ökar prestanda. I NUMA tilldelas CPU i en nod specifika arbetsbelastningar, vilket minimerar behovet av att få tillgång till minne från andra noder, något som skulle kunna leda till flaskhalsar.SQL Server använder **batch-mode processing** och **parallellism** effektivt för att maximera användningen av flera CPU
och kärnor i både SMP och NUMA.

CPU related configuration settings

- Processor affinity mask
  - Which CPU cores can we use (sets a schedule online or offline)
  - Most frequently set to 0, meaning all CPUs
- Cost threshold for parallelism
  - The breaking point for when SQL Server will possibly let the query go parallel.
  - Based on "hypothetical seconds" (seconds on a machine from ca 1998)
  - The default 5 is generally considered a way to low value
    - Increase to for instance 50 or 100
    - Total cost for query execution is higher when going parallel because of synchronization of the threads
    - For instance on a laptop from 2020, 5 corresponds to 0.082 seconds.
- Max degree of parallelism
  - Can be set at server, database and query level
  - Recommended config:
    - Same as number of cores up to 8.
    - If more than 1 numa node:
      - If up to 16 logical CPU per node then same as logical CPUs per node
      - If more then 16 logical CPUs per node then half the number of CPUs per node

Processor Affinity Mask:
Processor affinity mask i SQL Server kontrollerar vilka specifika CPU-kärnor SQL Server-tjänsten kan använda. Detta gör att du kan binda SQL Server-processer till specifika processorer, vilket kan optimera prestanda och förhindra att SQL Server konkurrerar med andra applikationer om samma CPU-resurser. Genom att använda processor affinity kan du också undvika problem som CPU-thrashing och förbättra processor-cache-prestandan genom att hålla processer på samma CPU-kärnor.

Cost Threshold for Parallelism:
Cost Threshold for Parallelism definierar den kostnadsnivå (i SQL Server optimeringsenheter) som krävs för att en fråga ska köras parallellt. Standardvärdet är 5, vilket innebär att alla frågor med en beräknad kostnad över 5 kommer att köras parallellt. Genom att öka detta värde kan du begränsa parallell exekvering till mer resurskrävande frågor och undvika överanvändning av parallellism för enklare frågor som kör snabbare i en sekventiell plan.

Max Degree of Parallelism (MAXDOP):
Max Degree of Parallelism (MAXDOP) anger det maximala antalet processorer som SQL Server kan använda för att köra en enskild fråga parallellt. Standardvärdet är 0, vilket tillåter SQL Server att använda alla tillgängliga processorer. Genom att justera MAXDOP kan du kontrollera hur mycket parallellism SQL Server ska använda, vilket kan optimera prestanda och förhindra överbelastning vid frågor som kräver mycket processorkraft.

Configure the max degree of parallelism Server Configuration Option
https://docs.microsoft.com/en-us/sql/database-engine/configure-windows/configure-the-max-degree-of-parallelism-server-configuration-option

Understanding and Using Parallelism in SQL Server
https://www.red-gate.com/simple-talk/databases/sql-server/learn/understanding-and-using-parallelism-in-sql-server/

Configure the cost threshold for parallelism Server Configuration Option
https://docs.microsoft.com/en-us/sql/database-engine/configure-windows/configure-the-cost-threshold-for-parallelism-server-configuration-option

Preemptive vs. non-preemptive scheduling

- Each CPU core can run one task at a time
- Preemptive scheduling
  - Is driven by the view of prioritized computation
  - Means that a low-priority process is pre-empted out of the processor by a high-priority process
- Non-preemptive scheduling
  - Means that the priority of a process doesn't matter
  - Is where a process is not preempted, and executes until it explicitly yields the processor
- Windows uses preemptive scheduling
- SQL Server uses non-preemptive scheduling

Understanding SQL Server Schedulers, Workers and Tasks
https://www.mssqltips.com/sqlservertip/4403/understanding-sql-server-schedulers-workers-and-tasks/

CPU Scheduling Basics – Windows and SQL Server
https://www.nocentino.com/posts/2016-05-26-cpu-scheduling-basics-windows-and-sql-server/

SQL SERVER : SQL Server Scheduler
https://www.sqlservercentral.com/blogs/sql-server-sql-server-scheduler

SQL Server on OS Scheduler and User Mode Scheduler

- Different names in different versions of SQL Server:
  - UMS in SQL Server 7.0 and 2000
  - SOS in SQL Server 2005 and later
- Enables non-preemptive scheduling
- Maintains five lists to facilitate scheduling:
  - Worker list
  - Runnable list
  - Waiter list
  - I/O list
  - Timer list

I SQL Server finns olika system för **scheduling** beroende på version.

1. **UMS (User Mode Scheduling)** användes i **SQL Server 7.0 och 2000**. Det ansvarade för att hantera arbetsbelastningen utan att behöva förlita sig på operativsystemets processhantering.

2. **SOS (SQL Operating System)** introducerades i **SQL Server 2005 och senare**. Det är en mer avancerad planeringsmotor för att hantera uppgifter i SQL Server på ett icke-preemptivt sätt, vilket innebär att SQL Server själv avgör när en tråd ska stoppas eller bytas, istället för att operativsystemet gör det.

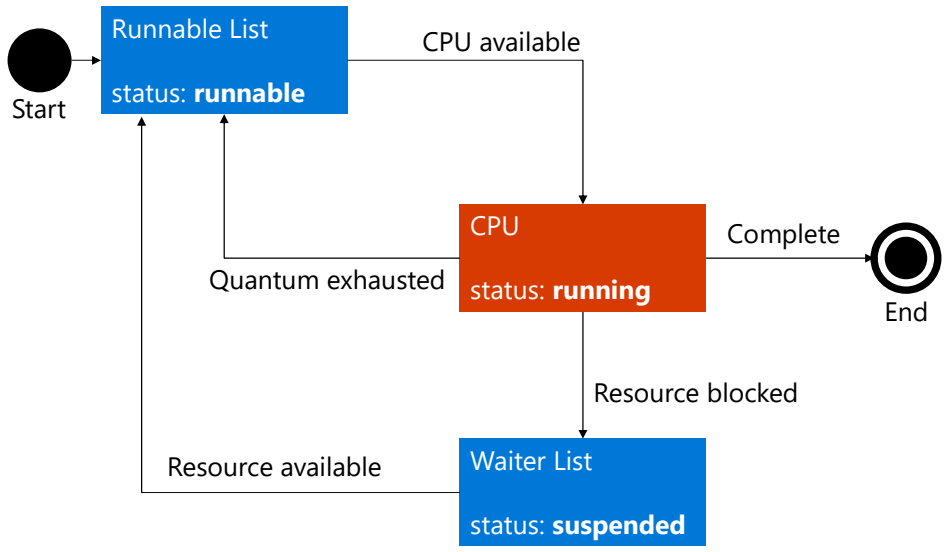
Fem schemaläggningslistor som används:

- **Worker list**: Håller en lista över tillgängliga arbetare (trådar).
- **Runnable list**: Innehåller de arbetare som är redo att köras men väntar på tillgång till CPU.
- **Waiter list**: Trådar som väntar på resurser, t.ex. en låsning eller I/O-operation.
- **I/O list**: Arbetare som väntar på slutförandet av en I/O-operation.
- **Timer list**: Innehåller arbetare som väntar på att en timer ska gå ut, ofta relaterat till tidsstyrda uppgifter.

Non-preemptive Scheduling:
SQL Server använder **icke-preemptiv schemaläggning**, vilket betyder att den egna schemaläggaren (SOS/UMS) har kontroll över när en uppgift får processortid, vilket undviker avbrott från operativsystemet.

Detta system möjliggör finare kontroll över CPU-resurser och effektiv multitasking inuti SQL Server.

SQL Server execution model



I SQL Server-exekveringsmodellen är ett **quantum** den tidsperiod under vilken en arbetare (worker thread) får processortid innan SQL Server schemaläggaren bestämmer om tråden ska fortsätta eller pausas till förmån för en annan. Ett quantum varar vanligtvis 4 millisekunder. Om en tråd inte är färdig när dess quantum tar slut, sätts den på **runnable list** tills det är dess tur igen. Detta bidrar till en rättvis fördelning av CPU-resurser och säkerställer att ingen tråd monopoliserar processortid.

SQL SERVER – Life of a SQL Query – Query States
https://blog.sqlauthority.com/2014/04/17/sql-server-life-of-a-sql-query-query-states/

SQL server DB status runnable, sleeping, suspended, running, pending and background meanings
http://sqlserversavari.blogspot.com/2019/10/sql-server-db-status-runnable-sleeping.html

In SQL Server, queries jump between three states: RUNNING (actively processing), RUNNABLE (queued for CPU), and SUSPENDED (waiting on resources like disk I/O). Imagine a grocery store checkout: RUNNING is your item being scanned, RUNNABLE is waiting in line, and SUSPENDED is like being paused for a price check. Wait types give detailed reasons for suspensions. It's crucial for database admins to monitor these states and wait types to troubleshoot and improve query efficiency. Tools like SQL Server Management Studio and SolarWinds Database Performance Analyzer help track and analyze these metrics, aiding in optimizing database performance by minimizing suspensions and ensuring queries run smoothly.

User Request Life Cycle

- Connection established: **sys.dm_exec_connections**
- Session ID assigned: **sys.dm_exec_sessions**
- Request created: **sys.dm_exec_requests**
- Task (s) created: **sys.dm_os_tasks**
- Task assigned to worker: **sys.dm_os_workers**
- Worker runs on operating system thread: **sys.dm_os_threads**
- Scheduler manages task activity: **sys.dm_os_schedulers**

Demo Monitoring scheduling

Thread and Task Architecture Guide
https://docs.microsoft.com/en-us/sql/relational-databases/thread-and-task-architecture-guide

Absolutely, Robert. Here's a breakdown of the SQL Server components you're asking about, distilled into a neat package:

1. **Connection established (`sys.dm_exec_connections`)**: This is where it all begins. A connection is made to SQL Server, kind of like dialing into a radio station. This DMV gives you info on each active connection to the server.

2. **Session ID assigned (`sys.dm_exec_sessions`)**: Once connected, you're given a session ID, sort of a "pass" into the concert. It represents an interactive SQL Server session, and this DMV shows each session's details.

3. **Request created (`sys.dm_exec_requests`)**: Now that you're in, you make a request, like asking the DJ for a song. This DMV shows info on each request running or waiting to run in SQL Server.

4. **Tasks created (`sys.dm_os_tasks`)**: Requests get broken down into tasks, similar to how a song is composed of notes. This DMV details tasks SQL Server is handling, whether active or waiting.

5. **Task assigned to worker (`sys.dm_os_workers`)**: Each task gets a worker, akin to assigning each note to an instrument. This DMV shows the workers (or threads) executing tasks.

6. **Worker runs on operating system thread (`sys.dm_os_threads`)**: Workers operate on OS threads, the individual musicians playing the notes. This DMV provides a view into these threads.

7. **Scheduler manages task activity (`sys.dm_os_schedulers`)**: Finally, the scheduler acts as the conductor, ensuring tasks (notes) are played at the right time and order. This DMV shows how SQL Server schedules and manages tasks.

Each step, from establishing a connection to task management by the scheduler, layers complexity and functionality, ensuring efficient operation and management of SQL Server processes.

Lesson: Waits and queues

- Overview of waits and queues
- Viewing wait statistics
- LCK_M_* wait types
- PAGELATCH_* wait types
- PAGEIOLATCH_* wait types
- CXPACKET and CXCONSUMER wait types
- WRITELOG wait type
- Other common wait types

Overview of waits and queues

- Waits
  - A request or task is said to “wait” if a required resource is not available
  - SQL Server tracks the resources being waited for as wait type
    - Resource waits—**Suspended** tasks wait for data pages
    - Signal waits—**Runnable** tasks wait for CPU time
- Queues
  - System resources and utilization, which are measured using Performance Monitor performance counters, DMVs, and other tools

I SQL Server, en **wait** inträffar när en uppgift behöver en resurs som inte är tillgänglig, vilket gör att uppgiften måste vänta. SQL Server registrerar dessa väntetider med olika **wait types**:

- **Resource waits**: Uppstår när en uppgift väntar på en resurs som data från en databas, t.ex. data som inte finns i minnet.
- **Signal waits**: Sker när uppgiften är redo att köras men väntar på CPU-tid.

Queues i SQL Server hanterar systemresurser och utnyttjandet av dessa kan övervakas med verktyg som **Performance Monitor** och **DMVs** (Dynamic Management Views).

SQL Server 2005 Waits and Queues – Microsoft
https://download.microsoft.com/download/4/7/a/47a548b9-249e-484c-abd7-29f31282b04d/Performance_Tuning_Waits_Queues.doc

In Review: SQL Server 2005 Waits and Queues
https://www.brentozar.com/archive/2015/10/in-review-sql-server-2005-waits-and-queues-2/

Viewing wait statistics

- When a thread waits on a resource, SQL Server tracks wait information
  - Aggregated wait statistics for all wait types since startup of instance
    - **sys.dm_os_wait_stats**
  - Current wait statistics
    - **sys.dm_os_waiting_tasks**
  - Aggregated wait statistics for active sessions
    - **sys.dm_exec_session_wait_stats**
    - Introduced in 2016
  - The actual execution plan

Demo Wait stats views

Boost SQL Server Performance with Wait Statistics
https://www.sqlshack.com/boost-sql-server-performance-with-wait-statistics/

SQL Server Wait Statistics (or please tell me where it hurts…)
https://www.sqlskills.com/blogs/paul/wait-statistics-or-please-tell-me-where-it-hurts/

sys.dm_os_wait_stats (Transact-SQL)
https://docs.microsoft.com/en-us/sql/relational-databases/system-dynamic-management-views/sys-dm-os-wait-stats-transact-sql

sys.dm_os_waiting_tasks (Transact-SQL)
https://docs.microsoft.com/en-us/sql/relational-databases/system-dynamic-management-views/sys-dm-os-waiting-tasks-transact-sql

sys.dm_exec_session_wait_stats (Transact-SQL)
https://docs.microsoft.com/en-us/sql/relational-databases/system-dynamic-management-views/sys-dm-exec-session-wait-stats-transact-sql

Troubleshooting SQL Server Wait Stats
https://www.sentryone.com/white-papers/troubleshooting-sql-server-wait-stats

Using Wait Stats to Find Why SQL Server is Slow
https://www.brentozar.com/sql/wait-stats/

Get Detailed Wait Stats with SQL Server Execution Plan

https://www.mssqltips.com/sqlservertip/5081/get-detailed-wait-stats-with-sql-server-execution-plan/

LCK_M_* wait types

- A thread waits on an incompatible lock
- Possible causes include:
  - A large update or table scan causing lock escalation
  - Unnecessary shared locks on data being accessed
- Possible solutions include:
  - Locking may or may not be a root cause:
  - Use **sys.dm_os_waiting_tasks** to find out the wait type that the lead blocker is waiting on
  - Find out which queries are waiting too long for locks using the blocked process report
  - Consider using a different isolation level
  - Review indexing strategy and optimize queries

I SQL Server representerar **LCK_M_*** väntetyper låsrelaterade väntetider. De uppstår när en uppgift väntar på att få tillgång till en resurs som är låst av en annan transaktion. Dessa väntetyper används för att spåra olika typer av lås:

- **LCK_M_S**: Väntar på ett delat (S) lås.
- **LCK_M_X**: Väntar på ett exklusivt (X) lås.
- **LCK_M_U**: Väntar på ett uppgraderbart (U) lås.

Dessa väntetider indikerar att SQL Server hanterar resurstillgång genom lås för att förhindra datakonflikter, vilket kan påverka prestanda vid långa väntetider.

SQL Server Wait Types
https://www.sqlskills.com/help/waits/#l

PAGELATCH_* wait types

- Occurs when a task waits to latch, (spärra) a page in the buffer pool
- Possible causes include:
  - Contention for “hot” pages
  - Contention for file allocation pages
- Possible solutions include:
  - Use a new indexing strategy or partitioning
  - Use fewer short-lived objects, or add data files

PAGELATCH_* väntetyper i SQL Server uppstår när SQL Server väntar på en intern latching-mekanism för att säkra tillgång till sidor i **buffertpoolen** (dvs. sidor i minnet, inte på disken). Detta skiljer sig från I/O-väntetider.

Vanliga exempel inkluderar:

- **PAGELATCH_SH**: Väntar på ett delat latch på en sida.
- **PAGELATCH_EX**: Väntar på ett exklusivt latch.

Dessa väntetider kan orsakas av hög konkurrens om samma sidor i minnet, ofta på grund av tunga insättningar eller uppdateringar i specifika tabeller.

Lås och **latchar** är båda mekanismer för att hantera resurskonflikter i SQL Server, men de används på olika nivåer:

- **Lås** skyddar databasobjekt (t.ex. tabeller, rader) mellan transaktioner för att säkerställa dataintegritet. De används under en längre tid för att förhindra konflikter mellan samtidiga användare som uppdaterar data.

- **Latchar** är lätta interna skyddsmekanismer som används av SQL Server för att säkra tillgång till minnessidor (t.ex. buffertpoolen) under mycket korta perioder. Latchar används för att säkerställa korrekt åtkomst på minnesnivå, snarare än transaktionsnivå.

SQL Server Wait Types
https://www.sqlskills.com/help/waits/#p

PAGEIOLATCH_* wait types

- A task waits for a data page to be retrieved from disk to memory
- Possible causes include:
  - I/O subsystem is slow
  - Poor query performance
- Possible solutions include:
  - Replace I/O subsystem
  - Review and update indexes and statistics
  - Identify queries with parallel scans and implicit conversions in the query plan
  - Reduce page splits

PAGEIOLATCH_* väntetyper uppstår när SQL Server väntar på att en sidläsning från **disken** ska slutföras och laddas in i minnet (buffertpoolen). Dessa väntetyper indikerar att en fråga väntar på I/O-operationer och kan vara en indikator på I/O-flaskhalsar eller långsamma diskar.

Exempel:
- **PAGEIOLATCH_SH**: Väntar på att läsa en sida för ett delat latch (läsåtkomst).
- **PAGEIOLATCH_EX**: Väntar på en exklusiv latch (skrivåtkomst).

Långa PAGEIOLATCH-väntetider kan indikera att systemet har I/O-prestandaproblem, t.ex. långsam disk eller överbelastad lagring.

SQL Server Wait Types
https://www.sqlskills.com/help/waits/#p

CXPACKET and CXCONSUMER wait types

- A task participating in a parallelized request completes before other tasks in the request
  - CXPACKET – producer is slow producing rows
  - CXCONSUMER – consumer is waiting for rows. Can generally be ignored.
- Possible causes include:
  - Bad query plan selection
  - Uneven distribution of work among tasks
  - Some parallel tasks slowed by other wait types
- Possible solutions include:
  - Update statistics
  - Resolve underlying problems
  - Reduce MAXDOP
  - Raise cost threshold for parallelism

CXPACKET och **CXCONSUMER** är båda väntetyper relaterade till parallellism i SQL Server.

CXPACKET (Class Exchange Packet):
Detta är en vanlig väntetyp som uppstår när en parallell fråga väntar på att en annan tråd ska slutföra sitt arbete. Den uppstår när trådar inte avslutar parallell bearbetning samtidigt, och SQL Server väntar på att alla trådar ska slutföra sitt arbete innan resultatet kan bearbetas. För mycket parallellism kan orsaka överdrivna CXPACKET-väntetider.

CXCONSUMER:
Introducerad i senare versioner av SQL Server, **CXCONSUMER** representerar tiden som konsumenttrådar väntar på att bearbeta data från producenttrådar i parallellfrågor. Den är mindre oroande än CXPACKET och fungerar som en mer intern del av parallellbearbetning.

För att optimera parallellism och hantera dessa väntetider kan du justera inställningar som **Cost Threshold for Parallelism** och **Max Degree of Parallelism (MAXDOP)**.

SQL Server Wait Types
https://www.sqlskills.com/help/waits/cxpacket/
https://www.sqlskills.com/help/waits/cxconsumer/

WRITELOG wait type

- A task is waiting for a transaction log buffer to be flushed to disk
- Possible causes include:
  - Poor log disk I/O
  - Many small transactions
  - Unnecessary indexes
  - Frequent page splits
- Possible solutions include:
  - Move the log to faster storage
  - Use fewer, larger transactions
  - Review indexing

SQL Server Wait Types
https://www.sqlskills.com/help/waits/writelog/

Other common wait types

- SOS_SCHEDULER_YIELD
  - Waiting for CPU
- THREADPOOL
  - Waiting for worker thread
- ASYNC_NETWORK_IO
  - Waiting for client to consume data
- RESOURCE_SEMAPHORE
  - Waiting for a memory grant
- CMEMTHREAD
  - Add or remove execution plan to cache
  - Possible indication of plan cache bloat

SOS_SCHEDULER_YIELD:
SOS_SCHEDULER_YIELD uppstår när en tråd frivilligt släpper CPU-tid och väntar på att återta den. Detta händer när en tråd har utnyttjat hela sitt **quantum** (vanligtvis 4 ms) och SQL Server-schemaläggaren flyttar den från "running" till "runnable"-listan för att ge andra trådar CPU-tid. Det indikerar ofta hög CPU-belastning och att fler trådar konkurrerar om CPU-tid.

Waiting for CPU:
SQL Server väntar på att tilldela en **CPU** för att bearbeta en uppgift. Det indikerar att en tråd är i "runnable"-tillstånd men inte har tilldelats CPU-tid ännu. Detta kan tyda på CPU-överbelastning eller ineffektiv schemaläggning.

THREADPOOL:
THREADPOOL-väntan inträffar när det saknas tillgängliga arbetstrådar för att bearbeta inkommande förfrågningar. Detta kan hända när SQL Server har nått sin maximala trådkapacitet. Det kan tyda på att servern är överbelastad eller att MAXDOP är felkonfigurerat, vilket leder till brist på arbetstrådar.

ASYNC_NETWORK_IO:
ASYNC_NETWORK_IO uppstår när SQL Server väntar på att en klient (t.ex. en applikation) ska konsumera data från servern. Detta tyder ofta på långsam dataöverföring eller att klienten inte hämtar data tillräckligt snabbt, vilket kan bero på nätverksproblem eller ineffektiv klientkod.

RESOURCE_SEMAPHORE:
RESOURCE_SEMAPHORE väntetyp uppstår när SQL Server väntar på att tilldela minnesresurser till en fråga som behöver ett minnesanslag. Detta tyder på att systemet är

pressat på minne och kan orsaka förseningar i frågeexekveringen, ofta relaterat till dålig minneshantering eller stora frågor.

CMEMTHREAD:
CMEMTHREAD-väntetyp uppstår när flera trådar försöker samtidigt få tillgång till minnesstrukturer som används för att cachera frågor eller exekveringsplaner. Detta kan indikera **plan cache bloat** (överbelastning) eller dålig cachehantering, där SQL Server spenderar onödig tid på att hantera minnescacher.

SQL Server Wait Types
https://www.sqlskills.com/help/waits/sos_scheduler_yield/
https://www.sqlskills.com/help/waits/threadpool/
https://www.sqlskills.com/help/waits/async_network_io/
https://www.sqlskills.com/help/waits/resource_semaphore/
https://www.sqlskills.com/help/waits/cmemthread/

V1 Lab 1: SQL Server Architecture, Scheduling, and Waits

- Exercise 1: Recording CPU and NUMA Configuration
- Exercise 2: Monitoring schedulers and user requests
- Exercise 3: Monitoring waiting tasks and recording wait statistics

Estimated time: 60 minutes

Lab 1: Architecture, scheduling, and wait statistics

- Ex 1: Document the hardware configuration
- Ex 2: Determine the major wait statistics

Estimated Time: 30 minutes