Module 2

SQL Server I/O

Module Overview

- Core concepts of I/O
- Storage solutions
- I/O setup and testing

Lesson: Core concepts of I/O

- I/O patterns for data files (mdf and ndf)
- I/O patterns for transaction log files (ldf)
- Recovery related aspects
- Input/Output Operations Per Second
- Throughput
- Latency Factor
- Magnetic Disk and SSD

I/O patterns for data files (mdf and ndf)

- Reads
  - Synchronous, when data is requested
  - Read-ahead (EE has deeper RA)
  - Buffer ramp-up (EE)
  - Merry-go-round scanning (EE)
    - One scan operation jumps in on some other on-going scan
    - Only for allocation order scans (IAM)
      - Requires heap and NOLOCK
- Writes
  - Asynchronous
    - Checkpoint
    - Lazywriter
  - Exception is minimally logged operations which are synchronous

Investigating I/O using Extended Events
https://techcommunity.microsoft.com/t5/core-infrastructure-and-security/dissecting-sql-server-physical-reads-with-extended-events-and/ba-p/370393

Självklart!

Buffer Ramp-up (EE): Detta fenomen inträffar när SQL Server Enterprise Edition snabbt läser in data i bufferpoolen för att optimera frågeprestanda. Det hjälper till att minimera lästid från disk genom att effektivt utnyttja minnet, vilket snabbar upp dataåtkomst och förbättrar övergripande systemrespons.

Merry-go-Round Scanning (EE): Detta är en funktion i SQL Server Enterprise Edition som optimerar hur stora tabell- eller indexscanningar hanteras. Istället för att flera frågor läser samma data flera gånger från disk, delar de på den inlästa datan, vilket minskar I/O-belastningen och förbättrar frågeprestanda genom mer effektiv användning av buffertminnet.

I SQL Server finns det flera typer av skrivningar som sker för att hantera dataändringar och säkerställa dataintegritet. Här är en närmare titt på dessa:

Writes:
Dessa är skrivoperationer till disk som uppstår när data i minnet (buffer cache) måste skrivas tillbaka till permanent lagring (som diskar) för att säkerställa att ändringar bevaras.

Asynchronous Writes

Asynkrona skrivningar innebär att SQL Server inte väntar på att själva skrivoperationen ska slutföras innan det fortsätter bearbeta andra uppgifter.

Checkpoint:
Under en checkpoint skrivs alla ändrade sidor (dirty pages) i buffer cache till disk. Detta sker periodiskt för att minska mängden arbete under en återställning efter krascher. Checkpoint skrivningar är asynkrona och hjälper till att hålla transaktionsloggen hanterbar.

Lazywriter:
Lazywriter är en bakgrundsprocess som ansvarar för att frigöra minnesutrymme genom att skriva ut mindre använda sidor (dirty pages) från buffer cache till disken när minnet behövs för nya frågor. Denna skrivning sker också asynkront.

Exception - Minimally Logged Operations (Synchronous Writes)
Minimally logged operations, som BULK INSERT, SELECT INTO, och vissa indexoperationer, använder sig av minimalt loggade skrivningar, vilket gör att stora mängder data kan skrivas snabbt utan att allt loggas i transaktionsloggen. Dessa operationer är synkrona, vilket innebär att SQL Server väntar tills skrivningen är färdig innan det går vidare till nästa operation.

Sammanfattningsvis: Checkpoints och Lazywriter hanterar skrivningar asynkront för att optimera prestanda, medan minimalt loggade operationer är undantaget då dessa kräver synkron skrivning till disken.

I/O patterns for transaction log files (ldf)

- Some operations reads the transaction log
  - Backup
    - Log backup
    - Other backup types (includes log records produced while backup was running)
  - Rollback
  - Recovery
    - Start-up
    - Fail-over
    - Restore
  - Transactional replication and Change Data Capture
  - Create a database snapshot
    - DBCC CHECKDB uses snapshot internally
  - Availability Groups / Database mirroring (if falls behind)
  - CHECKPOINT in simple recovery (need to read the log in order to clear the log)
- Writes
  - Synchronous sequential writes

SQL Server Transaction Log Architecture and Management Guide
https://docs.microsoft.com/en-us/sql/relational-databases/sql-server-transaction-log-architecture-and-management-guide

Synchronous sequential writes i SQL Server innebär att skrivoperationerna sker i en ordnad följd, där varje skrivning måste slutföras innan nästa kan påbörjas. "Synchronous" betyder att SQL Server väntar på att varje skrivoperation ska slutföras innan det fortsätter med nästa steg i bearbetningen. Dessa skrivningar är vanliga i transaktionskritiska delar av systemet där integriteten och säkerheten hos data är avgörande.

Ett tydligt exempel på synchronous sequential writes är skrivning till **transaktionsloggen**. Varje transaktion som gör en förändring i databasen måste loggas i transaktionsloggen, och SQL Server måste vänta på att denna loggning är färdig innan den fortsätter med transaktionen. Den här processen är sekventiell och synkron för att säkerställa att all data skrivs korrekt och i rätt ordning.

Dessa skrivningar är viktiga för att garantera dataintegritet, men de kan också bli en prestandaflaskhals om loggfilens lagring är långsam, eftersom SQL Server måste vänta på att varje skrivning till transaktionsloggen ska slutföras innan nästa kan börja.

I/O block size

- Different operations uses different I/O sizes

Operation	I/O block size
Transaction log write	512 bytes – 60 KB
Checkpoint/Lazywriter	8KB – 1 MB
Read-ahead scans	128KB – 512KB
Bulk loads	256KB
Backup/Restore	1MB
Columnstore read-ahead	8MB
File initialization	8MB
In-memory OLTP checkpoint	1MB

What is SQL Server's IO Block Size?
https://blog.purestorage.com/purely-technical/what-is-sql-servers-io-block-size/

Recovery related aspects

- Keep number of virtual log files to a reasonable number
  - Shortens recovery, failover etc.
- Turn on accelerated database recovery (2019)
  - Shortens recovery, failover etc
  - “Immediate” rollback
  - Less space used for transaction log
- Turn on indirect checkpoints
  - Default for new databases as of 2016
    - Lots of databases were born in an earlier version
  - Shorter checkpoints
  - Faster backups

För att hålla antalet virtuella logfiler (VLF:er) i SQL Server lågt och därmed förbättra prestandan för logghantering, kan du följa dessa rekommendationer:

1. **Ange rätt storlek på transaktionsloggen från början**:
- Vid skapandet av en databas, sätt en initial storlek på transaktionsloggen som matchar din förväntade arbetsbelastning. En större initial storlek minskar behovet av framtida autogrowth, vilket i sin tur minskar antalet VLF:er.

2. **Undvik små autogrowth-inställningar**:
- Konfigurera transaktionsloggens autogrowth för att växa i stora steg istället för små. Om autogrowth-värdet är för litet, kommer SQL Server att skapa många små VLF:er, vilket kan leda till prestandaproblem.
- Rekommendationen är att ställa in autogrowth till 512 MB eller 1 GB för att hålla antalet VLF:er lågt.

3. **Använd manuella loggutökningar**:
- Ibland kan du manuellt utöka storleken på transaktionsloggen istället för att förlita dig på autogrowth. Detta ger dig bättre kontroll över storleken på VLF:erna.

4. **Minska antalet VLF:er i befintliga databaser**:
- Om din databas redan har för många VLF:er, kan du krympa loggen och sedan växa den manuellt för att minska antalet VLF:er. Detta görs i två steg:
  1. Krymp loggfilen (men detta bör undvikas ofta då det kan orsaka fragmentering och andra problem).
  2. Expandera loggfilen manuellt till en större storlek, antingen i ett enda steg eller i större

tillväxtblock.

5. **Kontrollera antalet VLF:er**:
  - Använd följande SQL-skript för att se antalet VLF:er i en databas:
    ```sql
    DBCC LOGINFO;
    ```
    Skriptet visar hur många VLF:er som finns i din databas. Om du ser ett stort antal VLF:er kan det vara dags att överväga att justera loggens storlek eller tillväxtmönster.

Genom att följa dessa steg kan du hålla antalet VLF:er hanterbart och därmed förbättra både återställningsprestanda och övergripande hantering av transaktionsloggen i SQL Server.

SQL Server Transaction Log Architecture and Management Guide
https://docs.microsoft.com/en-us/sql/relational-databases/sql-server-transaction-log-architecture-and-management-guide

A Busy/Accidental DBA's Guide to Managing VLFs
http://adventuresinsql.com/2009/12/a-busyaccidental-dbas-guide-to-managing-vlfs/

Virtual Log Files
https://dallasdbas.com/virtual-log-files/

Accelerated Database Recovery in SQL Server 2019
https://www.mssqltips.com/sqlservertip/5971/accelerated-database-recovery-in-sql-server-2019/

More Reasons to Enable SQL Server Indirect Checkpoints
https://www.mssqltips.com/sqlservertip/7108/sql-server-indirect-checkpoints-why-enable/

ALTER DATABASE [DatabasensNamn] SET ACCELERATED_DATABASE_RECOVERY = ON;

Aktivering av Accelerated Database Recovery (ADR) kan öka användningen av diskutrymme för databasen eftersom det behåller versionshistorik för varje rad inom databasen för att snabbt kunna återställa transaktioner. Detta kan leda till större lagringskrav, särskilt för databaser med hög transaktionsvolym eller stora datamängder.

ALTER DATABASE [DittDatabasNamn] SET TARGET_RECOVERY_TIME = 60 SECONDS;

Det är viktigt att välja en TARGET_RECOVERY_TIME som balanserar mellan återställningstid och systemprestanda. En lägre återställningstid kan leda till frekventare checkpoints, vilket potentiellt kan påverka prestandan, medan en längre återställningstid kan minska antalet checkpoints och därmed påverka hur snabbt systemet kan återställas efter ett fel.

Lesson: Storage solutions

- Direct-Attached Storage
- Storage Area Network
- Windows Storage Spaces
- SQL Server data files in Microsoft Azure
- Selecting a storage solution

Direct-Attached Storage

- One or more storage devices is directly attached to the server through a host bus adapter
- Dedicated to a specific server
- Advantages:
  - Easy and fast to provision
  - Easy monitoring and troubleshooting
- Disadvantages:
  - Expansion may be limited by hardware design
  - Lacks flexibility—unused capacity cannot be shared with other servers

Storage Area Network

- Pools of drive arrays linked together with a network
- Each server connects into this network and can share drives, cache, and throughput with many more servers
- Advantages:
  - Increases disk utilization and reduces management
  - Mirroring, snapshots, continuous data protection, clustering, and geoclustering only offered by SANs
- Disadvantages:
  - Unpredictable performance, higher latency, limited bandwidth, and high cost

Windows Storage Spaces

- Software RAID managed by Windows
- Physical disks are grouped into storage pools
- Storage Spaces enables volumes to be created from storage pools
- Storage Spaces supports resiliency:
  - Mirroring: similar to RAID 1
  - Parity: similar to RAID 5
  - Simple: similar to RAID 0, no resiliency

Storage Spaces overview
https://docs.microsoft.com/en-us/windows-server/storage/storage-spaces/overview

Database files in Microsoft Azure

- SQL Server instance connects to database files held as Page Blobs in Azure Blob storage
  - Instance may be on-premises or on an Azure virtual machine
- Advantages:
  - Unlimited storage—pay for what you use
  - Easy migration
  - Centralized storage
  - Azure snapshot backup
- Disadvantages:
  - Maximum file size 1 TB
  - Difficult to predict costs
  - Not all SQL Server features available
    - Filestream
    - In-Memory OLTP

SQL Server data files in Microsoft Azure
https://docs.microsoft.com/en-us/sql/relational-databases/databases/sql-server-data-files-in-microsoft-azure

Selecting a storage solution

- No one-size-fits-all solution
- Many factors to consider:
  - Performance requirements
  - Organizational cloud strategy
  - Existing solutions
  - Budget
  - Urgency

Lesson: I/O Setup and testing

- NTFS Allocation Unit size
- Storage performance testing
- I/O related wait types

NTFS Allocation Unit size

- Sets the smallest unit of consumption on NTFS volumes for buffered I/O
- SQL Server mainly does unbuffered I/O
- Set during formatting
- Default value is 4 KB
- For SQL Server, a 64-KB allocation unit size is recommended:
  - Only affects a few operations because most SQL Server I/O is unbuffered
  - Improves read-ahead performance
  - NTFS compression may not be used
  - Less meta-data

SQL Server I/O block size
https://blog.purestorage.com/purely-technical/what-is-sql-servers-io-block-size/

More on SQL Server I/O, including elaboration on Allocation Unit size
https://www.nocentino.com/posts/2021-12-10-sqlserver-io-size/

Example of how you can measure I/O size. This example is for a log records, but can be adapted to other operations
https://www.brentozar.com/archive/2012/05/how-big-your-log-writes-spying-on-sql-server-transaction-log/

För att ändra NTFS **Allocation Unit Size** till 64 KB i Windows, behöver du formatera om disken eller partitionen. Detta innebär att all data på disken kommer att raderas, så se till att säkerhetskopiera viktig information innan du fortsätter. Här är stegen för att göra det:

Steg för att ändra NTFS Allocation Unit Size till 64 KB:

1. **Öppna File Explorer**:
   - Gå till "Den här datorn" (This PC) och identifiera den disk eller partition som du vill formatera om med 64 KB som allocation unit size.

2. **Högerklicka på Disken eller Partitionen**:
   - Högerklicka på den disk eller partition du vill formatera om och välj **Format** i menyn.

3. **Välj inställningar i Format-fönstret**:

- Under fältet **File System**, se till att det är satt till **NTFS**.
- I rullgardinsmenyn för **Allocation Unit Size**, välj **64K**.
- Skriv ett namn i fältet för **Volume Label** om du vill.
- Se till att **Quick Format** är valt om du vill att formateringen ska gå snabbare.

4. **Formatera disken**:
- Klicka på **Start** för att påbörja formateringen. Kom ihåg att all data på disken kommer att raderas.

5. **Bekräfta varningen**:
- Windows kommer att varna dig om att alla data kommer att raderas. Klicka på **OK** för att fortsätta.

6. **Vänta tills formateringen är klar**:
- När formateringen är slutförd kommer disken att ha en NTFS file system med en 64 KB allocation unit size.

Alternativt: Använd kommandoraden med DiskPart
Du kan också använda kommandoraden för att formatera om disken med 64 KB Allocation Unit Size.

1. **Öppna Command Prompt som administratör**:
- Tryck på **Windows-tangenten**, skriv `cmd`, högerklicka på **Command Prompt** och välj **Kör som administratör**.

2. **Använd DiskPart**:
- Skriv `diskpart` och tryck **Enter** för att starta DiskPart-verktyget.

3. **Välj din disk**:
- Skriv `list volume` och tryck **Enter** för att se en lista över volymer.
- Skriv `select volume X`, där **X** är volymnumret för den disk du vill formatera.

4. **Formatera med 64K Allocation Unit Size**:
- Skriv följande kommando för att formatera volymen med 64 KB allocation unit size:
  ```cmd
  format fs=ntfs unit=65536 quick
  ```
- Tryck **Enter**. Detta startar formateringsprocessen med 64 KB allocation unit size.

Efter dessa steg kommer disken att vara formaterad med 64 KB som allocation unit

size.

Storage performance testing

- Diskspd
  - A general-purpose load generator for I/O subsystems
  - Can be configured to mimic SQL Server I/O
  - Suitable for use as a performance-tuning tool
  - Replaces SQLIO
- ChrystalDiskMark
  - GUI on top of DISKSPD
  - You might want to change default values so it matches SQL Server I/O sizes
- Or, run BACKUP and DBCC CHEDKDB and compare the new with the old
  - Backup to NUL to test reading of data only
    - Note that encryption (backup or database) and compression will add CPU load
  - CHECKDB does more than raw reading of source data

```
DiskSpd.exe -c500G –d600 -r –w20 -t8 -o8 -b8K -h -L H:\testfile.dat
```

Demo Diskspd

Ja, här är en genomgång av de olika parametrarna i din **Diskspd.exe**-kommando:
Struktur av kommandot:
C:\diskspd\Diskspd.exe -b8K -d20 -h -L -o2 -t4 -r -w30 -c50M c:\diskspd\io.dat **Förklaring av parametrarna:**

1.-b8K

1. Anger blockstorleken som används för I/O-operationer.
2. **Här:** 8 KB per operation.

2.-d20

1. Anger testets varaktighet i sekunder.
2. **Här:** Testet körs i 20 sekunder.

3.-h

1. Aktiverar "hardware write caching bypass".
2. **Detta betyder:** Skriver direkt till disken, förbi OS-cache.

4.-L

1. Aktiverar latensmätning för att få statistik om fördröjningstider i I/O.

5.-o2

1. Ställer in antalet **utestående I/O-operationer** (Queue Depth).
2. **Här:** 2 samtidiga operationer per tråd.

6.-t4

1. Antal trådar som används för testet.
2. **Här:** 4 trådar.

7.-r

1. Anger att testet ska använda **random access** istället för sekventiell åtkomst.

8.-w30

1. Sätter skriv-andelen i en **blandad läs/skriv-belastning**.

2. **Här:** 30% av operationerna är skrivningar, resten (70%) är läsningar.

1.-c50M

1. Skapar en testfil på 50 MB för att köra testet mot.

2.c:\diskspd\io.dat

1. Detta är filen som används för att genomföra testet.

Vad gör detta test?

•Simulerar en I/O-belastning med 8 KB random läs/skriv-operationer.
•Kör testet i 20 sekunder.
•Använder 4 trådar med 2 samtidiga I/O-operationer per tråd.
•Använder en testfil på 50 MB.
•30% av operationerna är skrivningar, 70% är läsningar.
•Latensdata samlas in, och hårdvarucache ignoreras.

Användningsområde:

Detta kan vara ett bra test för att simulera en verklig belastning på en disk, exempelvis för att mäta prestanda för en databas eller en applikation som gör blandade läs/skriv-operationer.

Resultat för testet

CPU-användning

•**Snittanvändning:** 9.8% (av alla kärnor).
•**Individuell kärnanvändning:**

- CPU 0: **17.89%**
- CPU 1: **9.77%**
- CPU 2: **8.28%**
- CPU 3: **3.28%**

•**Slutsats:** Belastningen var relativt låg och ojämnt fördelad mellan kärnorna.

Total I/O-prestanda

•**Total mängd data: 609 MB** skrivna/lästa.
•**Total IOPS: 3718.79** operationer per sekund.
•**Total bandbredd: 29.05 MiB/s**.

Per tråd:

•**Genomsnittlig latens: ~2.1 ms**.
•**Standardavvikelse i latens: ~3 ms** (relativt låg variation).

Läsoperationer (Read IO)

•**Total mängd data: 426 MB**.
•**Läsningar per sekund: 2600 IOPS**.
•**Genomsnittlig latens: ~2.87 ms**.
•**Hög standardavvikelse i latens: 3.33 ms** → vissa läsningar tar längre tid.

Skrivoperationer (Write IO)

•**Total mängd data: 183 MB**.
•**Skrivningar per sekund: 1118 IOPS**.
•**Genomsnittlig latens: ~0.46 ms**.

•**Lägre standardavvikelse: 0.86 ms** → mer konsekvent prestanda än läsningarna.
Latensfördelning (Percentiler)
Här visas hur lång tid 99%, 99.9% osv. av I/O-operationerna tog.
•**50:e percentilen (median):**
- Läsningar: **1.6 ms**
- Skrivningar: **0.32 ms**

•**90:e percentilen:**
- Läsningar: **7.16 ms**
- Skrivningar: **0.76 ms**

•**99:e percentilen:**
- Läsningar: **15.17 ms**
- Skrivningar: **2.81 ms**

•**Max latens:**
- Läsningar: **87.72 ms**
- Skrivningar: **83.41 ms**

Slutsats:
•De flesta skrivningar är snabba (**under 1 ms**).
•Läsningar har större variation och kan ibland bli mycket långsamma.
•Några enstaka extrema fall når **87 ms**, vilket kan indikera I/O-stall på disken.
Övergripande slutsats
•Diskens prestanda är okej men inte fantastisk.
•Läsningar har hög latensspridning, vilket kan tyda på att disken är mer optimerad för skrivning.
•Skrivningar är snabbare och mer förutsägbara.
•IOPS på 3700 och 29 MiB/s är inte exceptionellt högt, vilket kan betyda att det är en vanlig SSD eller en mekanisk disk.

Testing I/O performance for SQL Server using ChrystalDiskMark
https://www.brentozar.com/archive/2012/03/how-fast-your-san-or-how-slow/

DISKSPD, github
https://github.com/microsoft/diskspd

Using Diskspd to test SQL Server Storage Subsystems
https://www.sqlshack.com/using-diskspd-to-test-sql-server-storage-subsystems/

BACKUP DATABASE [MinDatabas] TO DISK = 'NUL';

I/O related wait types

- PAGEIOLATCH
  - A task waits for a data page to be retrieved from disk to memory
  - Reduce waits by
    - Improving I/O subsystem
    - Tune query so it does less I/O
- WRITELOG
  - Writing to log is at the latest done synchronously as end of transaction
  - Reduce waits by
    - Fewer, larger transactions
    - Fewer indexes
    - Improving I/O subsystem

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

Optional Lab 2: SQL Server I/O

- Ex 1: Determine if a query is I/O bound
- Ex 2: If time permits, improve the performance of the query

Estimated Time: 30 minutes