https://www.sqldbachamps.com
https://github.com/PMSQLDBA/PraveenMadupu
Telegram Group→SQLDBACloud-DevOps https://t.me/+r2OYsT2qRZJkYWVl

Praveen Madupu
Mb: +91 98661 30093
Database Administrator

🧠 Understanding SQL Server Memory and Disk I/O Relationship

SQL Server relies heavily on **memory** (RAM) for performance. The **Buffer Pool** stores frequently accessed data pages and index pages.

When a query requests data:

- If the data is already in memory → SQL Server reads it quickly (logical read).
- If not → it must fetch from disk (physical read), which is much slower and increases **disk I/O load**.

A **healthy SQL Server** typically has:

- **Buffer Cache Hit Ratio > 95%**, meaning most reads are served from memory.

When available memory is insufficient, SQL Server frequently swaps data between memory and disk, leading to **high I/O latency and performance degradation.**

⚠️ Common Causes of High Disk I/O (Due to Memory Pressure)

Cause	Description	Impact
Insufficient memory allocation	SQL Server's max server memory setting is too low, preventing efficient caching.	Frequent physical reads from disk.
Large working set	Queries scan huge tables or datasets beyond available memory.	Excessive I/O and long query times.
Missing or outdated indexes	Poor query plans cause full table scans instead of index seeks.	Unnecessary reads, tempdb usage increases.
Heavy TempDB workload	Large sorts, hash joins, and version store (snapshot isolation) operations.	TempDB disk contention and latency.
Aggressive Checkpoints / Lazy Writer activity	Dirty pages flushed frequently to disk due to memory pressure.	Increased disk writes and I/O latency.

🔍 Real-Time Symptoms You'll Notice

Symptom	Tool / DMV to Check	What It Means
Slow query performance	Query Store, sys.dm_exec_query_stats	High I/O cost queries
PAGEIOLATCH_* waits	sys.dm_os_wait_stats	Waiting on data to be read from disk
High read/write latency	sys.dm_io_virtual_file_stats	Disk I/O subsystem under stress
Memory pressure messages	Error Log (Msg 701, "insufficient memory")	SQL Server hitting memory limits

🧩 Step-by-Step Resolution Plan

📄 Step 1: Check Current Memory Usage

Run the following DMV queries: -- Check SQL Server memory settings

```
EXEC sp_configure 'max server memory';
```

-- Memory distribution

```
SELECT total_physical_memory_kb/1024 AS TotalRAM_MB,
    available_physical_memory_kb/1024 AS AvailableRAM_MB,
    total_page_file_kb/1024 AS PageFile_MB
FROM sys.dm_os_sys_memory;
```

-- Buffer pool usage

```
SELECT (cntr_value/1024) AS BufferPool_MB
FROM sys.dm_os_performance_counters
WHERE counter_name = 'Database Pages';
```

✅ **Resolution:**

- If max server memory is too low, increase it gradually (e.g., allocate 75–80% of total server RAM to SQL Server).

  Example:

```
EXEC sp_configure 'show advanced options', 1;
RECONFIGURE;
EXEC sp_configure 'max server memory', 32768; -- For 32 GB allocation
RECONFIGURE;
```

📝 **Step 2: Verify Buffer Pool Efficiency**

Run:


```
SELECT
  (a.cntr_value * 1.0 / b.cntr_value) * 100 AS BufferCacheHitRatio
FROM
  sys.dm_os_performance_counters a
JOIN
  sys.dm_os_performance_counters b
ON
  a.object_name = b.object_name
WHERE
  a.counter_name = 'Buffer cache hit ratio'
  AND b.counter_name = 'Buffer cache hit ratio base';
```

✅ **Resolution:**

- If ratio < **95%**, SQL Server is relying too much on disk → tune indexes or add more memory.

📝 **Step 3: Identify Queries Causing Excessive Disk Reads**

```
SELECT TOP 10
  total_logical_reads, total_physical_reads,
  execution_count, total_elapsed_time/1000 AS TotalTime_ms,
  SUBSTRING(qt.text, (qs.statement_start_offset/2)+1,
    ((CASE qs.statement_end_offset
      WHEN -1 THEN DATALENGTH(qt.text)
      ELSE qs.statement_end_offset END
     - qs.statement_start_offset)/2)+1) AS QueryText
FROM sys.dm_exec_query_stats AS qs
CROSS APPLY sys.dm_exec_sql_text(qs.sql_handle) AS qt
ORDER BY total_physical_reads DESC;
```

✅ **Resolution:**

- Tune high I/O queries by:
  - Creating missing indexes (Database Tuning Advisor or Missing Index DMV)
  - Rewriting queries to reduce table scans
  - Updating outdated statistics

📝 **Step 4: Analyze TempDB Usage**

Check TempDB file space:

```
USE tempdb;
GO
SELECT
   SUM(user_object_reserved_page_count)*8/1024 AS UserObjects_MB,
   SUM(internal_object_reserved_page_count)*8/1024 AS InternalObjects_MB,
   SUM(version_store_reserved_page_count)*8/1024 AS VersionStore_MB
FROM sys.dm_db_file_space_usage;
```

✅ **Resolution:**

- Add multiple TempDB data files (1 per logical CPU core up to 8).
- Move TempDB to a fast SSD disk.
- Review query patterns that overuse TempDB (large sorts, row versioning).

📝 **Step 5: Measure Disk Latency**


```
SELECT
   DB_NAME(database_id) AS DatabaseName,
   file_id,
   io_stall_read_ms/num_of_reads AS ReadLatency_ms,
   io_stall_write_ms/num_of_writes AS WriteLatency_ms
FROM sys.dm_io_virtual_file_stats(NULL, NULL);
```

✅ **Resolution:**

- Ideal: Read/Write latency < 10 ms.
- If > 20 ms, check:
  - Underlying storage performance.
  - Virtual machine storage throttling.
  - Disk queue length and IOPS via PerfMon.

📝 **Step 6: Monitor Checkpoints and Lazy Writer**

```
SELECT * FROM sys.dm_os_performance_counters
WHERE counter_name IN ('Checkpoint pages/sec', 'Lazy writes/sec');
```

✅ **Resolution:**

- Frequent checkpoints/lazy writes = memory pressure.
- Increase memory or tune queries to reduce dirty page creation.

Preventive Best Practices

Area	Recommendation
Memory Configuration	Set max server memory appropriately to avoid OS starvation and SQL paging.
Index Maintenance	Regular index rebuild/reorganize to reduce full scans.
Statistics Update	Schedule auto-update statistics to keep query optimizer efficient.
TempDB Optimization	Use multiple data files, pre-size files, and fast storage.
I/O Monitoring	Track latency and throughput weekly with sys.dm_io_virtual_file_stats or PerfMon.
Query Tuning	Avoid SELECT *, prefer covering indexes, and review execution plans regularly.

Example Real-Time Scenario (Interview / Production)

Issue: Users report slow performance during peak hours.

Observation: PAGEIOLATCH_SH waits increasing in sys.dm_os_wait_stats.

Diagnosis: Buffer Cache Hit Ratio = 86% → queries reading from disk.

Action: Increased max server memory from 16 GB to 28 GB, rebuilt missing indexes on largest table.

Result: Buffer cache hit ratio rose to 98%, average query time dropped from 12 sec → 3 sec.

Praveen Madupu
Mb: +91 98661 30093
Database Administrator