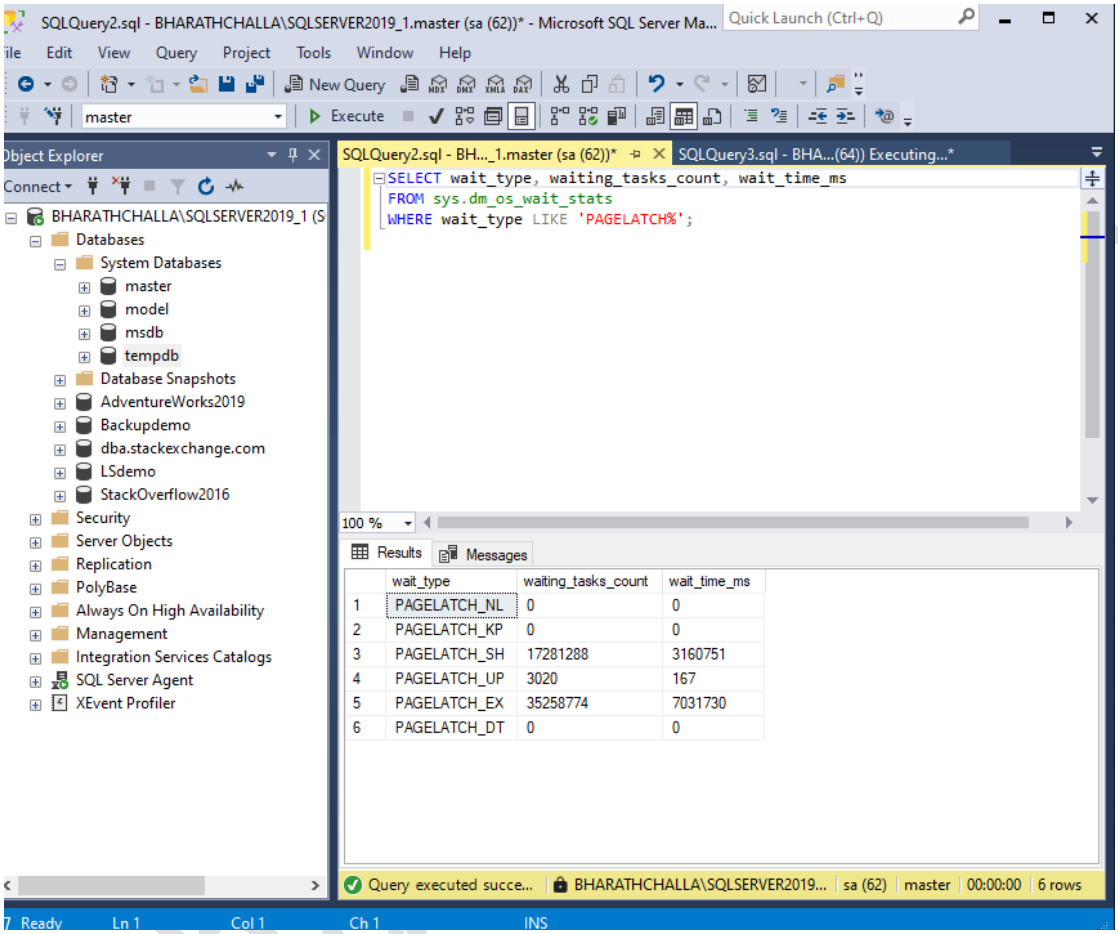
Issue: TempDB PAGELATCH contention occurs when multiple sessions compete for allocation pages (PFS, GAM, SGAM) in TempDB. This causes severe performance degradation and high wait times like PAGELATCH_EX and PAGELATCH_SH.

Symptoms Observed: High waits on PAGELATCH types

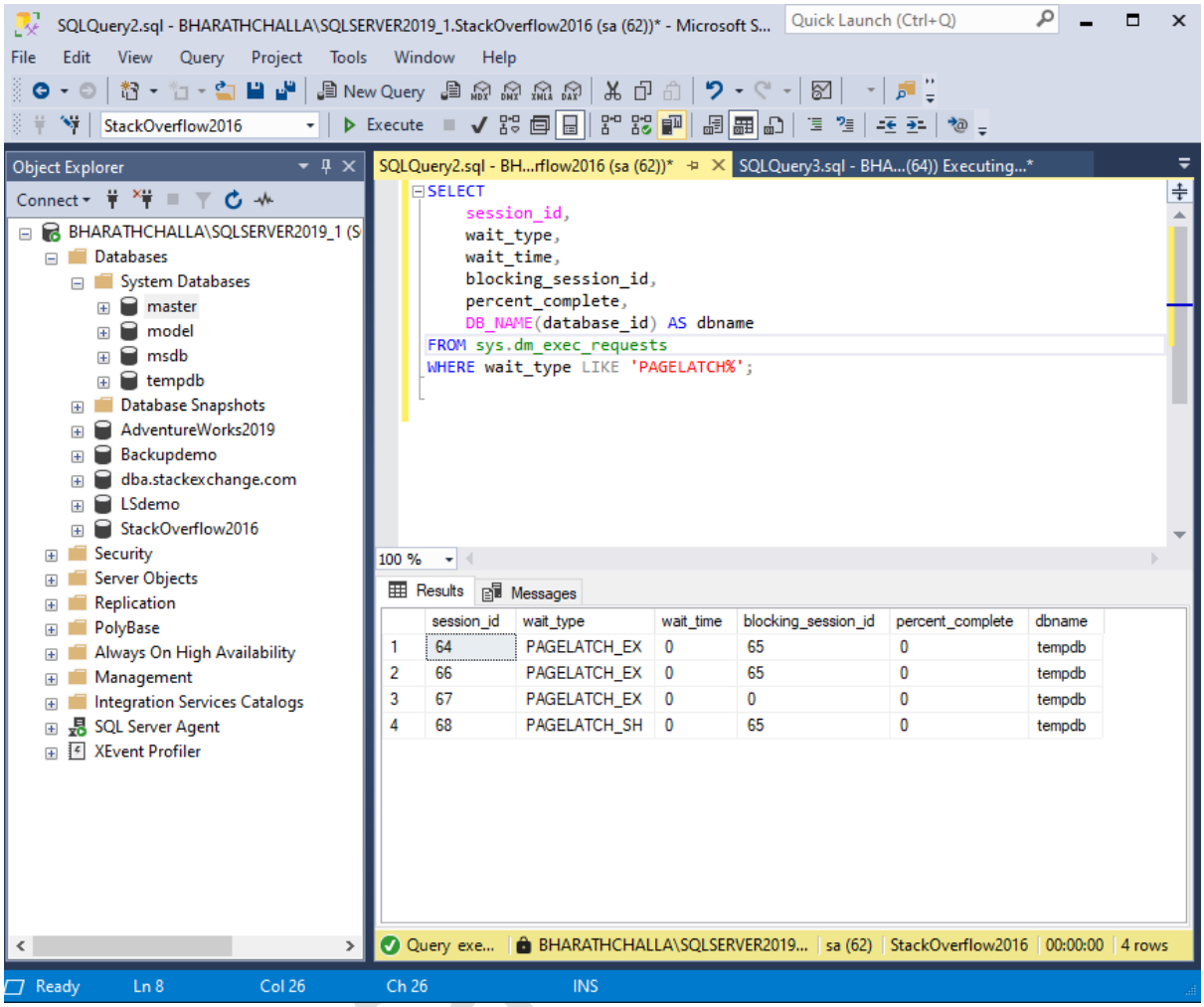


Identify Sessions Impacted:

We checked active requests waiting on PAGELATCH:

```
SELECT
   session_id, wait_type, wait_time, blocking_session_id,
   percent_complete, DB_NAME(database_id) AS dbname
FROM sys.dm_exec_requests
WHERE wait_type LIKE 'PAGELATCH%';
```

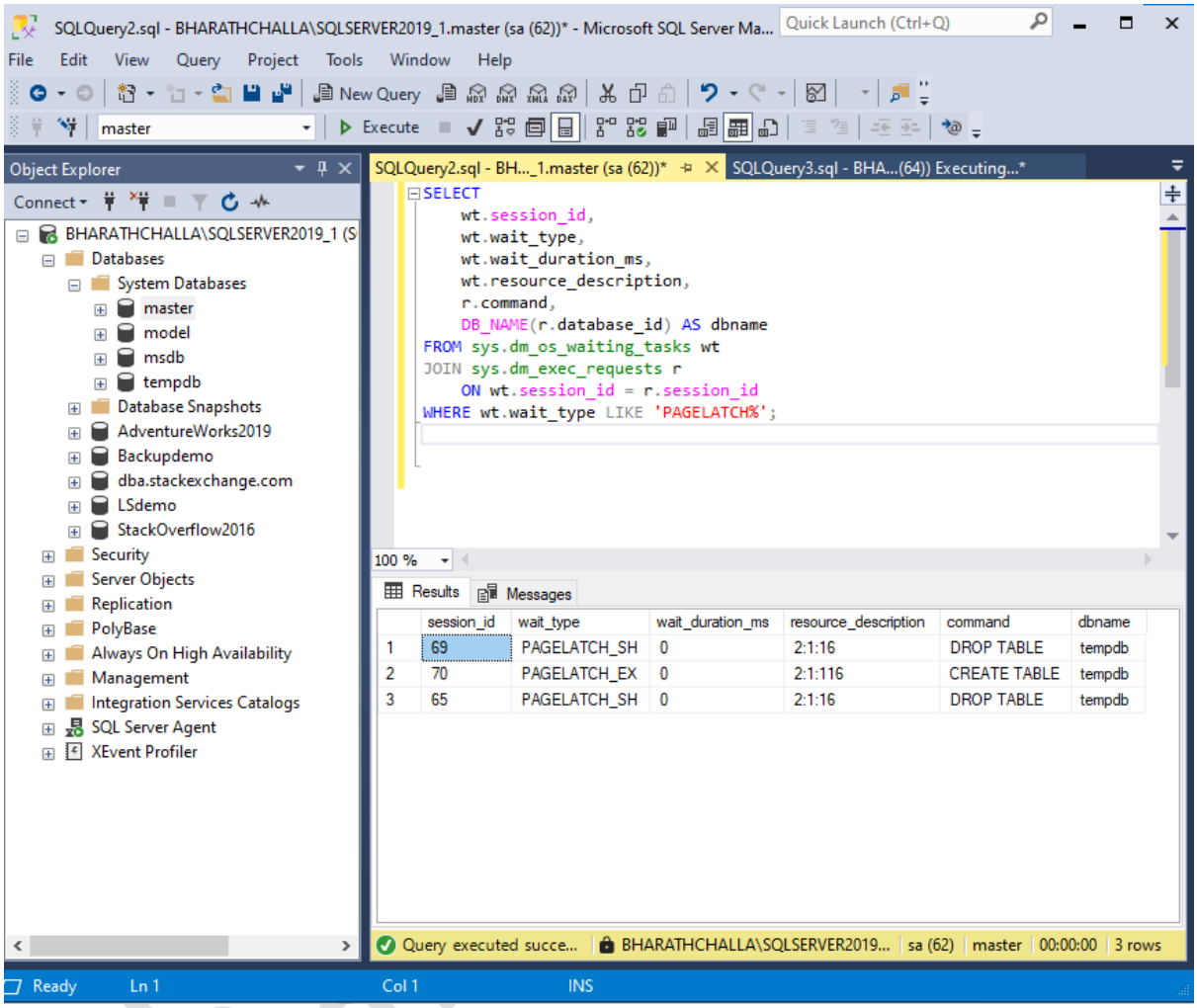
Output showing sessions stuck on TempDB: check below screenshot with session_id 64, 66, 67, 68 showing PAGELATCH waits.



Deep-Dive: Which Tasks Are Waiting?

```
SELECT
    wt.session_id,
    wt.wait_type,
    wt.wait_duration_ms,
    wt.resource_description,
    r.command,
    DB_NAME(r.database_id) AS dbname
FROM sys.dm_os_waiting_tasks wt
JOIN sys.dm_exec_requests r
    ON wt.session_id = r.session_id
WHERE wt.wait_type LIKE 'PAGELATCH%';
```

Output: Shows DROP TABLE & temp table workload check below screenshot

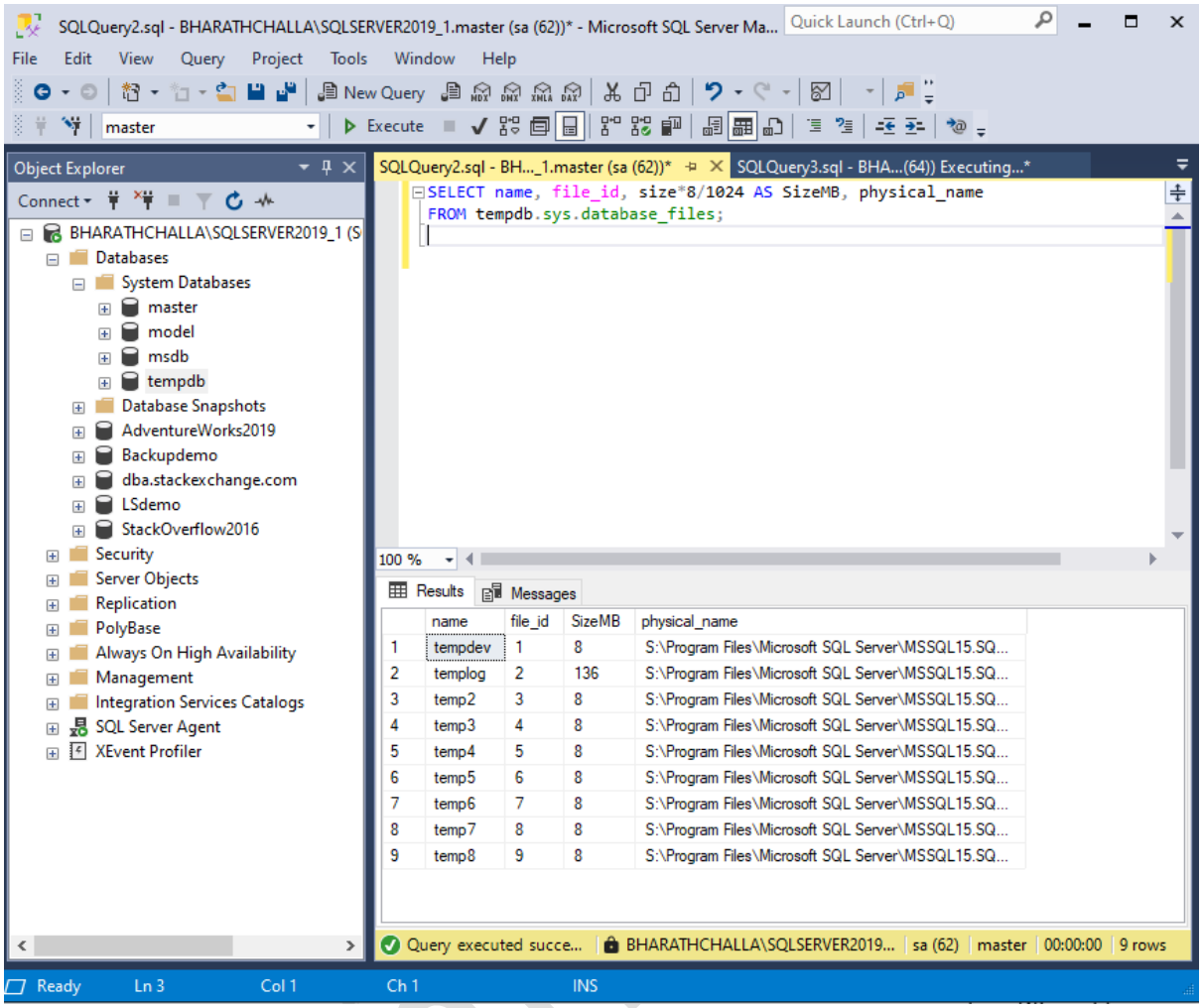


Confirm TempDB Files & Configuration:

SELECT name, file_id, size*8/1024 AS SizeMB, physical_name

FROM tempdb.sys.database_files;

Output: Showing only small TempDB files on S: drive



Fix Implemented: Increase TempDB Files & Size

We added **4 new TempDB data files** matching the recommended count (equal to number of CPU cores up to 8).

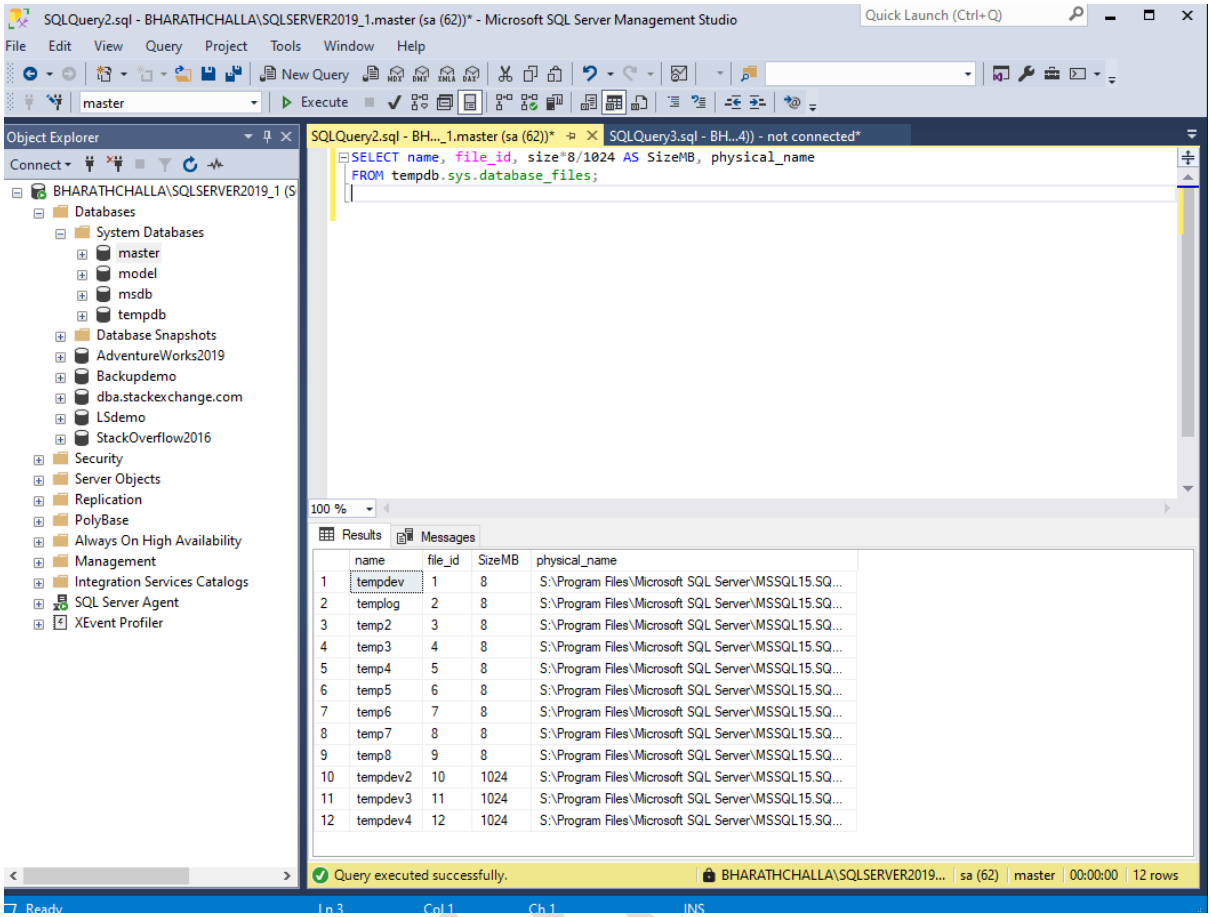
ALTER DATABASE tempdb

ADD FILE (NAME = tempdev2, FILENAME = 'S:\...\tempdb_mssql_2.ndf', SIZE = 1GB, FILEGROWTH = 512MB),

(NAME = tempdev3, FILENAME = 'S:\...\tempdb_mssql_3.ndf', SIZE = 1GB, FILEGROWTH = 512MB),

(NAME = tempdev4, FILENAME = 'S:\...\tempdb_mssql_4.ndf', SIZE = 1GB, FILEGROWTH = 512MB);

After Restart: New TempDB Files Visible



Validate Improvement:

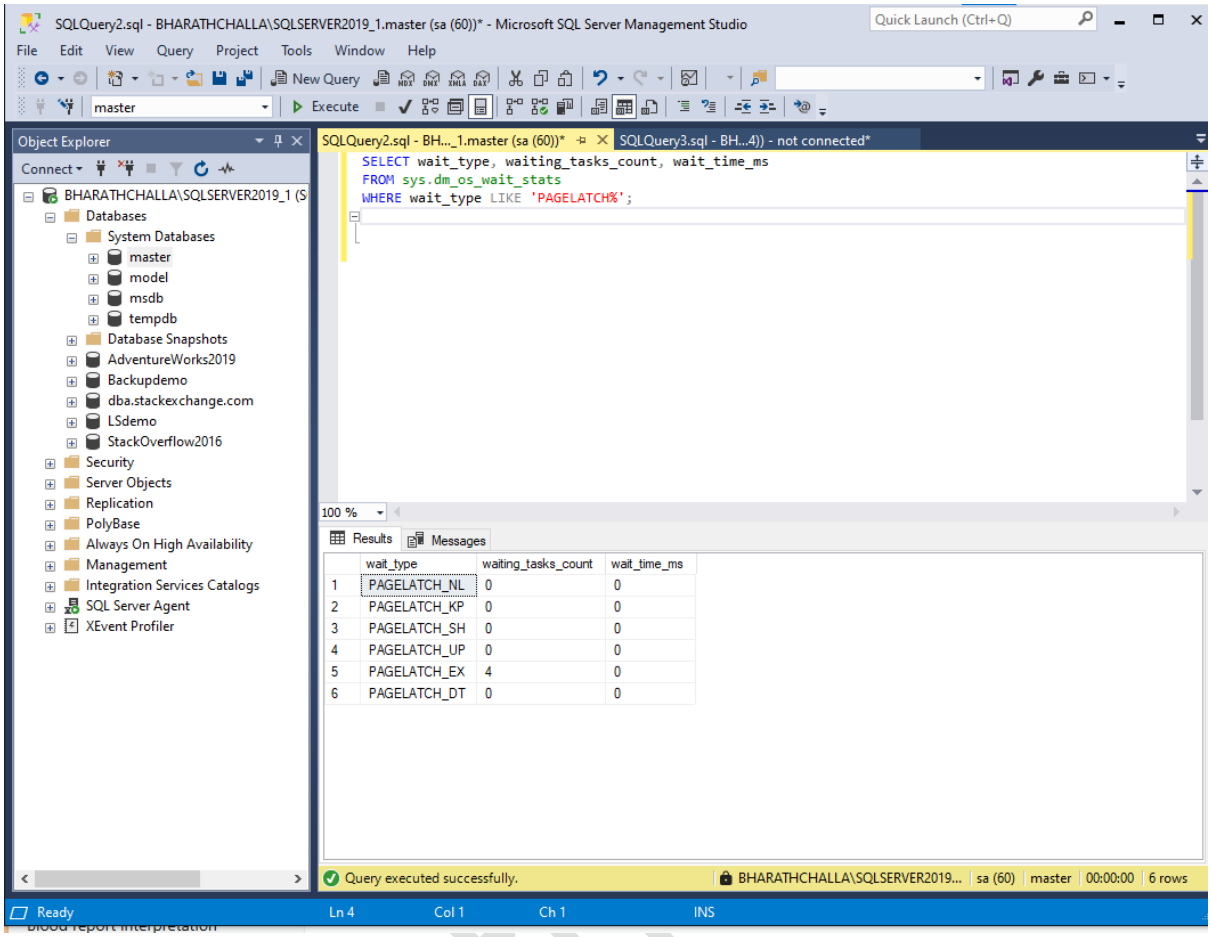
After restart, we checked wait stats again:

SELECT wait_type, waiting_tasks_count, wait_time_ms

FROM sys.dm_os_wait_stats

WHERE wait_type LIKE 'PAGELATCH%';

Output: PAGELATCH waits drastically reduced



Root Cause Summary:

Cause:

TempDB had only 1 small data file (8 MB) and all temp workloads forced allocation on a single file → latch contention on PFS/GAM/SGAM pages.

Fix:

- Increased TempDB data files to optimal count
- Set proper initial size & growth
- Restarted SQL Server to recreate TempDB cleanly

Outcome:

- PAGELATCH waits disappeared
- Performance stabilized
- Temp table workload executed instantly

BIGGGG NOTEEE:

If downtime is NOT allowed, What exactly CAN and CANNOT be done?

WHAT *CAN* BE DONE WITHOUT RESTART (Only Reduces Impact – Not a Fix)

1. Add more TempDB files

You already did this. It helps *new* allocations distribute better.

But existing latch hotspots **WILL NOT MOVE** until restart.

2. Kill or slow the workload causing contention

This is the most effective online action.

Run:

```
SELECT
   r.session_id,
   r.command,
   r.status,
   wt.wait_type,
   wt.resource_description,
   DB_NAME(r.database_id) AS dbname,
   r.cpu_time,
   r.total_elapsed_time
FROM sys.dm_os_waiting_tasks wt
JOIN sys.dm_exec_requests r ON wt.session_id = r.session_id
WHERE wt.wait_type LIKE 'PAGELATCH%';
```

Then kill the offenders:

KILL <session_id>;

Purpose: Removes immediate pressure.

Not a fix: Pressure returns when workload comes again.

3. Force cache clean-up

This reduces tempdb allocations for many workloads.

```
DBCC FREESYSTEMCACHE ('ALL');
DBCC FREEPROCCACHE;
```

Again — *temporary* relief.

4. Stop the specific job/query creating temp tables repeatedly

If you know which job or process is writing heavily to TempDB, you can:

- Stop the SQL Agent job

- Pause ETL
- Cancel a bulk process

This reduces pressure but **does not fix latch contention**

5. Throttle concurrency

If your workload is highly parallel (many requests hitting tempdb at once):

ALTER DATABASE SCOPED CONFIGURATION SET MAXDOP = 1;

This forces temp table creation to be serial instead of parallel.

Useful when you cannot restart.

6. Increase memory grant limit

Heavy memory spills to tempdb cause latch pressure.

Try:

EXEC sys.sp_configure 'min memory per query', 2048;
RECONFIGURE;

This is only useful in specific workloads.

7. Reduce version store pressure

If you use Read Committed Snapshot Isolation (RCSI) or Snapshot Isolation, version store grows in TempDB.

You can check version store usage:

SELECT * FROM sys.dm_tran_version_store_space_usage;

If it's high → reduce snapshot usage.

WHAT *CANNOT* BE DONE WITHOUT RESTART (Hard Limitations)

These are the reasons you *must* restart:

1. You cannot redistribute allocation map pages (PFS, GAM, SGAM) without restart

Latch contention happens on these pages:

- PFS (Page Free Space)
- GAM (Global Allocation Map)
- SGAM (Shared Global Allocation Map)

They are created **only once at startup**.

When SQL is already running:

- Adding tempdb files *does not rebalance* the first file
- "Hot pages" stay hot
- Latches continue

Only restart recreates these pages.

2. You cannot move existing temp table metadata

All active sessions keep using the old allocation pages.

Restart kills all sessions → resets metadata → new distribution.

3. You cannot shrink TempDB properly

DBCC SHRINKFILE often fails for tempdb because active sessions hold pages.

Only restart clears everything.

4. You cannot reset mixed extents behavior

Even though SQL 2019 uses uniform extents, certain workloads still hit mixed extents metadata — fixed only at startup.

⭐ **Final Clear Statement**

NO online method can completely remove PAGELATCH contention.

You can only *reduce* or *delay* it.

The only *real fix* is:

- **Add equal tempdb files**
- **Restart SQL Server**
- **Contention disappears**