SQL Server TempDB contention refers to performance bottlenecks in the TempDB database, caused by multiple sessions or queries competing for the same resources in TempDB. TempDB is a system database used by SQL Server for temporary storage needs, such as temporary tables, result sets, sorting operations, and version stores.

Contention in TempDB can lead to significant performance issues in SQL Server environments.

Causes of TempDB Contention

1. **Metadata Contention**:
   - **PFS (Page Free Space)**: These pages track space allocation for each 8KB data page. If many sessions try to allocate or deallocate space in TempDB simultaneously, they may contend for access to these PFS pages.
   - **SGAM (Shared Global Allocation Map)**: These pages track mixed extent allocations (when small objects share space). High contention on SGAM pages can occur in workloads with frequent object creation.
   - **GAM (Global Allocation Map)**: These pages track extent allocations (groups of eight pages). Contention here is less common but can occur under very high workloads.

2. **Object Allocation Contention**:
   - This occurs when multiple sessions try to create or drop temporary objects, such as temp tables or table variables, at the same time. These operations require TempDB pages to be allocated or deallocated, leading to contention on allocation pages like PFS, GAM, and SGAM.

3. **Version Store Contention**:
   - The version store in TempDB is used to track row versions for snapshot isolation, read-committed snapshot isolation, and certain triggers. Heavy use of these isolation levels can cause contention when SQL Server tries to store row versions.

4. **I/O Contention**:
   - TempDB can experience I/O bottlenecks, particularly if it’s hosted on slower disks. High disk latency or throughput issues will cause performance degradation when SQL Server reads from or writes to TempDB files.

Symptoms of TempDB Contention
- **High Wait Times**: When contention occurs, queries will experience high wait times, particularly on `PAGEIOLATCH_XX`, `PAGELATCH_EX`, or `PAGELATCH_SH` waits, which indicate contention on TempDB pages.
- **Slow Performance for Queries Using TempDB**: Queries that rely heavily on TempDB (like those using temporary tables, table variables, or sorts) will perform slower.
- **High CPU Usage**: The system may experience high CPU usage due to repeated retries for TempDB page allocation.
- **Blocking**: Sessions may block each other while waiting to acquire TempDB pages.

Identifying TempDB Contention

You can identify TempDB contention by monitoring:

- **Wait statistics**: Look for specific wait types like `PAGELATCH_EX` or `PAGELATCH_SH` on `tempdb` database pages.

- **Dynamic Management Views (DMVs)**: Query DMVs such as `sys.dm_exec_requests`, `sys.dm_db_file_space_usage`, and `sys.dm_os_wait_stats` to find sessions waiting on TempDB resources.

Ways to Mitigate TempDB Contention

1. **Increase the Number of TempDB Data Files**:
   - A common recommendation is to increase the number of TempDB data files to match the number of CPU cores (up to 8 files), or at least to 1 data file per 2-4 cores. This reduces contention on allocation pages.

2. **Evenly Size TempDB Files**:
   - Ensure that all TempDB data files are of equal size to avoid SQL Server disproportionately using one file, which can lead to contention.

3. **Use Trace Flags (If Needed)**:
   - Trace flags like **TF 1118** prevent SQL Server from using mixed extents, which reduces contention on SGAM pages.

4. **Monitor and Optimize TempDB Usage**:
   - Review how applications use TempDB (e.g., reduce unnecessary use of temporary tables, avoid large sorts, and optimize queries). This helps lower the overall load on TempDB.

5. **Fast Storage**:
   - Ensure TempDB is placed on fast I/O subsystems such as SSDs to reduce I/O contention.

6. **Memory Optimization**:
   - Increase the memory available to SQL Server to reduce the need for TempDB (e.g., by reducing excessive use of sorts or spills to TempDB from hash joins).

By managing and optimizing TempDB, you can significantly reduce contention and improve overall SQL Server performance.