sp_CheckTempdb: Check SQL Server tempdb for Performance Issues.

what it does, why it matters, and how to use it. This also highlight the main findings and capabilities.

Source:
https://straightpathsql.com/archives/2025/05/introducing-sp_checktempdb-check-your-sql-server-tempdb-for-performance-issues/
Author: **Jeff Iannucci**

✅ **What is sp_CheckTempdb — and why was it created**

- The article begins by reminding that the system database tempdb in SQL Server is used not only for temporary tables, but also for **table variables, cursors, sorts, joins, memory spills, integrity checks, version store, etc.**
- Because tempdb is constantly used by many internal operations, **its correct configuration and health are critical for overall SQL Server performance**.
- The authors at Straight Path Solutions recognized a gap: while there are many community tools for performance analysis, there was no simple, free tool focused solely on **checking tempdb configuration and usage health** — so they built one: **sp_CheckTempdb**.
- The goal: allow DBAs to quickly get an overview of tempdb configuration, detect misconfigurations, spot usage problems, or investigate performance issues related to tempdb.

🔎 **What sp_CheckTempdb Does — Modes & Checks**

The procedure provides multiple "modes," each returning different kinds of information, depending on what you want to examine.

- **@Mode = 0:** Checks for potential configuration or performance issues with tempdb — e.g., misconfigured files, poor file size/growth, slow I/O, file placement, number of files, uneven file size, etc.
- **@Mode = 1:** Reports current configuration of all tempdb files — similar to what you see in SSMS "File Properties": file IDs, logical names, types (data vs log), filegroup, sizes, autogrowth settings, max sizes, physical file paths, etc.
- **@Mode = 2:** Provides a detailed view of **current tempdb usage** — for both data and log files; shows how much space is used, how much is free, breakdown by object usage (internal, version store, session usage, etc.). Useful when investigating "who/what is using tempdb now."
- **@Mode = 3:** Checks for **metadata or allocation contention** in tempdb (e.g., PFS/GAM/SGAM contention, latch waits) — these are classic problems when tempdb is misconfigured for multi-core/multi-user systems.

Additional parameters you can pass to customize the output:

- @Size — to report sizes in MB (default) or GB.
- @UsagePercent — threshold (default 50%) for flagging tempdb files that are heavily used.
- @AvgReadStallMs and @AvgWriteStallMs — thresholds for read/write latency (default 20 ms), to help detect slow I/O or storage bottlenecks.

If you execute sp_CheckTempdb without specifying @Mode, by default it returns both Mode 1 (configuration overview) and Mode 0 (potential issues) — a good "baseline health check."

✅ **Requirements to Use sp_CheckTempdb**

- Your SQL Server instance must be **SQL Server 2014 or newer**. Older versions may skip certain checks because the required DMVs are missing.
- You need the **VIEW SERVER STATE** permission — because the tool uses DMVs and system tables to collect data across the instance.

🧰 **Why It's Useful — When and How to Use It**

- **Proactive health check:** Run regularly (e.g., monthly, quarterly) to spot misconfigurations before they cause performance issues — e.g., wrong number of files, uneven growth settings, poor file placement. This helps avoid tempdb contention or unexpected I/O problems.
- **Troubleshooting:** When you suspect performance issues, blocking, slow queries, or tempdb-related errors — run Mode 2 or 3 to see which sessions are consuming tempdb, or whether there's I/O latency or allocation contention.
- **Review after changes:** After server upgrades, configuration changes, or workload shifts — use sp_CheckTempdb to validate that tempdb is still properly configured for the workload.

📝 **Key Takeaways (as the Author Does)**

- tempdb is more than "just temp tables." It underpins many operations — sorting, version store (for snapshot isolation or ADR), internal operations, memory spills, etc. If tempdb is poorly configured, overall performance suffers — often silently until a spike or failure.
- Simple but common mistakes — wrong number of data files, uneven sizes, poor autogrowth settings, slow IO, log vs data file mismatch — can cause major performance bottlenecks. A tool like sp_CheckTempdb helps quickly identify those.
- The tool is free, easy to install, requires minimal permissions (VIEW SERVER STATE), and offers flexible reporting. It is a **useful addition** to the "DBA toolbox" for both proactive maintenance and emergency troubleshooting.

Bottom Line"

If you manage SQL Server instances — especially ones under heavier load, with many users, or dynamic workloads — tempdb deserves regular attention. The newly released sp_CheckTempdb from Straight Path Solutions provides a **quick, effective, and free** way to audit and monitor tempdb health and usage. It helps you catch configuration mistakes, avoid contention, monitor usage, and troubleshoot performance issues — all with minimal effort.

Here's an example of what output from **sp_CheckTempdb** might look like for each mode, based on typical data. This will help you understand what kind of info the tool returns and how to interpret it.

🟢 Sample Output for sp_CheckTempdb

Mode 0 — Configuration & Performance Issues Check

Check Category	Issue Detected	Recommendation	Severity
Number of Data Files	Only 2 data files configured (CPU count = 16)	Increase to 8 data files (recommended CPU/2 to CPU/4)	Medium
File Size Disparity	Data files have uneven sizes (1GB vs 3GB)	Resize files to be equal in size	Medium
Autogrowth Settings	Autogrowth set to 1MB (too small)	Set autogrowth to at least 64MB	Low
Log File vs Data File Size	Log file size larger than data files	Resize log file appropriately	Low
I/O Latency	Average write stall time = 35ms (>20ms threshold)	Investigate storage subsystem for latency issues	High

Mode 1 — TempDB File Configuration Overview

File ID	Logical Name	Type	Filegroup	Size (MB)	Max Size	Autogrowth	Physical Path
1	tempdev	Data	PRIMARY	2048	Unlimited	64MB	D:\SQLData\tempdb.mdf
2	templog	Log	NULL	1024	Unlimited	64MB	D:\SQLLogs\templog.ldf
3	tempdb_mssql_2	Data	PRIMARY	2048	Unlimited	64MB	D:\SQLData\tempdb_mssql_2.ndf
4	tempdb_mssql_3	Data	PRIMARY	2048	Unlimited	64MB	D:\SQLData\tempdb_mssql_3.ndf

Mode 2 — Current TempDB Usage Breakdown

File Name	Total Size (MB)	Used Space (MB)	Free Space (MB)	Usage %	Allocation by Object
tempdev	2048	1024	1024	50%	Internal objects (30%), User objects (20%)
templog	1024	600	424	59%	Log entries
tempdb_mssql_2	2048	1500	548	73%	Version store (40%), User objects (33%)
tempdb_mssql_3	2048	1200	848	59%	Internal objects (45%), Version store (14%)

Mode 3 — TempDB Metadata Contention & Wait Stats

Contention Type	Wait Type	Wait Count	Avg Wait Time (ms)	Recommendation
PFS Latch	PAGELATCH_UP	15000	5	Add more data files to reduce contention
GAM Latch	PAGELATCH_EX	3000	10	Check for uneven file size or placement
SGAM Latch	PAGELATCH_UP	4000	7	Review autogrowth settings and file counts

How to Interpret

- **Mode 0:** Highlights potential configuration problems and performance risks — these should be prioritized for fixing.
- **Mode 1:** Gives a snapshot of the current tempdb files for inventory and sizing verification.
- **Mode 2:** Shows who/how tempdb space is being used currently — useful during troubleshooting.
- **Mode 3:** Shows latch contention issues — a classic tempdb bottleneck sign — and points to misconfiguration like too few files or skewed sizes.

Install and run sp_CheckTempdb in your SQL Server environment, plus sample T-SQL commands to use it and generate reports:

🟢 **Installing sp_CheckTempdb**

1. **Download or create the stored procedure** script from the official source (e.g., Straight Path SQL website). https://github.com/Straight-Path-Solutions/sp_CheckTempdb
2. Open **SQL Server Management Studio (SSMS)** or your preferred SQL editor.
3. Connect to the SQL Server instance where you want to check tempdb.
4. Execute the script to **create the stored procedure** in the master database (or a central DBA tools database).

🟢 **Sample T-SQL to Install (Example snippet)**

```
USE master;
GO
-- Create or replace the stored procedure
CREATE PROCEDURE sp_CheckTempdb
(
   @Mode INT = 0,
   @Size VARCHAR(2) = 'MB',
   @UsagePercent INT = 50,
   @AvgReadStallMs INT = 20,
   @AvgWriteStallMs INT = 20
)
AS
BEGIN
   -- Procedure logic here (download from official source)
   -- This is just a placeholder snippet for example
   PRINT 'sp_CheckTempdb installed successfully.'
END;
GO
```


Note: For the full procedure code, download the script from the official sp_CheckTempdb page.

🟢 Running sp_CheckTempdb

1. Run default checks (Mode 0 and Mode 1 combined):

```
EXEC sp_CheckTempdb;
```

2. Get detailed configuration overview (Mode 1):

```
EXEC sp_CheckTempdb @Mode = 1;
```

3. See current usage breakdown (Mode 2):

```
EXEC sp_CheckTempdb @Mode = 2;
```

4. Check for tempdb metadata contention (Mode 3):

```
EXEC sp_CheckTempdb @Mode = 3;
```

5. Customize thresholds (e.g., usage percent 75%, read stall 30 ms):

```
EXEC sp_CheckTempdb @Mode = 0, @UsagePercent = 75, @AvgReadStallMs = 30, @AvgWriteStallMs = 30;
```

🟢 Generating a Sample Report

You can run the default:

```
EXEC sp_CheckTempdb;
```

and **export the results** from SSMS as CSV or Excel:

- Right-click result grid → **Save Results As…**
- Choose CSV or Excel file to share with your team.

Or you can script automated runs and export with PowerShell or SQL Agent jobs to regularly check tempdb health.

🟢 Additional Tips

- Run with **VIEW SERVER STATE** permissions.
- Schedule runs monthly or after configuration changes.
- Combine output with other health checks for full server insight.
- Use findings to tune number of tempdb files, growth settings, and investigate storage I/O.

sp_CheckTempdb stored procedure script and a PowerShell script example to automate running it and emailing the report.

1️⃣ Full sp_CheckTempdb Stored Procedure Script

Because the full script is quite long, I'm providing you a **downloadable SQL file** link from the official source, plus a simplified version snippet here to get started.

Official download page: https://straightpathsql.com/tool/sp_checktempdb/

Simplified example snippet (create the procedure in your SQL Server):

```
USE master;
GO
CREATE PROCEDURE sp_CheckTempdb
(
   @Mode INT = 0,
   @Size VARCHAR(2) = 'MB',
   @UsagePercent INT = 50,
   @AvgReadStallMs INT = 20,
   @AvgWriteStallMs INT = 20
```

```
)
AS
BEGIN
  SET NOCOUNT ON;
  -- Simplified example only: replace with full script from official site
  IF @Mode = 0 OR @Mode IS NULL
  BEGIN
    PRINT 'Running tempdb configuration and performance checks...';
    -- Add checks here...
  END
  ELSE IF @Mode = 1
  BEGIN
    PRINT 'Reporting tempdb file configuration...';
    -- Add file info query here...
  END
  ELSE IF @Mode = 2
  BEGIN
    PRINT 'Reporting current tempdb usage...';
    -- Add usage query here...
  END
  ELSE IF @Mode = 3
  BEGIN
    PRINT 'Checking tempdb allocation contention...';
    -- Add contention query here...
  END
END;
GO
```


Note: To get the full detailed script with all logic and queries, download from the official site above.

2 PowerShell Script to Run sp_CheckTempdb and Email Results

Below is a **PowerShell script example** that:

- Connects to your SQL Server
- Runs sp_CheckTempdb (mode 0 by default)
- Saves output to a CSV file
- Emails the CSV as attachment

PowerShell Script

```
# Define parameters
$serverInstance = "YOUR_SERVER_INSTANCE"
$database = "master"
$outputCsvPath = "C:\Temp\TempdbCheck.csv"
$emailFrom = "sqlcheck@yourdomain.com"
$emailTo = "dba_team@yourdomain.com"
$emailSubject = "Tempdb Health Check Report"
$smtpServer = "smtp.yourdomain.com"

# Import SQL Server module (if needed)
```

```
Import-Module SqlServer

# Define the query to run
$query = "EXEC sp_CheckTempdb @Mode = 0;"

# Run the query and export results to CSV
Invoke-Sqlcmd -ServerInstance $serverInstance -Database $database -Query $query | Export-Csv -Path
$outputCsvPath -NoTypeInformation

# Prepare email message
$mailMessage = New-Object System.Net.Mail.MailMessage
$mailMessage.From = $emailFrom
$mailMessage.To.Add($emailTo)
$mailMessage.Subject = $emailSubject
$mailMessage.Body = "Attached is the latest tempdb health check report."
$mailMessage.Attachments.Add($outputCsvPath)

# Setup SMTP client and send email
$smtp = New-Object Net.Mail.SmtpClient($smtpServer)
$smtp.Send($mailMessage)

Write-Host "Tempdb health check report generated and emailed successfully."
```


How to Use the PowerShell Script

1. Modify variables:
    - $serverInstance to your SQL Server instance name
    - $outputCsvPath to a valid path on your machine
    - Email fields: $emailFrom, $emailTo, $smtpServer
2. Run the script in PowerShell with appropriate permissions and network access to SMTP.
3. Schedule the script in **Windows Task Scheduler** or similar for automated reporting.