4 – How #Tables Affect TempDB

Part 2: Many Queries at Once

4 p1

TempDB tables come & go.

	New objects created	Objects dropped
System databases	Almost never	Almost never
User databases	Every now and then	Every now and then
TempDB	CONSTANTLY	CONSTANTLY

4 p2

I'll create an extreme example.

This stored proc only has one job: populate a #table.

```
USE StackOverflow2013;
GO
CREATE OR ALTER PROCEDURE dbo.TempTable AS
    SELECT TOP 1000 Id, AboutMe
    INTO #t1
    FROM dbo.Users WITH (NOLOCK)
    OPTION (MAXDOP 1);
GO
```

4 p3

When I run just one, it's fast:

We don't wait on disk: all of this happens in memory.

Messages

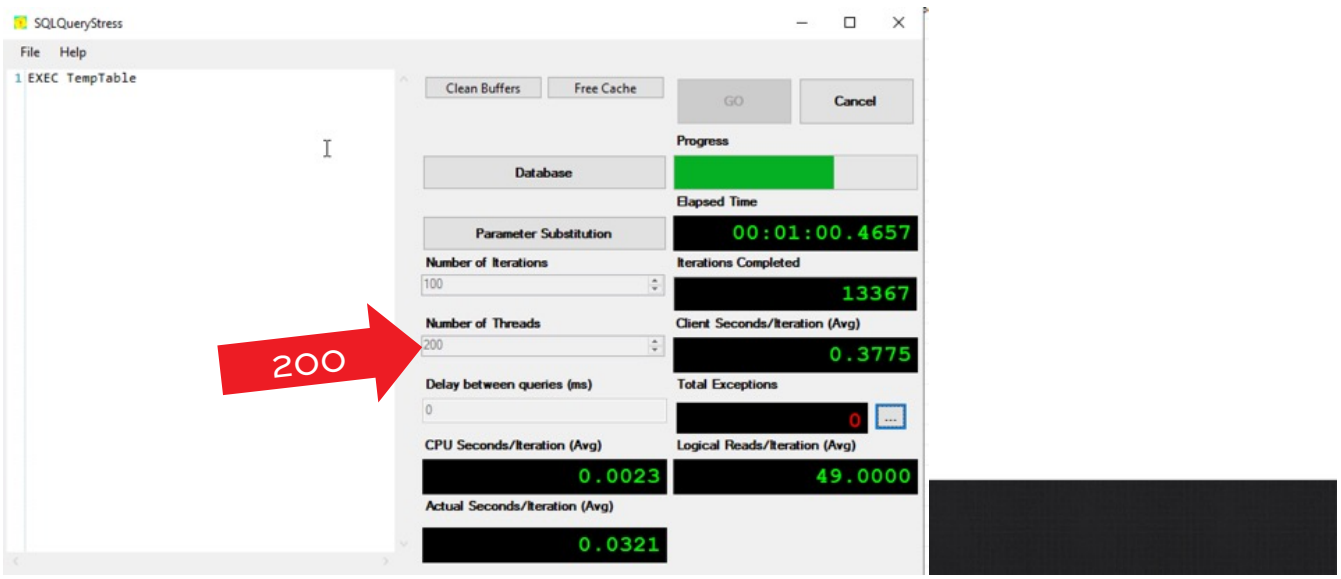
```
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 0 ms.
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 0 ms.
Table 'Users'. Scan count 1, logical reads 49, physical reads 0,

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 3 ms.
```

Fast

4 p4

But when I run 200x at a time...



This causes a problem.

SQL Server was never really designed for constant creation/dropping of objects.

To create or drop an object, SQL Server has to write to special system pages.

- PFS: Page Free Space pages
- SGAM: Shared Global Allocation Map pages
- GAM: Global Allocation Map pages

To lock these, SQL Server uses a more lightweight form of locking called page latching. (PAGELATCH)

sp_WhoIsActive shows blocking

```
1  sp_WhoIsActive
```

200 %

Results | Messages

	dd hh:mm:ss.mss	session_id	sql_text	login_name	wait_info	CPU	tempdb_allocations	tempdb_current	blocking_session_id
1	00 00:00:01.956	58	<?query -- INSERT INTO #Upd...	SQL2019\Brent	NULL	1,566	224	72	NULL
2	00 00:00:00.983	80	<?query -- EXEC TempTable --?>	SQL2019\Brent	(81ms)PAGELATCH_UP:tempdb:4(GAM)	3	176	32	95
3	00 00:00:00.796	74	<?query -- SELECT TOP 1000 I...	SQL2019\Brent	(78ms)PAGELATCH_UP:tempdb:4(GAM)	2	160	48	95
4	00 00:00:00.773	210	<?query -- EXEC TempTable --?>	SQL2019\Brent	NULL	0	0	0	NULL
5	00 00:00:00.753	217	<?query -- SELECT TOP 1000 I...	SQL2019\Brent	(108ms)PAGELATCH_UP:tempdb:4(GAM)	2	160	64	103
6	00 00:00:00.746	193	<?query -- SELECT TOP 1000 I...	SQL2019\Brent	(79ms)PAGELATCH_UP:tempdb:4(GAM)	2	160	72	210
7	00 00:00:00.743	200	<?query -- SELECT TOP 1000 I...	SQL2019\Brent	(75ms)PAGELATCH_UP:tempdb:4(GAM)	2	160	88	210
8	00 00:00:00.743	115	<?query -- EXEC TempTable --?>	SQL2019\Brent	(69ms)PAGELATCH_UP:tempdb:4(GAM)	2	160	8	95
9	00 00:00:00.740	103	<?query -- SELECT TOP 1000 I...	SQL2019\Brent	(146ms)PAGELATCH_UP:tempdb:4(GAM)	0	0	0	95
10	00 00:00:00.740	151	<?query -- SELECT TOP 1000 I...	SQL2019\Brent	(28ms)PAGELATCH_UP:tempdb:4(GAM)	2	160	72	95
11	00 00:00:00.736	154	<?query -- EXEC TempTable --?>	SQL2019\Brent	(15ms)PAGELATCH_UP:tempdb:4(GAM)	2	160	24	95
12	00 00:00:00.716	158	<?query -- SELECT TOP 1000 I...	SQL2019\Brent	(120ms)PAGELATCH_UP:tempdb:4(GAM)	2	160	64	95
13	00 00:00:00.716	197	<?query -- SELECT TOP 1000 I...	SQL2019\Brent	(116ms)PAGELATCH_UP:tempdb:4(GAM)	2	160	96	210
14	00 00:00:00.716	124	<?query -- EXEC TempTable --?>	SQL2019\Brent	(120ms)PAGELATCH_UP:tempdb:4(GAM)	2	160	32	95
15	00 00:00:00.713	150	<?query -- SELECT TOP 1000 I...	SQL2019\Brent	(107ms)PAGELATCH_UP:tempdb:4(GAM)	2	176	152	95
16	00 00:00:00.706	178	<?query -- SELECT TOP 1000 I...	SQL2019\Brent	(99ms)PAGELATCH_UP:tempdb:4(GAM)	2	176	88	-4
17	00 00:00:00.700	121	<?query -- SELECT TOP 1000 I...	SQL2019\Brent	(89ms)PAGELATCH_UP:tempdb:4(GAM)	2	160	48	95
18	00 00:00:00.700	72	<?query -- SELECT TOP 1000 I...	SQL2019\Brent	(88ms)PAGELATCH_UP:tempdb:4(GAM)	2	176	56	95
19	00 00:00:00.696	219	<?query -- SELECT TOP 1000 I...	SQL2019\Brent	(99ms)PAGELATCH_UP:tempdb:4(GAM)	4	176	56	103
20	00 00:00:00.693	212	<?query -- SELECT TOP 1000 I...	SQL2019\Brent	(84ms)PAGELATCH_UP:tempdb:4(GAM)	2	176	96	103

sp_WhoIsActive shows blocking with PAGELATCH_UP waits

```
1  sp_WhoIsActive
```

200 %

Results | Messages

	dd hh:mm:ss.mss	session_id	sql_text	login_name	wait_info	CPU	tempdb_allocations	tempdb_current	blocking_session_id
1	00 00:00:01.956	58	<?query -- INSERT INTO #Upd...	SQL2019\Brent	NULL	1,566	224	72	NULL
2	00 00:00:00.983	80	<?query -- EXEC TempTable --?>	SQL2019\Brent	(81ms)PAGELATCH_UP:tempdb:4(GAM)	3	176	32	95
3	00 00:00:00.796	74	<?query -- SELECT TOP 1000 I...	SQL2019\Brent	(78ms)PAGELATCH_UP:tempdb:4(GAM)	2	160	48	95
4	00 00:00:00.773	210	<?query -- EXEC TempTable --?>	SQL2019\Brent	NULL	0	0	0	NULL
5	00 00:00:00.753	217	<?query -- SELECT TOP 1000 I...	SQL2019\Brent	(108ms)PAGELATCH_UP:tempdb:4(GAM)	2	160	64	103
6	00 00:00:00.746	193	<?query -- SELECT TOP 1000 I...	SQL2019\Brent	(79ms)PAGELATCH_UP:tempdb:4(GAM)	2	160	72	210
7	00 00:00:00.743	200	<?query -- SELECT TOP 1000 I...	SQL2019\Brent	(75ms)PAGELATCH_UP:tempdb:4(GAM)	2	160	88	210
8	00 00:00:00.743	115	<?query -- EXEC TempTable --?>	SQL2019\Brent	(69ms)PAGELATCH_UP:tempdb:4(GAM)	2	160	8	95
9	00 00:00:00.740	103	<?query -- SELECT TOP 1000 I...	SQL2019\Brent	(146ms)PAGELATCH_UP:tempdb:4(GAM)	0	0	0	95
10	00 00:00:00.740	151	<?query -- SELECT TOP 1000 I...	SQL2019\Brent	(28ms)PAGELATCH_UP:tempdb:4(GAM)	2	160	72	95
11	00 00:00:00.736	154	<?query -- EXEC TempTable --?>	SQL2019\Brent	(15ms)PAGELATCH_UP:tempdb:4(GAM)	2	160	24	95
12	00 00:00:00.716	158	<?query -- SELECT TOP 1000 I...	SQL2019\Brent	(120ms)PAGELATCH_UP:tempdb:4(GAM)	2	160	64	95
13	00 00:00:00.716	197	<?query -- SELECT TOP 1000 I...	SQL2019\Brent	(116ms)PAGELATCH_UP:tempdb:4(GAM)	2	160	96	210
14	00 00:00:00.716	124	<?query -- EXEC TempTable --?>	SQL2019\Brent	(120ms)PAGELATCH_UP:tempdb:4(GAM)	2	160	32	95
15	00 00:00:00.713	150	<?query -- SELECT TOP 1000 I...	SQL2019\Brent	(107ms)PAGELATCH_UP:tempdb:4(GAM)	2	176	152	95
16	00 00:00:00.706	178	<?query -- SELECT TOP 1000 I...	SQL2019\Brent	(99ms)PAGELATCH_UP:tempdb:4(GAM)	2	176	88	-4
17	00 00:00:00.700	121	<?query -- SELECT TOP 1000 I...	SQL2019\Brent	(89ms)PAGELATCH_UP:tempdb:4(GAM)	2	160	48	95
18	00 00:00:00.700	72	<?query -- SELECT TOP 1000 I...	SQL2019\Brent	(88ms)PAGELATCH_UP:tempdb:4(GAM)	2	176	56	95
19	00 00:00:00.696	219	<?query -- SELECT TOP 1000 I...	SQL2019\Brent	(99ms)PAGELATCH_UP:tempdb:4(GAM)	4	176	56	103
20	00 00:00:00.693	212	<?query -- SELECT TOP 1000 I...	SQL2019\Brent	(84ms)PAGELATCH_UP:tempdb:4(GAM)	2	176	96	103

Blocking

Long delays

GAM Page

```
1  sp_BlitzFirst @ExpertMode = 1, @Seconds = 60
```

200 %

Results | Messages

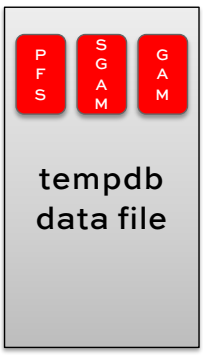
run_date	elapsed_time	session_id	database_name	query_text	query_plan	live_query_plan	query_cost	status	wait_info	top_session_waits	blocking_session_id	op

	Priority	FindingsGroup	Finding	Details	HowToStopIt
1	0	sp_BlitzFirst 2...	From Your Community Volunteers	<?ClickToSeeDetails -- We hope you found this tool useful. -- ?>	NULL
2	200	Wait Stats	PAGELATCH_UP	<?ClickToSeeDetails -- For 191 seconds over the last 60 seconds, SQL Server was waiting on this particul...	<?ClickToSe
3	250	Server Info	Batch Requests per Sec	<?ClickToSeeDetails -- 441.20 -- ?>	NULL
4	250	Server Info	CPU Utilization	<?ClickToSeeDetails -- 32%. Ring buffer details: <Record id="63" type="RING_BUFFER_SCHEDULER ...	NULL
5	250	Server Info	SQL Compilations per Sec	<?ClickToSeeDetails -- 221 -- ?>	NULL
6	250	Server Info	SQL Re-Compilations per Sec	<?ClickToSeeDetails -- 0 -- ?>	NULL
7	250	Server Info	Wait Time per Core per Sec	<?ClickToSeeDetails -- 0.85 -- ?>	NULL
8	251	Server Info	Database Count	<?ClickToSeeDetails -- 4 -- ?>	NULL

Top wait

	Pattern	Sample Ended		wait_type	wait_category	Wait Time (Seconds)	Per Core Per Second
1	WAIT STATS	2020-11-29 05:07:43.8082576 -08:00		PAGELATCH_UP	Buffer Latch	191.6	0.8
2	WAIT STATS	2020-11-29 05:07:43.8082576 -08:00	60	PAGELATCH_EX	Buffer Latch	6.5	0.0
3	WAIT STATS	2020-11-29 05:07:43.8082576 -08:00	60	PAGELATCH_SH	Buffer Latch	3.2	0.0
4	WAIT STATS	2020-11-29 05:07:43.8082576 -08:00	60	LOGMGR_FLUSH	Tran Log IO	0.9	0.0
5	WAIT STATS	2020-11-29 05:07:43.8082576 -08:00	60	MEMORY_ALLOCATION_EXT	Memory	0.6	0.0

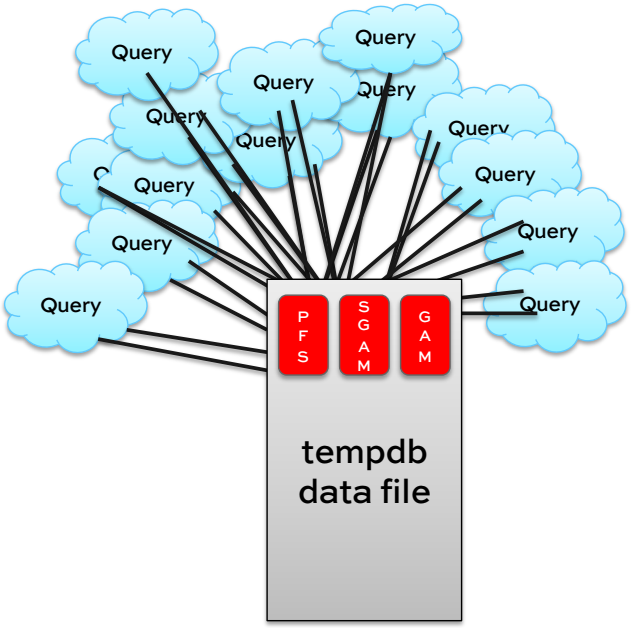
Normal databases have 1 data file.



This file will have one of each of these special PFS, SGAM, GAM pages.

(It gets more pages as its size grows.)

This is fine for regular databases, but...



Not in TempDB.

4 p11

This isn't a disk storage problem.

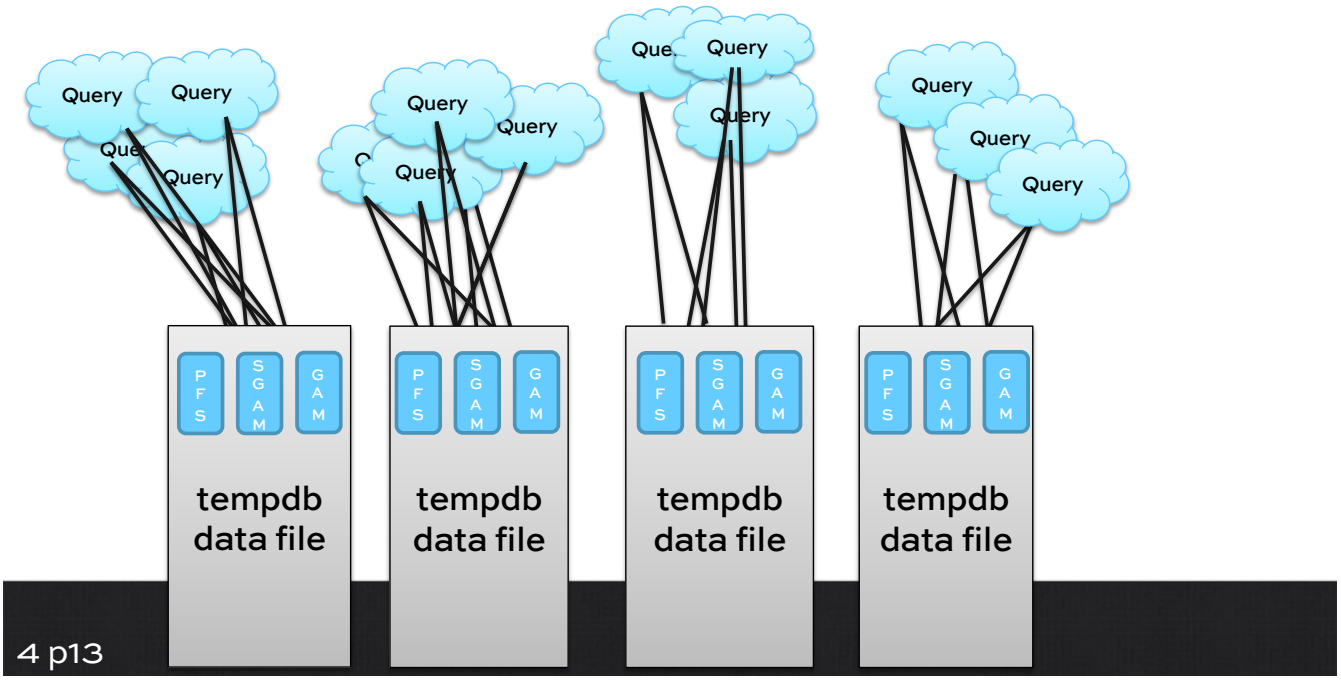
It's about contention for a system page.

Like all database pages, their primary home is on disk, but in this case, they're cached in RAM.

We have these contention problems even when the data is stored in RAM.

The fix is a little weird: create multiple data files.

4 p12



SQL balances work across files.

Each data file has its own PFS/SGAM/GAM pages.

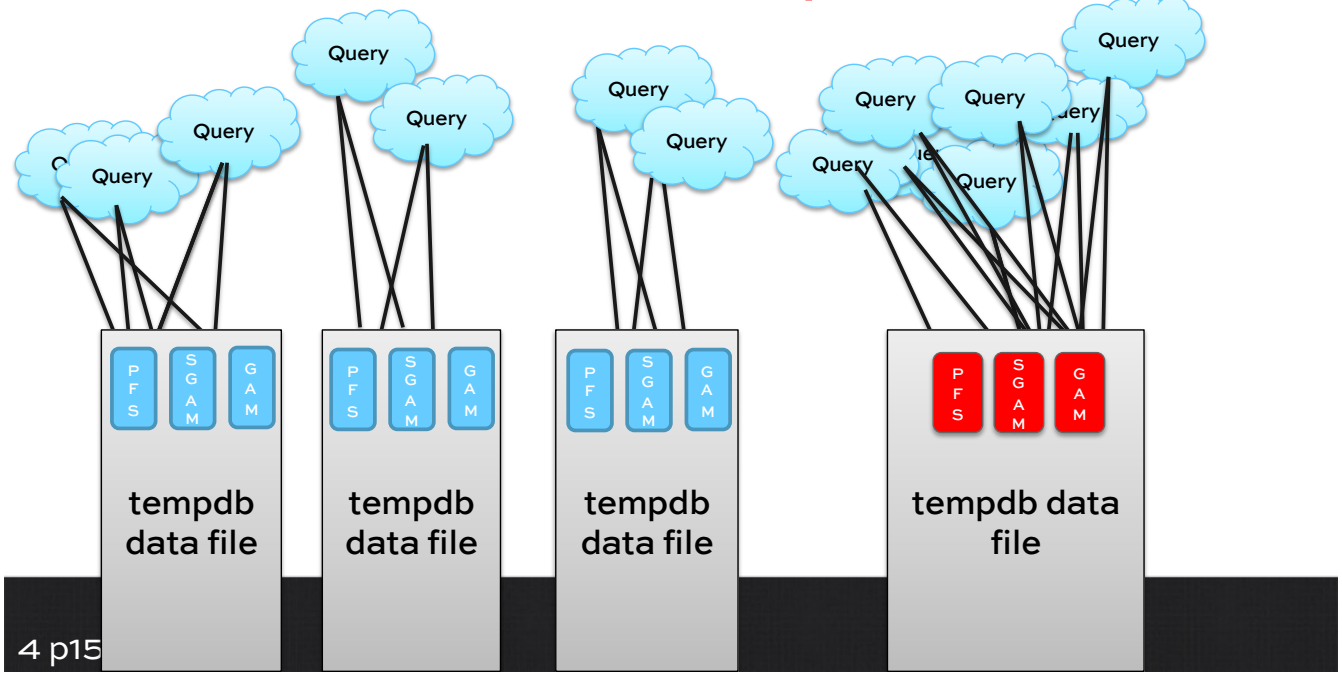
As load increases, SQL Server will automatically balance the temp table creations across data files.

This has nothing to do with storage throughput: we just need more PFS/SGAM/GAM pages!

SQL balances work across data files based on how full each data file is. (This is called “proportional fill.”)

4 p14

File sizes need to be equal, or else...



How to configure TempDB

Create 4-8 equally sized data files.

The exact number is less important than just “more.”

Not sure about the size?
Start with 25% of the total data size on the server.

Leave autogrowth on (because people & SQL Server use it, and running out of space is bad.)

Don’t shrink them.

4 p16

How to re-configure a server.

If it only has 1 TempDB data file:

Take that size, divide it by 4. That's your new file size.

Shrink the existing data file down to the new file size.

Add 3 more data files with that same size.

Restart the SQL Server service.

4 p17

More mitigation methods

SQL Server 2014 & prior, consider:

- Trace flag 1117: grows all data files in a filegroup equally (but also applies to user databases)
- Trace flag 1118: doesn't use mixed extents in TempDB, alleviates some of the pressure

SQL Server 2019 & newer:

- Memory-Optimized TempDB Metadata
  (but be careful here: there've been bugs and feature interop bugs w/columnstore)

4 p18

Sometimes, this still isn't enough

If you have thousands of queries per second that:

- All create temp tables
- Especially if they don't get table reuse (like we talked about in the last module)
- You've already added TempDB data files
- You're still seeing PAGELATCH% waits specifically in TempDB GAM/SGAM pages

Then let's explore changing the code.

4 p19

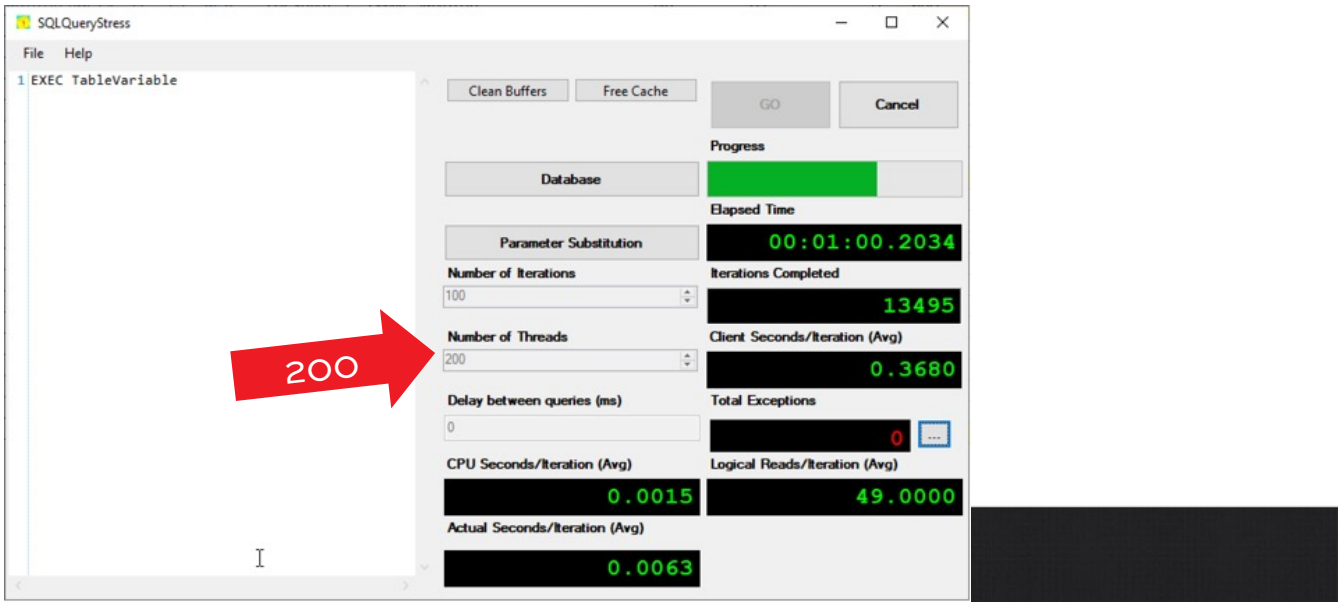
How about table variables?

```
CREATE OR ALTER PROCEDURE dbo.TableVariable AS
BEGIN
    DECLARE @TableVariable TABLE (Id INT, AboutMe NVARCHAR(MAX));

    INSERT INTO @TableVariable (Id, AboutMe)
    SELECT TOP 1000 Id, AboutMe
    FROM dbo.Users WITH (NOLOCK)
    OPTION (MAXDOP 1);
END
GO
```

4 p20

Same workload



Still has PAGELATCH waits and blocking, but...

```
1  sp_WhoIsActive
```

200 %

Results | Messages

	dd hh:mm:ss.mss	session_id	sql_text	login_name	wait_info	CPU	tempdb_allocations	tempdb_current	blocking_session_id
1	00 00:00:30.907	58	<?query -- WAITFOR TIME @Finish...	SQL2019\Brent	(29908ms)WAITFOR	981	232	56	NULL
2	00 00:00:00.120	75	<?query -- EXEC TableVariable --?>	SQL2019\Brent	(5ms)PAGELATCH_UP:tempdb:4(GAM)	2	160	8	240
3	00 00:00:00.120	118	<?query -- EXEC TableVariable --?>	SQL2019\Brent	(9ms)PAGELATCH_UP:tempdb:4(GAM)	2	160	32	240
4	00 00:00:00.120	174	<?query -- EXEC TableVariable --?>	SQL2019\Brent	(1ms)PAGELATCH_UP:tempdb:4(GAM)	2	160	8	103
5	00 00:00:00.120	184	<?query -- INSERT INTO @TableVar...	SQL2019\Brent	(5ms)PAGELATCH_UP:tempdb:4(GAM)	2	160	80	103
6	00 00:00:00.114	221	<?query -- INSERT INTO @TableVar...	SQL2019\Brent	(10ms)PAGELATCH_UP:tempdb:4(GAM)	1	160	104	103
7	00 00:00:00.114	103	<?query -- INSERT INTO @TableVar...	SQL2019\Brent	(10ms)PAGELATCH_UP:tempdb:4(GAM)	2	160	48	240
8	00 00:00:00.064	113	<?query -- INSERT INTO @TableVar...	SQL2019\Brent	(7ms)PAGELATCH_UP:tempdb:4(GAM)	1	160	160	240
9	00 00:00:00.064	68	<?query -- INSERT INTO @TableVar...	SQL2019\Brent	NULL	1	160	160	NULL
10	00 00:00:00.064	195	<?query -- INSERT INTO @TableVar...	SQL2019\Brent	NULL	1	152	152	103
11	00 00:00:00.064	200	<?query -- INSERT INTO @TableVar...	SQL2019\Brent	(8ms)PAGELATCH_UP:tempdb:4(GAM)	1	160	120	103
12	00 00:00:00.064	212	<?query -- INSERT INTO @TableVar...	SQL2019\Brent	(5ms)PAGELATCH_UP:tempdb:4(GAM)	1	128	128	103
13	00 00:00:00.054	187	<?query -- INSERT INTO @TableVar...	SQL2019\Brent	(8ms)PAGELATCH_UP:tempdb:4(GAM)	0	128	128	103
14	00 00:00:00.054	194	<?query -- INSERT INTO @TableVar...	SQL2019\Brent	(9ms)PAGELATCH_UP:tempdb:4(GAM)	0	120	120	103
15	00 00:00:00.054	231	<?query -- INSERT INTO @TableVar...	SQL2019\Brent	(9ms)PAGELATCH_UP:tempdb:4(GAM)	1	136	136	103

4 p22

There's way less of it overall.

```
1  sp_BlitzFirst @ExpertMode = 1, @Seconds = 60
```

200 %

Results | Messages

run_date	elapsed_time	session_id	database_name	query_text	query_plan	live_query_plan	query_cost	status	wait_info	top_session_waits	blocking_session_id	op

	Priority	FindingsGroup	Finding	Details	HowToStopIt
1	0	sp_BlitzFirst 2020-11-1...	From Your Community Volunteers	<?ClickToSeeDetails -- We hope you found this tool useful. -- ?>	NULL
2	1	No Problems Found	From Your Community Volunteers	<?ClickToSeeDetails -- Try running our more in-depth checks with sp_Blitz, or there may not b...	NULL
3	250	Server Info	Batch Requests per Sec	<?ClickToSeeDetails -- 446.43 -- ?>	NULL
4	250	Server Info	CPU Utilization	<?ClickToSeeDetails -- 27%. Ring buffer details: <Record id="67" type="RING_BUFFER_SC...	NULL
5	250	Server Info	SQL Compilations	<?ClickToSeeDetails -- 223 -- ?>	NULL
6	250	Server Info	SQL Re-Compilat	<?ClickToSeeDetails -- 0 -- ?>	NULL
7	250	Server Info	Wait Time per Core per	0.12 -- ?>	NULL
8	251	Server Info	Database Count		NULL

Temp tables had ~190 secs

	Pattern	Sample Ended	Seconds Sample	wait_type		Wait Time (Seconds)	Per Core Per Second
1	WAIT STATS	2020-11-29 05:11:42.8375541 -08:00	60	PAGELATCH_UP	Buffer	25.5	0.1
2	WAIT STATS	2020-11-29 05:11:42.8375541 -08:00	60	PAGELATCH_SH	Buffer Latch	1.8	0.0
3	WAIT STATS	2020-11-29 05:11:42.8375541 -08:00	60	LOGMGR_FLUSH	Tran Log IO	0.9	0.0

Microsoft knows this is still a problem, so...

SQL Server 2019 added a fix.

SQL Server 2019 has a fix.

Microsoft took the In-Memory OLTP technology (Hekaton) and applied it to some of TempDB's internal tables.

```
/* SQL Server 2019 brings a new system-level
feature to help solve this: */
ALTER SERVER CONFIGURATION SET MEMORY_OPTIMIZED TEMPDB_METADATA=ON;
/* Restart the SQL Server instance for it to take
effect, then check it: */
SELECT SERVERPROPERTY('IsTempDBMetadataMemoryOptimized');
GO
```


Run the TempTable load test:

Temp tables' page latch waits are now competitive with table variables – without changing code:

	Priority	FindingsGroup	Finding	URL	Details
1	0	sp_BlitzFirst 2020-11-14 00:00:00.0000000 +00:00	From Your Community Volunteers	http://FirstResponderKit.org/	<?ClickToSeeDetails -- We hope you found this tool useful. -- ?>
2	50	In-Memory OLTP	Garbage Collection in Progress	https://BrentOzar.com/go/garbage/	<?ClickToSeeDetails -- 201788 rows processed (from SQL Server)
3	50	Query Problems	Plan Cache Erased Recently	http://www.BrentOzar.com/askbrent/plan-cache-era...	<?ClickToSeeDetails -- The oldest query in the plan cache was cre
4	250	Server Info	Batch Requests per Sec	http://www.BrentOzar.com/go/measure	<?ClickToSeeDetails -- 468.53 -- ?>
5	250	Server Info	CPU Utilization	http://www.BrentOzar.com/go/cpu	<?ClickToSeeDetails -- 32%. Ring buffer details: <Record id="2" ty
6	250	Server Info	ilations per Sec	http://www.BrentOzar.com/go/measure	<?ClickToSeeDetails -- 235 -- ?>
7	250	Server Info	Sec	http://www.BrentOzar.com/go/measure	<?ClickToSeeDetails -- 0 -- ?>

Temp tables had ~190 secs

	Pattern	Sample Ended	Seconds Sample	wait_type		Wait Time (Seconds)	Per Core Per Second	Signal Wait Time (Seconds)	Percent Signal Waits
1	WAIT STATS	2020-11-30 04:56:05.8814896 -08:00	59	PAGELATCH_UP	Buffer Latch	21.3	0.1	1.5	7.0
2	WAIT STATS	2020-11-30 04:56:05.8814896 -08:00	59	PAGELATCH_SH	Buffer Latch	1.6	0.0	1.0	62.5
3	WAIT STATS	2020-11-30 04:56:05.8814896 -08:00	59	LOGMGR_FLUSH	Tran Log IO	1.0	0.0	1.0	100.0
4	WAIT STATS	2020-11-30 04:56:05.8814896 -08:00	59	MEMORY_ALLOCATI...	Memory	0.6	0.0	0.0	0.0

There's also a new warning:

In-Memory OLTP shows up in the headline news:

	Priority	FindingsGroup	Finding	URL	Details
1	0	sp_BlitzFirst 2020-11-14 00:00:00.0000000 +00:00	From Your Community Volunteers	http://FirstResponderKit.org/	<?ClickToSeeDetails -- We hope you found this tool useful. -- ?>
2	50	In-Memory OLTP	Garbage Collection in Progress	https://BrentOzar.com/go/garbage/	<?ClickToSeeDetails -- 201788 rows processed (from SQL Server

```
<?ClickToSeeDetails --
201788 rows processed (from SQL Server YYYY XTP Garbage Collection:Rows processed/sec counter)
This can happen due to memory pressure (causing In-Memory OLTP to shrink its footprint) or
due to transactional workloads that constantly insert/delete data.
 -- ?>
```

4 p27

Table variable test may hit snags

In some of my tests, it actually ran slower:

Transactions failing/aborted

	Priority	FindingsGroup	Finding	Details
1	0	sp_BlitzFirst 2020-11-14 00:00:00.0000000 +00:00	From Your Community Volunteers	<?ClickToSeeDetails -- We hope you found this tool useful. -- ?>
2	50	In-Memory OLTP	Garbage Collection in Progress	<?ClickToSeeDetails -- 37599 rows processed from SQL Server YYYY XTP Garbage Collection:I
3	50	Query Problems	Plan Cache Erased Recently	<?ClickToSeeDetails -- The oldest query in the plan cache was created at Nov 30 2020 4:53AM
4	100	In-Memory OLTP	Transactions Aborted	<?ClickToSeeDetails -- 766 transactions aborted from SQL Server YYYY XTP Transactions:Tran
5	200	Wait Stats	PAGELATCH_UP	<?ClickToSeeDetails -- For 98 seconds over the last 60 seconds, SQL Server was waiting on this
6	250	Server Info	Batch Requests per Sec	<?ClickToSeeDetails -- 356.20 -- ?>
7	250	Server Info	CPU Utilization	<?ClickToSeeDetails -- 40%. Ring buffer details: <Record id="8" type="RING_BUFFER_SCHED
8	250	Server Info	SQL Compilations per Sec	<?ClickToSeeDetails -- 178 -- ?>

	Pattern	Sample Ended	Seconds Sample	wait_type	wait_category	Wait Time (Seconds)	Per Core Per Second	Sign
1	WAIT STATS	2020-11-30 05:01:52.0869917 -08:00	60	PAGELATCH_UP	Buffer Latch	98.2	0.4	1.5
2	WAIT STATS	2020-11-30 05:01:52.0869917 -08:00	60	PAGELATCH_SH	Buffer Latch	1.9	0.0	1.1
3	WAIT STATS	2020-11-30 05:01:52.0869917 -08:00	60	LOGM	Tran Log IO	0.6	0.0	0.6
4	WAIT STATS	2020-11-30 05:01:52.0869917 -08:00	60	CATION_EXT	Memory	0.4	0.0	0.0

Oof

Table variables had ~20 secs

4 p29

Other issues with the feature:

There have been a lot of issues with In-Memory TempDB since 2019 came out:

- You can't create columnstore indexes in TempDB
- Tons of KB articles – a sampling:
  - https://support.microsoft.com/en-us/help/4537751/kb4537751-fix-error-due-to-explicit-transaction-isolation-level-hint-w
  - https://support.microsoft.com/en-us/help/4528490/kb4528490-fix-access-violation-occurs-when-attempting-to-fetch-the-iam
  - https://support.microsoft.com/en-us/help/4528139/kb4528139-fix-debug-assertion-failed-error-if-you-create-objects-that
  - https://support.microsoft.com/en-us/help/4528492/kb4528492-fix-dtc-transaction-scenario-reports-unsupported-transaction

4 p30

Microsoft knows this is still a problem, so...

SQL Server 2022: yet more fixes.

4 p31

What's New in 2022

https://docs.microsoft.com/en-us/sql/sql-server/what-s-new-in-sql-server-2022?view=sql-server-ver16

System page latch concurrency enhancements	Concurrent updates to global allocation map (GAM) pages and shared global allocation map (SGAM) pages reduce page latch contention while allocating/deallocating data pages and extents. These enhancements apply to all user databases and especially benefit tempdb heavy workloads.

4 p32

What we don't know for sure yet

How well it'll work

- No demos or performance numbers yet
- They've claimed fixes in the past

If 2022 changes guidance on the # of tempdb files

- Setup still creates multiple data files

Recap

The problem:

Some workloads constantly create/drop temp tables and table variables.

This is a problem for SQL Server because it has to track where those objects live in a few kinds of pages:

- GAM: Global Allocation Map pages
- SGAM: Shared Global Allocation Map pages
- PFS: Page Free Space pages

SQL Server hits internal bottlenecks on controlling access to those pages.

4 p35

The easy first solution

Create at least 4-8 equally sized TempDB data files.

This isn't about faster access to storage.

This IS about adding more GAM/SGAM/PFS pages and letting SQL Server balance load across them.

If you see PAGELATCH% waits, specifically on TempDB pages (as visualized by sp_WhoIsActive), then you need to use additional solutions, too.

4 p36

Additional solutions

- More TempDB data files
- Changing code from temp tables to table variables
- Enabling SQL Server 2019's In-Memory Optimized TempDB

4 p37