BRENT OZAR
UNLIMITED®

Poison Wait: THREADPOOL

Like Deadpool, but much less funny.

3.1 p1

© Brent Ozar Unlimited®. All rights reserved.

Agenda

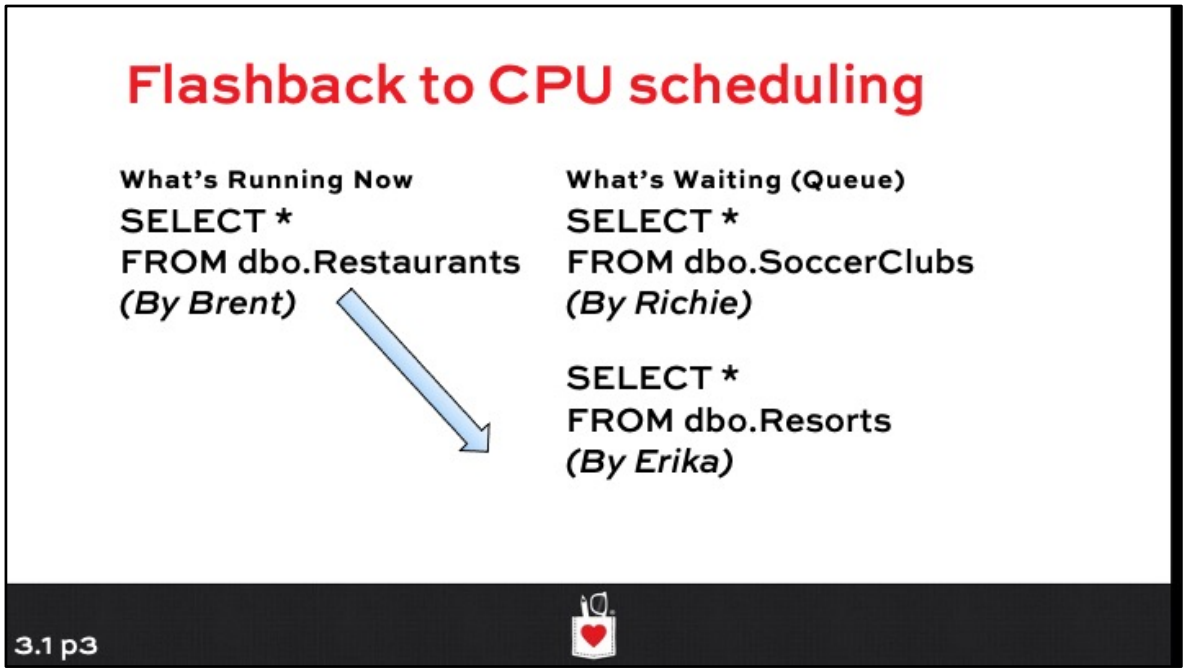
We already covered running a lot of normal queries: SOS_SCHEDULER_YIELD

Running an abnormal query: blocking

Running an abnormal query AND a lot of normal ones: THREADPOOL

3.1 p2

© Brent Ozar Unlimited®. All rights reserved.



But the instant my query needs something that isn't in cache, like I need to wait for a locked page or I need to wait for something to come back from disk, I go to the back of the line.

© Brent Ozar Unlimited®. All rights reserved.

SOS_SCHEDULER_YIELD

A task uses CPU for up to 4 milliseconds straight

At that point, it yields the CPU scheduler

- Counts as 1 SOS_SCHEDULER_YIELD wait
- The clock starts on the wait time
- The clock stops when it can get back on a CPU

The longer the average wait time,
the worse off your CPUs are.

3.1 p4

© Brent Ozar Unlimited®. All rights reserved.

Some admins: ¯_(ツ)_/¯ at SOS_SCHEDULER_YIELD

"Our CPU load is high, but SQL still works."

"We use virtualization, and we expect high CPU."

"We paid a lot for these licenses, and we want to use them, not have them sitting around idle."

And that's okay…until it's not.

3.1 p5

© Brent Ozar Unlimited®. All rights reserved.

Let's run an
*ab*normal query.

3.1 p6

© Brent Ozar Unlimited®. All rights reserved.

Take out a lock

```
BEGIN TRAN
UPDATE dbo.Users
SET Reputation = 1000000
WHERE Id = 26837;
```

3.1 p7

© Brent Ozar Unlimited®. All rights reserved.

When COUNT(*) tries to run...

It starts counting rows, but ...

It gets to User Id 26837 and stops

Which means it stops consuming CPU

But it's still consuming worker threads while it waits to move forward

3.1 p8

© Brent Ozar Unlimited®. All rights reserved.

sp_BlitzFirst warns you.

```
sp_BlitzFirst
```

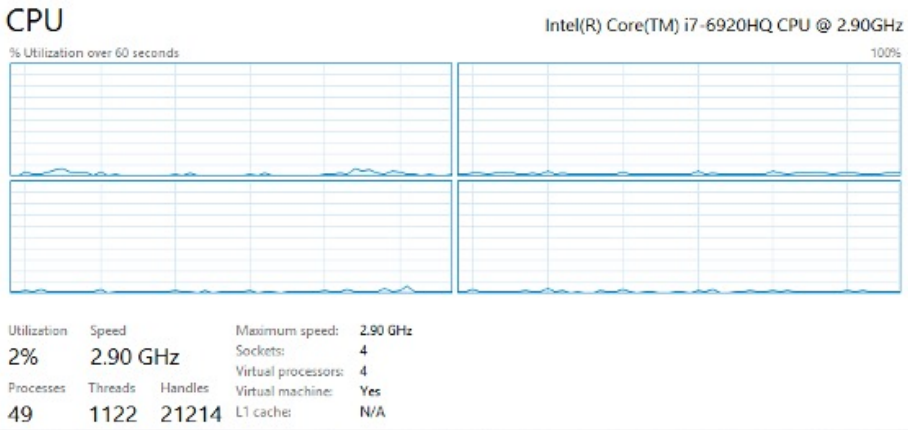
150 %

Results | Messages

	Priority	FindingsGroup	Finding	QueryText	Details
1	0	sp_BlitzFirst 2018-0...	From Your Community Volunteers	NULL	<?ClickToSeeDetails -- We hope you
2	1	Query Problems	Long-Running Query Blocking Others	BEGIN TRAN UPDATE dbo.Users SET Reputation = 1...	<?ClickToSeeDetails -- Query in Sta
3	1	Query Problems	Long-Running Query Blocking Others	SELECT COUNT(*) FROM dbo.Users;	<?ClickToSeeDetails -- Query in Sta
4	1	Query Problems	Long-Running Query Blocking Others	SELECT COUNT(*) FROM dbo.Users;	<?ClickToSeeDetails -- Query in Sta
5	1	Query Problems	Long-Running Query Blocking Others	SELECT COUNT(*) FROM dbo.Users;	<?ClickToSeeDetails -- Query in Sta
6	1	Query Problems	Long-Running Query Blocking Others	SELECT COUNT(*) FROM dbo.Users;	<?ClickToSeeDetails -- Query in Sta
7	1	Query Problems	Long-Running Query Blocking Others	SELECT COUNT(*) FROM dbo.Users;	<?ClickToSeeDetails -- Query in Sta
8	1	Query Problems	Long-Running Query Blocking Others	SELECT COUNT(*) FROM dbo.Users;	<?ClickToSeeDetails -- Query in Sta
9	1	Query Problems	Long-Running Query Blocking Others	SELECT COUNT(*) FROM dbo.Users;	<?ClickToSeeDetails -- Query in Sta
10	1	Query Problems	Long-Running Query Blocking Others	SELECT COUNT(*) FROM dbo.Users;	<?ClickToSeeDetails -- Query in Sta
11	1	Query Problems	Long-Running Query Blocking Others	SELECT COUNT(*) FROM dbo.Users;	<?ClickToSeeDetails -- Query in Sta
12	50	Query Problems	Plan Cache Erased Recently	NULL	<?ClickToSeeDetails -- The oldest q
13	200	Wait Stats	CXPACKET	NULL	<?ClickToSeeDetails -- For 3126 sec
14	200	Wait Stats	LCK_M_S	NULL	<?ClickToSeeDetails -- For 48 secon

© Brent Ozar Unlimited®. All rights reserved.

CPU is idle – no one is working.



3.1 p10

© Brent Ozar Unlimited®. All rights reserved.

Blocking usually looks like this.

We have ugly queries running, and normally they need CPU, but they also need locks.

Our monitoring tools would alert us that we have large numbers of long-blocked queries.

We can still run diagnostic queries just fine.

In most shops, this is where alarm bells get raised.

But not everywhere.

3.1 p11

© Brent Ozar Unlimited®. All rights reserved.

Let's run an *ab*normal query, plus LOTS of normal queries.

3.1 p12

© Brent Ozar Unlimited®. All rights reserved.

Leave the blocking query running

```
BEGIN TRAN
UPDATE dbo.Users
SET Reputation = 1000000
WHERE Id = 26837;
```

3.1 p13

© Brent Ozar Unlimited®. All rights reserved.

Start a big load test

Stop SQLQueryStress with **150 threads.**

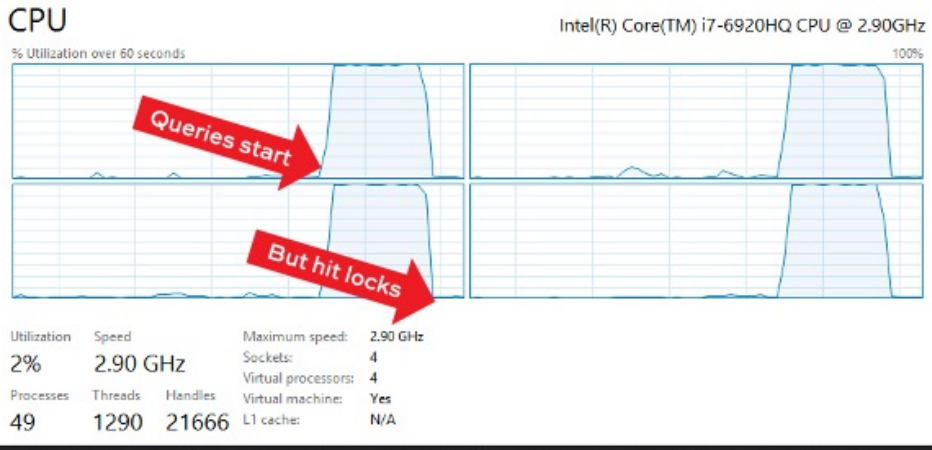
What do your CPUs look like now?

What are your wait stats?

3.1 p14

© Brent Ozar Unlimited®. All rights reserved.

CPU goes to 100%, then drops.



3.1 p15

© Brent Ozar Unlimited®. All rights reserved.

Simple DMV queries are failing

```
/* Overall workers */
SELECT s.cpu_id, w.scheduler_address, COUNT(*) AS workers
FROM sys.dm_os_workers w
INNER JOIN sys.dm_os_schedulers s ON w.scheduler_address = s.scheduler_address
WHERE s.status = 'VISIBLE ONLINE'
GROUP BY s.cpu_id, w.scheduler_address
ORDER BY s.cpu_id, w.scheduler_address;
```

100 %

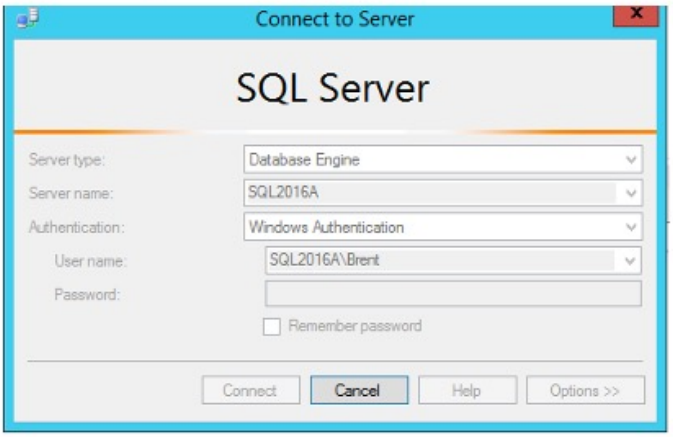
Messages

```
An error occurred while executing batch.
Error message is: ExecuteReader requires an open and available Connection. The connection's current state is closed.
```

3.1 p16

© Brent Ozar Unlimited®. All rights reserved.

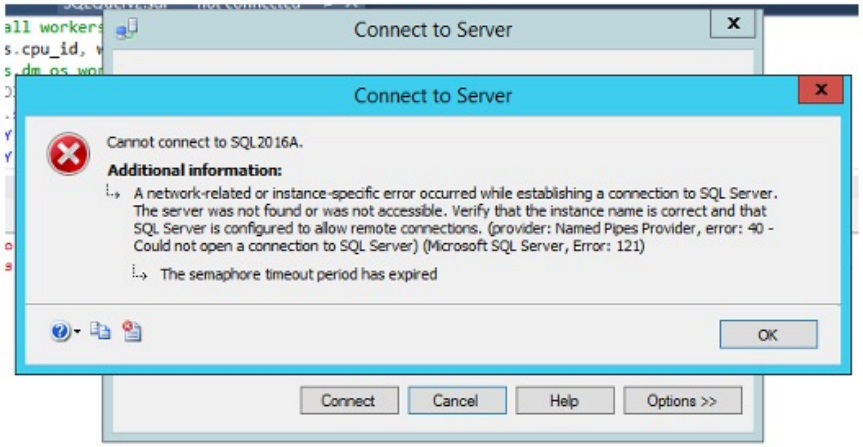
And when you try to reconnect...



3.1 p17

© Brent Ozar Unlimited®. All rights reserved.

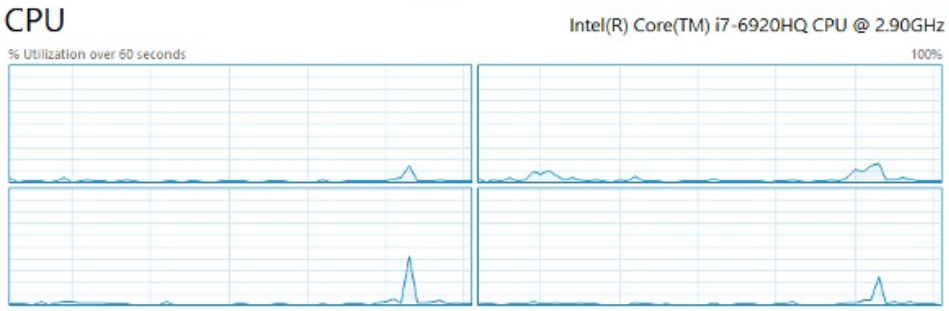
We start getting login failures



3.1 p18

© Brent Ozar Unlimited®. All rights reserved.

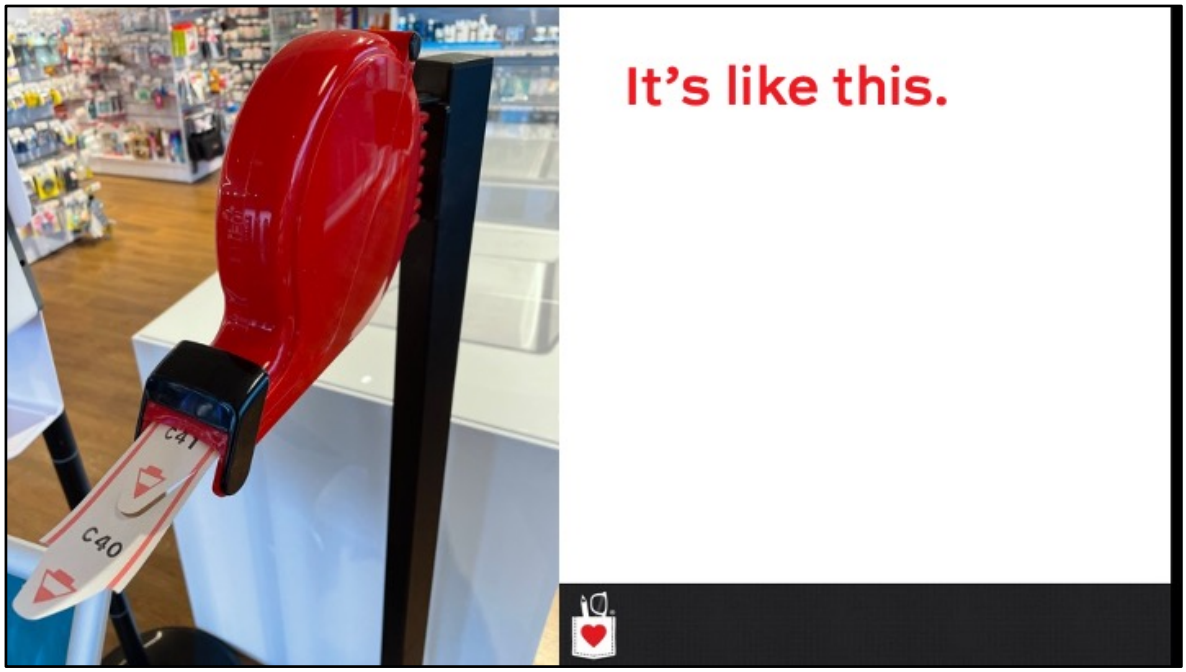
But Task Manager says we're idle.



Utilization	Speed		Maximum speed:	2.90 GHz
1%	2.90 GHz		Sockets:	4
			Virtual processors:	4
Processes	Threads	Handles	Virtual machine:	Yes
48	1257	21789	L1 cache:	N/A

3.1 p19

© Brent Ozar Unlimited®. All rights reserved.



© Brent Ozar Unlimited®. All rights reserved.

This is THREADPOOL.

We have ugly queries running, and normally they need CPU, but they also need locks.

Our monitoring tools can't connect to SQL Server: if we look at historical data, it looks like just holes of missing data.

You rarely see the THREADPOOL wait directly, but this is what it feels like: being unable to connect.

Let's dig deeper.

3.1 p21

© Brent Ozar Unlimited®. All rights reserved.

How to dig deeper

At startup, the SQLOS sets aside:

- A scheduler (but not a CPU core)
- A worker thread mapped to that scheduler
- A small amount of RAM
- A dedicated port to connect to this scheduler

It's called the Dedicated Admin Connection (DAC).

Only usable by one session at a time

Learn more: BrentOzar.com/go/dac

3.1 p22

© Brent Ozar Unlimited®. All rights reserved.

The Dedicated Admin Connection (DAC)

Reserves CPU resources for an administrator

- Only allows single threaded queries
  (this isn't for maintenance, it's just for emergencies)
- Only one person, one session can use it at a time

You need to enable this for remote access to use it

- "Remote" mean access over TCP/IP

This is a simple sp_configure change and does not require a restart

3.1 p23

© Brent Ozar Unlimited®. All rights reserved.

Enable the remote DAC

```
/* look for pending changes before you go farther*/
SELECT *
FROM sys.configurations
WHERE value <> value_in_use;
GO

/* This sets the value */
exec sp_configure 'remote admin connections', 1
GO
/* This make it take effect */
RECONFIGURE
GO
```

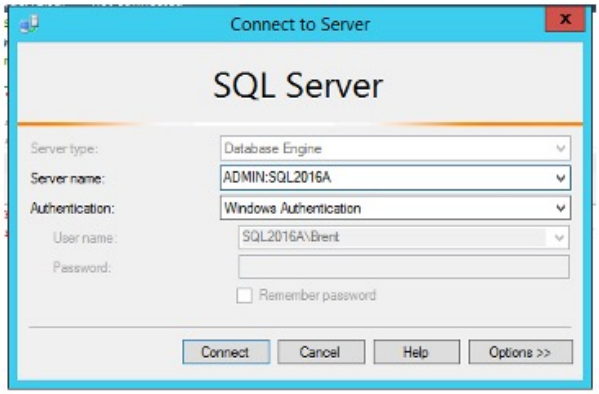
3.1 p24

© Brent Ozar Unlimited®. All rights reserved.

Connecting with the DAC

Use the prefix "admin:"

You may get an error. That's Intellisense, trying to connect in a secret session, and failing miserably.



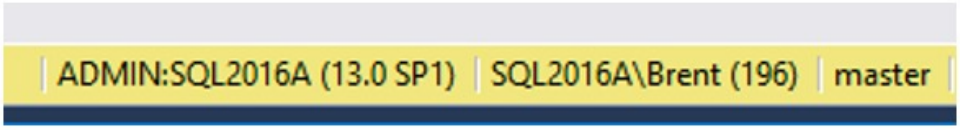
3.1 p25

© Brent Ozar Unlimited®. All rights reserved.

How to tell you got the DAC

Look at the bottom right of your session window!

It should show that you're connected to admin:instancename if it worked



If someone else forgot to disconnect, you can find out who they are by using the query at http://BrentOzar.com/go/DAC

3.1 p26

© Brent Ozar Unlimited®. All rights reserved.

Now you can troubleshoot.

Check sys.dm_os_workers: we do indeed have many, and we've hit (or exceeded) max worker threads

```sql
/* Overall workers */
SELECT s.cpu_id, w.scheduler_address, COUNT(*) AS workers
FROM sys.dm_os_workers w
INNER JOIN sys.dm_os_schedulers s ON w.scheduler_address = s.scheduler_address
WHERE s.status = 'VISIBLE ONLINE'
GROUP BY s.cpu_id, w.scheduler_address
ORDER BY s.cpu_id, w.scheduler_address;
```

100 %

Results | Messages

	cpu_id	scheduler_address	workers
1	0	0x000000C6E0180040	137
2	1	0x000000C6E01A0040	135
3	2	0x000000C6E01C0040	136
4	3	0x000000C6E01E0040	138

3.1 p27

© Brent Ozar Unlimited®. All rights reserved.

What's going on?

We have a lot of queries waiting on locks, but specifically how many?

sp_WhoIsActive

100 %

Results | Messages

	dd hh:mm:ss.mss	session_id	sql_text	login_name	wait_info	tempdb_allocations	tempdb_current	blocking_session_id	reads
1	00 00:20:55.486	58	<?query -- BEGIN TRAN UPDATE dbo.Users SET Re...	SQL2016A\Brent	NULL	0	0	NULL	24
2	00 00:17:55.006	60	<?query -- SELECT COUNT(*) FROM StackOverflow.d...	SQL2016A\Brent	(1074957ms)LCK_M_S	0	0	56	4
3	00 00:17:55.006	61	<?query -- SELECT COUNT(*) FROM StackOverflow.d...	SQL2016A\Brent	(1074957ms)LCK_M_S	0	0	56	4
4	00 00:17:55.006	56	<?query -- SELECT COUNT(*) FROM StackOverflow.d...	SQL2016A\Brent	(1074958ms)LCK_M_S	0	0	58	0
5	00 00:17:54.870	62	<?query -- SELECT COUNT(*) FROM StackOverflow.d...	SQL2016A\Brent	(1074943ms)LCK_M_S	0	0	56	0
6	00 00:17:54.800	63	<?query -- SELECT COUNT(*) FROM StackOverflow.d...	SQL2016A\Brent	(1074777ms)LCK_M_S	0	0	56	0
7	00 00:17:54.766	64	<?query -- SELECT COUNT(*) FROM StackOverflow.d...	SQL2016A\Brent	(1074746ms)LCK_M_S	0	[illegible]	[illegible]	0
8	00 00:17:54.666	65	<?query -- SELECT COUNT(*) FROM StackOverflow.d...	SQL2016A\Brent	(1074639ms)LCK_M_S	0	[illegible]	[illegible]	0

Not 150 or 151

Query executed successfully. | ADMIN:SQL2016A (13.0 SP1) | SQL2016A\Brent (196) | master | 00:[illegible] | 138 rows

3.1 p28

© Brent Ozar Unlimited®. All rights reserved.

We ran out of worker threads.

My 4-core VM's max: 512.

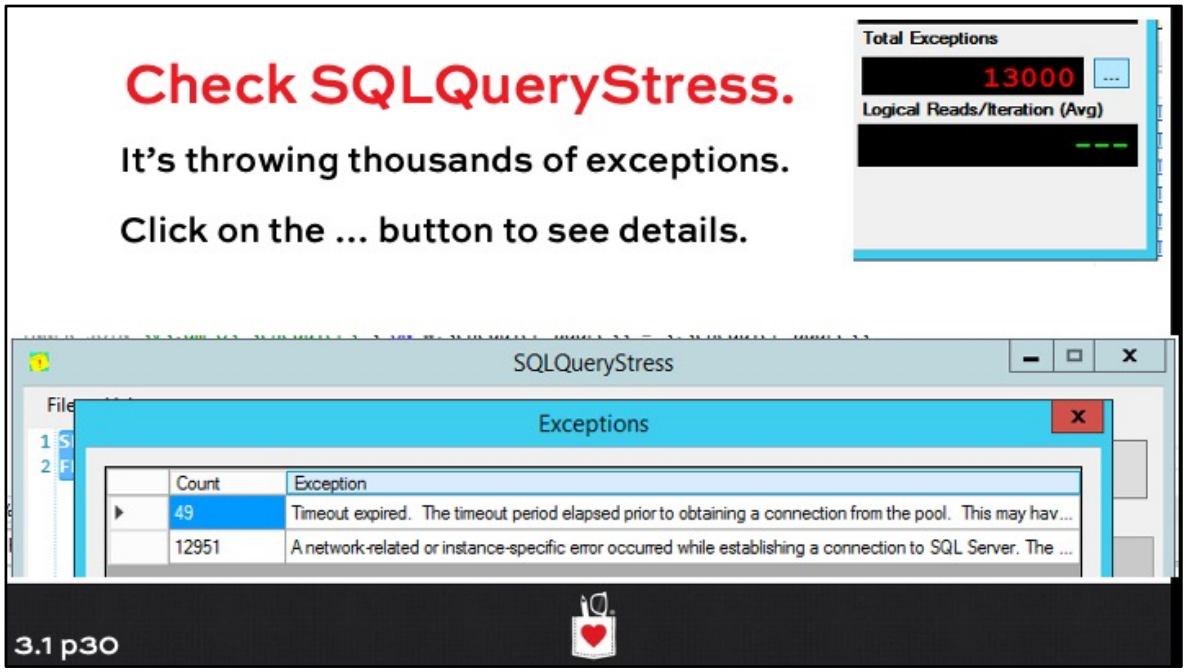
150 queries * 4 threads each: need at least 600.

So not all 150 can run, and that's why new queries can't start either.

Number of CPUs	32-bit computer	64-bit computer
<= 4 processors	256	512
8 processors	288	576
16 processors	352	704
32 processors	480	960
64 processors	736	1472
128 processors	4224	4480
256 processors	8320	8576

3.1 p29

© Brent Ozar Unlimited®. All rights reserved.



© Brent Ozar Unlimited®. All rights reserved.

THREADPOOL symptoms

Normally, the server is alright. Not great, just alright.

But every now and then:

- We have a lot of active worker threads
- For some reason, they can't make progress (it's usually due to a blocking query)
- SQL Server runs out of available worker threads and starts recording THREADPOOL waits
- New queries can't get started, so
- Monitoring tools can't get data
- The server just feels frozen – but CPU use is low

3.1 p31

© Brent Ozar Unlimited®. All rights reserved.

That wait is poison

That's why we report any occurrences of THREADPOOL wait as an urgent poison wait in tools like sp_Blitz. If you're seeing *any* of it, the server is unusable during that time.

```
sp_BlitzFirst @SinceStartup = 1
```

00 %

Results | Messages

	Pattern	Sample Ended	Hours Sample	wait_type	Wait Time (Hours)	Per Core Per Hour
1	WAIT STATS	2017-02-23 09:12:54.3986015 -05:00	5.1	CXPACKET	92.3	0.0
2	WAIT STATS	2017-02-23 09:12:54.3986015 -05:00	5.1	LCK_M_S	8.6	0.0
3	WAIT STATS	2017-02-23 09:12:54.3986015 -05:00	5.1	SOS_SCHEDULER_YIELD	0.4	0.0
4	WAIT STATS	2017-02-23 09:12:54.3986015 -05:00	5.1	THREADPOOL	0.1	0.0

3.1 p32

© Brent Ozar Unlimited®. All rights reserved.

How to react when it strikes

Connect to the DAC

Run sp_WhoIsActive and look for a blocking firestorm

Collect the query, plan, and app sources for the lead blocker and the blocked queries

```
sp_WhoIsActive @get_plans = 1
```

100 %

Results | Messages

	dd hh:mm:ss.mss	session_id	sql_text	login_name	wait_info	tempdb_allocations	tempdb_current	blocking_session_id
1	00 00:23:34.050	58	<?query -- BEGIN TRAN UPDATE dbo.Users SET Re...	SQL2016A\Brent	NULL	0	0	NULL
2	00 00:20:33.570	60	<?query -- SELECT COUNT(*) FROM StackOverflow.d...	SQL2016A\Brent	(1233535ms)LCK_M_S	0	0	56
3	00 00:20:33.570	61	<?query -- SELECT COUNT(*) FROM StackOverflow.d...	SQL2016A\Brent	(1233535ms)LCK_M_S	0	0	56
4	00 00:20:33.570	56	<?query -- SELECT COUNT(*) FROM StackOverflow.d...	SQL2016A\Brent	(1233536ms)LCK_M_S	0	0	58
5	00 00:20:33.434	62	<?query -- SELECT COUNT(*) FROM StackOverflow.d...	SQL2016A\Brent	(1233421ms)LCK_M_S	0	0	56
6	00 00:20:33.364	63	<?query -- SELECT COUNT(*) FROM StackOverflow.d...	SQL2016A\Brent	(1233349ms)LCK_M_S	0	0	56
7	00 00:20:33.330	64	<?query -- SELECT COUNT(*) FROM StackOverflow.d...	SQL2016A\Brent	(1233324ms)LCK_M_S	0	0	56

© Brent Ozar Unlimited®. All rights reserved.

Should you kill the lead blocker?

That's a judgment call: rollbacks are single-threaded.

We don't judge here.

But we don't believe all queries have the right to life.

3.1 p34

© Brent Ozar Unlimited®. All rights reserved.

How to fix THREADPOOL long-term

3.1 p35

© Brent Ozar Unlimited®. All rights reserved.

Variables in the equation

Server-level:	Number of cores	4
	Number of worker threads SQL Server creates by default	512
	Cost Threshold for Parallelism	5
	How parallel a query will go	MAXDOP
Query-level:	Number of simultaneous queries	?
	Cost of those queries	?
	Lock requirements	?

3.1 p36

© Brent Ozar Unlimited®. All rights reserved.

Adding more cores

Licensing makes this expensive, but it has two effects:

1. The amount of available CPU time goes up

2. The amount of worker threads may go up if we add enough cores

Number of CPUs	32-bit computer	64-bit computer
<= 4 processors	256	512
8 processors	288	576
16 processors	352	704
32 processors	480	960
64 processors	736	1472
128 processors	4224	4480
256 processors	8320	8576

3.1 p37

© Brent Ozar Unlimited®. All rights reserved.

Will going to 8 cores help?

If we spend on licensing and hardware, worker threads goes up to 576.

That's not gonna be enough to run 150*4 threads.

And the added CPU power won't help since we're waiting around on blocking.

Number of CPUs	32-bit computer	64-bit computer
<= 4 processors	256	512
8 processors	288	576
16 processors	352	704
32 processors	480	960
64 processors	736	1472
128 processors	4224	4480
256 processors	8320	8576

3.1 p38

© Brent Ozar Unlimited®. All rights reserved.

Variables in the equation

Server-level:	Number of cores	4
How about this?	Number of worker threads SQL Server creates by default	512
	Cost Threshold for Parallelism	5
	How parallel a query will go	MAXDOP
Query-level:	Number of simultaneous queries	?
	Cost of those queries	?
	Lock requirements	?

3.1 p39

© Brent Ozar Unlimited®. All rights reserved.

Max Worker Threads

This sp_configure setting lets you override the number of worker threads SQL Server allows.

We could raise it into the thousands, but:

- Our blocked queries won't finish faster
- We're only buying a small amount of time
- If we get more end user SELECT loads, we're still gonna hit THREADPOOL
- We can run into memory starvation issues

3.1 p40

© Brent Ozar Unlimited®. All rights reserved.

It gets even worse

What happens when our hundreds of simultaneous SELECT queries are actually working?

Each query will get even less CPU time per second, because when it yields the CPU scheduler, it's going to have to wait even longer to get back on.

When you see THREADPOOL waits, this isn't the fix.

3.1 p41

© Brent Ozar Unlimited®. All rights reserved.

Variables in the equation

Server-level:	Number of cores	4
	Number of worker threads SQL Server creates by default	512
	Cost Threshold for Parallelism	5
	How parallel a query will go	MAXDOP
Query-level:	Number of simultaneous queries	?
	Cost of those queries	?
	Lock requirements	?

How about these?

3.1 p42

© Brent Ozar Unlimited®. All rights reserved.

Our queries are going parallel.

We could either:

- Raise Cost Threshold for Parallelism high enough that small queries don't go parallel, or
- Set Max Degree of Parallelism so they use less worker threads (or even only 1 thread)

To try it, you'll need to stop the load test first (because in-flight queries aren't affected.)

3.1 p43

© Brent Ozar Unlimited®. All rights reserved.

MAXDOP 1 kinda fixes it.

Queries go single-threaded, and use less worker threads.

But they're still blocked. Users won't be happy.

The server is just now responsive again, and our monitoring tools show things like blocking.

When there's no blocking, our queries will SUCK.

This isn't really the right long-term fix
(but can be good emergency duct tape.)

3.1 p44

© Brent Ozar Unlimited®. All rights reserved.

Variables in the equation

Server-level:	Number of cores	4
	Number of worker threads SQL Server creates by default	512
	Cost Threshold for Parallelism	5
	How parallel a query will go	MAXDOP
Query-level:	Number of simultaneous queries	?
	Cost of those queries	?
	Lock requirements	?

How about this?

3.1 p45

© Brent Ozar Unlimited®. All rights reserved.

True story

Our application runs on about a dozen app servers.

It's not fast enough, so we spun up more. About 300.

Every 5 minutes, they look for work in a queue table.

The queue table got big enough that the look-for-work query started going parallel.

Every 5 minutes, boom: SQL Server seems frozen.

3.1 p46

© Brent Ozar Unlimited®. All rights reserved.

True solution

"Can we put an index on that queue table?"

"No, it's a vendor app."

"Can you do me a favor and
shut down 290 of the app servers?"

"You'd better be right about this."

"Wow, it's amazing!
So fast! Much wow!"

3.1 p47

© Brent Ozar Unlimited®. All rights reserved.

Variables in the equation

Server-level:	Number of cores	4
	Number of worker threads SQL Server creates by default	512
	Cost Threshold for Parallelism	5
	How parallel a query will go	MAXDOP
Query-level:	Number of simultaneous queries	?
How about this?	Cost of those queries	?
	Lock requirements	?

3.1 p48

© Brent Ozar Unlimited®. All rights reserved.

Reducing query costs

Run sp_BlitzIndex, and look for missing indexes with high impact (there's usually a very active table with no indexes on it at all, or a desperately needed index)

Run sp_BlitzCache @SortOrder = 'reads' and look at queries run frequently that desperately need indexes (and you may have to hand-tune the indexes here)

That usually makes THREADPOOL disappear.
And it's free! This is my favorite fix. (No, I'm not free.)

3.1 p49

© Brent Ozar Unlimited®. All rights reserved.

Variables in the equation

Server-level:	Number of cores	4
	Number of worker threads SQL Server creates by default	512
	Cost Threshold for Parallelism	5
	How parallel a query will go	MAXDOP
Query-level:	Number of simultaneous queries	?
	Cost of those queries	?
How about this?	Lock requirements	?

3.1 p50

© Brent Ozar Unlimited®. All rights reserved.

Reducing blocking issues

Finding the lead blocker during THREADPOOL issues is really tough.

Monitoring systems don't help because they can't gather data during these windows either.

When the SQL Server seems frozen:

- Connect to the DAC
- Run sp_WhoIsActive to find the lead blocker
- No, doing this in an Agent job won't work

3.1 p51

© Brent Ozar Unlimited®. All rights reserved.

When you find the lead blocker

Can we make this transaction shorter?
(Is it doing multiple things in one long block?)

Can we make this transaction faster?
(Does it need an index to find the right rows?)

If those fail, can a different isolation level help?
(We cover RCSI and snapshot in the LCK module.)

3.1 p52

© Brent Ozar Unlimited®. All rights reserved.

© Brent Ozar Unlimited®. All rights reserved.

THREADPOOL means...

We’ve run out of worker threads, new queries can’t start

SQL Server seems frozen, and monitoring tools fail

Windows OS CPU metrics seem totally fine, idle

Connect with the DAC, find the lead blocker

To fix it: tune indexes, queries to avoid the blocking storm and reduce parallelism requirements

Don’t throw CPU or worker threads at it:
it’s expensive and ineffective under most circumstances

3.1 p54

© Brent Ozar Unlimited®. All rights reserved.