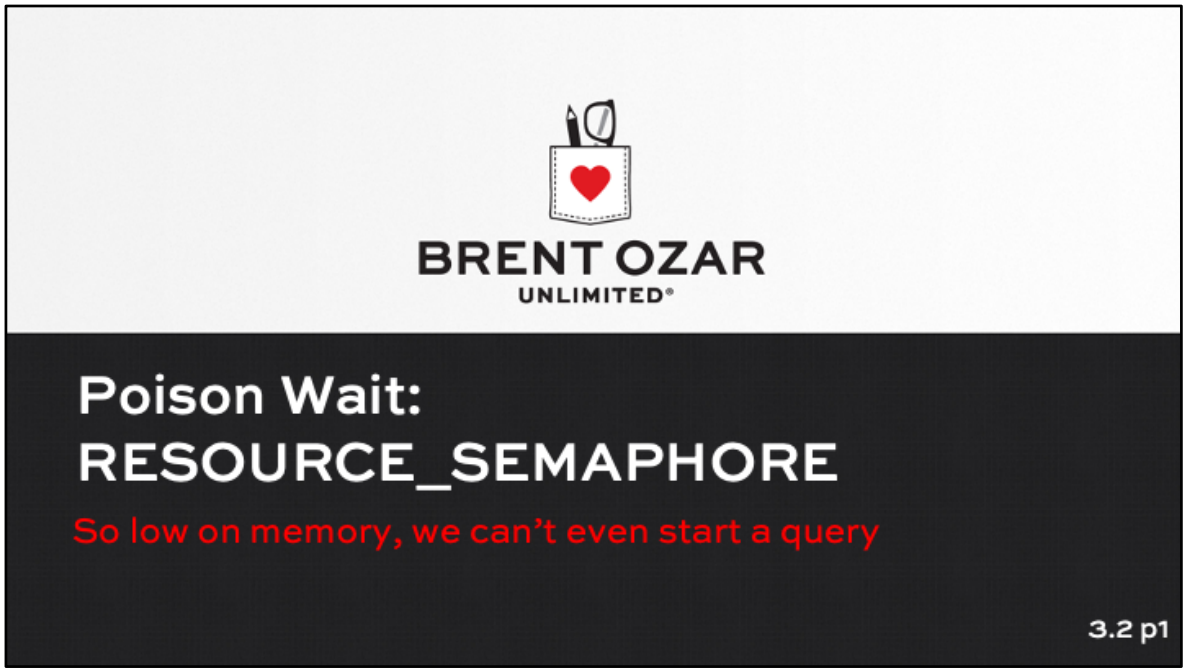


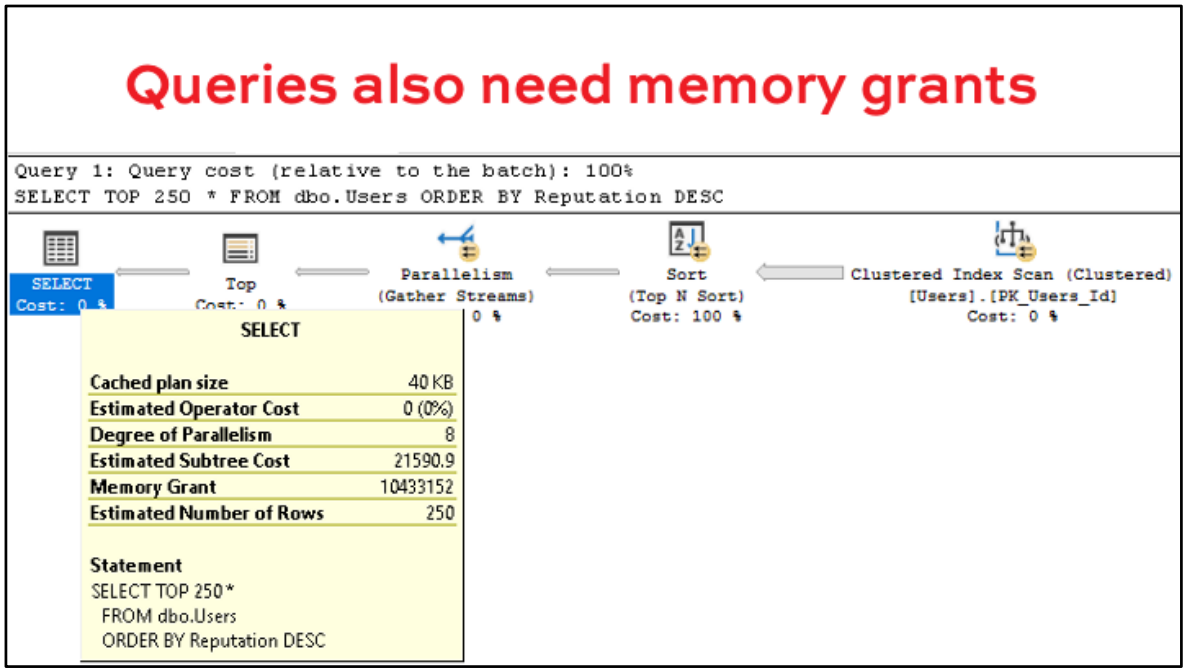
We often talk about these two memory pools in SQL Server

Execution Plan cache
(metadata about how to run a query, stored in memory to be reused)

Data cache
aka "Buffer Pool"
(Data pages from your tables which have been read by queries, stored in memory to be reused)

3.2 p2

Queries also need memory grants



Memory grants

This is sometimes called “query workspace memory”

Queries need memories for things like:

- Join operators (these may secretly build temp objects that need a good chunk of memory)
- Sort operators (same thing)
- Parallelism

SQL Server estimates how much it needs for this before the query starts

3.2 p4

Actual Plan details

Granted memory =
How much workspace memory the query actually got (in KB)

Requested memory =
Ideal memory grant (KB)

Required memory =
The *minimum* grant the query needed to get started (KB)

Max used memory =
Just what it sounds like (KB)

** SQL Server 2012 and higher*

Estimated Subtree Cost	21590.9
Memory Grant	10433152
MemoryGrantInfo	
DesiredMemory	70376896
GrantedMemory	10433152
GrantWaitTime	0
MaxQueryMemory	10433152
MaxUsedMemory	1273360
RequestedMemory	10433152
RequiredMemory	5632
SerialDesiredMemory	70371728
SerialRequiredMemory	512
Optimization Level	FULL

3.2 p5

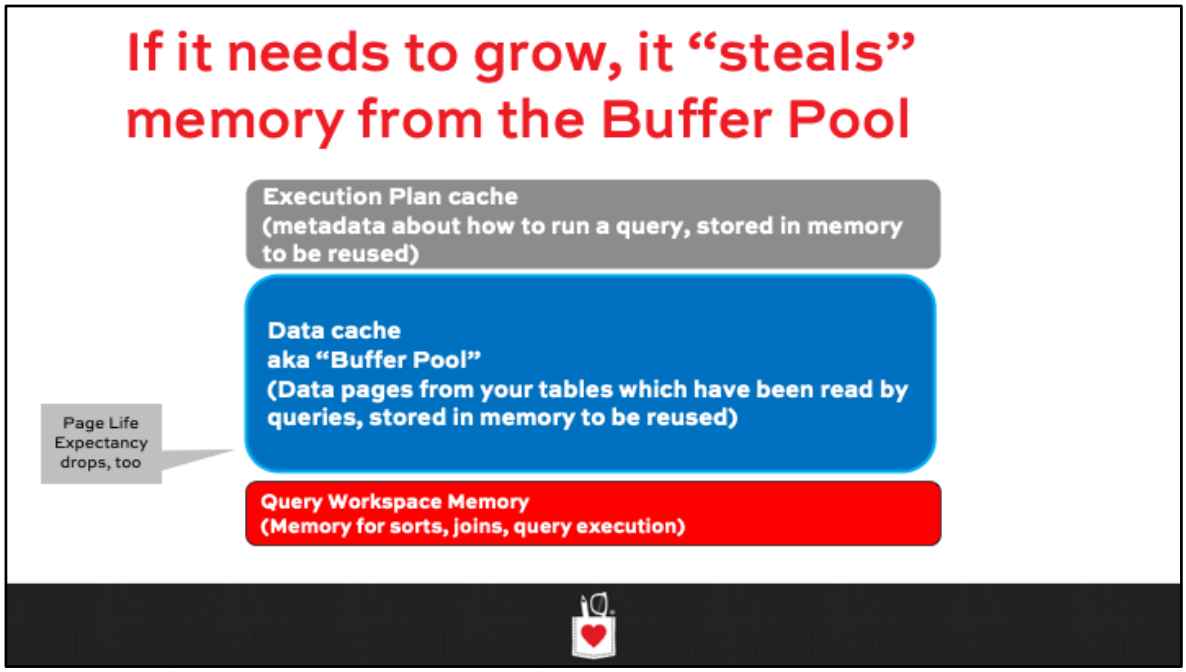
Query workspace memory has to come from somewhere

Execution Plan cache
(metadata about how to run a query, stored in memory to be reused)

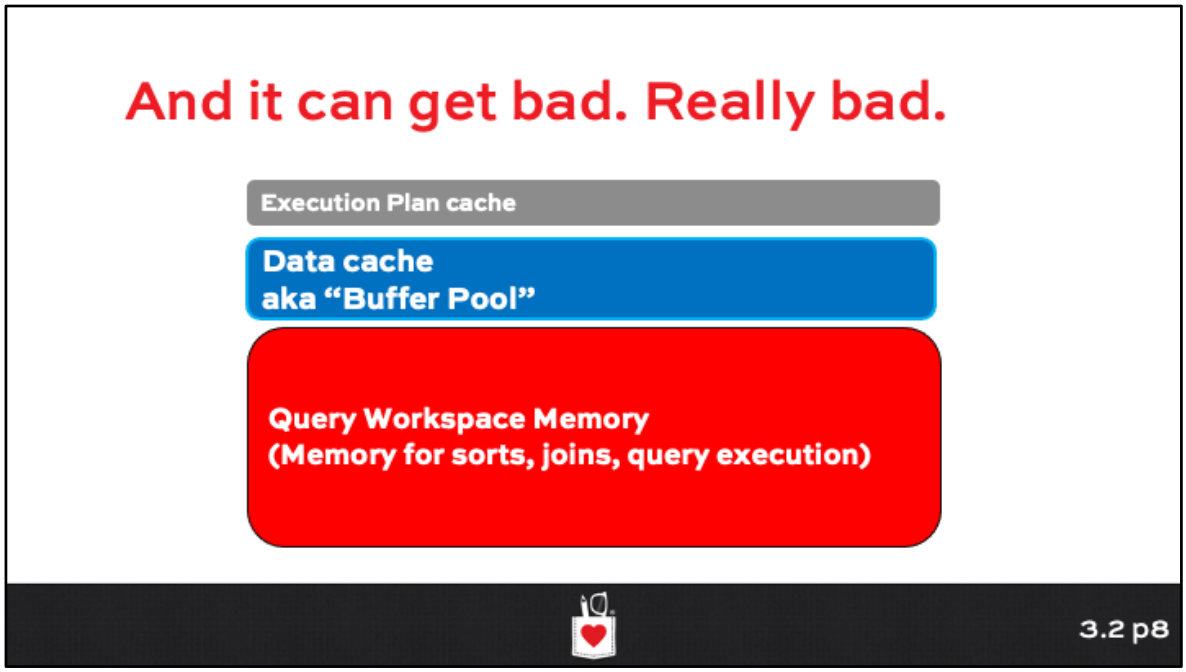
Data cache
aka "Buffer Pool"
(Data pages from your tables which have been read by queries, stored in memory to be reused)

Query Workspace Memory
(Memory for sorts, joins, query execution)

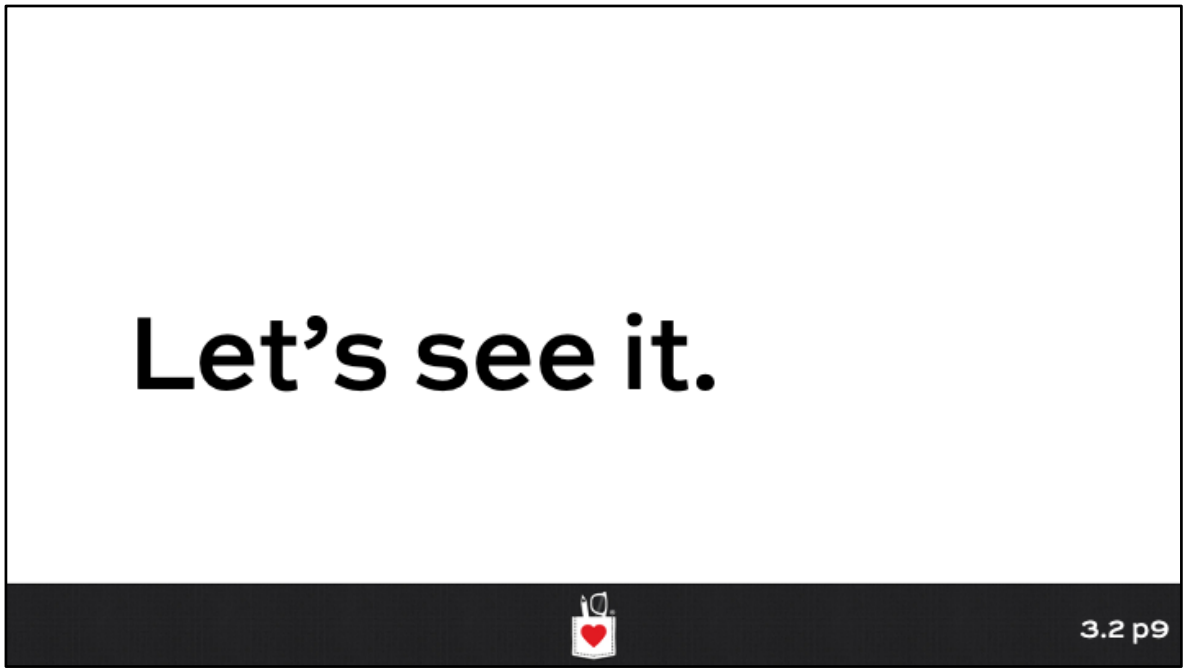
3.2 p6



So does the execution plan cache, but whatever.



So does the execution plan cache, but whatever.



No skimping here.

Give your SQL Server plenty of memory.

```
EXEC sys.sp_configure N'max server memory (MB)', N'55000';
EXEC sys.sp_configure N'cost threshold for parallelism', N'5';
EXEC sys.sp_configure N'max degree of parallelism', N'0';
GO
RECONFIGURE
GO
USE StackOverflow;
GO
DropIndexes;
GO
```

3.2 p10

BIG DATA

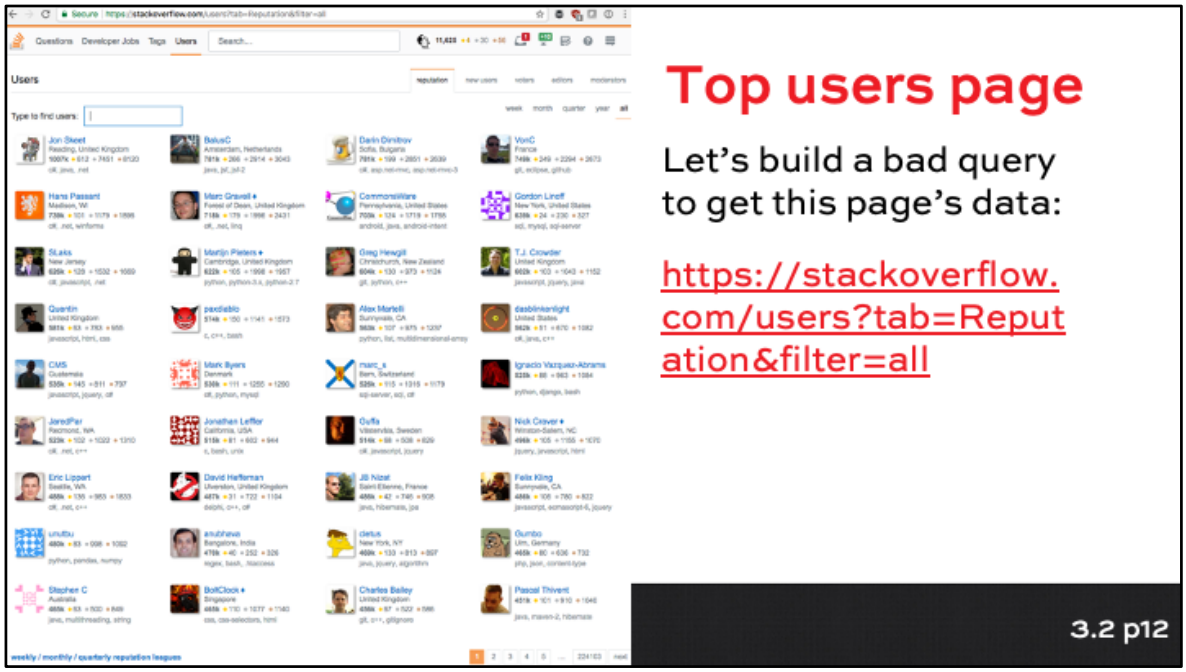
Let's change the data types on the Users table and make them big.

```
ALTER TABLE dbo.Users
  ALTER COLUMN DisplayName NVARCHAR(400);
ALTER TABLE dbo.Users
  ALTER COLUMN Location NVARCHAR(1000);
ALTER TABLE dbo.Users
  ALTER COLUMN WebsiteUrl NVARCHAR(2000);
```

This will happen instantly: it's a metadata-only change.

The table's not actually getting bigger.

3.2 p11



The query is simple

```
SELECT TOP 250 *
FROM dbo.Users
ORDER BY Reputation DESC;
```

Test it. How fast is it?

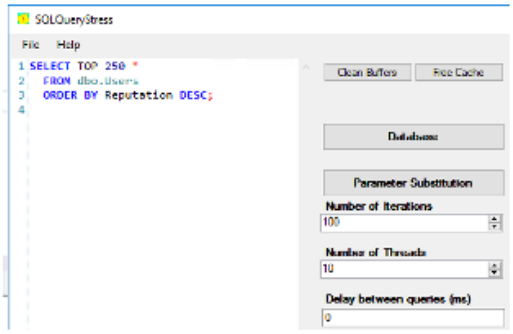
How many reads does it do?

How much CPU time does it take?

3.2 p13

Let's stress test it.

But just a little stress:
run it in SQLQueryStress
on just 10 threads,
100 iterations per thread.



A new wait type appears!

```
sp_BlitzWho
```

150 %

Results | Messages

	run_date	elapsed_time	session_id	database_name	query_text	wait_info
1	2018-02-15 00:37:17.460	0:00:00:10:000	62	StackOverflow	SELECT TOP 250 * FROM dbo.Users ORDER BY R...	CXPACKET (128 ms)
2	2018-02-15 00:37:17.460	0:00:00:10:000	63	StackOverflow	SELECT TOP 250 * FROM dbo.Users ORDER BY R...	CXPACKET (11941 ms)
3	2018-02-15 00:37:17.460	0:00:00:09:000	64	StackOverflow	SELECT TOP 250 * FROM dbo.Users ORDER BY R...	CXPACKET (21 ms)
4	2018-02-15 00:37:17.460	0:00:00:09:000	65	StackOverflow	SELECT TOP 250 * FROM dbo.Users ORDER BY R...	RESOURCE_SEMAPHORE (39716 ms)
5	2018-02-15 00:37:17.460	0:00:00:09:000	66	StackOverflow	SELECT TOP 250 * FROM dbo.Users ORDER BY R...	RESOURCE_SEMAPHORE (39632 ms)
6	2018-02-15 00:37:17.460	0:00:00:09:000	67	StackOverflow	SELECT TOP 250 * FROM dbo.Users ORDER BY R...	RESOURCE_SEMAPHORE (39276 ms)
7	2018-02-15 00:37:17.460	0:00:00:09:000	68	StackOverflow	SELECT TOP 250 * FROM dbo.Users ORDER BY R...	RESOURCE_SEMAPHORE (38988 ms)
8	2018-02-15 00:37:17.460	0:00:00:09:000	69	StackOverflow	SELECT TOP 250 * FROM dbo.Users ORDER BY R...	RESOURCE_SEMAPHORE (38756 ms)
9	2018-02-15 00:37:17.460	0:00:00:09:000	70	StackOverflow	SELECT TOP 250 * FROM dbo.Users ORDER BY R...	RESOURCE_SEMAPHORE (38584 ms)

3.2 p15

ExpertMode shows grants, too

Requires current patches of 2012 & newer

```
sp_BlitzWho @ExpertMode = 1
```

Messages

grant_time	requested_memory_kb	grant_memory_kb	is_request_granted	required_memory_kb	query_memory_grant_used_memory_kb	ideal_memory_kb	wait_order	wait_time_ms	next_candidate_for_memory_grant
Feb 15 2018 12:37AM	10433152	10433152	Memory Request Granted	5632	775608	70376896	N/A	N/A	N/A
Feb 15 2018 12:37AM	10433152	10433152	Memory Request Granted	5632	2312	70376896	N/A	N/A	N/A
Memory Not Granted	10433152	NULL	N/A	5632	NULL	70376896	0	20984	Yes
Memory Not Granted	10433152	NULL	N/A	5632	NULL	70376896	2	20891	No
Memory Not Granted	10433152	NULL	N/A	5632	NULL	70376896	1	20506	No
Memory Not Granted	10433152	NULL	N/A	5632	NULL	70376896	3	10531	No
Memory Not Granted	10433152	NULL	N/A	5632	NULL	70376896	5	10281	No
Memory Not Granted	10433152	NULL	N/A	5632	NULL	70376896	4	10531	No

3.2 p16

Performance counters show info

‘Pending’ memory grants indicate RESOURCE_SEMAPHORE / workspace memory grant waits

```
SELECT object_name, counter_name, instance_name, cntr_value
FROM sys.dm_os_performance_counters
where
    (counter_name like 'Active memory grants count%' or
    counter_name like 'Pending memory grants count%' or
    counter_name like 'Max request memory grant (KB)%')
    and instance_name like 'default%'
    and cntr_type=65792
ORDER by object_name, counter_name, instance_name
GO
```

100 %

Results | Messages

	object_name	counter_name	instance_name	cntr_value
1	SQLServer:Resource Pool Stats	Active memory grants count	default	3
2	SQLServer:Resource Pool Stats	Pending memory grants count	default	5
3	SQLServer:Workload Group Stats	Max request memory grant (KB)	default	781384

3.2 p17

What we saw

A procedure wanted a high memory grant

- The amount it wanted was more than was allowed by default for a single request (based on default configuration), so it had to take a reduced grant

When we ran threads running 8 of this query simultaneously…

- SQL Server limited the number that could execute at once
- The waiting queries showed RESOURCE_SEMAPHORE wait

Queries did complete, but they were very slow

- In the real world, this can easily cause application timeouts and the feeling that the SQL Server just isn't working

What causes big memory grants?

Queries processing a lot of data

- As data grows, queries have to do more work for hash joins, sorts, parallelism, etc.

Running lots of queries

- As your user-base grows, more people using the SQL Server can run more large queries at the same time

Optimization problems

- Joins with functions in them are very difficult for SQL Server to estimate and can be prone to high over-estimation
- Linked server queries (particularly to other databases) can vastly over-estimate the data returned from the remote side

Bugs in SQL Server

- We ran into one case where SQL Server 2005 transactional replication was vastly over-estimating the rows used on internal queries

3.2 p19

How do I know if I'm having this problem?

3.2 p20

1. At the server level

Wait stats (sys.dm_os_wait_stats DMV)

- You see RESOURCE_SEMAPHORE waits
- Even low values are a sign that things are sometimes going really wrong

Performance counters

- SQLServer: Memory Manager – Memory Grants Pending
- You want this counter to be at 0
- > 0 means someone has to wait to get a workspace memory grant and is being queued

3.2 p21

2. At the DMV details level, 2008

- Currently executing memory grants
- Historic memory grants (and reduced grants)

This info is in two DMVs:

- sys.dm_resource_governor_resource_pools
- sys.dm_resource_governor_workload_groups

You get the info even if...

- You haven't configured resource governor specially
- You're using Standard Edition

3.2 p22

2. 2012 SP3, 2014 SP2, newer

KB 3107398 – sys.dm_exec_query_stats added total, last, min, max for grant, ideal, DOP, threads

KB 3107397 – actual rows read in query plans

KB 3107401 – query hints for min & max grant percent

KB 3107172 – XE details for tempdb spills

KB 3107173 – XE details for ideal grants, DOP

3.2 p23

Shown in sp_BlitzWho

Also called by sp_BlitzFirst @ExpertMode = 1

Shows currently running queries, like sp_WhoIsActive, but with more tuning details:

```
sp_BlitzWho @ExpertMode = 1
```

Messages

grant_time	requested_memory_kb	grant_memory_kb	is_request_granted	required_memory_kb	query_memory_grant_used_memory_kb	ideal_memory_kb	wait_order	wait_time_ms	next_candidate_for_memory_grant
Feb 15 2018 12:37AM	10433152	10433152	Memory Request Granted	5632	775003	70376896	N/A	N/A	N/A
Feb 15 2018 12:37AM	10433152	10433152	Memory Request Granted	5632	2312	70376896	N/A	N/A	N/A
Memory Not Granted	10433152	NULL	N/A	5632	NULL	70376896	0	20984	Yes
Memory Not Granted	10433152	NULL	N/A	5632	NULL	70376896	2	20891	No
Memory Not Granted	10433152	NULL	N/A	5632	NULL	70376896	1	20906	No
Memory Not Granted	10433152	NULL	N/A	5632	NULL	70376896	3	10531	No
Memory Not Granted	10433152	NULL	N/A	5632	NULL	70376896	5	10281	No
Memory Not Granted	10433152	NULL	N/A	5632	NULL	70376896	4	10531	No

3.2 p24

Shown in sp_BlitzCache

And you can sort: @SortOrder = 'memory grant'
('average memory grant' works too)

```
sp_BlitzCache @SortOrder = 'memory grant'
```

150 %

Results | Messages

	erage Rows	Minimum Memory Grant KB	Maximum Memory Grant KB	Minimum Used Grant KB	Maximum Used Grant KB	Average Max Memory Grant	Min Spills	Max Spills	Total Spills	Avg Spills
1	i0.00	10433152	10433152	1273360	1273360	10433152.00	0	0	0	0.00
2	i0.00	10433152	10433152	1273440	1273440	10433152.00	0	0	0	0.00
3	i0.00	10433152	10433152	1273352	1273424	186306.2857	0	0	0	0.00
4	l.00	175016	175016	2888	2960	58338.6667	0	0	0	0.00
5	l.5714	167496	167496	2152	2224	23928.00	0	0	0	0.00
6	00	20768	20768	2576	2576	20768.00	0	0	0	0.00
7	00	19208	19208	1584	1584	19208.00	0	0	0	0.00
8	00	3984	3984	776	776	3984.00	0	0	0	0.00

3.2 p25

And we show grants as warnings

That Erik, smart fella:

	Query Type	Warnings
Y R...	Statement	Parallel, Multiple Plans, Plan created last 4hrs, Expensive Sort, Row Goals
Y R...	Statement	Parallel, Multiple Plans, Plan created last 4hrs, Expensive Sort, Row Goals
Y R...	Statement	Parallel, Multiple Plans, Plan created last 4hrs, Expensive Sort, Row Goals
ALE...	Statement	Compilation Timeout, Plan Warnings, Function Join, Forced Serialization, Unused Memory Grant, Plan created last 4hrs, Table Variables, Long Running With Lo
ALE...	Statement	Compilation Timeout, Plan Warnings, Function Join, Forced Serialization, Unused Memory Grant, Plan created last 4hrs, Table Variables, Table Scans, Long Ru
bl.cr...	Statement	Compilation Timeout, Plan Warnings, Implicit Conversions, Function Join, Forced Serialization, Plan created last 4hrs, Table DML, Row Goals, MSTVFs
AS [I...	Statement	Compilation Timeout, Plan Warnings, Implicit Conversions, Function Join, Forced Serialization, Unused Memory Grant, Plan created last 4hrs, Row estimate misn
p.cr...	Statement	Compilation Timeout, Plan Warnings, Implicit Conversions, Function Join, Forced Serialization, Plan created last 4hrs, Row estimate mismatch, MSTVFs
bl I...	Statement	Forced Serialization, Unused Memory Grant, Plan created last 4hrs
bl I...	Statement	Forced Serialization, Unused Memory Grant, Plan created last 4hrs

3.2 p26

What can fix the issue?

There are three options

1. Taming the problem with Resource Governor (2008+, Enterprise Edition only)
2. Tuning queries and/or indexes
3. Adding more memory

Of these three options, two of them work well

Guess which two?

3.2 p27

1) The Resource Governor

You can ...

- Classify queries into groups
- Control memory allocations by group

But, let's stop and think about this:

- Wouldn't a better way to raise the amount of memory grant available be to add more memory in a vast majority of cases?
- If we're just classifying off queries, are we actually making them *faster*?

Downsides:

- If you do this wrong, you make everything slower
- This is an Enterprise Edition feature, which makes it more expensive than memory

3.2 p28

2) Tuning queries and indexes

This is the best long term solution

Sometimes queries are vastly OVER estimating their memory grant
- This can be due to code problems
- This can be due in part to bad statistics

Find the queries running when this wait starts happening

Look at which ones have the big memory grant, and tune them

Tools to help:
- **sp_BlitzCache @SortOrder = 'memory grant'**
- sp_BlitzWho shows queries running now
- A monitoring tool that trends wait stats, running queries

3.2 p29

3) Adding more memory

Take a hard look and see if your instance is under provisioned

Memory is one of the cheapest ways you can improve performance

As you saw in the demo here, it's not just important for caching data pages

It's also vitally important to make sure your queries can do sorts, joins, etc!

3.2 p30

Standard Edition memory limits

SQL Server 2005 and SQL Server 2008:

- Standard Edition does not have a 64GB memory limit, so having at minimum 128GB of memory is a no-brainer if your data working set needs it

SQL Server 2008R2 and SQL Server 2012:

- If you have less than 64GB of memory, add memory to at least use what you can and see if it alleviates the problem
- If you can't fix the issue with indexes and queries after that, an upgrade may be needed to get more memory

SQL Server 2014+

- SQL Server's buffer pool can be allowed 128GB of memory

3.2 p31

RESOURCE_SEMAPHORE fixes

Check your configuration:

- What is "max server memory (MB)" set to?
- How much memory does the SQL Server have altogether?

Review data sizes:

- How much data is in the databases?
- How much data churn in memory is going on at the time that RESOURCE_SEMAPHORE happens? (One way to check this is to see if the "page life expectancy" counter drops or stays at a low level during the period, indicating data churn)

3.2 p32

RESOURCE_SEMAPHORE fixes

What queries are running when this occurs?

- What is their desired memory grant?

Do the queries need that much of a memory grant?

- In some cases it's an over-estimate– is that the case? This can be due to linked servers, functions in joins, etc.

Is parameter sniffing part of the problem?

- Sometimes an execution plan with a very high grant gets in cache and is then re-used over and over again – but most values the plan is run with don't need the high grant

Can indexes help?

- Index tuning can make queries more efficient and dramatically reduce the size of memory grants

3.2 p33

Recap

3.2 p34

RESOURCE_SEMAPHORE

Queries need a memory grant to start running

SQL Server can't grant it due to bad memory pressure

Find the queries causing it:

- sp_BlitzCache @SortOrder = 'memory grant'
- sp_BlitzWho

Tuning queries & indexes is usually the cheapest fix

Got less than 32-64GB RAM? May need to add some.

3.2 p35

Setting up for the lab

1. Restart the SQL Server service (clears stats)
2. Restore your StackOverflow database
3. Copy & run the setup script:
   BrentOzar.com/go/serverlab5
4. Start SQLQueryStress:
   1. File Explorer, D:\Labs, run SQLQueryStress.exe
   2. Click File, Open, D:\Labs\ServerLab5.json
   3. Click Go

3.2 p36