



BRENT OZAR
UNLIMITED®

Assess a Workload and Find Out What's Slow

How to assess a workload

Two part methodology

1. Measure and interpret bottlenecks using SQL Server wait statistics
2. Identify top queries and assess how they relate to the bottlenecks

Measure bottlenecks with wait statistics



© SQLIntersection. All rights reserved.
<http://www.SQLintersection.com>

SQL Server loves cooperation



© SQLIntersection. All rights reserved.
<http://www.SQLintersection.com>

Meet the SQLOS

SQL Server is uppity– it takes over work from the OS

Why not let Windows do the work?

- Windows now prevents any single thread from taking over
- Threads get a time-slice, then Windows pulls ‘em off the processor (this thread has been pre-empted!)

SQL Server is different: threads co-operatively yield to each other

SQL “Schedulers” manage the process of putting threads into action when another thread yields



5

© SQLIntersection. All rights reserved.
<http://www.SQLintersection.com>

Why yield?

CPU is limited

When a query is waiting on another resource, it doesn't need to be active on the CPU



6

© SQLIntersection. All rights reserved.
<http://www.SQLintersection.com>

Executing != Running

An executing query may be waiting:

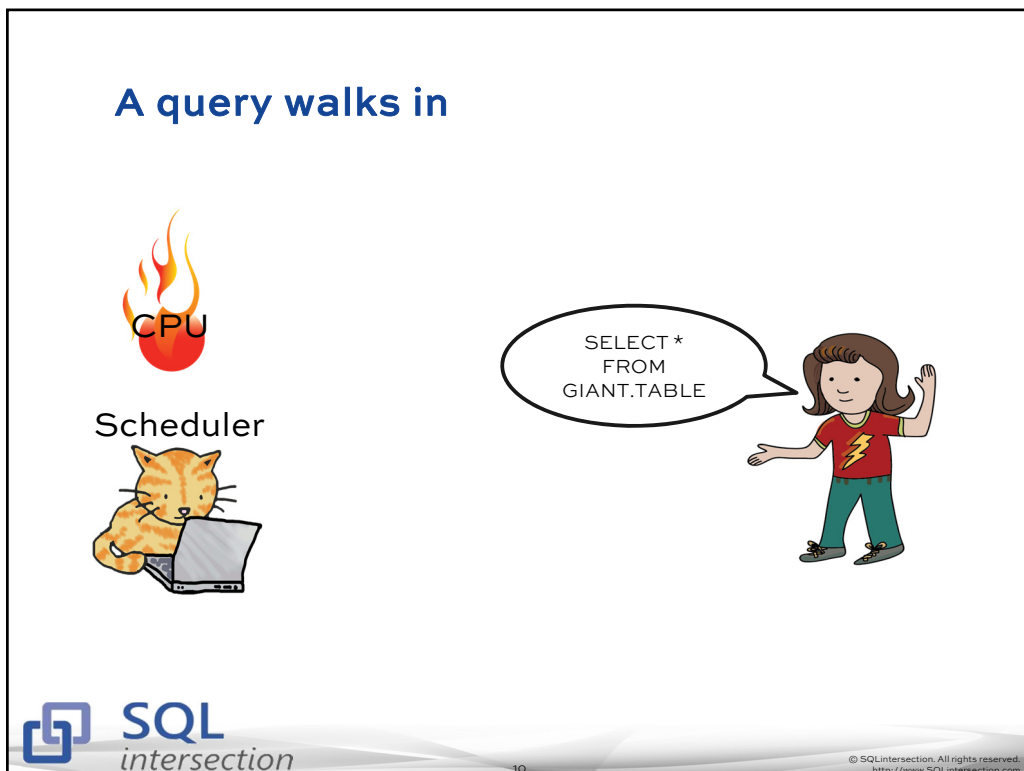
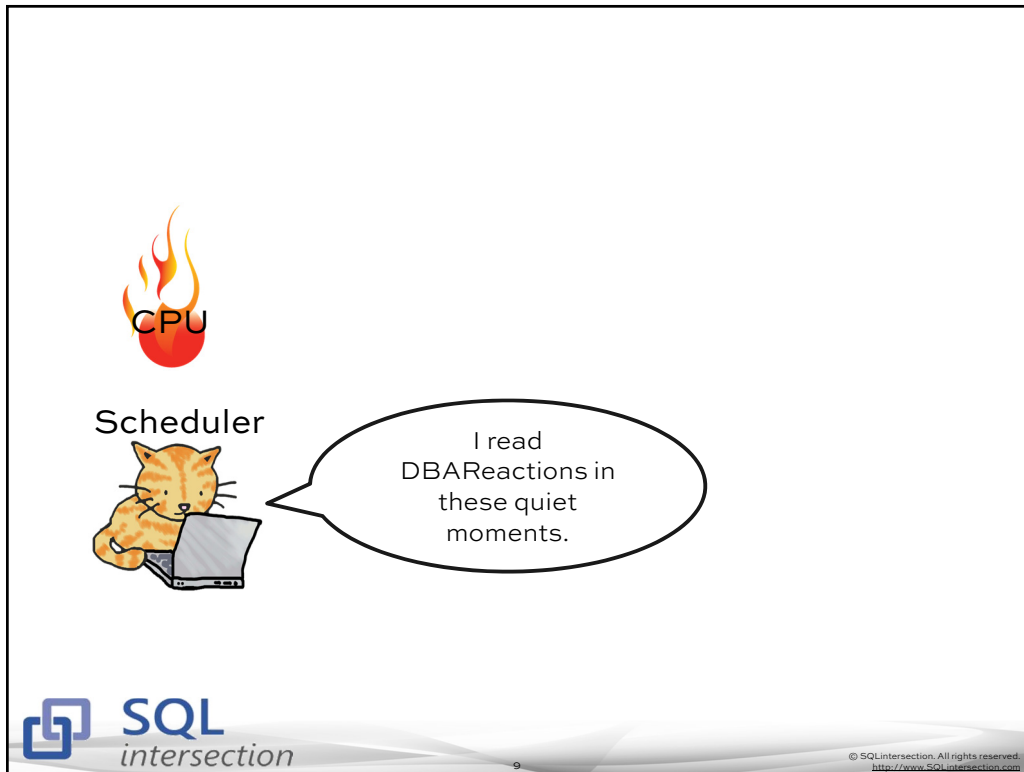
- For a resource (a lock, a page from storage, etc)
- For a signal/CPU (if it's ready to go again)

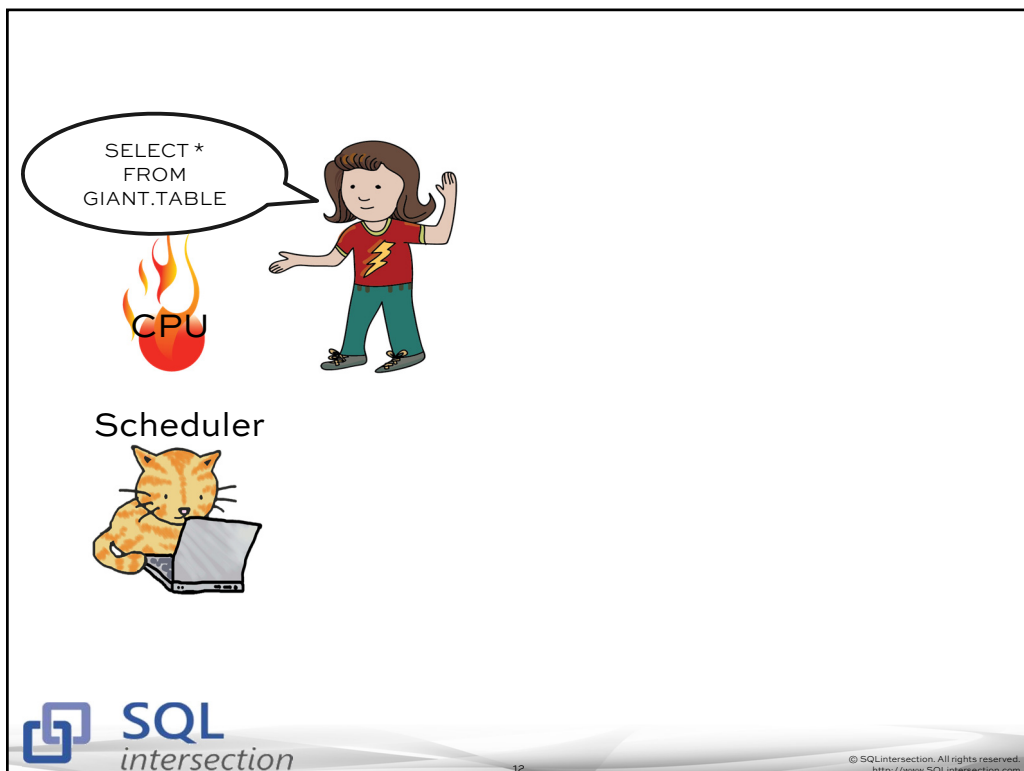
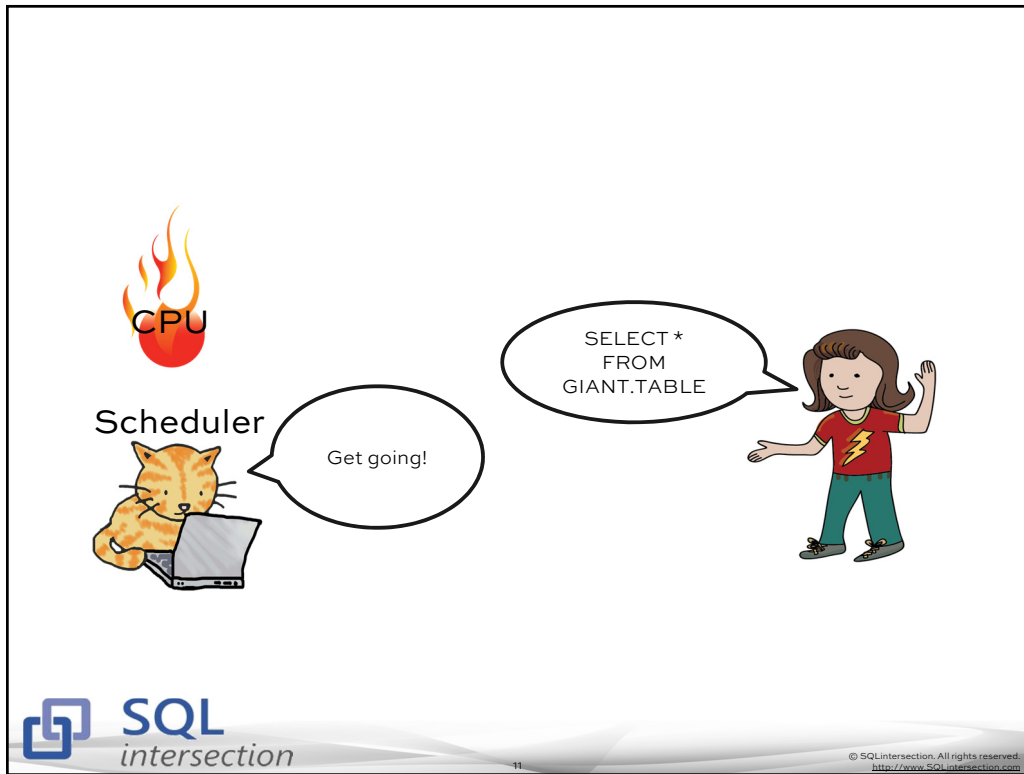
SQL Server tracks:

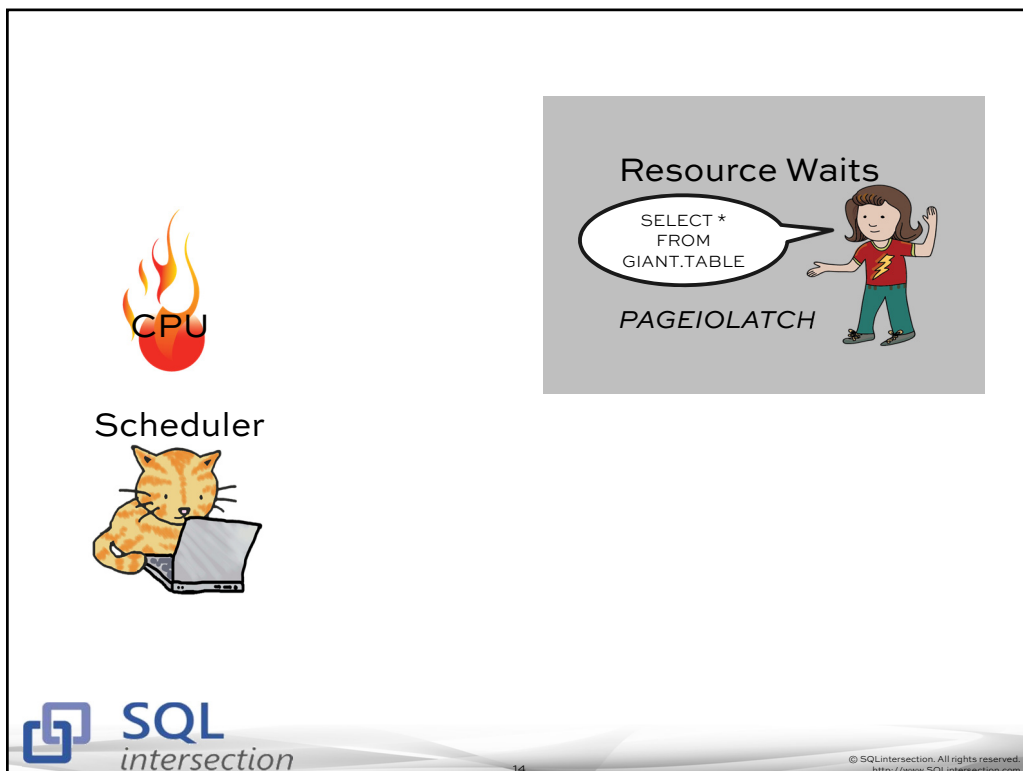
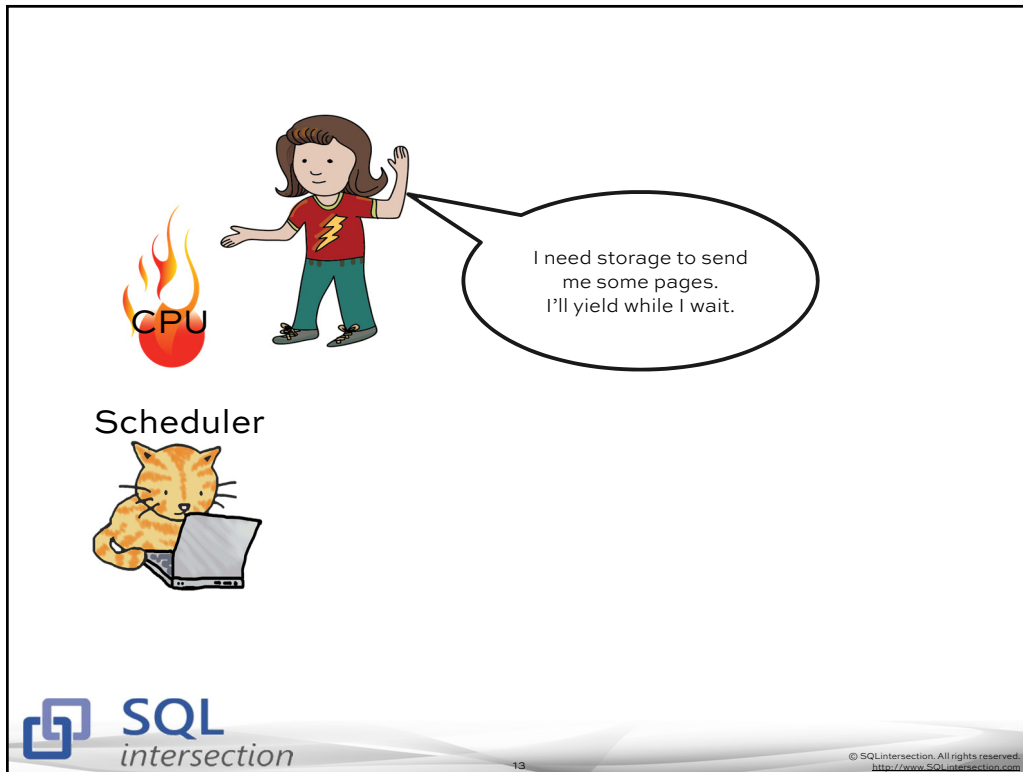
- What queries have run
- What execution plans they used
- How long queries were waiting
- What they were waiting for

Cat pictures will make this clearer

Let's look at an example







A new query...



CPU

Scheduler



Resource Waits

SELECT *
FROM
GIANT.TABLE

PAGEIOLATCH



SELECT COL1
FROM
GIANT.TABLE
WHERE COL2='y'



CPU

Scheduler



New guy, it's
all you.

Resource Waits

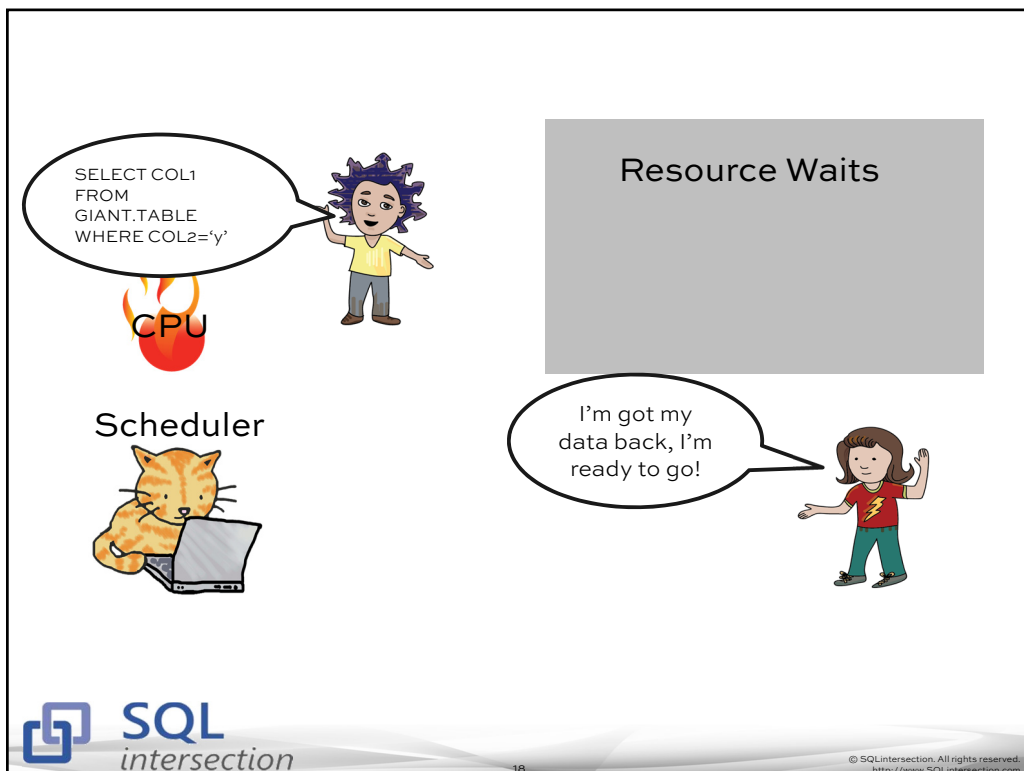
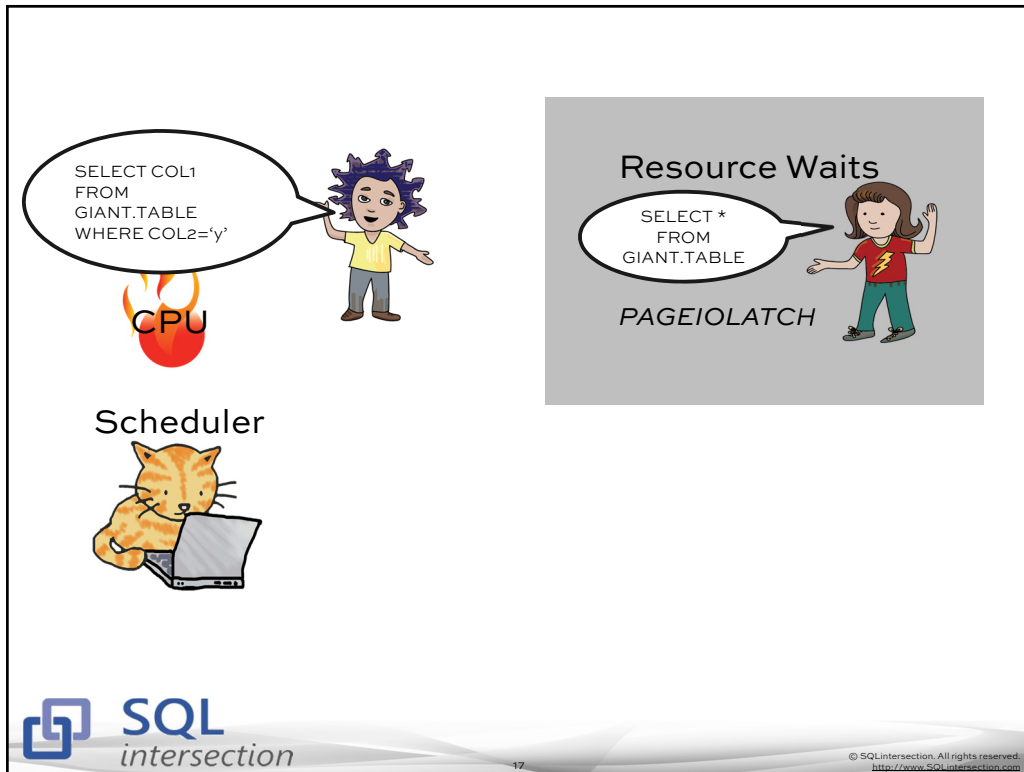
SELECT *
FROM
GIANT.TABLE

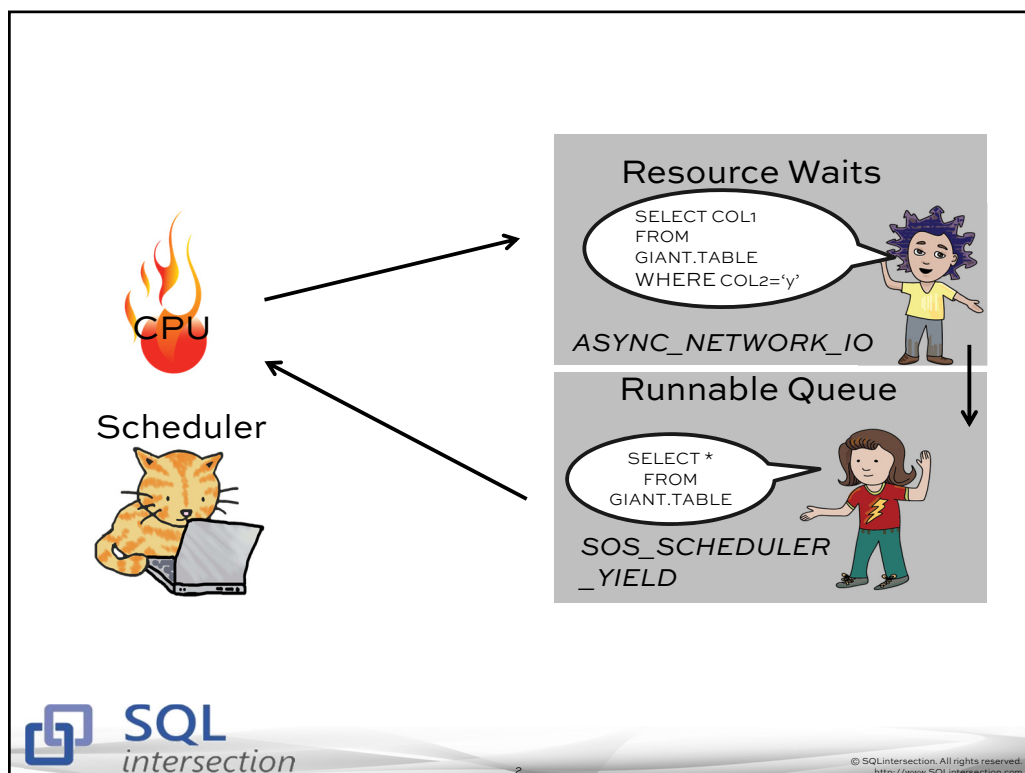
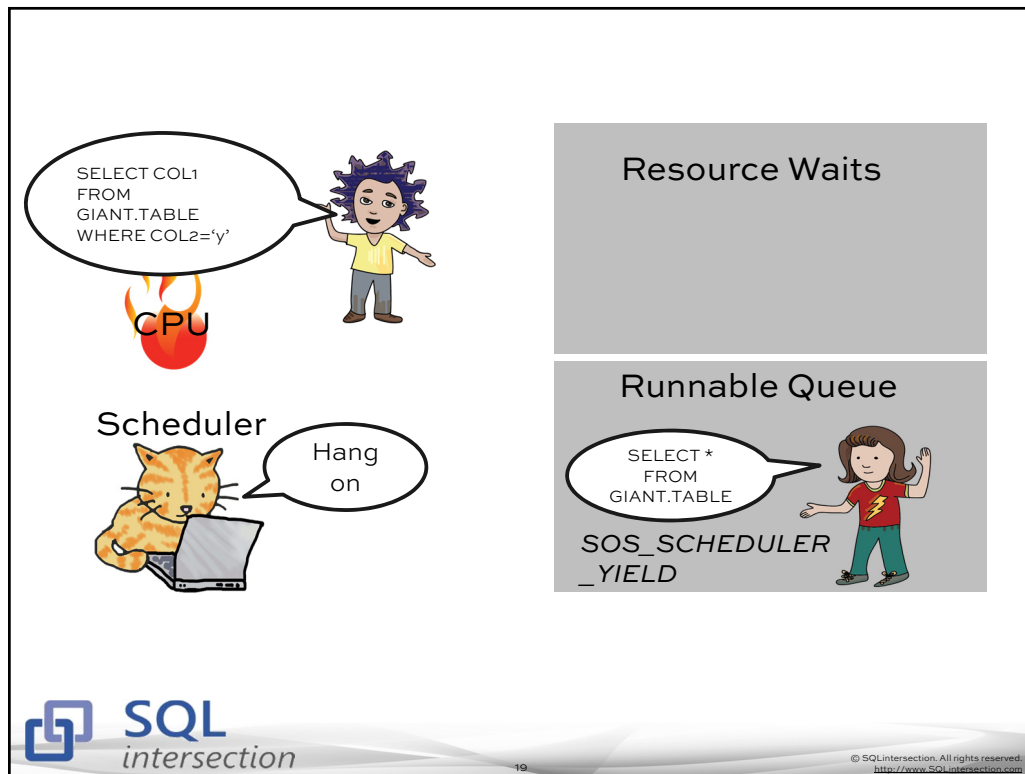
PAGEIOLATCH



SELECT COL1
FROM
GIANT.TABLE
WHERE COL2='y'







Waits identify SQL Server's bottlenecks

Is your server slowed down by....

- Lock waits?
- Storage waits?
- Delays by the application consuming data?
- CPU related waits?

Dynamic Management views:

- sys.dm_os_wait_stats ← Aggregate waits
- sys.dm_os_waiting_tasks ← Who is waiting on what (*right now*)

How to track wait stats and find top queries with free tools

Healthy vs. unhealthy monitoring

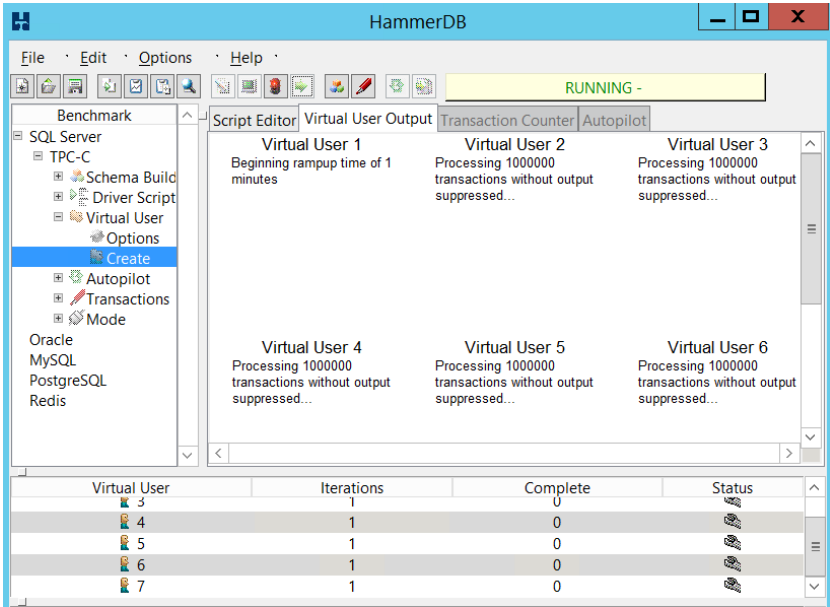
We're fans of:

- Tracking waits with a mature monitoring tool
- Taking a sample of waits and activity over a short duration with sp_AskBrent® (or your favorite script)
- Looking at queries running this instant and their related waits with Adam Machanic's sp_WhoIsActive
- Periodically reviewing top queries with sp_BlitzCache® (or your favorite script)

We're not big fans of:

- Automating frequent queries of DMVs for data you're not actively using
- The built-in Activity Monitor in SQL Server Management Studio

Demo: The terrible tempdb workload



The screenshot shows the HammerDB application window. The title bar says "HammerDB". The menu bar includes "File", "Edit", "Options", and "Help". The toolbar contains various icons. The left sidebar shows a tree view with "Benchmark" expanded, containing "SQL Server", "TPC-C", "Schema Build", "Driver Script", "Virtual User", "Options", "Create", "Autopilot", "Transactions", and "Mode". Below this are "Oracle", "MySQL", "PostgreSQL", and "Redis". The main area is divided into tabs: "Script Editor", "Virtual User Output", "Transaction Counter", and "Autopilot". The "Virtual User Output" tab is active, showing six virtual users (1-6) and their status. Below this is a table with columns: "Virtual User", "Iterations", "Complete", and "Status".

Virtual User	Iterations	Complete	Status
3	1	0	
4	1	0	
5	1	0	
6	1	0	
7	1	0	

SQL intersection

© SQLIntersection. All rights reserved.
<http://www.SQLIntersection.com>

1. Query waits since startup

```
/* Pseudo-code for readability */
/* The full query is in your course downloads */

SELECT TOP 25
  /* Wait Type, Metrics */
FROM sys.dm_os_wait_stats os
LEFT JOIN
  (VALUES('IGNORE_ME')) /*waits to ignore list here*/)
  AS iw (wait_type) on os.wait_type=iw.wait_type
WHERE iw.wait_type is null
ORDER BY [Wait ms] DESC;
```

Waits since startup

Wait Type	Wait ms	% Wait Time	# Waits	Avg Wait ms	Sampled
CXPACKET	14,214,141	58.3	38,713,761	0.4	9/23/14 8:10 AM
PAGELATCH_UP	8,020,989	32.9	12,886,674	0.6	9/23/14 8:10 AM
PAGELATCH_SH	1,586,896	6.5	10,703,506	0.1	9/23/14 8:10 AM
LCK_M_X	243,721	1	11,212	21.7	9/23/14 8:10 AM
LCK_M_S	155,381	0.6	11,690	13.3	9/23/14 8:10 AM

Waits this instant: sp_WhoIsActive

```
EXEC dbo.sp_WhoIsActive;
GO
```

	dd hh:mm:ss.mss	session...	sql_text	login_na...	wait_info
1	00 00:20:12.073	55	<?query -- select cntr_value from sys.dm_os_perf...	sa	NULL
2	00 00:00:00.390	60	<?query -- SELECT TOP(5000) a.name, replicat...	sa	(199ms)LCK_M_X
3	00 00:00:00.373	61	<?query -- SELECT TOP(5000) a.name, replicat...	sa	(2ms)PAGELATCH_UP:tempdb:1(PFS)
4	00 00:00:00.373	59	<?query -- SELECT TOP(5000) a.name, replicat...	sa	(2ms)PAGELATCH_UP:tempdb:1(GAM)
5	00 00:00:00.373	56	<?query -- SELECT TOP(5000) a.name, replicat...	sa	(198ms)LCK_M_X
6	00 00:00:00.356	57	<?query -- SELECT TOP(5000) a.name, replicat...	sa	NULL
7	00 00:00:00.060	58	<?query -- SELECT TOP(5000) a.name, replicat...	sa	NULL

Waits over a sample period: sp_AskBrent

```
-- BrentOzar.com/AskBrent
EXEC dbo.sp_AskBrent @ExpertMode=1, @Seconds=10;
GO
```

100 % - <

Priority	FindingsGroup	Finding	URL
1	0	sp_AskBrent (TM) v10 as of Jan 22 2014 12:00AM	From Brent Ozar Unlimited http://www.BrentOzar.com/askbrent/
2	200	Wait Stats	CXPACKET http://www.brentozar.com/sql/wait-stats/#CXPACKET
3	200	Wait Stats	PAGELATCH_SH http://www.brentozar.com/sql/wait-stats/#PAGELATCH_SH
4	200	Wait Stats	PAGELATCH_UP http://www.brentozar.com/sql/wait-stats/#PAGELATCH_UP
5	210	Query Stats	Most Resource-Intensive Queries http://BrentOzar.com/go/topqueries
6	255	Thanks!	From Brent Ozar Unlimited http://www.BrentOzar.com/askbrent/

< |||

Pattern	Sample Ended	Seconds Sam...	wait_type	Wait Time (Secon...	Signal Wait Time (Secon...	Percent Signal Wa	
1	WAIT STATS	2014-09-23 08:20:31.230	10	CXPACKET	177.8	17.4	9.8
2	WAIT STATS	2014-09-23 08:20:31.230	10	PAGELATCH_UP	98.9	10.4	10.5
3	WAIT STATS	2014-09-23 08:20:31.230	10	PAGELATCH_SH	19.3	8.5	44.0
4	WAIT STATS	2014-09-23 08:20:31.230	10	LCK_M_X	4.5	0.0	0.0
5	WAIT STATS	2014-09-23 08:20:31.230	10	LCK_M_S	2.8	0.0	0.0
6	WAIT STATS	2014-09-23 08:20:31.230	10	LATCH_EX	1.6	0.7	43.8



© SQLIntersection. All rights reserved.
<http://www.SQLIntersection.com>

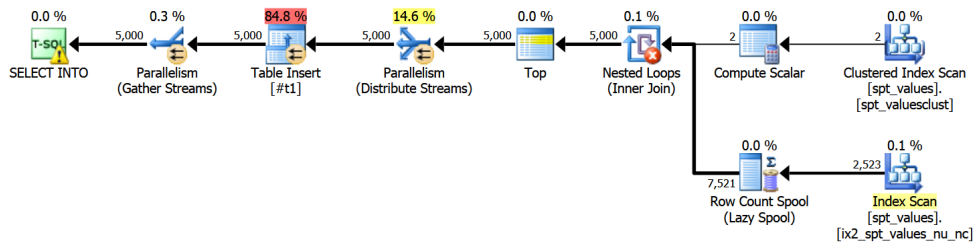
Top queries

```
exec sp_BlitzCache;
GO
```

Database	Cost	Query Text	Query Type	Warnings	Executions / Minute
1	CriticalAppDB	NULL	CREATE PROCEDURE dbo.tem...	Procedure: temptable	No Join Predicate, Parallel, Plan... 514.3333
2	CriticalAppDB	9.84455	SELECT TOP(5000) a.name, ...	Statement	No Join Predicate, Parallel, Plan... 514.3333
3	CriticalAppDB	NULL	CREATE PROCEDURE dbo.tabl...	Procedure: tablecol...	Implicit Conversions 117.3333
4	CriticalAppDB	0.435988	SELECT TOP 1 OBJECT_NAM...	Statement	Implicit Conversions 117.3333
5	CriticalAppDB	NULL	CREATE PROCEDURE dbo.use...	Procedure: userlist	Plan Warnings, Implicit Convers... 126.6667
6	CriticalAppDB	0.0120253	SELECT count(*) FROM sys...	Statement	Plan Warnings, Implicit Convers... 126.6667

Top query's execution plan

Does it use tempdb?



Waits and top queries

Wait types:

- PAGELATCH_UP
- PAGELATCH_SH

sp_WhoIsActive confirms SGAM and PFS page types

We have only one tempdb data file and heavy use of tempdb

First actions:

1. We add more data files to tempdb
2. We make sure the data files are all equally sized

Wait stat comparison

```
EXEC dbo.sp_AskBrent @ExpertMode=1, @Seconds=10;  
GO
```

```
--First Sample  
--CXPACKET 190.9  
--PAGELATCH_UP 94.6  
--PAGELATCH_SH 16.2
```

	Pattern	Sample Ended	Seconds ...	wait_type	Wait Time (Secon...
1	WAIT STATS	2014-09-23 08:52:20.810	10	CXPACKET	239.5
2	WAIT STATS	2014-09-23 08:52:20.810	10	PAGELATCH_UP	44.1
3	WAIT STATS	2014-09-23 08:52:20.810	10	PAGELATCH_SH	12.7

Top Waits

CXPACKET is even higher now

Remember that top query plan from sp_BlitzCache

- Parallel
- Est cost: 9.48

And why are we running these top queries so often?

- What is running them?
- Should it run at this frequency?

It's not uncommon to find this high execution rate in the real world

It might be your app, or it might be something else!

Observations

CXPACKET went up

PAGELATCH_UP and PAGELATCH_SH went down
(but are still significant)

We start thinking about that top query

- Does it really need to go parallel?

We test it out with a MAXDOP 1 hint and it's just as fast

We raise 'Cost Threshold for Parallelism'

```
EXEC sp_configure 'cost threshold  
for parallelism',50;  
GO  
  
SELECT *  
FROM sys.configurations  
WHERE value <> value_in_use;  
GO  
  
RECONFIGURE;  
GO
```

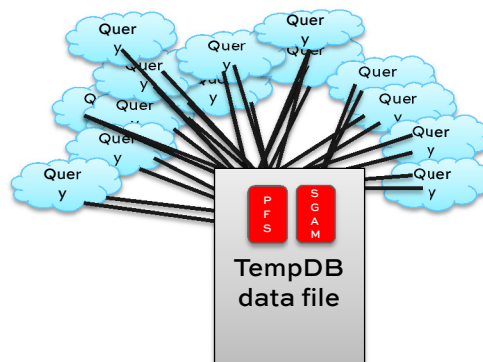
Wait stats after this change

```
EXEC dbo.sp_AskBrent @ExpertMode=1, @Seconds=10;  
GO
```

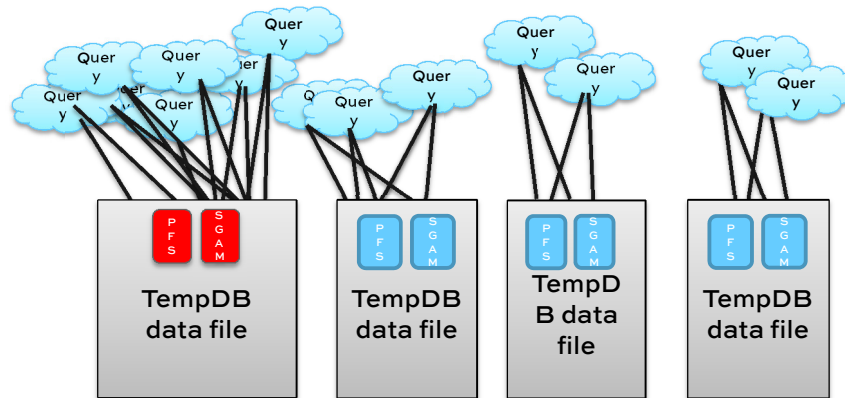
```
--First Sample  
--CXPACKET 190.9  
--PAGELATCH_UP 94.6  
--PAGELATCH_SH 16.2
```

100% - <					
Results Messages					
	Priority	FindingsGroup	Finding		URL
<	0	sp_AskBrent(TM) v10 as of Jan 23 2014 12:00AM - From Brent Green Unlimited			http://www.BrentGreen.com
	Pattern	Sample Ended	Seco...	wait_type	Wait Time (Seconds)
1	WAIT STATS	2014-09-23 09:01:41.183	10	PAGELATCH_UP	21.0
2	WAIT STATS	2014-09-23 09:01:41.183	10	SOS_SCHEDULER_YIELD	4.1
3	WAIT STATS	2014-09-23 09:01:41.183	10	PAGELATCH_SH	3.8

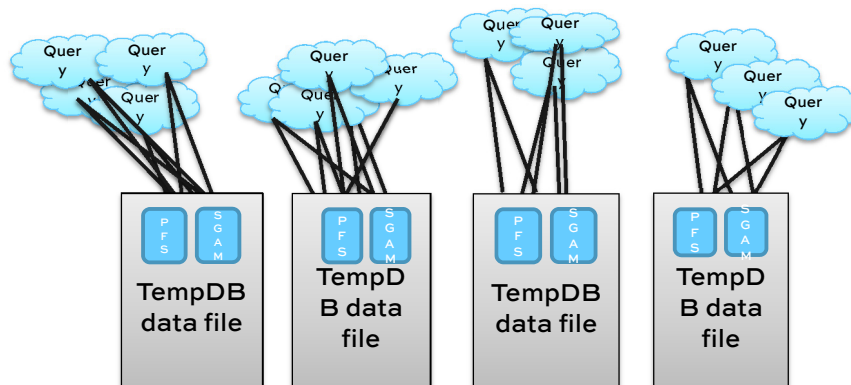
Allocating new objects: On tempdb data file



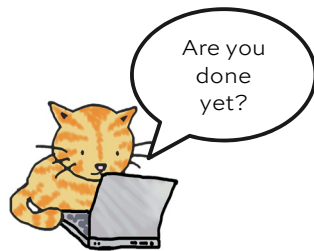
Four uneven tempdb data files



4 even tempdb data files



Parallelism takes work



CXPACKET



Done with
3991 of
3991 rows



Done with
4005 of
4005 rows



Done with
4200 of
5001 rows



Done with 2
of 2 rows

Parallel index scan

Ideally, work is distributed evenly.
That doesn't always happen.

Parallelism wasn't worth it here

Plan Tree										
Operation	Object	Est Cost	Est Subtree Cost	Actual Rows	Est Rows	Thread 0	Thread 1	Thread 2	Thread 3	Thread 4
SELECT INTO		100.0%	100.0%	5,000	5,000					
Parallelism (Gather Streams)		0.3%	100.0%	5,000	5,000	5,000				
Table Insert	[#12]	84.8%	99.7%	5,000	5,000	0	1,250	1,250	1,250	1,250
Parallelism (Distribute Streams)		14.6%	14.8%	5,000	5,000	0	1,250	1,250	1,250	1,250
Top		0.0%	0.3%	5,000	5,000	0	5,000			
Nested Loops (Inner Join)		0.1%	0.3%	5,000	5,000	0	5,000			
Compute Scalar		0.0%	0.0%	2	2					
Clustered Index Scan	[mssqlsystemresource].[sys].[spt_value...]	0.0%	0.0%	2	2	0	2			
Row Count Spool (Lazy Sp...)		0.0%	0.2%	5,000	7,521	0	5,000			
Index Scan	[mssqlsystemresource].[sys].[spt_value...]	0.1%	0.1%	2,523	2,523	0	2,523			

Plan tree viewed in SQL Sentry Plan Explorer.
Actual execution plan.

The terrible tempdb workload

Configuration	Top Wait Stats (30 second sample)	Avg Batch Requests / Min
1 tempdb file (default) Cost threshold 5 (default)	CXPACKET 506.2 PAGELATCH_UP 317.6 PAGELATCH_SH 64.4	642
1 tempdb file (default) Cost threshold 50	PAGELATCH_UP 68.0 SOS_SCHEDULER_YIELD 14.8 PAGELATCH_SH 9.0	1,631 WOW!
4 tempdb files cost threshold 50	PAGELATCH_UP 63.1 SOS_SCHEDULER_YIELD 18.7 PAGELATCH_SH 9.9	1,742
8 tempdb files cost threshold 50	PAGELATCH_UP 50.0 SOS_SCHEDULER_YIELD 21.9 PAGELATCH_SH 7.4	2,713
20 tempdb files cost threshold 50	PAGELATCH_UP 32.2 SOS_SCHEDULER_YIELD 27.8 PAGELATCH_SH 8.2	3,155 WOW!
	SOS_SCHEDULER_YIELD 34.3	



How did you know
it wasn't slow
storage?

- No PAGEIOLATCH_* waits
- Didn't see slow storage read and write times in sp_AskBrent®

How many tempdb files?

sp_Blitz® warns if you just have one, or if they're unevenly sized

You can check your count anytime:

- `exec tempdb..sp_helpfile`

Be proactive and don't just go with one tempdb data file– but don't go crazy.

For general guidelines: BrentOzar.com/go/setup

Danger...

Some free tools only show “resource” waits

Activity Monitor, I'm looking at you

I want to know about queries sitting, waiting in the runnable queue, too!

I also like knowing specific wait type names

Resource Waits

```
SELECT COL1  
FROM  
GIANT.TABLE  
WHERE COL2='y'
```



Runnable Queue

```
SELECT *  
FROM  
GIANT.TABLE
```



Activity monitor vs sp_AskBrent

Resource Waits				
Wait Category	Wait Time (ms/sec)	Recent Wait Time...	Average Waiter C...	Cumulative Wait Tim...
Buffer Latch	10414	10176	10.4	1267
Lock	382	458	0.4	61
Latch	99	93	0.1	12
Buffer I/O	0	0	0.0	0
Compilation	0	0	0.0	0
Logging	0	0	0.0	0
Memory	0	0	0.0	0
Network I/O	0	0	0.0	0
Other	0	0	0.0	0

Activity Monitor

Data File I/O				
Recent Expensive Queries				

```
SQLQuery2.sql - JUICE01.master (sa (52)) *  
exec sp_AskBrent @ExpertMode=1, @Seconds=10;  
GO
```

sp_AskBrent

Pattern	Sample Ended	Seconds Sample	wait_type	Wait Time (Seconds)	Signal Wait T
1 WAIT STATS	2014-09-23 09:19:26.293	10	CXPACKET	187.5	21.4
2 WAIT STATS	2014-09-23 09:19:26.293	10	PAGELATCH_UP	92.2	10.4
3 WAIT STATS	2014-09-23 09:19:26.293	10	PAGELATCH_SH	17.4	6.8

Analyzing wait statistics

Wait Type Decoder Ring

Avoid mistakes

Don't fixate on percentage wait time

Also look at:

- Total time waiting vs. sample interval
- Average wait time
- Number of waits

Don't neglect looking at the top queries running in this period: the wait stats are tied to those queries

Wait Type	What it means
CXPACKET	Parallel queries are occurring– which isn't necessarily a bad thing
SOS_SCHEDULER_YIELD	Waiting to get on a CPU to crunch some data (in the runnable queue)
PAGEIOLATCH_**	Waiting to pull data from storage into memory
WRITELOG	Flushing a write to the transaction log
ASYNC_NETWORK_IO	Sending data to the client (Network OR client app can be slow)
PAGELATCH_**	Protecting data in memory from inconsistencies (“Don't scribble on my page while I'm using it!”) – could be Tempdb page contention
LCK_**	Waiting to get a lock– and can't. Locks needed depends on isolation level, transaction status, and type of query (select vs modification)
LATCH_**	Contention over a page in memory. Often related to parallelism – but you have to check the subtype (“latch class”) to know for sure.

This isn't just about hardware.
Query and index patterns
cause waits, too.

Example pattern	What you see in sp_BlitzCache	Can drive up waits, including...
Implicit conversion	High avg logical reads High avg CPU	PAGEIOLATCH_** CXPACKET SOS_SCHEDULER_YIELD
Critical missing indexes		
Cursors / while loops	High executions/second	WRITELOG
“After” triggers	Execution plans for the triggers showing high execution/CPU/IO in the cache	LCK_** (which can lead to a big backlog of queries, which can lead to CXPACKET and PAGEIOLATCH_** waits)
App running giant reports	High avg duration High avg logical reads High avg rows	ASYNC_NETWORK_IO

What to do

Your Mission

This requires practice

This is complex

- Lots of red herrings out there
- Easy to get distracted

But it really works

- We use these tools and this methodology in real-life consulting

The more you do this, the better you get at it!

A big part of this is clearing your assumptions about what the problems are, and what can and can't be changed

Get free tools

BrentOzar.com/go/Active

BrentOzar.com/Blitz

BrentOzar.com/AskBrent

Action Plan

1. **Measure your bottlenecks using wait stats**
 - Normal / low volume times
 - Peak volume times
2. **Analyze your top queries**
 - Look for relationships with your bottlenecks
 - Anti-patterns
 - Frequently run queries
 - Queries using lots of IO
3. **Keep the information on hand. When performance is a problem, take more samples and identify what is different**