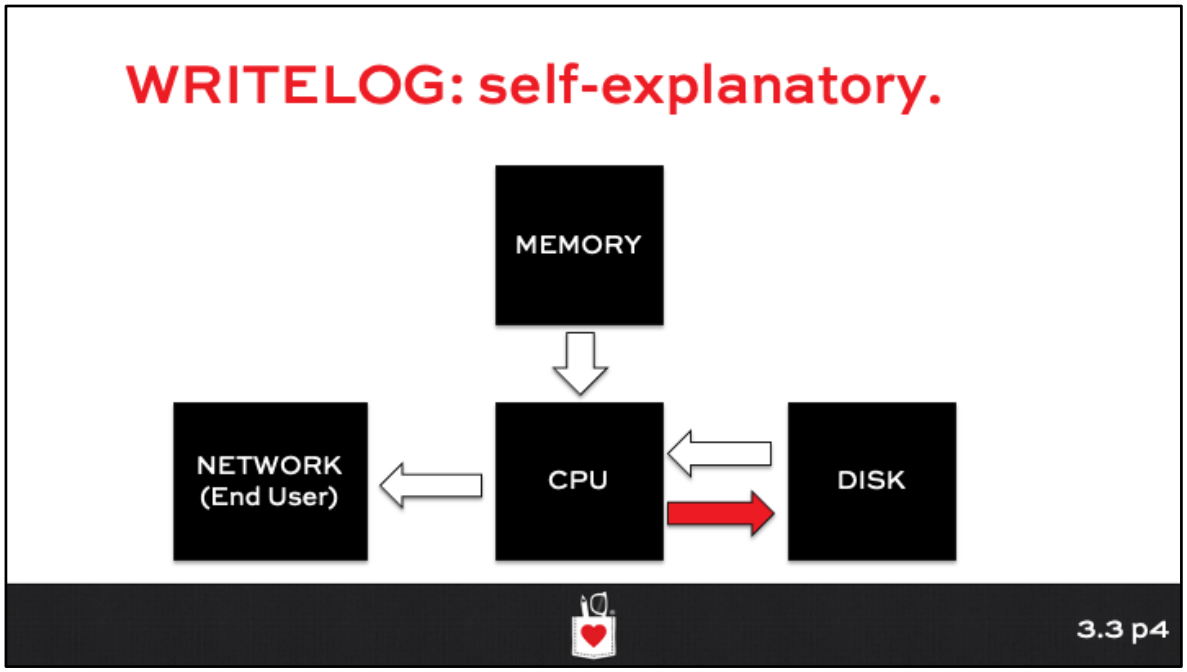
# Setting up for the lab

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

3.3 p1



# Two wait stat numbers matter.

Wait Time: total amount of time you've waited

Avg ms Per Wait: when we do wait on something, how long do we have to wait?

Shown off to the right in sp_BlitzFirst:

| Seconds Sample | wait_type | Wait Time (Seconds) | Per Core Per Second | Signal Wait Time (Seconds) | Percent Signal Waits | Number of Wai | Avg ms Per Wait |
|---|---|---|---|---|---|---|---|
| 16 | SOS_SCHEDULER_YIELD | 3555.8 | 55.6 | 3555.8 | 100.0 | 15714 | 226.3 |
| 16 | CXPACKET | 1027.7 | 16.1 | 1.5 | 0.1 | 985 | 1043.4 |
| 16 | THREADPOOL | 3.1 | 0.0 | 0.0 | 0.0 | 57 | 54.9 |
| 16 | LATCH_EX | 1.0 | 0.0 | 0.3 | 30.0 | 204 | 4.8 |
| 16 | LATCH_SH | 0.0 | 0.0 | 0.0 | 0.0 | 7 | 4.7 |

3.3 p5

# These two won't match.

**WRITELOG average wait time, milliseconds:**
when your query is waiting on the log file,
this is how long it's waiting.

**Drive & file response time:**
can be higher or lower, because you're not always waiting for these files.

Your storage team only cares about the latter.

3.3 p6

# Perfmon counters for more info

Performance Monitor counter:
Physical Disk: Avg Sec/Write (aka write latency)

Reported in whole seconds – use 3 decimal places for MS.

Microsoft says >3 milliseconds log writes are slow. Me: 20.

Good for both physical and virtual servers

Related counters: Physical Disk: Reads/sec, Writes/sec
(shows how much we're asking storage to work)

The more we ask storage to work, the slower it'll get.

3.3 p7

# Solving WRITELOG is easier.

The transaction log is typically relatively small.
(Unless you're storing files in the database. Don't do that.)

Delayed Durability: new option in 2014 to consider transactions committed before they hit the log file.

**Just one database involved?** Use a dedicated pair of mirrored drives, ideally solid state.

**Multiple databases involved?** May need to stripe across many drives in a RAID 10, ideally solid state.

# Kinda like WRITELOG.

Only seen in Always On Availability Groups in synchronous commit mode.

Only affects data modification (not selects).

Wanna go faster?

- Switch to async (ha ha ho ho)
- Check waits on the sync secondaries (may be disk-bottlenecked)
- Check network latency between replicas
- Separate non-critical data (staging tables, scratch space) into separate, non-sync AG databases

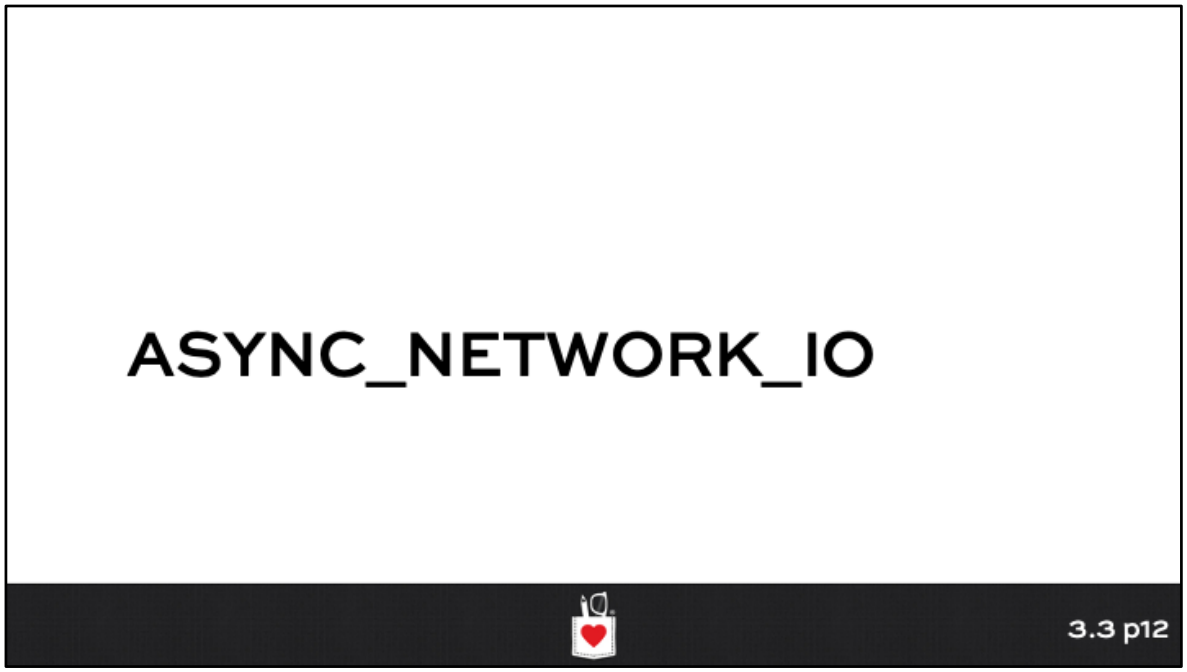
3.3 p10

## In-depth info from Microsoft

In case none of the following seem to apply to you:

https://blogs.msdn.microsoft.com/sql_server_team/troubleshooting-high-hadr_sync_commit-wait-type-with-always-on-availability-groups/

3.3 p11



## Our routine

1. Start a workload
2. Check vital stats with sp_BlitzFirst
3. List possible root causes
4. List mitigation options
5. Apply one of them
6. Check vital stats again

3.3 p13

# Preparation

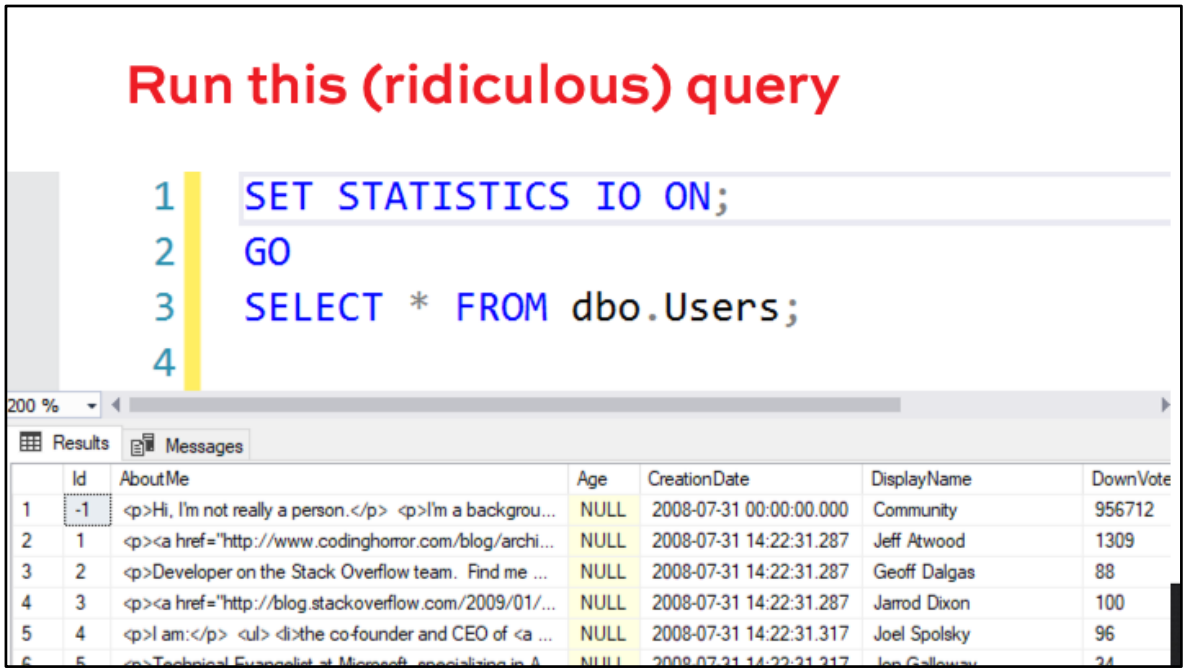
Attach the StackOverflow database

Set your max memory up high, parallelism to defaults:

```
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
```

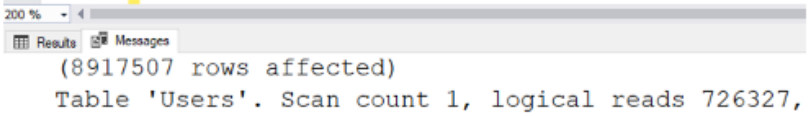
Set up a new window for sp_BlitzFirst:

```
sp_BlitzFirst @ExpertMode = 1, @Seconds = 30
```



# The query reads a lot of data

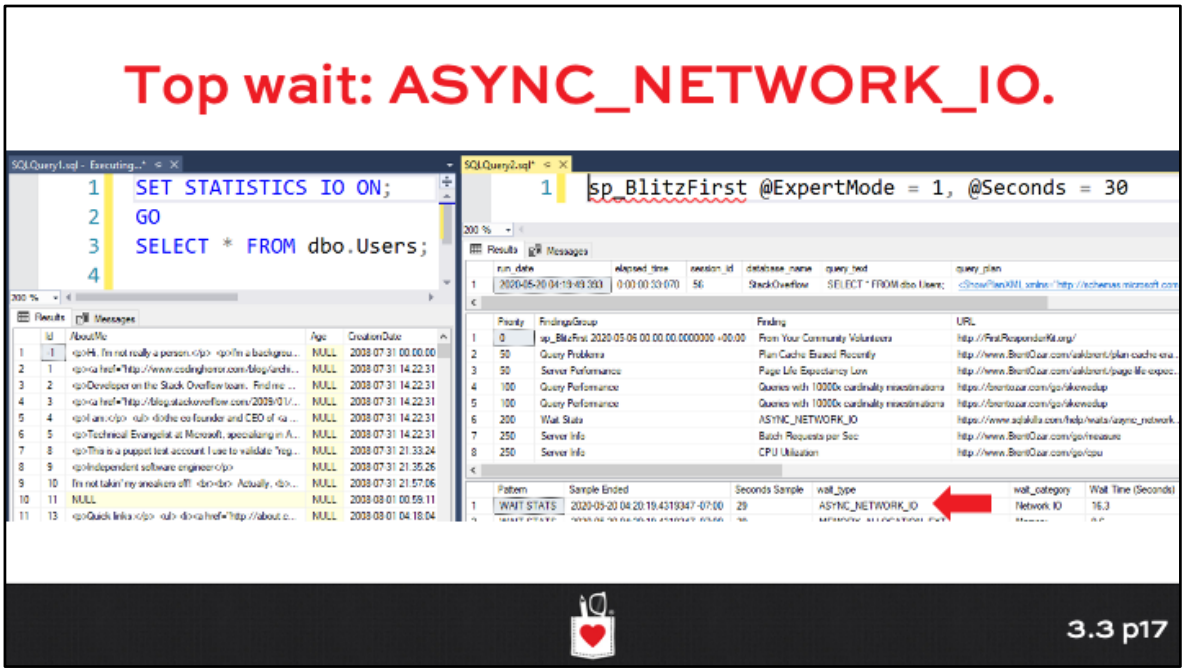
```
1  SET STATISTICS IO ON;
2  GO
3  SELECT * FROM dbo.Users;
4
```




```
(8917507 rows affected)
Table 'Users'. Scan count 1, logical reads 726327,
```

But reading data isn't the bottleneck:
this table fits in RAM, and is completely cached.

# Top wait: ASYNC_NETWORK_IO.



# ASYNC_NETWORK_IO

Slow client machines, underpowered app server VMs

Application processing data row-by-row
instead of just getting it all from SQL Server first

Slow network connections, especially WANs or VPNs

Good news! It's not a database problem.*

* - It's always a database problem.

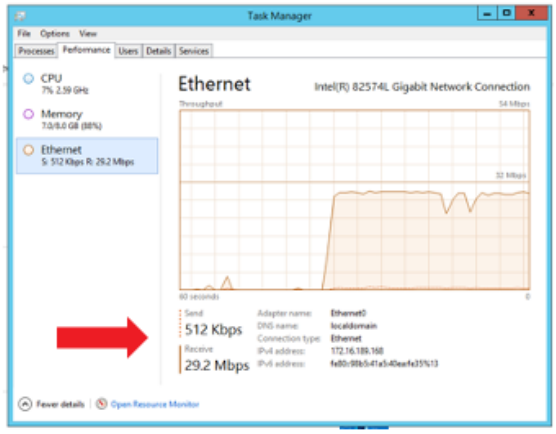
3.3 p18

# Sanity check: is it network?

1Gb Ethernet:~100MB/sec

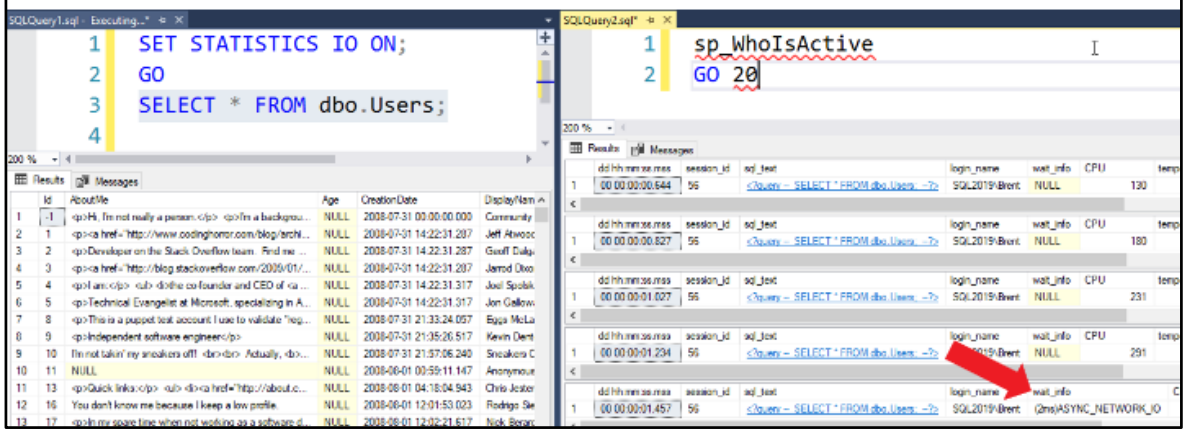
If you're actually saturating the network with query results, start asking tough questions.

(This screenshot is me downloading an ISO file.)




# Tracking down the apps involved

Run sp_WhoIsActive repeatedly and look for ASYNC_NETWORK_IO in the wait_info:



# Scroll across to the app, host

| host_name | database_name | program_name | st |
|---|---|---|---|
| SQL2019 | StackOverflow | Microsoft SQL Server Management Studio - Query | 2 |

Go to the application owner with the query & host

Ask to see the source code for what's running that query to find out if it's doing row-by-row processing

Check the app server to see if it's maxed out on CPU or RAM

3.3 p21

# Recap

3.3 p22

## Hardware waits

WRITELOG: we need to write less to the log, or get lower-latency transaction log storage.

HADR_SYNC_COMMIT: the cost of sync replication.

ASYNC_NETWORK_IO: it's not our problem, but we have to help the developers and sysadmins find it.

## 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

3.3 p24