

Performance Monitoring AlwaysOn Availability Groups

Anthony E. Nocentino
aen@centinosystems.com



Anthony E. Nocentino

- **Consultant and Trainer**
- **Founder and President of Centino Systems**
 - Specialize in system architecture and performance
 - Masters Computer Science (almost a PhD)
 - Friend of Redgate - 2016
 - Microsoft Certified Professional
- **email:** aen@centinosystems.com
- **Twitter:** @nocentino
- **Blog:** www.centinosystems.com/blog
- **Pluralsight Author:** www.pluralsight.com



Overview

- Motivation
- How availability groups move data
- Impact of replication latency on availability
- Monitoring techniques
- Demo
- Dealing with replication latency

Why is this important?

- Recovery Objectives
 - Recovery Point Objective - RPO
 - Recovery Time Objective - RTO
- Availability
 - How much data can we lose?
 - How fast will the system fail over?
- Monitoring and Trending
 - Establish a baseline for analysis - are we meeting those objectives?
 - Impact on resources
- Ownership
 - All of the components are monitored by the DBA

Data Movement In Availability Groups

- Transaction log blocks are replicated to secondaries
- Replication mode
 - Synchronous
 - Asynchronous
- Database mirroring endpoint

Network Based Replication

- Strong working relationship with network team
 - Maintenance - patching, network outages, database
- Network conditions can impact your AG's availability
 - Latency - how **long** it takes for a packet of data to traverse the network from source to destination.
 - Bandwidth - how **much** data can be moved in a time interval

Network Latency

- Often measured in milliseconds, sometimes microseconds
- Directly impacts network throughput
 - TCP sliding window
- `ping` isn't your best measure of latency, by default it doesn't include any load...measure **your** workload
- It's often up to us to **PROVE** to the network team there is an issue
 - `Pinging 192.168.2.1 with 32 bytes of data:`
 - `Reply from 192.168.2.1: bytes=32 time=1001ms TTL=128`

Database Synchronization States

- Not synchronizing
- Synchronized
- Synchronizing
- Reverting
- Initializing

<https://msdn.microsoft.com/en-us/library/ff877972.aspx>

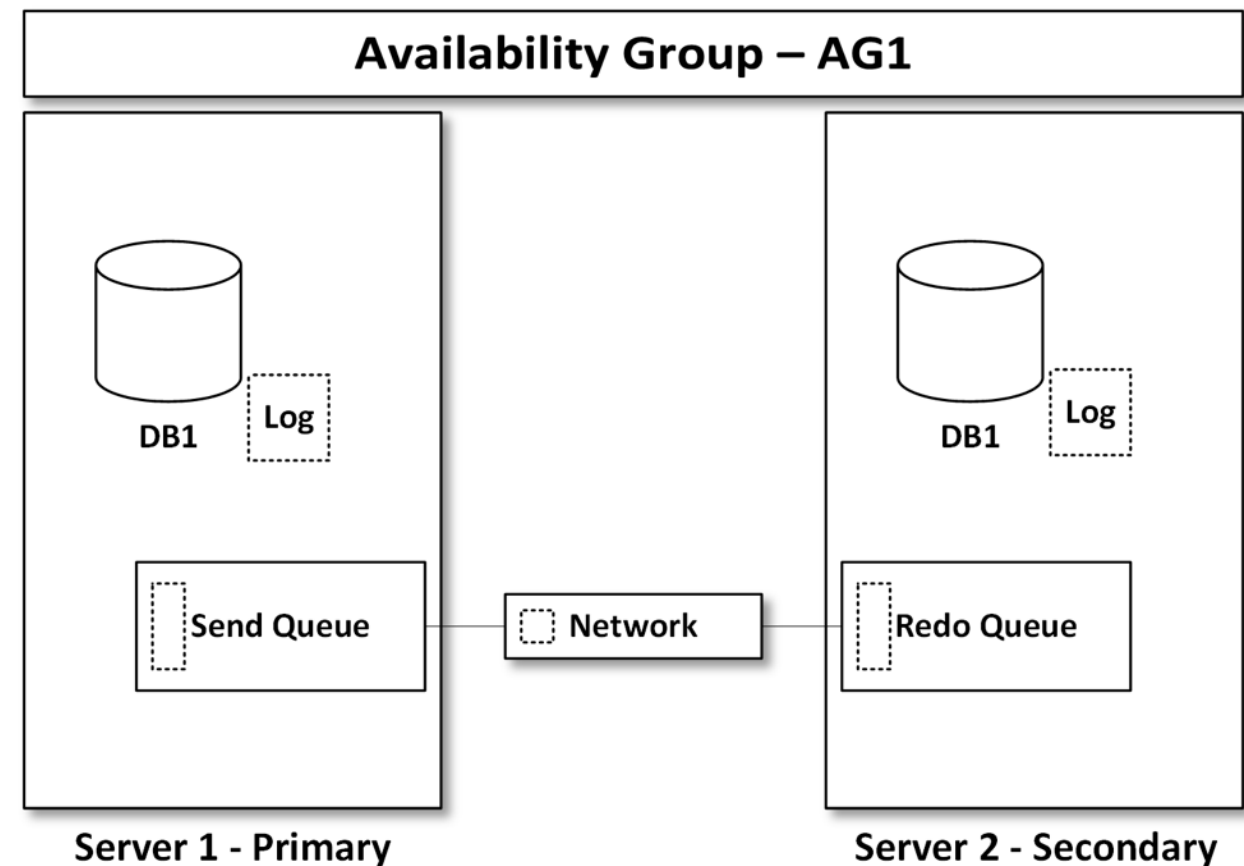
Failover Modes

- Automatic
 - Synchronous mode only
 - Synchronization state must be synchronized
- Manual
 - Synchronous or Asynchronous

<https://msdn.microsoft.com/en-us/library/hh213151.aspx>

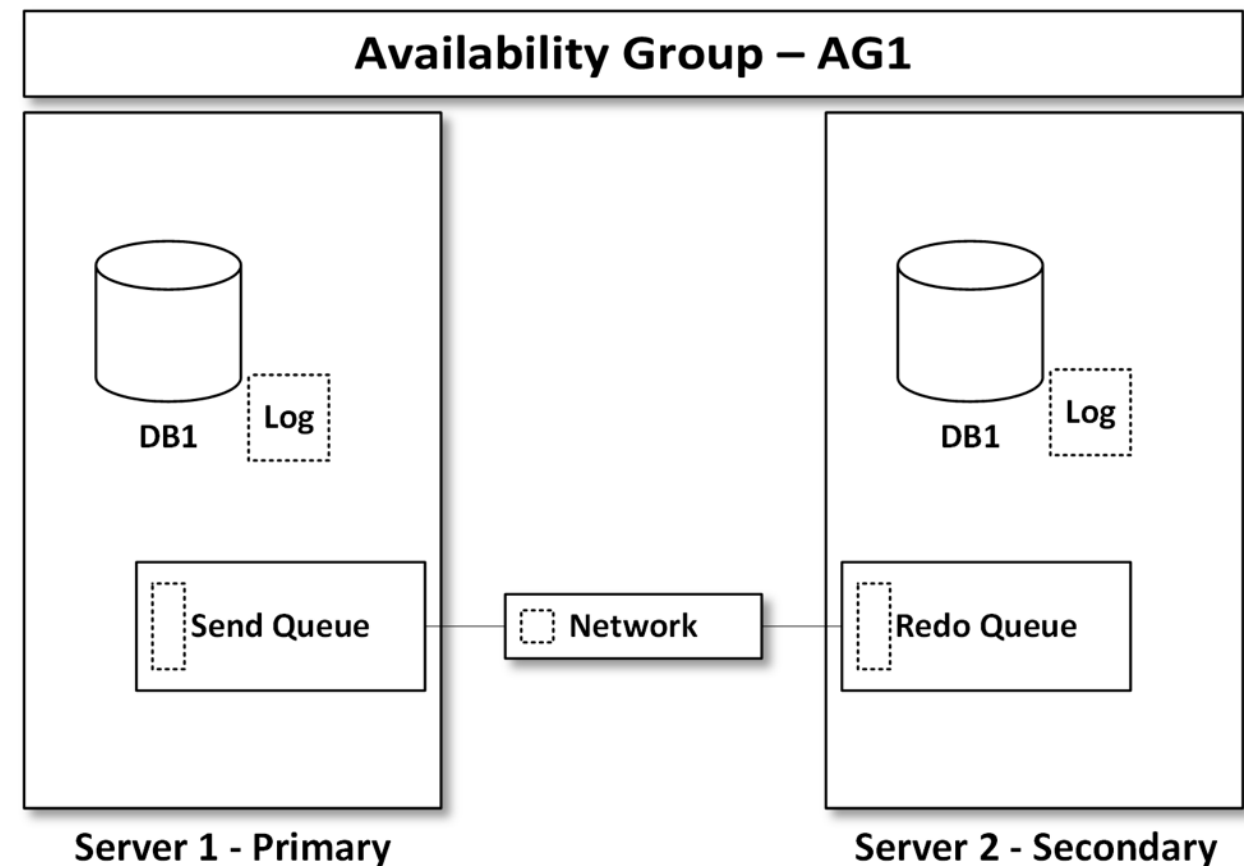
Send Queue

- Queues log blocks to be sent to the secondaries
- Each replica maintains it's own view of the send queue
- Queued data is at risk to data loss in the event of a primary failure
- The send queue can grow due to an unreachable secondary, network outage, network latency and large amount of data change



Redo Queue

- Queues log blocks received on the secondary
- Each replica has its own redo queue
- On failover, the redo queue must be completely processed
- The redo queue can grow due to a slow disk subsystem or resource contention or sustained outage and subsequent reconnection of a secondary



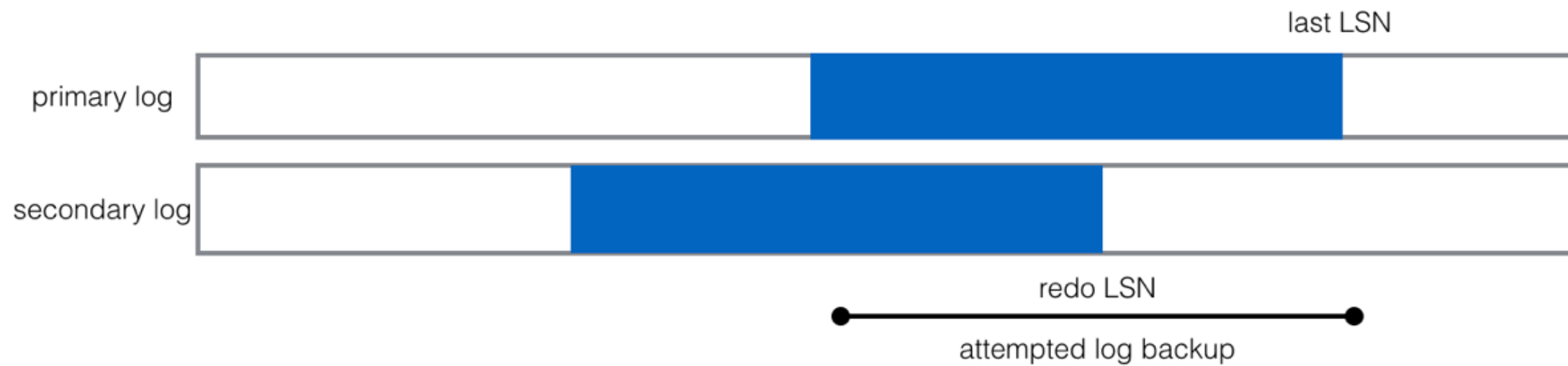
Send Queue Impact on Availability

- When log generation on primary exceeds the rate they can be sent to the secondaries...
 - No automatic failover
 - Data loss
 - Stale data for reporting from secondaries
 - Stale data for off-loaded backups on secondaries
 - Off-loaded log backups can fail
 - Transaction delay
 - Fill up transaction logs
- **Even in synchronous mode!**

Redo Queue Impact on Availability

- When log blocks received on the secondary exceed the rate they can be processed by the redo thread...
 - Delayed failover
 - Detect failure
 - Process Redo Queue
 - Crash recover database
 - Stale data for reporting from secondaries
 - Stale data for off-loaded backups on secondaries
 - Off-loaded log backups can fail
 - Transaction delay

Log Backups



Disconnected Replica

- When a synchronous secondary exceeds its session time-out, it changes to asynchronous commit mode
- When the secondary comes back online, it will attempt to re-sync, and resume synchronous mode
- `HADR_SYNCHRONIZING_THROTTLE` - time to go from `SYNCHRONIZING` to `SYNCHRONIZED`

Transaction Delay

- In synchronous mode, when secondaries are behind, queries on the primary can be delayed
 - HADR_SYNC_COMMIT
 - HADR_SYNCHRONIZING_THROTTLE - replica back online

Maintenance Events That Can Impact Availability

- Bulk data modifications
 - Database maintenance
 - Network or server maintenance
 - Unplanned outages
-
- Carefully plan maintenance
 - Collaborate with other teams!

Monitoring AG Performance

- Dynamic Management Views
 - `sys.dm_hadr_database_replica_states`
- Perfmon Counters
 - SQL Server:Availability Replica
 - Replication data - messages sent, bytes sent, flow control
 - SQL Server:Database Replica
 - Database data - log bytes sent, queue sizes, transaction delay per database

Measuring Replication Latency

- `sys.dm_hadr_database_replica_states`
 - `log_send_queue_size`
 - `log_send_rate`
 - `redo_queue_size`
 - `redo_queue_rate`
- On the primary there's a row for each database on each replica
- On the secondaries there's a row for each database on that replica
- Pull replication
- Offline
 - `log_send_queue_size` changes to NULL

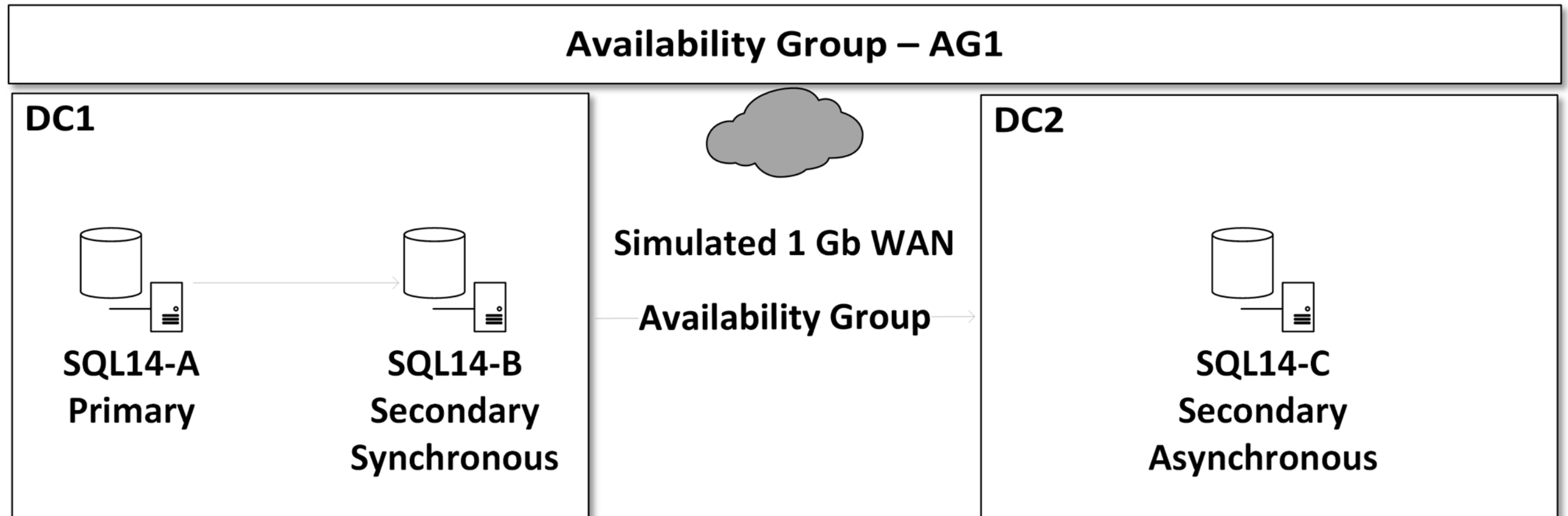
Measuring Replication Latency - **ugh!!!**

- **Well, it looks like `sys.dm_hadr_database_replica_states` doesn't report the correct values for `log_send_rate` and `redo_queue_rate`**
 - Documented as KB
 - Reported on Connect
 - <https://connect.microsoft.com/SQLServer/Feedback/Details/928582>
 - Known bug in SQL Server 2012 or 2014
 - <https://support.microsoft.com/en-us/kb/3012182>
 - Cumulative Update 5 or better
 - Observed in SQL 2016 CTP 3.3 - just increases
 - Perfmon!

Monitoring Tools

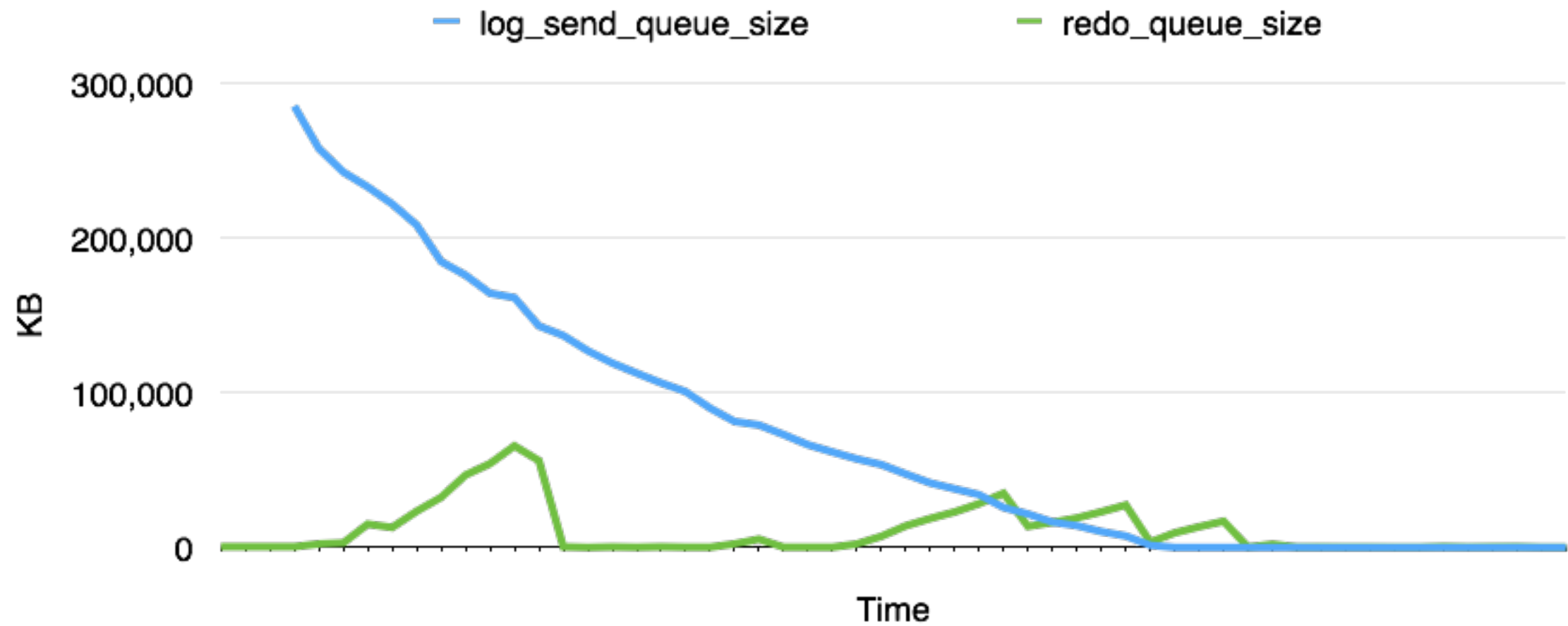
- Build your own
- AlwaysOn Dashboard
- Third Party Tool
 - SQL Sentry Performance Advisor
 - Redgate SQL Monitor

Demo

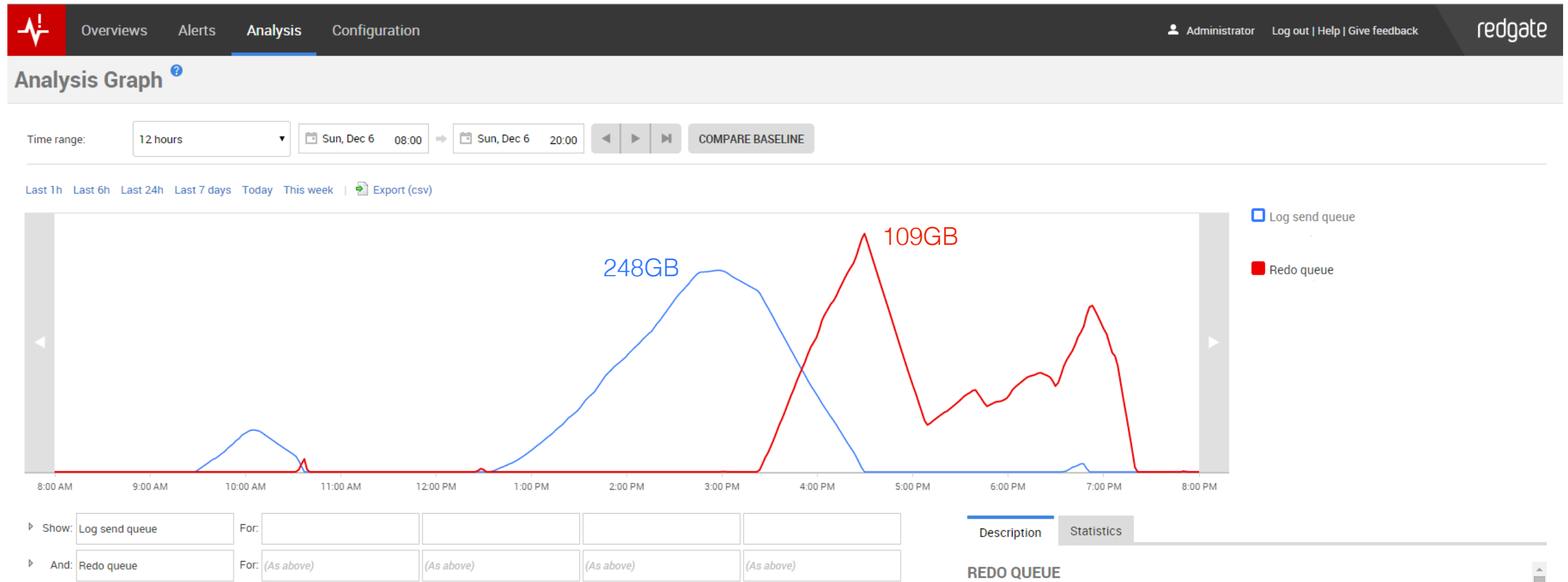


Demo

Demo



Real World Example



Dealing With Slow Replication Latency

- Identify your bottleneck and mitigate it
 - Minimize log generation
 - Use smart index maintenance
 - More bandwidth
 - Perhaps a dedicated network connection
 - Better hardware
 - Log throughput on secondaries needs to be equal to primary
- Upgrade SQL Server
 - 2012 single threaded redo - ~45MB/sec
 - 2016 multi-threaded redo - ~600MB/sec

Key Takeaways

- It is imperative to track and trend replication latency in your Availability Groups so you can answer the questions
 - How much data can I lose?
 - How long it will take to failover?
- Monitor and trend `send_queue` and `redo_queue` in `sys.dm_hadr_database_replica_states` on replicas to measure availability impact
- Understand how much log is generated in your databases
- Understand your system's operations, consider downtime for patching and network maintenance

Key Takeaways

- Plan database maintenance
- Use a smart index maintenance strategy!
- Offloaded backups
 - If availability is most important, backup on primary

Need more data?

<http://www.centinosystems.com/blog/talks/>

Links to resources

Demos

Presentation

aen@centinosystems.com



Free SQL Monitor!

Send me a tweet **@nocentino** and **@redgate #sqlsatchi**

What was that “one thing” you walked away with from today’s talk?

aen@centinosystems.com

Thank You!

**Thanks to the SQLSaturday Chicago Team!
Andy, Bill, Bob, Frank, Lowry and Wendy**



Questions?

Free SQL Monitor!

Send me a tweet **@nocentino** and **@redgate #sqlsatchi**

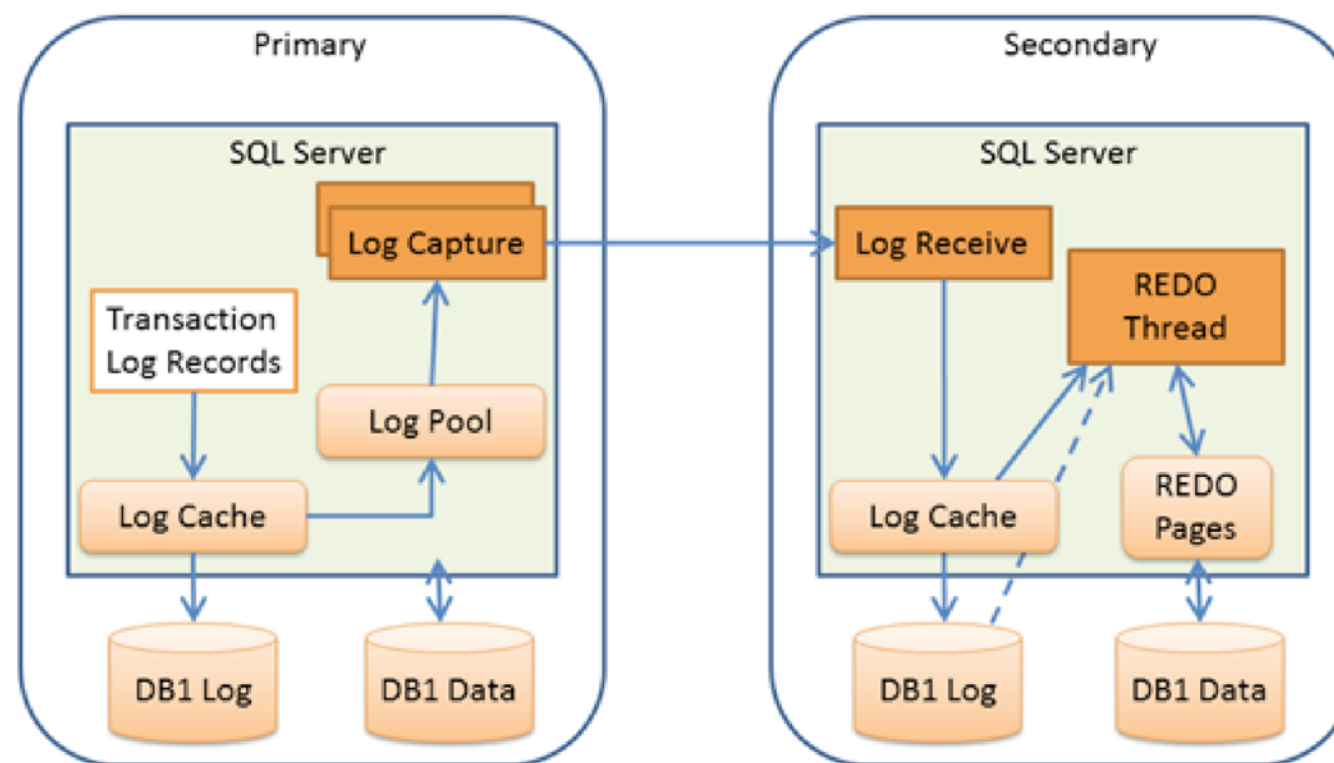
What was that “one thing” you walked away with from today’s talk?

aen@centinosystems.com

References

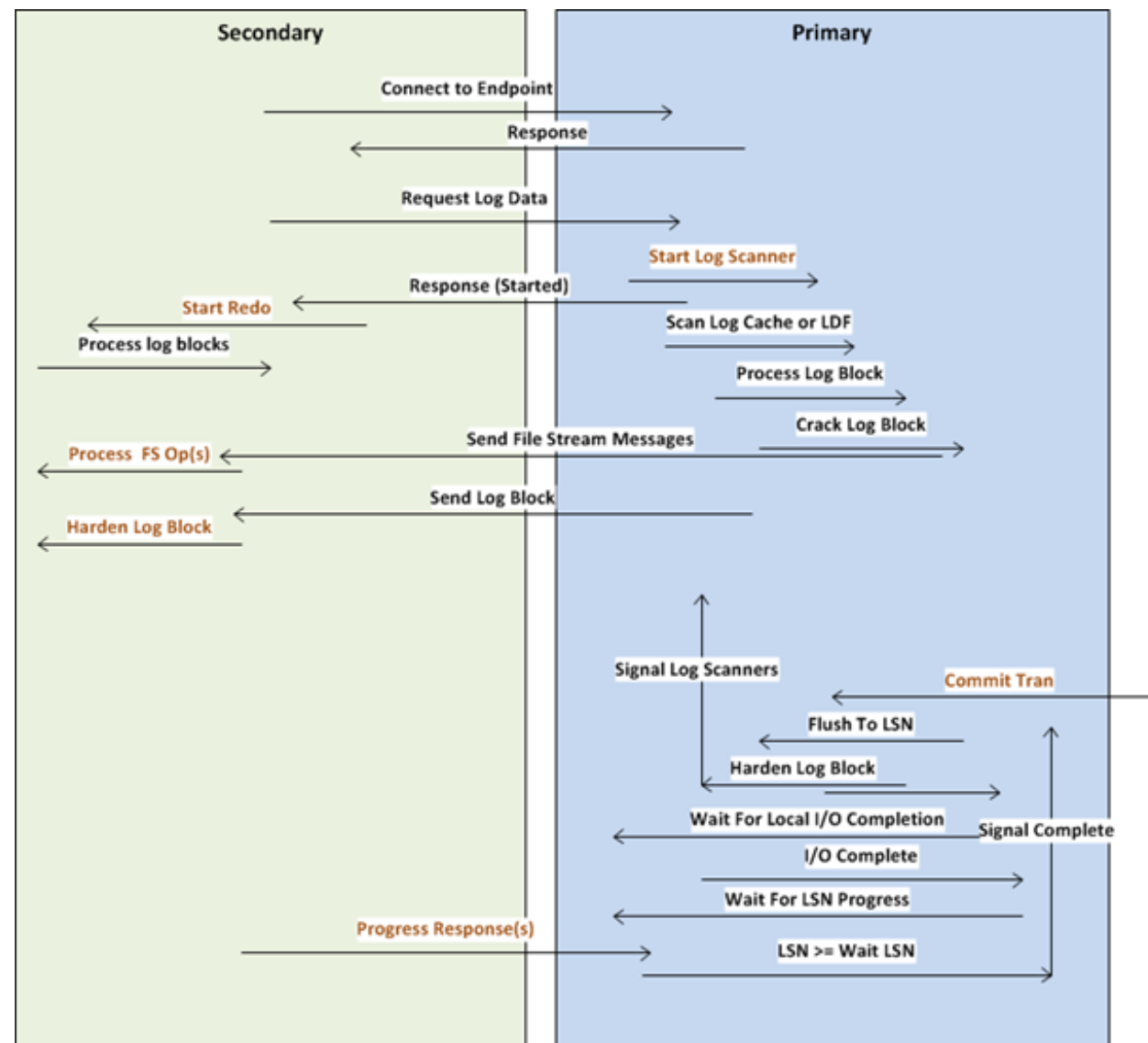
- <http://www.centinosystems.com/blog/sql/designing-for-offloaded-backups-in-alwayson-availability-groups/>
- <http://www.centinosystems.com/blog/sql/designing-for-offloaded-log-backups-in-alwayson-availability-groups-monitoring/>
- <http://www.centinosystems.com/blog/sql/monitoring-availability-groups-with-redgates-sql-monitor>
- <https://msdn.microsoft.com/en-us/library/ff878537.aspx>
- <https://msdn.microsoft.com/en-us/library/ff877972.aspx>

Data Movement in Availability Groups



From: SQLCAT's Guide to High Availability Disaster Recovery - <http://bit.ly/1U0vsss>

Synchronous Commit



From: CSS SQL Engineers Blog - <http://bit.ly/1P8SzNc>

Wait stats - sync vs. async

Synchronous - HADR_SYNC_COMMIT

```
sqldk.dll!XeSosPkg::wait_info::Publish+0x138
sqldk.dll!SOS_Scheduler::UpdateWaitTimeStats+0x2bc
sqldk.dll!SOS_Task::PostWait+0x9e
sqlmin.dll!EventInternal<SuspendQueueSLock>::Wait+0x1fb
sqlmin.dll!SequencedObject<LogBlockId,SequencedWaitInfo<LogBlockId>0>::WaitUntilSequenceAdvances+0x160
sqlmin.dll!HaDrCommitMgr::HardenNotifyInternal+0x1af
sqlmin.dll!HaDrCommitMgr::HardenNotify+0xac
sqlmin.dll!RecoveryUnit::NotifyHardenParticipants+0x1e2
sqlmin.dll!RecoveryUnit::HardenLog+0x217
sqlmin.dll!XdesRMFull::CommitInternal+0x6b8
sqlmin.dll!XactRM::SinglePhaseCommit+0x1a1
sqlmin.dll!XactRM::CommitInternal+0x472
...
```

Asynchronous - WRITELOG

```
sqldk.dll!XeSosPkg::wait_info::Publish+0x138
sqldk.dll!SOS_Scheduler::UpdateWaitTimeStats+0x2bc
sqldk.dll!SOS_Task::PostWait+0x9e
sqlmin.dll!SQLServerLogMgr::WaitLCFlush+0x219
sqlmin.dll!SQLServerLogMgr::LogFlush+0x29e
sqlmin.dll!SQLServerLogMgr::WaitLogWritten+0x17
sqlmin.dll!RecoveryUnit::HardenLog+0x25e
sqlmin.dll!XdesRMFull::CommitInternal+0x6b8
sqlmin.dll!XactRM::SinglePhaseCommit+0x1a1
sqlmin.dll!XactRM::CommitInternal+0x472
...
```

Availability Group Flow Control

- Used in response to network and system conditions
- Log blocks exchange sequence numbers
- The AG will enter flow control mode IF:
 - The **primary** detects too many unacknowledged messages, the primary stops sending messages
 - The **secondary** needs to tell the primary to back off, likely due to resource constraints, it will send a flow control message to the primary to back off
- Primary polls every 1000ms for a change in flow control state
- Secondary will message primary to leave flow control mode

From: SQL Server PFE Blog - <http://bit.ly/1ZpGyIL>