Administering and Optimizing Availability Groups Correctly

Victor Isakov
MCA, MCM, MCT, MVP

SQL Saturday #468, Sydney
27th February 2016

Microsoft CERTIFIED Architect
Microsoft CERTIFIED Master
Microsoft CERTIFIED Trainer
MVP Microsoft Most Valuable Professional

Housekeeping

Mobile Phones

please set to “stun” during sessions

Evaluations

complete online to be in the draw for fantastic prizes

SESSIONS	EVENT
http://www.sqlsaturday.com/468/sessions/sessionevaluation.aspx	http://www.sqlsaturday.com/468/eventeval.aspx

Wifi Details

Login: Password:

Victor Isakov

Microsoft CERTIFIED Architect | Microsoft CERTIFIED Master | Microsoft CERTIFIED Trainer | MVP Microsoft Most Valuable Professional

- Contact Victor Isakov if you want to:
  - Upgrade your SQL Server environment
  - Optimize Licensing Footprint
  - Substantially Improve performance
  - Provide In-House, Customised Training
  - Assess Your High-Availability and Disaster Recovery
- www.linkedin.com/in/VictorIsakov
- victor@sqlserversolutions.com.au

PASS SQL saturday

Abstract

Although Availability Groups are relatively easy to implement, there are many complexities when it comes to administering and optimizing them--especially if you are familiar with Failover Clustering. In some cases, your existing database solutions perform slower. In this session, we examine the various potential problems you can have when implementing Availability Groups and the potential solutions to these problems.

This session covers areas such as performance, administration, and security management. You learn how to manage complex jobs, backup strategies, and logins within an Availability Group environment. You also learn how Availability Groups might dramatically affect your application performance, how to recognise such performance problems, and how to resolve them. With a number of real-world examples, this practical session enables you to maximize the benefits of Availability Groups.

Prologue

- Customer

Prologue

- Solution Architect

Prologue

- Victor Isakov

“Famous” Quote

“For a solution to be a successful it must both deliver and be maintainable by the customer”

- Victor Isakov

- No-one in particular

- Sometime in 2015

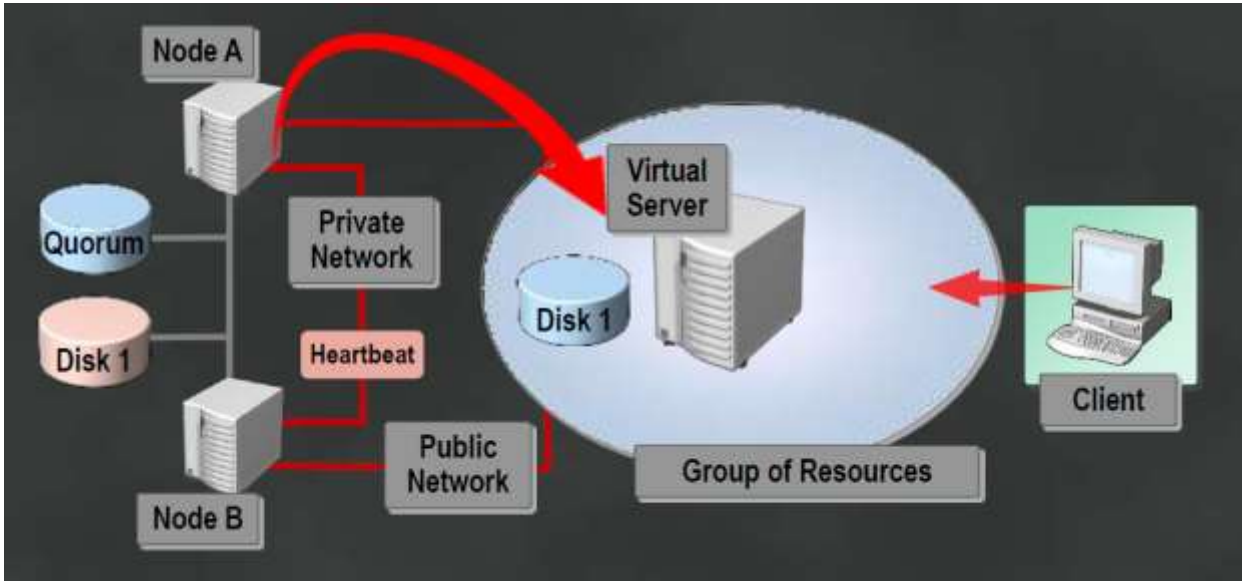
Agenda

- High Level Architecture
- Considerations
- Risks
- Configuration
- Performance
- Administration

ARCHITECTURE

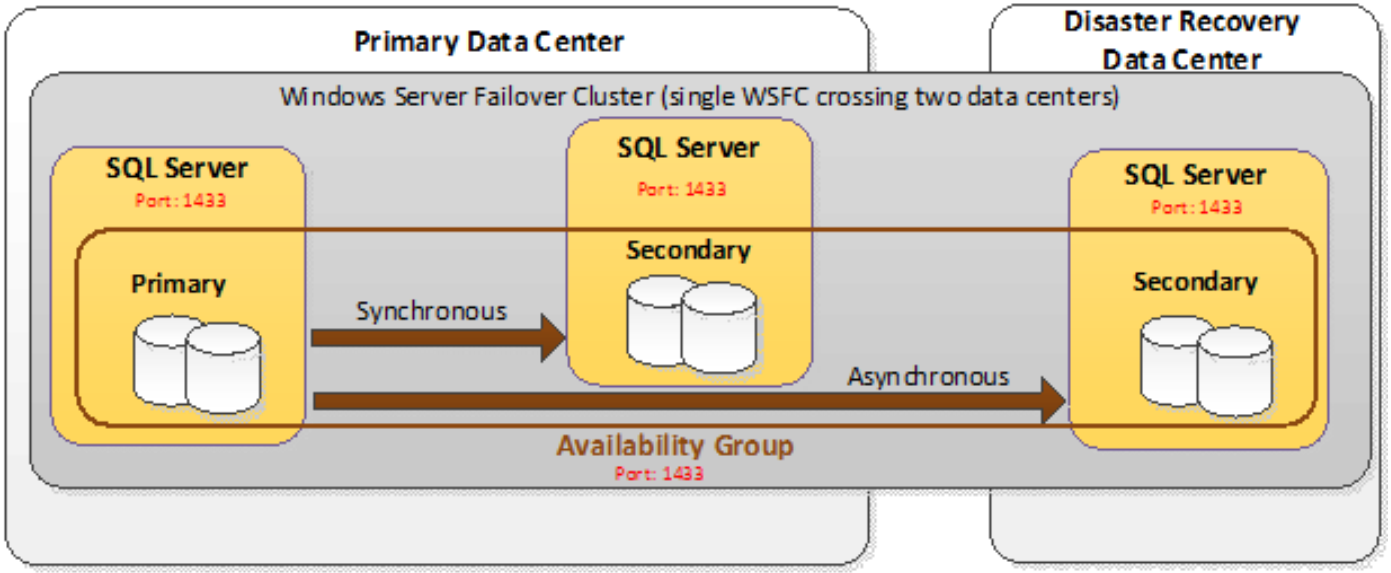
Failover Clustering

- Shared storage
- SQL Server instance only ever running one cluster node



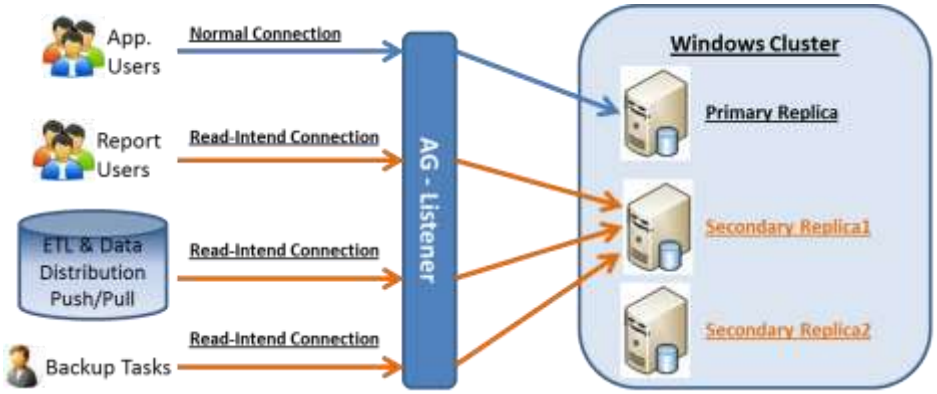
Availability Groups

- Separate storage
- Each SQL Server instance is running



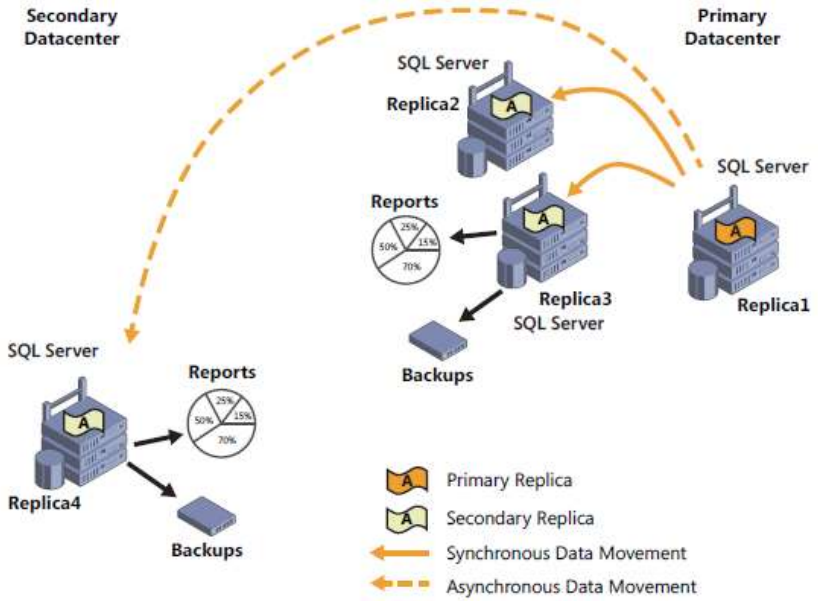
Listener

- Virtual Network Name (VNN)
  - IP address
  - Port
- Don’t forget an AG can have multiple Listeners!



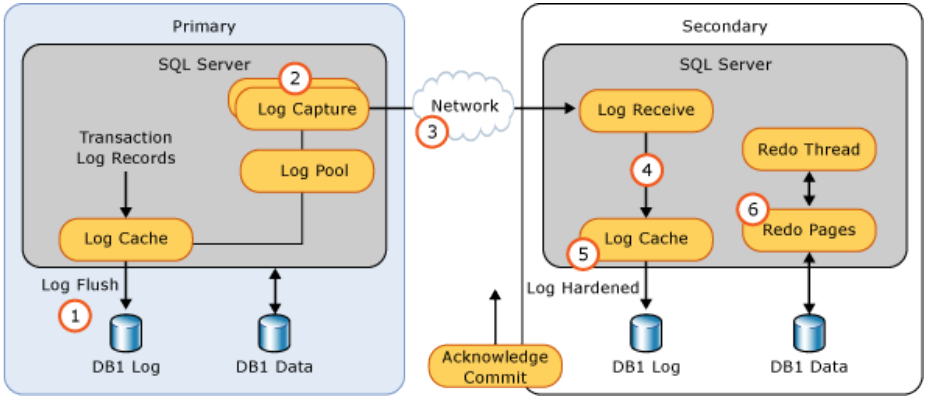
Availability Groups Use Case

- AGs “shine” where you have the ability to scale out the database solution(s)
  - Readable Secondaries
  - Off-load backups
  - Read-only Routing



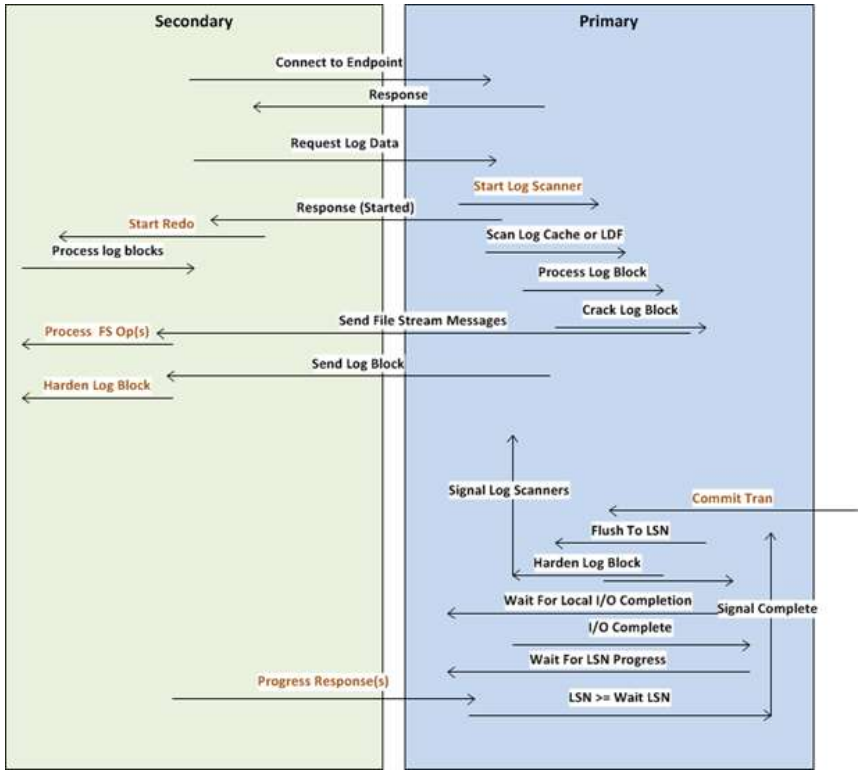
AG Data Synchronization Process

1. Log generation
2. Capture
3. Send
4. Receive and cache
5. Harden
   - HADR_LOGCAPTURE_SYNC wait
6. Redo



AG Data Synchronization Process

1. Connect
2. Request Data
3. Start Log Scanner
4. Redo
5. Progress Response(s)
6. **COMMIT TRAN**
7. Local Flush To LSN
8. Log Block Message
9. Progress Response
10. Commit Complete

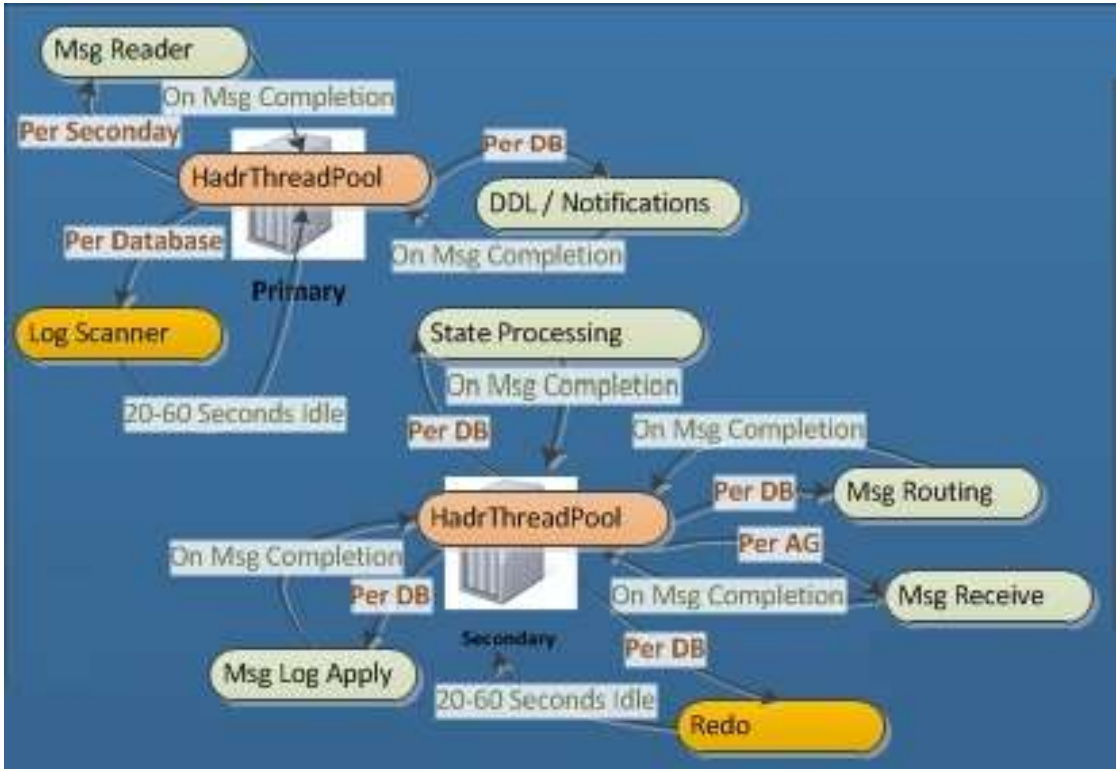


AG Thread Usage

- Database Mirroring (DBM) used dedicated threads per database
- Availability Groups use a request queue and worker pool to handle the requests
- The **HadrThreadPool** is shared among all AG enabled databases

AG Thread Usage

- Primary
- Secondary



AG Thread Usage

- Threads are shared on an on-demand basis, as follows:
  - Typically, there are 3–10 shared threads
    - This number can increase depending on the primary replica workload
  - If a given thread is idle for a while, it is released back into the general SQL Server thread pool
    - Normally, an inactive thread is released after ~15 seconds of inactivity
      - However, depending on the last activity, an idle thread might be retained longer
- In addition, availability groups use unshared threads, as follows:
  - Each primary replica uses 1 Log Capture thread for each primary database
  - In addition, it uses 1 Log Send thread for each secondary database
    - Log send threads are released after ~15 seconds of inactivity
  - Each secondary replica uses 1 redo thread for each secondary database
    - Redo threads are released after ~15 seconds of inactivity
  - A backup on a secondary replica holds a thread on the primary replica for the duration of the backup operation

AG Thread Usage

- Primary
  - On the primary messages the active log scanner is the log pole
  - When a secondary is ready to receive log blocks a message is sent to the primary to start the log scanning.
  - This message is handled by a worker in the **HadrThreadPool**
  - The startup and tearing down of a log scan operation can be expensive so the request will retain the worker thread, waiting on new log record flushes, until it has been idle for at least 20 seconds, usually 60 seconds before returning the message to the pool for reuse
  - All other messages acquire a worker, perform the operation and return the worker to the pool

AG Thread Usage

- Secondary
  - The expensive path on the secondary is the redo work
  - Similar to how the primary waits for idle log scan activity the secondary will wait for idle redo activity for at least 20 seconds before returning the worker to the pool

HadrThreadPool

- High Level Formula for AG Worker needs

Minimum Pool Size ~= (Maximum Active Databases) * 2
(Redo or Scan Per Database)
+ At Least 1 Message Handler

- The formula uses a 2x factor calculation
- For a database that is under heavy activity, backups frequently active and file stream activity a 5x factor would be max use case calculation at full utilization
- Database activity is key to the worker use and reuse
- Be sure to add the number of DB Replicas to your calculation

HadrThreadPool Configuration

- Default Timeout = 2000ms
- Default Max Workers = 200
- Workers to Leave = 40
- Min Workers = 2
- Capped at the sp_configure 'max worker threads' minus 40 level
  - To increase the size of the HadrThreadPool increase the max worker thread setting
    - Increasing the **max worker thread** setting can reduce the buffer pool size

CONSIDERATIONS

Availability

- Theoretically Availability Groups offer faster failover
  - Both the Primary Replica and the Failover Replica are running
- Definitely the case if you are using In-memory OLTP tables
  - With Failover Clustering the database engine will have to load all of the In-Memory OLTP data upon a failover incident
- You can only have one fail-over partner
  - SQL Server 2016 will address this limitation

Database Recovery Model

- With Availability Groups the database needs to be in FULL recovery model
  - This can potentially impact a number of areas
    - Storage space requirements
      - Important to capacity plan correctly
  - Backup strategy
  - Transaction log management
    - BACKUP LOG VictorDB TO DISK = ‘nul’

Application

- Will your application support Availability Group features?
- Appropriate network library support
- How will Availability Groups impact performance?

- DO NOT RELY ON HARDWARE SOLVING PERFORMANCE PROBLEMS

Design is Critical

- Good performance and ease of administration starts with good design
- Multiple Availability Groups
- Multiple Listeners
- Separate networks
- Storage considerations
  - Log files
- Hard to change some of these things without incurring substantial outage

Design is Critical

- Good performance and ease of administration starts with good design
- Multiple Availability Groups
- Multiple Listeners
- Separate networks
- Storage considerations
  - Log files
- Hard to change some of these things without incurring substantial outage

RISKS

Applications

- Good performance and ease of administration starts with good design
- Multiple Availability Groups
- Multiple Listeners
- Separate networks
- Storage considerations
  - Log files
- Hard to change some of these things without incurring substantial outage

Application Performance

- The sum of Availability Group's "moving parts" might contribute to a degradation in performance for your application
- Perform a Proof-of-Concept (POC)
- If you are also upgrading between releases
  - Test without Availability Groups
  - Test with Availability Groups

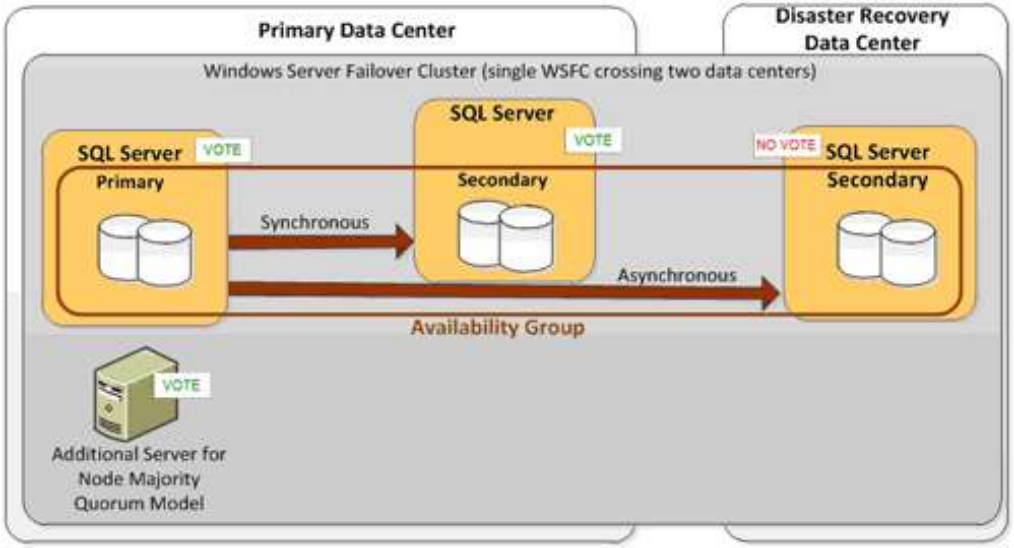
Organizational Maturity

- Availability Groups are a lot more difficult to administer / maintain
- SQL Agent Jobs
  - Backups
    - Full database backups
    - Log backups
- External Dependencies
- Is you DBA (team), assuming you have one, mature enough?
- Process. Process. Process.
  - Release Management
  - Change Management

CONFIGURATION

Quorum

- Ensure you have configured quorum correctly
  - Cluster
  - Availability Group



MultiSubnetFailover

- The **MultiSubnetFailover** connection property indicates that the application is being deployed in an availability group or Failover Cluster Instance, and that SQL Server Native Client will try to connect to the database on the primary SQL Server instance by trying to connect to all the IP addresses
  - Enables faster failover for all Availability Groups and failover cluster instance in SQL Server 2012 and will significantly reduce failover time for single and multi-subnet AlwaysOn topologies
  - During a multi-subnet failover, the client will attempt connections in parallel
  - During a subnet failover, SQL Server Native Client will aggressively retry the TCP connection
- Microsoft highly recommends to always use this

RegisterAllProvidersIP

- Determines whether or not the IP addresses of all the provider IP address resources are registered or not of the Client Access Point (CAP) in the Windows Server Failover Cluster (WSFC)
- Reduce re-connection time after a failover for clients whose client connection strings specify **MultiSubnetFailover = True**, as recommended

HostRecordTTL

- Controls how long DNS records are cached by client
- By default, clients cache cluster DNS records for 20 minutes
- By reducing **HostRecordTTL**, the Time to Live (TTL), for the cached record, legacy clients may reconnect more quickly
- However, reducing the **HostRecordTTL** setting may also result in increased traffic to the DN servers

PERFORMANCE

Most Important Take-Away

- **Do not assume that your database solution performance will be the same as it was prior to implementing the same solution on an Availability Group**
  - **ESPECIALLY IF YOU ARE UPGRADING BETWEEN VERSIONS**
- Perform a Proof-of-Concept (POC)
- If you are also upgrading between releases
  - Test without Availability Groups
  - Test with Availability Groups

Waits to Watch Out For

Wait	Description
HADR_SYNC_COMMIT	• Waiting on response from remote replica that the log block has been hardened. • *This does not mean the remote, redo has occurred* but instead that the log block as been successfully stored on stable media at the remote, replica. • Accumulation of HADR_SYNC_COMMIT wait time is the remote activity and you should look at the network and log flushing activities on the remote replica.
THREADPOOL	• Occurs when a task is waiting for a worker to run on. • This can indicate that the maximum worker setting is too low, or that batch executions are taking unusually long, thus reducing the number of workers available to satisfy other batches.
WRITELOG	• Waiting on local I/O to complete for the specified log block • Accumulation of WRITELOG wait time is the local log flushing and you should look at the local I/O path constraints.

Delayed Durability

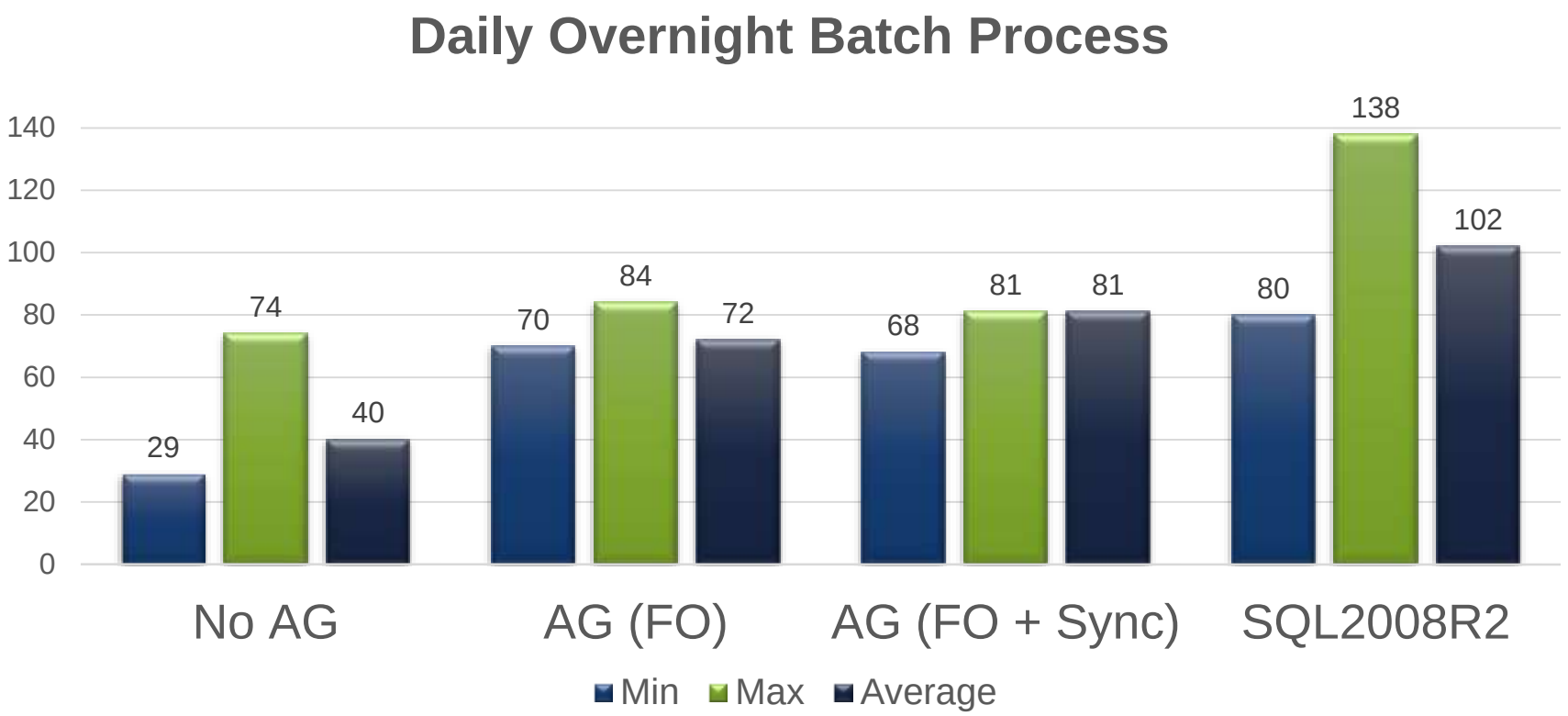
- Available since SQL Server 2014
- Allows client applications to continue without waiting for acknowledgement of COMMIT from transaction log
- Not hardening logs effectively before commit
- Potentially very useful for Availability Groups
- **BE AWARE OF IMPLICATIONS!**

CASE STUDY

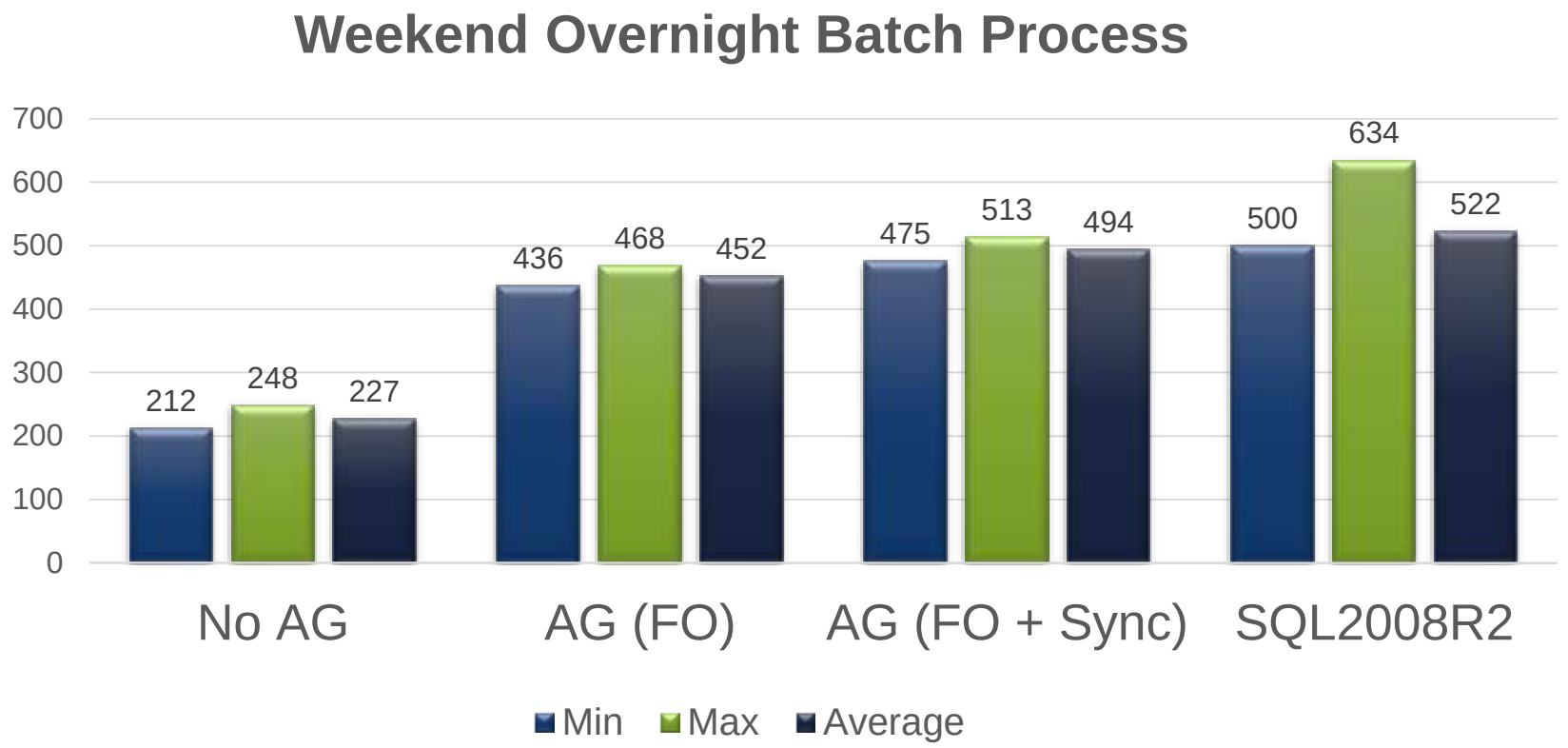
Financial Company

- CRM database solution
  - 1.5TB OLTP database
  - 6TB document store database
- Existing solution
  - SQL Server 2008 R2
  - Failover Cluster + disk-based SAN
- New solution
  - SQL Server 2014
  - Availability Groups + local PCIeSSDs
- Key batch process
  - End-of-Day
  - End-of-Week

Availability Group Overhead



Availability Group Overhead



ADMINISTRATION

Logins

- Logins are not automatically synchronized between SQL Server instances participating in an Availability Group
- Windows logins not so much a problem
  - Ensure they are created on all replicas
- SQL logins more of a problem
  - SIDs need to be the same
- Use **sp_change_users_login @Action='Report'** to detect orphaned users

Jobs

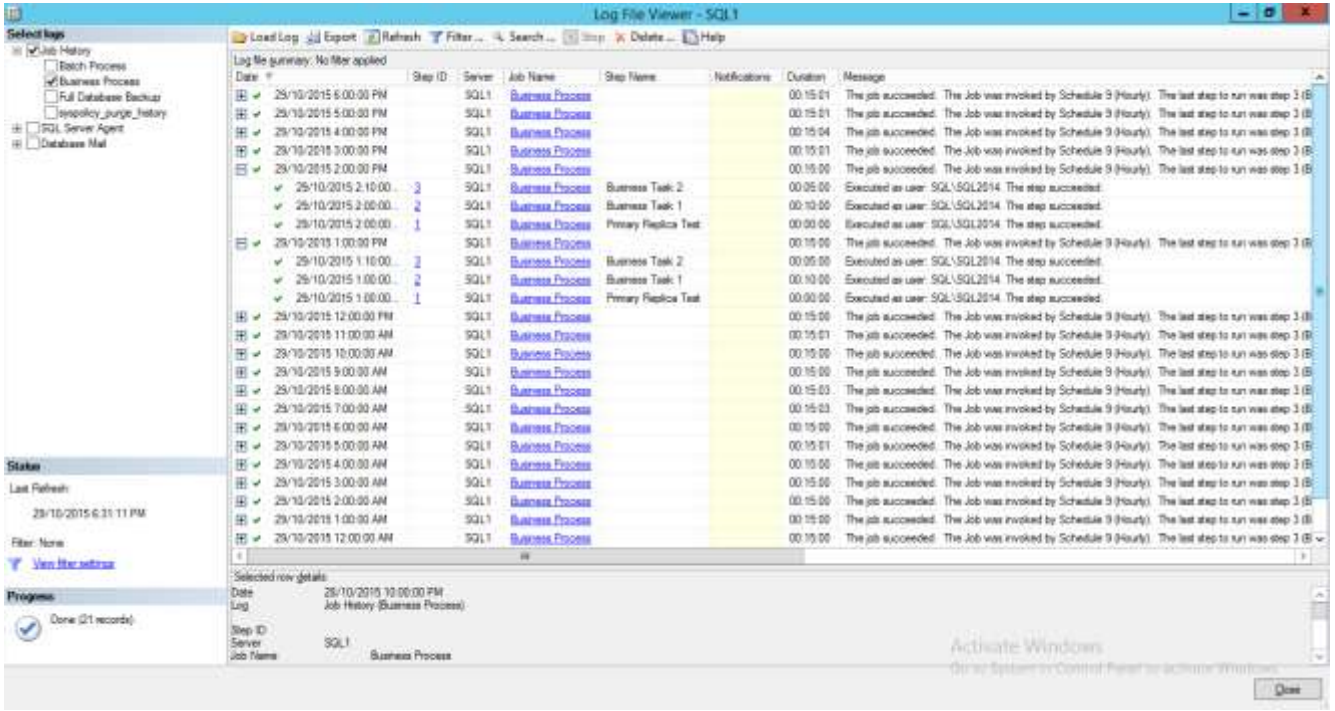
- Simply put, a complete and utter dog's breakfast
- What are your options?
- Quick search via Internet revealed
  - Use fn_hadr_group_is_primary in job step(s)
  - External Scheduler
  - Alert Driven Job Enablement
    - Error 1480
  - Use a separate Job Server
    - Master/Target servers

Jobs

- sys.fn_hadr_backup_is_preferred_replica
  - Used to determine if the current replica is the preferred backup replica
- fn_hadr_group_is_primary ('dbname')
  - Used to determine if the current replica is the primary replica
  - Introduced in SQL Server 2014
- sys.dm_hadr_availability_replica_states
  - Use for SQL Server 2012

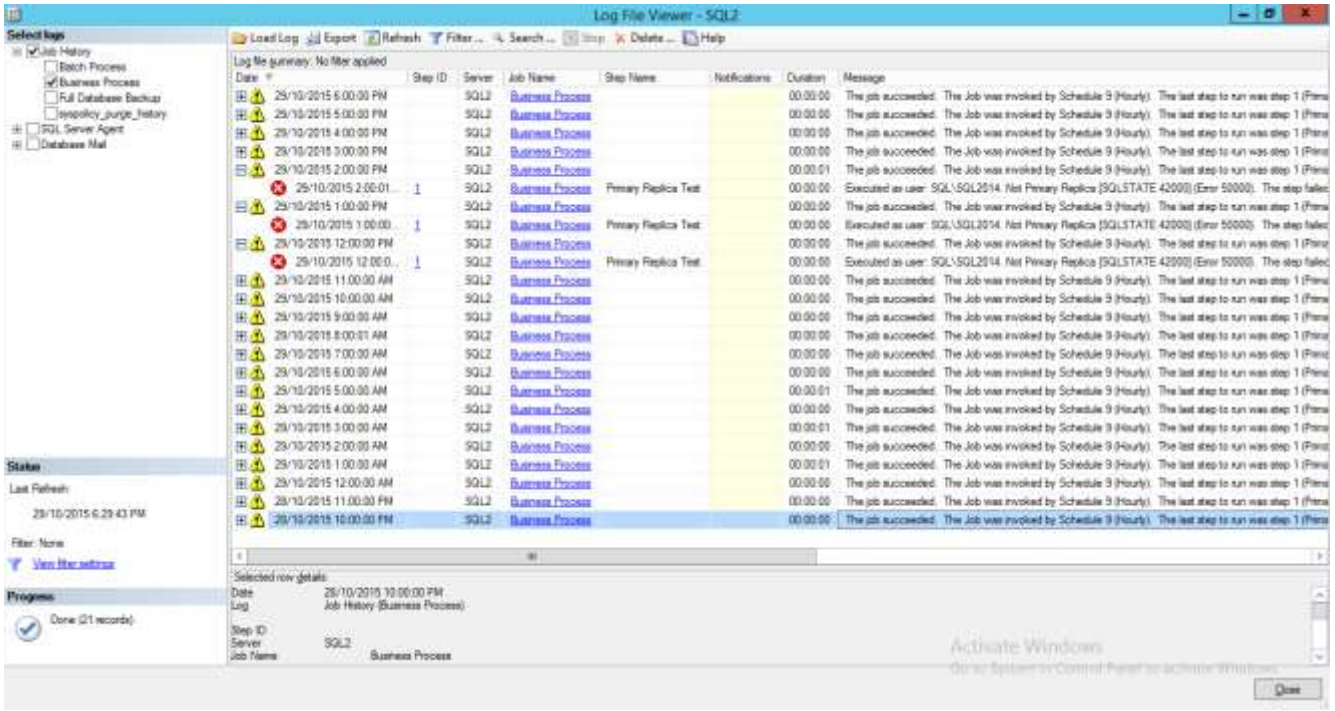
Job History

- Primary Replica



Job History

- Secondary Replica



SQL Server Configuration

- Make sure you have a process that ensures that any SQL Server configuration changes are applied to all Availability Group Replicas
  - Failover partners at least
- Use Policy Based Management (PBM)?

Availability Group Limitations / Issues

- Lots of limitations you will discuss in the future when it's "too late"
  - Cannot set database to READ_ONLY
  - Cannot move database files
  - Cannot detach a database
  - Cannot take database offline?
  - Much more difficult to restore
- Removing database from availability group

- Make sure you "practice" first in your non-PROD environment
- Ideally document processes before you need them

Operational Maturity

- It’s all about processes
  - Understanding technology helps ☺
- Release Management
- Change Management
- Perform Proof-of-Concepts (POCs)

Questions?

In partnership with PASS

Please make sure you visit our fantastic sponsors:

PASS SQL saturday