IDERA® | SQL Server Whitepaper

# WHY YOUR ALWAYSON SOLUTION **IS NOT ALWAYS ON**

BY ROBERT L DAVIS

How to avoid pitfalls that can cause unexpected failures with AlwaysOn technologies

## INTRODUCTION

Microsoft markets a variety of SQL Server availability technologies under the umbrella term AlwaysOn. AlwaysOn is a broad term covers many features including things as simple as online index rebuilds and automatic page repair. The two main AlwaysOn features are Availability Groups (AGs) and Failover Cluster Instances (FCIs). The name AlwaysOn, however, is somewhat misleading.

The term is a marketing term, and like many marketing terms, it is an overstatement of what you can actually expect from the group of technologies. These features will help you achieve a certain level of high availability, but 100% uptime is not a realistic expectation. Additionally, it is not simply up to the feature to keep you online. It requires an effort on your part to make it effective. Achieving high availability with Availability Groups or Failover Clustering requires proper planning, proper hardware, and a proper understanding of the technology.

This paper will delve into three issues that can cause your AlwaysOn solution to not be always on. The most important configuration decision you may make is the quorum mode. Much of the paper will be spent talking about quorum modes and how to choose the correct mode for your solution, including some non-standard options you may want to consider. Then we will look at problems that may arise from having Windows cluster nodes that are not members of an AG and potential issues from using multi-subnet listeners with applications that do not support multi-subnet failovers.

# QUORUM CONFIGURATION

Before we can dive into an advanced topic like troubleshooting locking, we need to define some terms to differentiate concepts that may not be well understood by a lot of database administrators (DBAs) and SQL developers. Locking, blocking, and deadlocks are related, to a degree, but they  are not interchangeable.

## Node Majority: voting is based on the number of active nodes

- A node must have a majority of votes to be online as primary
- Default setting is one vote per node
- A node must be online and accessible to vote affirmatively
- Recommended quorum mode for AGs or FCIs when there is an odd number of nodes

## Node & File Share Majority: voting is based on the number of active nodes plus one vote for a file share resource

- A node must have a majority of votes to be online as primary
- Default setting is one vote per node plus one vote for the file share
- A node must be online and accessible to vote affirmatively
- Connectivity by any node to the file share counts as an affirmative vote for the share
- File share resource should not be physically located on any node of the cluster because losing the server would remove two votes from quorum voting
- Recommended quorum mode for AGs (over disk related quorums) when there are an even number of nodes
- Effective quorum mode for FCIs as well when there are an even number of nodes

## Node & Disk Majority: voting is based on the number of active nodes plus one vote for a shared disk cluster resource

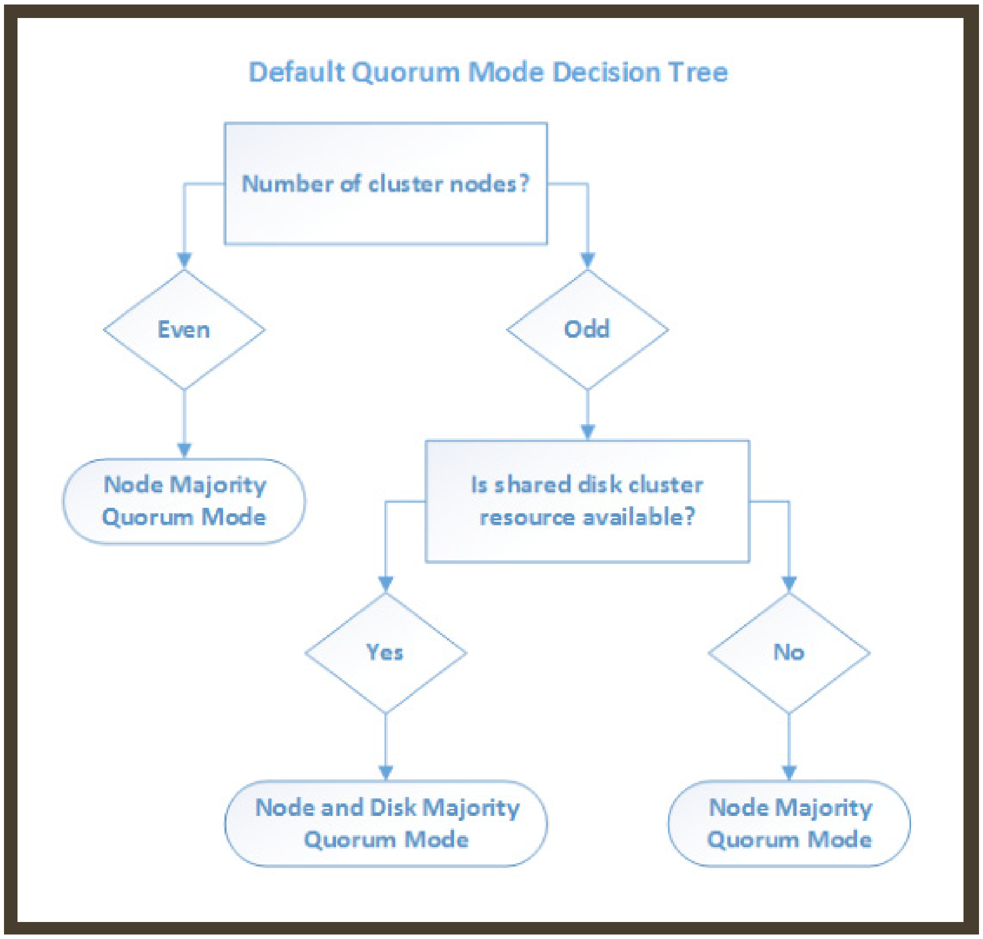
- A node must have a majority of votes to be online as primary
- Default setting is one vote per node plus one vote for the disk resource
- A node must be online and accessible to vote affirmatively
- Connectivity by any node to the disk resource counts as an affirmative vote for the disk
- The disk resource should not be assigned to a SQL Server cluster and should be able to fail over as a stand-alone resource
- When there are an even number of nodes, effective quorum mode for FCIs but not recommended for AGs

## Disk-only: no quorum required, active node is determined by the shared disk cluster resource

- The individual nodes do not vote and only the single vote of the disk-resource is required to be online as primary
- A node must have connectivity to the disk resource to gain an affirmative vote and be online
- If the single disk resource is offline, no node may be online as a primary (single-point-of-failure)
- Not an effective quorum mode as it is a single-point-of-failure. Exists only for backwards compatibility for legacy (upgraded) systems

When WSFC selects a quorum mode, it only considers the number of nodes and, the node count is even, whether or not a shared disk cluster resource is available. If the number of nodes is odd, it will always select simple node majority. If the number of nodes is even, and a shared disk cluster resource is available, it will select the node and disk majority quorum mode. If a shared disk cluster resource is not available, it will select simple node majority. If you wish to use node and file share majority, you must change the quorum mode from the default.

The logic used for the default quorum configuration can be expressed by the below decision tree:


Default Quorum Mode Decision Tree
Number of cluster nodes?
Even
Odd
Node Majority Quorum Mode
Is shared disk cluster resource available?
Yes
No
Node and Disk Majority Quorum Mode
Node Majority Quorum Mode

## NON-VOTING NODES

For most solutions, quorum seems simple and straightforward. Nearly 100% of cases, you simply  want to make sure that there are an odd number of votes and all other best practices are followed, such as ensuring that a file share witness is not physically located on a cluster node. However, another option that is available with AGs is to change the voting weight of each node to ensure that only certain nodes are allowed to vote.

As stated above, the default setting is for any node or voting member to count as a single vote.  The vote weight can be changed to zero to make a node a non-voting member. This is particularly useful when you have a node that has been designated for disaster recovery (DR) purposes only and under normal circumstances will never act as a primary node. This property of a cluster node is NodeWeight and can be configured via PowerShell, Cluster Administrator quorum wizard, or cluster.exe via the Windows command line (Windows Server 2008 R2 or earlier). Cluster.exe was deprecated in Windows Server 2008 and is no longer available. I recommend using PowerShell as the configuration tool of choice.

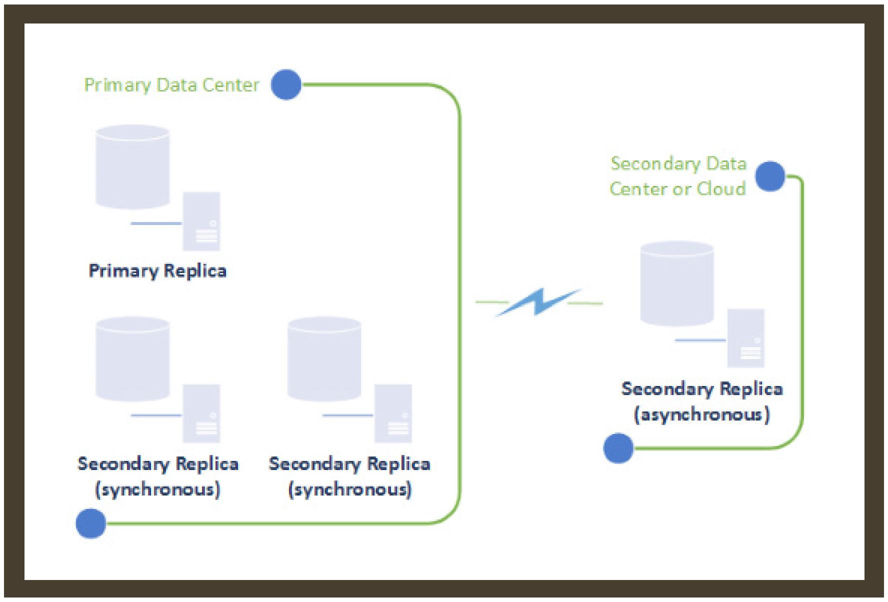
**Note:** Setting the NodeWeight property on Windows Server 2008 and Windows Server 2008 R2 requires installing hotfix KB2494036 on all cluster nodes. The hotfix adds the NodeWeight property and without it, NodeWeight will be reported as null or empty. The property already exists in Windows Server 2012 and newer versions.

When you have an even number of nodes, I do not recommend reducing the voting members by making one node a non-voting member. This ultimately reduces the resiliency of the system and the number of nodes failures that the system can withstand. If your only goal is to have an odd number of voting members, then you should add a file share resource or shared disk cluster resource to increase the voting members by one. Creating a non-voting member should only be done to accommodate a node that may be expected to be disconnected at times and should not be able to affect quorum of the other nodes.

For example, consider a Windows cluster (AG or FCI) with four nodes:

| Total Nodes | Quorum Configuration | Total Votes | Voting Nodes that Can Fail |
|---|---|---|---|
| 4 | Nothing | 4 | 1 (3 required for a majority vote) |
| 4 | Add a file share or disk witness | 5 | 2 (3 required for a majority vote) |
| 4 | Change 1 node to non-voting | 3 | 1 (2 required for a majority vote) |

A common scenario where a non-voting member may be desirable would be where multiple nodes are configured in a primary data center as synchronous failover replicas, and an additional node is configured in a remote data center or in a cloud provider as an asynchronous replica for DR purposes. The image below helps explain the node configuration for this particular scenario.


Primary Data Center
Primary Replica
Secondary Replica (synchronous)
Secondary Replica (synchronous)
Secondary Data Center or Cloud
Secondary Replica (asynchronous)

In this scenario, there is a primary and two synchronous secondary replicas in the primary data center and one asynchronous replica in a remote data center or cloud provider. This is a total of four nodes; however, the remote replica will only be brought online and used if there is a disaster and all replicas in the primary data center are offline. In this scenario, the remote replica could only be brought online through manual intervention to force quorum of the remote server.

As stated earlier, I recommend using PowerShell to view and configure the NodeWeight property. The Get-ClusterNode commandlet can be used to both view and set the NodeWeight property. The following command will connect to the local cluster return all cluster nodes with name, node weight, and current state.

```
Get-ClusterNode | Format-Table name, nodeweight, state –AutoSize
```

To set the node weight of a cluster node, you specify the node you want to change and set the node weight value, 0 for no vote and 1 for vote. For example, if I have a cluster node named DC2DRNode that I want to disable voting, the following command will set the NodeWeight property of that specific node to 0:

```
(Get-ClusterNode DC2DRNode).NodeWeight = 0
```

If you are configuring node weight remotely, you will need to be able to specify the name of the cluster as well as the name of the node you want changed. The following script will allow you to easily specify the cluster name and node name.

```
$Cluster = “ClusterName”
$Node = “NodeName”
(Get-ClusterNode –Cluster $Cluster –Name $Node).NodeWeight = 1
```

## DYNAMIC WEIGHTING

By default, dynamic configuration of quorum is enabled in Windows clustering, and this allows the cluster to manage quorum in a simple two-node cluster, the most commonly used scenario. Because a simple two-node cluster has an even number of nodes and the additional overhead of a file share or shared disk cluster resource may not be desirable, the cluster can set an additional property called DynamicWeight.

When a two-node cluster starts, it randomly selects one node to be the active node, and it sets the DynamicWeight of the inactive node to zero. If you view the NodeWeight property, you will see that both nodes have a weight of one, but if you further inspect the DynamicWeight property, you will see that one node is one and one node is zero. With a two-node cluster and no additional voting members, the only way you could possibly have a majority of notes is if both nodes are online at all times.

In this special case, the DynamicWeight will determine if the particular instance can be online or not. This is the only case where a single node may decide on its own, without any further votes, that it can be online. This is to prevent the active node from going offline every time it loses connection to the inactive node. This only works for a two-node cluster because if more than one node was allow to decide for itself that it could be online, you could easily end-up with a split-brain scenario where more than one node is active at the same time simply because they are disconnected from each other.

The DynamicWeight property is important to consider for a simple two-node cluster, but unlike NodeWeight, it is a read-only property that can only be set by the cluster. We can view this property by adding it to the PowerShell command we used earlier to see the NodeWeight:

```
Get-ClusterNode | Format-Table name, dynamicweight, nodeweight, state –AutoSize
```

The output from the above command would look like:

```
Name         DynamicWeight     NodeWeight      State
-----        -------------     ----------      -----
Node1                    1              1         Up
Node2                    0              1         Up
```

Note in the above output that the NodeWeight for both nodes is one, but the DynamicWeight has been modified so that Node2 is zero. With this configuration, if communication is lost to Node2, the DynamicWeight will ensure that that Node1 is able to stay online. If Node2 is still online, but it loses communication with Node1, Node2 will not come online. Again, a superior configuration would be to add a file share resource or shared disk cluster resource as a voting member so that wither node is able to negotiate whether it should be online or not.

## NON-AG MEMBER CLUSTER NODES

The predecessor to Availability Groups was database mirroring, and like database mirroring, AGs are also configured at the database level. Though not the most common layout for database mirroring, there were scenarios where mirroring was configured to have different groups of databases mirrored to different servers. This configuration is supported in AGs as well, but because of the additional component of Windows clustering, there is an added risk.

Consider a scenario where there are three servers in a Windows cluster for use with Availability Groups, and you wish to configure two AGs so that under normal operations, each active AG node has an active node and share a common failover node. With FCI, this is known as an N+1 configuration where you have N active nodes plus 1 common inactive failover node. Also like an N+1 configuration, we would be using multiple SQL Server instances for this scenario.

Because the Windows cluster spans three nodes but the AGs only span two nodes, you can end up in a situation where the cluster node that owns the cluster resource does not host one of the AG nodes. Normally, the cluster resource should be the same node as the AG node. However, if both nodes for an AG go offline (power outage, data center issue, reboot of both servers), the cluster resource may fail over to the only other available node which does not host an AG node.

If the cluster resource is owned by a resource that does not host the AG, the listener may appear to hang when it tries to bring the AG listener online. Due to configured dependencies, the database will not come online until the listener comes online. The cluster resource does not have an IP configured for the listener so the listener will fail to come online and stick in a resolving state. Attempts to manually bring the listener resource online will fail.

This apparent hang of listener resolution can be resolved by forcing the cluster resource to fail over to the node that hosts the AG, but the issue is not readily understandable and your AG may be offline for many minutes. The cluster may eventually fail over the cluster resource to another node, but the time this takes to happen has been inconsistent. I have seen it take as little as a few minutes to as long as 90 minutes to auto-resolve.

## MULTI-SUBNET FAILOVER

Another common scenario where you may experience unexpected outages with Availability Groups is when you have configured a multi-subnet failover with clients that are not multi-subnet compatible. When a listener uses multiple IP addresses for different subnets, all IP addresses are returned when querying DNS for the listener name. Older applications that do not support multi-subnet failovers may attempt to use the non-active IP address and rather than trying the other IP addresses, it will simply fail.

When you are using an application that is multi-subnet compatible, you should always include the "MultiSubNetFailover=True" parameter in the connection string. This will ensure that it tries every IP address until it connects to one.

If you must support older legacy applications that do not support multi-subnet failovers, you can adjust the RegisterAllProvidersIP property to disable registration of all listener IP addresses in DNS. If this property is disabled (set to zero), only the current active IP address will be registered in DNS. This may result in a longer failover time, from the client perspective, because the cluster has to deregister the current IP address and register the newly active IP address plus any DNS replication time, if any. These operations can easily take longer than the retry mechanism for multi-subnet failover.

This property is enabled by default and is easily modifiable with PowerShell with the Get-ClusterResource commandlet. As before, if you are using an older version of PowerShell, you may need to import the FailoverClusters module first in order to use this commandlet.

```
$Listener = “Listener Name”
Get-ClusterResource $Listener | Set-ClusterParameter RegisterAllProvidersIP 0
```

After executing the above command, the extra IP addresses will be deregistered from DNS and once the changes have propagated through DNS, the connection timeouts from applications that do not support multi-subnet failovers should stop.

## CONCLUSION

Quorum mode, Windows cluster nodes that do not host an AG  node, and multi-subnet listeners that must support clients that are not compatible with multi-subnet failovers are three common scenarios that many DBAs do not even consider when planning to deploy an AlwaysOn solution. These issues are generally not considered until problems arise from them. With careful planning and preparation, you can be proactive and prevent these issues from taking your solution offline.

The information contained in this paper will help you choose the proper quorum mode and configuration for just about any AG or FCI deployment. If choosing to configure an N+1 configuration or multi-subnet failover, you are aware of potential problems that can arise and how to prepare for and prevent them.

## DETAILS

**Writer** Robert L Davis

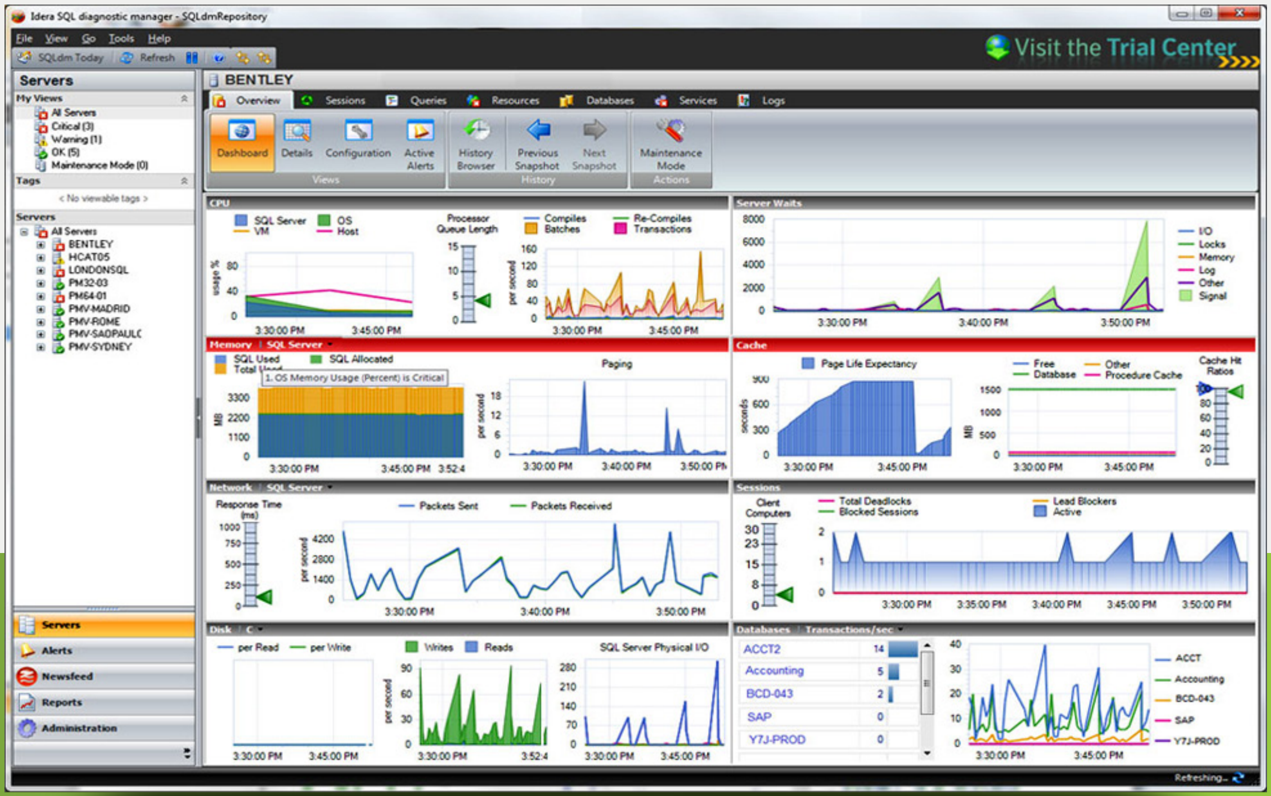
**Applies To** SQL Server 2012, SQL Server 2014, SQL Server 2016

# SQL DIAGNOSTIC MANAGER

## ACHIEVE 24/7 SQL MONITORING

- Performance monitoring for physical and virtual environments
- Query plan monitoring to see the causes of blocks and deadlocks
- Integrated SQL Doctor expert recommendations
- Easy integration with Microsoft SCOM
- Predictive alerting with settings to avoid false alerts
- Web-based dashboard with at-a-glance views of top issues and alerts

**Start for FREE**



IDERA

IDERA.com