


Realtime
publishers

The Shortcut Guide[™] To



Untangling the
Differences Between
High Availability and
Disaster Recovery

sponsored by

MARATHON
Run to Infinity

Richard Siddaway

Chapter 3: Configuring Your Environment for High Availability	38
Reliability and High Availability	38
Reliability Built In.....	38
High Availability Designed In	40
Monitoring the High-Availability Systems.....	40
Causes of Non-Availability.....	41
Bad Design	41
Single Points of Failure.....	42
Mistakes	42
Cost Cutting	43
High Availability in the Environment	44
Network.....	44
Servers.....	44
Storage.....	46
Applications	47
Infrastructure	48
Windows Clustering.....	48
Is Clustering the Answer?	49
High-Availability Clusters	50
Network Load Balancing Clusters	52
Geographic Clustering	52
Exchange	53
Replication for High Availability and Disaster Recovery	54
Are Backups Needed?	54
SQL Server	54
Mirroring.....	54
Log Shipping	55

The Challenge of Virtualization	55
Virtual High Availability	56
What Is Being Protected?	56
Combine with Other Methods	56
Combine with Disaster Recovery.....	57
The Link to Disaster Recovery.....	57
High Availability and Disaster Recovery Convergence.....	57
Knowing When to Stop.....	57
Summary: Cluster Round	58

Copyright Statement

© 2010 Realtime Publishers. All rights reserved. This site contains materials that have been created, developed, or commissioned by, and published with the permission of, Realtime Publishers (the “Materials”) and this site and any such Materials are protected by international copyright and trademark laws.

THE MATERIALS ARE PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. The Materials are subject to change without notice and do not represent a commitment on the part of Realtime Publishers its web site sponsors. In no event shall Realtime Publishers or its web site sponsors be held liable for technical or editorial errors or omissions contained in the Materials, including without limitation, for any direct, indirect, incidental, special, exemplary or consequential damages whatsoever resulting from the use of any information contained in the Materials.

The Materials (including but not limited to the text, images, audio, and/or video) may not be copied, reproduced, republished, uploaded, posted, transmitted, or distributed in any way, in whole or in part, except that one copy may be downloaded for your personal, non-commercial use on a single computer. In connection with such use, you may not modify or obscure any copyright or other proprietary notice.

The Materials may contain trademarks, services marks and logos that are the property of third parties. You are not permitted to use these trademarks, services marks or logos without prior written consent of such third parties.

Realtime Publishers and the Realtime Publishers logo are registered in the US Patent & Trademark Office. All other product or service names are the property of their respective owners.

If you have any questions about these terms, or if you would like information about licensing materials from Realtime Publishers, please contact us via e-mail at info@realtimepublishers.com.

[Editor's Note: This book was downloaded from Realtime Nexus—The Digital Library for IT Professionals. All leading technology books from Realtime Publishers can be found at <http://nexus.realtimepublishers.com>.]

Chapter 3: Configuring Your Environment for High Availability

The first two chapters looked at high availability and disaster recovery. Each topic was explained, with a discussion of how the two traditionally separate areas are converging due to advances in technology.

This chapter will return to the topic of high availability to consider how you can configure your environment for high availability. As we saw in Chapter 1, high availability is not simply a matter of implementing a particular technology. The environment has to be administered by people with the correct level of skills and knowledge. These administrators need to be working with the correct processes to ensure the systems remain available. No two environments will present completely identical issues; however, there are a number of themes that we can address that will aid you in creating a high-availability environment of your own. The starting point of your configuration is ensuring that your systems are reliable.

Reliability and High Availability

Reliability might seem to be the same as availability, but they are quite distinct concepts. Chapter 1 provides definitions of availability. Reliability can be considered to be the ability of a system to keep working under normal operating conditions. Some perspectives might extend this definition to cover abnormal conditions, but I think doing so confuses the discussion as to where to draw the line. With such a definition, we would quickly overlap into high availability and disaster recovery, so we will stick with normal conditions.

Systems that remain available are designed from the ground up with reliability in mind. Trying to bolt a high-availability solution onto a badly designed system is not going to work. Designing in high availability starts with ensuring they are reliable.

Reliability Built In

How do we build reliability into our systems? There isn't a single answer to this question. Chapter 1 highlighted the need to consider the hardware that you buy for your servers. It needs to provide resiliency by specifying redundant components so that as many single points of failure as possible will be eliminated.

Can you eliminate all single points of failure? The brutal answer is probably not. Figure 3.1 illustrates this point. Your knowledge of a particular system or situation can be divided into three main areas:

- The first area covers what you know. This is the area where you understand the system and its environment. There are no surprises and you have full control. In the ideal world, this area is as large as possible.
- The second area covers things you don't know. This could include the timetable of vendor upgrades or patches for your system. Another item to think about in this category is the growth patterns of the system usage. You don't know these considerations but you know that you need to find out. This area implies a level of risk to the system, but the risk is containable because you are aware of it and can plan mitigating actions.
- The third area, labeled "What we don't know we don't know," is the area that can really cause problems. This is where the surprises live—the issues that suddenly blow up and cause your systems to fail. They can be as simple as road works you were unaware of cutting a power supply or can be as subtle as a bug in the software causing a data corruption. This is the area you have to strive to minimize.

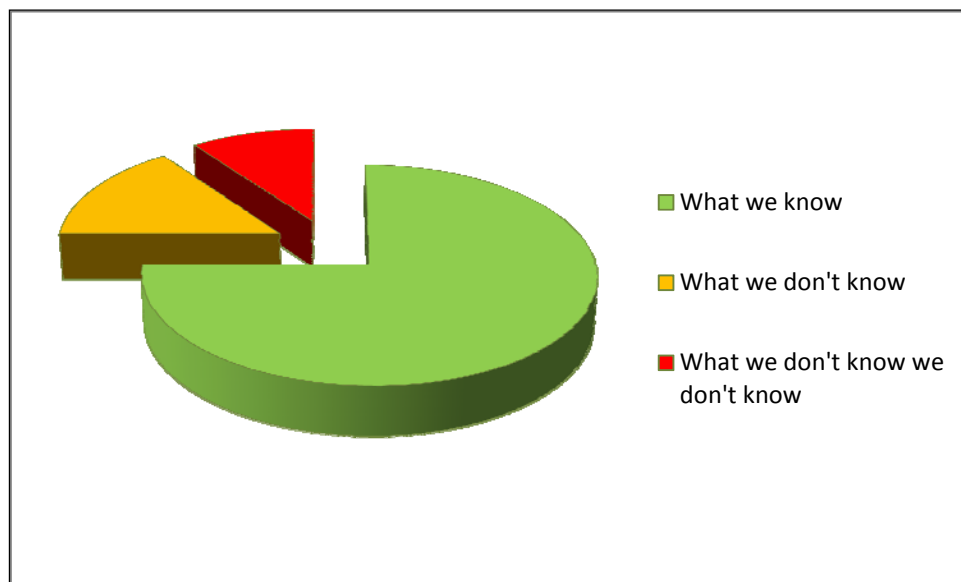


Figure 3.1: Areas of knowledge.

Reliability is not just a hardware issue. It is also a process issue. If the administration processes allow a junior admin to perform unscheduled changes during the working day, your system cannot be classed as reliable!

A correctly structured change control system is essential to increasing the reliability of your systems. You know what is happening. You know when it's happening, and you've examined the impact on your systems. You have thus diminished the area of unknowns in Figure 3.1. The next step, after designing your technology and processes to give you reliability is to design high availability into the systems.

High Availability Designed In

High availability is not an optional extra. It is either designed into your environment from the beginning or you don't have it. Chapter 1 looked at the areas of infrastructure you need to consider when designing for high availability:

- Network
- Applications
- Data
- Infrastructure
- Servers

Virtualization vendors can bring another aspect of high availability into your environment, as you will see later in the chapter. The ability for a virtual machine to failover from one host to another with no break in service is often claimed as the solution to your high-availability needs. This approach does increase the availability of your systems—at the server level; however, it does not, and cannot, create a high-availability environment all by itself.

Monitoring the High-Availability Systems

Systems, of any sort, just don't work of their own. They need to be monitored. Monitoring needs to start as the system is tested and commissioned. If you don't have historic data on the state of the system, how do you know it's changing?

Monitoring doesn't exist as a "tick box" activity. The results of monitoring need to be examined and changes made in response to any adverse trends or events. The changes could be required to provide extra resources (for example, CPU or memory), extra capacity (for example, another server in the Web farm or extra disk space), tuning of queries or indexes in a database.

So what should be monitored? The usual suspects are CPU usage, memory usage, and disk usage and capacity. To that list we need to add items such as:

- Network traffic
- Load on network devices
- Firewall logs
- Security patch levels
- Virtual host resource usage (if the virtual hosts become overloaded the performance of the guests will suffer and your users will start reporting that the system is unavailable)
- Database performance (for example, indexing, fragmentation, query speed, and so on)

The list continues but the essence is that for your system to remain available, it must be understood from end to end. In addition, the up-to-date state of that system must be known. It can't be assumed that the system is in good health just because the users are not complaining. If you know how your systems should be behaving and you know how they are behaving, you are well on the way to preventing problems rather than reacting to them.

We have considered how you can get to a high-availability situation, but what obstacles do you have to overcome? What stops your systems from being available?

Causes of Non-Availability

Chapter 2 looked at the types of disaster that could affect the availability of your systems. There are other more fundamental issues that could cause your systems to have a level of availability that is unacceptable.

Bad Design

A fundamental cause of systems not supplying the expected level of availability is that they were not designed correctly in first place. This point cannot be emphasized enough. It does not matter how many layers of process and expert administrators are put in place, if the system is designed incorrectly, it will not work correctly. I have examined many systems over the years in my capacity as a consultant. In a majority of cases, the underlying problems with those systems can be traced back to poor design.

Would you buy a house or use a bridge that was designed by someone who didn't know how to perform the task correctly? A quick look at any of the online forums will show that there are many people attempting to design, and implement, systems who don't really understand what they are doing. Before the design is started, ensure that the technologies that will be used are correctly understood. This doesn't mean a quick read through of a glossy pamphlet. It means an in-depth investigation of the technology. The technology should be understood to the point of being able to answer these two questions:

- What does the technology do?
- What doesn't it do?

The first can probably be answered easily by a quick review of a vendor's Web site. The second question is the one that you really need to answer. What are the flaws in the system? Which pieces of functionality don't quite work as advertised? Is there anything that doesn't work at all?

Once the technology is understood and it is decided that it is a suitable solution for the problem you have to solve, the next step is to decide how to implement it. This is where best practice comes into play. There is an established body of best practice for most, if not all, of the major components that will be implemented in your high-availability systems. Find it, read it, and apply it at the design stage. It may be impossible to retrofit the system once it is in production.

It is almost certain that the initial cost estimates of your design will result in an explosion along the lines of “How Much?” At this point, the design needs to be modified to reduce cost but keep as much availability as possible. There will always be a cost constraint, but that should not compromise the design to the point that you re-introduce single points of failure.

Single Points of Failure

One of the design steps for any system should be to scrutinize for single points of failure. If you are designing a high-availability system, you need to eliminate such points. If the system is designed to a less robust set of criteria, you need to understand where the single points of failure are so that they can be monitored. Plans also need to be created to deal with a failure. Think of it as a mini disaster recovery scenario and apply the techniques discussed in Chapter 2.

The design needs to be reviewed to ensure that single points of failure don’t creep into the system. The design team needs to perform this task initially, but it should also be reviewed by people who haven’t been involved in the design. If there is sufficient technical expertise within the organization, they can be utilized; otherwise, consider using external consultants. When implementing a business critical system, a few days consultancy may be a small price to pay compared with the costs and implications of a significant period of downtime due to a design flaw.

Whoever does the review, it must be performed with a critical attitude. The reviewer must be looking for problems, mistakes, risks, and any other negative connotations. If the reviewer, or organization, has been exposed to the De Bono Six Thinking Hats concepts, Black Hat thinking should be applied here.

Mistakes

Despite everything you do, mistakes happen. Any system involving people must by definition be vulnerable to human error. The object of your design, planning, and process creation is to minimize the impact of human error. Chapter 4 covers this in greater depth, but the points that you need to consider here are:

- Learn from mistakes—Discover what went wrong and change the process, methods, tools, or other items to prevent the mistake happening again
- Learn from others—If you hear of an issue affecting an organization with a similar configuration, make sure that your organization changes to prevent the same mistake
- Document events—If something goes wrong, make sure it, and the solution, is documented and the documentation is made available to the people who need it
- If there is an issue with a business critical system, make sure that a root-cause analysis is performed to identify the true cause of the issue and that the solution is implemented and documented

One of the most common mistakes is to impose cost cutting on a high-availability environment.

Cost Cutting

Cost cutting is a fact of life for many, if not most, organizations, especially when the economic climate is poor. The only ways an organization can be more profitable is to increase revenues and/or reduce costs. An obvious cost is that associated with providing high-availability environments. Figure 3.2 should be considered during any discussions of this nature.

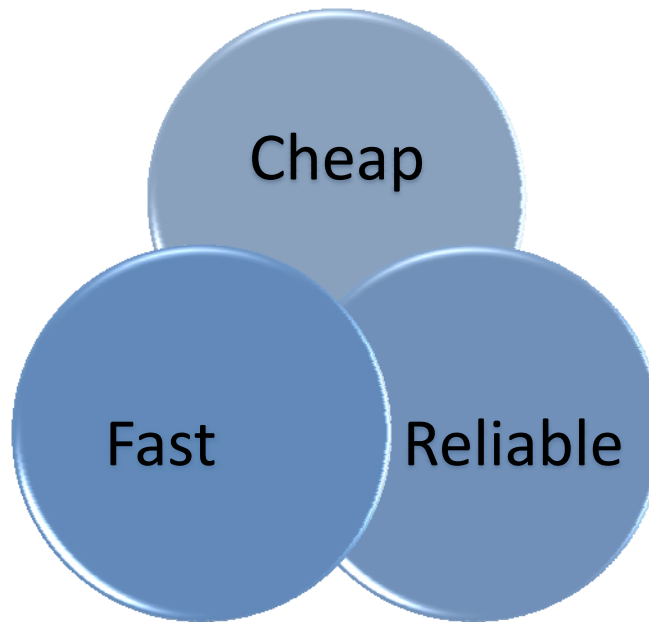


Figure 3.2: Cost, speed, and reliability relationship.

In many areas of IT, you can have two out of the three options in Figure 3.2: cheap, fast, and reliable. As an example, consider disk storage:

- RAID 0 is fast and cheap, but it's not the most reliable configuration
- RAID 10 is fast and reliable, but it's not cheap
- RAID 5 is reliable and relatively cheap, but it doesn't have the fastest write speed

Which storage configuration you need is obviously dictated by your application's needs, but cost, speed, and reliability all play a factor. This becomes even more complicated when you consider adding Fiber Channel, SATA, and iSCSI configurations into the mix. This comes back to the need to know and understand the technologies you are proposing to use.

After going through the design steps and working out what you need to do, the next step is to start implementing your high-availability solution.

High Availability in the Environment

We normally think about supplying high availability for servers, but as we have seen, you need to consider the whole environment. It is possible, if not probable, that you can't create an end-to-end high-availability environment. Such being the case, protect what you can now and start building the business case for next year's budget.

The environment can be thought of as a stack of sections that we need to protect individually and collectively. At the bottom of the stack is the network. If this stops working, everything else fails.

Network

The network is one of the areas of the environment that is only really considered when it goes wrong. A bit like a highway, we expect the network to be there and carrying traffic. When there is a problem, everything grinds to a very rapid halt.

Is your network resilient? If I walked into your organization and pulled the network cable out of the server or a switch, would it keep working? If the answer to these questions isn't a resounding yes, there is some work to be done on the network. If I can actually pull the cable, your physical security may need reviewing.

Some further questions that need answering regarding the network include:

- Does your network supply redundant routes to the servers?
- Is the link to the desktop redundant? It usually isn't and a failure there will take the application out of service as far as the users are concerned.
- If your servers have multiple network cards, are they teamed to ensure connectivity in the event of a failure?

Don't forget the links to the ISPs. Having multiple ISPs is a good thing. Do you have multiple independent links to those ISPs? If you have truly separate links to the ISPs, do they terminate in the same room?

Have another look at Figure 1.5 and think about all the different places that a single point of failure could be introduced into the environment. How many of them can you remove? If you can't remove them, do you have a plan for a rapid recovery in the event of a failure? Do you have spare, preconfigured equipment ready to slot into place if needed?

High availability needs a lot of planning and preparation. That extends to the server estate as well.

Servers

The server hardware is essential to your high-availability scenario, as it hosts the applications you need to drive your business processes. We have already discussed the servers themselves in terms of buying from recognized manufacturers and ensuring that as much resiliency as possible is built into the hardware.

One thing we haven't covered is the size of those servers. Until recently, there has been a tendency to implement one server per application, especially where the funding has come through projects. This has led to the server sprawl that is common in many organizations. The move of virtualization into mainstream use and other consolidation techniques means that many organizations are using a smaller number of larger servers. This has a good effect as it reduces power consumption and air conditioning requirements, but it can have a downside from a high-availability viewpoint.

The phrase "putting all of your eggs in one basket" comes to mind. When you have many servers, each of which hosts a single application, the impact of a single server failing is minimized. Now you have multiple applications on a single server, or even worse, a single database server supporting multiple applications. The loss of a server like that has a much bigger impact on the business.

Consolidation is a good thing, especially from the perspective of whoever holds the purse strings, but some of that savings has to be used to fund ensuring the high availability of those bigger servers. We will examine how you can provide high availability for your virtualization environment later in the chapter.

One final point on virtualization and consolidation regards the administrative effort required in the environment. Assume we start with 100 physical servers in our environment. Our analysis shows that we need five physical hosts to support the virtual machines we will create. Once we have finished the migration to the virtual environment, we have at least 105 servers to support (100 virtual and five hosts). We might have a few more if we use physical management servers. This can put additional burdens on your administrators and may turn them into an area of potential failure if they don't know the technology or don't have the capacity to absorb the extra work.

An organization with a significant number of servers will have them rack mounted in a data center. The data center needs power to keep everything working. Questions to ask regarding power:

- Does your environment have a guaranteed power supply?
- If you lose the power supply, is there a backup in the form of a generator?
- When was the failover to the generator last tested?
- How long does it take?
- Do you need to protect the servers with a UPS?

Power is essential to a computing environment. If it isn't available, the systems will stop. The other threat from the power supply is "brown outs" and spikes (that is, dips and peaks in the supply). They can cause a lot of damage to your servers. How are you going to protect them? Does your UPS protect from these issues?

Computers produce heat. Lots of computers in a data center produce lots of heat. High temperatures can cause server to fail. A significant fraction of the energy budget for a data center is spent dealing with that heat. Modern, recently built data centers are tending towards water- or air-based cooling methods rather than air conditioning. Most of our data centers don't have these facilities, so we rely on air conditioning units to remove the heat our computers produce.

Does your data center have redundancy in its provisioning of air conditioning? If you have dual units, can one of them supply all the cooling needs? If there are more than two, what is the minimum number required to cool the data center?

A last thought on servers concerns drivers. Microsoft continually researches the reasons for Windows systems failing. One of the biggest causes is bad drivers. Make sure that the drivers you use on servers are correct. Manufacturers will update drivers periodically. How do you test those drivers before bringing them into production? One of the big problems that can cause downtime is the storage drivers used in the servers. The interface between server and storage seems to be a particularly vulnerable one. What can you do to protect this linkage?

Storage

In my experience, storage drivers have been a particular vulnerability. In this category, I also include the firmware upgrades that are often necessary for storage systems. If these aren't kept up to date, there is the possibility that your system will become vulnerable to a problem. In addition, the vendor may reduce, or even remove, the level of support they are prepared to offer. However, the upgrade may be invasive and as it could affect the whole environment, may be viewed as an unacceptable risk.

This is an issue that all organizations have to face. It can, and will, cause problems. Before purchasing any storage, find out how it will be upgraded and try to get a handle on the frequency of the upgrades. Luckily, they are not as frequent as operating system (OS) patches.

Disk configurations can have a serious impact on availability. This is not the place for a full discussion of possible configurations; it is assumed that the general RAID configurations are understood, but some general points can be made:

- RAID 0 will not contribute any resiliency to your system. If any disk in the array is lost, it implies that all the data on your volume is lost. This is a bad thing to happen. RAID 0 should not even be used for temporary storage such as a SQL Server tempdb, as a failure will bring the system down.
- RAID 1 is fast and secure. RAID 10 gives better performance but at a much increased cost point.
- RAID 5 provides good read performance, relatively poor write performance, and reasonable resiliency.

Don't forget the controllers when considering disks. If you only have a single controller and it fails, you have lost access to your data and the system is unavailable. It provides another example of needing to think through all the possibilities to remove single points of failure. The need to supply redundant hardware like this also increases the price of the system!

While considering storage systems, does yours allow for a hot spare so that a disk is automatically configured to join the array if one fails? Do you use that facility? Many systems allow for the "hot swapping" of disks (that is, if a disk fails, it can be removed and a replacement one substituted). If you have this facility, will you use it in the event of a failure? Have you tested it to ensure that your systems aren't adversely affected?

The last point I want to consider on storage is concerned with performance. The configuration and type of disks used will affect performance. As an example, creating a RAID 5 array of SATA disks as the target of a write-intensive application will probably not give you the performance you require. Your users will complain and the system will be judged as unavailable.

Applications

This is where you reach the core of your high-availability system. If you remember back to Figure 1.2, the application is what directly supports the business process. Everything else exists to support the application. Your applications can cause loss of availability for a number of reasons:

- Badly installed or configured
- A software problem (bug) that causes failure, or even worse, data corruption
- A patch or upgrade that goes wrong, doesn't work, or has a bug
- Over consumption of resources that leads to processing bottlenecks at the hardware level.

One consideration that is often overlooked regarding applications is the suitability of a particular application for the high-availability solution you are proposing. There is no point in planning to use a cluster if the application isn't cluster aware. You won't get the full benefits of the cluster and in some circumstances may have less availability compared with installing on a single server.

Who is writing and supporting the applications? How are the applications tested before being put into production?

Changes to applications in the production environment must be under change control. A change to an application can go wrong and be just as damaging as any other failure. Ensure the change approval board has infrastructure and application representatives to guarantee all points are covered.

Infrastructure

Infrastructure covers all the other bits you need to support the high-availability system. Things like:

- Backup system—Is it aware of the application and does it produce a backup that can be used to restore the application? Has the restore process been tested, proven, and documented?
- Anti-malware system—How does that need to be configured for the application? Are there files that shouldn't be scanned or only scanned when the application is shut down? How are updates to the anti-malware software applied? Can they cause a system failure?
- Basic infrastructure services such as Active Directory (AD) and DNS—What is required for the application and how does it interact with these services? What do clients need? Will the application require extra domain controllers in the environment due to the load it imposes?

Now that you have thought about some of the issues involved in configuring high availability, you need to look at the technologies that directly supply high availability starting with the traditional approach of failover clustering.

Windows Clustering

Clustering for Windows systems has gone through a number of iterations, and name changes, since its original introduction as “Wolfpack” with Windows NT. A cluster of Windows Server 2008 R2 machines can now include as many as 16 nodes, allowing for an impressive amount of computing power.

Setting up a Windows cluster has an unjustified reputation as being a difficult task. If the instructions available from the Microsoft Web site for your particular version are followed in the correct sequence, setup is a straightforward activity. The problems come when people try to take shortcuts or adopt the “It’s just Windows, how difficult can it be?” attitude. This is when things go wrong and the environment that was supposed to incorporate high availability suddenly doesn’t. In fact it is usually the opposite!

The creation of a Windows cluster has become easier over time culminating in the command-line setup available via the Windows PowerShell cmdlets in Windows Server 2008 R2.

Windows PowerShell

PowerShell is the automation engine and scripting language Microsoft is building into all its major products. Version 1 was released in November 2006 with version 2 becoming available with the introduction of Windows 7/Windows Server 2008 R2.

As well as the basic functionality, PowerShell introduces command-line, or script-based, management for a number of features in Windows Server 2008 R2, including:

- AD and GPOs
- IIS
- Failover clusters and NLB
- Remote Desktop Services
- Best practice analysis and troubleshooting

PowerShell is also available in Exchange 2007/2010, SQL Server 2008, SharePoint 2010, and members of the System Center family of products. A number of third-party vendors such as Quest, Citrix, and VMware are also adopting PowerShell.

Failover clustering may be the first-choice answer when contemplating high availability, but it isn't suitable for all situations.

Is Clustering the Answer?

There are a number of situations where failover clustering will not supply the levels of high availability you require. For instance:

- Some applications are unsuitable for installing on a cluster. It may be possible to install the application, but if it isn't cluster aware, it won't automatically failover. This negates the whole point of using failover clustering.
- Functionality, such as being a domain controller, cannot fail over between physical nodes; it is tied to the physical computer rather than the cluster.
- There are applications, such as Exchange 2010, that cannot be installed on to a failover cluster. They have other methods of high availability.
- You need multiple instances of the application (for example, a Web farm or terminal services farm) and an approach such as network load balancing is a better approach.

This still leaves a high percentage of scenarios in which you can use clustering.

High-Availability Clusters

The following discussion assumes that the native Windows clustering technologies are being used. There are third-party solutions that enhance and/or replace Windows clustering.

You need to make a number of decisions regarding your cluster configuration. The first decision point involves the witness/quorum functionality; in other words, how you want your cluster to be controlled in terms of which physical node controls which resource in the cluster. In earlier versions of Windows, a disk would be used to provide this functionality—known as the quorum disk. One slight problem is that the quorum then becomes a single point of failure in the cluster!

The latest versions of Windows alleviate this problem by using a “majority node set” configuration in which the nodes effectively vote for which of them has control; however, this requires a higher number of nodes to be online to be effective. Alternatively, it is possible to use a hybrid approach in which the two options are combined to present the maximum availability of the cluster as a whole.

Traditionally, clusters have been created with pairs of computers (see Figure 3.3). Each cluster will have an active node and a passive (spare) node. The active node provides the service, and the passive node waits until there is an issue with the active node that causes the application to fail over.

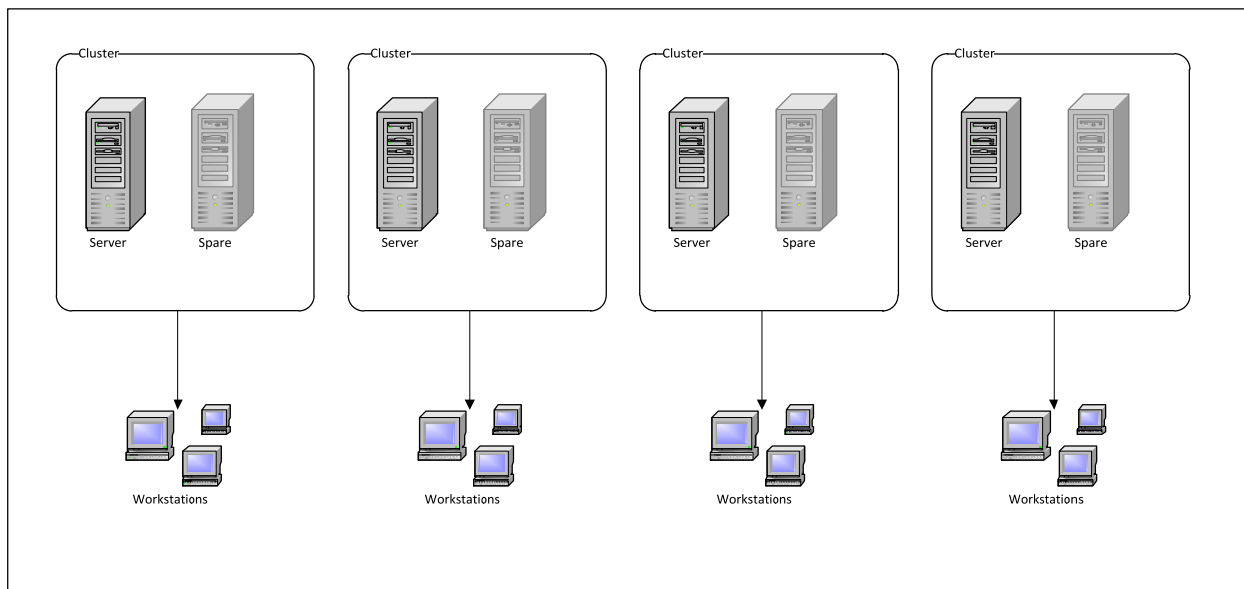


Figure 3.3: An environment with multiple two-node clusters.

This is an expensive use of resources in that 50% of the cluster resources are effectively idle. Many organizations will attempt to overcome this by configuring the cluster as active-active (that is, both nodes supply a service and the two services are combined on a single node in the event of a failure). This is commonly seen with database systems. The drawback to this approach is that the surviving node may be overwhelmed by the load and unable to cope. The applications become unresponsive, and the users report the system as unavailable at which point the statement will be made “You spent on all of this money on high availability and it doesn’t work.”

The other option is to combine the clusters and increase the ration of active to passive nodes (see Figure 3.4). This may also be driven by applications such as Exchange 2007 that cannot be installed in an active-active configuration. Where you have a set of similar workloads, for instance, Exchange systems, database systems, or even simple file and print, you can combine them into an M+N cluster. In this configuration, there are M active nodes and N passive nodes. Figure 3.4 shows the four clusters from Figure 3.3 re-arranged as a single cluster with six active nodes and two passive nodes. This setup has decreased the passive resources by 50%, making better use of the computing power in the environment.

If this style of cluster is to be adopted, it is essential that the applications installed onto the cluster and the workloads produced are compatible. Attempting to create a cluster where some nodes ran Exchange and others ran SQL Server, for instance, would not be a good idea.

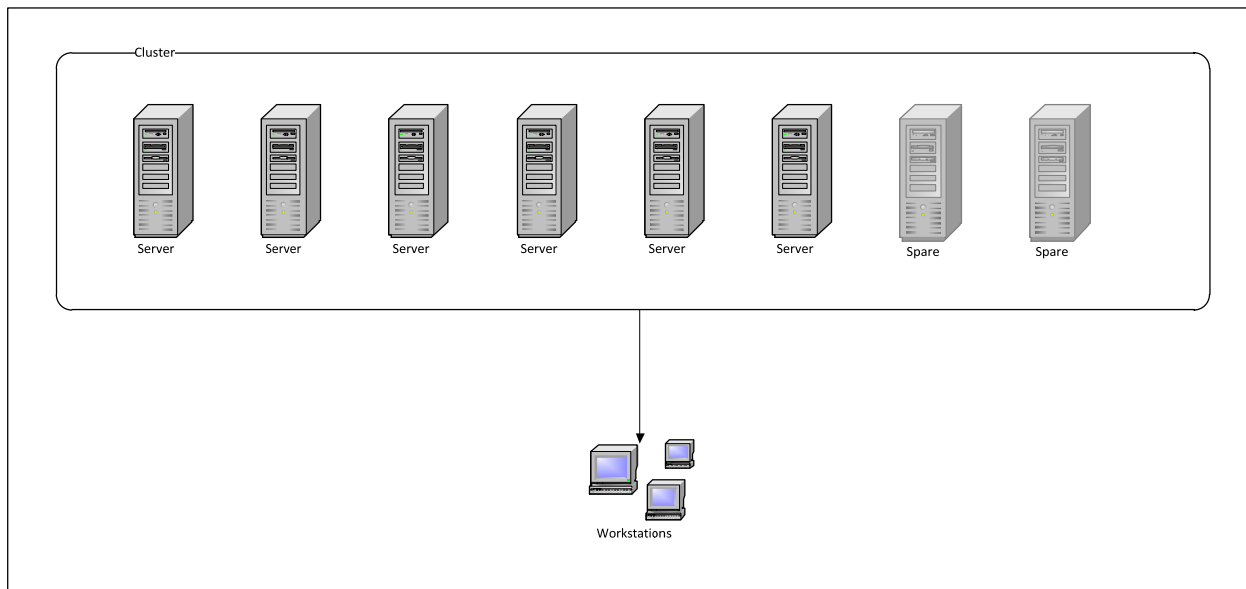


Figure 3.4: M+N-style cluster with multiple active and passive nodes.

The combination of the clusters should reduce the administration overhead as there is only a single cluster to manage. An environment of this type would require very skilled and experienced administrators. It is not the type of environment to aim for as your first attempt at clustering.

Failover clustering provides protection to the application from hardware problems by enabling it to failover (move) to another physical node. There are times you need to be able to increase the capacity of your application as well as provide high availability. In this case, it may be possible to use network load balancing.

Network Load Balancing Clusters

Network load balancing clusters enable an application to be installed on a number of servers and allow for the users to see that cluster as a single entity. Imagine Figure 3.4 with eight active servers. The users would see a single machine that supplied the application. In reality, the load would be balanced across the servers in a manner that is transparent to the users. In the event of a node failing, the remaining nodes would distribute the traffic.

This sounds like a really good idea that could solve many problems, however, there are some limitations. The applications have to be TCP/IP based and capable of working in this manner. This effectively limits the use of this technology to Web-based (HTTP/HTTPS) applications or a situation such as a terminal services farm where a broker service can distribute the workload between the nodes. SQL Server can be used in this way if the database is read only.

Network load balancing needs careful configuration of the network elements; otherwise, the switches providing connectivity can be overwhelmed. There are alternatives to using network load balancing such as DNS Round Robin (which is not recommended) or hardware-based load balancers. Interestingly, the latest versions of Office Communication Server (OCS) does not support network load balancing.

There are some applications that have their own high-availability options that resemble network load balancing. These are specialized solutions that are beyond the scope of this discussion.

One problem with cluster solutions is that they are tied to a single data center. If the data center is lost, or even if just connectivity is lost, the application becomes unavailable. This problem can be resolved using geographic clustering.

Geographic Clustering

Geographic, or stretch, clustering is illustrated in Figure 3.5. One or more clusters are created in separate data centers. In the event of a failure, the application fails over to the node in the other data center.

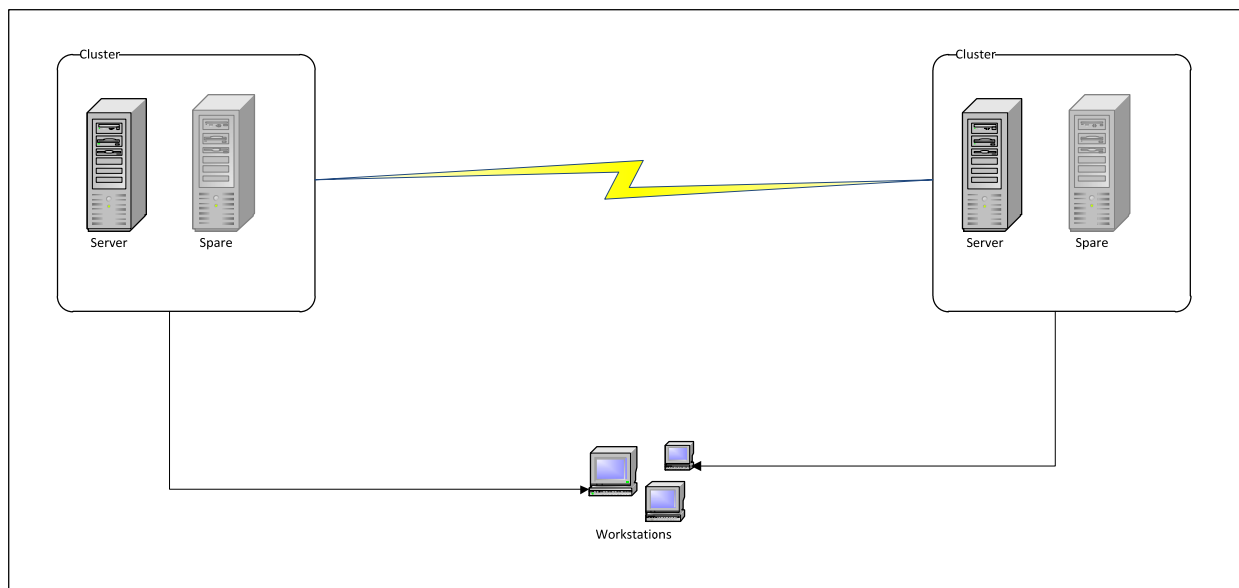


Figure 3.5: Geographic cluster.

In order for this scenario to work, the data required by the application has to be replicated between the data centers. This could be managed by the storage layer using block-level replication technologies or possibly by the application. Prior to Windows Server 2008, this scenario would require the use of third-party solutions. The changes introduced with this version of Windows included the ability for the cluster nodes to be on separate subnets and for the cluster heartbeat to be configured to allow for latency across the WAN links. There are some issues with this approach in that a network failure could potentially leave both sides of the cluster thinking they were the only one available, which could lead to a duplication or even corruption of data.

The replication of data between data centers adds a level of complication. It may be a better solution to utilize a product that synchronizes servers and data so that the whole system is kept up to date. Failover would be managed automatically and, because the same application synchronizes the data and the application, it is easy to administer.

This consideration of clustering and failover between geographically separated sites is the area where the boundaries between high availability and disaster recovery begin to blur. We will return to this idea shortly, but first we need to consider two of our major applications.

Exchange

In Exchange 2003 and earlier, we would use clustering to protect the mail service. In the event of a hardware failure, the Exchange service would fail over to another node in the cluster and the users could start accessing mail again. This does nothing to protect the email data. In the event of a disk failure, the mail database would be lost and the service would be unavailable.

The introduction of Exchange 2007 changed the rules; it introduced a number of ways of replicating data between mail servers. The servers could be in different data centers. Exchange 2010 refined this approach in that clustering is no longer available; the replication technologies have been combined and moved to the database level.

Replication for High Availability and Disaster Recovery

The replication of data gives you a disaster recovery approach. Your data is safe, offsite in another data center where it will be available if and when you need it. The failover between the instances of data can be automatic, and fast, depending on the configuration adopted. This gives us a high-availability capability. Both sides of the continuity coin are covered with a single “out-of-the-box” technology. At this point, the difference between high availability and disaster recovery effectively vanishes. One question you still need to consider revolves around backups.

Are Backups Needed?

The Exchange replication technologies allow you to produce more than one copy of the database. If you have enough copies of your mail databases, do you still need to perform backups?

It has been stated that if you have three, or more, copies of the database all in different physical locations, it is not necessary to perform a backup. This does not take into account the requirement to retrieve deleted mail or a deleted mailbox. It would also require a restore to recover from a situation where an event had corrupted the database, and the corruption had replicated rendering all copies useless.

SQL Server is the other major application we need to consider.

SQL Server

SQL Server is to be found in nearly all Windows environments, either directly supporting an application or acting as a data repository for an application such as SharePoint. The traditional approach to high availability, like Exchange, is to use clustering. In recent versions of SQL Server, this approach has been augmented by database mirroring.

Mirroring

There are multiple ways to configure mirroring: asynchronous for high performance and synchronous for high safety. In asynchronous mirroring, a transaction is committed on the principal (primary server) as soon as it has been sent to the mirror (secondary server). Conversely, in synchronous mirroring, the transaction isn't committed on the principal until it has been written to the transaction log on the mirror database. This gives a higher level of protection but could impact performance.

A further level of complexity is added by the failover options. The options previously described require a manual or forced failover. A forced failover is used in disaster recovery situations where communication has been lost with the principal but synchronization may not be complete. This can lead to some data loss. However, if synchronous mirroring with a witness (think quorum in clustering) is configured, failover can be automatic if the application is written to support multiple data sources.

Database mirroring can provide high availability and disaster recovery. It works at the level of the data compared with clustering, which operates at the service level. As with Exchange, you can supply high availability and disaster recovery using an “out-of-the-box” technology.

Log Shipping

Log shipping has been available for a long time. It consists of a simple process:

1. Take a periodic transaction log back up.
2. Copy the backup file to another server.
3. Restore the backup file.
4. Keep the target of the restore in a recovering configuration so that further restores can be applied.

This process is a cheap but very effective disaster recovery technology. It is not really suitable for a high availability solution due to the need for a manual failover, though it could be considered for second- or third-line applications (that is, those that are not judged business critical).

The Challenge of Virtualization

Virtualization is a technology that most organizations are either actively introducing or investigating. It supplies a number of advantages to an organization in that it

- Reduces the number of physical servers
- Maximizes use of hardware resources
- Reduces power and cooling requirements

A highly virtualized environment introduces some risks in that a larger number of servers, and therefore services, are dependent on fewer pieces of physical hardware. The failure of a virtual host or the disk system supporting the virtual machines could make a significant part of the infrastructure unavailable. We can apply the concepts we have used with physical servers in the virtual environment.

Virtual High Availability

The major virtualization platforms supply a way to build high availability into the virtual environment. This involves the migration of the virtual machine from one physical host to another. It is possible to perform this automatically or manually, to balance the load on the virtual hosts. This migration will reduce the possibility of a virtual host becoming unresponsive due to the load, which would make the applications in the virtual machines unavailable.

In the event of the complete sudden failure of a virtual host, the virtual machines can be configured to restart on another host. This will involve some downtime, but it will be minimal and the failover is automatic. Failover in this manner is analogous to the failover of an application between cluster nodes. The clients will need to reconnect to the applications, and it is possible that some data may be lost.

What Is Being Protected?

Virtual high availability is protecting the service. There is no protection for the data. That has to be arranged by another method. Virtualization is often hailed as a high-availability method, but in and of itself, it does not supply a full high-availability capability in the same way as clustering does not supply full high availability.

One further issue with virtualization is that the administrators need skills in the virtualization platform as well as the OS of the virtual machines and the applications they host. Our environments are becoming more complex with time, and the number of people who really understand the full technology stack we deploy today is in short supply. If you want your virtual environment to have high availability, the people administering it must have the correct skill level.

Combine with Other Methods

The virtual environment does not exist in a vacuum. It is possible to combine virtual high availability with other methods. A Web farm is a prime candidate for virtualization. Spread the virtual machines across a number of hosts and use network load balancing to distribute the load. If your virtualization platform can be configured to prevent all the virtual machines that comprise the Web farm from migrating to the same host, you will have a robust environment.

Virtual machines can be clustered, though it adds little in the way of high availability for the additional layer of complexity. One possibility is to use a virtual machine as the passive node of a cluster. This reduces the hardware needs and gives an additional layer of protection to the application. Virtualization can be used with the data replication or system synchronization techniques discussed earlier.

Combine with Disaster Recovery

It is possible to combine the high availability obtained through virtualization with disaster recovery. One example is to use a virtual environment as the target for the replication techniques discussed earlier. The primary systems could be physical (or virtual) and the replication targets could be virtual machines in a remote data center.

There is also the possibility of creating virtual environments in both data centers and replicating the virtual machine data between them using storage-based replication. The virtual machines can be configured to fail over between data centers. In the worst case scenario, the virtual machines that have failed due to the loss of the data center can be restarted in the secondary site with minimal downtime and data loss. In this type of scenario, consider running 50% of the virtual machines in each data center to minimize the disruption. An environment configured in this way will require very good, reliable, resilient network links between the data centers.

The Link to Disaster Recovery

If we take an approach that high availability takes place within the data center and disaster recovery takes place across data centers, the techniques discussed in this chapter show a convergence in the provision of high availability and disaster recovery.

High Availability and Disaster Recovery Convergence

Clustering is an obvious high availability technique, but as soon as you introduce the concept of geographic clusters, you are stepping into the realm of disaster recovery. The replication techniques for Exchange and SQL Server started out providing a disaster recovery capability but have morphed into high availability techniques as well.

A virtual environment can step over high availability and disaster recovery when data replication enables the virtual machines to be started in a remote data center with minimal downtime.

Applications that synchronize system and data to create linked machines also span high availability and disaster recovery. If the machines are in different data centers, a disaster recovery provision is automatically created. In this case, there is no downtime in the event of failure. High availability and disaster recovery form a single expenditure.

Knowing When to Stop

There is a tendency to over-engineer amongst IT professionals. It happens partly because the requirements we work to are often vague—especially growth patterns. I was once asked to create a system where the database would house somewhere between 600 and 6 million records! There is a significant difference in disk capacity for those end points.

With high availability and disaster recovery, over-engineering should be avoided. The cost of providing this functionality is going to be high without putting too many layers in place. There is a risk that trying to achieve too much might actually reduce the availability if the technologies aren't compatible. When trying to provide high availability and disaster recovery, look at the technologies available and see where there is opportunity for convergence.

Summary

High availability is a must-have in many organizations. They cannot afford downtime. The concept of creating a high-availability environment is straightforward:

- Identify which applications have to be protected. Remember that the applications support business processes and ultimately it is the organization's ability to perform those processes that you are protecting.
- Pick the right technologies. Ensure that the technologies actually do what they say. Test failover under a variety of conditions. Don't forget that the people administering those technologies may need to be trained.
- Protect the data as well as the service.
- Link to disaster recovery whenever possible. If you are providing high availability, can this be stretched to give disaster recovery as well? Could a disaster recovery technology be used that also gives sufficient high availability to satisfy your organization's needs?

As a final thought, remember to document everything and practice high availability and disaster recovery failovers often enough that the techniques and skills are usable if the real thing happens.

Download Additional Books from Realtime Nexus!

Realtime Nexus—The Digital Library provides world-class expert resources that IT professionals depend on to learn about the newest technologies. If you found this book to be informative, we encourage you to download more of our industry-leading technology books and video guides at Realtime Nexus. Please visit <http://nexus.realtimepublishers.com>.