Four Strategies for Ensuring Your High Availability Investment Will Pay Off

Lessons Learned from Best-in-Class High Availability Implementations

Introduction

Two of the top concerns for every IT organization are downtime and data loss. In fact, the cost and risks associated with downtime and lost data are steep enough to justify a fairly heavy investment in high availability (HA) and disaster recovery (DR) solutions. Indeed, most industry analyst research indicates that the average spending on HA/DR in a large enterprise is around 7-15% of the total IT budget.

Despite the plethora of High Availability technologies that can be found in the typical enterprise datacenter—clustering, load-balancing, replication, as well as new technologies such as grid computing, parallel clusters, and virtualization-based High Availability—downtime and data loss are still quite common.

The bottom line is the correlation between the levels of investment in High Availability and the actual risk reduction gained by the enterprise is still largely marginal.

This is not to say that investing in High Availability is useless but rather an indication that something critical is still missing. While the right ingredients might be in place, the overarching command and control capabilities in current High Availability implementations are still lacking.

Understanding High Availability Challenges

There are three underlying issues that make High Availability challenging:

- Configuration drift
- Cross-domain and cross-vendor integration
- Lack of automation and standardization

Configuration Drift

Configuration drift is, most likely, the one factor that contributes most to reducing the effectiveness of HA solutions. Changes to an IT environment are applied on a daily basis: operating systems, patches, and software are installed or updated; storage allocations are changed; kernel, system and networking parameters are adjusted; hardware configurations (server, network, SAN) are changed or updated; the list goes on.

Each time a change is made, the IT professional must consider if there are implications for the High Availability environment. In many cases there are, and so action needs to be taken to keep the environments in synch. Usually it's a matter of applying the same change, but not always (as we'll discuss later).

This fact, in itself, introduces the risk that some required changes might be left out. It's extremely difficult to notice such discrepancies, especially when multiple teams, such as storage, server and DBA, must all take part. But even if the change control processes are perfect, once an update has been made to all components (e.g., all nodes of the same cluster), there's another frustrating asymmetry that must be faced: most HA solutions involve both passive and active components. Even when so-called "active-active" configurations are used, in most cases this is only a means to improve utilization. For example, application A runs on node 1. Node 2 is a standby that does not concurrently run application A. To better utilize node 2, we may choose to let it run application B in the meanwhile. And here's the asymmetry: we can always tell if the active components work, simply because they are in use; but what about the standbys? How can we know they are ready to take over when needed? Consider, for example, a cluster standby missing SAN paths to a shared storage volume, or one missing a correct startup parameter. Evidently, this would not be detected unless the failover process is actively tested. Regretfully, failover testing cannot happen as frequently as it should, which could leave systems with hidden vulnerabilities for weeks or months. There's more to be said about current testing practices, but this will be covered in the next section.

In summary, a redundancy environment could easily contain hidden risks, and the more time that passes between testing its validity, the more likely it is to fail.

Cross-Domain and Cross-Vendor Integration

To add complexity, an HA system typically incorporates multiple components, spanning different IT disciplines, as illustrated in Figure 1, which is a somewhat simplified diagram.

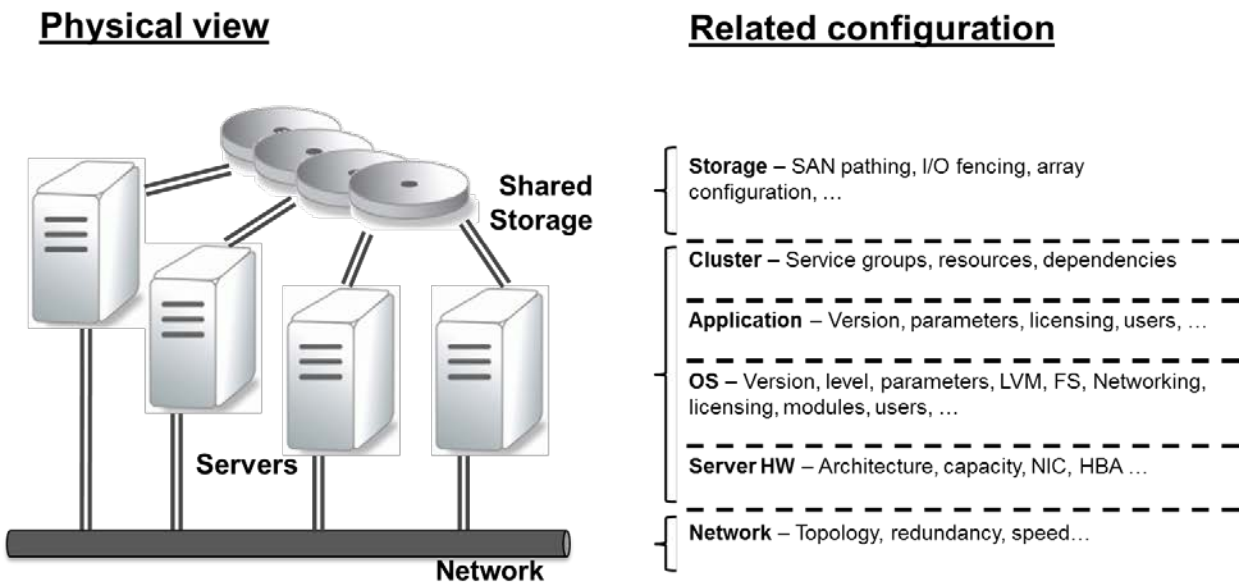

Figure 1

The Human Factor

Often, more than one subject matter expert is required to correctly configure the relevant layers. Any miscommunication might result in hidden discrepancies. For example, wishing to eliminate any single-point-of-failure in a mission-critical database, a DBA may configure redundant database control files, taking care to place each copy on a different filesystem. However, it may be that all those file systems actually reside on the same physical SAN volumes, a fact the DBA could not readily identify.

The Vendor angle

Another important aspect adding to the complexity is the need to use hardware and software from multiple vendors (storage, server, OS, cluster software, multi-pathing, etc.). Each vendor will typically publish specific guidelines and best practices for recommended setting of software and hardware, minimum required setting and configuration of other components, and so on. It is not uncommon to find these instructions ambiguous, if not even contradicting.

Lack of Automation and Standardization

Given the diversity of vendors, there is no standard toolkit for managing HA configurations in a consistent manner to help avoid configuration drift. Instead, IT administrators must use multiple point-solution tools such as storage resource management tools (e.g., ECC, HiCommand), cluster management consoles (e.g., VCS Manager, MSCS Cluster Manager, HA/CMP Cluster Manager, Oracle Enterprise manager, etc.), network management tools, server provisioning tools, and other (e.g., vCenter and SRM in VMware environments).

In recent years there has been an increased awareness of this problem by some vendors of verification tools (e.g., VCS FireDrill, Oracle CVU and Microsoft Cluster Configuration Validation Wizard). While this is a positive trend, IT professionals should be aware that none of these solutions can guarantee error-free configuration, and indeed, all of these tools are limited to the relatively obvious sanity checks.

A Deeper Dive into Configuration Drifts

Over the last six years, Continuity Software invested significant resources in identifying and classifying configuration drifts and High Availability risks. While it is impractical to cover all possible permutations, this paper will highlight the top issues most frequently encountered by large enterprises worldwide.

Where to Look for Configuration Drifts?

The following paragraphs will describe the top areas of configuration drift uncovered in each of the main IT layers illustrated in Figure 1, including:

- Cluster-related
- Application-related
- Operating system and hardware
- Network
- Storage and Storage Area Network (SAN)

Cluster-related Configuration Drifts

Setting up a correctly configured cluster is a delicate task. Even though most vendors will provide a variety of excellent administration and best practice guides, the volume of detail and variety of complex configuration options present significant opportunities for error. Even when thoroughly tested and validated, clusters can easily become unstable as a result of routine maintenance, hiding many configuration drifts that might result in either unplanned downtime or data loss in the moment of truth.

Here are the most frequently reported risk areas:

- **Storage access** – As more shared storage volumes are added, there is a chance that some of the standbys will not have correct SAN access. It is extremely difficult to notice such discrepancies, especially since standby nodes usually refrain from using shared storage resources until they need to take over. Even if all devices are accessible, you must pay attention to verifying that standby nodes have the same number of configured SAN I/O paths as well as the same redundancy setting. It is also advisable to monitor path availability. Poor performance after a failover is often associated with a standby having less available I/O paths than the formerly active node.
- **Degraded mode / bad state** – Clusters usually keep track of their state, as it may change from the optimum. Some state changes could be the result of an unnoticed degradation in some of the components (e.g., faulty or “noisy” Ethernet adapters). Other could be the result of incomplete maintenance activity (e.g., standby was suspended, or “frozen” to allow an upgrade, but was not brought back to normal state). Correctly monitoring your cluster state could save hours of unplanned downtime. Certain states are particularly dangerous, as they might lead to a “split brain” scenario, in which the standby decides to take over even though the primary is fully active. This usually results in complete data loss. Take particular care monitoring cluster heart-beat and I/O fencing configuration correctness.

- **Incorrect resource configuration** – Most clusters have no easy way of identifying whether a configuration file/repository on one of the nodes is different than the others. Make sure to compare them periodically.
- **Cluster resource to physical resource mismatch** – It is not uncommon to find cluster resources that point to a non-existing physical resource such as a missing mount-point [UNIX] or the wrong drive letter [Windows], the wrong Ethernet adapter, etc.
- **Quorum / I/O fending device** – Incorrect access to Quorum devices might result in the "split brain" scenario. Make a habit of verifying the devices are visible. (See "Testing, Auditing and Automation" below.)
- **Storage control devices** – This is an often overlooked area. Many storage arrays (e.g., Symmetrix, HDS) will allow a standby to take over shared devices only if it can communicate with the array. Usually, you need to present a host with at least one storage control device per concurrent operation. If you have too few control devices assigned to a host, it might hang during an attempt to take over multiple resources. Consult your cluster admin guide to determine the right number of devices to configure.

Application-related Configuration Drifts

Most clusters do not share application binaries and configuration files. When these are updated, you must remember to perform the same maintenance operations on all cluster nodes, which might result in configuration drift. It is a good practice to periodically audit your configuration to verify you have:

- Same installed versions /patches
- Identical or compatible configuration files
- License information
- Network objects: listening port, listener configuration
- All application data on shared storage. (In geo-clusters, also make sure it is also on replicated storage, as some storage resources are private to a local cluster and others are global.)

Operating System and Hardware Configuration Drift

Periodically verify your cluster nodes have similar configuration. In large environments you might be surprised at some of the differences you will find:

- Hardware resources (memory, CPU)
- OS version, patches, licensing
- Installed system utilities, such as LVM, multi-pathing, storage and HBA utilities
- Kernel configuration (e.g., you have increased I/O queue depth on number of max processes on some nodes, but not all)
- Network services (e.g., time, DNS) - In geo-cluster or metro-cluster configurations make sure that nodes point to a local service (e.g., to a local DNS server rather than to the same DNS server as nodes on the other site).

- Networked storage – Make sure that critical mounted networked file systems are accessible by all nodes (with the same options, mode, permission, protocol version, etc.). Pay attention to geo or campus clusters, as you should keep each node pointing at local resources.
- HBA and multi-pathing configuration differences

Network Configuration Drifts

- Link / team speed, mode (e.g., 2 teamed 1Gbps Ethernet adapters in one node vs. 1 100Mbps link in another)
- Hidden single point of failure (e.g., both private or public links on the same switch / VLAN)
- Low level / low latency stack issues (e.g., LLC, serial heartbeat):
    - Misconfiguration
    - Some are non-routable
- Firewall configuration – Make sure internal firewalls have the same ports allowed. External firewalls should allow same access rights to all cluster nodes.

Storage and SAN Configuration Drifts

- Missing SAN access to shared devices as a result of zoning or masking misconfiguration
- Non-redundant SAN I/O paths
- SAN security – A non-cluster member that has access to a shared storage device. This is a relatively vulnerable spot; storage team should periodically verify that only valid cluster nodes can access cluster volumes.
- Replication issues in geo-clusters – Make sure all data is replicated. If you are using more than one storage volume, make sure all volumes are on the same storage consistency group.
- Mixed storage tiers – It is highly recommended to make sure all shared storage devices are based on the same storage architecture and tier.

Lessons Learned from Best-in-Class High Availability Implementations

Continuity Software's close work with hundreds of very large enterprises exposes us to a variety of High Availability management approaches. Here are four key lessons learned from IT organizations that have the best grip on High Availability.

Lesson #1: Encourage Standardization

The best run IT shops minimize the number of possible HA configurations, and strive to standardize and re-use the same design patterns, such as standardizing Windows clusters to either MSCS or VCS, using the same storage architecture with all clusters, using the same software versions and patch-levels on all clusters, and so forth.

When possible, it is also recommended to use the same, internally certified, "golden image" to template all your cluster nodes.

Finally, it is important to document and publish your standards to facilitate consistency of future HA systems. Some important areas to include are:

- Minimum hardware requirements (power, internal disks, NICs, HBAs and ports)
- Networking standards (e.g., private vs. public network requirements, proprietary low-latency protocol configuration, firewall requirements)
- Software requirements (e.g., cluster software, multi-pathing software, custom storage agents and CLIs, runtime frameworks, etc.). Try to specify exact versions.
- Storage requirements (e.g., multi-pathing, zoning and masking guidelines, control device requirements and best practices)
- Naming convention (for nodes, virtual IPs, services, etc.)

Lesson #2: Develop a Collaborative Culture

Another interesting common trait of enterprises with effective HA is a cross-domain culture. A successful implementation requires correct configuration of network, storage, server and often database as well. Continuity's experience indicates that deployment predominated by just one team (often the server group only) are usually more error prone; without getting other teams educated and engaged, sub-optimal or even incorrect configurations might be reached.

We recommend forming an HA committee, that will:

- Include members from all relevant teams
- Make sure all teams have a high degree of education on HA principles and technical requirements
- Jointly design and periodically review HA architectures and configurations
- Jointly define auditing and testing goals

Many organizations new to this concept are skeptical at first, and are primarily concerned that this process is ineffective, or a waste of time. In reality, once geared up, there is very little overhead. A one-hour team review each month will usually suffice, except when new designs or architecture refresh processes are in motion. The payoffs far outweigh the time invested: better communication and increased awareness translates to more efficient deployments and dramatically reduced time to resolve issues should they occur.

Lesson #3: Conduct Fail-over Testing

Frequently testing and auditing your HA configurations is perhaps the single most important factor in ensuring successful recovery.

The most effective approach requires rotating your active nodes regularly and frequently (e.g., fail-over to a different node each weekend, and let it run in production the following week). Regretfully, this approach is appropriate in less than 5% of the environments. Most view this approach as impractical due to one of the following two reasons:

1. Fail-over is still risky and involves downtime, and therefore requires business approval, which can simply not be granted that frequently.
2. Production and standby systems are not always fully symmetrical. For example:
    - Standbys have less capacity, so you cannot afford to let them run your production applications for the entire week.
    - Standbys are located in sites with sub-optimal network (bandwidth, response times).
    - Standbys are installed with less critical applications (e.g., development or testing) that cannot be also installed on the primaries, rendering server rotation impractical.

While some cluster tools offer the ability to automate (or “Fire drill”) cluster fail-over, these tools exhibit several shortcomings:

- Automated fire drills will not test your systems under full load, causing you to miss configuration gaps such as insufficient resources, I/O paths, kernel parameters, etc.
- Automated fire drills will not test network configuration, since they operate in an isolated environment.
- Automated fire drills do not use real storage, and therefore will not detect storage connectivity issues.
- Automated fire drills will not detect certain cluster configuration discrepancies leading to split-brain scenarios (see examples in previous sections).

With these shortcomings in mind, it is important to note that automated fire drills are not, in and on themselves, sufficient to prove the validity of the fail-over process, and must always be complemented by a complete fail-over test or automated auditing (see Lesson #4).

Lesson #4: Automate Auditing

For those systems that cannot rotate between servers, the most successful approach relies on automated configuration auditing. This involves either using a dedicated tool or a set of custom, home-grown scripts. Here are some guidelines for a successful auditing environment:

- As a minimum, automate the collection of relevant configuration items (hardware, OS and software configuration, storage allocation, cluster configuration, networking configuration, etc.). Automatic data collection can dramatically reduce the time and effort involved in testing, auditing, and preparation for future downtime. Without regularly collected configuration data, it is almost impossible to perform post-mortem analysis when actual downtime does occur.
- The next level, which could prove more difficult to reach unless dedicated tools are used, is to automatically search for known vulnerabilities, such as those previously described. Being non-intrusive in nature, automated tests can be run every day.

High Availability Validation Automation and AvailabilityGuard/Cluster

As discussed earlier, the single most important factor for improving your HA environment readiness is automated testing and auditing. Writing your own scripts, while a valid approach, is often difficult and limited because:

- It requires writing and debugging a large number of scripts (some relatively complex).
- You need to make sure you configure and run the scripts on all relevant hosts (existing and new).
- You are limited to what your own experience teaches you.
- Personnel changes could render the most skillfully designed scripts impossible to maintain

AvailabilityGuard/Cluster offers an alternative approach which can prove more cost-effective and much more comprehensive than homegrown solutions.

AvailabilityGuard/Cluster employs an agent-less technology that runs on a single dedicated server. Setting up AvailabilityGuard is extremely simple, and could be accomplished in less than a day. At the end of this process you can configure AvailabilityGuard to scan your environment for High Availability vulnerabilities every day of the week, allowing you to:

- Automatically discover your servers, clusters, storage arrays, SAN configuration, replication configuration, database configuration and more
- Have complete visibility into detailed configuration information for all layers in your IT stack, store the information in a central repository, track change history, and generate custom reports
- Automatically test the validity of your HA configuration against a risk-detection knowledgebase containing over 4,900 different potential failure points updated on a weekly basis.
- Present and communicate the identified risks to your IT counterparts in an actionable format, including graphical diagrams of the environment at risk, a description of the root cause, and remediation instructions.

The bottom-line benefits delivered by AvailabilityGuard/Cluster ensure that your High Availability investment will pay off when you need it by:

- Providing end-to-end 24/7 visibility into your entire IT stack, the equivalent of running a complete High Availability test every day
- Eliminating manual labor associated with documenting, auditing, and testing your High Availability environment
- Minimizing downtime risk by capturing configuration drifts as they occur, providing expert advice on how to fix them, and keeping your environment consistent and recoverable.

Additional Resources

www.continuitysoftware.com