Hewlett Packard Enterprise

Achieving the highest possible availability in an OpenVMS cluster configuration

Contents

Technical white paper

Introduction

Is there a potential single point of failure hidden somewhere in your OpenVMS cluster configuration? Is your OpenVMS cluster really configured for the highest possible availability? OpenVMS clusters have a strong reputation for high availability, but there are a number of things you can do to ensure that you get the highest possible availability from your investment. This white paper provides details on potential failure sources in an OpenVMS cluster configuration, and discusses available techniques, tools, configuration modifications, and operational procedures you can use to eliminate or mitigate these risks.

Principles of availability

Measuring availability

Availability is defined as the amount of time that a particular computing resource is available, measured as a fraction of the total elapsed time. The closer the fraction is to 1.0, the higher the availability. This fraction is commonly expressed in percentage form, and so the closer the percentage value approaches 100 percent, the higher the availability.

A metric commonly used to quantify availability is the concept of "nines." This refers to the number of numeral nines in the availability percentage number. The more "nines" in the percentage number, the higher the level of availability, and the lower the average amount of downtime suffered per year. Table 1 illustrates this concept:

Table 1. Availability "nines"

Availability (fraction)	Availability (percent)	Number of nines	Downtime per year	Units of downtime
0.9	90%	1	5.22	weeks
0.99	99%	2	3.65	days
0.999	99.9%	3	8.77	hours
0.9999	99.99%	4	52.6	minutes
0.99999	99.999%	5	5.26	minutes
0.999999	99.9999	6	31.6	seconds
0.9999999	99.99999	7	3.16	seconds
0.99999999	99.999999	8	316	milliseconds

The following sections describe techniques you can use to achieve the highest possible levels of availability.

The most failure-prone components in hardware

It can help to identify which components are the most likely to fail, and then to ensure that these components have redundancy added to allow you to continue operating despite their failure. Components which tend to fail the most frequently include:

- Components with moving parts. This would include components like fans, and the rotating spindles and moving disk head actuators in disk drives.
- Components which generate a lot of heat. This would include power supplies, as well as the hottest semiconductor chips (particularly those which have the highest number of actively-switching transistors, operating at the highest clock rates), like CPU chips.
- Components which are very new (not yet "burned in"). The tendency for electronic components to fail most frequently early in life is termed "Infant Mortality."
- Components which are very old (because they become "worn out").

The most failure-prone components in software

It can also be helpful to know which components are the most likely to fail on the software side:

- Newly-written software, including V1.0 of anything
- New features in existing software
- Recent fixes for bugs
- Seldom-used code paths:
  - Error-handling or recovery code
  - Previous-unused features
- Old code which has worked for years, but around which the environment has changed recently

Facilities and infrastructure

If power or cooling for a data center fails, all the equipment within that data center goes down. So it is important to consider the environment (physical facilities and infrastructure) in achieving high availability.

Common practices to maximize availability of facilities and infrastructure include the use of Uninterruptible Power Supply (UPS) systems, installing a generator to cover periods of loss of utility power, and in some cases, dual utility power feeds, from different power substations or even different utility companies. Air conditioning (A/C) is typically configured with multiple independent units so that proper temperature and humidity can be maintained despite the loss of one (or more) units.

Research by the Uptime Institute (uptimeinstitute.org/) has shown that a data center with all the current industry "best practices" in place can achieve at best "4 nines" of availability. The obvious conclusion from this finding is that you will need multiple (redundant) data centers to go beyond "4 nines" of availability.

Eliminate single points of failure

A "single point of failure" (SPOF) is any component within the configuration which, if it fails, can effectively cause the entire configuration to "go down" and stop providing service. The basic principle of availability here is that you can't depend on anything which has only one instance.

SPOFs can be eliminated by introducing redundancy of components. "2X" redundancy is where either of two redundant components can operate alone if the other one fails. "3X" redundancy is where any one of 3 components can meet the need alone. "N+1" redundancy is where operation can continue despite the failure of any single component, using the capacity of the remaining components in the set (but if a second component fails, downtime results).

A company handling stock trading had a 2-site OpenVMS disaster-tolerant cluster in place. In the early morning hours of December 29, 1999, just days before "Y2K," a security guard on his first day on the job heard an audio alarm sounding on one of the UPS units. Those who have worked with UPS systems soon learn that the audio alarm is just there mostly to function as a marketing device—to tell you that the UPS (for which you spent a good deal of money) has saved you from a power outage. You can just press the "alarm silence" button to acknowledge that the UPS has done its job, and thus return to peace and quiet. Not knowing this, the security guard pressed the "emergency power off" button in a panic and shut off power to the entire data center. There was not enough time before the stock market opened to restore the power, and reboot all the systems. Because this customer had a dual-site OpenVMS disaster-tolerant cluster, processing continued uninterrupted, using the servers and storage at the other site, except for one small but important detail. It was only a short time after the power was lost that IT personnel discovered that there was only one Domain Name Service (DNS) server for the company, and the DNS server had unfortunately been located at the site which was now down. A new DNS server had to quickly be configured using a server at the other site.

An Hewlett Packard Enterprise customer purchased a high-end storage subsystem, pleased with how its internal redundancy prevents downtime. (Hewlett Packard Enterprise even shot a bullet through one without interrupting its operation: see Disaster-proof video). However, no amount of internal redundancy could save the box (or the data contained within it) when a water sprinkler in the ceiling above it started spraying water. Any single individual hardware box can be a SPOF under such circumstances, and it is only by providing redundancy in another physically-separated box (located a "safe" distance away) that you can prevent any single box from being a SPOF.

Insufficient capacity can also be a SPOF. At one fast-growing customer site, one air conditioning (A/C) unit developed a water leak. Because it is dangerous to have water in the presence of the electrical connections under a raised floor, the defective A/C unit was shut down and its water supply turned off. Unfortunately, due to rapidly-expanding workload, the amount of equipment in the data center had also expanded, and the remaining A/C units were no longer able to handle the load, so the temperature started to climb at an alarming rate. A funnel had to be hurriedly arranged to capture the leaking water, with a garden hose to carry the water outside, so they could turn the A/C unit back on.

Inadequate performance can also adversely affect availability. A service which may be technically available but providing unacceptably-high response times is arguably not really available. So in addition to sufficient redundancy, sufficient capacity needs to be provided (and maintained) as well.

In planning, you should consider the consequences of failure of any component, however unlikely a failure might be. Even components thought to be very reliable and proven can sometimes fail.

An entire data center could be a potential SPOF

An entire data center site could be considered to be a SPOF, since there exist many hazards that can disable an entire data center. Loss of power for an extended period of time, exposure of employees to an infectious disease, a nearby fire or chemical spill, a fire or explosion which destroys the site, a flood which swamps the site—any of this event could cause an entire data center site to be disabled or destroyed and thus out of service.

It is possible to set up a multisite, geographically-dispersed OpenVMS cluster such that it becomes disaster-tolerant, that is, able to survive even total destruction of one or more data centers and still continue operating uninterrupted, with no data loss.

Going one step further, even an entire geographically-dispersed disaster-tolerant OpenVMS cluster could be considered to be a potential SPOF, because there are some crucial components in a cluster that, were they to fail, could affect the operation of the entire cluster. Examples of potential SPOFs for an entire cluster include:

- A single user authorization file (SYSUAF), user identifier file (RIGHTSLIST), or queue manager file
- A single shared system disk

While it is possible to run a cluster with multiple SYSUAF and RIGHTSLIST files and/or with multiple queue files, this is not the design center for clusters, and results in either decreased functionality and/or harder system management, so few customers do that in practice. Running with multiple system disks is more practical, and many customers do that. (Some customers have system roots for all nodes on all their system disks, so that if a system disk gets damaged, the nodes using that system disk can quickly be rebooted from another system disk.)

Some customers have provisions in place which allow them to move customer accounts (along with their data) between two different clusters, so that if one cluster experiences some sort of problem, they can move the customer workload fairly quickly to a different cluster. These clusters can be geographically distributed as well.

Reliable Transaction Router (RTR) software can route transactions to two different back-end disaster-tolerant clusters, and thus prevent any single cluster from being a SPOF.

Detect and repair failures rapidly

Two measures of equipment reliability are Mean Time Between Failures (MTBF) and Mean Time To Repair (MTTR). Both of these affect availability. MTBF measures how likely it is that a component will fail, measured as the average amount of time between successive failures of that component. Then, once a failure has occurred, MTTR measures how quickly the component can be repaired and placed back into service. If only 2X redundancy is provided, and one of the redundant components fails, a fast (low) MTTR is important because a failure of the second component before the repair can be completed would result in an outage.

In an OpenVMS Cluster environment, it is easy to set up redundancy for many components, such as using multiple server nodes to protect against failure of a server, using host-based OpenVMS Volume Shadowing software to protect against disk or disk controller failures, and using multiple LANs to protect against LAN failures. But failures of servers, disks/controllers, and LAN components need to be detected and need human intervention so that repairs can be promptly done. If failures remain undetected and unresolved, eventually you will run out of redundancy and a failure will result in downtime. So failures should be detected quickly so repairs can be initiated quickly, and this implies that proactive monitoring is essential.

A disaster-tolerant cluster site had 2 redundant DS-3 links between sites. I enabled LAVC$FAILURE_ANALYSIS so that Operator Communications (OPCOM) messages would be printed on the consoles whenever a failure occurred in the extended LAN (see the article LAN Cluster Interconnect Monitoring in the OpenVMS Technical Journal, Volume 2). One day I glanced at the GIGAswitch/FDDI switches and noticed that one of the link lights was out, indicating that one of the two DS-3 links was down. I checked the console logs and discovered that LAVC$FAILURE_ANALYSIS had dutifully generated OPCOM messages six days earlier, but these had slipped by the computer operators unnoticed. (We subsequently set up a scan profile for Console Manager Software to detect the LAVC$FAILURE_ANALYSIS messages [which all have a prefix of "%LAVC"] and take suitable action, such as sending out email or a pager message to a system manager.)

Shadowset membership for disks protected by Volume Shadowing likewise needs to be monitored. Volume Shadowing software generates OPCOM messages which can be detected by a Console Management system, and other software such as HPE CockpitMgr or CA System Watchdog for OpenVMS, or even something as simple as the sample DCL command procedure SHAD_TRACK.COM from the OpenVMS Freeware CD V6 directory [KP_CLUSTEROOLS] (available from the HPE website at h41379.www4.hpe.com/openvms/freeware/collections.html) can be used to monitor shadowset membership and report any anomalies.

Take advantage of increased redundancy levels with OpenVMS

OpenVMS Clusters often support not only 2X or N+1 redundancy, but 3X, 4X, or even higher levels of redundancy. These higher levels of protection mean that the protection of redundancy isn't lost even after a failure occurs and an outage can be avoided despite multiple successive (or simultaneous) failures.

You can configure multiple LAN adapters in an OpenVMS cluster node, and multiple independent LANs to connect those LAN adapters to. The number of these connections is limited only by OpenVMS device names, which use the 26 letters of the alphabet for the controller letter.

Experience has shown that higher availability in an OpenVMS cluster is often achieved when there is at least one private LAN available as a dedicated cluster interconnect, isolated from any company-wide network.

You can directly connect the LAN adapters on two OpenVMS cluster nodes using an Ethernet cross-over cable, and although this passive cable doesn't answer to any Simple Network Monitoring Protocol (SNMP) requests from network monitoring tools like HPE Network Management Center (formerly HPE OpenView) and thus can't be monitored using SNMP, it is more reliable in practice than a network switch, because it has no active parts (and you can always monitor it with LAVC$FAILURE_ANALYSIS). A cross-over cable also has slightly lower latency for lock requests than a switch. For small clusters (example, 2 or 3 nodes) cross-over cables can form a fast and inexpensive private cluster interconnect. For larger clusters, a pair of dedicated switches is more practical for the private cluster interconnect than using cross-over cables.

Most OpenVMS clusters are connected to one or more corporate LAN networks. If you enable OpenVMS cluster traffic on the corporate LAN, you can use it as a backup cluster interconnect. If you set a lower priority for those LAN adapters using the SCACP utility, the corporate LAN will be used only as a last resort.

OpenVMS supports the use of multiple Fibre Channel Host Bus Adapters (HBAs) per node, connected to multiple independent SAN fabrics. Industry standard practice for high availability SANs is to have 2 redundant SAN fabrics, but OpenVMS clusters are not limited to 2 fabrics; many high-availability cluster customers use 4 or even 6 independent SAN fabrics, and use Volume Shadowing to shadow disks across controller pairs located on different SAN fabric pairs.

OpenVMS Volume Shadowing allows up to 6-member shadowsets as of OpenVMS version 8.4 (up from the 3-member shadowsets supported earlier). This allows disk redundancy to be preserved even after the failure of four disks (or the controller [pairs] serving them), and allows data to be preserved (albeit without redundancy) even after the failure of five disks or controller (pairs).

In addition to Volume Shadowing, most customers using OpenVMS Clusters for extremely high availability also use mirroring (or at least RAID-5) down at the disk controller subsystem level for additional protection. This minimizes the MTTR for the failure of an individual disk compared with shadowing unprotected disks, as a spare disk can be automatically provided by the controller and the mirrorset (or RAID-5 array) redundancy quickly restored, without any human intervention. If unprotected disks were shadowed, a disk failure would require a time-consuming shadowing Full-Copy operation for repair, and a lower level of redundancy would be available until the Full-Copy operation was completed.

By spreading the shadowset members containing business data across multiple sites which are geographically separated, an OpenVMS cluster configuration can be disaster-tolerant. Some customers have 3-site disaster-tolerant OpenVMS clusters, where each of the 3 sites has at least one member of each shadowset, so there are always up-to-date copies of data safely kept at 3 different geographically-separated sites. Also, having 3 sites means that redundancy of storage, servers, and data center facilities is preserved even after the total destruction of one entire site, and computing could continue (albeit without site redundancy) even after the total destruction of two entire sites. I'm aware of more than a dozen lottery operators around the world which use such a 3-site OpenVMS disaster-tolerant cluster configuration, to minimize the chances that they will ever have to halt the flow of money into the business.

With the support for 6-member Host-Based Volume Shadowing (HBVS) shadowsets in OpenVMS V8.4, it would be possible to build disaster-tolerant OpenVMS clusters with up to 6 geographically-separated sites, which would allow computing to continue uninterrupted, with zero data loss, even after the total destruction of 5 out of 6 of those data center sites.

In disaster-tolerant cluster configurations with multiple main sites like this it is possible for an OpenVMS Cluster Quorum Node to be located at an additional site. The Quorum Node can provide a tie-breaking vote for the quorum scheme so that cluster operation can continue, uninterrupted and without manual intervention, despite the loss of all but one of the main sites. It would even be possible to configure three Quorum Nodes at three separate sites to provide redundancy for the Quorum Node function. Some customers use HPE Integrity Virtual Machines (HPE Integrity VM) software to run multiple OpenVMS cluster nodes as HPE VM instances and thus allow a single Integrity Server to provide a tie-breaking vote to multiple OpenVMS clusters at once.

Solve problems once and for all

When failures do occur, it is important to determine the underlying reason for the failure, called the "root cause," so that the original problem can be fixed, to prevent a recurrence of the same problem. It is important to record data about failures so troubleshooting and determination of the root cause can be successful. If a system crash occurs, you should be sure to always take crash dumps and log calls to HPE Technology Services so that the reason for the crash can be ascertained.

At one Hewlett Packard Enterprise customer site, downtime was so painful that after a system crash the system was immediately rebooted, not taking even the few minutes of downtime required to write the crash dump file. An Hewlett Packard Enterprise consultant convinced them to institute a policy of always taking crash dumps; no matter how painful the downtime was, so that sufficient troubleshooting information would be available to allow the cause of the crashes to be determined. For the next few weeks, crashes were especially painful, but quite quickly the customer started to get answers back from HPE Technology Services identifying the causes of the crashes, the associated ECO kits were installed, and before long it became very rare that a system would ever crash. The short-term pain needed to gather sufficient data for failure analysis proved in the end to be a large long-term gain.

A Console Management system such as HPE CockpitMgr, TDi Technologies ConsoleWorks, Computer Associates Console Management for OpenVMS, Itheon iAM: Consoles, or Heroix RoboCentral should be put into place for any devices which have a serial console interface or a console port accessible remotely over the network. Servers, some switches, and some disk controllers direct error messages to their consoles, and output generated near the time of a failure can provide crucial clues as to its cause. (To aid in this troubleshooting, you should ensure that all of the hardware clocks on all devices are always kept in synchronization, including the clock for the Console Management system.) Also, the remote console access provided by a Console Management system helps improve MTTR for a failed system, because a system can be rebooted quickly and put back into service, even if a system manager is not physically present in the data center.

Log service calls for any failures you experience. Follow up on these calls to ensure that the vendor resolves the problem. Insist on getting fixes for any problems you experience, even if they might rarely occur. You are paying for the service contract; make sure you get your money's worth from it.

Record data and have tools in place in case you need them

In addition to recording and utilizing data such as crash dumps and error log files, additional tools should be in place to help provide the highest cluster availability.

HPE Availability Manager (AM) is a powerful tool for system monitoring and troubleshooting (see h20565.www2.hpe.com/hpsc/doc/public/display?docId=emr_na-c04623064). It is free from Hewlett Packard Enterprise for the VAX and Alpha platforms and the Data Analyzer and Data Server are free on Integrity. The Data Collector for AM is licensed for Integrity Servers as part of the High Availability Operating Environment (HA-OE), the earlier Enterprise Operating Environment (EOE), or via purchase of an a-la-carte license. One unique advantage of Availability Manager is that it talks to the system using network packets which arrive at high Interrupt Priority Level (IPL) and thus it can troubleshoot problems even when a system is hung and you can't log in to it. I particularly like the Lock Contention Data Collection capability in AM because it can provide clues about potential performance bottlenecks before they even start to be noticed by the users.

As noted earlier, poor performance can also be a cause of effective loss of availability. It is important that at least some flavor of performance data collector be in place and running all the time so that if performance problems arise, sufficient data will be available to diagnose and solve those performance problems.

T4 & Friends is a suite of tools available free from Hewlett Packard Enterprise (see h41379.www4.hpe.com/openvms/products/t4/). T4 collects OpenVMS Monitor data and also has a set of new data collectors for several areas which Monitor does not cover, such as LAN and IP traffic, the Dedicated-CPU Lock Manager, Extended File Cache (XFC), and Fibre Channel disk performance. The accompanying Timeline Visualizer (TLviz) tool allows graphing of T4 performance data as well as regression analysis to help determine cause-and-effect relationships.

Avoid problems

Some problems are caused by human error that occurs long before the actual outage.

Ensure that all equipment and cables are fully and correctly labeled. This prevents errors like unplugging the wrong cable or the wrong disk during maintenance work.

Always ensure that documentation is accurate and up to date.

Consider component maturity

The reliability of individual hardware and software products tends to get better over time, as experience is gained in actual field use and fixes and improvements are made. Products will have Engineering Change Orders (ECOs) issued to correct problems, and other quality and performance improvements will often be made. As a result, a piece of advice commonly heard is "Never use V1.0 of anything." So if your desire is to achieve the highest availability, don't be the first to use a new product or technology. There has just not yet been enough accumulated experience in the field to find and correct the problems.

On the other hand, if you have a product in place and it's working well; don't be too quick to discard it just because it's not so popular these days. In other words, don't throw out a perfectly good "mature" product just because it's not "cool" anymore.

On the opposite end of the spectrum, don't be the last to continue using an old product or technology way past the end of its market life. Support tends to go away, or at least to degrade in quality over time, as no new fixes are developed, developers move on to working with newer products, fewer spare parts are available, and knowledge of the product fades in the minds of support personnel over time.

Implement a test environment

One characteristic of sites which achieve the highest availability is that they have a test environment in which new products, new ECO kits, and new application software are tested. Customers without such a test environment are forced to test changes in their live production environment, a practice much higher in risk to their availability.

In setting up such a test environment, endeavor to duplicate the production environment as closely as possible. Run your unique applications, not just generic load-test software like the User Environment Test Package (UETP). Have exactly the same hardware, software, and firmware as the production environment if possible. Provide the same scale if possible, so that full-scale load tests can be performed.

One of the best examples I have seen in practice was a customer in the New York City area who had a 2-site disaster-tolerant OpenVMS cluster. They had a full test environment running in parallel across the same two sites (experiencing the same inter-site latency), using the same exact models of server and storage hardware, with the same software versions, hardware ECO levels, and firmware revisions. They could test their application from end to end, all the way from the user clients to the disaster-tolerant OpenVMS cluster at the core. The only thing their test environment could not afford to provide was enough servers to match the number of production systems and thus enough capacity to test at full-scale production load levels.

Once your test environment is in place, test new things there, before applying them in the production environment. Many customers take this strategy to additional levels to further protect the production environment, with some doing unit testing of the code in a standalone mode in the development environment first; next, testing the code with the rest of the application in a Test environment; next, passing the code across to a Quality Assurance department who do more-involved testing in an environment closely matching the actual production environment, and only after all these stages of testing are successfully completed would the change be allowed into production.

If expense is an impediment, you can often leverage test equipment for Development, Test, Quality Assurance, and/or Disaster Recovery purposes to help justify the cost. Often as older hardware is retired from production it is moved down the line and repurposed to support these additional stages of testing.

Software versions and firmware revisions

What version of software and firmware you use can have an effect on availability.

Use supported versions at all times. Troubleshooting assistance is available for officially-supported versions, and any problems you encounter will get fixed. Test new versions in a test environment first.

Patches and ECO kits

Monitor the releases of software patches and ECO kits. This way, you will know what potential problems have been found and fixed. Hewlett Packard Enterprise allows you to sign up on an email distribution list which announces new patches/ECO kits. This information is also archived in the ECO notes file on the DECUServe system. As you learn what problems are being found and fixed, consider the likelihood you will encounter a given problem, and use this as one factor in deciding how quickly you implement a given patch or ECO kit.

If you require high availability, consider letting a patch or ECO kit "age" for a while after release before installing it in production. This also helps avoid the problem of installing a patch kit only to find out shortly afterward that it has been recalled due to some problem with the patch. But avoid waiting too long to install patches or ECO kits in production, or you may suffer a problem which has already been fixed, and which might have been avoided entirely. Install patches and ECO kits in your test environment first to detect problems with the patch in your specific environment.

Managing change

Changes in the configuration or environment inherently introduce risks. By managing change, you can manage the associated risks.

When a new problem occurs, it is common practice to scrutinize recent changes to see if they could possibly have caused or contributed to the problem. So change tracking is a valuable part of change management.

But keep in mind that things outside the control of the change management process can also cause problems. One Hewlett Packard Enterprise customer had a severe performance problem suddenly appear in a two-site long-distance disaster-tolerant cluster. All of a sudden it made a great deal of difference at which site the node assigned the lock mastership duties for certain lock trees was located, whereas it had never mattered before. The customer developed procedures to control lock mastership placement and mitigated the performance problem, but wondered why this mattered so much now. It turned out that a flood had washed out a highway bridge which happened to carry the fiber optic links for both of the two different vendors contracted for inter-site DS-3 links, and while both vendors' SONET rings wrapped and hid the failure, the inter-site distance and therefore packet latency between sites was suddenly much greater than it had ever been before. Careful scrutiny was made of recent changes in the change management system, but this change wasn't one which was under the control of any change management system. Ensuring that the network circuits actually had diverse routes would have been helpful, instead of merely depending on having two different vendors for the two redundant inter-site links.

As change is managed, there must be a trade-off and a balance between keeping up to date and minimizing unnecessary change. It is unwise to have no change management process in place. But if, because of excessive fear, the change management process becomes instead more of a change prevention process in practice, that can also be harmful. And keep in mind that even if a system itself does not change internally, the environment in which it operates will inevitably change, and so changes must be made within it to accommodate changes in its outside environment. If these changes are not made in a timely fashion, the risk is increased.

A required piece accompanying any proposed change should be detailed (and tested) procedures for reversing it ("backing out" the change) if something goes wrong.

Try to change only one thing at a time. This makes it much easier to determine the cause of a problem and makes it easier to back out a change if necessary.

Plan for regular, periodic scheduled downtime for various maintenance activities. With today's technology, it is often true that you **will** have to have **some** downtime. But by your decisions, you can often exercise the choice as to whether you want it to be mostly **planned** or **unplanned** downtime.

Computer application of biodiversity

This concept can be applied in computing. Different implementations of a technology, or different generations of technology, are vulnerable to different faults. Probably because different people are involved, different bugs tend to surface and different weaknesses are present. So using a mix of different implementations of technology has the potential to improve the survivability of the system as a whole.

Within the world of biological study is a concept called "Biodiversity." When a population is threatened, small differences may help some to survive a given threat.

For example, there was a widely reported outage of a Telecom organization's X.25 network a few years back. They did an upgrade of routers to a new version of firmware, all at the same time, and a fault caused their entire X.25 network to be down for 18 hours. With no diversity in the network, it all succumbed to the same problem.

I worked at a customer site where there was in place a mixture of GIGAswitch/FDDI switches and bridge/routers for the LANs. There was also a mix of DECnet, Local Area Transport (LAT) and TCP/IP protocol stacks. Their disaster-tolerant OpenVMS cluster used both the GIGAswitch/FDDI and bridge/router boxes for the cluster interconnect. One day a fault occurred which took the entire IP network down. (Rumor says a loop developed in the LAN configuration that wasn't properly handled by a Spanning Tree algorithm, causing broadcast packets to loop and multiply to the point of network "meltdown.") During this outage some important lessons were learned:

- The OpenVMS disaster-tolerant cluster continued to function, using the GIGAswitch/FDDI. Having a mix of LAN switches allowed the cluster to continue. Using cross-over cables for a smaller (2–3 nodes) cluster would also have worked.
- DECnet and LAT protocol communications continued to work, allowing access to systems which could not be reached using TCP/IP. HPE Reliable Transaction Router software allows use of both DECnet and TCP/IP, with the ability to prioritize either as the primary transport and with the ability to fail over to the other protocol in the event of a problem with the primary protocol. This could be used to help protect against problems on either the TCP/IP or DECnet networks.
- DE602 Ethernet adapters had been introduced which used one vendor's Ethernet interface chips. DE602s which were exposed to the IP network problem traffic "locked up solid" and would not communicate until the system was powered down and back up. Some systems with DE602s crashed immediately. Older DE500 Ethernet adapters with another vendor's chip in the design, although exposed to the same IP network traffic, continued to function normally. Systems with a mix of DE602 and DE500 adapters could continue to communicate using the DE500 adapters. A firmware fix for the DE602 adapters later reportedly fixed the problem, but this event pointed out that having a variety of implementations of hardware in redundant configurations could allow you to continue operating in the face of a problem.

The same principle could be applied to storage hardware. Using Volume Shadowing software to mirror disks between different disk controller families could provide higher potential availability and performance than using multiples of the same controller models. EVA and XP storage is older and less stylish now than the new, fast, and trendy 3PAR storage, but it remains proven and reliable. Using a mix of storage controllers might allow you to take advantage of the strengths (and mask the weaknesses) of the different controller models.

Minimize complexity, and number of components which can fail and cause an outage

Try to minimize the overall complexity of the configuration. If there are a set of components which must all be present for the application to work, try to minimize the number of such components which are required for successful operation.

Use the simplest hardware necessary to do the job. One customer had an HPE AlphaServer ES40 system which "locked up" due to a hardware problem and caused a 26-second outage for the rest of the nodes in the cluster. It turned out the AlphaServer ES40 was basically just being used as a quorum node. The extra complexity of having 4 CPUs, several memory boards, and additional PCI I/O cards in place only contributed to the complexity of the system and increased the chances of a hardware failure. Using a simple single-board single-CPU AlphaServer DS10L would have been better.

It can help availability to minimize node count in clusters. The more the systems in a cluster, the higher the chance that a system may crash and disrupt the rest of the cluster. So wherever possible, use fewer more-powerful nodes for a high-availability cluster, rather than more of less-powerful nodes.

The customer mentioned above was also using 5-node clusters, where two nodes ran one application and two other nodes ran a completely-separate application. Any time there was a problem with the 5-node cluster, it affected both applications at once. We advised them to place the two separate applications onto two separate 3-node clusters, so that a problem with one application cluster would not affect the other.

Reduce the extent of impact of a failure

It can be helpful to follow the popular adage: "Don't put all your eggs in the same basket." Minimize the effect of any single failure.

- Preserve the independence of different resources.
- Divide services into portions so that failure of one portion doesn't affect all portions. This is a trade-off between span of impact, and complexity and thus frequency of failures.

Be cautious about sharing resources among unrelated services, despite the lure of cost savings. Popular but potentially-dangerous areas include:

- Shared storage subsystems, and storage virtualization
- Server virtualization

It can be difficult to diagnose the source of a performance problem in one environment, for example, when that environment shares an underlying physical server with multiple virtual systems, or shares a storage subsystem with other environments. Basically, you have to know what is going on in all the environments simultaneously before you can detect the potentially-adverse effects of interference between the environments.

Realize that anything which is connected together might have interactions. Try not to connect together redundant components.

Sometimes an inoperative component can prevent another redundant component from working. So incorporate into the system design ways of removing, isolating, or avoiding the use of a failed component.

Summary

This article has provided a variety of ideas on how to improve the availability of your OpenVMS cluster environment. We have looked at how high availability is measured, and identified the most failure-prone components in hardware, software, and data centers. We have seen examples of single points of failure in action, and looked in depth at the capabilities OpenVMS cluster technology has to overcome them. We looked at how processes, procedures, and configurations affect availability. Use these techniques to improve the availability of your OpenVMS cluster.

Resources

Learn more about maximizing availability
availabilitydigest.com

Learn more at

hpe.com/docs/openvms

Sign up for updates

© Copyright 2013, 2016 Hewlett Packard Enterprise Development LP. The information contained herein is subject to change without notice. The only warranties for Hewlett Packard Enterprise products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. Hewlett Packard Enterprise shall not be liable for technical or editorial errors or omissions contained herein.

4AA4-7735ENW, October 2016, Rev. 1

Hewlett Packard Enterprise