Principles of Troubleshooting and Problem-Solving

Based On A Case Study

How do we solve problems?

The first step is to accurately describe the problem. This involves identifying and documenting:

1) What was done (that is, the sequence of events and actions that occurred)
2) What results were expected, based on what took place
3) What actual results were observed, instead of the results expected

> *HP worked with a customer recently who had a 3-site disaster-tolerant OpenVMS Cluster in place. In their thorough testing in preparation for going live, they ran the new system with live inputs each day, verifying that they got the same results as the old system they were intending to replace. Then at night, they ran various stress tests. They had an application test where they first backed up the production data state, then restored the data to a known starting point. Next, they "played back" a large set of transactions on which the application operated to modify that data, and then the resulting data was examined to see if the correct results were generated. Any discrepancies between the correct results and the observed results would indicate an application error. After one run in the summer of 2009, during which a failure of the storage subsystem at one site in their disaster-tolerant cluster configuration was simulated by disabling SAN ports and thus logically disconnecting the controllers holding one copy of the storage from the SAN, the customer found that if they afterward examined a file with a text editor from one OpenVMS node in the cluster, they saw certain data in the file, while if they looked at the same file from a different OpenVMS node, the text editor displayed different data within that same file. They were, of course, expecting to see the same data in the file regardless of which node in the cluster they examined it from.*

It is important to record the evidence of the anomaly in such a way that it can be readily shared with the Engineering organization supporting the product. Both the exact steps taken and the exact output seen are important to record.

> *Unfortunately, this customer's test procedure involved restoring the original Production data after the night-time test, and this process destroyed all the evidence of the corruption seen.*

At this stage, it may be necessary to clarify whether the observed behavior is really incorrect, or merely unexpected. Sometimes a result may be unexpected or unintuitive, but still meet the design and operational goals of the product.

Since an OpenVMS cluster is designed such that the environment (including the contents of files) will always appear identical to users regardless of which node the data is accessed from, this appeared to be incorrect behavior.

At one stage in our analysis, we found differences in the data between the copies of files on different shadowset members in the same shadowset, but the differences were in data areas beyond the End-of-File mark within a file, which had never been written to. We realized that if the shadowset had been created from the start with multiple members using an $INITIALZE/SHADOW command, and disk high-water marking were disabled, and if no Full Merge operation had subsequently occurred on the shadowset, then areas on the shadowset which had never been written to might indeed contain different data, and this situation could be perfectly normal and valid for the Volume Shadowing product.

Differences under other circumstances, or in other areas of the files, would be another matter entirely.

At this stage, it is important to record the details of the environment in which the anomaly ***does*** occur, and any environment (or variation of the environment) in which it does ***not*** occur. It usually turns out that some factors about the environment are crucially important to being able to reproduce the problem, while others do not matter at all. At this stage, we don't know what is important and what is not. So we record all the details we can.

As I mentioned briefly before, the customer had a three-site OpenVMS disaster-tolerant cluster, with storage located at two sites and a quorum node located at a third site to provide a tie-breaking node for the cluster. The storage was StorageWorks EVA controllers, and the SANs at the two sites were connected using Dense Wave Division Multiplexing (DWDM) technology to extend the SANs between sites. There were a total of seven OpenVMS nodes; two of the nodes were Alphaserver systems, while the remaining five were Integrity Server systems. All nodes were running OpenVMS Version 8.3. The OpenVMS nodes were connected using dual Gigabit Ethernet LANs as the cluster interconnect, which were also connected between sites over DWDM. Volume Shadowing Software was used with the intent of keeping the contents of the disks at the two main sites identical at all times, so that if one site were destroyed, the application could continue to operate, uninterrupted, and without data loss, using the surviving copy of the storage.

The customer actually had three of these disaster-tolerant OpenVMS clusters in Production, which were all spread across the three sites. There was a 7-node cluster, a 5-node cluster, and a 3-node cluster. They also had a test cluster located entirely within one site, and some other test systems. The anomaly had been observed in the largest cluster, the 7-node cluster, but had not been observed in the other two Production clusters during the test. It had also not ever been observed in the test cluster. It had also not been observed in many other repetitions of this same application test in the Production environment in which the logical disconnection of the SAN did not occur.

Based on this information, the next step is to be able to reproduce the problem, with the goal of being able to cause the same problem, repeatedly (or at least in some fraction of the attempts). This involves repeating the same steps (or possibly variations on those steps, trying to find a set of steps which can reliably reproduce the problem), and checking for anomalous results.

> *HP's Multi-Vendor Systems Engineering organization sent a team of engineers on-site to help the customer. While the MSE team was on-site, the application test was repeated, including logically disconnecting one set of EVA storage from the SAN. Files were examined with the text editor, and cases found where the contents appeared to be different from different nodes in the cluster (and **not** just in areas beyond the end-of-file mark).*

During this step of reproducing the problem, it may be possible to narrow down the specifics of the environment needed to reproduce the problem.

> *It was observed that when the contents of a file were examined from any of the OpenVMS nodes located within either one of the two main geographical sites, the contents would appear identical, while examining the data from OpenVMS nodes located at different geographical sites would result in differing displays of the file contents.*

At this point, it is usually helpful to list all the possible causes you can think of which might conceivably produce the anomaly observed. To do this most effectively, it is necessary to have some knowledge of the internal operation of the hardware and software products involved, or to work cooperatively with a level of HP support personnel who have such knowledge and can contribute at this stage of the analysis.

- *Data could be corrupted by a CPU hardware problem within an OpenVMS node*
- *Data could be corrupted by a hardware problem with memory within an OpenVMS node*
- *Data could be corrupted by a hardware or firmware problem with a Fibre Channel Host Bus Adapter (HBA) as it passes data to or from a storage subsystem*
- *Getting different data returned from different nodes might conceivably be caused by a bug within the eXtended File Cache (XFC) code within OpenVMS which happens to cause differing cache contents on different nodes in the cluster*
- *Because Volume Shadowing keeps multiple copies of data, if somehow a difference developed between the contents of different shadowset member disks, that might explain seeing different data at different times when the data is accessed*
- *A problem in SAN switch hardware might conceivably corrupt data as it passes through the SAN*
- *Data might be being corrupted as it passes through the DWDM equipment between sites, resulting in different data at the two sites*
- *Data might be being corrupted by a hardware or firmware problem within an EVA storage controller*
- *Data might be corrupted by a hardware or firmware problem on a disk within the EVA controller*

Successful troubleshooting at this point involves identifying methods which can eliminate one or more of these potential causes as the actual cause.

Hardware problems in a specific piece of equipment can often be ruled out by using a different physical piece of hardware. By swapping out hardware or changing the test to instead use a different piece of hardware, you can often eliminate failure of a specific piece of hardware as a potential source of a problem.

(It is necessary to keep in mind that identical instances of the same equipment with the same hardware logic, software and/or firmware could have identical problems in software, firmware or hardware logic, so swapping with identical equipment may not be sufficient to eliminate those factors as potential root causes. It may be necessary to swap a piece of equipment with functionally-equivalent but completely-different equipment to isolate the problem. For example, the same identical problem might occur on multiple EVA controllers of the same hardware model and firmware revision, but that storage might be swapped with XP or MSA or local SCSI storage to see if the problem was or was not related to an EVA problem. If the problem remained when using XP or MSA or local SCSI storage, one could then rule out the EVA as the root cause.)

- *The problem occurred on both Alphaserver and Integrity Server nodes in the cluster. This made it unlikely that a CPU hardware problem or CPU hardware logic error was the root cause, because Alphaserver and Integrity Server nodes use completely different CPU designs.*
- *The problem occurred on multiple nodes within the cluster, and occurred despite having different subsets of nodes involved in different tests. This made it unlikely that a memory hardware problem was the root cause, because each node has separate and independent memory hardware.*
- *The problem occurred on both Alphaserver and Integrity Server nodes in the cluster. Alphaserver systems and Integrity Servers use different brands of HBAs, from Q-Logic and Emulex. So it was unlikely that a hardware, hardware logic, or firmware problem on an HBA was the root cause.*
- *Tests were done which used only the nodes and storage at a single site within the cluster, and the problem could still be reproduced. This allowed us to eliminate the DWDM link between sites as a potential contributing factor to the problem.*
- *The problem occurred regardless of which EVA storage controller was used to store the data for a test. This allowed us to eliminate a problem with a specific disk within an EVA storage controller as a potential source of the problem. (We could not eliminate a hardware logic or firmware problem within the EVA as a cause at this point, because all the EVA storage subsystems were of the same model and had the same firmware.)*

Knowledge of product internals of Volume Shadowing Software became crucial at this point, because we were able to identify that because of the SYSGEN parameter settings for SHADOW_SITE_ID on each node, coupled with $SET DEVICE/SITE values set for each shadowset member, nodes at a given geographical site, while writing simultaneously to

shadowset members at both sites, would tend to only read from the shadowset member physically located at the same site. So by choosing to look at a file from a node at one site, we could see the data for the shadowset member at that same site, and looking from a node at the opposite site allowed us to see the data for the shadowset member at that site, and thus to detect differences. Knowledge of the existence of the diagnostic command $ANALYZE/DISK/SHADOW and that it would read from the shadowset members at both sites and compare their contents, regardless of which node in the cluster the command was invoked from, also proved key at this point.

It can be helpful at this stage to be familiar with what types of failures most commonly occur, and when in the life cycle of a product the failures typically occur.

For example, in computer hardware, electronics tend to fail very early in their lifetime, and then experience a long period of time before they tend to start failing frequently again, due to old age. Then when hardware failures do occur, the items which fail most often and most quickly are those which contain moving parts, which tend to wear out over time, and those which generate a lot of heat. So, for example, we find that fans and power supplies, as well as spinning magnetic disks, tend to fail.

In computer software, we find failures occur most often in areas where the code has been modified recently, either to add a new feature, or in making a change to correct a different problem. Code which has been stable for a long period of time tends to continue working well, without problems.

In this customer's case, it was initially hard for us to believe that Volume Shadowing code would be the source of the difficulty, as it had a long history of stable operation over many years at many customer sites. It was only when we realized that a new Volume Shadowing feature had been recently introduced (as of version 8.3), called Automatic Mini-Copy on Volume Processing (AMCVP) that it started to make sense that we might be seeing a defect in this area. And with that realization came the ability to focus our testing on areas connected with this new feature.

Once the problem can be reproduced, the next step is usually to try to simplify the environment required to reproduce the problem, keeping only those characteristics of the environment which are actually required to duplicate the problem, while eliminating extraneous factors.

Some of the steps taken to simplify the test environment at this customer site included:

- *Reducing the number of shadowsets involved from several down to just one.*
- *Replacing the disabling of SAN ports serving an EVA with the equivalent action of un-presenting a specific disk LUN. This allowed testing of the disruption of access to a LUN with much less disruption (and risk) to the overall environment than affecting an entire EVA.*
- *Replacing the customer's complex application test with a set of small batch jobs which simply moved copies of some of the customer's data files onto the test shadowset from multiple jobs on each node at once.*
- *Reducing the number of nodes from which the shadowset was mounted, and from which data was copied to the shadowset. The failure occurred with 6 nodes and with 4 nodes, but*

not always with 3 nodes, and in a single test with 1 node involved we were not able to reproduce the problem.

It can also be helpful at this stage to think of ways to highlight the problem or to make it more readily visible.

By initializing the contents of a test shadowset to contain blocks containing all zeroes, it becomes obvious what data has been written to the disk. And a block which has not been written to since the volume was initialized stands out prominently because it remains at the initial state of containing all zeroes.

To be successful in troubleshooting requires a certain degree of discipline and order. It is helpful to always follow the principle that you ***only change one thing at a time***. By only changing one aspect of the test at a time, it is easy to conclusively identify exactly what factor is involved in any difference between two consecutive tests. And this also makes it easy to return to the previous state if a given test gives unsatisfactory results (i.e. the test is unable to reproduce the problem again after a change in the test procedure). It is also crucial to accurately record any changes made, and the test results.

It often takes several iterations of testing and thought to simplify the test.

We were able to simplify the steps required to reproduce the problem down to these synthetic test steps, which we then passed along to OpenVMS Engineering to help them reproduce the problem. These steps also allowed us to highlight the problem:

1) *Initialize the shadowset members to contents of all zeroes using $INITIALIZE/ERASE*
2) *Mount the shadowset on nodes from which we will be doing the $COPY operations*
3) *Start copying files with DCL $COPY commands to the shadowset using 4 Copy jobs per node, for some number of nodes (preferably 4 or more nodes)*
4) *While the copying is going on, un-present one shadowset member from the EVA using CommandView.*
5) *Watch the shadowset member get thrown out and a mini-merge bitmap get converted to multi-use.*
6) *Wait two or three minutes for Copy I/Os to continue*
7) *Re-mount the thrown-out shadowset member back into the shadowset with a Mini-Copy operation.*
8) *After the Mini-Copy has proceeded for at least 2-3 minutes, the batch jobs doing $COPY operations can be terminated; the damage will already have been done by that point.*
9) *Allow the Mini-Copy to complete.*
10) *Use $ANALYZE/DISK/SHADOW to see if corruption is present, and to examine the pattern of corruption (always zeroes within entire blocks on one of the two members, and in each case we always see zeroes on the member which has been removed from the shadowset and later re-inserted with a Mini-Copy operation)*

Once the steps required to reproduce the problem can be identified, it can be helpful to create scripts you can use to reproduce the problem each time. These same scripts can also be provided to the Engineering organization to make it easier for them to reproduce the problem.

> *DCL command procedures were developed in the customer's test environment to automate most of the steps shown above, with user prompts included to allow the test to pause or wait in between steps until the person controlling the test is ready to take the next step. These scripts also made it easier for HP personnel on-site to reproduce the problem later and then, in time, to confirm that the patch kit released by OpenVMS Engineering actually fixed the problem in the customer's test environment.*

Working together with HP, we can solve problems and improve the quality of HP products for everyone's benefit. Involve HP as soon as a problem is detected. HP may make suggestions that are totally impractical or at least difficult to implement; be honest about the impact of a suggestion in your environment, and an alternative may be found that has less impact. Sometimes the "No Pain, No Gain" principle may apply, and some degree of pain may be required to get through and solve a problem. And finally, since you are the one most familiar with your environment, your advice and intuition are crucial to the success of the troubleshooting process.

> *Thanks to the excellent cooperation of the customer along with the expert help of folks from HP Services' Multivendor Systems Engineering team, with OpenVMS Engineering was able to determine the root cause of the problem and quickly produced a patch kit that resolved this problem.*

Keith Parris, September 2010, based on original document v1.1