When Disasters Collide

A Many-Fanged Tale of Woe, Coincidence, Patience, and Stress, Continued . . .

DOUG HUGHES

Doug Hughes is the manager for the infrastructure team at D. E. Shaw Research, LLC., in Manhattan. He is a past LOPSA board member and was the LISA '11 conference co-chair. Doug fell into system administration accidentally, after acquiring a BE in Computer Engineering, and decided that it suited him.
doug@will.to

In my previous column we left off with the third of four major incidents. As a recap, we had a 160 TB storage backup server almost lose all of its data, a peculiar WAN failure involving some closely spaced and mismatched fiber-optic transceivers, and a flash SSD failure on a primary application server, all around the same time. I left a teaser about our big storage system. The story continues . . .

Issue 4

Prior to this issue happening, we had an unfortunate misunderstanding with the storage vendor about how we wished the metadata to be arranged. We wanted it mirrored across two RAM/flash devices in case one died. This required a policy-level declaration in the file system (GPFS) that had never been set! So, at the inception of this incident, we were several weeks into trying to correct this oversight by restriping all of the metadata back to spinning media in preparation for repairing the RAM/flash LUN setup. A restripe operation also takes care of ensuring appropriate metadata replication at the same time. Alas, going from RAM/flash to spinning media drops the IOPS by about 1000:1, so this is a tedious process and has many other impacts.

A little bit of background about the architecture is necessary to understand the failure that I'm about to describe. The storage "system" is made up of four Linux hosts which serve GPFS and NFS to clients and also talk to two large storage cabinets attached to a Fibre Channel SAN network. Everything else I will be describing lives in these large cabinets (except for the RAM/flash devices described earlier; those are separate). The first storage cabinet has two storage controllers connected to a number of disk shelves. Each shelf is divided into a left half and a right half. The storage controllers are each directly connected to all shelves. Controller 1 normally services the left half and controller 2 normally services the right half of each shelf. If one of the storage controllers dies, the other controller will be able to service the disks. The second cabinet has an identical set of shelves but no head controllers. Each shelf in the second cabinet is chained to the corresponding shelf in the first cabinet (i.e., cabinet *A* first shelf is chained to cabinet *B* first shelf, and so on). The storage controllers each have five, two-channel SAS cards. Each SAS channel (port) connects to an I/O module serving one-half of a storage shelf. Each storage controller also has four Fibre Channel ports for talking to the Linux hosts, for a total of eight optical ports (A and B path) across the two storage controllers. Since we have four Linux hosts with two Fibre Channel ports, each host is connected to both controllers simultaneously using direct-attach point-to-point

Fibre Channel links. Every host and controller can communicate on all channels (SAS and FC) simultaneously for purposes of throughput , balancing, and failover.

Each storage shelf (holding all of the disks) has two I/O cards, one of which is an A path, and the other of which is a B path. All SAS and Fiber Channel paths are active at all times. Internally stored configuration and physical cabling together control which paths are in use as primary and which are secondary, and the primary/secondary paths are alternated to achieve the best balance.

Here is a summary of various failure scenarios:

- GPFS Linux server fails: NFS and GPFS failover to another of the three directly attached servers.
- A Linux/controller Fibre Channel cable or card fails: a path to that controller is disabled on that host, and three other paths are available to that controller. Half the capability of a given Linux node is disabled, but the other FC path remains active.
- A storage controller fails: half of the paths to storage are lost. Uncommitted blocks may be lost. Disk commit journals are lost (see below) because journals are kept per storage controller for the RAID groups for which it is defined as the primary.
- A storage controller SAS card fails: half the storage in two shelves is inaccessible, because it is a two-port card. Because the storage controller itself is still alive, a failover for these two paths does not occur to the other controller. Only if this controller totally fails does the second controller take control of all paths. It's all or nothing. The controller also keeps journals of RAID logical units (LUNs [5]) to replay when the card is replaced. The LUNs are RAID-6 [4] (double parity), so everything keeps running even though half of the RAID-6 LUNs could be degraded by two disks. Another single disk loss in the wrong stripe would cause data loss.
- A storage shelf card fails: half the disks in that shelf are unavailable. Half the logical units are down (one disk out of two). Storage controller journals mark blocks to rebuild on the missing disks once the card is replaced.

A commit journal is a journal kept on each of the disk storage controller nodes that keeps track of the state of the dual parity blocks in that system. If an I/O card (on the controller or in the shelf) fails, the journal will buffer RAID information so that a full rebuild is not necessary when the card is replaced; only blocks that have been written while the disks have been unavailable will need to be affected. Also, if a disk is pulled from the system and put back in, the storage controller will recognize the disk as one of its own and will rebuild the blocks of data that it missed while it was unplugged, allowing for a fast rebuild on insert. This is important, and I'll explain why in a few paragraphs.

The Incident

Sometime in the midst of the WAN outage and just prior to the backup server outage, one of our storage controller heads had an incident. It was a lot like a seizure. The storage system got very slow. We saw lots of disturbing messages on the console about timeouts and retries of various components, and disks started to disappear from the storage controller view. Eventually, a storage controller timed out two of the disk channels and all of the disks attached to that storage controller node on those two channels. This made half of our RAID-6 LUNs degraded by two disks. (Half because the other controller was fine, and each controller is primary

for half of all disks, as explained above.) The normal procedure here is to swap out the controller I/O module, which we did, and then reset the controller. Unfortunately, this failed. More serious messages happened. We got the vendor on the phone, and they suggested we leave the storage controller off while they shipped us a replacement. They shipped overnight. We were degraded, but we could wait. It wasn't a comfortable wait, given that we were down by two disks in many RAID-6 LUNs, and one more disk failure out of about 450 meant lots of lost data, but we had little choice. At the time, what we didn't realize was that we lost all of the journals for this storage controller too, or we would have been all the more apprehensive.

The vendor was able to beat the 4 p.m. shipping deadline for priority overnight, and we had the replacement storage controller the next morning. We swapped out the controller in about an hour (they have a lot of cables, and they need to be attached to the right places!) and started the rebuild process. It was then that we realized that things were worse than they had seemed: the journals were gone and we would have to rebuild 38 16TB logical units and a number of smaller ones. Worse, with this generation of storage controller node, only one disk out of each pair (two were missing) could be rebuilt at a time, and only six rebuilds could run in parallel per storage controller head node. This was a week-long series of rebuilds where any single drive failure in any one of the logical units would result in permanent data loss at a massive scale.

Then the second controller failed . . .

Appallingly, while the first storage controller was rebuilding, within two hours of its being replaced, the second storage controller failed! Luckily, this failure did not start with I/O modules failing (for some value of lucky). This failure resulted in losing two disks on every other RAID-6 stripe in the system because of the channel failures to those shelves. So, now we are down two disks in every RAID-6 stripe in the storage system! Previously, it was only half of the stripes.

After replacing the second storage controller node, we started getting more errors on the same two channels that had failed in the first place. The vendor, now on high alert, advised us that we should power cycle the two storage shelves full of disks to reset their I/O modules. Knowing that we could not tolerate another single disk loss without data loss on a very large scale (because data blocks are striped across all RAID-6 logical units by GPFS), this operation had to be done exactly right. It's worth noting that at some point in the past, the entire cabinet full of disks had been moved from another building, and some of the power cables were mislabeled. To be specific, the labels were correct at the time, but they had been attached to the wrong chassis on re-cabling. The person who was cycling the power looked at the label, flicked the power switch, and hysterics ensued. I was watching the console and started seeing a continuous stream of errors on the third channel, muttered an expletive, and took the extreme action of doing a forced shutdown of all four Linux nodes to try to preserve file-system integrity. Had the technician looked at the overall situation carefully instead of in a hurry, he would have noticed that the disk shelf he was power cycling could not have been either of the shelves in question. The only ones that should have been cycled were the last two in the system, the bottommost two. The shelves are identified in such a way that this is a fairly easy thing to double check. He had turned off the third from the bottom. OUCH.

Since the replacement of the second controller, we had been running the file system un-shared and un-mounted. Normally, it serves around 640 TB of primary chemistry data to analysis clusters and serves as the primary write target for the

chemistry supercomputers, but we wanted to get things stable. After the unfortunate cycling of the third shelf, and the emergency shutdown of the file system, we started bringing things up carefully. First the shelf was turned back on. Then we started the Linux storage servers one at a time to make sure they could see the disks, and then we mounted the cluster file system one node at a time. So far, so good. The file system was mountable, basic commands like `ls` and `df` were okay, and there didn't seem to be any permanent harm done. However, now we had to deal with the repercussions of the loss of the third disks in many stripes. And we dreaded that we would have to run an fsck!

Thus, we got the vendor on the phone again, although this wound was self-inflicted. The I/O card in the storage shelf was replaced, channel messages were no longer occurring, and things were generally stable with a lot of broken disks. Here's what we knew at the time:

- All RAID-6 stripes were now running bare-minimum.
- One more disk failure out of a little over 800 would result in a lot of data loss.
- Half of the disks marked failed did have RAID journals, so recovery should be relatively quick for those.
- Many of the remaining disks had been in low-power mode (MAID [1]).
- The emergency shutdown might require an fsck.

Our data flow is such that the current working set of chemistry data is typically data that has been written in the last month. Any files that have not been modified in two months and accessed in three months are migrated to MAID storage, and those disks spin to low power mode to save energy. These migrations happen every Sunday. This is important because we know that these disks were not being written during the failure, so it should be possible to mark them as "okay."

First, we targeted the low-hanging fruit. Step 1: run the command to rebuild all the LUNs with journals. This brought half of the disks online in about two hours. This was a good start! At this point we decided to bring the file system online. The supercomputers, expensive resources, were stalled at not being able to write data anywhere, and the top priority was to have them running again. So we took the risk and brought the file system online so it could start collecting data again. We put network quality of service (QoS) policies in place to throttle any analysis jobs (NFS reads) from overwhelming the file system. We were running severely degraded, with each storage controller rebuilding the maximum of six logical units simultaneously, albeit quite slowly. We knew this was going to take one to two weeks to complete. We contemplated bringing other storage online in place to collect the partial writes and then moving things over, but a lot of things are more ingrained that they ought to be. This would have been a time-consuming job with a lot of logistics and double changes followed by data migration, and time was not exactly an abundant resource.

Next we targeted the MAID RAID-6 stripes. There is an undocumented command that can be used, under strict vendor supervision, to mark a 10-disk stripe unit as okay as long as you are absolutely positive that nothing has been changed. If you use this command and are wrong, the consequences are rather dire. So, like Henry V unto the breach, we charged, marking all of them as good! Except there was a problem. The vendor informed us that, according to the data parity logs, something *had* been written there! We pondered this for a while. There couldn't have been anything written to there, could there? After some thought we concluded that what happened was that during the emergency shutdown initiated by the third shelf loss

(see OUCH, above), the Linux nodes tried to shut down the file system gracefully, instead of just powering off the hosts immediately. This initiates a GPFS shutdown which will flush any superblock updates and other bookkeeping data.

We theorized that the master RAID-6 block records on the MAID disks got updated during this event. Luckily, there's an undocumented way to fix this as well. The vendor had us run several commands that generated 128,000 rows of data about disk blocks on the suspected LUNs for each storage controller. They analyzed them and gave us a series of commands to run to regenerate the parity for each of these blocks. This is basically the equivalent of telling it to rebuild the parity on this disk from all of the others, and just assume that everything is good. With a little help from Expect [2], this took another couple of hours. Finally, we could mark the stripes as OK and that left just ~30 LUNs to rebuild the hard way. But we could do this while the data continued to roll in and out.

So we continued to operate, QoS throttles in place (adjusted as needed), partially degraded, and still hadn't run the fsck, yet. We waited for the full rebuild to complete, which took five days for the first disk in each RAID-6 group, and another five days for the second disk. A very large sigh was had after the first five days concluded and all LUNs were only degraded by one disk each. Ideally, the storage controller would be a little bit smarter about this and build both RAID-6 parity blocks simultaneously to economize the presence of data already in memory. Actually, this should be required in a disk storage architecture, but I know there are many instances out there of similar behavior.

Finally, all of the RAID parity had been rebuilt; it was time to run the fsck, just to make sure everything was okay. So, we started an online fsck. After about 10 hours of running, it reported that it had encountered a series of errors requiring an offline fsck! This was not heartening. A number of things about this were bad:

1. It was somewhere in phase two of multiple undisclosed phases.
2. The fsck had no estimate of how much time was left to go. We had no bounds of what to expect from an offline fsck and how much downtime would be required.
3. We'd had enough downtime.
4. The messages pointed to some kind of a mismatch in a file that appeared to be perfectly fine when examined by hand.

So we contacted the vendor again. They suggested we run it in offline mode, as indicated. "@$(&% no," we said. We can't afford to be down for an undisclosed number of further days. What does this error mean? They had to consult with IBM about the message that we received. The next day we had an answer. It turned out that this error was an internal conflict of some kind. They suggested that we update to the next release of GPFS and try again, since it was an identified bug. We did this. Rolling upgrades in GPFS are fairly easy and require no downtime. We started the fsck again and the error was gone! The fsck ran successfully and without incident; it took 18 hours. Where did this leave us? Unfortunately, we were still without mirrored metadata. However, as I write this article that situation has finally been corrected by the installation of a larger flash media device pair capable of holding all metadata with room to grow. (The mirroring to the new devices took less than a day!)

Ponderables

- Explore all failure modes of your storage. Some may be surprising.
- Pay attention to your hardware architecture. Play "what if" scenarios.
- Insist on on-site spares for components that can leave you in severely degraded states. In our case, we had spares of disks, I/O modules, and power supplies, but the most serious loss is when a storage controller head gets lost after losing an I/O module. We did not have this spare. (We do now!)
- Demand explanation on error messages that you don't understand, that appear to be contradictory, or that seem patently false.
- Never defer label maintenance on critical systems!
  - Correct labels are imperative in a crisis.
  - Don't trust that labels are correct when getting it wrong could mean catastrophe. If the person powering down the shelves under this stressful condition had used a sanity check, he would have realized that he was power cycling the wrong storage shelf.
- Router/switch-based throttles and QoS are useful for asset protection. Use them to limit maximum bandwidth to clusters when needed.
- Sometimes it's better to keep the full implications of how close one has been to disaster close to the vest until the worst is over. Worrying about things that are outside of one's control can cause a lot of stress. On the other hand, some management groups insist on knowing the worst case scenario, so withholding in those situations can be "bad for your career." This is a tough call. I simply told management that things were quite dire, while not sharing that it was a 10 out of 10 sort of situation until afterwards.
- At some point, most people need to make a call about whether to leave storage online in degraded mode while the problem is being debugged—so people can get work done—or to take it offline if it is safer to do so. These are always excruciatingly hard decisions. I have decided in each direction at different times. May you never have to make this choice in your career! This is a decision where management backing is very helpful, but not always available (weekends, holidays, vacations, the middle of the night, etc.).
- You've tested your backups recently, right? You have, HAVEN'T YOU!?
- Is your tape data something that can be restored from easily?

Finally, the trials were over. Feeling some empathy for Job [3], we heaved a collective sigh of relief. Looking back over the whole series of unfortunate events (with apologies to Lemony Snicket), it seems highly improbable that they were unrelated, but they were. One is tempted to think of things like localized power disturbances, or other things, but some of the incidents were in completely different buildings, involving an entirely different electrical service substation from the utility, and different power sources and high voltage feeders.

One final note on backups and restores. During the heart of the incident, one member of the team was working on restoring the most recent working set of data—the last two months—to an alternate server in case the worst happened and we had to zero out the entire file system from scratch and restore from tape. This was a frighteningly real scenario, so he was testing out the most optimal way to get the data from tape. This is one of those cases where I'm sure most of us have read advice about testing our restores, and most of us back-burner this because there always seem to be higher-priority things. You've heard it before, you'll hear it again:

go do it. If you need a full day, show this horror story to your management and get them to hold off the customers while you complete this important task.

We're still testing various alternatives to make retrieval of the most recent data as efficient as possible, which means as few tapes as possible. We are currently at a 2:1 overhead. This means to retrieve N TB of data, we need 2*N tapes, which isn't terrible, but definitely has room for improvement.

May your transceivers be well-matched, your RAID controllers have sufficient redundancy, your journals be recoverable, your power be reliable, your flash devices be well behaved, and your users be tolerant and patient. I'd say "may your restores be fast and optimal," but I think you'll better appreciate the sentiment of: may your tapes be write-only.

References

[1] MAID: http://en.wikipedia.org/wiki/Massive_array_of_idle_disks.

[2] Expect: http://www.nist.gov/el/msid/expect.cfm.

[3] Job: http://en.wikipedia.org/wiki/Book_of_Job.

[4] RAID-6 is dual-parity, allowing the failure of three disks before data is lost.

[5] LUN: a logical partition of a disk or multiple disks to provide aggregation and/or partitioning that can be used with permissions models to enable or restrict access to storage. http://en.wikipedia.org/wiki/Logical_Unit_Number.

For part 1 of this article, see the June 2012 *;login:* page at https://www.usenix.org/publications/login/june-2012-volume-37-number-3.