isoho.st incident report – 2013/05/29

EXECUTIVE SUMMARY

Soon after 07:00 on 20 May, our support team were informed about a degradation in storage performance by one of our customers. Specifically, certain I/O workloads were experiencing high latency.

Upon investigation, it was confirmed that this was a general issue affecting all isoho.st customers and we proceeded to trouble-shoot the components of our infrastructure. Without having identified an obvious cause, we deemed the problem to be of high enough severity to require immediate action, and thus began making unscheduled emergency changes to isolate the problem and eliminate potential causes.

A partial solution was implemented by close of business on the same day as the initial report, with the reporting customer indicating that the changes had resulted in dramatically improved performance. Unfortunately, these early improvements only served to alleviate symptoms, thus masking the real problem without addressing the primary causal issue.

Ultimately, the problem was due to relatively minor (when considered in isolation) software bugs and limitations in three individual core upstream software components of our infrastructure (namely the Linux kernel, librbd and the QEMU virtual machine emulator) that under certain workloads severely impacted on cluster wide latency. The subtle interactions between these components under these conditions conspired to make fault finding challenging, because the problems existing in higher layer components served only to amplify the effect of the core bug in a lower layer system component which by itself was almost imperceptible when tested in isolation.

Continual improvements were made to the systemic I/O performance of our storage cluster on an almost daily basis with a final resolution being rolled out on the morning of 27 May.

ROOT CAUSE ANALYSIS

The root cause was the “stable pages” regression published in a December 5, 2012 Linux Weekly News article (http://lwn.net/Articles/528031/). This Linux kernel performance regression has a particularly severe impact on XFS write latency, the underlying file system utilised by our back end storage engine. This problem in and of itself, however, would not have been sufficient to cause the particular issue experienced by our customers had it not been for a simultaneous limitation in librbd, which provides the block device interface backing the virtual disks provided to customer VMs. The issue with librbd was that cache flush operations were performed synchronously, which coupled with the way that QEMU works meant that entire customer VM processes would stall on I/O while higher than usual latencies caused by the “stable pages” regression were amplified during cache write back.

DETAILED RECORD OF EVENTS

Monday: After receiving the initial fault report, our support team immediately set about in an attempt to reproduce the problem in a test VM running on our production infrastructure. The nature of the problem meant that it was difficult to find a representative work load that also exhibited the problem, but by late morning, the fault condition could be reliably reproduced in a test VM and our team set about trying to diagnose the cause of the problem.

Initially a configuration issue was suspected when when it was discovered that the XFS file systems backing our storage engine were operating using the default *atime* mount option. In our architecture, this effectively promoted every disk read operation into a write, however, while this configuration oversight was certainly not ideal for throughput, it should not have caused the kinds of I/O latencies that were being observed. The I/O scheduler of the cluster hosts was also switched to the "deadline" implementation, a configuration that is arguably more appropriate to our workload.

The process of safely dismounting and remounting the backing file systems while not incurring any down time for customer VMs requires that each cluster node be taken down in turn, forcing their backing stores out of synchronisation with the rest of the cluster. The resynchronisation process must be allowed to complete before proceeding to each subsequent node, but it too was taking longer than expected due to the underlying I/O condition. Nevertheless, by late afternoon, all cluster nodes had been successfully reconfigured and performance had improved dramatically.

In retrospect, remounting these file systems in *noatime* mode made the improvement that it did because the additional write load on the Linux kernel was significantly reduced, thus reducing the surface area for triggering the actual underlying "stable pages" problem.

Tuesday: After further testing in collaboration with the reporting customer it was discovered that the latency issue had stubbornly remained, although it was now much harder to trigger. Additionally, around midday we received yet another report of high latencies from a second customer.

Considerable time was then spent benchmarking I/O performance at various layers of our architectural stack. Initially we were only able to reproduce the latency experienced at the VM layer at the XFS file system layer on one of our cluster hosts, at which point we began to suspect some kind of hardware issue was at play. In order to eliminate this particular host's hardware from contention, the host was taken out of service, however, after considerable time spent replicating that host's data to other cluster nodes, the problem simply manifested itself again on another cluster host.

Further testing under various workloads eventually revealed that the problem could be reproduced at the XFS layer on any of the cluster hosts. Thus, barring any hardware batch issues at play, the issue was most certainly software, likely a Linux kernel issue.

The downed host was brought back into service and our team continued to research potential causes while the cluster was again left to re-balance storage over night.

Wednesday: By now it was clear that the Linux kernel was at issue and further research revealed the most likely culprit was the the "stable pages" regression mentioned in the root cause analysis above.

The morning was spent reverting the "stable pages" patch that was introduced in the 3.0 version of the Linux kernel on each cluster node. This was a time consuming process requiring live migration of running customer VMs off of each host prior to rebooting, followed by the same storage resynchronisation process described earlier before migrating the VMs back.

Thankfully, the effort paid dividends and I/O performance at the XFS layer appeared to be restored to acceptable levels, however, further testing revealed that while the issue had been largely resolved, some small, but unacceptable, and infrequent I/O latencies were still noticeable at the VM layer.

Thursday: By this stage we were confident (erroneously, due to the multiple contributory factors at play) that we had isolated the problem to a Linux kernel issue, and so after more testing we proceeded to upgrade to the 3.9 version of the kernel on all hosts, which implements a different approach to resolving the “stable pages” issue. We also tried several other kernel builds experimenting with various configuration options that may affect I/O latency. Unfortunately, nothing we tried improved matters at the cost yet more time in terms of testing, reboots, VM migrations and storage resynchronisation.

Friday: Further research revealed that even acceptable underlying file system latencies could result in large perceptible VM stalls when write back caching is enabled, due to the fact that librbd's cache flushing mechanism was synchronous (http://tracker.ceph.com/issues/3737).

As a low impact work around, we disabled write back caching for all customer VMs bringing overall latencies back to acceptable levels. At this stage, the QEMU interface to librbd was still synchronous, but VM stalls were now rare occurrences and of very short duration because the underlying “stable pages” issue had been resolved and it was no longer possible to accumulate a large batch of cached writes to flush.

Weekend: Over the course of the weekend we deployed the latest version of librbd, which has asynchronous flush support, in a test environment with a patched version of QEMU that utilises this new interface. After satisfying ourselves with respect to the stability of the updates, we rolled out the changes to customers on Monday and re-enabled write back caching for all VMs.

SERVICE IMPROVEMENT STRATEGY

Our post-mortem analysis of the fault has identified the following weaknesses in our process:

a) The pre-commissioning performance test of our platform did not simulate the particular I/O load pattern that caused this particular problem to manifest.

b) Trial-and-error troubleshooting on the production cluster without the luxury of downtime for running (albeit degraded) services implied significant delays to resolution.

These issues with our process will be addressed as a matter of urgency, however, we feel it is useful to highlight the following positive outcomes:

a) During the course of solving this problem we made changes to configurations that more than doubled our I/O throughput, meaning we are now able to offer storage performance that is better than our competitors, while still maintaining redundancy and a distributed architecture.

b) The nature of the problem is now understood, a series of software bugs that have now been fixed, meaning that re-occurrence of this particular issue is exceedingly unlikely.

c) We have re-affirmed that our software stack technology choices are sound, and can continue to scale to meet the needs of our customers.

REQUIRED CUSTOMER ACTION

The upgrades to the QEMU machine emulator require customer VMs to be power cycled in order to take advantage of the changes. This should be done at the customer's earliest convenience via the

SSH admin interface. Note that the VM must be “powered down” and then back up again, as a reboot does not restart the actual underlying QEMU process.

It is also beneficial for customers to switch to the “noop” I/O scheduler within their VMs, as this further provides a significant I/O performance boost on our architecture.