Main page
Recent changes
Server admin log (Prod)
Server admin log (RelEng)
Deployments
SRE/Operations Help
Incident status

Cloud VPS & Toolforge
Cloud VPS documentation
Toolforge documentation
Request Cloud VPS project
Server admin log (Cloud VPS)

Tools
What links here
Related changes
Special pages
Permanent link
Page information
Cite this page

Print/export
Create a book
Download as PDF
Printable version

Incident documentation/20140806-Upload

< Incident documentation

Contents [hide]

Summary

From approximately 2014-08-06T06:00:00Z to 2014-08-07T06:36:00Z (start time is fuzzy as problem gradually ramped in), serving of some fraction of the images from upload.wikimedia.org was impacted (slow downloads, later actual failures) by an issue with our Varnish cache servers, especially for the two US datacenters.

Timeline

- Some maintenance work (wiping Varnish caches over a days-long schedule to minimize impact) began on 2014-08-05.
- First indication of a growing issue was a report of failed puppet runs on machines in the ulsfo datacenter in the early hours of 2014-08-07. These puppet failures noted that the machines had difficulty downloading package updates from cabon.wikimedia.org in the eqiad datacenter, but nothing conclusive or service-impacting was noted. The initial investigation was not very deep; nobody was actively on-hours and looking into it, and there weren't any related reports of anything service-impacting.
- A few hours later, as the puppet failures continued, it was finally noticed that the immediate cause was bandwidth saturation on a network link between ulsfo and eqiad.
- At this point we started suspecting the Varnish maintenance process (which was paused hours earlier) could be involved, as the refilling of wiped upload caches could be a significant contributor to the link saturation.
- After some brief investigation on this, it was decided to take the interim step of re-routing (at the DNS layer) upload.wikimedia.org traffic that would normally be bound for ulsfo over to eqiad to alleviate the bandwidth constraint, and that improved the situation somewhat.
- Shortly afterwards, it was discovered that even eqiad was not serving upload image traffic reliably from the Varnish cache. Investigation of this revealed that 2/8 of the upload caches (those that had already been hit in the maintenance wipe process) in eqiad were running with tiny caches (~3GB instead of ~300GB), causing most traffic to go through uncached and overloading the backend image services, causing service failures even from eqiad. This explains the bandwidth saturation from ulsfo more-completely as well, as the affected fraction of upload caches everywhere had similarly-reduced cache sizes (although esams was, by chance, less-affected as its cache sizes were only reduced to ~42GB).
- The cache sizes were corrected on those two machines and service was restored to its normal working state from an end-user perspective, although much work remained over the next two days to address deeper issues, bring our global cache efficiency back up to normal levels, and restore normal DNS-based traffic routing of some requests back to ulsfo.

Conclusions

- From a technical perspective, the direct reason for both the bandwidth saturation and partial unavailability of images was that as caches were being wiped administratively, they were being recreated much smaller than before; too small to be effective in their role as caches. Our upload.wikimedia.org traffic (much more-so than other cached services) is reliant on persistent Varnish disk-cache efficiency to handle our normal production loads at all.
  - The general process for wiping a cache is to shut down the varnish backend process, delete the cache file, and then restart varnish and let it create a new cache file.

    - However, even when given correct commandline arguments for the size of the cache files to be created, Varnish's behavior is to explicitly check available disk space on the filesystem the cache file is to be created on, and if the requested size does not appear to fit, the requested size is reduced to fit the available space. Arguably this is a mis-feature, especially given that the action taken (drastic reduction of requested size) is not logged to syslog.
    - The filesystems the cache files live on are XFS, and XFS has an issue (it's debatable whether it's a bug; it's probably an intentional performance optimization) where when a very large file is deleted from a filesystem (such as our 300GB giant cache file on a 304GB filesystem), it can take several minutes for the free space on the filesystem to be adjusted for the removal.
    - The combination of these two mis-features/bugs is that if our Varnish caches are restarted immediately after deleting a very large cache file, the cache file is silently created with a much smaller size than requested.
- From a less-technical perspective, this could have been prevented by either of two things:
    - Monitoring gave little-to-no indication of the issue at hand as it progressed, other than the indirect reported failures of puppet downloading package updates over a saturated WAN link.
    - The human executing the cache wipes should have taken a more-thorough look at the results of the process, at least on the first several machines of the ~100 involved in the process, even though no production issue was apparent.

Actionables

- Status: 🟧 **on-going** - As with most incidents, exercising more caution with major operations work would be a good idea.
- Status: 🟧 **on-going** - Audit the current cache sizes (filesystem size, configured file size, actual file size) on all Varnish caches and fix outstanding issues there. It was noted during the outage-recovery work that a handful of our caches were already running reduced sizes from isolated incidents of the same behavior in the distant past, but it has gone unnoticed. Additionally, some of our cache sizes are explicitly configured incorrectly in puppet, but the mistake is being covered up by the automatic resize-to-disk-space behavior.
- Status: ⬜ **Declined** - Fix our custom-patched Varnish implementation to always fail to start if it cannot allocate the requested cache size, so that such issues are more-immediately apparent.
- Status: ⬜ **Declined** - Investigate the XFS angle more-deeply. Is there a way (mount flag? newer kernel implementation?) to get it to report accurate free disk space more-quickly? If not, should we investigate whether other filesystems that lack this issue can give us comparable performance? Even with a fixed Varnish, future cache wipes will be problematic as long as this issue persists. I have a short, temporary shellscript for the wipe-restart process now which seems to reliably work around it (and will be used to complete the ongoing maintenance from the start of this issue), but it's needlessly complicated and adds significant cache uptime delays to the wipe-restart process.
- Status: ⬜ **Declined** - Add more Varnish-related icinga checks? This needs some investigation as to what the best metrics would have been to catch this. Most-direct would be a check for whether cache filesystems are emptier than expected, but this could become annoying during installation and maintenance. Another that would have pointed us in the right direction more-quickly (and may be a good metric for many other future failure scenarios) is alerts on hitting backend connection limits, etc.
- Status: ⬜ **Declined** - Monitor link bandwidth? At least for our explicit inter-datacenter links with known bandwidth limits, it would be good to have alerts when we're at or near saturating the links.

Category: Incident documentation