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/20200204-app server latency

< Incident documentation

document status: in-review

Contents [hide]

Summary

Babel is a MediaWiki extension that displays infoboxes on user pages, expressing the user's proficiency in speaking one or more languages. When looking up language data for a user, it first consults the **WANObjectCache**, backed by **memcached**. If that user's data isn't in the cache, the Babel extension makes an API call to itself (that is, from Babel on an appserver to Babel on an API server), which fetches the language data from the database and caches it for next time. The cache hit ratio is normally about 90%.

From 16:03 to 16:12 UTC on 2020-02-04, the WANObjectCache hit ratio for Babel keys dropped, bottoming at about 52%. (graph) The total rate of cache lookup attempts -- that is, hits plus misses -- remained roughly constant at about 10,000 per minute (graph), suggesting that either the Babel keys were suddenly evicted from the cache and had to be repopulated, or Babel traffic suddenly shifted to a new set of keys that was not yet cached. (TODO: Determine which, and clarify.)

Because of the increased cache misses, the rate of Babel API requests also increased. These requests were sent over HTTPS (CommonSettings.php) so the extra computation due to TLS caused high CPU load on the appservers (graph). Each request hung for ten seconds waiting for a response before timing out (graph), which tied up appserver resources and delayed other traffic. Appserver errors and latency recovered immediately when the Babel cache hit ratio returned to normal, but a Kartotherian outage, which may have been triggered by the same spike in memcached misses, continued for some time.

It seems that the origin of the incident is a further slowdown of the appservers in response to a surge in traffic that happened around 16:01 (graph). While the surge seems relatively small, it corresponds with the initial slowdown of application servers. TODO: Supplement graph links with inline static images.

Impact

About 12 million varnish-text requests were dropped or delayed in a ten-minute period, spanning all five caching data centers; that represents about 20% of traffic in that window.

TODO: Fill in the methodology behind that number, which is based on eyeballing the missing area under global text cache request rate (graph).

Detection

The first Icinga CRITICAL alert in #wikimedia-operations was at 16:03 (for "phpfpm_up reduced availability"). The first page was at 16:05 ("LVS HTTPS IPv6 #page on text-lb-ulsfo-wikimeda.org_ipv6 is CRITICAL: CRITICAL - Socket timeout after 10 seconds"). In all, we received 239 IRC-only alerts and 7 pages. (These numbers reflect only the MediaWiki latency issue, not the Kartotherian issue.)

All seven paging alerts were of the form "LVS HTTPS IPv{4,6} #page on text-lb.{DATACENTER}.wikimedia.org_ipv{4,6} is CRITICAL: CRITICAL - Socket timeout after 10 seconds" across all five caching data centers.

Timeline

All times in UTC, 2020-02-04.

- 15:38 General appserver slowdown begins. 75th percentile latency increases from the usual ~300 ms to 1500-4000 ms. During this slowdown interval, users perceive slower load times but the site is fully available; error rates are not elevated. (graph). This is NOT considered as part of the main outage, but rather a degradation related to a spike in memcached requests.
- 16:01 The outage begins on the appservers, with a spike of slow requests, in response to what looks like a slight surge in external requests.
- 16:03 Babel cache misses begin to spike (timestamp based on Babel requests in API server logs). Appserver errors and latency immediately spike. **OUTAGE BEGINS**
- 16:03:31 First Icinga CRITICAL alert in #wikimedia-operations. **OUTAGE DETECTED**
- 16:05:12 First Icinga #page.
- 16:12 Babel cache misses return to normal (timestamp based on Babel requests in API server logs). Appserver errors immediately recover; latency returns to slowdown levels. **OUTAGE ENDS**
- 16:19 First alert related to Kartotherian. Kartotherian issues continued for some time after this outage, and won't be discussed here – see 20200204-maps.
- 16:52 ladsgroup deploys #570084, reducing the overall rate of Babel calls. (SAL)
- 17:10 ladsgroup deploys #570089, lowering the timeout from 10 seconds to 2 seconds. (SAL)
- 17:34 Joe increases weight on mw12[3-5].* to 15 (SAL)
- 17:42 General appserver slowdown ends. 75th percentile latency returns to normal.

Conclusions

What weaknesses did we learn about and how can we address them?

The following sub-sections should have a couple brief bullet points each.

What went well?

- *for example: automated monitoring detected the incident, outage was root-caused quickly, etc*
- Incident detection was fast and reliable. The hard slowdown began at 16:01, we got paged within 4 minutes.

What went poorly?

- *for example: documentation on the affected service was unhelpful, communication difficulties, etc*
- It's not obvious what caused the outage from any of our multitude of dashboards. It was impossible to determine during the ongoing outage either
- Because of the lots of interdependencies between the appservers and the api clusters, if the api slows down, we have a backscatter on the mediawiki side.
- If appservers are slow, we seem to have more cache misses in babel, or better we seem not to cache the value. This created a cascading effect where appservers and api clusters will start being too slow, and cause slowness to one another.
- php-fpm metrics aren't available during the outage for the most part, as we collect them from the php-fpm server and it won't respond when fully overloaded.

Where did we get lucky?

- Nothing to report I guess.

How many people were involved in the remediation?

10 WMF SREs, 1 WMF software engineer, and 2 WMDE software engineers troubleshooting the issue; plus 1 WMF SRE incident coordinator.

Links to relevant documentation

https://wikitech.wikimedia.org/wiki/Application_servers/Runbook

Actionables

TODO: This is an incomplete list of actionables, not all actionables have tasks, and not all tasks are tagged "incident."

- Eliminate all Mediawiki appserver self-calls over HTTPS. Short-term, move them to HTTP; longer-term, to Envoy or similar. https://phabricator.wikimedia.org/T244843
- Make timeouts on http calls short enough not to cause a cascading outage when one cluster is slow. Done for Babel.
- Implement some automated way of sharing incident docs with WMDE phab:T244395
- MediaWiki's HTTP call abstraction should pass through the X-Wikimedia-Debug header if it was set in the original request
- Fix a bad interaction where Wikidata/CachingPropertyInfoLookups don't actually cache data in WANObjectCache, leading to many repeated calls for the same memcached key on the same server. phab:T243955 Done.
- Reduce read pressure on memcached servers by adding a machine-local Memcache instance phab:T244340
- Investigate and propose options on getting structured data from API and application server logs, to improve observability of exactly what's expensive on appservers. phab:T235773
- Investigate why a slowdown in response times from the API causes a surge in cache misses for Babel data in Wikibase. See https://phabricator.wikimedia.org/T244877

Categories: Incident documentation | Incident documentation drafts