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/20190802-api-overload-parsoid

< Incident documentation

document status: final

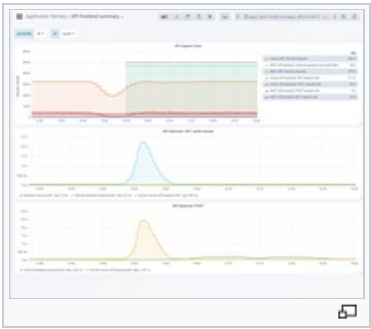
Contents [hide]

Summary

The api-appserver cluster has been overwhelmed with expensive parsoid-batch calls from particular pages, resulting in brownouts and frontend availability at ~97%.

Impact

The mediawiki API has suffered the most impact, with internal services and external API users impacted as well. See API frontend graph, the slowdown was severe enough that a depression in request rates was observed.



Detection

Alerts for appservers "high CPU" went off on IRC during brownouts through the first half of the day (6-12 UTC) followed by general widespread alerts for affected internal systems (restbase, recommendation, mobileapps alerts went off) starting around 15 UTC

Timeline

All times in UTC.

- 7:00 / 9:30 / 12:30 / 13.00 High CPU load API appservers alerts and recoveries for brownouts, no widespread impact
- 13:25 investigation on the general issue begins
- 13:45 weights of mw12[23].* found not to be aligned with the rest of the fleet and fixed
- 14:50 High CPU load on API appservers start coming in again, this time more severe and affecting other services
- 14:54 Low HTTP availability alerts fire, together with recommendation_api, restbase, mobileapps
- 14:55 Investigation on the ongoing incident begins
- 15:02 Recoveries start coming in by themselves, and investigation continues
- 15:34 Offending page identified as one on itwiki using Lua, and requested by Google's rest api crawler

Conclusions

API appservers are susceptible to storms of many long-running parsoid-batch requests, in particular external

requests which end up exhausting the cluster's resources and affecting all API users.

What went well?

An outage condition was identified quickly, and investigation began quickly.

What went poorly?

- Identifying the expensive API calls could have been faster.
- Unbounded parallelism of expensive API calls must be limited, both in rate and concurrency. Internal update requests like this from Change-Prop are both rate-limited and blacklisted automatically if the rate of errors is too high, but external requests are not.
- I (Ppchelko) assume that Varnish would attempt to retry a request on timeout, which in turn multiplies the load.

Where did we get lucky?

The cluster recovered by itself once requests ended and recovery has been fast.

Links to relevant documentation

Where is the documentation that someone responding to this alert should have (runbook, plus supporting docs). If that documentation does not exist, there should be an action item to create it.

Actionables

Explicit next steps to prevent this from happening again as much as possible, with Phabricator tasks linked for every step.

NOTE: Please add the #wikimedia-incident Phabricator project to these follow-up tasks and move them to the "follow-up/actionable" column.

- Limit external requests concurrency/parallelism (bug T167906)
- Icinga alerts for appserver high CPU should be less spammy (bug T228878)
- Bigger idea: it would be nice if there was, aggregated across the whole cluster, some report available of what request types, users, user-agents, etc were using the most appserver CPU.
- …

Category: Incident documentation