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/20150424-Parsoid

< Incident documentation

Contents [hide]

Summary

On April 24, 2015, Friday, serious corruptions were noticed from Visual Editor saves and it was determined that this was arising from newly deployed Parsoid code Thursday night. That deploy was reverted around 2:50 pm CT.

Timeline

All times in CT

- ~1:45 am Ori deployed fresh code to the Parsoid cluster. This was meant to be an emergency deploy of Tim Starling's patch in response to 20150423-Commons, but unintentionally deployed all recent changes from Parsoid master.
- ~10:45 am I (Subbu) noticed a weird preview bug that rendered a preview from frwiki for a template addition on mediawiki wiki. James and I were baffled, but we put it away as some weird transient bug since neither of us could reproduce it again.
- ~1:20 pm Luis Villa filed a weird corruption bug on metawiki.
- ~1:37 pm Alex Monk encountered more corruptions (via Review changes in VE) on wikitechwiki.
- ???? James also was able to reproduce corruptions on different wikis.
- ~2:30 pm James escalates the matter and brings this to the notice of Subbu and Gabriel and it was determined, based on corruptions being seen on private wikis (that doesn't use RESTBase), that this was a Parsoid-specific issue.
- ~2:50 pm Subbu reverted Thursday night's deploy.
- ~2:55 pm James reported that he could confirm that he was not seeing the corruptions anymore.
- 3pm - 6pm Arlo, Gabriel, Scott, and Subbu investigate the deployed patches and after a lot of discussion and hypotheses about what is going on, the source of the problem is traced to a race condition in wiki config initialization. This race condition was introduced by a code refactor and was fixed here.

How many edits were affected?

The Parsoid cluster has 24 cores with about 12 Parsoid workers per core which brings it to 288 workers. Because of the bug, whenever a Parsoid worker started up, it could cache an incorrect wiki API uri for some wiki configs. The incorrect caching requires a race condition to manifest where two (parse/save/preview) requests from two different wikis came back to back before the mediawiki API config response for the first request was finished. Given that config responses are satisified quickly (need config response time data here), this probably did not manifest sufficiently commonly.

In the 13 hour period when this bug was active, the save rate was about 0.245 / sec (is a screenshot helpful / necessary here?). So, in the 13 hours the bug was active, ~40 save requests came to every affected worker. However, even on an affected worker, the bug only affected saves that was to a wiki for which the api URI was corrupted. So, probably a small fraction of the 40 requests were corrupted.

So, while the actual number of corruptions is unclear, it appears that it would be a small number.

The edits themselves are not salvageable. They just need to be reverted.

Could this have been detected earlier?

A number of factors prevented this from being detected earlier:

- The deploy happened late Thu night (early Friday morning) and the fact that additional patches had gone out became clear only Friday morning (~10 am CT).
- No corruptions or bug reports came in all that time. As explained in the previous section, the race condition is subtle enough that most edits just went through fine.
- This bug escaped our testing process and it is unclear at this point, what kind of automated tools might have detected these corruptions.
- It only took an actual reproduction by Krenair and James before we could nail this down clearly to Parsoid and specifically to the code deployed the previous night. At that point, we reverted immediately.

Conclusions

- It turned out (fortuitously) that all through Thursday, we had been fixing various things and had brought roundtrip testing results to a satisfactory state where master was in a "deployable state". So, ideally, nothing bad should have happened even though Thursday's deploy had unintentionally deployed multiple patches instead of cherry-picking. Nothing in any of our tests revealed the race condition that had been introduced in the refactor. That patch had gone through code review as well. So, while the corruption could have been prevented if Thursday's deploy had cherry-picked Tim's patch, we would have deployed that exact same code on Monday, April 27th around 3pm and would have been none the wiser about the race condition till someone noticed a corruption and reported it.
- We also need to figure out how to stress test Parsoid code to detect these kind of bugs.
- Parsoid aspires to be a stateless service. With such noble aspirations, it is important to actually not to have any mutable global state. ParsoidConfig has for a long time been a blot on this. We have had three different bugs (this being the latest) arising from **mutable** global state attached to ParsoidConfig. Early on in Parsoid's development (in the Jan - July 2013 timeframe), we had similar config-confusion-based corruptions (T51411). Most recently, we traced nondeterministic citation ids being assigned because of page-global state maintained in a per-process Cite extension object (T93973, T63165).

Actionables

- We are now considering a different testing mode where we stress test parsoid code under heavy load and concurrency to root out some of these concurrency-based issues. It is unclear at this time what form that will take and how feasible and useful it will be, but that is now on our radar. **Tracked in T98245**
- We decided to eliminate all mutation from Parsoid config that introduced this subtle race condition and are doing additional cleanup to facilitate that. **Now done with this, this, and this.**
- Audit code to verify that all mutable global state is removed from the Parsoid codebase to make it a truly stateless service **Tracked in T93974 to remove another future potential source of mutable global state**.
- We'll have to make sure to clearly highlight Parsoid's deployment workflow so non-Parsoid devs know what patches are considered deployment-safe. **Deployment workflow @ https://wikitech.wikimedia.org/wiki/Parsoid#Deploying_changes and linked from Useful links for Parsoid developers.**

Category: Incident documentation