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/20150423-Commons

< Incident documentation

Contents [hide]

Summary

Wikimedia's MediaWiki API cluster experienced a series of three cascading failures between 03:40 and 04:55 UTC. The failures were caused by the lack of sensible limits at multiple layers of infrastructure on object size, resource utilization, concurrency, and retries.

The failures were set in motion by edits to a set of user sub-pages on Commons. Each of those pages contained a distinct image gallery with over 10,000 images. Based on the title and contents of the pages, they appear to include all images uploaded within a certain date range. The pages appear to have been created as an experiment in creating a counter-vandalism tool.

These pages had been receiving regular updates from administrator tools which spider stale file references. The slow parse log contains dozens of references to these pages each day from April 11 onwards. Commons administrator Magog the Ogre noticed this and alerted the user via a note on the user's talk page.

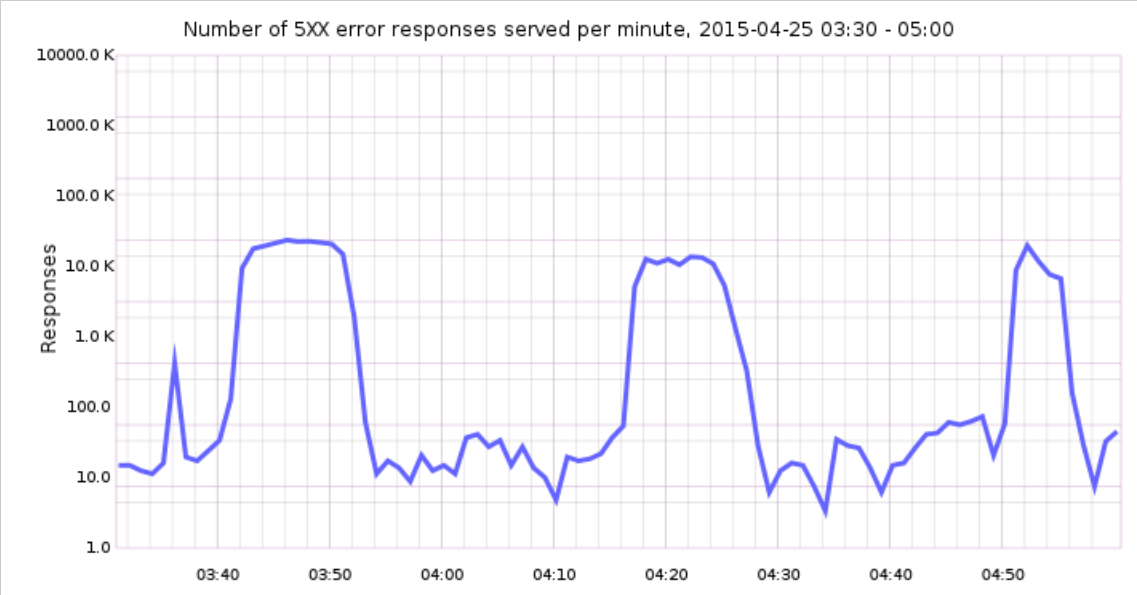
When a page is edited, it must be rendered to HTML by both the traditional PHP parser and Parsoid. We generate and store the Parsoid HTML eagerly rather than on-demand to ensure that VisualEditor (which operates on the Parsoid HTML) loads quickly on any page. Parsoid is invoked to parse the page via an asynchronous job, enqueued by MediaWiki at the time of edit. At the time of writing, two Parsoid parse jobs are enqueued for each edit: one to populate the Parsoid Varnish cache, the other to populate RESTBase.

Parsoid uses the PHP implementations of tag extensions and the wikitext templating functionality via the PHP API. Parsoid's requests to the MediaWiki API are numerous and highly parallel; it is the biggest user of the MediaWiki API in terms of both workload and request count. When one of its requests to the MediaWiki API times out, Parsoid automatically retries it up to 8 times. RESTBase retries Parsoid requests up to 5 times. The the existence of parallel parse operations, automatic retry behavior, and the concurrent firing of HTTP requests, combined with generous (or altogether missing) timeouts means there is no effective ceiling on the amount of work a single user action can generate. The 1:1,000 API request log shows 456 parse requests during or immediately before the outage, suggesting that in total about half a million such requests were made.

To parse edits to the massive gallery tags on Commons, Parsoid performed requests to the `action=parse` and `action=expandtemplates` entry-points of the PHP API. The PHP API in turn requested file metadata for each image included in the gallery. This caused a storm of database queries for file metadata originating in `LocalFile::loadFromDB`. As load on the database servers mounted, queries began to queue.

HHVM was not configured to enforce a time limit on request handling. As a result, HHVM threads were perfectly happy to wait on the database indefinitely, ultimately leading to thread starvation. When the requests failed (whether due to HHVM thread starvation or due to exceeding Parsoid's timeout), Parsoid re-tried them. Because a request to Parsoid results in multiple parallel requests to the MediaWiki API, and because Parsoid enforces a constant timeout on these requests, the overload came in at least three distinct waves, lasting several minutes each.

Graphs



Rate of 5XX error responses served per minute during the outage. The flat peaks suggests metrics infrastructure saturation. The actual rate may have been higher than is indicated.

Tasks

- `I1fa012ca1`: HHVM: Limit wall execution time of FCGI reqs to 290s
- `I8798d51c5`: Exponentially increase the request timeout
- `I1c319f0eb`: Reduce API concurrency and retries
- Task T75960: Implement file usage limits in parsing
- Task T97204: Make sure timeouts are staggered, and there are no retries on timeout from a lower level
- Task T64615: Easy DOS vectors in the API
- Task T97192: Support lowering the API request timeout per request
- Task T97204: RFC: Request timeouts and retries
- Task T97226: Include the request ID in API request logs
- Task T96360: img_metadata queries for Djvu files regularly saturate s4 slaves
- Task T45888: Batch Parsoid's API requests

Conclusions

Service tiering needs to come with clear SLA's and every layer needs to independently report its own health. A global transaction ID to follow requests through layers would be invaluable during complex interactions. This is already set and forwarded by RESTBase, but not yet forwarded by Parsoid for its backend requests.

We should not (publicly) expose API end points that consume unreasonable resources. Server-side timeouts should be set lower than client-side timeouts, and HTTP timeout responses (status `503`) should only be retried if allowed by the server with a `Retry-After` header.

Category: Incident documentation