

Closed

NFS-02 outage

NFS-02 outage

Context

At 00:50 UTC Jul 20 nfs-02 stopped sending metrics and went offline. Last received data was load in the thousands and 100% cpu usage.

Timeline

On date: 2017-07-20

- 00:50 UTC - pagerduty reports nfs-02 is down (start of the slack discussion: <https://gitlab.slack.com/archives/C101F3796/p1500511840864796>)
- 00:52 UTC - @briann and I confirm we can't access the server via ssh, although tcp handshake finishes fine. Azure console reports server is up w/o errors, but cpu usage is 100%.
- 00:54 We tweet status
- 01:03 We discover that ssh connection can get through up to the pubkey authentication, but still timeouts.
- 01:05 We're still seeing logs being sent from nfs-02 to logstash, with OOM invocation against gitaly process.
- 01:11 We decide not to perform hard reboot to avoid potential data corruption, but rather wait for it to stop trashing. Disabling gitaly features has no noticeable effect. At the same time, I (ilya) decide not to open a ticket cause I'm not seeing any Azure-caused effect this time.
- 01:20 We started investigating whether this was related to pushing a lot of tags in gitlab-ee
- 01:31 We're not seeing any logs from nfs-02 for 25 minutes.
- 01:41 We're attempting a restart of the box from Azure console. Another option would be stop/start, which will take 10-15 minites and might cause data corruption.
- 01:55 After several restart requests (all of which were reported as successful) some lonely acpi signal probably got through. Server is rebooted cleanly and we can access it over ssh.
- 01:58 Stan expired caches for nfs-02, we're confirming metrics are up in prometheus.
- 02:05 We start to analyze logs. First findings: 252 OOM invocations for gitaly and git processes from 00:48 to 01:02 UTC.

Root Cause Analysis

Why did this outage occur?

- Why? nfs-file-02 stopped responding for about 50 minutes from 00h50 until 01h40
- Why? The server was unable to handle new incoming requests due to being out-of-memory and CPU.
- Why? About 800 https post-upload-pack requests came in within a 3 minute period. Normally the server would be able to handle these relatively easily, but in this case, the requests took the server down.
- Why? Each post-upload-pack request needed to do more compression than normal since the gitlab-ee repository had recently had a large number of tags deleted and had not been git gc 'ed. If the GC had occurred, the post-upload-pack processes would have not each consumed as many resources as they did?
- Why? Sidekiq was down and therefore the usual git gc processes had not taken place

What went well

- We didn't powercycle the host -- we avoided the potential data corruption and learned that ACPI might still pass through to the kernel long after Azure reported that the action is finished.

What can be improved

- We can limit the gitaly processes from eating all the ram by using cgroups, or containerizing the app in the long run.

Corrective actions

- <https://gitlab.com/gitlab-com/infrastructure/issues/2364>

cc @gl-infra

Edited 2 years ago by Ilya Frolov

Linked issues ⓘ

0

Related merge requests

1

Closed

NFS-02 outage

gitlab-org/gitaly:240

Daniele Valeriani [\[GitLab\]](#) · [@omame-gitlab](#) · 2 years ago

/cc [@andrewn](#) since you took some notes in [gitlab-org/gitaly#403 \(closed\)](#).

Ernst van Nierop · [@ernstvn-gitlab](#) · 2 years ago

Thanks for writing this up [@ilyaf](#) . Can you take a stab at the 5 whys?

Related in terms of process around the outage but not related to the root cause of this outage: we don't seem to have clear runbooks for NFS incidents; made an issue for that at <https://gitlab.com/gitlab-com/infrastructure/issues/2304>

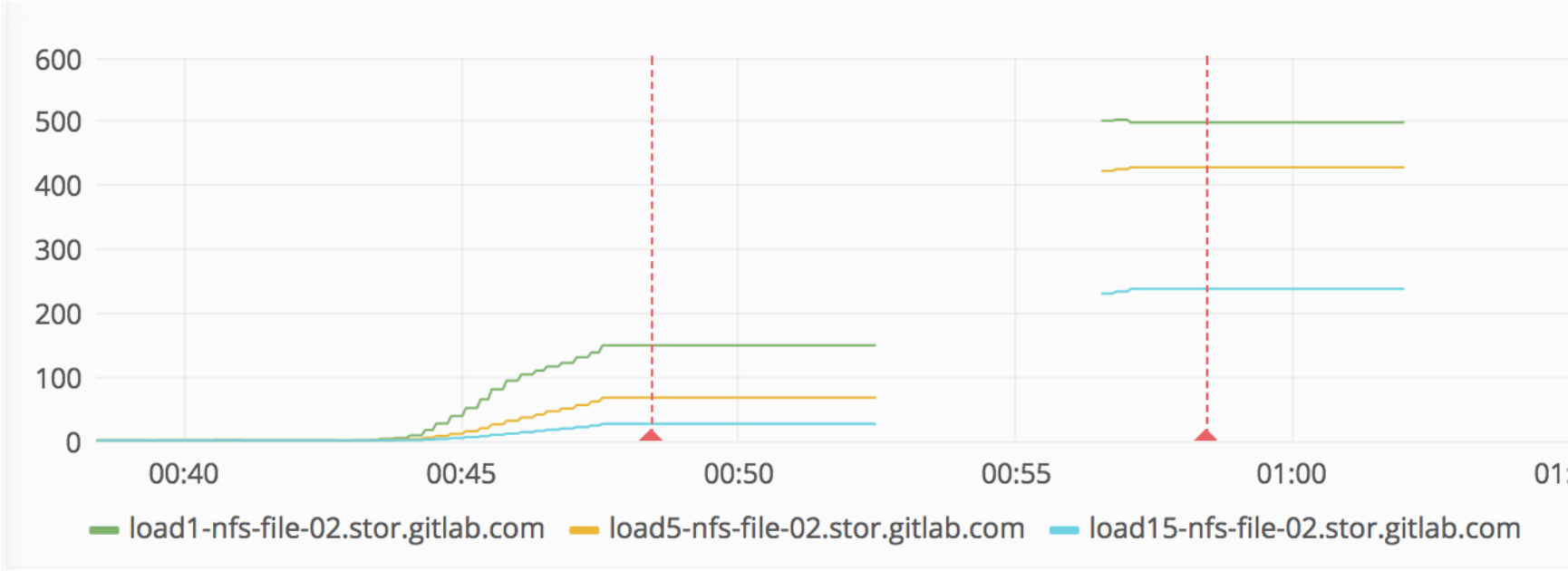
Andrew Newdigate · [@andrewn](#) · 2 years ago

Thanks [@omame](#) , here are my notes....

<https://performance.gitlab.net/dashboard/db/gitaly-nfs-metrics-per-host?refresh=30s&orgId=1&from=1500511104217&to=1500516254706&var-fqdn=nfs-file-02.stor.gitlab.com>

Maintainer

Load started going up at 00:43:17 UTC



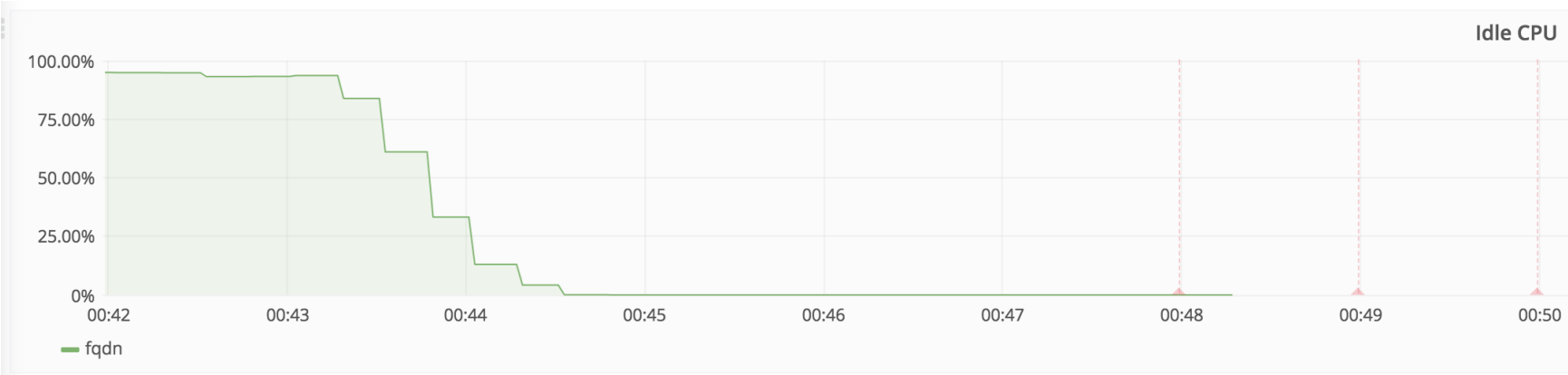
load1 plateaued at 150 at **00:44:35 UTC**

Alerts

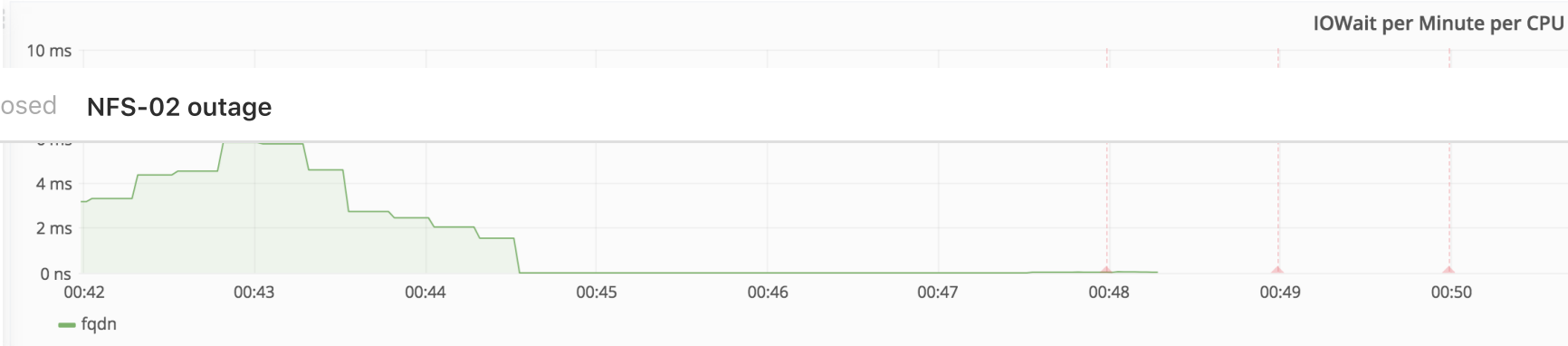
- [GitalyFileServerDown](#) alert fired at **00:49:00 UTC**
- [GitNFSServerDown](#) alert fired at **00:50:00 UTC**

What Caused the Problem?

From the looks of these graphs, it was most likely CPU bound. Idle CPU quickly dropped to zero.



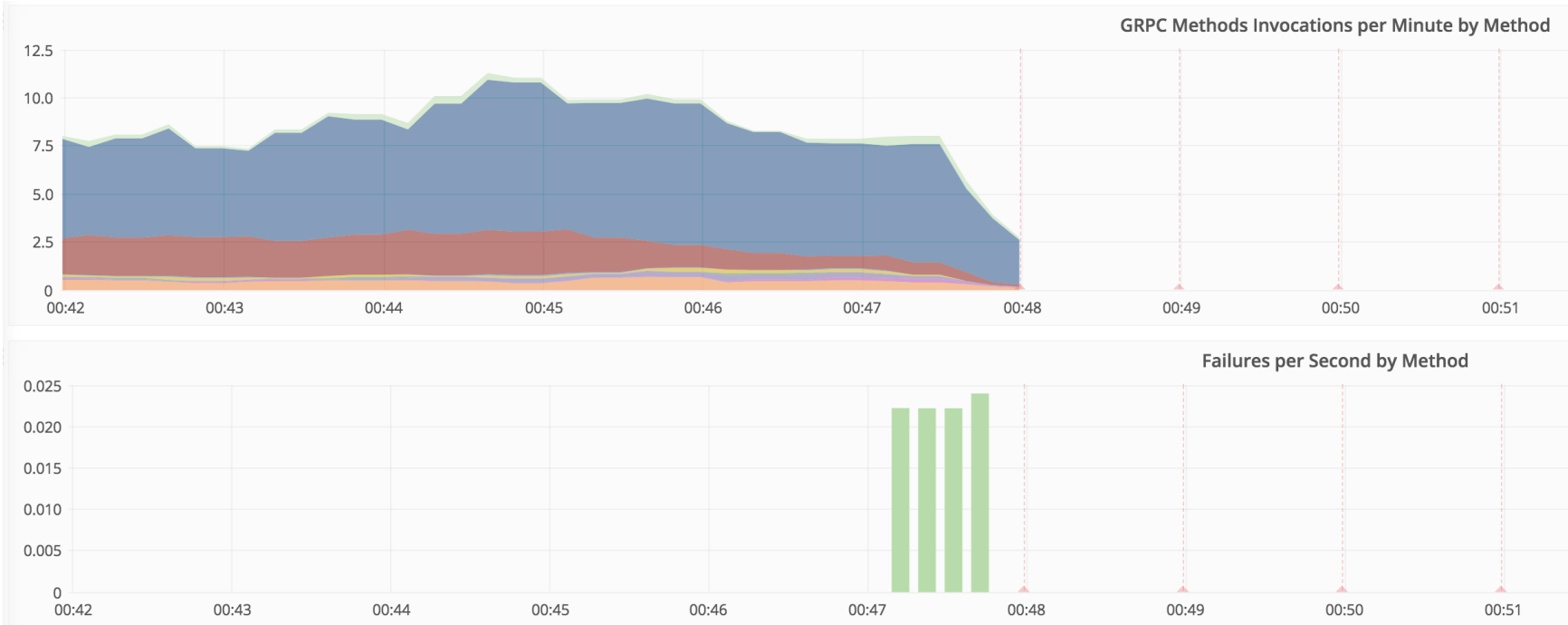
However, IO Wait also dropped to zero, indicating that the process may not have been accessing IO:



Gitaly

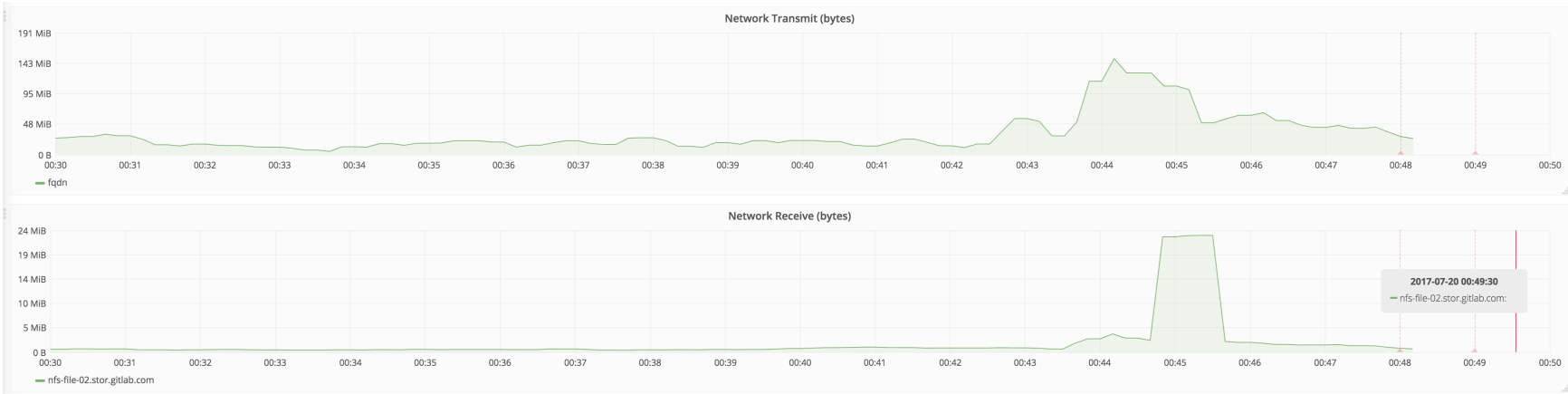
On `nfs-file-02` , Gitaly was handling about ~10 requests per second and this was initially sustained until **00:47:29 UTC** when it rapidly dropped off.

At the same time, a series of `PostUploadPack` method invocations failed, between ~**00:47:00 UTC** and ~**00:47:00 UTC** and ~**00:47:44 UTC**



Network

During the outage period there was locally high network traffic, but well down on the daily peaks



gitlab-org/gitlab-ee theory

One theory is that a push to the `gitlab-org/gitlab-ee` repo by one of the release managers may have invoked a bug in Gitaly.

This push happened just prior to the outage and may have caused a bug in Gitaly.

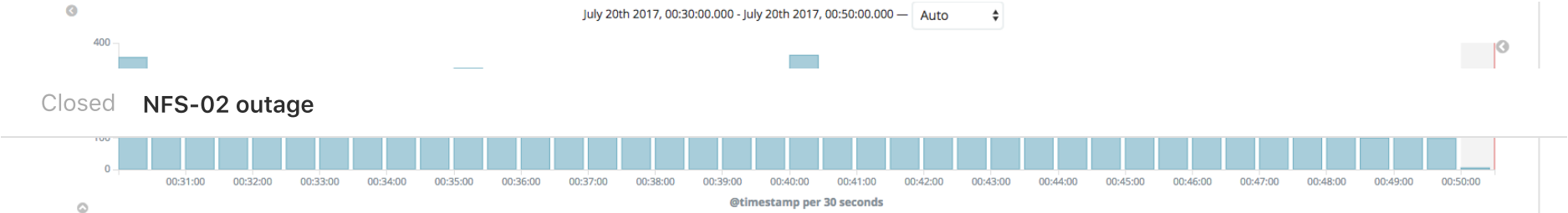
Here are all git https calls to the repo during the period: <https://log.gitlap.com/goto/d4b3a185eb73c22bb8916f9beba4b33a>

Not much stands out, except that there is a clear time at which requests started failing: at **00:47:39 UTC** the last request succeeded, at **00:48:25 UTC** the requests started failing.

The longest request to this repo during the period in question was 174s for a `git-upload-pack (fetch)` at **00:46:47 UTC**: [https://log.gitlap.com/app/kibana#/doc/logstash-*/logstash-2017.07.20/gitlab?id=AV1ddi5NdNoov6mdEVZl&g=\(\)](https://log.gitlap.com/app/kibana#/doc/logstash-*/logstash-2017.07.20/gitlab?id=AV1ddi5NdNoov6mdEVZl&g=()).

This request succeeded.

About 500 git ssh operations per minute were taking place (using Gitlab-Shell, non-Gitaly) during this period, including several git pushes to `gitlab-org/gitlab-ee` .



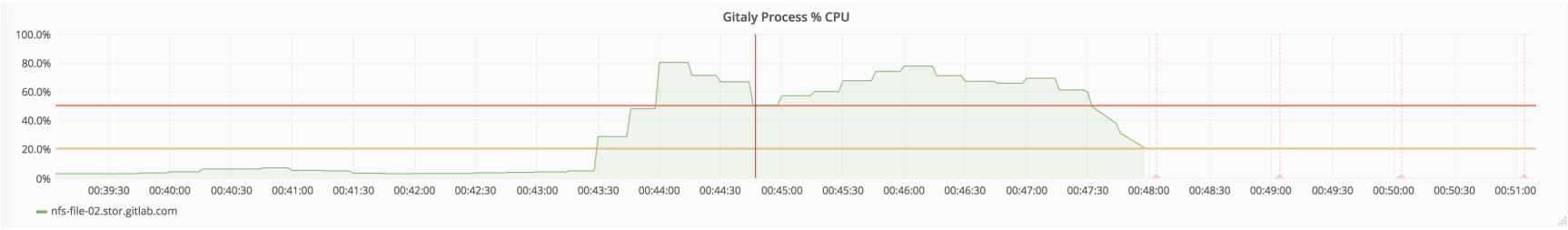
<https://log.gitlab.com/goto/a32d5622195db1ceb4ac2a948b8d1bc3>

Currently it does not seem like Gitaly was involved in any pushes to `gitlab-org/gitlab-ee` during the period in question, so either

- 1. Gitaly was not the problem
- 2. The problem was Gitaly, but involving an operation other than a `git push`

Gitaly CPU

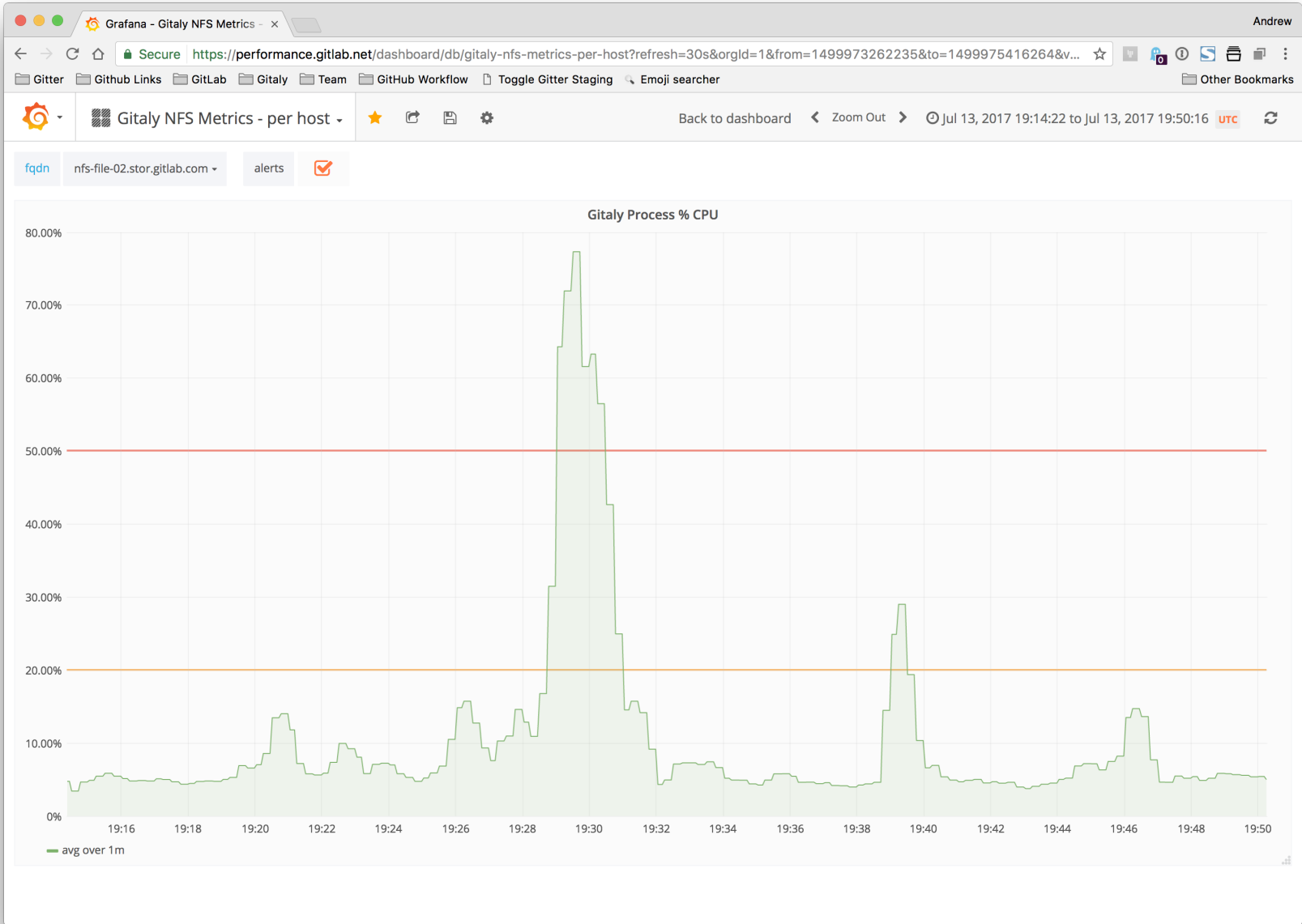
This graph shows CPU usage for the Gitaly process.



<https://performance.gitlab.net/dashboard/db/gitaly-nfs-metrics-per-host?refresh=30s&orgId=1&from=1500511143979&to=1500511870176&var-fqdn=nfs-file-02.stor.gitlab.com&panelId=13&fullscreen>

So, just before the outage, CPU usage went up.

However, relative to the instance size (and number of cores) this CPU usage is not particularly high (100% = 100% of a single core). Additionally, CPU has got this high in the past, on this server, without causing issues. For example:



<https://performance.gitlab.net/dashboard/db/gitaly-nfs-metrics-per-host?refresh=30s&orgId=1&from=1499973262235&to=1499975416264&var-fqdn=nfs-file-02.stor.gitlab.com&panelId=13&fullscreen>

I'm going to continue my investigations....



Ilya Frolov @ilyaf · 2 years ago


I didn't attempt this [@ernstvn](#), because i'd probably stop at second "why" due to lack of gitaly internals knowledge:

- Closed

NFS-02 outage
- why? .. and here its already out of my understanding, but its the most interesting part.

Although I can provide all the syslogs from that time and aid someone who understands more than I do.


As for `Nfs02` -- we had `Nfs02` wimpact (because we were lucky and patient enough not to hit stop/start). The incident itself might have lead to cascading failure, and possible data corruption, which would have resulted in 2 hour downtime (the `xfs_repair` takes at least that time). So I'd rather treat this as a real, `Nfs02` incident.



Andrew Newdigate [@andrewn](#) · 2 years ago

Maintainer

[@ilyaf](#) personally, I'm not completely convinced off this being a Gitaly issue yet. I will continue trying to find the cause but I think it would still be worth considering all possibilities.



Ilya Frolov [@ilyaf](#) · 2 years ago

If that helps:

```
Jul 20 00:48:06 nfs-file-02.stor.gitlab.com kernel: [397366.773843] Out of memory: Kill process 14876 (gitaly
Jul 20 00:48:09 nfs-file-02.stor.gitlab.com kernel: [397370.186770] Out of memory: Kill process 14876 (gitaly
Jul 20 00:48:16 nfs-file-02.stor.gitlab.com kernel: [397376.824377] Out of memory: Kill process 14876 (gitaly
```

full snippet here: <https://gitlab.com/snippets/1668667> 252 OOM invocations in 10 minutes. Look at the pid -- if i'm not mistaken, gitaly was spawning `git` processes faster than OOM killed them (basically a fork bomb).

But the question is `why` here, and this is where I have no power :)

Edited by [Ilya Frolov](#) 2 years ago




Andrew Newdigate [@andrewn](#) · 2 years ago

Maintainer


[@ilyaf](#) ~~would it be possible to get those times in real time format please?~~ nevermind :)

Edited by [Andrew Newdigate](#) 2 years ago



Ilya Frolov [@ilyaf](#) · 2 years ago


[@andrewn](#) Just in case, I copied the full syslog from that day to my homedir on nfs-02: `syslog.1_nfs02_outage.xz`. You can grab it from there or let me know if you need it and I'll copy it where you need.



Andrew Newdigate [@andrewn](#) · 2 years ago


Maintainer

Thanks [@ilyaf](#), I don't have shell access. Could you share the syslog via google drive please?



Ilya Frolov [@ilyaf](#) · 2 years ago


Done, sent you the link. Its pretty big, so you can use `xzgrep` or `xzcat` to search it.



Ilya Frolov [@ilyaf](#) · 2 years ago

Ping me if you need any other files from it -- or [@eReGeBe](#) or [@ahmadsherif](#) should have access there too.

We're keeping a week of file if i'm not mistaken, but I didn't back up anything except `/var/log/syslog` from that day. Maybe gitaly logs themselves have more info, I just don't know where to look.



Paul Charlton [@techguru](#) · 2 years ago

[@ilyaf](#) if the `dmesg` has not rolled over it already, it may have useful information not in the syslog regarding OOM reaper



Andrew Newdigate [@andrewn](#) · 2 years ago

Maintainer

Fork Bomb

The OOM killer first kicked in at **00:48:05 UTC**, which dumped a list of running processes to the syslog. In this we found 1846 `git` and 992 `git-upload-pack` processes running.

Closed **NFS-02 outage**

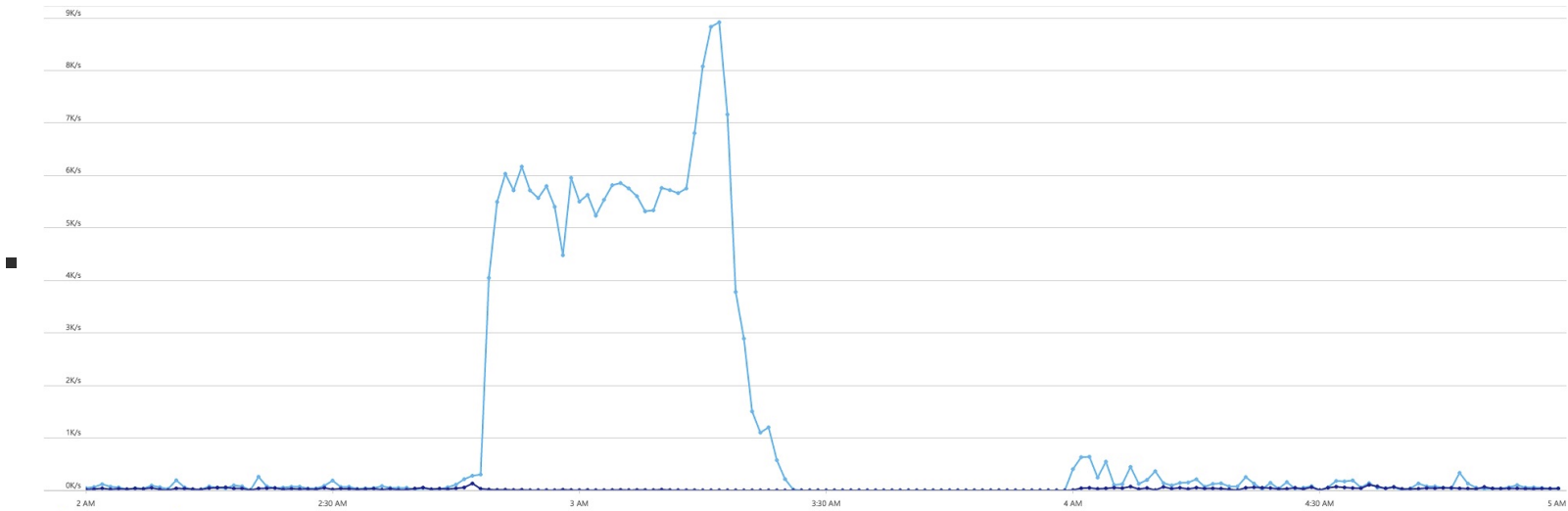
The `git-upload-pack` processes mostly had a consistent rss of ~1500, the `git` processes were divided roughly 50/50 between small rss ~3000 and large rss ~150000.

What caused the fork bomb?

Some possible theories:

1. **An IO operation logjam occurred, causing the git processes to stack up**

- Evidence in favour:
 - The `gitlab-ee` tag deletion occurred just prior to the outage and would have invoked `git-gc` processes on sidekiq which may have led to IO bottleneck. Since `git-gc` occurs over NFS, this would not show up in the logs.
- Evidence against:
 - Azure volume IO graphs show peak corresponding with the time of the problem (not before).



2. **A massive surge of request hit `nfs-file-02`**

- Evidence in favour:
 - Occam's razor: an incoming request spawns a git process. A lot of git processes means a lot of requests.
- Evidence against:
 - We do not see the requests in the logs. This doesn't mean that they didn't happen. No logs were shipped to logstash during the forkbomb and currently gitaly only logs the request `request_id` on `Git::Request::IO::Backend`.
 - There is a possibility that several thousand incoming requests arrived almost instantaneously, caused the forkbomb, caused the server to crash and then failed to finish any of the requests.
 - The requests would need to be near instantaneous, which doesn't fit with Ruby single-threaded model unless many processes did this simultaneously which would be unlikely.
 - The logs may reside on the machine, even though they didn't get shipped. I've requested that [@omame](#) or [@ilyaf](#) retrieve them in case they hold more details.

3. **Gitaly malfunctioned, started spawning multiple git processes for single incoming requests**

- Evidence in favour:
 - In the absence of other theories, this must be a possibility
- Evidence against:
 - Unlikely and no evidence in favour either.

4. **Gitaly commanded git to spawn multiple child git processes for each incoming request**

- Evidence in favour:
 - In the absence of other theories, this must be a possibility
- Evidence in favour:
 - Unlikely and no evidence in favour either.

Edited by [Andrew Newdigate](#) 2 years ago



Andrew Newdigate [@andrewn](#) · 2 years ago

Maintainer

We discussed this in slack, over here: <https://gitlab.slack.com/archives/C101F3796/p1500628248491320>

Without knowing the full cause of this issue, it's difficult to perform a five-why's analysis, but while we investigate further, it's worth raising some issues that would prevent this from occurring again.

More Visibility

1. Gitaly should generate logs or metrics to indicate the start of a request, so that we know what's happening when the process dies. Currently gitaly only logs afterwards, which is too late.
2. Instrument gitaly's `NewCommand` method to log every spawned process.

More Protection

1. NFS servers are critical infrastructure. We should make sure that a surge, malfunction or any other resource allocation issue does not affect the operation of the server as a whole. [@ilyaf](#) suggested putting the `gitlab` service inside a Linux cgroup to protect

Closed

NFS-02 outage

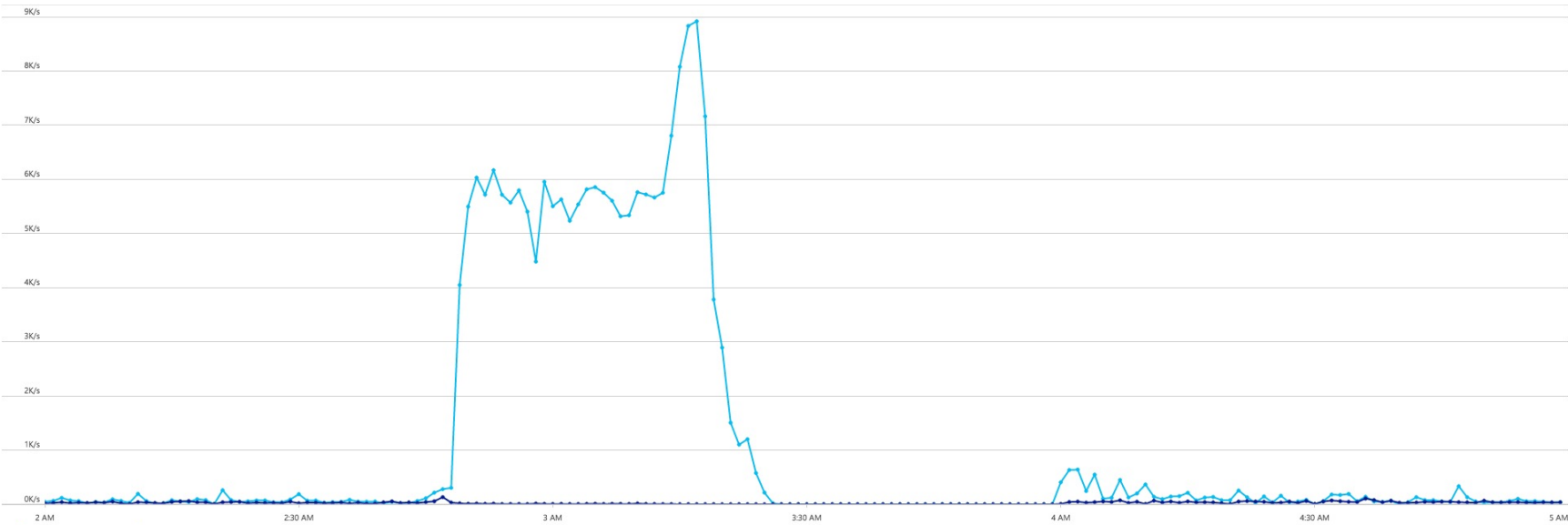
This would ensure that `gitlab` and its child processes are limited to a certain amount of memory, leaving the server with enough memory to continue operating NFS and SSHd.



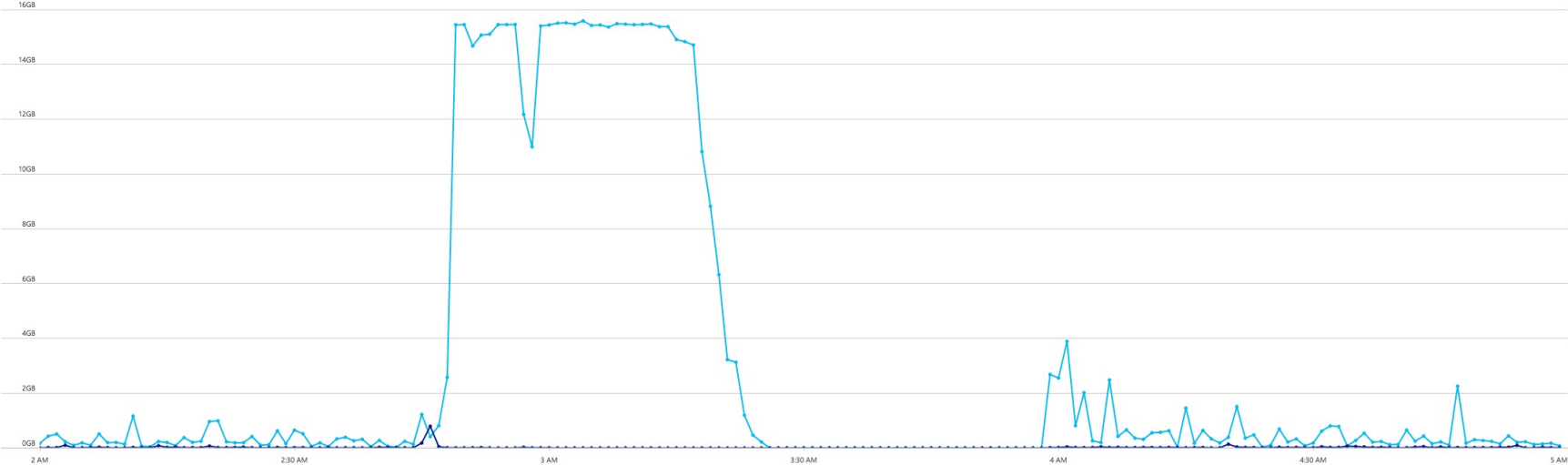
Daniele Valeriani [GitLab] [@omame-gitlab](#) · 2 years ago

Here is what Azure saw during the outage (times are UTC+2).

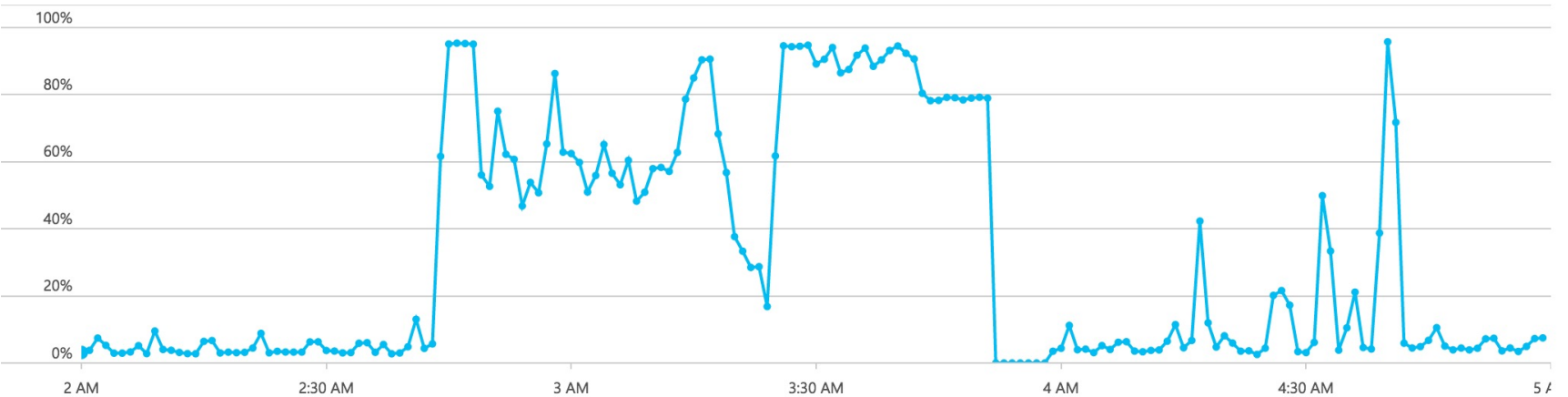
IOPS (spike is read):



Bytes read/written (spike is read):



CPU usage:



Daniele Valeriani [GitLab] [@omame-gitlab](#) assigned to [@ilyaf](#) and [@andrewn](#) 2 years ago



Andrew Newdigate [@andrewn](#) · 2 years ago

Maintainer

Previously, I suggested that one possible option was I `NI YñG YñG Gi BñWAGYñDñ nfs-file-02` , but was unable to find evidence of this due to log shipping failures.

I've since investigate further and have found more evidence that this was indeed the problem.

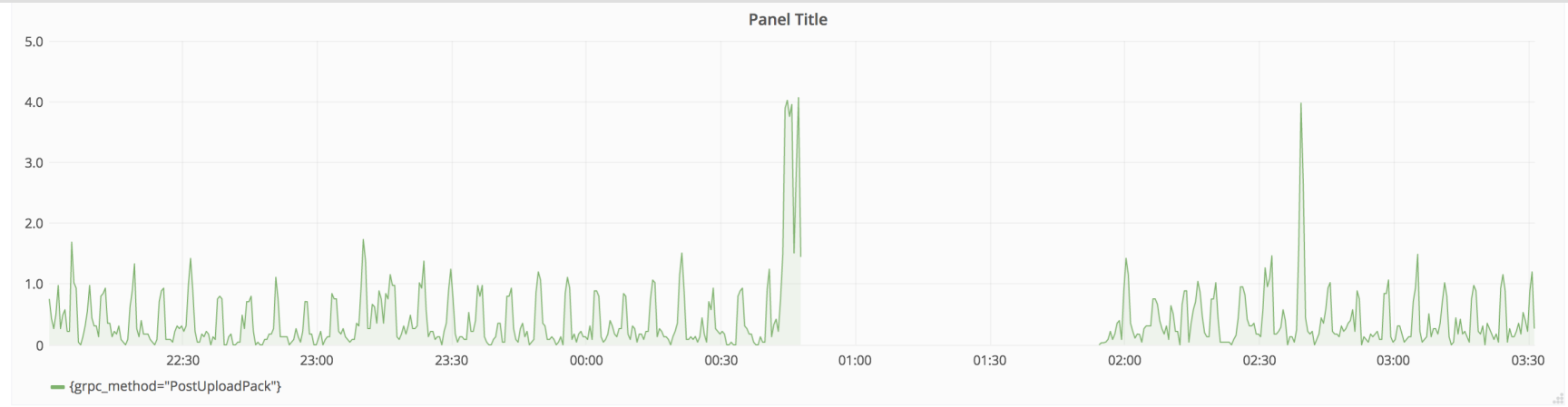
As part of the GRPC Prometheus middleware, we do in fact log the arrive of a new request to the server.

Between **00:42:28 UTC** and **00:47:28 UTC**, a three minute period, we received **809** `PostUploadPack` messages ~4.5 req/second.

Compare this to the base rate of `PostUploadPack` pack messages, at ~0.5 req/sec.

Since there had just been a large deletion of tags from this database, it's likely that the `PostUploadPack` requests required more CPU than a small fetch involving only a few extra commits. Therefore, the surge, combined with the unusual amount of CPU each request

Closed NFS-02 outage



Daniele Valeriani [GitLab] @omame-gitlab · 2 years ago

The question now is: what caused the surge?



Andrew Newdigate @andrewn · 2 years ago

Maintainer

@omame CI runners, people polling the repo for changes

Edited by Andrew Newdigate 2 years ago



Andrew Newdigate @andrewn · 2 years ago

Maintainer

There's a youtube video (I'll find it) of a GitHub engineer at a conference discussing the same issue, with servers polling git repos to check for new code which caused them the same surges.

The solution was to limit the number of concurrent requests per repo, putting the extra clients on hold while waiting (not rejecting them outright, just slowing them down).

The video also goes on to say that the worst offender for causing these surges to their infrastructure was in fact GitHub themselves and that the fix worked well.

I suspect we may put the most load on our own infrastructure in the same way.

Adding this sort of check to Gitaly would be relatively simple and could be implemented as a middleware, so it wouldn't need to complicate the route code.



Andrew Newdigate @andrewn · 2 years ago

Maintainer

Here is the video: "Slow GitHub Down with GitMon"

<https://youtu.be/f7ecUqHxD7o?t=8m37s>

@gl-gitaly 👉 this video is well worth watching



Daniele Valeriani [GitLab] @omame-gitlab · 2 years ago

@omame CI runners, people polling the repo for changes

Do you have data to back that? Were all requests coming form the same source? My question was more specific.



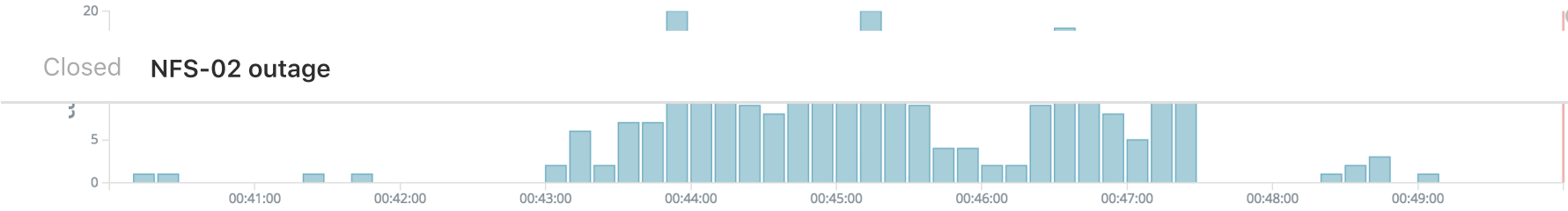
Andrew Newdigate @andrewn · 2 years ago

Maintainer

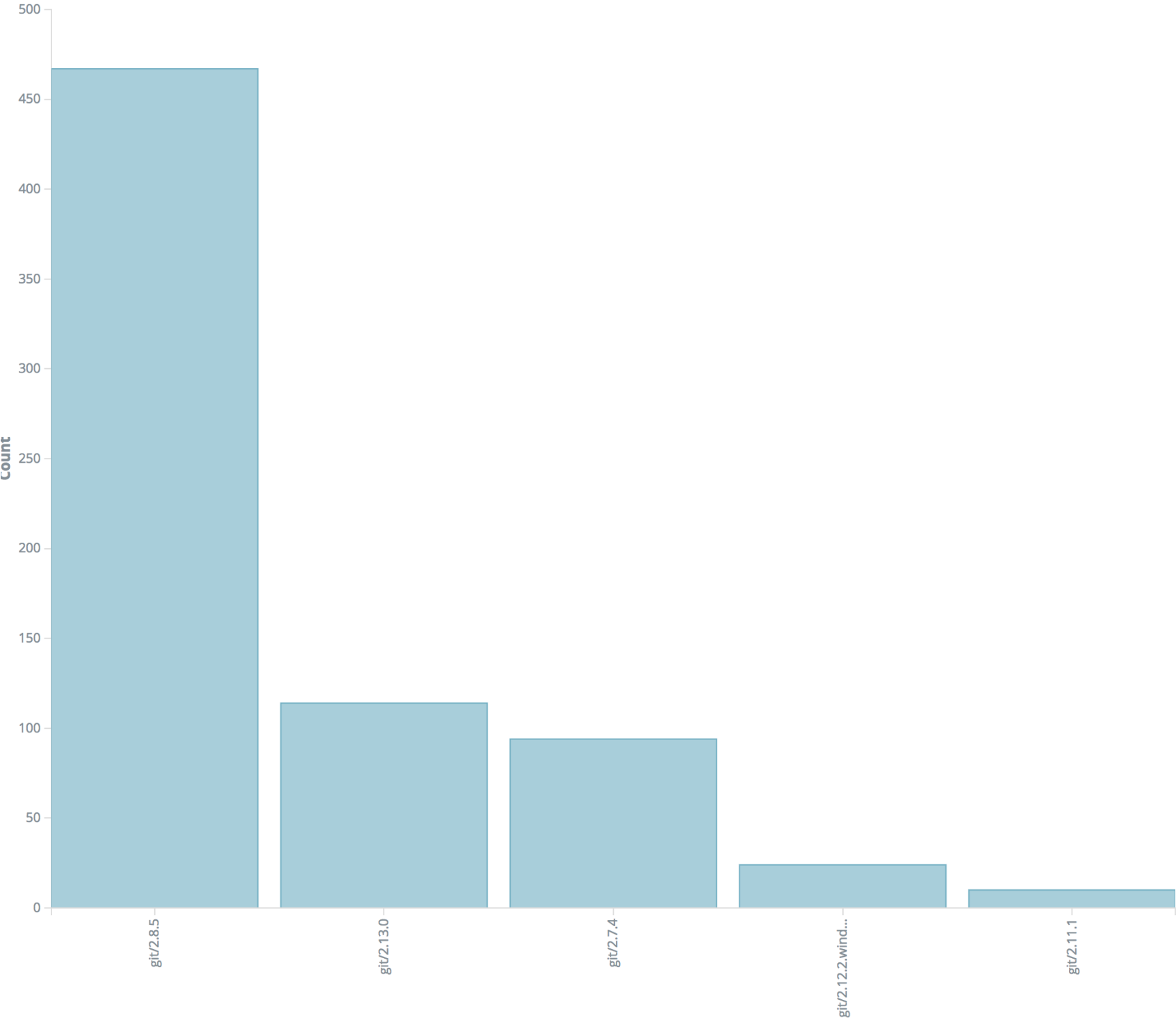
Not that much unfortunately.

haproxy requests to <https://gitlab.com/gitlab-org/gitlab-ee.git/info/refs?service=git-upload-pack>, between **00:40:00 UTC** and **00:50:00 UTC**.

These represent git http remote fetches in the lead up to the outage.



Who were these clients?



Mostly git/2.8.5 .



Brian Neel @briann · 2 years ago

If garbage collection relies on Sidekiq, we do know that Sidekiq (at least on the dedicated Sidekiq nodes) was hung during this outage. Is it possible garbage collection was simply disabled?



Andrew Newdigate @andrewn · 2 years ago

Maintainer

@briann was sidekiq hung before the outage, or as a result of the outage?



Andrew Newdigate @andrewn mentioned in merge request [gitlab-org/git!240 \(merged\)](#) 2 years ago



Brian Neel @briann · 2 years ago

@andrewn Sidekiq was hung due to the security release deployment. It happened around 22:59 UTC, over an hour prior.



Brian Neel @briann · 2 years ago

Sidekiq wasn't fixed until around 04:20 UTC.


 **Andrew Newdigate** @andrewn · 2 years ago

Maintainer

Right, that's really useful. @brianna


Closed NFS-02 outage

- Mass tag deletions get pushed to the repository. As a result, the repo needs a GC!
- No GC or repacks occur because no sidekiq
- Then, a large number of concurrent PostUploadPack fetches
- Since the repo is in need of a GC (and none has happened), each individual PostUploadPack needs to consume much more CPU to generate packfiles on the fly for the connection.
- The number of parallel PostUploadPack requests, combined with the processing each request requires (due to the non-GC'ed repo) slows everything down and locks things up

 **Ilya Frolov** @ilyaf · 2 years ago

@andrewn if there's no new info regarding this, can you update the RCA section with latest data and close this issue? I think we can just open a corrective action issue with https://gitlab.com/gitlab-com/infrastructure/issues/2314#note_35595638 and iterate from there.

@omame what do you think, apart from issue body update, we done here?

 **Paul Charlton** @techguru · 2 years ago

@ilyaf @andrewn @omame


1. is there an architectural imbalance for scaling with the current deployment of Gitaly? (i.e.: 12 file servers, but could be dozens of FE/SideKiq nodes generating traffic.)
2. How much memory and CPU is available on the NFS nodes (seems like they need to scale vertically due to small number) whereas the FE/Sidekiq are scaling horizontally using cheaper ram/cpu cores.
3. Should there be a Gitaly tier between the FE/Sidekiq and the NFS boxes?

 **Ernst van Nierop** @ernstvn-gitlab · 2 years ago

@ilyaf

As for NFS -- we had NFS impact (because we were lucky and patient enough not to hit stop/start). The incident itself might have lead to cascading failure, and possible data corruption, which would have resulted in 2 hour downtime (the xfs_repair takes at least that time). So I'd rather treat this as a real, NFS incident.

Duly noted! And another good reason to follow the same workflow for NFS incident since it is hard to tell whether it will stay minor or become major.

 **Ernst van Nierop** @ernstvn-gitlab mentioned in issue [#2304 \(closed\)](#) 2 years ago

 **Andrew Newdigate** @andrewn · 2 years ago

Maintainer

@techguru really great questions!

is there an architectural imbalance for scaling with the current deployment of Gitaly? (i.e.: 12 file servers, but could be dozens of FE/SideKiq nodes generating traffic.)

Yes, unfortunately there is an imbalance.

What would be ideal is to spin up dozens on File Servers each running Gitaly to spread the load around.

Unfortunately, at present, each NFS server is a single point of failure for the entire GitLab.com site and as such each additional NFS server we add will increase the chances of failure -- NFS is a single point of failure.

There are various initiatives currently underway to mitigate this, but until they are in place we are very reluctant to solve this problem by throwing more boxes at it.

Once Gitaly goes through Gitaly and we can disable NFS on our infrastructure, we'll be able to scale up our file server hosts (as well as eventually do many other things, like multi-file-server replication, or even multi-cloud replication).

How much memory and CPU is available on the NFS nodes (seems like they need to scale vertically due to small number) whereas the FE/Sidekiq are scaling horizontally using cheaper ram/cpu cores.

The file server nodes are beefy machines, but if we need to, we still have headroom to scale them vertically with more cores and more memory. Obviously this will cost a lot to run, but it's a temporary measure until everything is off NFS.

Closed **NFS-02 outage**

other side of an NFS mount. In other words, we're moving where the i/o happens, but not the throughput.

This is obviously not true for all operations (`pack-file` being the CPU-bound headache we're dealing with in this issue) but I'm confident that we can use a variety of approaches to get around this (per repo throttling, git tuning parameters, cgroups, etc, etc.

1. Should there be a Gitaly tier between the FE/Sidekiq and the NFS boxes?

This is always an option. We build a series of "pods", each consisting of one nfs server, two or more gitaly servers with volumes mounted off the single nfs server. Workers communicate with the Gitaly servers. This would allow us to scale CPU and memory horizontally until they hit i/o bandwidth limits.

This would, however, be a detour from our goal and I would rather try focus on the simple solution for now unless we have to.



Andrew Newdigate [@andrewn](#) · 2 years ago

Maintainer

[@ilyaf](#) for some reason I'm unable to edit the description, so I'll post the Five Whys here:

Why did this outage occur?

- Why? `nfs-file-02` stopped responding for about 50 minutes from 00h50 until 01h40
- Why? The server was unable to handle new incoming requests due to being out-of-memory and CPU.
- Why? About 800 `https post-upload-pack` requests came in within a 3 minute period. Normally the server would be able to handle these relatively easily, but in this case, the requests took the server down.
- Why? Each `post-upload-pack` request needed to do more compression than normal since the `gitlab-ee` repository had recently had a large number of tags deleted and had not been `git gc` 'ed. If the GC had occurred, the `post-upload-pack` processes would have not each consumed as many resources as they did?
- Why? Sidekiq was down and therefore the usual `git gc` processes had not taken place

Edited by [Andrew Newdigate](#) 2 years ago



Andrew Newdigate [@andrewn](#) mentioned in issue [gitlab-org/gitaly#429 \(closed\)](#) 2 years ago



Andrew Newdigate [@andrewn](#) · 2 years ago

Maintainer

Corrective actions

- ☐ Put Gitaly / Gitlab on the NFS servers inside a cgroup to protect critical services such as `nfsd` and `sshd`. [@ilyaf](#) please could you create an issue for this?
- ☐ Add per-repo throttling to Gitaly: [gitlab-org/gitaly#429 \(closed\)](#).



Andrew Newdigate [@andrewn](#) closed 2 years ago



Pablo Carranza [GitLab] [@pcarranza-gitlab](#) · 2 years ago

Nice analysis [@andrewn](#) 👍

Sidekiq was down due to a deployment happening or was it something else?

We could add that one to the why's too.



Brian Neel [@briann](#) · 2 years ago

[@pcarranza](#) A bug in the CI regex DoS prevention included with the security release DoS'd Sidekiq.

Edited by [Brian Neel](#) 2 years ago



Ilya Frolov [@ilyaf](#) mentioned in issue [#2364 \(closed\)](#) 2 years ago



Ilya Frolov [@ilyaf](#) changed the description 2 years ago



Ilya Frolov @ilyaf · 2 years ago

Closed **NFS-02 outage**

able to update it now.

In the corrective action I've specified the most boring solution we can do (but it can be done in, like, 5 minutes across all the fleet, which is nice), but I also cc:-ed everyone so that we'll get a lot of other ideas :)



Ilya Frolov @ilyaf mentioned in issue [#2493 \(moved\)](#) 2 years ago



Andrew Newdigate @andrewn mentioned in issue [gitlab-org/omnibus-gitlab#2622](#) 2 years ago



Achilleas Pipinellis 🦉 @axil removed milestone 2 years ago



Andrew Newdigate @andrewn moved to [production#213 \(closed\)](#) 1 year ago



Andrew Newdigate @andrewn mentioned in issue [production#219 \(closed\)](#) 1 year ago

Please [register](#) or [sign in](#) to reply