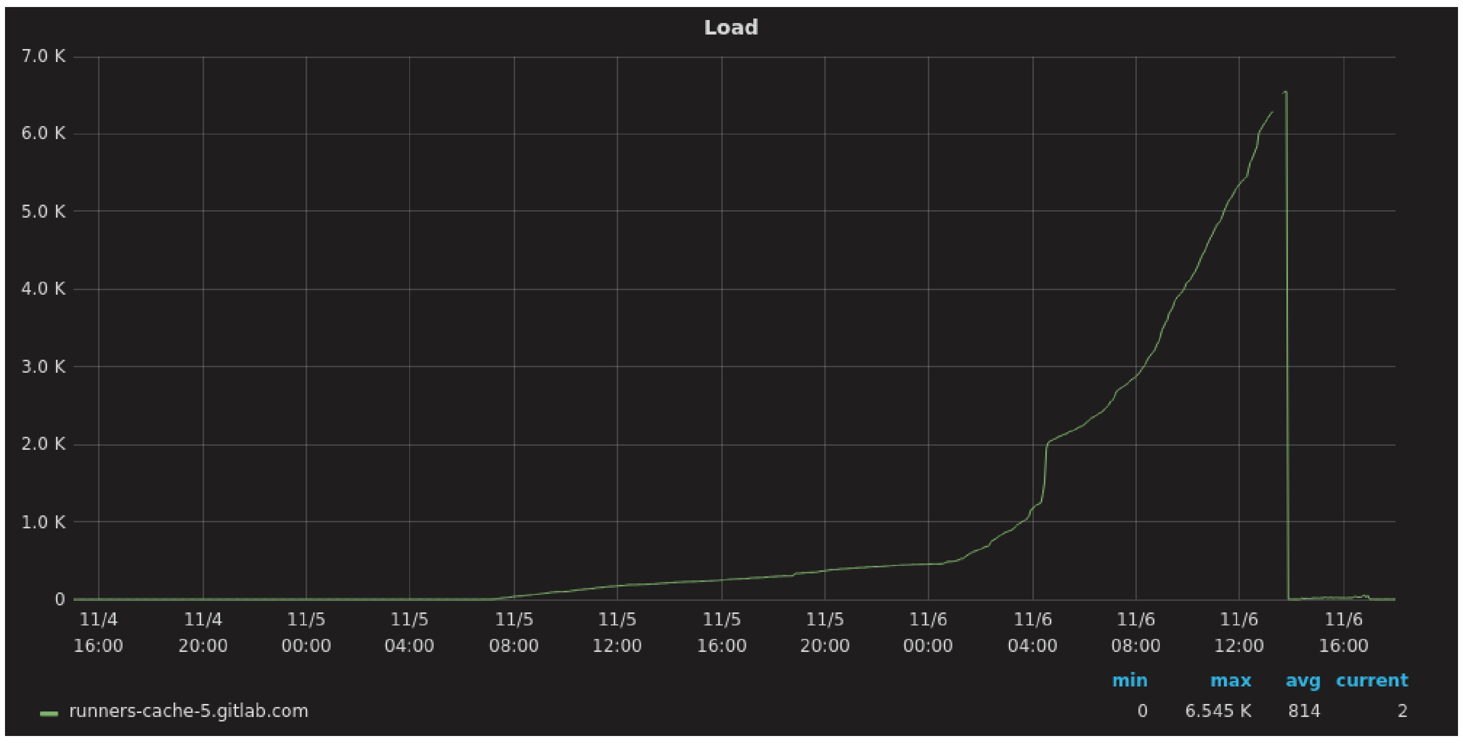
Closed (moved) Opened 2 years ago by **Tomasz Maczukin** 🌴

# GitLab.com Runners cache server outage at 2017-11-05/06

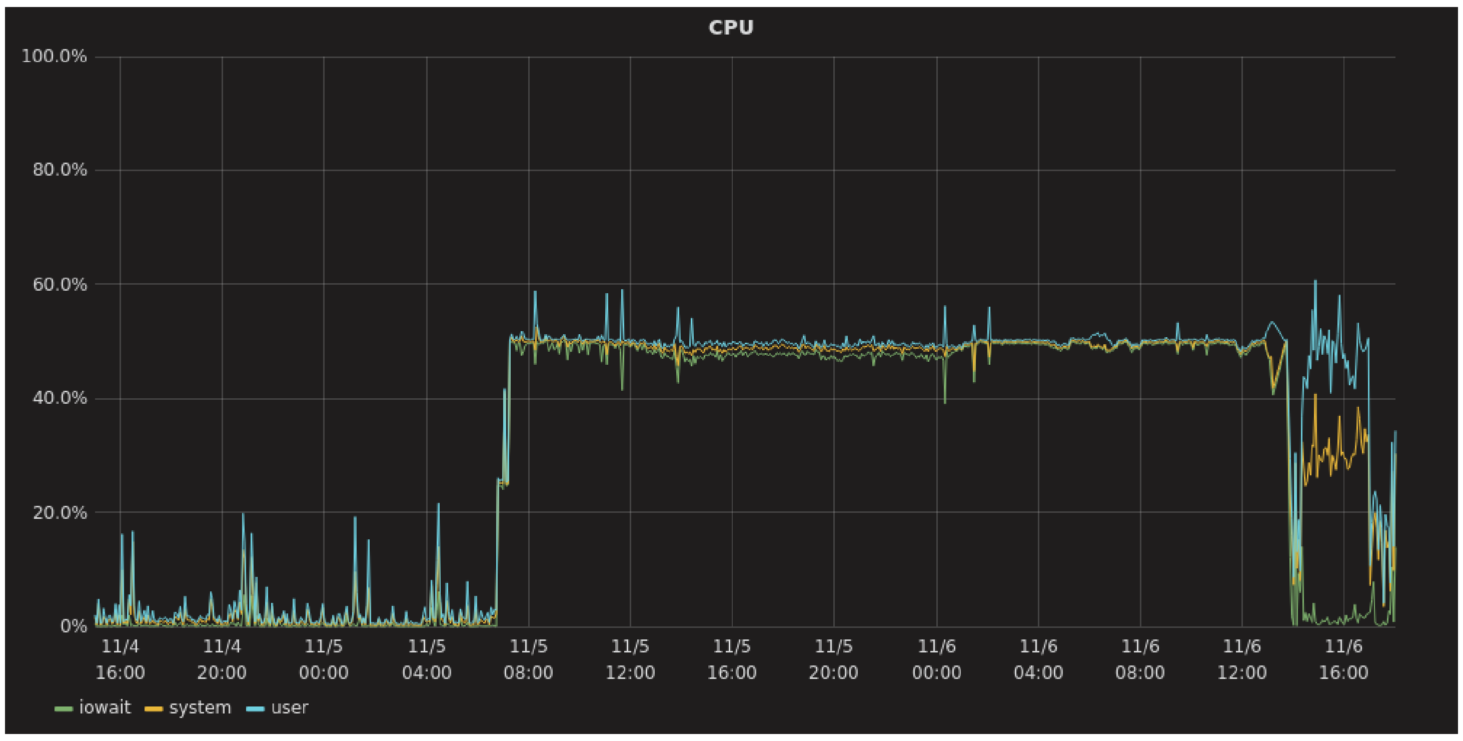
## Outage timeline

At 2017-11-05 we've started to receive reports about cache operations on our CI jobs failing on both downloading and uploading steps (support-forum#2653, support-forum#2655, support-forum#2659). Before we've checked whats going on, one of the users noticed an extremely high load on runners-cache-5.gitlab.com server (https://gitlab.com/gitlab-com/support-forum/issues/2653#note_45842001).

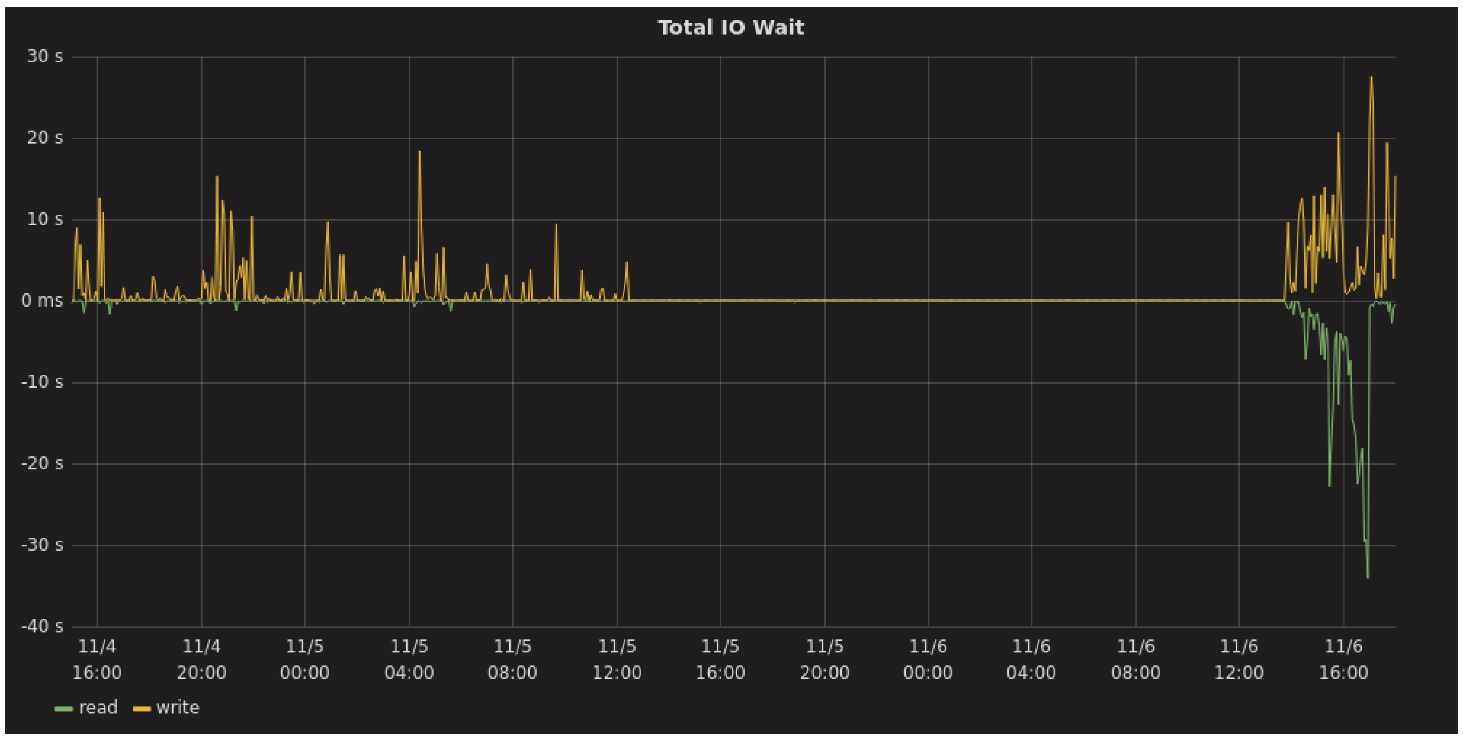
At 2017-11-06 ~14:00 UTC the load on this cache server was at level **~6.5k**!

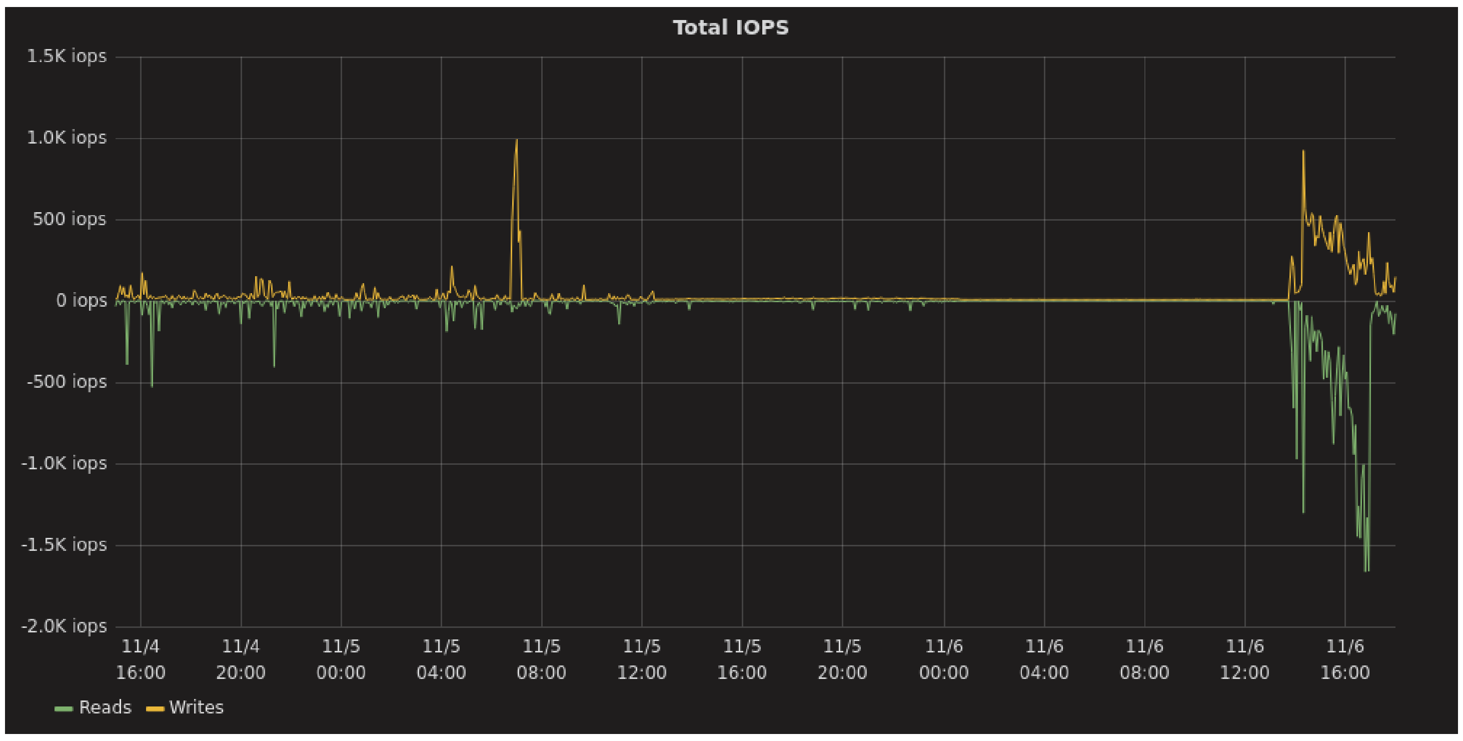


With @jtevnan we've started to investigate what's wrong on this machine. After looking on graphs and basic metrics on the host everything - excluding load - seemed to be OK. We were able to SSH into the server and work there without any problems. After a while @jtevnan enabled detailed CPU view in `htop` and noticed that a lot of CPU is used for IO operations. This indeed can be seen on the graphs:



At the same time IO operations on the machines were almost not visible:

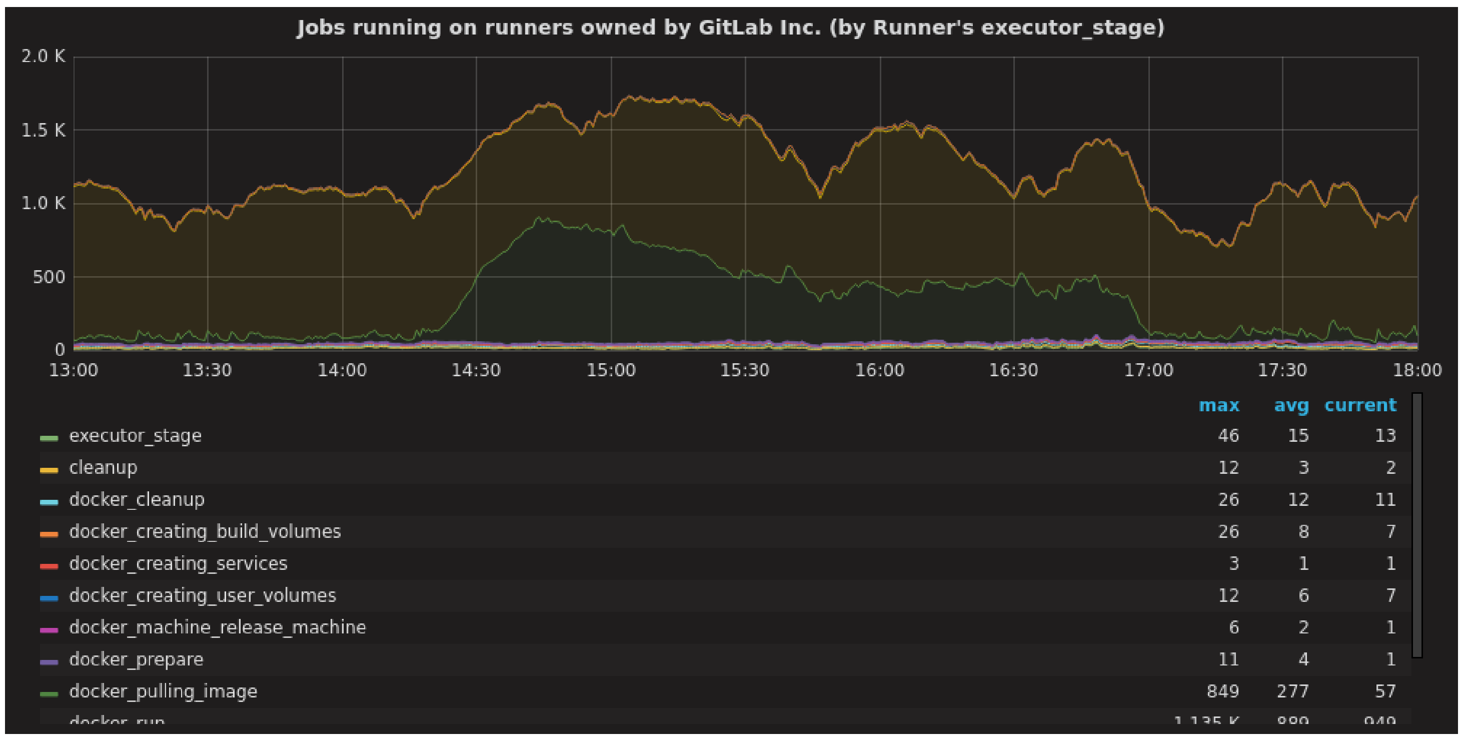




After further investigation we've discovered that `iostat -x` shows that `/dev/sda` utilization is at the level of 100%. On runners-cache-5.gitlab.com this disk is part of the đ! ŭZ ᾀ array used as storage for cache data. The disk itself is a Digital Ocean external volume (which most probably is attached through network to the hypervisor).

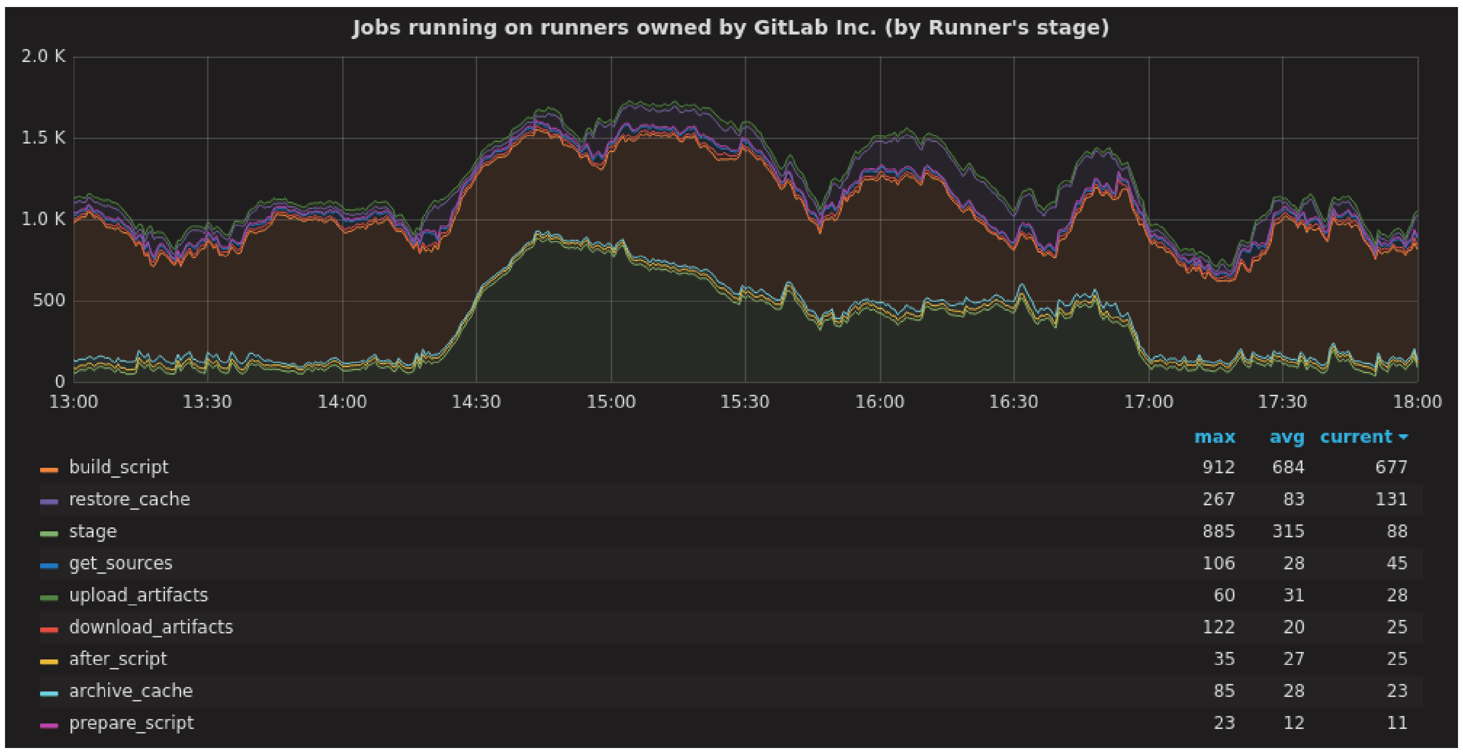
After another few checks we've been unable to discover what causes the 100% utilization of this disk. Since this was a RAID 0 array there was no rebuilt process that could be running on the disk. We've decided to reboot the machine at ~14:20 UTC. The reboot took a while (longer than usual for Digital Ocean droplets) but after the machine was restarted the problem disappeared. We've been monitoring the machine for next few minutes and everything looked fine and the initial issue was resolved.

After 2.5 hour @ayufan noticed that jobs are being processed much longer than usual. We've found that there is a lot of jobs waiting for Docker image to be pulled:



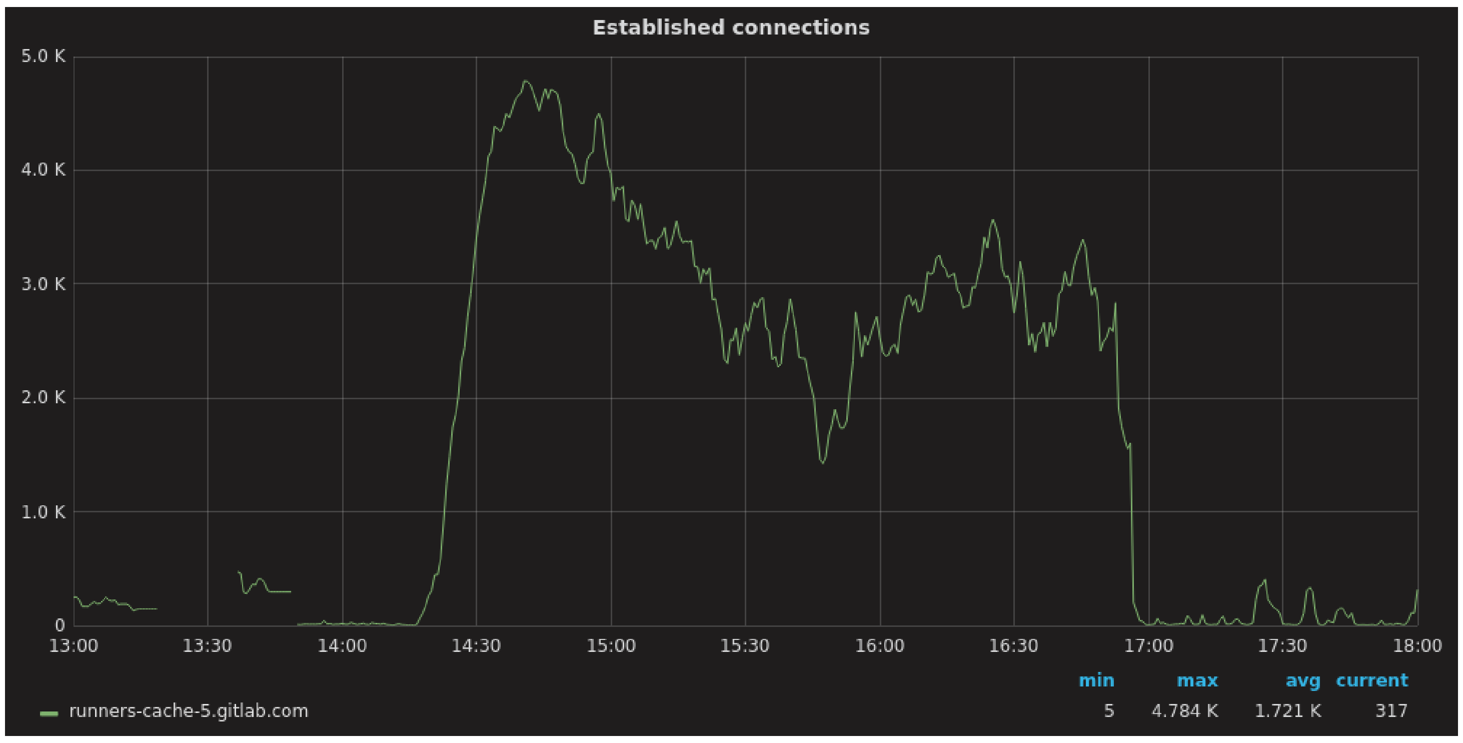
| | max | avg | current |
|---|---|---|---|
| executor_stage | 46 | 15 | 13 |
| cleanup | 12 | 3 | 2 |
| docker_cleanup | 26 | 12 | 11 |
| docker_creating_build_volumes | 26 | 8 | 7 |
| docker_creating_services | 3 | 1 | 1 |
| docker_creating_user_volumes | 12 | 6 | 7 |
| docker_machine_release_machine | 6 | 2 | 1 |
| docker_prepare | 11 | 4 | 1 |
| docker_pulling_image | 849 | 277 | 57 |
| docker_run | 1.135 K | 889 | 949 |

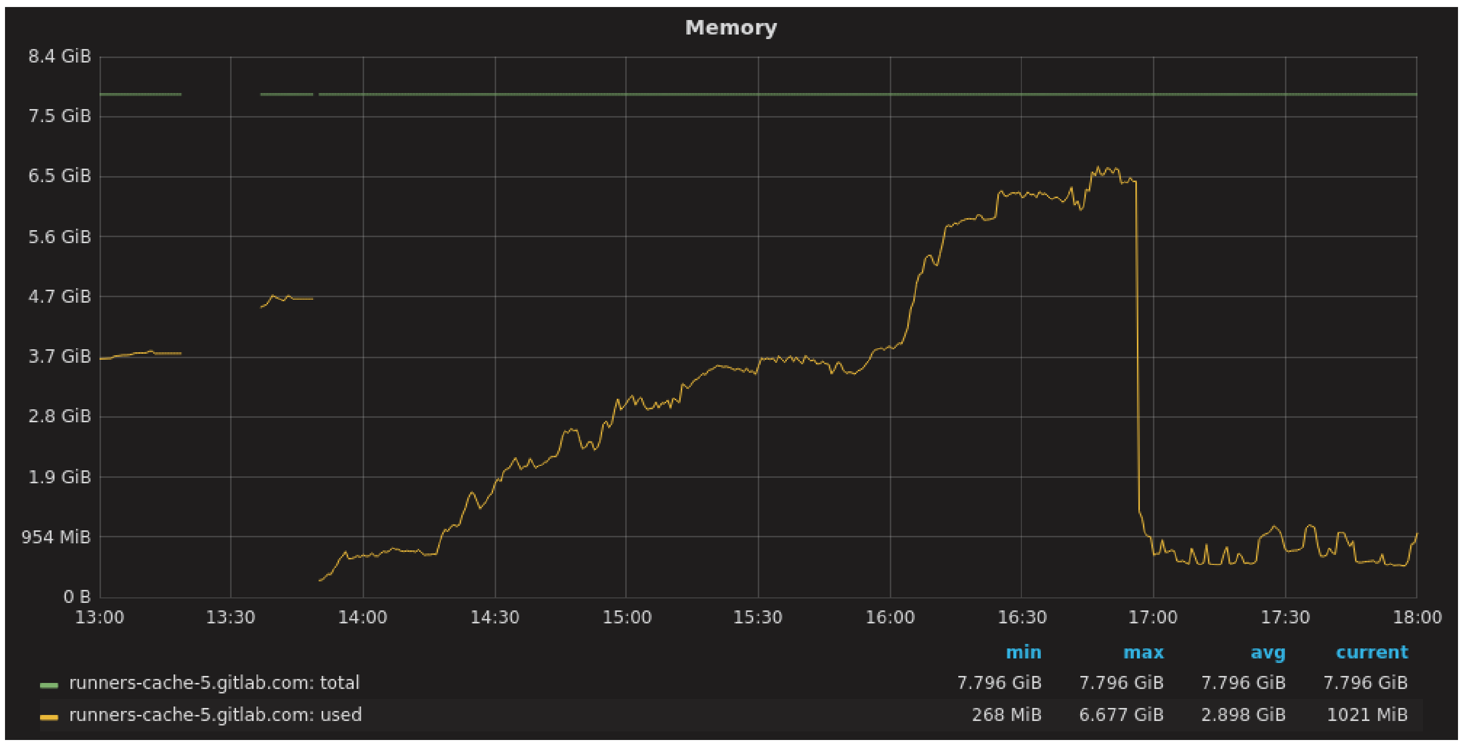
and more than usual (but not too much) jobs that are waiting for cache operations to be finished:

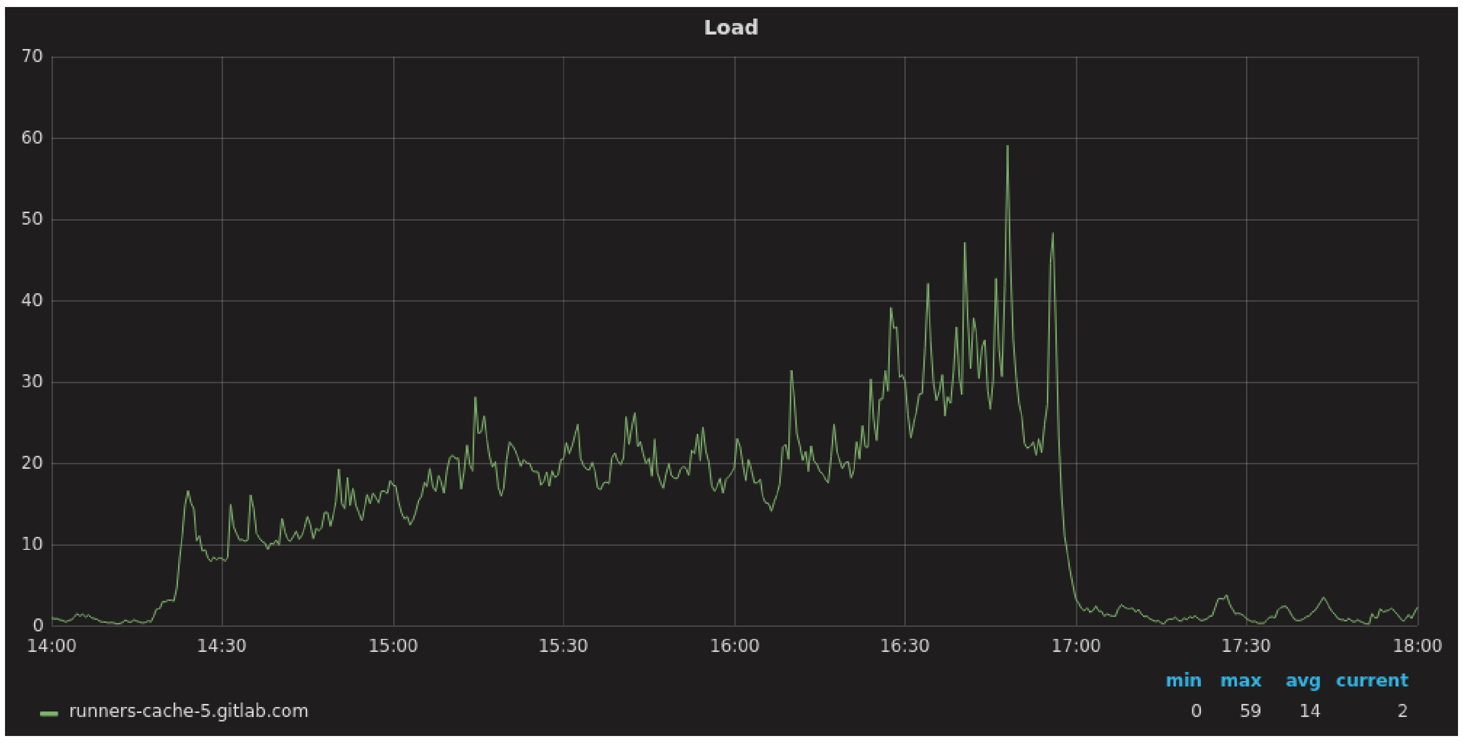


| | max | avg | current |
|---|---|---|---|
| build_script | 912 | 684 | 677 |
| restore_cache | 267 | 83 | 131 |
| stage | 885 | 315 | 88 |
| get_sources | 106 | 28 | 45 |
| upload_artifacts | 60 | 31 | 28 |
| download_artifacts | 122 | 20 | 25 |
| after_script | 35 | 27 | 25 |
| archive_cache | 85 | 28 | 23 |
| prepare_script | 23 | 12 | 11 |

After looking on the graph it's obvious that the number of jobs waiting for image started to grow at the moment when runners-cache-5.gitlab.com host was restarted. After looking on few more metrics it become obvious that the problem is caused by cache server:







The load was not so big as before the reboot, but the number of established connections and memory usage was much higher than it should. Since the problem was not in cache operations but in Docker images pulling I've decided to check if Docker Registry mirror was started. And that was the problem!

Registry mirror, that is configured on our Runners, was disabled few months ago when we were investigating previous performance problems on runners-cache-5 and it was not working up to 2017-11-06. After machine reboot, Docker Registry Mirror was started automatically. The performance of Docker Registry Mirror was enough small to affect jobs performance.

After stopping Docker Registry Mirror at ~17:00 UTC the issue was resolved (what can be seen on the graphs above).

## What went wrong

Initial issue - cache problems:

- There was no alerting about the extremely high load on the server! While preparing a general alerting for load metric, having different machines types in fleet, is not easy, having a load at level **6.5k** should definitely

alert us. I've searched our Slack channel with alerts and for the time when the load on runners-cache-5.gitlab.com was growing I haven't seen any alert. Looking on our alerting rules I see that we have an alert for `load1 > 200` only for DB instances. We should have this also for other machines.
- Even after we've noticed the high load and the high `iowait` CPU utilization, we had no information what causes it. Both IO metrics that we are tracking at https://performance.gitlab.net/dashboard/db/host-stats?refresh=1m&orgId=1&var-node=runners-cache-5.gitlab.com were silent. It was `iostat -x` which led us to the conclusion that this is a problem with disk in RAID array that causes the issue.

Secondary issue - Docker Registry Mirroring problems:

- Docker Registry Mirror was disabled some time ago due to performance issues. It was done manually for test purposes, and never turned into final solution in our chef configuration. This ended with Docker Registry mirror being started after machine rebooted which caused the problem.
- There was no alerting that could inform us that something is wrong. It was @ayufan's personal feeling that there is a slowness in jobs processing while having 25-50% of jobs waiting for Docker image pull to be finished could be easily turned into alerting rule.

## Actions to be taken

1. Prepare a generic alert for load metric (#3159 (closed)).
2. Investigate if there is any metric that tracks the same data that we've discovered with `iostat -x`'s utilization field. If there is something - turn this into alerting rule (#3159 (closed)).
3. Decide what to do with Docker Registry Mirroring and implement it finally in the role definition (#3158).
4. Prepare alerting rules for unusual proportions of different states and stages while jobs processing (#3159 (closed)).

Edited 2 years ago by Tomasz Maczukin

Linked issues 0

Kamil Trzciński 🔴 @ayufan · 2 years ago Developer

Thanks @tmaczukin for awesome explanation :)

Kamil Trzciński 🔴 @ayufan · 2 years ago Developer

@tmaczukin Would you mind starting a meta issue to track all needs for alerting? it seems to be yet another thing to monitor, as this can be considered as a outage .

Tomasz Maczukin 🌴 @tmaczukin · 2 years ago Developer

@ayufan I'll prepare issues for `Actions to be taken` and close this one after this. I can put all alerting actions to one of them and mark it with meta :)

Tomasz Maczukin 🌴 @tmaczukin mentioned in issue #3158 2 years ago

Tomasz Maczukin 🌴 @tmaczukin changed the description 2 years ago

Tomasz Maczukin 🌴 @tmaczukin mentioned in issue #3159 (closed) 2 years ago

Tomasz Maczukin 🌴 @tmaczukin changed the description 2 years ago

Tomasz Maczukin 🌴 @tmaczukin closed 2 years ago

Achilleas Pipinellis 🦅 @axil removed milestone 2 years ago

Andrew Newdigate @andrewn moved to production#272 (closed) 1 year ago

Steve Azzopardi @steveazz mentioned in issue production#2357 4 days ago

Please register or sign in to reply