Clos Closed **Incident Review for 2019-10-15 High latency for ci-runners**

Incident Review for 2019-10-15 High latency for ci-runners

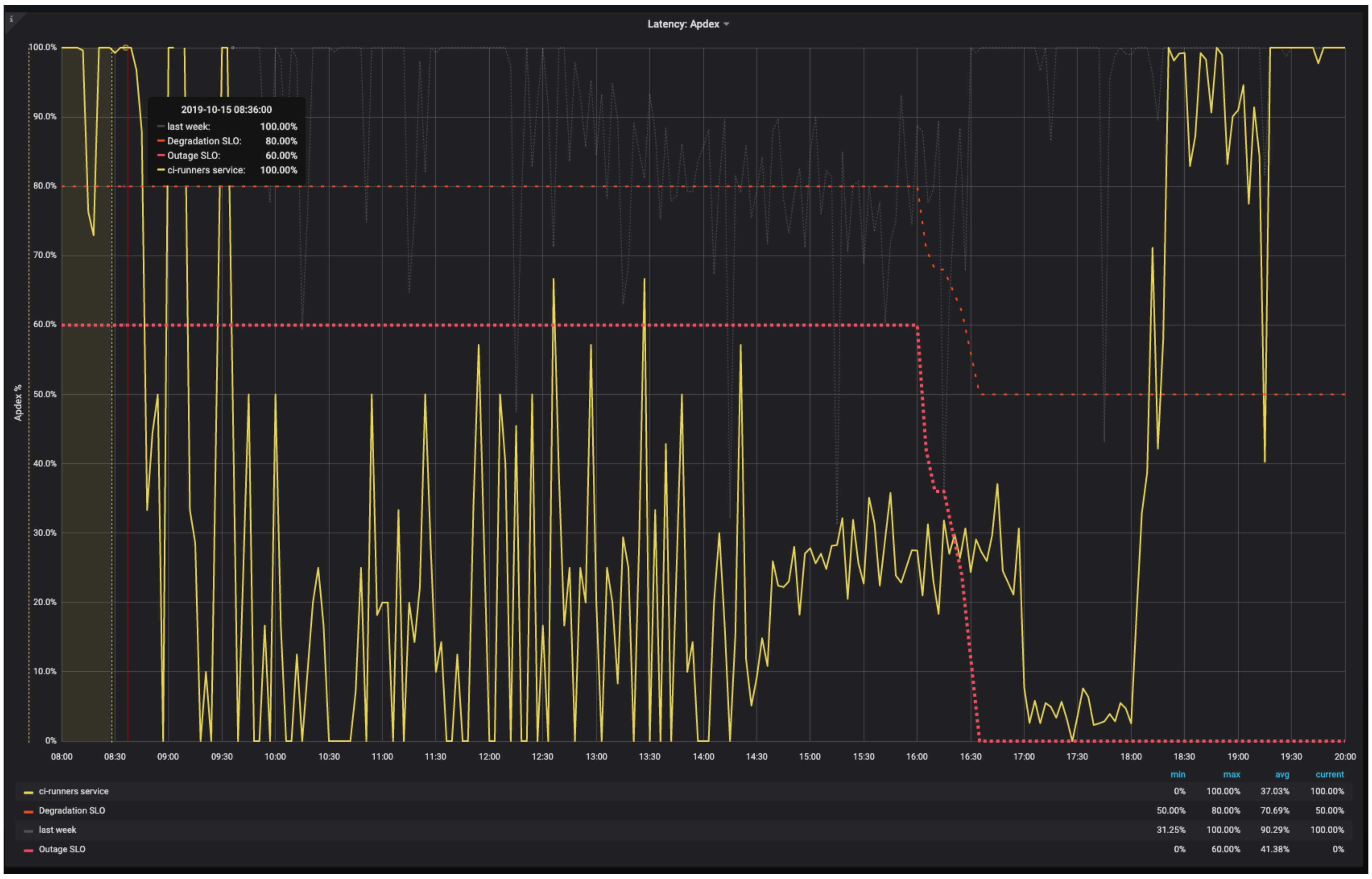
Incident Review for production#1249 (closed).

Summary

A brief summary of what happened. Try to make it as executive-friendly as possible.

- Service(s) affected : GitLab.com CI runners
- Team attribution : Infrastructure
- Minutes downtime or degradation : 2019-10-15T08:48Z to 2019-10-15T18:21Z - 9h33m(573 minutes)

- Graph of Runner Latency Dip
- Graph of Service Platform Metric



	min	max	avg	current
ci-runners service	0%	100.00%	37.03%	100.00%
Degradation SLO	50.00%	80.00%	70.69%	50.00%
last week	31.25%	100.00%	90.29%	100.00%
Outage SLO	0%	60.00%	41.38%	0%

Impact & Metrics

- What was the impact of the incident?

Severely degraded state (to the point we could say outage) for CI runners for Gitlab.com

- Who was impacted by this incident?

All GitLab.com users running any kind of CI pipeline.

- How did the incident impact customers? (i.e. preventing them from doing X, incorrect display of Y, ...)

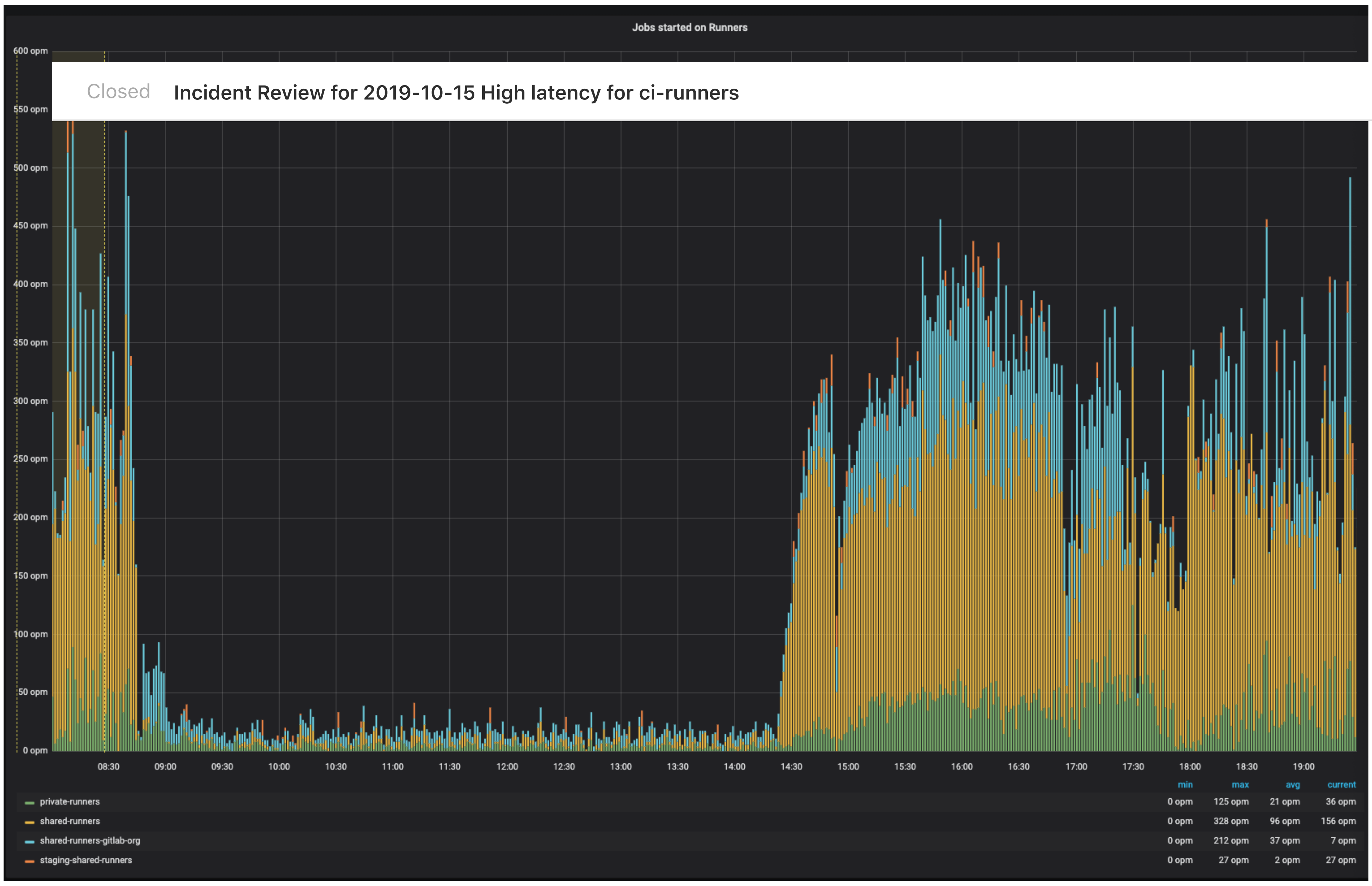
Users of GitLab.com were seeing all of their CI jobs "stuck" in a pending state for many hours. This meant users were not having CI jobs run to test or deploy their code.

- How many attempts were made to access the impacted service/feature?

Looking at the pending queue graph for CI jobs, there were as many as approximately 18,500 jobs pending running. A normal level is around 2,000 to 3,000 pending jobs. Per the latency graph above, many of those pending jobs were waiting for well over an hour for a runner machine.

The graph of jobs started on runners shows the clear bottleneck and delay.

Closed **Incident Review for 2019-10-15 High latency for ci-runners**



- How many customers were affected?

Making a rough guess. From Prometheus data we can say that between 08:45 and 14:22(5h37m or 337m), we were severely throttled and degraded.

During that time the data show we ran approximately these number of jobs on our runners:

Runner	Number of Jobs
shared-runners-gitlab-org	3096
shared-runners	1603
private-runners	1616
staging-shared-runners	209

Grabbing the same data timeframe from the week before (2019-10-08) for reference:

Runner	Number of Jobs
shared-runners-gitlab-org	16130
shared-runners	72616
private-runners	10225
staging-shared-runners	1697

From 14:22 to 18:21(declared end of incident - 3h59m - 239m), the data show we ran

Runner	Number of Jobs
shared-runners-gitlab-org	15956
shared-runners	44288
private-runners	8925
staging-shared-runners	752

Summarizing at a high level. We can confidently say we significantly delayed more than 10,000 jobs on shared runners for gitlab-org (ourselves) and more than 40,000 for shared runners - users of GitLab.com

- H

Reference discussion above.

Detection & Response

Start with the following:

- How was the incident detected?
- Did alarming work as expected?,
- How long did it take from the start of the incident to its detection?
- How long did it take from detection to remediation?
- Were there any issues with the response to the incident? (i.e. bastion host used to access the service was not available, relevant team member wasn't page-able, ...)

Answers to the five questions above grouped and answered in the comment below: #8176 (comment 232020889)

Root Cause Analysis

Our GCE API quota was decreased

This is the root cause of the whole incident. The 09:30 drop in request rate is time-correlated with a quota increase for SSD space in that same project. I made that request, have re-checked my confirmation emails, and am 100% sure that neither I nor a Google rep mentioned a simultaneous quota decrease elsewhere!

What went well

Start with the following:

- Identify the things that worked well or as expected:

1. Metrics dashboards were very clear about showing the decrease in running jobs and pending queue.

- Any additional call-outs for what went particularly well.

What can be improved

Start with the following:

- Using the root cause analysis, explain what can be improved to prevent this from happening again.
- Is there anything that could have been done to improve the detection or time to detection?
- Is there anything that could have been done to improve the response or time to response?
- Is there an existing issue that would have either prevented this incident or reduced the impact?
- Did we have any indication or beforehand knowledge that this incident might take place?

1. During the recovery, there was a Dockerhub outage which contributed to many jobs failing (possibly speeding up the recovery of the pending queue).
2. The waiting length of time to re-approve getting our API quotes increased back to the levels we expected was long on the order of 3 hours. There was a period of approximately 4 hours we were well below our expected API limit.

Corrective actions

- List issues that have been created as corrective actions from this incident.
- For each issue, include the following:
  - - Issue labeled as corrective action .
  - Include an estimated date of completion of the corrective action.
  - Incldue the named individual who owns the delivery of the corrective action.

1. (Look at local registry cache)[#3158] - this would have helped with the recovery. A subsequent dockerhub outage created many failed jobs which we could have prevented

Guidelines

- Blameless RCA Guideline
- 5 whys

Edited 8 months ago by David Smith

Linked issues 2

Relates to

Closed **Incident Review for 2019-10-15 High latency for ci-runners**

production#1249

Decide what to do with Docker Registry mirroring for our CI fleet

#3158

Related merge requests 1

Lower CI runner apdex alerting threshold

gitlab-com/runbooks!1531

Craig Furman @craigf mentioned in issue production#1249 (closed) 8 months ago

Craig Furman @craigf · 8 months ago Owner

The incident is not over (yet), but we expect a quota increase from GCP to come through soon that should allow the backlog to begin draining. I think there are a few things that went wrong, and some corrective actions we can take.

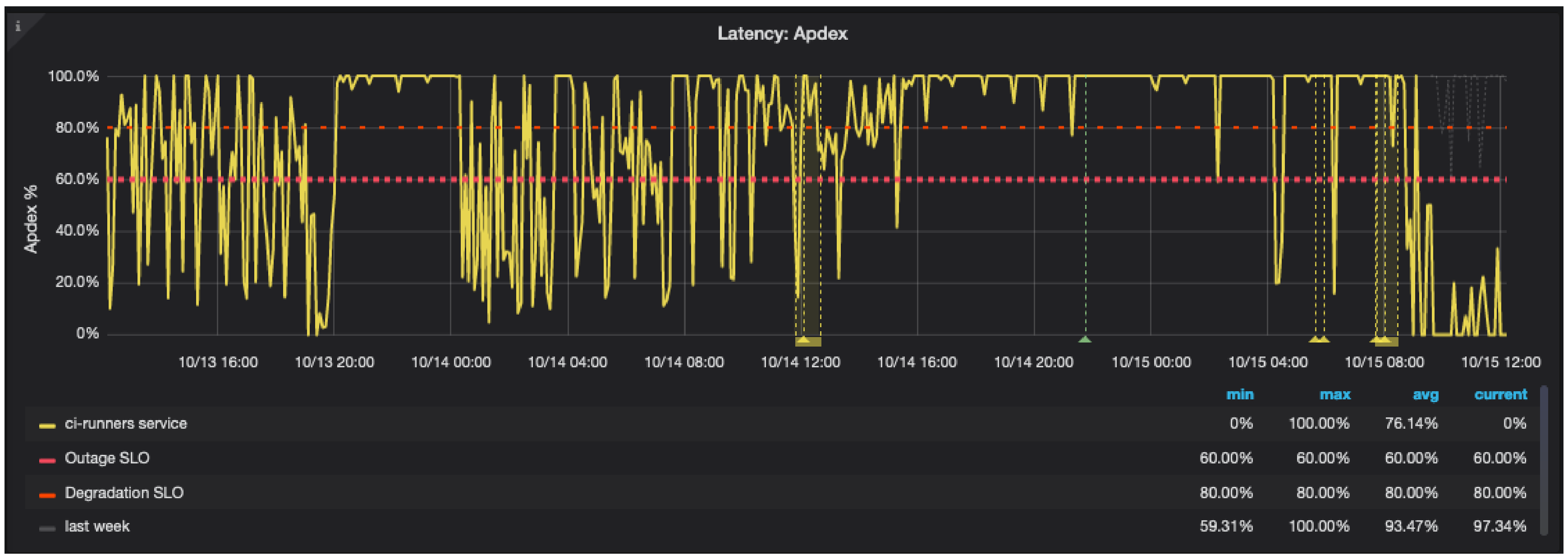
Our GCE API quota was decreased

This is the root cause of the whole incident. The 09:30 drop in request rate is time-correlated with a quota increase for SSD space in that same project. I made that request, have re-checked my confirmation emails, and am 100% sure that neither I nor a Google rep mentioned a simultaneous quota decrease elsewhere!

We are reaching out to Google to find out why this quota was decreased. We were under the impression that we were running at 12k, not 2k.

We were not aware of the problem for a few hours

I had silenced the latency alert for CI runners. Our current latency goal for CI runners is for 95% of jobs to start within 60 seconds. We sent a paging alert when the proportion was below 80%, as a stepping stone. Even this was not reflective of where we are currently: the alert was paging a lot, particularly during a scheduled job spike at 4am UTC every day. There was no obvious short-term solution, and after discussions about how we calculate the SLO and weekend capacity I silenced it, and failed to keep on top of that thread.



	min	max	avg	current
ci-runners service	0%	100.00%	76.14%	0%
Outage SLO	60.00%	60.00%	60.00%	60.00%
Degradation SLO	80.00%	80.00%	80.00%	80.00%
last week	59.31%	100.00%	93.47%	97.34%

You can see from the above chart (showing 2 days of data) that this incident clearly resulted in worse latency than "the usual". Ideally, the on-call would have been paged, but it's also important to be able to distinguish signal from noise, and spot short-term emergencies while working towards unrelated long-term improvements. With that in mind I think a sensible corrective action would be to lower the SLO alerting threshold further. We would be alerting only in far worse conditions than our current goal, but it would achieve the aims stated in the previous sentence.

We did not receive an alert for exceeding our quota

This morning we received a `CICDGCPQuotaCriticalUsage` alert because we were getting close to an SSD space quota in that same project. We should check whether or not we have alerting in place for all quotas, and also for exceeding quotas (not just approaching them).

cc @gitlab-com/gl-infra

Edited by Craig Furman 8 months ago

Collapse replies

Closed Incident Review for 2019-10-15 High latency for ci-runners

gitlab-com/runbooks!1531 (merged) proposes lowering the latency alerting threshold for CI so that we can unsilence the alert without getting pager fatigue.

Tomasz Maczukin 🌴 @tmaczukin · 8 months ago Developer

> We should check whether or not we have alerting in place for all quotas, and also for exceeding quotas (not just approaching them).

For the API quotas `Read requests` and `Operation read requests`, which was the cause of production#1249 (closed), we don't have alerting. The `CICDGCPQuotaCriticalUsage` alert is using Prometheus metrics exported by https://gitlab.com/gitlab-org/ci-cd/gcp-exporter. This is a custom executor that we've created when we've started the CI migration from DigitalOcean to Google Cloud Platform.

Unfortunately, I was unable to find any way to query the current limit and usage for the `Read requests` and similar quotas. A lot of quotas that we can see via GCP's web GUI are not accessible with an API. To have alerts, we need to find a way to access the measurements.

Andrew Newdigate @andrewn · 8 months ago Maintainer

@tmaczukin for this, I think it would be reasonable to request that Google Cloud to allow quotas to be exported to stackdriver metrics.

From there, we could pick them up and push them onwards to Prometheus.

Once we have this, I we could model these quotas as saturation points, this will also give us notice when we are approach quota limits in future, through the capacity planning dashboards.

Tomasz Maczukin 🌴 @tmaczukin · 8 months ago Developer

Seems like a plan :)

Please register or sign in to reply

Craig Furman @craigf marked this issue as related to production#1249 (closed) 8 months ago

Craig Furman @craigf added IncidentReview label 8 months ago

Craig Furman @craigf assigned to @craigf 8 months ago

Craig Furman @craigf mentioned in merge request gitlab-com/runbooks!1531 (merged) 8 months ago

David Smith 🌴 @dawsmith changed title from **RCA for 2019-10-15 High latency for ci-runners** to **Incident Review for 2019-10-15 High latency for ci-runners** 8 months ago

David Smith 🌴 @dawsmith changed the description 8 months ago

David Smith 🌴 @dawsmith marked this issue as related to #3158 8 months ago

David Smith 🌴 @dawsmith · 8 months ago Owner

Noting discussion on #3158 as a corrective action related to dockerhub follow up issues.

David Smith 🌴 @dawsmith changed the description 8 months ago

David Smith 🌴 @dawsmith changed the description 8 months ago

David Smith 🌴 @dawsmith changed the description 8 months ago

David Smith 🌴 @dawsmith changed the description 8 months ago

David Smith 🌴 @dawsmith · 8 months ago Owner

Another question for the team: @andrewn @tmaczukin @craigf - should we look at running CI in multiple regions? More open ended question is - would running in multiple regions have helped mitigate this kind of problem? Are the costs in complexity and performance worth thi

Closed **Incident Review for 2019-10-15 High latency for ci-runners**

Collapse replies

Andrew Newdigate @andrewn · 8 months ago Maintainer

@dawsmith: @edjdev raised this point in the availability and performance meeting on Tuesday night and there was general agreement that this should be done: link to item in agenda here: https://docs.google.com/document/d/1SanPUz86cIyRQR5kRmXyCLLE8sZVpx0auu_W6jY94W4/edit#bookmark=id.r454tw5ybzzg

Craig Furman @craigf · 8 months ago Owner

The quotas that was throttled (GCE API access) were not regional and so multi-regional CI would not have helped in this particular incident.

@tmaczukin please correct me if I'm wrong, but in I think running in multiple regions can be accomplished by configuring new and/or existing runner managers to provision VMs in a different subnetwork, in a new region.

Running in multiple regions sounds sensible in general though. @andrewn is there an appetite for reducing our peak VM count in us-east1, or aiming to keep that roughly the same but creating new runner-managers configured to provision job VMs in other regions? I ask because this overlaps with our work on Cloud NAT (https://gitlab.com/gitlab-com/gl-infra/infrastructure/issues/7960#note_229627390). The IPs assigned to the NAT must be GCE "regional" static IPs. These are pinned to a region and cannot be moved.

While we're talking to Google about the possibility of reserving a contiguous block for CI in us-east1, this would be an ideal time to estimate capacity in other regions and get more blocks in those.

Andrew Newdigate @andrewn · 8 months ago Maintainer

> @andrewn is there an appetite for reducing our peak VM count in us-east1, or aiming to keep that roughly the same but creating new runner-managers configured to provision job VMs in other regions?

@craigf I suspect that the only reason you would want to do this would be as a backup, since (again, speculation at this point) the bandwidth costs of spreading this traffic around would be excessive.

@davis_townsend we could use the Google BigTable billing data to work out what out current CI bandwidth costs are, and then...

1. Calculate what sending (5%, 10%, 50%) of that traffic to another GCP region would cost
2. Calculate what sending (5%, 10%, 50%) of that traffic to AWS would cost

I suspect that it will be very expensive, hence @edjdev's suggestion of sending only a small amount of traffic to another location, with the ability to ramp it up if our another cloud provider issue occur in future.

Craig Furman @craigf · 8 months ago Owner

> bandwidth costs

I had forgotten to consider that. Are you referring to registry/api access from CI machines? That all goes via internet egress anyway, as we access these services via their regular public interfaces. I don't think moving region matters for this aspect, but is there another aspect you're referring to?

> sending only a small amount of traffic to another location

Sure, but this still grates on our egress IP story. We could either spend $ on provisioning IP blocks speculatively, or break customer firewall whitelists if we were to ramp up traffic in those regions. This is a product trade-off.

Anthony Sandoval @AnthonySandoval · 8 months ago Owner

We should be able to mitigate the networking cost by spinning up additional registries in each local region and keeping traffic internal to the GCP region. Yes, I realize that second part is a substantial amount of work, because as you say:

> That all goes via internet egress anyway.

But, why does it need to? I've seen multiple issues discussing routing internal API traffic to a dedicated fleet. We could quite easily test the viability of this strategy with registry. The application is stateless and GCS is available everywhere. Consequently, we should be able to run registry active-active in any region we desire.

Edit: I realize an IncidentReview issue is not the appropriate place to have this discussion. If there's interest in taking this on, we should open a separate issue.

Edited by Anthony Sandoval 8 months ago

Closed **Incident Review for 2019-10-15 High latency for ci-runners**

@andrewn It's hard to do a comprehensive cost analysis on this without understand all the interactions between these services and which parts are contributing to most of this cost today.

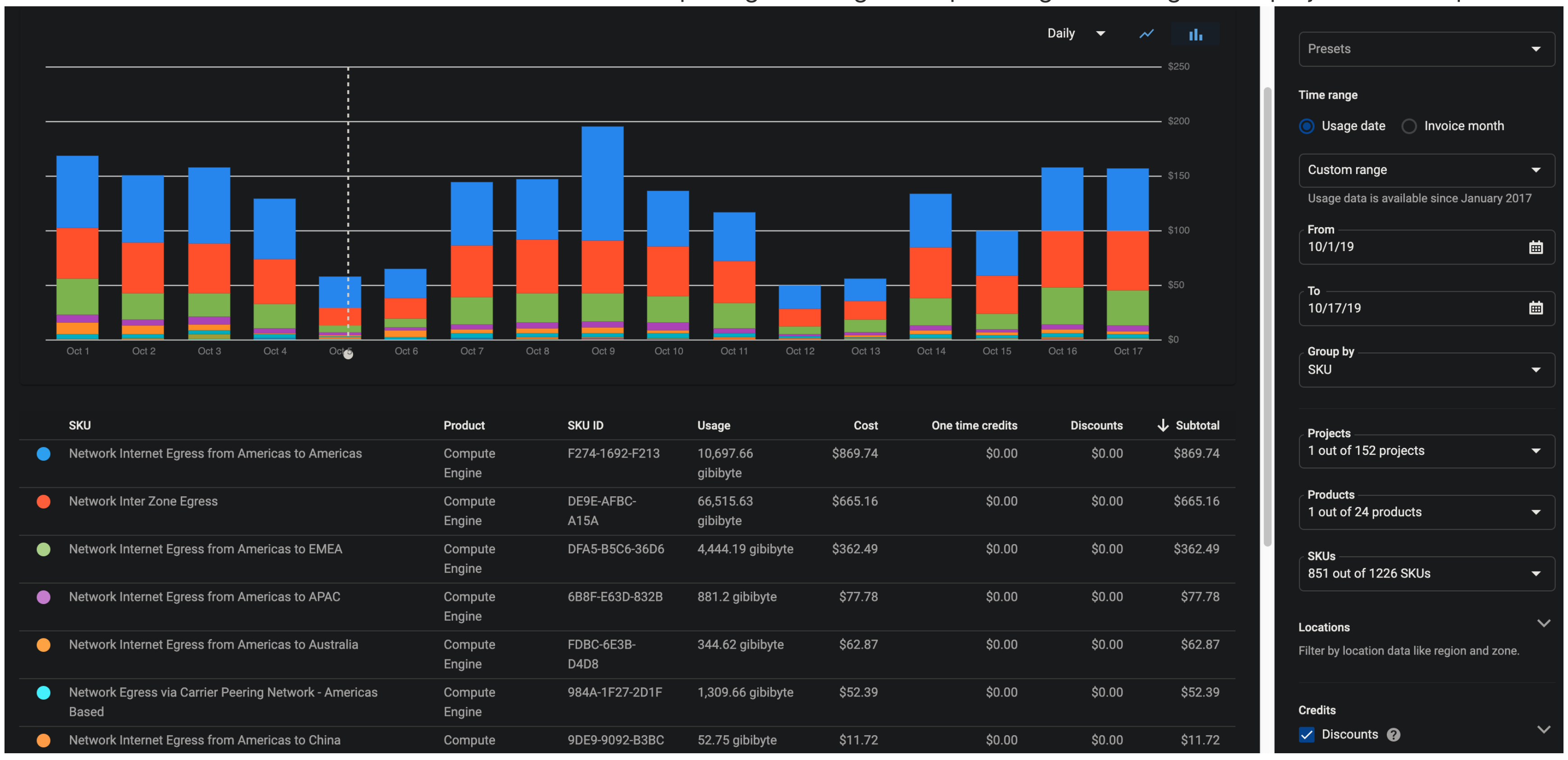
That being said, Anthony's point is spot on in terms of inter-zone rates being much more cost effective than egressing to the internet if possible. Some important rates for estimating cost to keep in mind when talking about an assumed scale of over 10TB/month are:

- $.01/GiB Inter-Zone Egress in Americas
- $.02/GiB Inter-Zone Egress in Europe
- $.08/GiB for Internet Egress for the major regions like Americas and EMEA
- $.08/GiB for Intercontinental Egress (Network Inter Region Egress from Americas to EMEA)

So if we're just talking about these internet egress costs switching to other regions or AWS, I actually don't think there would be much material impact as long as you're in the major regions in Americas or EMEA because we already get charged a pretty high rate of around $.08/GiB regardless (assuming we're using 10+ TB/month)

However, if you're talking about the services now having to use internet egress or intercontinental egress rather than the Network Inter Zone Egress, then we're talking about an additional $.07 per GiB that is transferred. Now I don't know how these services are setup, but for example if there's data being transferred from registry servers that Anthony mentioned from America to EMEA, then this may be an additional $.07/GiB both transferring this data out to the ci servers and then back into the registry servers, which could ultimately be an additional $.14/GiB for data to go back and forth.

Here's a screenshot of the current month network related pricing for Google Compute Engine in the gitlab-ci project if it's helpful:



	SKU	Product	SKU ID	Usage	Cost	One time credits	Discounts	↓ Subtotal
●	Network Internet Egress from Americas to Americas	Compute Engine	F274-1692-F213	10,697.66 gibibyte	$869.74	$0.00	$0.00	$869.74
●	Network Inter Zone Egress	Compute Engine	DE9E-AFBC-A15A	66,515.63 gibibyte	$665.16	$0.00	$0.00	$665.16
●	Network Internet Egress from Americas to EMEA	Compute Engine	DFA5-B5C6-36D6	4,444.19 gibibyte	$362.49	$0.00	$0.00	$362.49
●	Network Internet Egress from Americas to APAC	Compute Engine	6B8F-E63D-832B	881.2 gibibyte	$77.78	$0.00	$0.00	$77.78
●	Network Internet Egress from Americas to Australia	Compute Engine	FDBC-6E3B-D4D8	344.62 gibibyte	$62.87	$0.00	$0.00	$62.87
●	Network Egress via Carrier Peering Network - Americas Based	Compute Engine	984A-1F27-2D1F	1,309.66 gibibyte	$52.39	$0.00	$0.00	$52.39
●	Network Internet Egress from Americas to China	Compute	9DE9-9092-B3BC	52.75 gibibyte	$11.72	$0.00	$0.00	$11.72

Presets

Time range: [✓ Usage date] Invoice month

Custom range

Usage data is available since January 2017

From: 10/1/19

To: 10/17/19

Group by: SKU

Projects: 1 out of 152 projects

Products: 1 out of 24 products

SKUs: 851 out of 1226 SKUs

Locations

Filter by location data like region and zone.

Credits

[✓] Discounts

Note: Pricing can be found here:

- GCP: https://cloud.google.com/compute/network-pricing#general_network_pricing
- AWS: https://aws.amazon.com/ec2/pricing/on-demand/

Craig Furman @craigf · 8 months ago Owner

@ansdval I agree it would be nice for CI to access our API and registry internally, and in fact I created an issue for it a while back: #7948. If you are aware of duplicates we should consolidate them.

We would need to be very careful though: if we VPC-peered CI and gprd, arbitrarily code running in CI VMs could reach out into our gprd private network. Our firewall rules are not sufficiently hardened against this sort of thing - only against public internet traffic. I'm not sure it's worth introducing this risk.

Please register or sign in to reply

Craig Furman @craigf · 8 months ago Owner

@dawsmith answers to the questions in the description. Some of this duplicates #8176 (comment 230741524).

All times UTC.

Did alarming work as expected?

Our CI latency SLO alert, which used to page when the p80 for the time taken to begin running a scheduled job was over 60s, was silenced by me. The alert had been extremely noisy and there were known issues with the "hot pool" size and even the way we

As a corrective action we have lowered the SLO alert to p50 (gitlab-com/runbooks!1531 (merged)), which while far from our desired SLO, allows us the on-call SRE to distinguish signal from noise, and short-term emergencies from longer-term latency issues.

> How long did it take from the start of the incident to its detection?

As seen in charts posted elsewhere, our GCE API usage dropped sharply from ~10K reqs per 100s to 2k, the quota limit, at around 08:30. Our CI pending job queue often builds and is drained, and subjectively it did not appear to be particularly higher than normal until 09:00-09:30. Therefore the start of the incident is either 08:30 or 09:00-09:30, depending on how you look at it.

It was detected (by reports from users) at 10:53. So the start-detection time was either 2.5 or 1.5 hours.

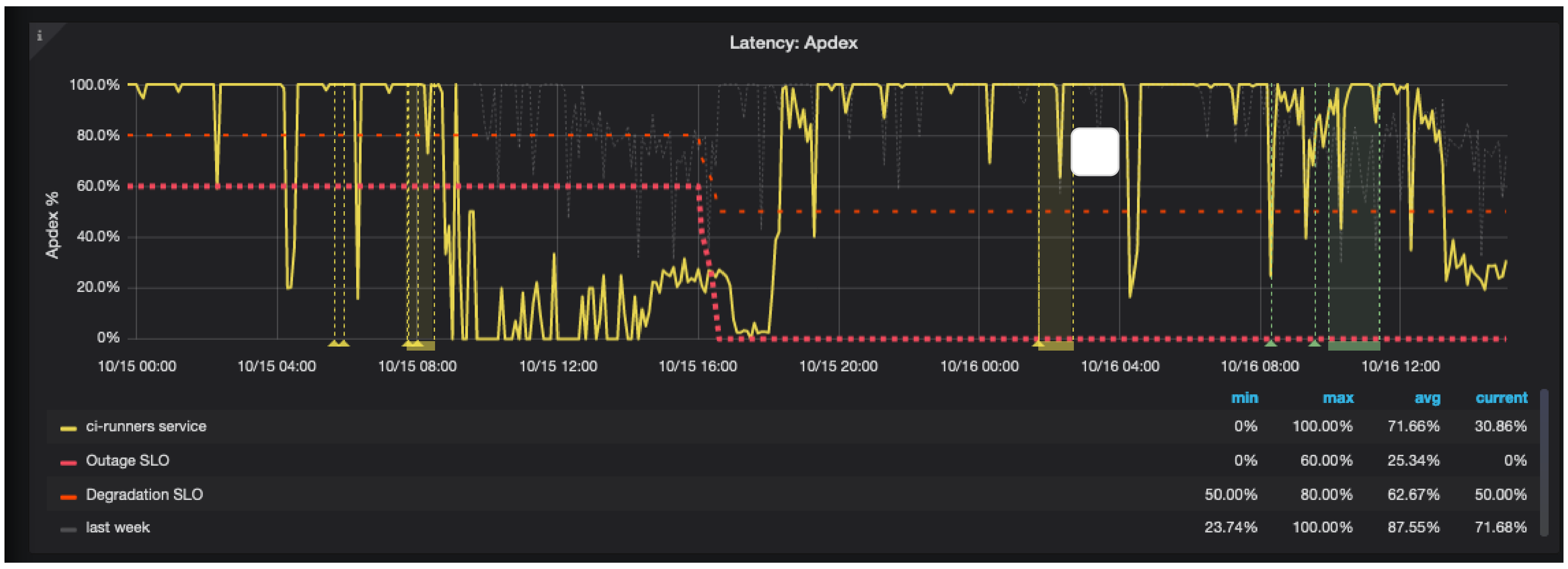
> How long did it take from detection to remediation?

We quickly diagnosed the cause as API throttling, and by 11:10 had contacted Google support asking them to raise the quota. Google support did not raise the quotas even back to our original setting of 12k each until about 15:00.

This caused our job queue to begin draining. The customer experience was still degraded as the queue was still large, latency was still very elevated across the board. This chart shows the beginnings of recovery at 15:00, and restoration of normal service at about 18:00.

Detection to partial remediation: 4 hours

Detection to full remediation: 7 hours



	min	max	avg	current
ci-runners service	0%	100.00%	71.66%	30.86%
Outage SLO	0%	60.00%	25.34%	0%
Degradation SLO	50.00%	80.00%	62.67%	50.00%
last week	23.74%	100.00%	87.55%	71.68%

> Were there any issues with the response to the incident? (i.e. bastion host used to access the service was not available, relevant team member wasn't page-able, ...)

See #8176 (comment 230741524) for a fuller explanation, but in summary:

start-to-detection time could have been improved had the SRE on-call (me 😬) not silenced a latency alert. As stated above the new threshold should allow us to distinguish true emergencies such as this incident from less-than-ideal-but-transient performance issues.

There is nothing we could have done ṙ ẤẃǴḲǴẎ to improve resolution time. We identified the underlying issue very quickly, but the power to resolve it was entirely in Google's hands.

David Smith 🌴 @dawsmith changed the description 8 months ago

Craig Furman @craigf · 8 months ago Owner

I created gitlab-org/ci-cd/docker-machine#8 as a possible corrective action that could mitigate the effects of a repeat of this incident, and if the investigation proves fruitful it would improve our performance near the edge of our quota anyway, which is necessary to keep scaling.

I'm going to unassign myself from this as I'm not really working on it at the moment. Of the problems I outlined in #8176 (comment 230741524) there are no more corrective actions that I can think of (as sadly we can't currently query our API quotas in order to alert on them).

Craig Furman @craigf unassigned @craigf 8 months ago

Andrew Newdigate @andrewn mentioned in issue #8215 8 months ago

David Smith 🌴 @dawsmith changed the description 8 months ago

Closed **Incident Review for 2019-10-15 High latency for ci-runners**

Rachel Nienaber @rnienaber · 8 months ago Developer

Adding wg-isolation here because all customers using ci-runners were impacted. Even though the root cause was out of our control (API quota adjustmenet) we may still want to discuss if there is anything we need to consider for this item.

Edited by Rachel Nienaber 8 months ago

Rachel Nienaber @rnienaber added wg-isolation label 8 months ago

Anthony Sandoval @AnthonySandoval closed 4 months ago

ops-gitlab-net 💬 @ops-gitlab-net mentioned in issue #9192 (closed) 4 months ago

Please register or sign in to reply