Closed Opened 1 year ago by David Smith 🌴

RCA for 2018-10-24 Incident - back up of Merge Request Queue processing in Sidekiq

Please note: if the incident relates to sensitive data, or is security related consider labeling this issue with security and mark it confidential.

Copying information from:
https://docs.google.com/document/d/1yWJFbf3z7PBgdxQAxrRPFg0B2fkUjPFZY39G6QmHXU8/edit#heading=h.si79cynhlbp7

Summary

Working notes from the day(s) of the event:

Starting from issue: #5215 (closed)
Slack threads: 1st thread: https://gitlab.slack.com/archives/C101F3796/p1540390946000100 2st thread: https://gitlab.slack.com/archives/C101F3796/p1540404258000100 3st thread: https://gitlab.slack.com/archives/C101F3796/p1540405525000100 Root cause thread: https://gitlab.slack.com/archives/C101F3796/p1540409583000100
https://gitlab.slack.com/archives/C101F3796/p1540412033000100

Contributors: Stan, Valery, Northrup, Mike Kozono, Dave, Jose

Service(s) affected : Team attribution : Minutes downtime or degradation :

Impact & Metrics

Start with the following:

- What was the impact of the incident? (i.e. service outage, sub-service brown-out, exposure of sensitive data, ...)

Actions that normally finish in a few seconds, but which are delayed over several minutes, appear to be broken until they finish. Some actions were run multiple times.

- Who was impacted by this incident? (i.e. external customers, internal customers, specific teams, ...) All GitLab.com users
- How did the incident impact customers? (i.e. preventing them from doing X, incorrect display of Y, ...) Actions that normally finish in a few seconds, but which are delayed over several minutes (IDK what the peak delay was), appear to be broken until they finish. E.g.: Merge requests diffs not being updated Merge request widgets not updating Repo mirrors not working

Some actions were run multiple times, e.g.: Many system notes were duplicated

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

Include any additional metrics that are of relevance.

Provide any relevant graphs that could help understand the impact of the incident and its dynamics.

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 memeber wasn't page-able, ...)

{{The first alert

Mirror updates overdue

Sidekiq availability and service operation rates

UpdateMergeRequestsWorker running jobs

UpdateMergeRequestsWorker errors

Postgres locks

Sidekiq queues https://dashboards.gitlab.net/d/9GOIu9Siz/sidekiq-stats?orgId=1&panelId=3&fullscreen&from=1540375200000&to=1540418400000

Detection & Response

How was the incident detected? PullMirrorsOverdueQueueTooLarge alert https://gitlab.slack.com/archives/C101F3796/p1540398268000100 and reports to #production, and issues being created Did alarming work as expected? Yes and no Yes - PullMirrorsOverdueQueueTooLarge worked No - Further alarming around the system operating outside of normal - pressure on Redis and other areas only was going to #alerts-general - SRE thinks we are ready to push some of those up to alerting via PagerDuty How long did it take from the start of the incident to its detection? The alarm was the first warning

How long did it take from detection to remediation? About 42 minutes until we shut off the feature and recycled the workers, but the queues took another 3 hours to drain due to duplicate jobs locking rows and being killed and retried.

Were there any issues with the response to the incident? No, it just took time to diagnose and mitigate, and at the moment (6 hrs after detection) we still don't have a root cause. }}

Timeline

2018-10-24

14:29 UTC - Feature flag for gitlab_sidekiq_reliable_fetcher was enabled in production and staging.

16:11 UTC - John Northrup has noticed that "pipeline_processing:update_head_pipeline_for_merge_request" queue is growing. 16:24 UTC - We got the first alert "PullMirrorsOverdueQueueTooLarge". 16:59 UTC - Feature flag for gitlab_sidekiq_reliable_fetcher was disabled in production and staging.

17:24 UTC - Alert manager has reported that "PullMirrorsOverdueQueueTooLarge" is resolved 17:29 UTC - We've got a report from the GitLab team member that it takes too much time for MR to be updated 18:04 UTC - It still takes several minutes for MR to be updated 18:23 UTC - We see lots of UpdateMergeRequestsWorker jobs that fails due to "PG::QueryCanceled: ERROR: canceling statement due to statement timeout" 18:52 UTC - We tweet the message to GitLab.com status https://twitter.com/gitlabstatus/status/1055170090839097344 20:31 UTC - We tweet that we're back to normal 21:36 UTC - We noticed that there were duplicated jobs https://gitlab.slack.com/archives/C101F3796/p1540417012000100?thread_ts=1540409583.000100&cid=C101F3796 since multiple Sidekiq workers were processing the same job JID at the same time

Root Cause Analysis

Current theory: A bug in the reliable fetch gem gitlab-org/sidekiq-reliable-fetch#5 (closed) caused jobs to be duplicated each time a Sidekiq worker was started up. Certain kinds of duplicated jobs (UpdateMergeRequestsWorker in particular) caused too many SQL update queries that were targeted to the same record. It caused lots of locks in the database that in response caused a number of Sidekiq jobs to grow rapidly.

What went well?:

- We were alarmed about the problems and immediately started fixing it
- We were alarmed before we got any user report
- The situation has been stabilized pretty quickly

What went wrong?:

- We don't know the root of the problem yet. Probably because of a not sufficient monitoring and logging
- Regarding the new queue for update_head_pipeline_for_merge_request - did we miss something in communication with SRE / Engineering?
- We didn't declare an incident early - with no CMOC we didn't tweet every 15 minutes to give more frequent updates to our customers
- We didn't have any rollback option when we noticed the issue.This data migrations should be back compatible to a previous state.
- The sidekiq reliable fetch gem, which we forked off an inactive project, does not seem to have been run in an environment with more than one Sidekiq worker. We did not have any proof that it had been proven in a

production environment, so we should have treated it with more caution. More testing and comprehensive staging tests may not have caught this bug, but it would have increased our chances.

Where did we get lucky?

We got alarm notification from least expected place from

Open questions: We have identified the most likely cause of duplicate jobs, but it is not confirmed yet for sure gitlab-org/sidekiq-reliable-fetch#5 (closed)

Corrective actions

- We need to find out why some MRs have too many diffs https://gitlab.com/gitlab-org/gitlab-ce/issues/53153
- We have to find out why there were duplicates #5215 (closed)
- We need to reproduce the ReliableFetcher#requeue_on_startup doesn't work with multiple Sidekiq processes issue to confirm that it was the root cause. Then fix it. gitlab-org/sidekiq-reliable-fetch#5 (closed)

Guidelines

- Blameless RCA Guideline
- 5 whys

Linked issues 1

Relates to

2018-10-24 - Incident - back up of Merge Request Queue processing in Sidekiq
production#535

David Smith @dawsmith marked this issue as related to production#535 (closed) 1 year ago

David Smith @dawsmith closed 1 year ago

David Smith @dawsmith added unscheduled label 1 year ago

Please register or sign in to reply