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/20180728-eventlogging

< Incident documentation

Contents [hide]

Summary

On July 28th at around 15:48 UTC the Eventlogging Processors stopped working due to a regex used to recognize User Agent's patterns, taking too much time when applied to some Kafka events. The regex matching was keeping the thread consuming from Kafka busy, and causing the one doing heartbeats with the Kafka Brokers to wait for the Python GIL, eventually causing the consumer to fall out of the consumer group. In turn all the other Eventlogging Processor taking over were experiencing the same issue, falling off the consumer group as well and causing a complete stall of the Eventlogging's processing machinery. Ultimately, the impact was that Event Logging data processing was delayed for about 36 hours.

Eventlogging background

To understand the Eventlogging's architecture:

- Analytics/Systems/EventLogging
- Analytics/Systems/EventLogging/Architecture
- Analytics/Systems/EventLogging/Administration#Overview

All events sent by browsers (or other agents) to Wikimedia are collected on the Varnish caching hosts by Varnishkafka and sent to the Kafka Topic **eventlogging-client-side** (what we call raw events). Then on eventlog1002, special daemons called **Eventlogging Processors** consume events from the same topic, validate them, add information (like a categorization of the User Agent) and produce the new "refined" events to other Kafka topics, that will be consumed by other systems (that are not really relevant for this incident report). So the Processors are an essential part of the Eventlogging validation and data refinement process, and if they are stuck nothing is really produced to the Kafka Topics aimed for general consumption (by Data Analysts, etc..).

It is also important to specify that the Eventlogging Processors for historical performance reason don't use the same libraries for consuming and producing events from /to Kafka:

- Kafka Python is used for consuming events from the eventlogging-client-side topic (raw events).
- Confluent Kafka Python is used for producing "refined" events to various Kafka Topics (the ones aimed for general consumption).

Kafka background

The Kafka brokers use a setting called `connections.max.idle.ms` (default to 10min, we don't modify it) to ensure that idle TCP connections are closed after a certain amount of time if no traffic is registered. This helps Kafka cleaning up its file descriptors (and resources allocated) but it also causes consumers/producers to loose connections with Kafka if they are not active for 10 minutes.

Python specific considerations

Python's architecture leverages the concept of GIL for multi-threading applications. From Kafka Python 1.4.0 (we use 1.4.1), the Kafka Consumer heartbeat with the Kafka broker runs in a separate thread. As the following stack trace (grabbed via GDB during the outage) suggests, it seems that the main thread doing the regex parsing (Thread 1) was holding the GIL for a long time, preventing the Kafka Consumer thread (Thread 18) to send heartbeats to the Kafka broker and falling out of its Consumer group:

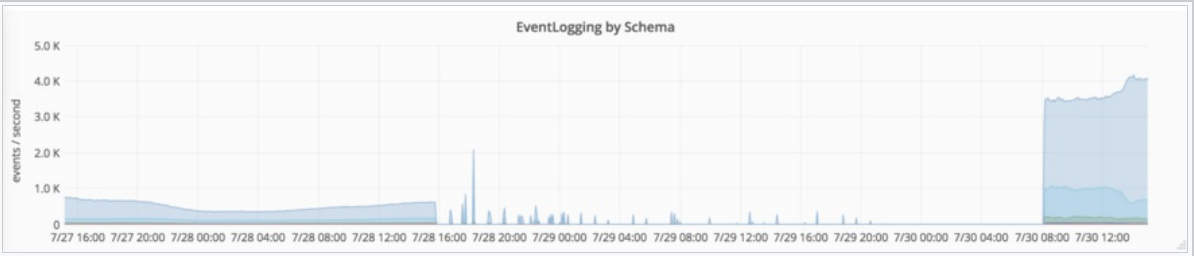
```
(gdb) thread apply all py-bt

Thread 18 (Thread 0x7f406e435700 (LWP 2053)):
Traceback (most recent call first):
  Waiting for the GIL
  File "/usr/lib/python2.7/threading.py", line 359, in wait
    _sleep(delay)
  File "/usr/lib/python2.7/dist-packages/kafka/coordinator/base.py", line 965, in _run_once
    self.coordinator._lock.wait(self.coordinator.config['retry_backoff_ms'] / 1000)
  File "/usr/lib/python2.7/dist-packages/kafka/coordinator/base.py", line 910, in run
    self._run_once()
  File "/usr/lib/python2.7/threading.py", line 801, in __bootstrap_inner
    self.run()
  File "/usr/lib/python2.7/threading.py", line 774, in __bootstrap
    self.__bootstrap_inner()

Thread 17 (Thread 0x7f406effd700 (LWP 2052)):
Traceback (most recent call first):

[..]

Thread 2 (Thread 0x7f40a683a700 (LWP 2037)):
Traceback (most recent call first):

Thread 1 (Thread 0x7f40acdd2700 (LWP 2032)):
Traceback (most recent call first):
  File "/usr/lib/python2.7/dist-packages/ua_parser/user_agent_parser.py", line 51, in Parse
    match = self.user_agent_re.search(user_agent_string)
  File "/usr/lib/python2.7/dist-packages/ua_parser/user_agent_parser.py", line 252, in
ParseUserAgent
    family, v1, v2, v3 = uaParser.Parse(user_agent_string)
  File "/usr/lib/python2.7/dist-packages/ua_parser/user_agent_parser.py", line 225, in Parse
    'user_agent': ParseUserAgent(user_agent_string, **jsParseBits),
  File "/srv/deployment/eventlogging/analytics/eventlogging/utils.py", line 405, in parse_ua
    parsed_ua = user_agent_parser.Parse(user_agent)
  File "/srv/deployment/eventlogging/analytics/eventlogging/parse.py", line 236, in <dictcomp>
    if k != 'capsule' and k not in capsule.keys()
  File "/srv/deployment/eventlogging/analytics/eventlogging/parse.py", line 232, in parse
    k: f(matches[k]) for k, f in caster_dict.items()
  File "/srv/deployment/eventlogging/analytics/bin/eventlogging-processor", line 129, in <module>
    event = parser.parse(raw_event)
```

Thread 1 was busy trying to match the user agent string with our list of regexes (that help to properly categorize a UA):

```
Thread 1 (Thread 0x7f40acdd2700 (LWP 2032)):
  46                              for group_index in range(1, match.lastindex + 1)]
  47            return match_spans
  48
  49        def Parse(self, user_agent_string):
  50            family, v1, v2, v3 = None, None, None, None
 >51            match = self.user_agent_re.search(user_agent_string)
  52            if match:
  53                if self.family_replacement:
  54                    if re.search(r'\$1', self.family_replacement):
  55                        family = re.sub(r'\$1', match.group(1), self.family_replacement)
  56                    else:
(gdb) py-locals
self = <UserAgentParser(v1_replacement=None, pattern=u'^(.*)/(\\d+)\\.?(\\d+)?.?(\\d+)?.?(\\d+)?
CFNetwork', v2_replacement=None, user_agent_re=<_sre.SRE_Pattern at remote 0x556186bd18b0>,
family_replacement=None) at remote 0x7f40a9de62d0>
user_agent_string = u'"Mozilla/5.0 (X11; Linux x86_64_128) AppleWebKit/11111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111..
[continue]
family = None
v1 = None
v2 = None
v3 = None
```

These regexes comes from the https://github.com/ua-parser/uap-python project.

Timeline



Graph showing the start and then end of the outage: a first big drop, from a previous steady state, is followed by little spikes and then finally recovery (more throughput due to a big backlog to process). The spikes in the middle were attempts of restart by puppet or from the operator, all ending up in the same hanging state while processing the "expensive" user agent strings.

Detailed timeline:

- 2018-07-28 15:48 - All Eventlogging Processors stopped working after trying to parse a very long user agent (causing a regex to take excessive amount of time to process due to backtracking).
- 2018-07-28 16:01 - First alarm fired for NavigationTiming schema's event throughput too low.
- 2018-07-28 17:15 - Luca (Analytics team) sees alarms on IRC and starts investigating.
- 2018-07-28 17:26 - First attempt to restart Eventlogging to see if the issue was temporary. After a brief recovery in Eventlogging's metrics, a sudden drop highlights the fact that a serious issue is ongoing.
- 2018-07-28 17:26 - Timo joins the investigation on IRC after seeing alarms for Navigation timing.
- 2018-07-28 19:11 - After reading a lot of logs and graphs, it is clear that something is blocking the Eventlogging Processors from pulling events from Kafka, but no indication about what it can be. Due to the fact that it was Saturday evening in Europe and that the Eventlogging issue was not impacting anything critical in production (except the Navigation timing's metrics), it was decided to restart the investigation the day after.
- 2018-07-29 09:09 - After reading an article about how to use gdb with Python on Debian, the Python stacktrace of all threads finally reveals itself in all its magnificence (before that it was only CPython low level C calls, not useful) and it is clear that a specific User Agent is responsible for this outage.
- 2018-07-29 09:33 - Other unrelated events cause pages to the SRE team and Riccardo the merciful agrees to spend his Sunday morning to help the Analytics team debugging Python regexes.
- 2018-07-29 14:00 - By this time it was clear that the regex causing the issue was prone to excessive backtracking in case of long strings, and a solution to break it down in multiple ones was proposed by Riccardo. While the solution was excellent, the Analytics team preferred to go for a temporary band aid (limiting the parsing of User Agents long maximum 800 chars) meant to restore the Eventlogging processing workflow and give time to the Analytics team to open a bug to the upstream package maintainers that keep the list of regexes (https://github.com/ua-parser/uap-core) to carefully vet Riccardo's solution before applying it (in order to avoid sneaky issues while recognizing/tagging User Agents). It was also decided to keep Eventlogging down until Monday morning (EU time) to have more people from Analytics to work on the band aid fix.
- 2018-07-30 08:03 - The Analytics team (Luca and Marcel) deploy https://gerrit.wikimedia.org/r/#/c/eventlogging/+/449121/ to eventlog1002 (after testing in deployment-prep) and Eventlogging restart processing events at full speed.

Conclusions

We rely on external libraries to recognize and match User Agents while processing Eventlogging data, and this can lead to unexpected failures like this one. The Analytics team need to find a way to apply pattern matching to User Agent in a time bound fashion, so that all the corner cases that may end up taking a ton of processing time will be skipped after a fixed amount of time is elapsed.

Links to relevant documentation

- Analytics/Systems/EventLogging
- Analytics/Systems/EventLogging/Architecture
- Analytics/Systems/EventLogging/Administration#Overview
- Analytics/Systems/EventLogging/Administration

Actionables

- Status: **Pending** - Set a timeout for regex parsing in the Eventlogging processors - phab:T200760
- Status: **Done** - Simplify and document how to increase log verbosity/level for Eventlogging - phab:T200765
- Status: **Done** - Open a upstream task to uap-core to review their regexes - https://github.com/ua-parser/uap-core/issues/332

Category: Incident documentation