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/20190402-0401KafkaJumbo

< Incident documentation

Summary

On April 2nd a pyspark (Apache Spark on Python) job launched on Hadoop caused two data-loss events for Varnishkafka. The job was using a very old Kafka client protocol, forcing the Kafka brokers to use extra heap memory to convert Kafka messages from one protocol to the other, eventually causing Java OutOfMemory exceptions. This instability in Kafka brokers caused lag in replication and temporary unavailability of some Kafka topic partitions, that caused Varnishkafka delivery failures (worst possible event for Analytics, since it means data not-delivered/dropped). The recovery was done after a (partial) roll restart of the Kafka Jumbo cluster. This action left the cluster into a suboptimal state, namely imbalance in number of topic partitions leadership role assigned to brokers; it was eventually fixed a day later with a manual rebalance of the cluster via the kafka `preferred-replica-election` command.

Post mortem of this event with plenty details is here: https://phabricator.wikimedia.org/T219842

Background

On every caching host (cpXXXX) there are one or more Varnishkafka daemons that read HTTP requests registered in the Varnish's shared memory log, format them following a json schema and then finally post them to the appropriate Kafka Jumbo topic. Several Analytics' Kafka consumers (like Eventlogging, Camus, etc..) will then pull the data to process it accordingly.

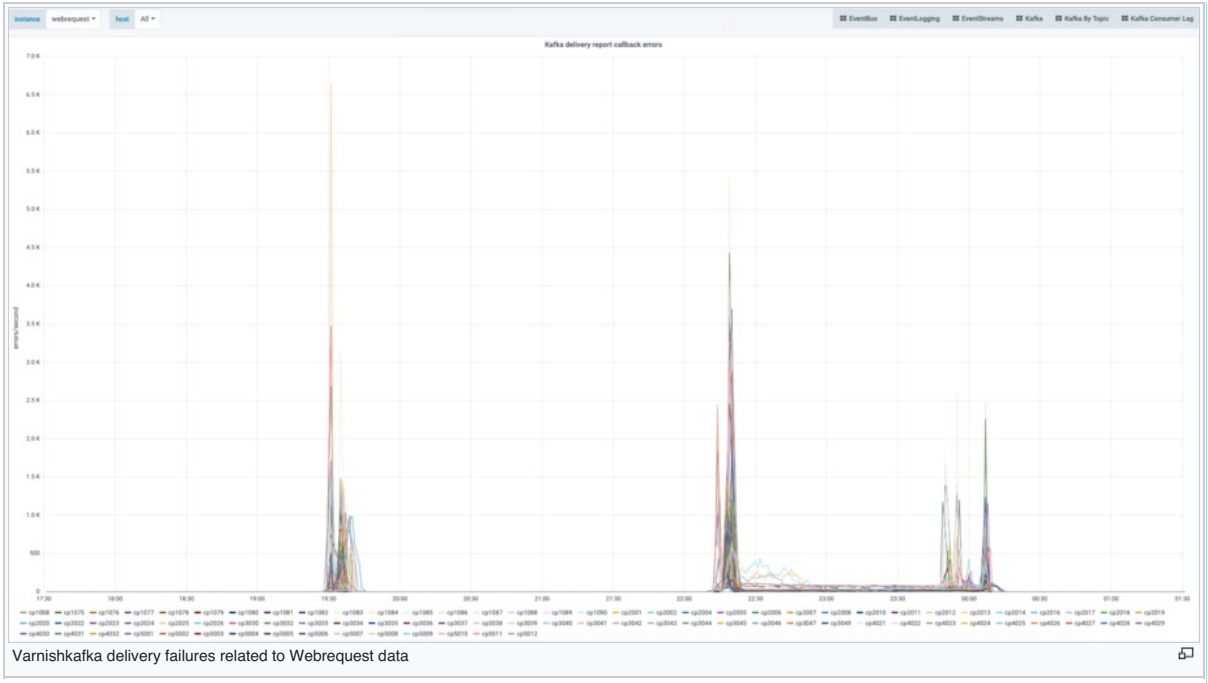
Varnishkafka leverages librdkafka to produce events/messages to Kafka, but it is an async client: librdkafka in fact immediately returns immediately to Varnishkafka when called using the async send API, but in reality all it does is simply adding the message to a queue and then send groups of events/messages in batches. If Kafka returns failures for a certain amount of time (either too many retries or the messages have been waiting for too long in the queue) then the delivery report failure callback is triggered, ending up in an increase in the related Graphite metric.

Impact

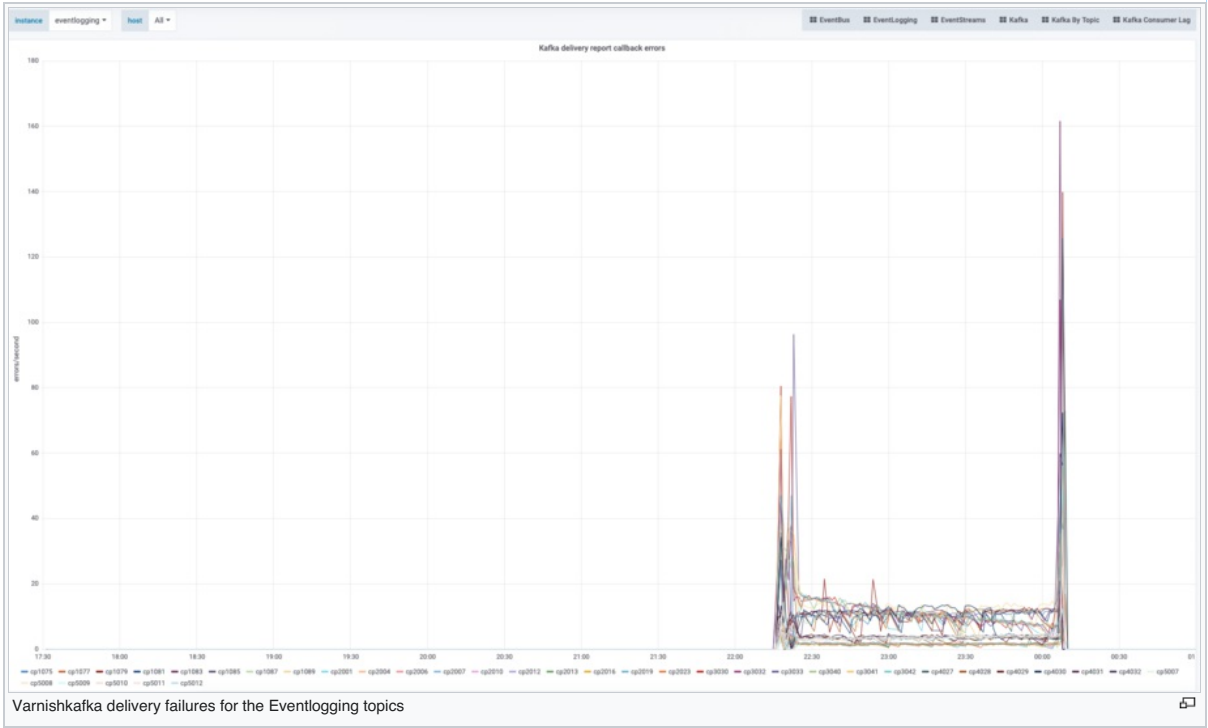
Partial loss of analytics events (eventlogging, event-bus and webrequest) for two time windows:

- 2019-04-01 19:28 -> 2019-04-01 19:45 UTC
- 2019-04-01 22:12 -> 2019-04-02 00:06 UTC

During these time frames the Kafka brokers were experiencing Java OutOfMemory exceptions, leading to replicas being not in sync and topic partition leaders not accepting traffic for brief amount of time. The major impact was of course to Kafka producers like Varnishkafka, that are also very sensitive to latency variations due to the amount of traffic handled. The loss is higher on Eventlogging's data (as per schema Kafka topics have only 1 partition), and smaller on Webrequest topics. Every callback error on this graph is an event that was not able to be posted to Kafka:



Varnishkafka delivery failures related to Webrequest data



Varnishkafka delivery failures for the Eventlogging topics

The Analytics team is currently working on a procedure to attempt to recover Eventlogging's data. This is possible since Varnishkafka produces events only to a topic called `eventlogging-client-side` on Kafka Jumbo (spanning several partitions of course). The Eventlogging daemons (running on eventlog1002) consume from that topic, identify and validate the events/messages and produce them to specific Kafka topics (corresponding to different Eventlogging schemas). Since the more specific Kafka topics have only one partition (replicated three times as the rest) it is easier that a broker failure causes more impact compared to a topic spanning multiple partitions. The recovery of data is possible up to the amount of data loss registered for the `eventlogging-client-side` topic of course.

Detection

Issue was detected by Icinga alarms at different levels: Varnishkafka delivery callback errors (producing events to Kafka Jumbo) and throughput alarms on Eventlogging topics. As the incident progressed, async/batch analytics jobs also reported errors in processing data.

Monitoring worked well in this case and, as Kafka cluster slowly recovered, so did the alarms.

Timeline

All times in UTC and happened on the 2018-04-01 (unless where the day is explicitly specified)

- [19:14 -> 19:34] Mjolnir's feature collection consumes data using KafkaRDD against a topic with 35 partitions (`mjolnir.msearch-prod-response`) using an old Kafka protocol/client version (0.8). This is a small test that leads to Kafka OutOfMemory events and some Varnishkafka delivery failures. No alert is fired, the Kafka Jumbo cluster self recovers.
- [21:35] Second and longer run of Mjonlir's feature collection starts (consuming again from Kafka using KafkaRDD and the old client).
- [22:14] Alarms start firing for Varnishkafka and Eventlogging throughput. This corresponds to the part of the Hadoop Job pulling data from Kafka. **OUTAGE BEGINS**

- [22:14 -> 22:20] The second run of the job caused a bigger impact since after 22:20 (when the Mjolnir's consumer stopped) two brokers were running in degraded mode (1002/1003). During this timeframe, the Kafka Jumbo Cluster was running in degraded mode, most of the events/messages produced to it are accepted but some of them are dropped as shown by Varnishkafka graphs. nuria, eberdnardson and mobrovac ping shdubsh on IRC as he is on PST timezone (it is night time in EU).
- [22:20 -> 23:28] metrics kept the same trend up to ~23:30, when the first restart of 1001 happened. Varnishkafka was still showing up timeouts, but a constant volume, not related to the peaks reached at 22:20. During this timeframe under replicated partitions was zero, and it spiked as soon as the first restart hit. Very interesting to notice that 1004 (acting as Kafka Controller) showed a offline count of several partitions, and we suspect it is the reason why Varnishkafka was still returning timeouts
- [23:28] (from SAL) restart kafka on kafka-jumbo1001
- [23:36] (from SAL) restart kafka on kafka-jumbo1002
- [23:47] (from SAL) restart kafka on kafka-jumbo1003
- [23:54] (from SAL) restart kafka on kafka-jumbo1004 (end of partial roll restart of the cluster, 1005 and 1006 are not restarted)
- [00:08] (2018-04-03) Varnishkafka data delivery failures stop and the related metrics drops to zero. We restart Kafka Mirror Maker running on all the Kafka Jumbo hosts. **OUTAGE ENDS**
- 2018-04-03 - elukey run a kafka preferred-replica-election on kafka-jumbo1001 to rebalance Kafka topic partitions leaders and reduce TLS latencies from Varnishkafka to Kafka Jumbo.

Considerations

There are several interesting considerations to make, most of them probably leading to follow ups:

- Varnishkafka's monitoring should have alarmed during the first event, probably the alert would have raised Analytics' attention before the main outage started.
- Kafka Jumbo brokers' heap size (2G) is probably not enough to handle out of the ordinary bursts of requests. Since Kafka it is a heavy user of the page cache we have always tried to leave as much memory as possible for it, keeping the heap size as small as possible. Having 4G of maximum heap size could be a good follow up probably.
- The Java OutOfMemory exceptions were triggered by an old client forcing Kafka to use extra heap size to convert data exchanged from one protocol to the other. We should investigate if Kafka is able to support only recent protocols rejecting old clients as protection measure.
- An investigation in T219842 led to the conclusion that the 1Gbps bandwidth provided by the Kafka Jumbo's NICs is not enough with the current level of traffic of the cluster. Daily traffic shows a lot of occurrences of 60/70% tx bandwidth saturated, with peaks of 80/90% for some seconds (shown by tools like `ifstat` that returns per second datapoints). The Hadoop job that caused the outage pulled data from a 35 partitions Kafka topic, and despite the evidence of complete bandwidth saturation in Prometheus metrics, it is easy to imagine (after what's written just before) that probably tx bandwidth saturation could have aggravated at least the start of the outage.
- Kafka Jumbo is clearly important from the Analytics team (and the foundation too), but there is no clear SLO (in SRE terms) formalized between SRE and Analytics.
- The Kafka dashboard stacked graphs, as noted in https://phabricator.wikimedia.org/T219842#5084186, were not ideal to spot imbalance in topic partitions leader assignment. Given the high latencies and the impact on the reliability of Varnishkafka, a new alarm is needed to get notified when a Kafka cluster is running in sub-optimal way.

Conclusions

There are clearly a lot of follow ups needed (as outlined in the above section) but there are also some nice things to consider:

- A good collaboration between Analytics / Core-Platform / SRE happened during the handling of the outage, leading to the service restored.
- Despite the fact that alarms/monitoring needs to be improved, we got notified soon about the issue and people were able to act quickly.

Spreading the knowledge about Kafka is also essential, the more SREs with experience on it the better for the future given how many important Kafka clusters we run nowadays.

Links to relevant documentation

https://wikitech.wikimedia.org/wiki/Kafka/Administration

Actionables

NOTE: Please add the #wikimedia-incident Phabricator project to these follow-up tasks and move them to the "follow-up/actionable" column.

- Convert mjonlir to use python Kafka consumer that supports newer protocol than 0.8 T219932
- Investigate if Kafka can decline requests from consumers with "old protocols" (old might be two minor versions behind) T219936
- Clear definition of Kafka Jumbo's SLO with the SRE team to increase support during the PST daytime.
- Tune Varnishkafka alerts to be more sensitive.
- File a request for 10G NICs to deploy on Kafka Jumbo nodes (and probably other two hosts/brokers to add next fiscal year). T220700
- Add non stacked graphs to https://grafana.wikimedia.org/dashboard/db/kafka to help people diagnose Kafka issues sooner [DONE]
- Add alarm to all Kafka clusters about imbalanced partition leadership assignment for a long amount of time.
- Backfill data [DONE] T220421

Category: Incident documentation