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/20190413-elasticsearch

< Incident documentation

Contents [hide]

1 Summary
  1.1 Impact
  1.2 Detection
2 Timeline
3 Conclusions
  3.1 What went well?
  3.2 What went poorly?
  3.3 Where did we get lucky?
4 Links to relevant documentation
5 Actionables

Summary

Search latency increased to the point where it had user impact and search was limited by pool counter. This happened twice. First instance was recovered by restarting elastic1027, second instance was recovered by switching part of the traffic to codfw.

Impact

All search queries reported significantly increased latency (2x-5x). Around 400 queries per minute were rejected with no search results out of total traffic of around 30k queries per minute.

Detection

Automated alerts on CirrusSearch latency first alerted the problem. Alerts for a downstream service, the recommendation api, also started flapping.

Timeline

This is a step by step outline of what happened to cause the incident and how it was remedied. Include the lead-up to the incident, as well as any epilogue, and clearly indicate when the user-visible outage began and ended.

All times in UTC.

2019-04-13

- *13:10 - 13:22 elastic1025 disk free declines from 20% to 11%*
- *13:17 shard recoveries increases from 8 to 18*
- *13:19 avg cpu usage across cluster increases from 40% to 60%*
- *13:20 user visible latencies start to increase*
- *13:24 shard recoveries increases from 12 to 32*
- *13:24 pool counter starts rejecting queries* **OUTAGE BEGINS**
- *13:34 - 13:43 elastic1023 disk free increases from 25% to 60%*
- *13:40 elastic1027 disk free increases from 25% to 50% (shards leaving?)*
- *13:45 elastic1026 and elastic1027 cpu pegged to 100%.* Many nodes across cluster showed elevated cpu usage.
- *14:00 (approx) _joe_ and shdubsh start look into flapping recommendation api alerts*
- *14:03 elastic1026 cpu usage returns to nominal levels*
- *14:22 - 14:47 - elastic1028 disk free increases from 20% to 85%*
- *15:48 shdubsh and _joe_ decide to elevate problem to search platform. ebernhardson recieved phone call from shdubsh.*

- *15:58 restart elasticsearch on elastic1027, cpu usage returns to nominal levels*
- *15:58 pool counter stops rejecting queries*
- *16:30 user visible latencies return to nominal levels* **OUTAGE ENDS**

2019-04-14

- *01:48 user visibile latencies start to climb* **OUTAGE RESUMES**
- *02:00 pool counter resumes rejecting queries*
- *02:00 elastic1027 pegs cpu to 100%*
- *02:26 elastic1027 cpu returns to nominal levels*
- *02:27 pool counter stops rejecting queries*
- *02:45 pool counter resumes rejecting queries*
- *03:27 pool counter stops rejecting queries*
- *03:48 elastic1023 pegs cpu to 100%*
- *03:54 pool counter resumes rejecting queries*
- *04:10 elastic1023 cpu returns to nominal levels*
- *04:51 ebernhardson starts looking into overload conditions*
- *04:58 shift most expensive query types (more_like and regex) for all wikis to codfw*
- *04:59 restart elasticsearch on elastic1027*
- *05:00 pool counter stops rejecting queries*
- *05:02 user visible latencies decline from 1200ms p95 to 600ms p95*
- *05:54 shift enwiki full text search traffic to codfw*
- *05:54 user visible latencies return to nominal levels (200ms p95)* **OUTAGE ENDS**

Conclusions

What weaknesses did we learn about and how can we address them?

The following sub-sections should have a couple brief bullet points each.

What went well?

- Automated alerting noticed the degraded user experience and send out alert emails
- Incidents resolved relatively quickly once search engineers were able to look at the problem
- In terms of total search traffic only a small fraction of requests were completely rejected. The incident was mostly a degraded, and not a broken, search experience.

What went poorly?

- Both incidents continued for multiple hours before being addressed
- Due to the incident occurring on a weekend no one was around to respond to alerts immediately
- The servers showing problems are known to be old and unlike the rest of the servers in the cluster. They are planned to be replaced Q1 of FY19-20, but perhaps they should have been replaced earlier.

Where did we get lucky?

- *for example: user's error report was exceptionally detailed, incident occurred when the most people were online to assist, etc*

Links to relevant documentation

Where is the documentation that someone responding to this alert should have (runbook, plus supporting docs). If that documentation does not exist, there should be an action item to create it.

Actionables

Explicit next steps to prevent this from happening again as much as possible, with Phabricator tasks linked for every step.

- Repair per-node elasticsearch latency metrics. These were not available during the incident and have been useful for diagnosing previous incidents. (merged)

Category: Incident documentation