Evolving Search

Elasticsearch at Kaufland e-commerce

Kaufland

e-commerce

Alexander Reelsen

alexander.reelsen@kaufland-ecommerce.com

Agenda

- Introduction
- Architecture
- Indexing
- Searching
- Running in production
- Improving relevance
- Performance surprises
- Summary

Introduction

Kaufland e-commerce

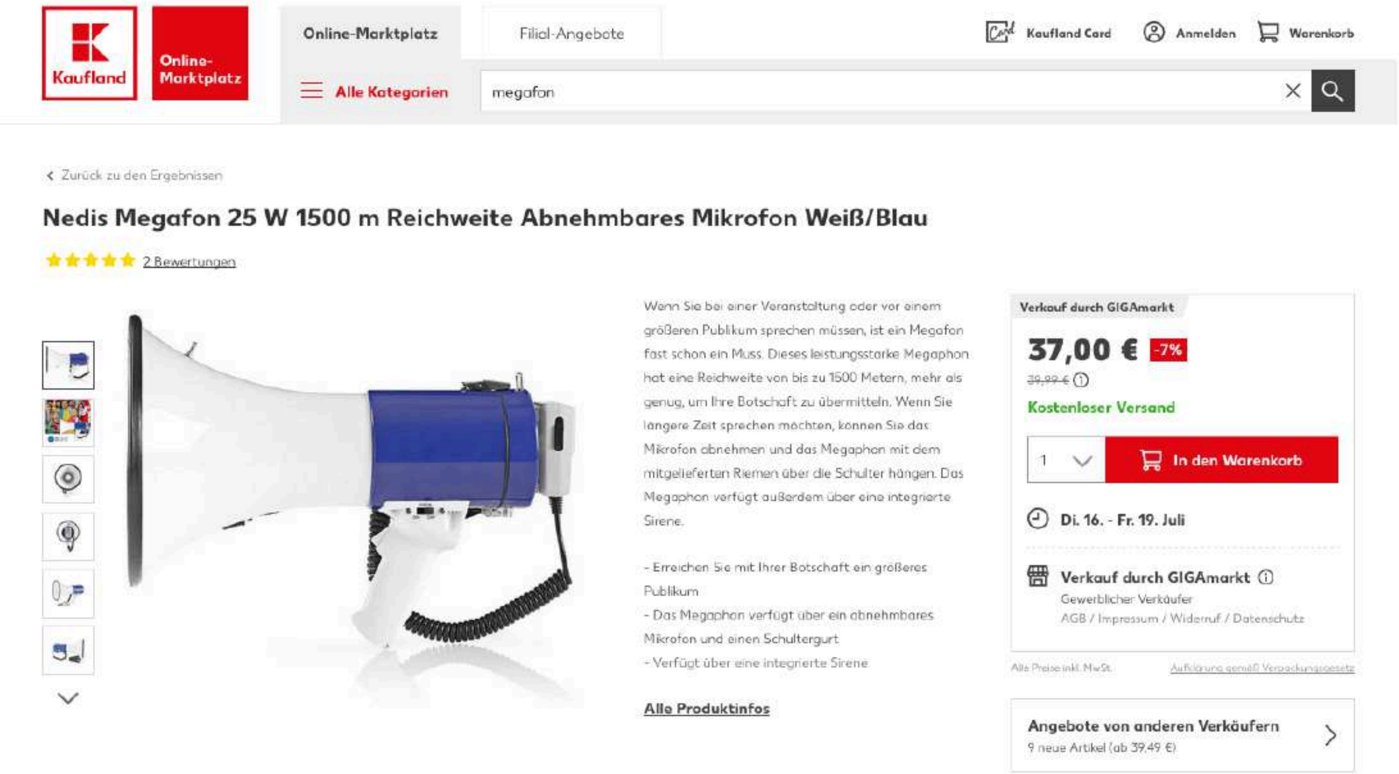
- Online B2C marketplace
- 11k sellers
- > 800 employees
- Five storefronts (de, cz, sk, pl, at)

About me

- Search Engineer @ Kaufland e-commerce
- Doing (Elastic)search since 2010
- Interested in all things scale & distributed
- Likes Java and the JVM, uses Typescript currently

Questions?

Questions?



Nedis Megafon 25 W 1500 m Reichweite Abnehmbares Mikrofon Weiß/Blau

Wenn Sie bei einer Veranstaltung oder vor einem größeren Publikum sprechen müssen, ist ein Megafon fast schon ein Muss. Dieses leistungsstarke Megaphon hat eine Reichweite von bis zu 1500 Metern, mehr als genug, um Ihre Botschaft zu übermitteln. Wenn Sie längere Zeit sprechen möchten, können Sie das Mikrofon abnehmen und das Megaphon mit dem mitgelieferten Riemen über die Schulter hängen. Das Megaphon verfügt außerdem über eine integrierte Sirene.

- Erreichen Sie mit Ihrer Botschaft ein größeres Publikum
- Das Megaphon verfügt über ein abnehmbares Mikrofon und einen Schultergurt
- Verfügt über eine integrierte Sirene

Verkauf durch GIGAmarkt

37,00 € -7%

~~39,99 €~~

Kostenloser Versand

Di. 16. - Fr. 19. Juli

Verkauf durch GIGAmarkt
Gewerblicher Verkäufer
AGB / Impressum / Widerruf / Datenschutz

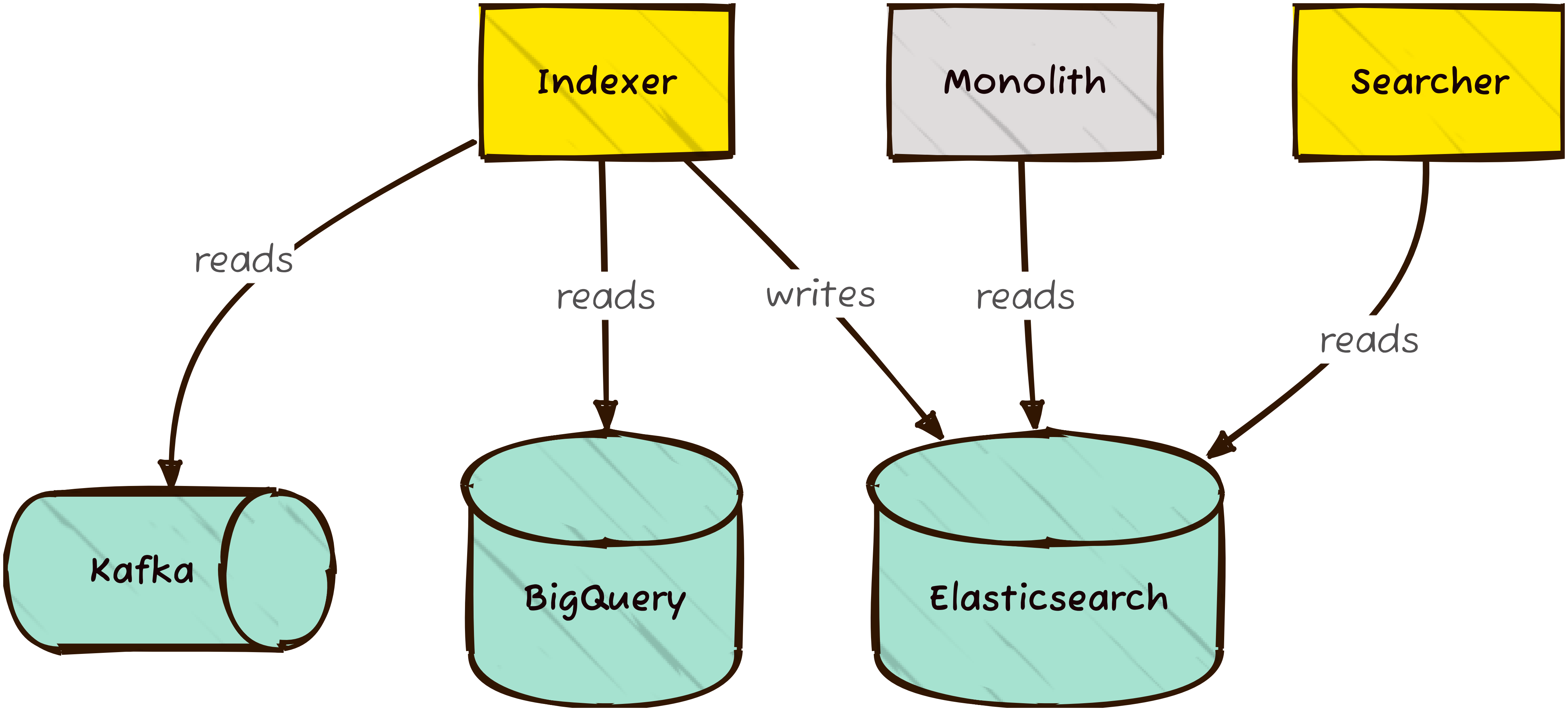
Alle Preise inkl. MwSt.
Aufklärung gemäß Verpackungsgesetz

Angebote von anderen Verkäufern
9 neue Artikel (ab 39,49 €)

Interrupt. All. The. Time.

Architecture

Architecture



Architecture

- Tech stack: K8s, gRPC, Typescript, Node, MongoDB, Redis, Elasticsearch, Kafka
- Monolith
  - Cronjobs that read from Elasticsearch
- SIB
  - reads continuously from Kafka
  - reads as cronjob from BigQuery
  - writes to Elasticsearch (thousands of updates per minute)

Architecture

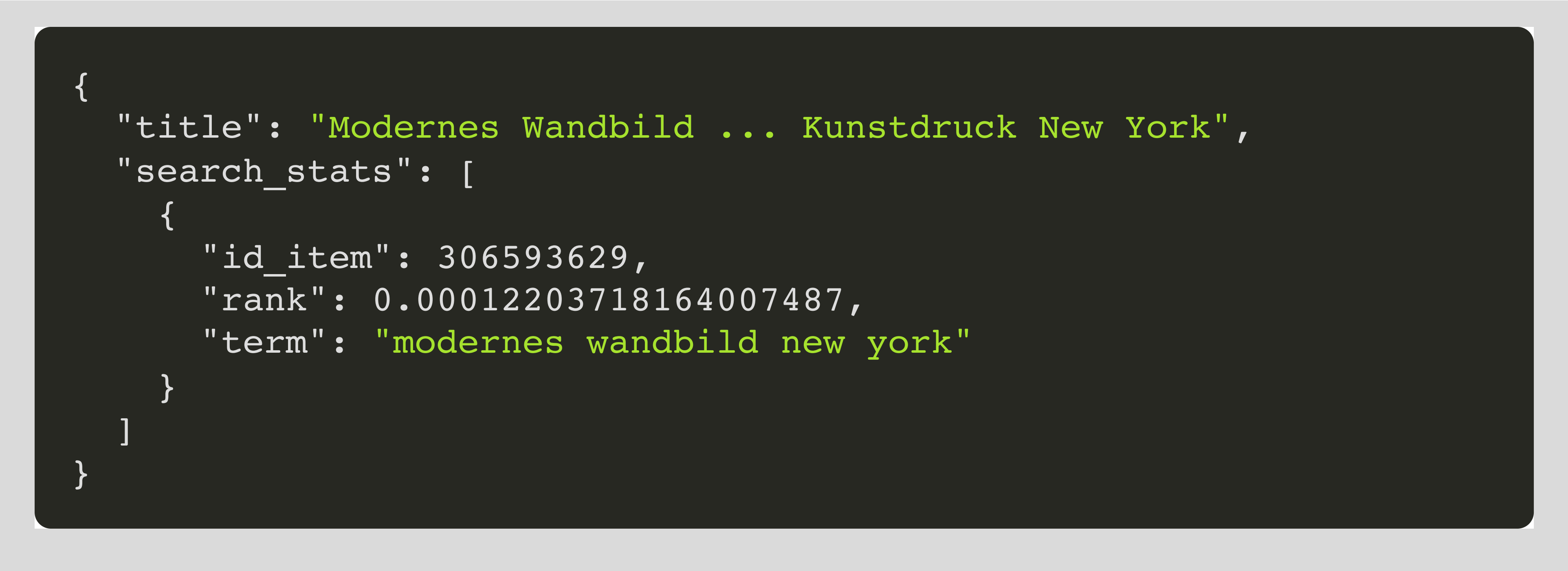
- SEB
  - Offers REST/gRPC endpoints for other Microservices
  - Search and Filtering/Facetting
  - Queries Elasticsearch
- Elasticsearch
  - Self hosted on GCP
  - Using ansible/terraform for deployment

Searches

- Phase 1: Analysis
  - Find shorter terms (agg)
  - Guess category (agg)
- Phase 2: Search
  - Product search: Find products (search)
  - Filter search: Find filters (aggs)

Phase: Analysis

- Use search learner data
- Information how users interacted with search terms and products
- Nested data structure



```
{
  "title": "Modernes Wandbild ... Kunstdruck New York",
  "search_stats": [
    {
      "id_item": 306593629,
      "rank": 0.00012203718164007487,
      "term": "modernes wandbild new york"
    }
  ]
}
```

Phase: Analysis - Category Guessing

- How do you guess a category id?
- Easy! Aggregate search on category id!
- 🚨 There are more iPhone accessoires than iPhones…
- Use search learner data, aggregate against the sum of the ranks!

Phase: Analysis - Shorter queries

- Use search learner data to find shorter queries
- `gummiseil gelb 8mm 20m`
  - What are the important parts of this query?
  - Can we extract the important parts?

Phase: Search

- `filter`:
    - Remove invalid products
    - Filter for shorter queries
- `must`:
    - term centric query
    - multiplied with `function_score`
        - `field_value_factor`: product rank
        - `script_score`: search learner rank

Search: FTS vs. signals

- Signals are strong in bigger store fronts
- FTS is stronger in smaller store fronts
- Tough to balance with a single query
- Do you really need FTS scoring, or just boolean matching?

Quiz: `minimum_should_match`

- What does `minimum_should_match: "2←1 5←2"` do?
- 1-2 terms: 100%
- 2-5 terms: One term does not need to match
- 6 or more terms: Two terms don’t need to match

Running in production

Cost reduction

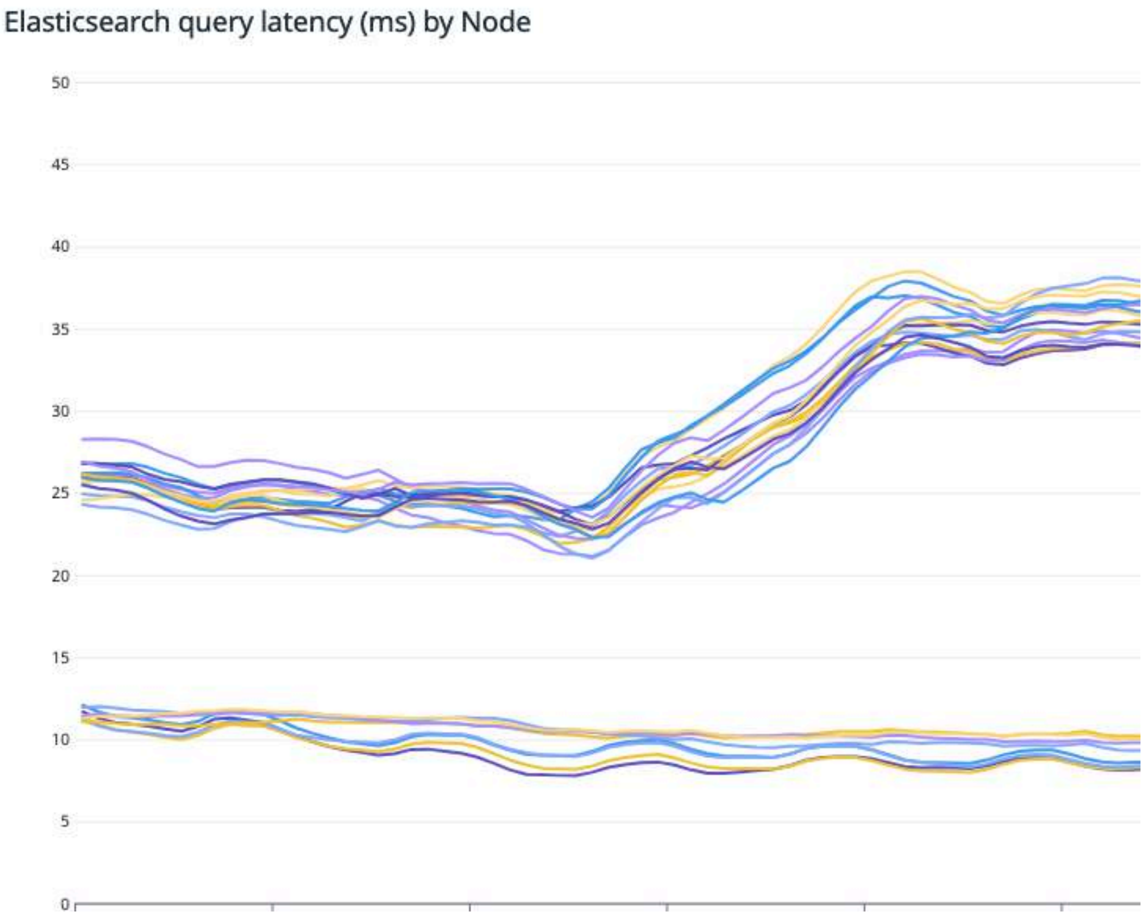
- Halved the data nodes
- Halved the cores
- Have enough room for A/B tests
  - 2nd storefront index with different mapping
- Services: Reduce k8s application waste

Journey to Elasticsearch 8

- Jan: 7.12.1
- Mar: 7.17.22
    - Monolith also talks to another 7.x cluster
    - Cronjob to clean caches was added, unfixed bug in 7.17
    - Introduced a bug that required restarts
- May: 8.14.1

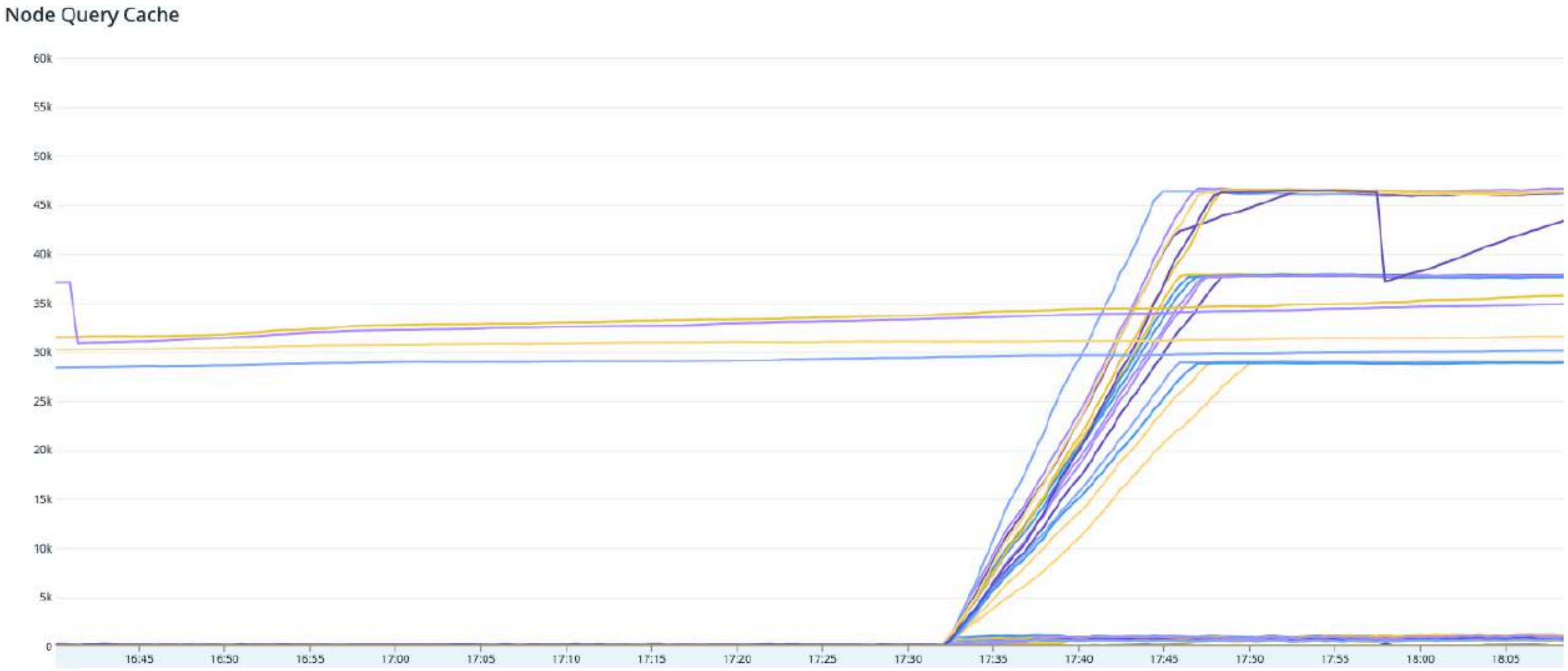
Node caches filling up

Increased query latency



Node caches filling up

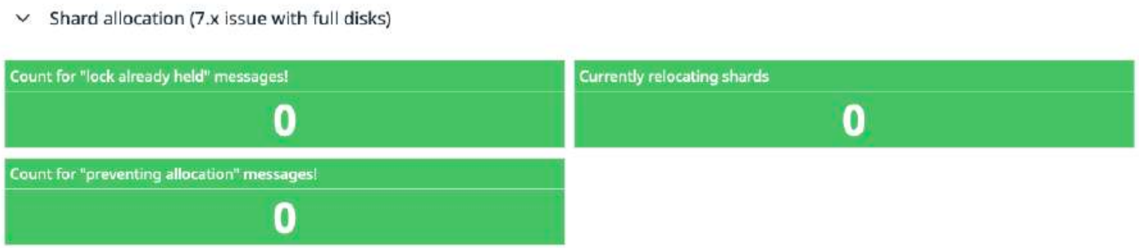
Node cache not used



unfixed bug in 7.17

Upgrade: Unreleased shard locks

- Shards lock was never released
- Could not be moved away
- Disk ran full due to other shards being moved around
- Solution: 8.x upgrade



Shard allocation (7.x issue with full disks)

Count for "lock already held" messages!

0

Currently relocating shards

0

Count for "preventing allocation" messages!

0

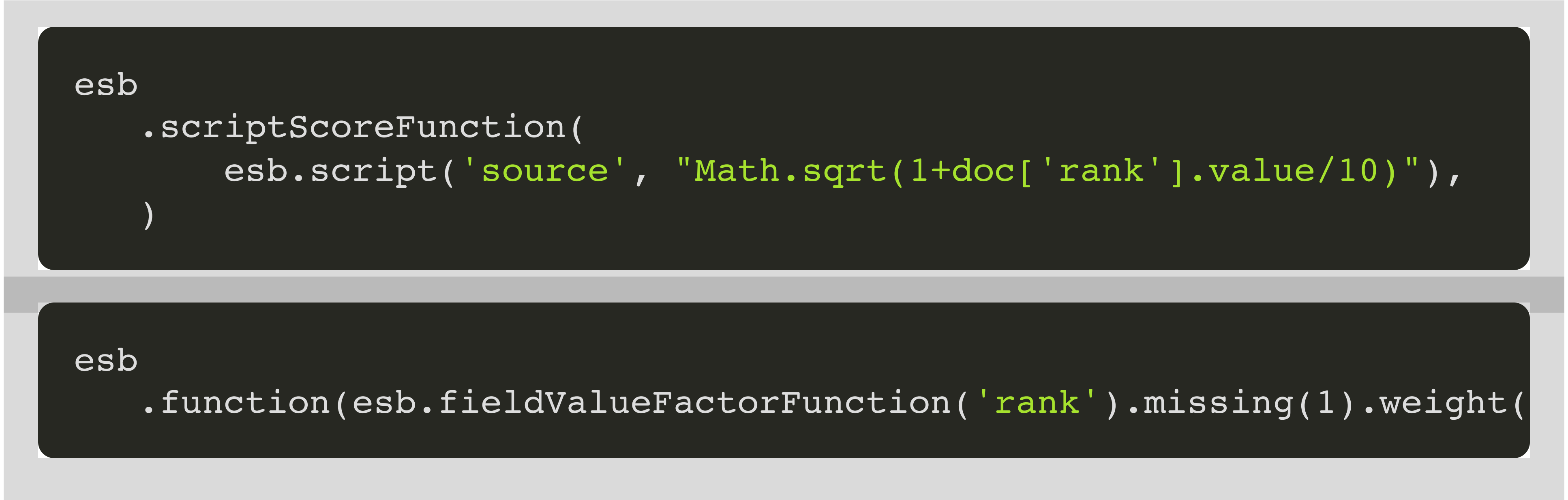
Performance

Performance

- Less scripting
- Unexpected refreshes
- Ensure `eager_global_ordinals`
- Range queries vs term queries
- `rank_feature` did not have any effect
- should/must/filter is not optimized - WAT?
- Switching off a microservice

Less scripting

- Check if scripting is needed, if not: remove!



```
esb
    .scriptScoreFunction(
        esb.script('source', "Math.sqrt(1+doc['rank'].value/10)"),
    )
```

```
esb
    .function(esb.fieldValueFactorFunction('rank').missing(1).weight(
```

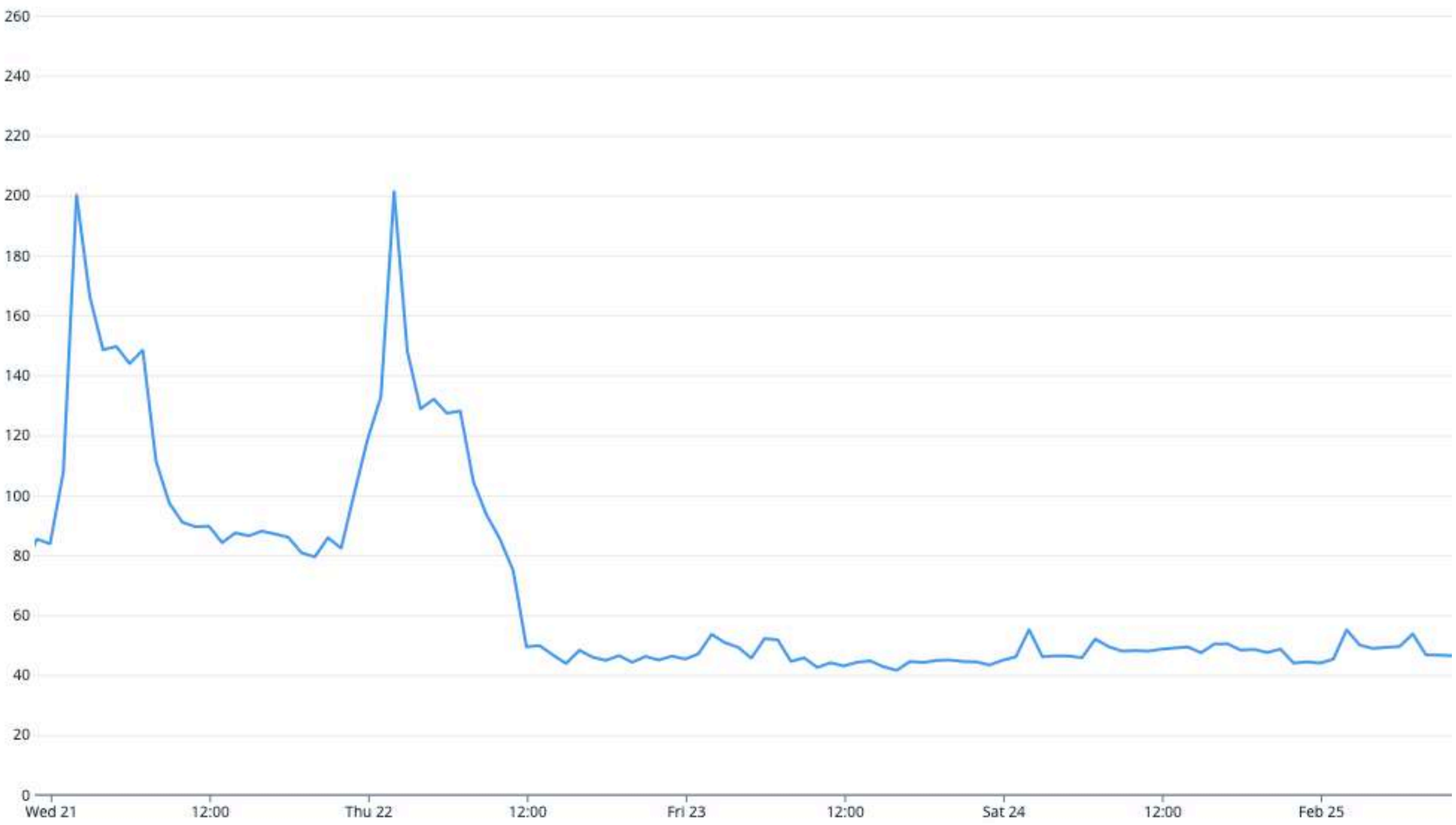
Unexpected refreshes

- `refresh_interval` was set to `5m`
- Dozens of refreshes happening within 5 minutes
- Realtime GET might need to refresh
- Update API calls realtime GET
- Translog location not available: Refresh
- After that: shard starts tracking translog location in-memory
- More on discuss forum

Set `eager_global_ordinals`

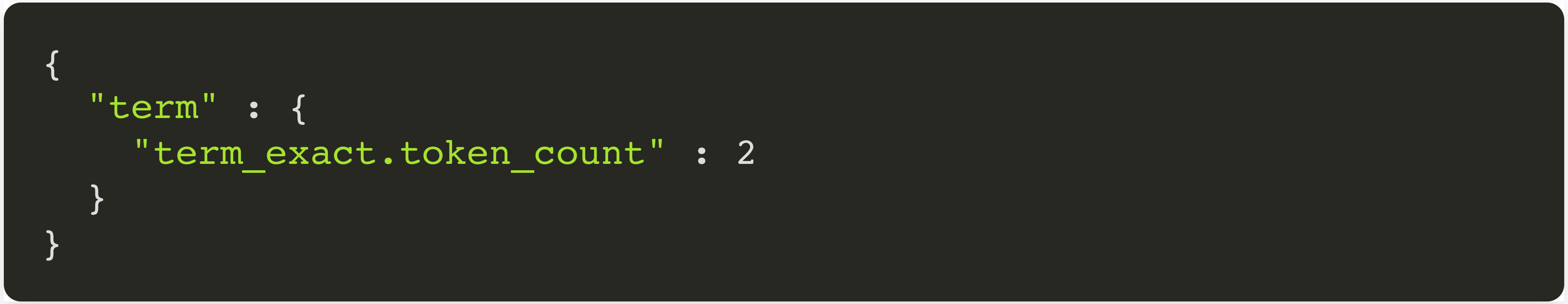
- `keyword` field aggregations
- map each term to an incremental int (ordinal mapping)
- per segment data structure, but requires a per-shard structure (global ordinals)
- created on query time
- 🚨 slow & unpredictable! 🚨

Set `eager_global_ordinals`



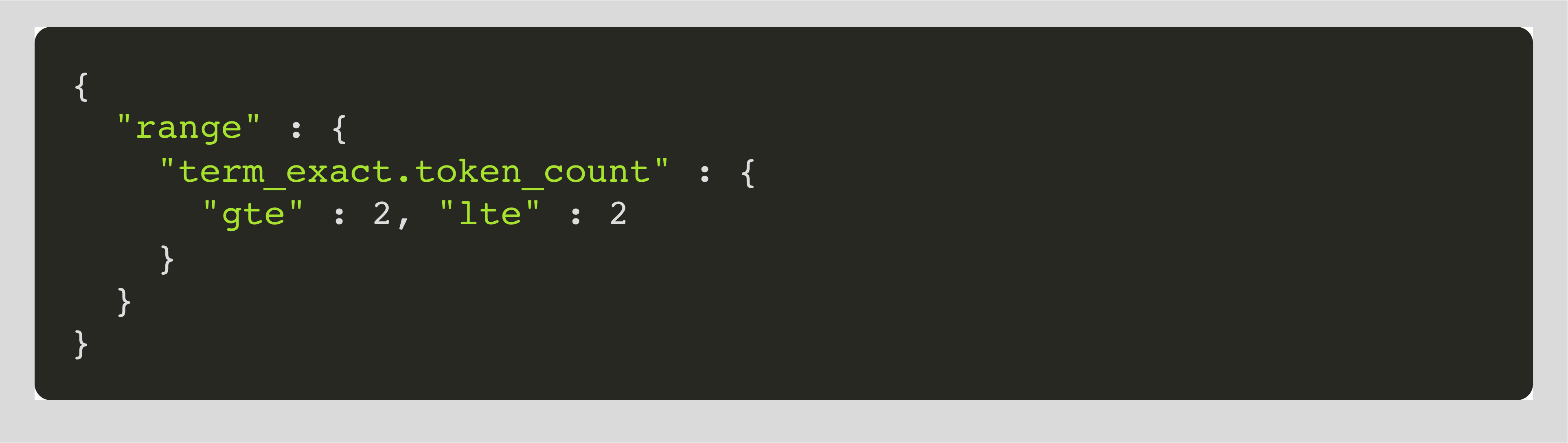
Range query vs. term query

What’s the difference?



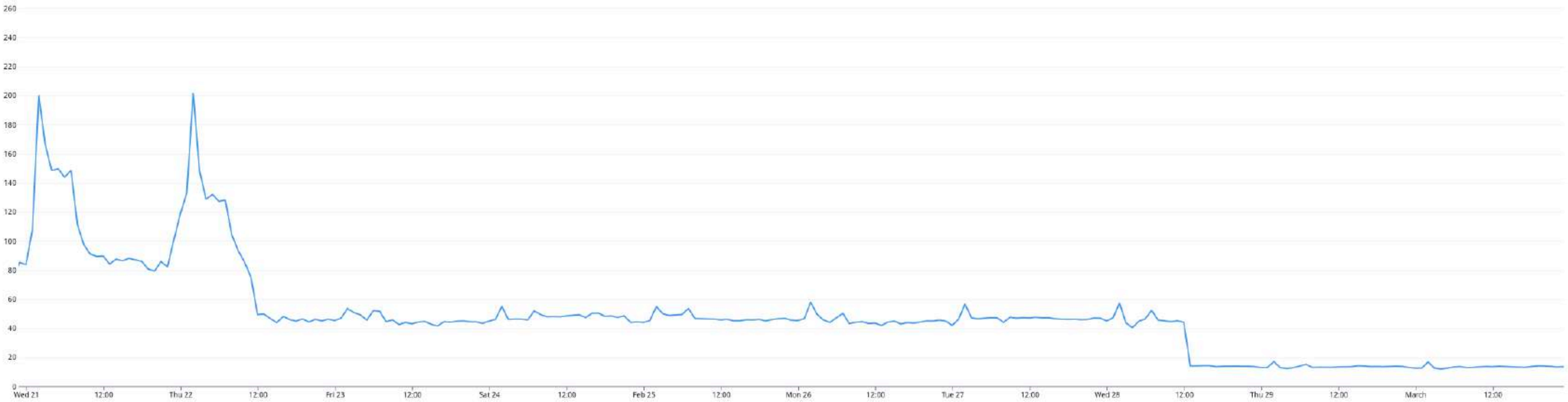
```
{
  "term" : {
    "term_exact.token_count" : 2
  }
}
```

vs.



```
{
  "range" : {
    "term_exact.token_count" : {
      "gte" : 2, "lte" : 2
    }
  }
}
```

Range query vs. term query



Seems to have been fixed in 8.x, no answer on discuss forum

`rank_feature` has no effect

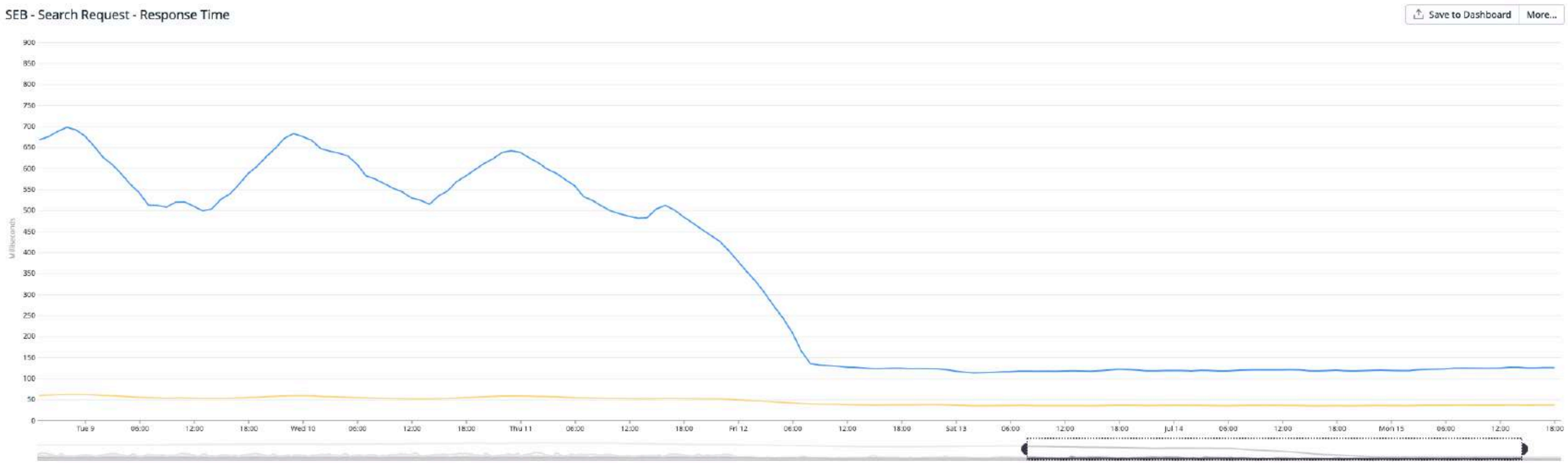
- Replacing `field_value_factor` with `rank_feature`
- No noticeable effect on latency
- Reason: unknown
- Filtering probably more expensive than scoring?

Tracking total hits: weird behaviour

- Hot dataset, bool query contains `must/should/filter` clauses, 11k results
  - `track_total_hits: false` → **400ms**
  - `track_total_hits: true` → **200ms**
- WTF?
- Dynamic pruning not working as expected (remember `rank_feature`)
- See discuss forum

Switching off a microservice

- Dropping one gRPC network hop
- Might be a few milliseconds, amirite?



Improving relevance

Keyword search

- Easy to understand
- Fast
- Debuggable

ELSER

- Great idea!
- English only
- Not based on our own data

Query Processing Pipeline

- Synonyms
- Unit detection
- Query expansion
- Query rewriting

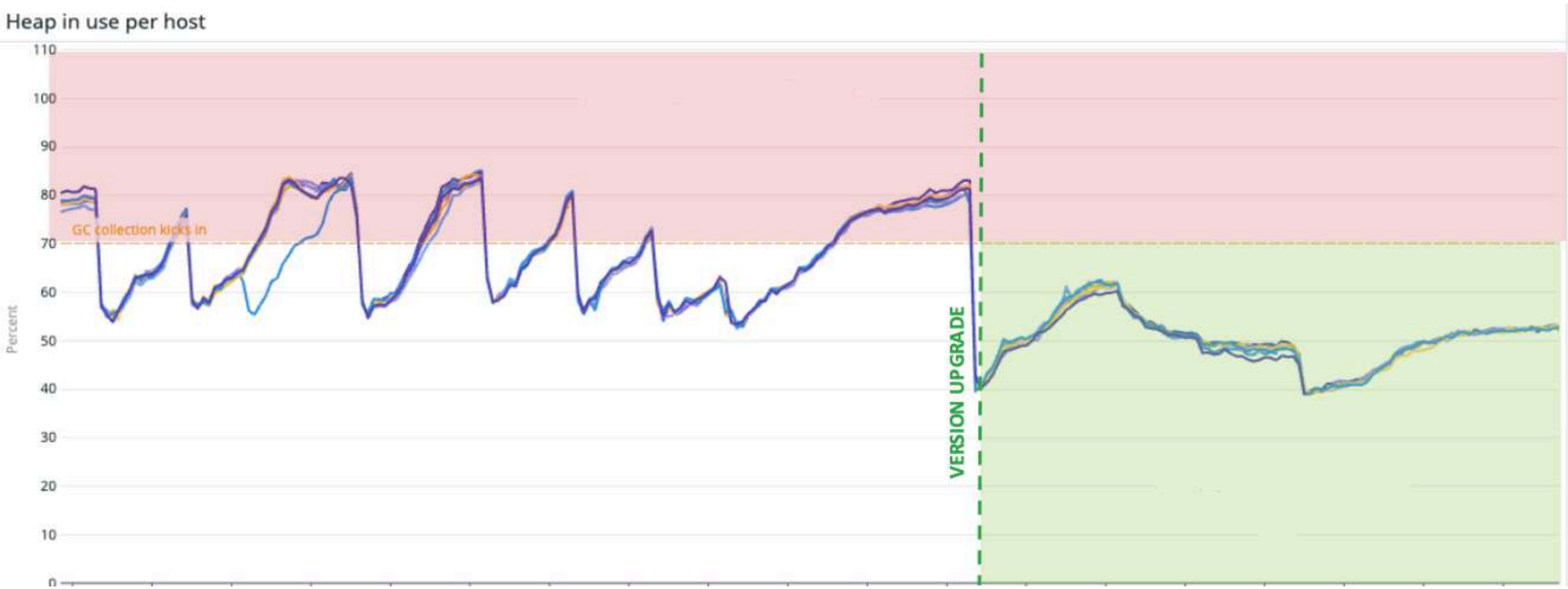
Query expansion

- Find alternative search queries for a search term
- Add to existing query
- Two additional queries: ESQL and scripted (single digit milliseconds)
- Ultimate goal: No human-made rotting synonyms, for all languages

Summary

Upgrade!

- Another cluster upgraded from 7.9 to 7.17
- Much less GC, CPU, nodes could be removed



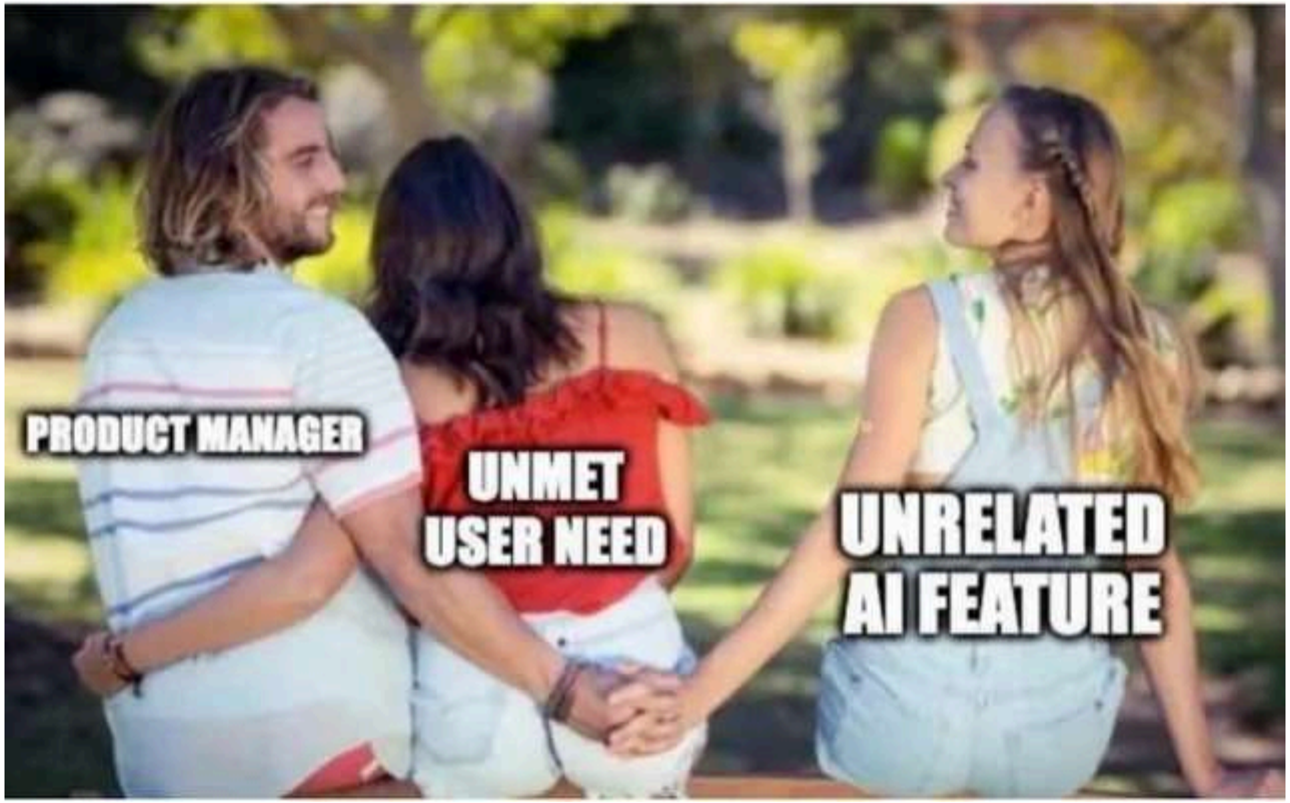
Summary

- Upgrade
- Invest in upgrade automation
- Monitor ~~costs~~ everything
- Relevancy improvement is a long term game
- E-commerce search keeps being hard

Thank you

One last thing

Every company’s tech department



Thank you

Questions? Answers!

alexander.reelsen@kaufland-ecommerce.com

P.S. We’re hiring