Productionalizing Elasticsearch at Scale

Talk By – Subramaniajjeva K

April 2018
Myself Jeeva, enthusiastic engineer, working as technical lead in Paypal.

I have been a backbone in designing and implementing a highly scalable system that powers the elastic search engine to process billions of records every day.

Having intense hands on experience on multiple cutting-edge technologies like Elasticsearch, Kafka, Akka, Scala and Reactive programming, I provide excellent expertise in building resilient distributed systems.

Contact me at subramaniajeeva [gmail, twitter, fb, linkedin]
Agenda

- Elasticsearch - How it works
- Scaling Elasticsearch in production
- Resilient by design
- Do’s and Don’ts
- Monitoring and Alerting
- Summary
Elasticsearch – Data Organization

Cluster

Node 1
- Shard 0
- Shard 1 Replica
- Disk xx GB

Node 2
- Shard 1
- Shard 0 Replica
- Disk xx GB

Index

Shard 0

Shard 1

Disk xx GB
Elasticsearch – Routing

Indexing request
Document A

{"text": "brown fox jumped into the well"}

Node 1
- Shard 0 Primary
- Shard 1 Replica

Node 2
- Shard 0 Replica
- Shard 1 Primary

Index document A to shard1
Replicate document A
Dissecting Indexing

### Source Text

**Document 1**

The *<em>Quick brown</em>* fox jumped over the lazy dog

**Document 2**

The two *<em>lazy</em>* dogs were slower than the less lazy *<em>dog</em>*, Rover

### html_strip filter

**Document 1**

The Quick brown fox jumped over the lazy dog

**Document 2**

The two lazy dogs were slower than the less lazy dog, Rover

### Standard tokenizer

**Document 1**

The Quick brown fox jumped over the lazy dog

**Document 2**

The two lazy dogs were slower than the less lazy dog, Rover

### Lower case filter

**Document 1**

the quick brown fox jumped over the lazy dog

**Document 2**

the two lazy dogs were slower than the less lazy dog, rover

### Stop words filter

**Document 1**

quick brown fox jumped over the lazy dog

**Document 2**

the two lazy dogs were slower than the less lazy dog, rover

### Inverted Index

<table>
<thead>
<tr>
<th>Term</th>
<th>Document</th>
</tr>
</thead>
<tbody>
<tr>
<td>quick</td>
<td>1</td>
</tr>
<tr>
<td>brown</td>
<td>1</td>
</tr>
<tr>
<td>fox</td>
<td>1</td>
</tr>
<tr>
<td>jumped</td>
<td>1</td>
</tr>
<tr>
<td>over</td>
<td>1</td>
</tr>
<tr>
<td>lazy</td>
<td>1, 2</td>
</tr>
<tr>
<td>dog</td>
<td>1, 2</td>
</tr>
<tr>
<td>two</td>
<td>2</td>
</tr>
<tr>
<td>dogs</td>
<td>2</td>
</tr>
<tr>
<td>slower</td>
<td>2</td>
</tr>
<tr>
<td>less</td>
<td>2</td>
</tr>
<tr>
<td>rover</td>
<td>2</td>
</tr>
</tbody>
</table>
Dissecting Search

Search for “The lazy Fox”

html_strip filter
Standard tokenizer
Lower case filter
Stop words filter

Inverted Index

<table>
<thead>
<tr>
<th>Term</th>
<th>Document</th>
</tr>
</thead>
<tbody>
<tr>
<td>quick</td>
<td>1</td>
</tr>
<tr>
<td>brown</td>
<td>1</td>
</tr>
<tr>
<td>fox</td>
<td>1</td>
</tr>
<tr>
<td>jumped</td>
<td>1</td>
</tr>
<tr>
<td>over</td>
<td>1</td>
</tr>
<tr>
<td>lazy</td>
<td>1, 2</td>
</tr>
<tr>
<td>dog</td>
<td>1, 2</td>
</tr>
<tr>
<td>two</td>
<td>2</td>
</tr>
<tr>
<td>dogs</td>
<td>2</td>
</tr>
<tr>
<td>slower</td>
<td>2</td>
</tr>
<tr>
<td>less</td>
<td>2</td>
</tr>
<tr>
<td>rover</td>
<td>2</td>
</tr>
</tbody>
</table>
Scaling Elasticsearch in production

- Use Bulk requests instead of single requests

- Async translog flush

- Higher refresh interval

- Routing

Latency < 10 ms
Resiliency

What is resiliency:

• Stay responsive in the face of failure

Ways to achieve:

• Timeouts

• Retries

• Backpressuring incoming requests

• Circuit breaker
Back-pressure

Kafka to ES stream

- Kafka Source
- Transformer
- Bulkizer
- ES client
- Sink

pull

No demand
Demand
Back-pressure

Kafka to ES stream

- Kafka Source
- Transformer
- Bulkizer
- ES client
- Sink

Push and pull arrows indicate data flow and back-pressure conditions.

No demand
Demand
Elasticsearch in production

**Do**
- Gate keeper between users and es
- Set same shard host allocation to true
- Use limited fields
- Choose correct analyzer
- Use latest version of elasticsearch (6.x)

**Don’t**
- High cardinality field aggregation
- Deep nested aggregation
Monitoring and Alerting

Monitoring app

ES metrics

Monitoring ES

kibana

Watcher
Summary

• *Elasticsearch should be used for what does best.*

• *To manage better proxy layer is a must.*

• *Defaults won’t work for all cases. Tuning is needed.*

• *When it fails, it fails miserably. Resiliency is a must during chaos.*

• *Better monitoring and alerting helps in faster triaging and less downtime.*
Q & A
Use cases

- Location search - Find merchants/customers within a range
- Anomaly detection - Detecting malicious activities
- Searching errors - Search errors from application logs
- Analytics - What is the trend of #transactions, error rate [kibana visualization]
- Full text search - keyword search, phrase search, exact match
- “Did you mean?” - Fuzzy search, suggest
References

- Squbs  - [http://github.com/paypal/squbs](http://github.com/paypal/squbs)
- Akka    - [https://github.com/akka/akka](https://github.com/akka/akka)
- Akka stream docs - [https://doc.akka.io/docs/akka/2.5/stream/index.html](https://doc.akka.io/docs/akka/2.5/stream/index.html)
- Backpressure - [https://www.lightbend.com/blog/understanding-akka-streams-back-pressure-and-asynchronous-architectures](https://www.lightbend.com/blog/understanding-akka-streams-back-pressure-and-asynchronous-architectures)
GREAT INDIAN DEVELOPER SUMMIT 2019
Conference: April 23-26, Bangalore

Register early and get the best discounts!

www.developersummit.com
@greatindiandev
bit.ly/gidslinkedin
facebook.com/gids19
bit.ly/saltmarchyoutube
flickr.com/photos/saltmarch/