Managing your Black Friday Logs

David Pilato
Developer | Evangelist, @dadoonet
Data Platform Architectures
The Elastic Journey of Data
The Elastic Journey of Data

Beats
- Log Files
- Wire Data
- Metrics
- your{beat}

Elasticsearch
- Master Nodes (3)
- Ingest Nodes (X)
- Data Nodes
  - Hot (X)
  - Data Notes
    - Warm (X)
The Elastic Journey of Data

Beats
- Log Files
- Wire Data
- Metrics
- your{beat}

Elasticsearch
- Master Nodes (3)
- Ingest Nodes (X)
- Data Nodes
  - Hot (X)
  - Warm (X)

Kibana
- Instances (X)
The Elastic Journey of Data
The Elastic Journey of Data

- **Beats**
  - Log Files
  - Wire Data
  - Metrics
  - your{beat}

- **Elasticsearch**
  - Master Nodes (3)
  - Ingest Nodes (X)
  - Data Nodes
    - Hot (X)
    - Warm (X)

- **Logstash**
  - Nodes (X)

- **Data Store**
  - Web APIs

- **Social Sensors**

- **Kibana**
  - Instances (X)
The Elastic Journey of Data

Beats

Log Files
Wire Data
Metrics
your{beat}

Elasticsearch

Master Nodes (3)
Ingest Nodes (X)
Data Nodes
Hot (X)
Data Notes
Warm (X)

Logstash

Nodes (X)

Kibana

Instances (X)

Data Store
Web APIs
Social Sensors

Queues
Storage
Metrics
Notification

@dadoonet
sli.do/elastic
The Elastic Journey of Data

Beats
- Log Files
- Wire Data
- Metrics

your{beat}

Data Store

Web APIs

Social Sensors

Messaging Queue

Kafka

Redis

Elasticsearch
- Master Nodes (3)
- Ingest Nodes (X)
- Data Nodes
  - Hot (X)
  - Warm (X)

Logstash

Nodes (X)

Kibana

Instances (X)

Queues

Storage

Metrics

Notification

@dadoonet
sli.do/elastic
Elasticsearch Cluster Sizing
Terminology

Cluster *my_cluster*

Server 1

Node A

Index *twitter*

Index *logs*
Partition

Cluster *my_cluster*

Server 1

Node A

Shards

Index *twitter*

Index *logs*

Log and metrics data

Sensor and device data

Web and social data

Data stores and streams
Distribution

Cluster my_cluster

Server 2

Node B

twitter shard P4

d1

d2
d6
	twitter shard P1

d5
d10
d3
d6

twitter shard P2

d12

twitter shard P3

d3
d9

d4
d11

d7
d4
d2
	d5

twitter shard P0

d3
d4
d9
d7
d11
d8
d10
d5

tweets

Server 1

Node A

logs shard P0

d7

d3

d1
d3

d6

tweets

logs shard P1

d2
d4

d5

d11

d8

d4

d9
Replication

Cluster *my_cluster*

- Primaries
- Replicas

![Diagram showing replication in a cluster with nodes A and B, including shards and replicas.](image-url)
Replication

Cluster my_cluster

- Primaries
- Replicas
Scaling

Data
Scaling

Data
Scaling

Data
Scaling

Big Data

...
Scaling

• In Elasticsearch, **shards** are the **working unit**
• More data -> More shards

But how many shards?
How much data?

• ~1000 events per second
• 60s * 60m * 24h * 1000 events => ~87M events per day
• 1kb per event => ~82GB per day
• 3 months => ~7TB
Shard Size

- It depends on many different factors
  - document size, mapping, use case, kinds of queries being executed, desired response time, peak indexing rate, budget, ...
- After the shard sizing*, each shard should handle 45GB
- Up to 10 shards per machine

* https://www.elastic.co/elasticon/conf/2016/sf/quantitative-cluster-sizing
How many shards?

- Data size: ~7TB
- Shard Size: ~45GB*
- Total Shards: ~160

- Shards per machine: 10*
- Total Servers: 16

3 months of logs

Cluster my_cluster

*https://www.elastic.co/elasticon/conf/2016/sf/quantitative-cluster-sizing
But...

- How many indices?
- What do you do if the daily data grows?
- What do you do if you want to delete old data?
Time-Based Data

• Logs, social media streams, time-based events
• Timestamp + Data
• Do not change
• Typically search for recent events
• Older documents become less important
• Hard to predict the data size
Time-Based Data

• Time-based Indices is the best option
  – create a new index each day, week, month, year, ...
  – search the indices you need in the same request
Daily Indices

Cluster *my_cluster*

logs-2019-02-20

Log and metrics data
Daily Indices

Cluster *my_cluster*

- logs-2019-02-20
- logs-2019-02-21

Log and metrics data
Daily Indices

Cluster *my_cluster*

- logs-2019-02-20
- logs-2019-02-21
- logs-2019-02-22

Log and metrics data
Templates

- Every new created index starting with 'logs-' will have
  - 2 shards
  - 1 replica (for each primary shard)
  - 60 seconds refresh interval

```
PUT _template/logs
{
  "template": "logs-*",
  "settings": {
    "number_of_shards": 2,
    "number_of_replicas": 1,
    "refresh_interval": "60s"
  }
}
```

More on that later
Alias

Cluster *my_cluster*

```
      d6
     /   
    d3   d6
     |   /  
    d1 d4  d2
     |   
    d5
```

```
logs-write

Log and metrics data

Logs

Application

logs-read

users

logs-2019-02-20

```
Alias

Cluster *my_cluster*

- **logs-write**
  - `logs-2019-02-20`
  - `d1`, `d2`, `d3`, `d4`, `d5`, `d6`

- **logs-read**
  - `logs-2019-02-21`
  - `d1`, `d2`, `d3`, `d4`, `d5`, `d6`

Application

- Log and metrics data
- users
Alias

Cluster `my_cluster`

- `d1`, `d2`, `d3`, `d4`, `d5`, `d6`

- `logs-2019-02-20`
- `logs-2019-02-21`
- `logs-2019-02-22`

Log and metrics data

Application

users
Detour: Rollover API

Do not Overshard

don't keep default values!

- 3 different logs
- 1 index per day each
- 1GB each
- 5 shards (default): so 200mb / shard vs 45gb
- 6 months retention
- ~900 shards for ~180GB
- we needed ~4 shards!
Scaling

But what happens if we have 2M users?
Scaling

Big Data

1M users

1M users
Scaling

Big Data

1M users

1M users

1M users
Scaling

Big Data

Users

@dadoonet

sli.do/elastic
Shards are the working unit

- **Primaries**
  - More data -> More shards
  - **write throughput** (More writes -> **More primary shards**)
- **Replicas**
  - high availability (1 replica is the default)
  - **read throughput** (More reads -> **More replicas**)
Detour: Shrink API

Detour: Split API

Optimal Bulk Size
What is Bulk?

1000 index requests with 1 document

1 bulk request with 1000 documents
What is the optimal bulk size?

- Beats
- Logstash
- Application

1000 log events

- 4 * 250 =
- 2 * 500 =
- 1 * 1000 =

Elasticsearch
- Master Nodes (3)
- Ingest Nodes (X)
- Data Nodes
  - Hot (X)
  - Warm (X)
It depends...

- on your application (language, libraries, ...)
- document size (100b, 1kb, 100kb, 1mb, ...)
- number of nodes
- node size
- number of shards
- shards distribution
Test it ;)

1000000 log events

Beats
Logstash
Application

4000 * 250 -> 160s
2000 * 500 -> 164s
1000 * 1000 -> 155s

Elasticsearch

Master Nodes (3)
Ingest Nodes (X)
Data Nodes Hot (X)
Data Notes Warm (X)
Test it ;)

```bash
input { stdin{} }
filter {} 
output {
  elasticsearch {
    hosts => ["10.12.145.189"]
    flush_size => 
    "$\{SIZE\}"
  }
}
```

```bash
DATE=`date +%Y.%m.%d`
LOG=logs/logs.txt 
exec_test () {
  curl -s -XDELETE "http://USER:PASS@HOST:9200/logstash-$DATE"
  sleep 10
  export SIZE=$1
  time cat $LOG | ./bin/logstash -f logstash.conf
}

for SIZE in 100 500 1000 3000 5000 10000; do
  for i in {1..20}; do
    exec_test $SIZE
  done;
```

In Beats set "bulk_max_size" in the output.elasticsearch

DATE=`date +Y.%m.%d`
LOG=logs/logs.txt
exec_test () {
  curl -s -XDELETE "http://USER:PASS@HOST:9200/logstash-$DATE"
  sleep 10
  export SIZE=$1
  time cat $LOG | ./bin/logstash -f logstash.conf
}

for SIZE in 100 500 1000 3000 5000 10000; do
  for i in {1..20}; do
    exec_test $SIZE
  done;
```
Test it ;)

- 2 node cluster (m3.large)
  - 2 vCPU, 7.5GB Memory, 1x32GB SSD
- 1 index server (m3.large)
  - logstash
  - kibana

<table>
<thead>
<tr>
<th># docs</th>
<th>100</th>
<th>500</th>
<th>1000</th>
<th>3000</th>
<th>5000</th>
<th>10000</th>
</tr>
</thead>
<tbody>
<tr>
<td>time(s)</td>
<td>191.7</td>
<td>161.9</td>
<td>163.5</td>
<td>160.7</td>
<td><strong>160.7</strong></td>
<td>161.5</td>
</tr>
</tbody>
</table>
Distribute the Load
Avoid Bottlenecks

```
output {
  elasticsearch {
    hosts => ["node1","node2"]
  }
}
```
Load Balancer

1000000 log events

Beats
Logstash
Application

LB

Elasticsearch

Node 1

Node 2
Coordinating-only Node
Test it ;)

- 2 node cluster (m3.large)
  - 2 vCPU, 7.5GB Memory, 1x32GB SSD
- 1 index server (m3.large)
  - logstash (round robin configured)
    - hosts => ["10.12.145.189", "10.121.140.167"]
  - kibana

<table>
<thead>
<tr>
<th>#docs</th>
<th>1000</th>
<th>5000</th>
<th>10000</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO Round Robin</td>
<td>163.5</td>
<td>160.7</td>
<td>161.5</td>
</tr>
<tr>
<td>Round Robin</td>
<td>161.3</td>
<td>158.2</td>
<td>159.4</td>
</tr>
</tbody>
</table>
Optimizing Disk IO
Durability

- Index a doc → Buffer
- Index a doc → Buffer
- Index a doc → Buffer
- Buffer → Lucene flush → Segment
**refresh_interval**

- Dynamic per-index setting
  
  ```
  PUT logstash-2017.05.16/_settings
  {
    "refresh_interval": "60s"
  }
  ```

- Increase to get better write throughput to an index
- New documents will take more time to be available for Search.

<table>
<thead>
<tr>
<th>time(s)</th>
<th>#docs 1000</th>
<th>#docs 5000</th>
<th>#docs 10000</th>
</tr>
</thead>
<tbody>
<tr>
<td>1s refresh</td>
<td>161.3</td>
<td>158.2</td>
<td>159.4</td>
</tr>
<tr>
<td>60s refresh</td>
<td>156.7</td>
<td>152.1</td>
<td>152.6</td>
</tr>
</tbody>
</table>
Durability

- index a doc
- lucene flush
- elasticsearch flush

- buffer
- trans_log

- lucene commit
- segment

@dadoonet
Translog fsync every 5s (1.7)

Primary

Replica

redundancy doesn’t help if all nodes lose power
Translog fsync on every request

- For low volume indexing, fsync matters less
- For high volume indexing, we can amortize the costs and fsync on every bulk
- Concurrent requests can share an fsync
Async Transaction Log

- `index.translog.durability`
  - request (default)
  - async
- `index.translog.sync_interval` (only if async is set)
- Dynamic per-index settings
- **Be careful, you are relaxing the safety guarantees**

<table>
<thead>
<tr>
<th>#docs time(s)</th>
<th>1000</th>
<th>5000</th>
<th>10000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request fsync</td>
<td>161.3</td>
<td>158.2</td>
<td>159.4</td>
</tr>
<tr>
<td>5s sync</td>
<td>152.4</td>
<td>149.1</td>
<td>150.3</td>
</tr>
</tbody>
</table>
Final Remarks

LET'S JUMP TO CONCLUSIONS
Final Remarks

• Primaries
  – More data -> More shards
  – Do not overshard!

• Replicas
  – high availability (1 replica is the default)
  – read throughput (More reads -> More replicas)
Final Remarks

- Bulk and Test
- Distribute the Load
- Refresh Interval
- Async Trans Log (careful)

<table>
<thead>
<tr>
<th>#docs per bulk</th>
<th>1000</th>
<th>5000</th>
<th>10000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>163.5</td>
<td>160.7</td>
<td>161.5</td>
</tr>
<tr>
<td>RR+60s+Async5s</td>
<td>152.4</td>
<td>149.1</td>
<td>150.3</td>
</tr>
</tbody>
</table>
Managing your Black Friday Logs

David Pilato
Developer | Evangelist, @dadoonet