Scaling Elasticsearch

James Adams

Scientific Computing Department
STFC Rutherford Appleton Laboratory

… at RAL

- Original use case — CASTOR event logging
- One day of logs from all instances:
  - ~33,000,000 events
  - ~15GB text
- Aiming to keep indices for at least 32 days
  - ~1,000,000,000 events
  - ~500GB text
- Only using old WNs
  - With 500GB disks (440GB usable)

…at RAL

- Wanted to use replication
  - Double storage requirement
- Clearly needed more than a single node
  - Overhead to deal with node failure
  - Ended up with ten 2008 WNs
- Still struggled to keep up with the rate
  - Peaks up to 2,000 events/second
  - Many cluster meltdowns
  - Started tuning

Logstash

- CASTOR events are a series of Key-Value pairs
- Messages are therefore fully tokenised
  - Dropped original message fields
  - Halved storage cost
  - Increased throughput

```
mutate {
    remove_field => [ "@message", "message" ]
}
```

Elasticsearch

- General purpose indexed document search system
  - Hides **very** complex systems behind a simple interface
- Default tuning is good
  - Great for search
- We can make it better for our use case
  - And make it worse for all others…

Elasticsearch

- Default tuning optimised for search
  - Logging is normally an index heavy workload
    - CASTOR especially
    - We'll optimise for that
- Settings
  - Per cluster
  - Per node
  - Per index
    - Change the defaults given to new indices

Number of Shards

- Number of shards an index is divided into by default

  `index.number_of_shards = 5`

  - We use Quattor to set this to `ELASTICSEARCH_NODE_COUNT / 2`
  - Aim to have at least have one shard (or replica) per node
- By default each shard has one replica
  - Mostly for handling node failure
  - More replicas increases search performance
  - Don't change this

Swap

- Never swap, never, ever, ever
  - **Best** Disable swap
  - **Good** Set `vm.swappiness = 1`
  - **Okay** Set `bootstrap.mlockall: true`
- Or do all of them, just to be safe

OS Limits

- Configure `/etc/security/limits.d/99-elasticsearch.conf`
- Be super generous with number of open Files
- Allow ES to lock all the memory it wants

```
elasticsearch   soft   nofile   65536
elasticsearch   hard   nofile   65536
elasticsearch   -      memlock  unlimited
```

JVM Settings

- Do not allow the JVM use more than 32GB of RAM
  - Uses uncompressed pointers above 32GB
    - Kills performance
  - Set following to same value in `/etc/sysconfig/elasticsearch`

```
ES_MIN_MEM = 31GB
ES_MAX_MEM = 31GB
ES_HEAP_SIZE = 31GB
```

- Will also need lots of files open

```
MAX_OPEN_FILES = 65535
```

- Leave the rest of the JVM alone

Lucene

- Logstash creates one index per day
- Each index has 5+5 shards (one per node)
- Each shard has hundreds of segments
  - Each is a file on disk
  - Immutable — great for filesystem cache
- 20,000 files per node
- Usage requires segments to be open in memory

Index Refresh Interval

- Interval at which Lucene segments are created
- Effectively the delay between indexing a document and it becoming visible in search results
- By default every shard of every index is refreshed once per second
- Segments are merged in the background
  - CPU and IO intensive
  - Throttled by `indices.store.throttle.max_bytes_per_sec`
  - Avoid touching it unless you have very fast storage

Index Refresh Interval

- If we don't care about near real-time search we can relax it
- For example:

```
index/refresh_interval = 30s
```

- Larger delay = fewer larger segments
  - Consumes less OS resources (file handles etc.)
  - Search queries every segment of every shard
    - Each segment has it's own terms dictionary
    - When saturated, search performance is multiplied by number of segments

Index Transaction Log

- Each shard has a write-ahead transaction log
  - Guarantees atomicity before data hits Lucene
- Commits to Lucene are triggered at thresholds
  - Default is size based

    ```
    index.translog.flush_threshold_size = 512mb
    ```
  - We also specify threshold for number of operations

    ```
    index.translog.flush_threshold_ops = 50000
    ```

Index Buffer Size

- Amount of memory reserved for indexing operations
- Per node setting
- Defaults to 10%
  - Needed it much higher
  - Rely on filesystem cache to accelerate searches

```
indices.memory.index_buffer_size = 50%
```

Curator

- Manage time based indices with simple rules
- Before you run out of
  - Disk space
  - Memory
  - Sanity
- Run as daily cron job

```
/usr/bin/curator delete --older-than 64
```

Conclusions

- We can now handle the load and more
  - Also collecting job histories from HTCondor
  - Starting to think about other services (Ceph?)
  - Migrating to a newer cluster with fewer nodes
- Easily keep 64 days CASTOR logs searchable

  ~2,000,000,000 events

  ~1TB total

Questions?

Further Reading:

Inside a shard
https://www.elastic.co/guide/en/elasticsearch/guide/current/inside-a-shard.html

Heap Sizing
https://www.elastic.co/guide/en/elasticsearch/guide/current/heap-sizing.html

Don't Touch These Settings!
https://www.elastic.co/guide/en/elasticsearch/guide/current/_don_8217_t_touch_these_settings.html

Performance Considerations for Elasticsearch Indexing
https://www.elastic.co/blog/performance-considerations-elasticsearch-indexing

Elasticsearch Indexing Performance Cheatsheet
https://blog.codecentric.de/en/2014/05/elasticsearch-indexing-performance-cheatsheet/

Visualizing Lucene's segment merges
http://blog.mikemccandless.com/2011/02/visualizing-lucenes-segment-merges.html