elasticsearch

Τεχνολογία Λογισμικού 2017-2018

Αναγνωστίδης Σωτήρης-Κωνσταντίνος

Who uses it


WordPress
GitHub
stackoverflow
NETFLIX
StumbleUpon
mozilla
CERN
theguardian
flickr
Etsy
accenture
JUST EAT
DELL

Search Engine

- Indexing of data
- Analyzing of data
  - Text Search
    - Tokenize
    - Analyze
  - Locations, Date, Score value, …

NoSQL

- Doesn’t have a structured schema
- Drops the acid requirements

Elasticsearch is a document oriented search engine

Insert documents

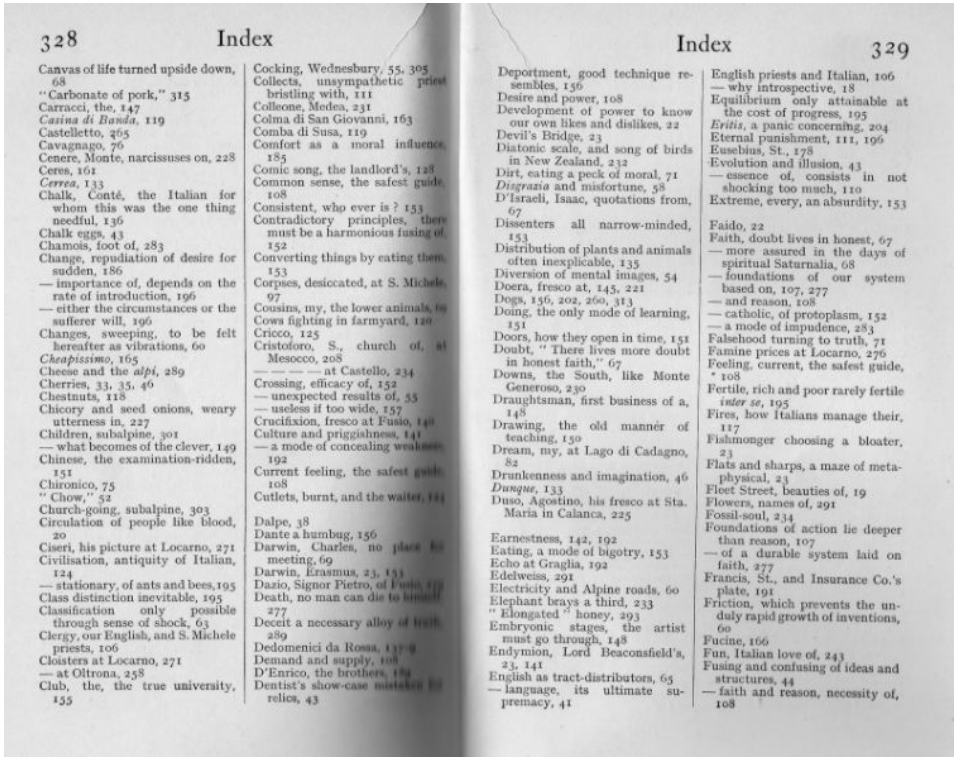
Delete documents

Retrieve documents

Analyze documents

Search documents

Appendix



Origin

Elastic search in built on the of a search software called lucene

Underline data structure;

- inverted index, designed to allow very fast full-text searches

Organizes documents into the inverted index

Example (documentation)

Sentences break into terms or tokens.

1. The quick brown fox jumped over the lazy dog
2. Quick brown foxes leap over lazy dogs in summer

Term	Doc_1	Doc_2
Quick		X
The	X	
brown	X	X
dog	X	
dogs		X
fox	X	
foxes		X
in		X
jumped	X	
lazy	X	X
leap		X
over	X	X
quick	X	
summer		X
the	X	

What are documents

Most entities or objects can be serialized into JSON!

With fields and values

- Simple
- Fast
- Easy to read


```
{
    "name":         "John Smith",
    "age":          42,
    "confirmed":    true,
    "join_date":    "2014-06-01",
    "home": {
        "lat":      51.5,
        "lon":      0.1
    },
    "accounts": [
        {
            "type": "facebook",
            "id":   "johnsmith"
        },
        {
            "type": "twitter",
            "id":   "johnsmith"
        }
    ]
}
```

Insert

Inserting = indexing

```
PUT /{index}/{type}/{id}
{
  "data1": data1,
  "data2": data2
}
```

Methods

GET

HEAD

PUT

POST

DELETE

It's all about REST semantics.

POST basically that you are posting a request which is going to modify the server stat

```
POST index/type
{
  "foo": "bar"
}
```

will generate an `_id` server side and will index the document with this `_id`.

PUT is used to send a resource to the server.

```
PUT index/type/id
{
  "foo": "bar"
}
```

will put or update a document named `index/type/id` in the server.

Index structure - schema

Mappings; field and its type

Settings; shards, replicas, …

```
PUT localhost:9200/vehicles
{
    "settings": {
    ...
    },

    "mappings": {
    ...
    }

}
```

Elasticsearch structure

Elasticsearch is distributed technology

One elasticsearch cluster is composed of multiple nodes

Nodes has different replicas

Physical representation: shards

Elasticsearch structure

Shards; splits of the index

Every node has a primary shard and replicas of others.

Document goes to each node based on its id after passing through a hash function. Buy this hash value the new document is sent to the node where the primary shard can be found

After a successful index, replicate to the other replicas

Every node knows where all the documents are in the cluster. Each request goes to the node that has the primary shard.

Elasticsearch structure

On get request (read request) the coordinator balances the requests equally to all nodes.

The requests are balanced in a round robin way.

Shards

Shard; lucene index

Elasticsearch makes lucene distributed

Each shard is a container of inverted indexes. It has multiple segments

Each segment is an inverted index.

Indexing

Analysis (more on this later)

The inverted index is created.

This information goes to a buffer.

Once this buffer gets filled up it gets committed to a segment

Segments are immutable. Once they are populated they can’t be changed

Review

cluster – An Elasticsearch cluster consists of one or more nodes and is identifiable by its cluster name.

node – A single Elasticsearch instance. In most environments, each node runs on a separate box or virtual machine.

index – In Elasticsearch, an index is a collection of documents.

shard – Because Elasticsearch is a distributed search engine, an index is usually split into elements known as shards that are distributed across multiple nodes. Elasticsearch automatically manages the arrangement of these shards. It also rebalances the shards as necessary, so users need not worry about the details.

replica – By default, Elasticsearch creates five primary shards and one replica for each index. This means that each index will consist of five primary shards, and each shard will have one copy.

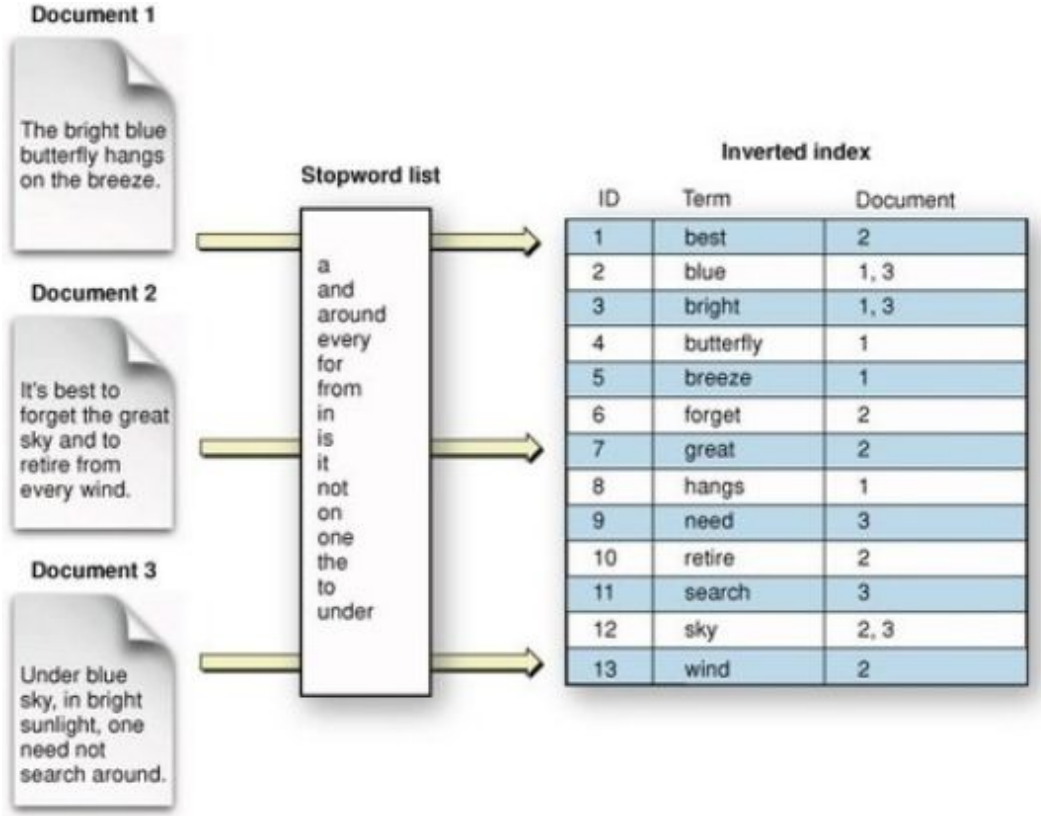
The more segments there are, the longer each search takes. So Elasticsearch will merge a number of segments of a similar size ("tier") into a single bigger segment, through a background merge process. Once the new bigger segment is written, the old segments are dropped. This process is repeated on the bigger segments when there are too many of the same size.

Segments are immutable. When a document is updated, it actually just marks the old document as deleted, and indexes a new document. The merge process also expunges these old deleted documents.

Analysis

Document is made into inverted index and stored into a shard segment

Create tokens so they can be found



Inverted index

- Tokenizer
- Filtering
  - Remove stop words (non important words)
  - Case insensitive
  - Stemming
  - Synonyms

In searching the same process happens

Index

Create index

```
PUT vehicles
{
  "settings": {
    "number_of_shards": 5
    , "number_of_replicas": 1
  },
  "mappings": {}
}
```

Create mapping

```
PUT vehicles/_mapping/cars
{
  "properties": {
    "name": {
      "type": "text",
      "analyzer": "standard"
    }
  }
}
```

GET vehicles

Also

New fields are automatically indexed by finding the datatype

… unless you set dynamic to false (error will be thrown)


```
{
  "vehicles": {
    "aliases": {},
    "mappings": {
      "cars": {
        "properties": {
          "name": {
            "type": "text",
            "analyzer": "standard"
          }
        }
      }
    },
    "settings": {
      "index": {
        "creation_date": "1519937080703",
        "number_of_shards": "5",
        "number_of_replicas": "1",
        "uuid": "bWYkqIasRvqGJfY_st8w5g",
        "version": {
          "created": "6000099"
        },
        "provided_name": "vehicles"
      }
    }
  }
}
```

Datatypes

string

`text` and `keyword`

Numeric datatypes

`long`, `integer`, `short`, `byte`, `double`, `float`, `half_float`, `scaled_float`

Date datatype

`date`

Boolean datatype

`boolean`

Binary datatype

`binary`

Range datatypes

`integer_range`, `float_range`, `long_range`, `double_range`, `date_range`

Complex (array, object, nested)

Geo (geopoint geoshape)

Specialised (ip, completion, token count, …)

Analyzers

1. Standard Analyzer
2. Simple Analyzer
3. Whitespace Analyzer
4. Stop Analyzer
5. Keyword Analyzer
6. Pattern Analyzer
7. Language Analyzers
8. Custom analyzers …

Query DSL (Domain Specific Language)

Query context

- Relevancy score

Filter context

Query Examples

Different score base on: number of letters in each field, frequency of the word in the document, in all documents.

```
GET vehicles/_search
{
  "query": {
    "match": {
      "name": "honda"
    }
  }
}
```

```
GET vehicles/_search
{
  "query": {
    "bool": {
      "must": [
        { "match": {
          "FIELD1": "TEXT1"
        }},
        { "match": {
          "FIELD2": "TEXT2"
        }}
      ]
    }
  }
}
```

Other key words:

Exists, must_not, multi_match

should, minimum_should_match

Numbers: range, gt, lt, gte, lte

Field boosting..

Aggregations

- Size, from
- Sort results by field
  - descending
  - ascending

- Stats
- Combination with queries, filters

```
GET vehicles/cars/_search
{
  "aggs": {
    "popular_manufacturer": {
      "terms": {
        "field": "manufacturer.keyword",
        "size": 10
      },
      "aggs": {
        "avg_price": {
          "avg": {
            "field": "price"
          }
        }
      }
    }
  }
}
```

elastic

logstash

References

https://www.elastic.co/guide/index.html

https://stackoverflow.com