Is the evolution of search algorithms finished?

Foidaș Andrei Ștefan

Group 933/2

AMS Mathematical Subject Classification

90B40 Search theory

ACM Computing Reviews Categories and Subject Descriptors

F.2 ANALYSIS OF ALGORITHMS AND PROBLEM COMPLEXITY

F.2.2 Nonnumerical Algorithms and Problems

Sorting and searching

# Experimenting

## Experimental modeling

In the experimentation section we will compare and analyze Amazon’s OpenSearch Service to its predecessor, Amazon Elasticsearch Service. We will experiment on three different domains: one of the first versions of Elasticsearch, the latest version of Elasticsearch and OpenSearch. Elasticsearch and OpenSearch are both search engines and data storage systems. They provide scalable search, fast access and response to very large volumes of data, as well as data visualization tools.

The main goal of this study is to show that search algorithms are constantly evolving, performing faster and with increased accuracy.

## Experimentation data

To try to simulate the real usages of such search engines, I created a Python script that generates large JSON files with various data types. We will conduct a bundle of different search tests on files of different lengths to try to figure out which search engines excels and in which domain.

When it comes to data validation, the search engines will accept only valid JSON objects or files, with a specific “bulk” format, hence any invalid data will be instantly rejected. Moreover, the search engines will return information related to the state of the response such as how many items fetched were successful, skipped and failed.

## Rigorous experiment model

The experiment is focused on directly comparing the three search domains through a series of different types of searches and sorting tests. We will not only look at the data resulted from the queries, but we will differentiate the search domains from multiple perspectives: indexes used, elapsed time, number of hits and successful finds such that meaningful conclusions can be derived. We will use the exact same queries on the exact same data set so we can get the most accurate results.

At the end of the experiment we will be able to not only the differences between the search engines, but see how they managed to evolve, what they prioritize now and which of suits a test field better.

# Case Study

## Creating search domains

With the help of the Amazon Web Services we will generate three search domains with similar settings but different versions: one of the first versions of the Elasticsearch Service, the newest version of Elasticsearch and one instance of the OpenSearch Service. All three domains are deployed for development and testing, using an instance of t3.small.search with three nodes and can be accessed publicly.

## Uploading data

In order to conduct the tests we desire, we need some data to work with. Both OpenSearch and Elasticsearch represent documents in JSON format, so naturally we will upload data to the search domains using JSON files. But in order to upload the data, the JSON file needs to have a specific bulk format:

* For OpenSearch versions:

{"index": {"\_index": "index", "\_id": "id"}}

{"field1": "field1", {"field2": "field2" …}

* For Elasticsearch versions:

{"index": {"\_index": "index", "\_type": "type", "\_id": "id"}}

{"field1": "field1", {"field2": "field2" …}

In order to generate JSON files with the specific bulk format I created a Python script that generates two files with the same JSON objects but in the two different bulk formats. The script can generate files with a specific number of elements and with different valid, randomly generated fields such as names, words, lists and numbers. It is important that the JSON files have a trailing newline.

For this experiment I generated files containing 10.000 objects: booksOS.json for OpenSearch and booksES.json for Elasticsearch. They contain the following information about books: the author, a couple of themes, the year it was written, a few characters from the book and its title. Note that all the information was randomly generated and it doesn’t exist in reality.

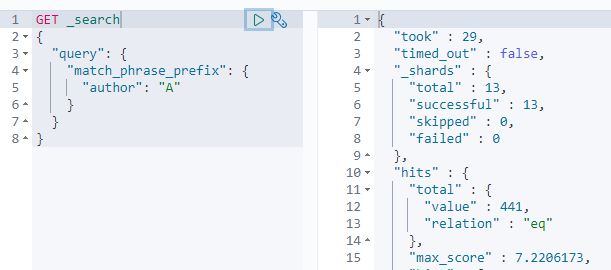
For uploading the randomly generated JSON files we will use a common HTTP client, **curl**, for brevity and convenience. The following command will be used in a local directory to upload a file into a search domain:

curl -XPOST -u 'master-user:master-user-password' 'domain-endpoint/\_bulk' --data-binary @file-name.json -H 'Content-Type: application/json'

After we are done uploading the data, we need to reindex our data such that we can work wind index aliases and store that index more cost-effective. For our example we will create an index pattern of the form: **books\*** that will help us retrieve the data more efficient from the search services.

## Comparing test results

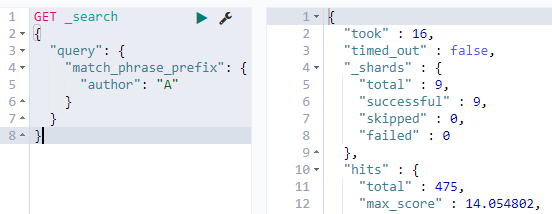
Open



New e



Old e



# Related work

Even though the differences may seem insignificant in such a small data set, when it comes to real life scenarios where petabytes of data need to be quickly accessed, that small distinction will make a difference. So looking into what kind of domain you should be choosing for your project is pretty important. But rather than taking their time comparing the older versions to the newer ones, most of the papers about Elasticsearch and OpenSearch are focused on the astonishing changes they brought in countless distinct fields.

In the paper [1] “Large-Scale Image Retrieval with Elasticsearch” it is described how Elasticsearch was used to obtain outstanding performance improvement when it comes to content-Based Image Retrieval in large archives by transform CNN features into textual representations and indexing them such that the Elasticsearch Service can operate on them.

With the growth of the Earth Science Information Partner (ESIP) federation, came also the growth of diverse data sets to enable understanding of the Earth as a system. The diversity and need of combining the information presents a challenge to finding useful data for a given study. [2] To address this issue, the ESIP is developing a federated space-time query framework, based on the OpenSearch convention. The novelty of OpenSearch was that the space-time query interface became both machine callable and easy enough to integrate into the web browser's search box. This changed offered great flexibility, simplicity and queries would run in real-time.

These search engines can be used to solve critical real life problems too. [3] Mayo Clinic, a nonprofit American academic medical center focused on integrated health care, education, and research, used to generate 0.7-2.2 million HL7 V2 messaged on a daily basis, which couldn’t be real-time stored or analyzed even with multiple RDBMS-based systems. Using Big Data technology coupled with the Elasticsearch technology, the platform could process over 62 million HL7 messages per day while Elasticsearch indexes could provide ultrafast test searching (0.2 seconds per query) on an indexes of up to 25 million HL7-derived JSON documents.

# Bibliography

[1] Giuseppe Amato, Paolo Bolettieri, Fabio Carrara, Fabrizio Falchi: Large-Scale Image Retrieval with Elasticsearch, SIGIR '18: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval (2018)

[2] Lynnes, C., Beaumont, B., Duerr, R. E., Hua, H.: Federated Space-Time Query for Earth Science Data Using OpenSearch Conventions, American Geophysical Union, Fall Meeting 2009, abstract id. IN33F-04 (2009)

[3] Dequan Chen, Yi Chen, Brian N. Brownlow, Pradip P. Kanjamala, Carlos A. Garcia Arredondo, Bryan L. Radspinner, Matthew A. Raveling: Real-Time or Near Real-Time Persisting Daily Healthcare Data Into HDFS and ElasticSearch Index Inside a Big Data Platform, IEEE Transactions on Industrial Informatics (2016)