ISSN (Print) : 0974-6846
Indian Journal of Science and Technology, *Vol 9(37), DOI: 10.17485/ijst/2016/v9i37/102108, September 2016* ISSN (Online) : 0974-5645

Research Document Search using Elastic Search

R. Vidhya[1*] and G. Vadivu[2]

[1]Department of Computer Science Engineering, SRM University, Kattankulathur - 603203, Tamil Nadu, India; vidhya.r@ktr.srmuniv.ac.in
[2]Department of Information Technology, SRM University, Kattankulathur - 603203, Tamil Nadu, India; vadivu.g@ktr.srmuniv.ac.in

Abstract

Objectives: Implementation of Elastic search server to create a search engine that helps in searching, retrieving and downloading research papers stored in Django framework database and indexed by Elastic search. **Analysis:** Elastic Search, a search server based on Lucene can be used to search all types of documents with the help of its scalability and near-real time search. **Findings:** A web application which queries and searches for relevant research papers, allows users to customize their search and suggest list of research papers related to the initial query. It displays papers' referenced authors. Item-based recommendations help users find more similar papers. **Applications:** Fast, Incisive Search against large volumes of data.

Keywords: Django, Elastic Search, Haystack, Lucene, REST

1. Introduction

The data that being generated in the present Era at such a fast rate is voluminous and complex. In addition to this, a system that processes schema-less data is a necessity. This is due to the very fact that present data is not limited to textual extensions. This wide variety ranging from music and video files to the data extracted to social media has put forward the need to have some new technologies that can support them easily. The RDBMS was never designed to handle such type of data. The case is not of its inability to evolve but something more fundamental. The way it sacrifices on availability of data over consistency and partial tolerance forms a very big problem currently[1]. They support more of batch processing and transactional services that have a particular schema, while the data that needs to be indexed may be unstructured or semi-structured. Hence there is a requirement of big data technologies that support schema-less content that may or may not along RDBMS[2]. Elastic search is one such NoSQL Big Data technology that has a promising future as a search-server to handle such large and complex data[3]. It is built to handle huge amounts of data volume with very high availability and to distribute itself across many machines to be fault-tolerant and scalable, all the while maintaining a simple but powerful API that allows applications from any language or framework access to the database. In addition, it provides its own Query Language called Query DSL that works over JavaScript Object Notations (JSON). This paper shows how Elastic search queries on datasets of research papers in portable document format (.pdf) and provide the output relevant to the query. Users can filter their search on the basis of title, author, published year and keywords. Search server will also provide item based recommendation of papers. The search-server has been integrated to web-framework Django for the front-end functioning of the project. The use of package django-haystack enables the use of elastic search with django in addition to handling fundamental functioning of elastic search.

2. Elastic Search

In[4] was initially written in Java to support free and open source information retrieval software library. But as it turns out, it was quite difficult to use since it's just a library and requires Java to work with it. Hence in early 2000s, a developer names Shay Banon started to work on an abstraction layer over Lucene that made working with search applications for Java programmers easier and

* *Author for correspondence*

named it Compass[5]. After some years, Compass libraries were rewritten to provide real-time, distributed and high-performance search engine. The standalone server was released with name of Elastic search. In[6] Elastic search is multitenant and carries out its operations using RE presentational **S**tate Transfer (REST) API. It uses basic HTTP interface to work on the lightweight JSON (javascript object notation) -queries. It's easy to start using documents as JSON documents that are far less verbose than XML and just appropriate for transmitting data sets. The REST API makes it easy to use different languages like Java, Ruby, Perl, Python, and more. It is much more advanced than traditional DBMS as it can support storage of large amount of data and provides near-real time responses to queries. The largest unit in Elastic search index and smallest unit is document. Elastic search is analogous to RDBMS Table 1. Document is indexed (stored in database) and the respective fields are mapped. The document stored is called index. Each elastic search index consists of fixed number of Lucene indices which are called shards. Elastic search manages distribution during runtime. When a new document is stored and indexed, Elastic search server defines the shard responsible for that document. This automatic sharding makes the distributive nature of Elastic search evident and balances load of shards too[7]. Elastic search server is a REST full server. Mapping in elastic search server defines how each document fields are stored, indexed and analyzed. Mapping can be implicit, already decided by the server or explicit i.e. customized which gives the chance to create complex indexes and analyze them accordingly. Explicit mapping allows filtered indexing that saves disk space and also optimal search. The search in elastic search is near-real time[8]. There is a need to refresh the server to get the result as of newly added indexes. The workflow of ES Figure1 can be explained in brief as follows[9]:

- Documents are uploaded or stored; they may be of any type and any size and in any number.
- Then the JSON Builder converts these documents from their respective type to JSON documents.
- Now, it's the duty of the Tokenizer, to break down the data into individual words.
- These words are indexed and mapping is also done so as to group the similar type of words into one mapping type. This ensures the faster retrieval of text as per the query fired by the user.
- The parser will parse the query and accordingly search and retrieve the searched text from the indexed documents.

The main advantages of elastic search are its horizontal (sharding) and vertical (powerful servers) scalability and its efficient way of handling agile data. But it lags behind in security, transactions and durability. Since it is a relatively new search tool (released in 2010), there's a need to develop mature client libraries.

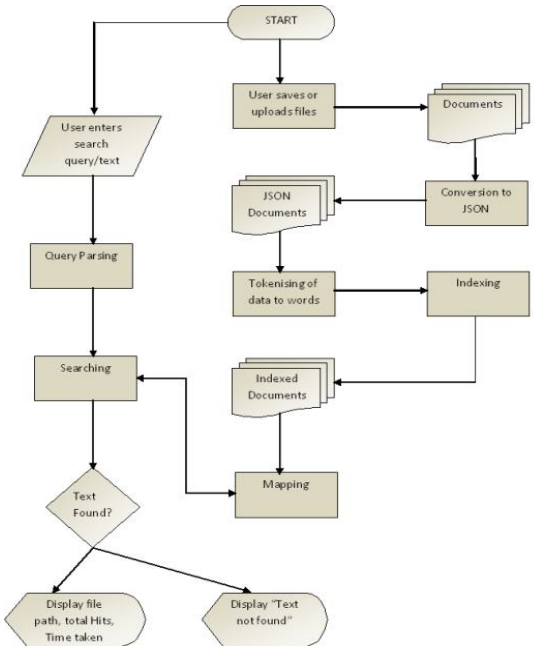

Figure 1. Elastic search Workflow.

Table 1. Analogy between Elastic search and RDBMS

Elastic Search Element	RDBMS Elements
Index	Database
Mapping	Schema
Document-Type	Table
Document	Row

3. Dataset

The dataset to carry out search and retrieval of research papers has been obtained from various sources. Some of them being from ACM digital library and Google Scholar. The data has been uploaded to the database of the Elastic search server with collaboration of db.sqlite3 which is primary storage for django modules. Haystack, which is

the communication link between Elastic search server and Django server, indexes the required fields of the papers being uploaded, example, title, author, published year. These are the fields which help the search API in the mapping of the results. Elastic search version 2.2 and Django Framework version 1.8.8 have been used.

3. Implementation

The application has the following modules:

- Site administration
- Search admin
- Search page
- Search Results
- Display Research Paper

Research Document Search application is built on Django Framework that connects the database, core code, static files and templates in one project. It is a high level Python web-framework based on MVC (model-views-controller)[10] architectural pattern. The Django Models is the data access layer, it contains all the information about the data being accessed and validates and what is the relationship between other data. Django Views form the logic that accesses the model and calls for appropriate template (html file). It is the bridge between models and templates. The controllers are used to set URL configuration that navigates to the view with appropriate user input. This configuration[11] is fundamental for all the modules.

3.1 Site Administration

The site administration provides with super user privilege. Django models are used to create fields corresponding to the document (some of them later indexed) which are title, author, keywords, published year, source and location where the media (research papers) are stored. The storage location of research paper is configured in the Django settings. These fields are added using admin module and stored in database. For these fields to appear as changeable by administrator, they need to be registered in "admin" module. Configuration of admin module depends on django-admin-tools and its template loaders.

3.2 Search Admin

After the fields are specified and sample dataset is uploaded using admin module, the indexing of the database is performed which will enable us to retrieve search results. This indexing is achieved by Search Indexes configured using haystack[12]. Search Index indexes the fields which are specified. These indexes and specified in text files in templates. Elastic search (that communicates with it using its server on HTTP interface) indexes the fields, making them searchable with appropriate logic in views.

3.3 Search Page

This is the home page of the application by default. It calls for querying the paper on the basis of the title, author, and year of publication or keywords. The logic of search is presented in Django views. The search is bifurcated into two; first being normal search and second is advanced search. Normal search takes the search text and matches it against the indexed title of the pages, if any result is found it returns the result. Advanced search searches for author, year if publication and keyword. Keyword search involves fully fledged document search. The views forward the logic to a template. The views class Search Query Set class of haystack that performs text matching with the context as present in the database. The papers that matched the query are stored in context dictionary with the number of results.

3.4 Search Results

Search page navigates to this page after the query matching. It displays the number of results that are relevant to the query and link to each one of them. The abstract is also displayed along with title, author and published in year. But the results will only be displayed if Elastic search server is on. Otherwise connection will be refused and number of results will be returned a zero. The code to display the results and its format is embedded in the respective template itself.

3.5 Display Research Paper

The search result page provides link with displaying papers in portable document format embedded on a web browser. The core logic of the template is specified in Django views. It specifies the location from where the document needs to be retrieved and obtains it primary key. They give it a unique identification. The paper is fetched from given location and so are the related information indexed along with it. List of authors involved and references cited are

listed along with it. The application also supports item-based recommendation. It searches for papers, again using Elastic search and Haystack, that are most similar or been visited recently. The interface provides direct link to the papers.

4. Future Enhancement

The document fields have been manually entered to simplify the UI. Since, Elastic search is still a new technology, it's not considered appropriate to use it as primary data source. It's more effective if kept closer to cache. But still, Elastic search can be used to parse the documents to obtain title, authors and keywords on its own so that there's no need to manually upload all the details. This can be done by converting .pdf to .txt and using a elastic search mapping plug-in Apache TIKA[13] that parses text or .doc files and automatically indexes the details. Search engine optimization can be implemented using Fuzzy Classification and Prediction[14]. The application can be configured with web-crawlers that will also fetch search results from other websites in order to widen its range and credibility[15].

5. Conclusion

Elastic search is quite accommodating in its usage and configuration. It is a good big data technology for search operations and data analytics. The performance of Elastic search depends upon the type of query, the size of result set and page size. When the result count or page size increases, the search time increases. Presently Elastic search has been adopted my major technological hubs like Facebook, Github, Amadeus to overcome shortcomings of old database technologies. It is believed, in some years, Elastic search may soon develop to be a primary big data technology.

6. References

1. Quora, Can we store big data in RDBMS. Available from: https://www.quora.com/Can-we-store-big-data-in-RDBMS-Why-and-why-not. Date Accessed: 1/08/2014.
2. NOSQL VS RDBMS - WHY THERE IS ROOM FOR BOTH. Available from: http://aisel.aisnet.org/sais2013/27/. Date Accessed: 18/05/2013.
3. Abubakar Y, Adeyi TGS, Auta IG. Performance Evaluation of NoSQL Systems Using YCSB in a resource Austere Environment. International Journal of Applied Information Systems (IJAIS). 2014 Sep; 7(8):23–7.
4. Sematext Blog "Elastic Search: Distributed, Lucene-based Search Engine. Available from: https://sematext.com/blog/2010/05/03/elastic-search-distributed-lucene/. Date Accessed: 3/05/2010.
5. The Future of Compass & Elastic search- Shay Banon. Available from: http://thedudeabides.com/articles/the_future_of_compass. Date Accessed: 07/07/2010.
6. Kononenko O, Baysal O, Holmes R, Godfrey MW, David. Mining modern repositories with Elastic search. Proceedings of the 11th Working Conference on Mining Software Repositorie. 2014. p. 328–31.
7. Elastic search refresh interval vs indexing performance. Available from: https://sematext.com/training/elastic-search/. Date Accessed: 8/07/2013.
8. You know for Search. Available from: https://www.elastic.co/blog/you-know-for-search-inc. Date Accessed: 13/07/2016
9. Sai Divya M, Goyal SK. ElasticSearch: An advanced and quick search technique to handle voluminous data. COMPUSOFT. An International Journal of Advanced Computer Technology. 2013 Jun; 2(6):171–5.
10. Holovaty A, Kaplan-Moss J. The Definitive Guide to Django: Web Development Done Right. Apress, (2nd edn). 2009 Jul.
11. Django Documentation, Available from: http://stackoverflow.com/questions/29957604/django-1-8-static-files-doesnt-work. Date Accessed: 30/04/2015.
12. Django-Haystack Documentation. Available from: https://pypi.python.org/pypi/django-haystack/2.4.0. Date Accessed: 09/06/2015.
13. Understanding Information Retrieval by Using Apache Lucene and Tika Part-1. Available from: https://dzone.com/articles/understanding-information. Date Accessed: 22/08/2014.
14. Senthil Kumar NK, Kishore Kumar K, Rajkumar N, Amsavalli K. Search Engine Optimization by Fuzzy Classification and Prediction. Indian Journal of Science and Technology. 2016 Jan; 9(2):1–5.
15. Kausar A, Dhaka VS, Singh SK. Design of Web Crawler for the Client – Server Technology. Indian Journal of Science and Technology. 2015 Dec; 8(36):1–7.