www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 6 Issue 6 June 2017, Page No. 21824-21829
Index Copernicus value (2015): 58.10 DOI: 10.18535/ijecs/v6i6.45

Paper on Searching and Indexing Using Elasticsearch

Darshita Kalyani, Dr. Devarshi Mehta
Gujarat, India

Abstract: In today's era, it is inconceivable to use traditional techniques / RDBMS to analyse the data as it is growing very quickly. Big data offers the solution for analysing large amount of data. Using technique of Elasticsearch, access to data can be made faster. Elasticsearch is a search engine based on Lucene. It is near real time search platform. Elasticsearch uses the concept of indexing to make the search faster. This paper elaborates the search technique of Elasticsearch.

Keywords: Elasticsearch, Lucene

1. Introduction

RDBMS support services that have a schema. Unstructured or semi- structured data may need to be indexed. It has become important to create a new platform to fulfil the demand of organization due to the challenges faced by traditional data. Big data helps to analyse growing volumes of structured transaction data, plus other forms of data that are often left untapped by conventional business intelligence (BI) and analytics programs [14]. Elasticsearch is a big data technology which is schema less. It is a tool used to search big data. Elasticsearch uses the concept of denormalization for search. Elasticsearch uses the indexing concept. It is a document oriented tool. Once the document is added, it can be searched within a next second as Elasticsearch is real time.

Elasticsearch is used for many use cases like analytics store, auto completer, spell checker, alerting engine, and as a general purpose document store; Full text search is one of it. It is a robust search engine that provides a quick full text search over various documents. It searches within full text fields to find the document and return the most relevant result first. The relevancy of documents is good as Elasticsearch uses boolean model to find document. As soon as a document matches a query, Lucene calculates its score for that query, combining the scores of each matching term. The relevance of the document can be calculated using practical scoring function.

Elasticsearch uses the concept of inverted index for searching. When the query is fired, Elasticsearch looks into inverted index table to find the required data. It will show the relevant document in which the term is contained. An inverted index consists of a list of all the unique words that appear in any document, and for each word, a list of the documents in which it appears. [5]. Inverted index is the most commonly used method for FTS (Full Text Search). The inverted index searches the relevant document by mapping the term to its containing document. In the dictionary the terms are sorted which results in quick search.

DOI: 10.18535/ijecs/v6i6.44

The following diagram depicts the working of Elasticsearch.

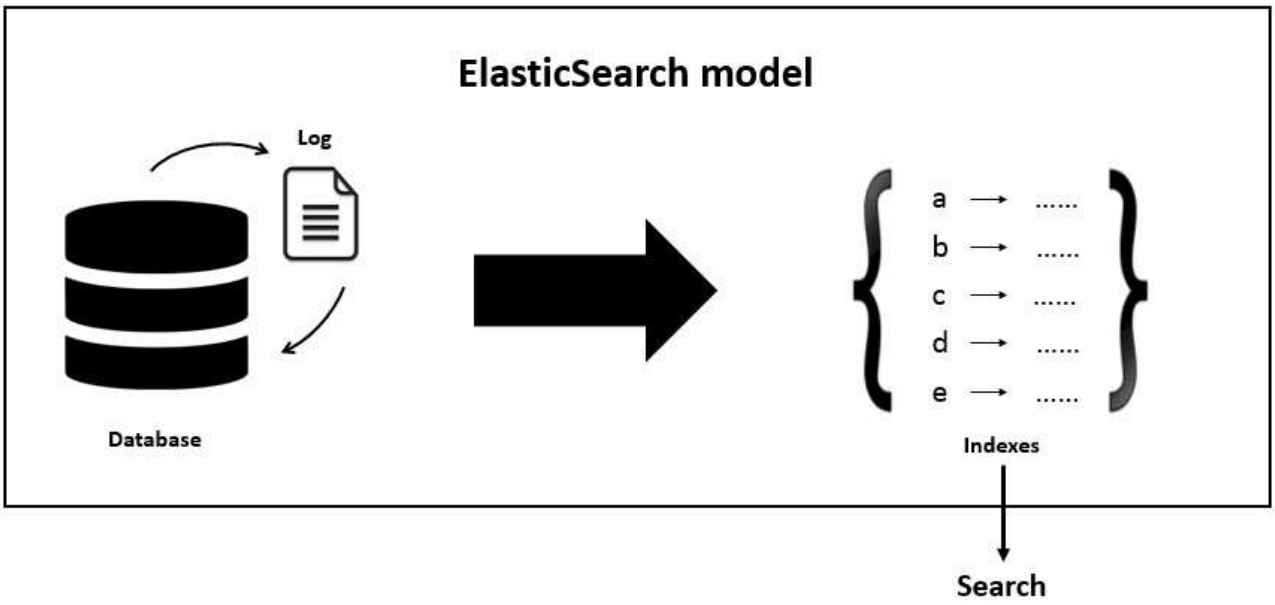


Figure 1: Elasticsearch Model [6]

Elasticsearch database is document oriented. By default, the full document is returned as part of all searches. This is referred to as the source. If the entire source document is not to be returned, then only a few fields from within source can be returned. The Elasticsearch uses the inverted index to search the term. The terms are sorted in ascending order.

2. Working of Elasticsearch

In Elasticsearch, a Document is the unit of search and index. An index consists of one or more Documents, and a Document consists of one or more Fields [4]. Elasticsearch is able to achieve fast search responses because, instead of searching the text directly, it searches an index instead.

Elasticsearch provides the ability to subdivide the index into multiple pieces called shards. When a new document is stored and indexed, Elasticsearch server defines the shard responsible for that document. When an index is created, user can simply define the number of shards. Each shard is in itself a fully-functional and independent "index" that can be hosted on any node in the cluster. Any number of documents can be uploaded irrespective of its type [2].

DOI: 10.18535/ijecs/v6i6.44

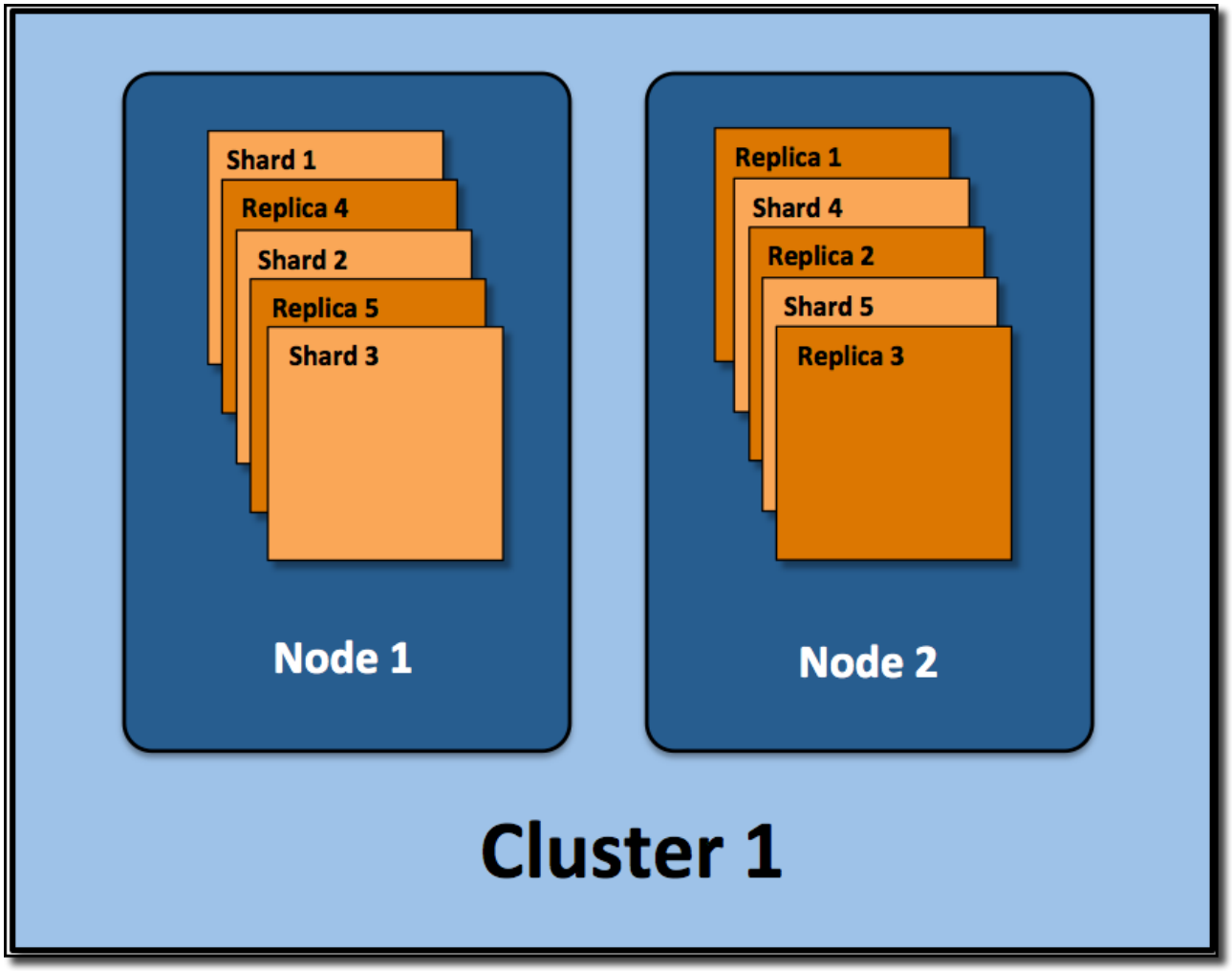


Figure 2: Elasticsearch Cluster [13]

An index is a collection of documents. 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. 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. Elasticsearch automatically arranges the five primary shards split across the two nodes. [13]

Elasticsearch indexses all fields of document by default so the field becomes searchable. An inverted index consists of a list of all the unique words that appear in any document, and for each word, a list of the documents in which it appears. Document consists of fields, and each field identified by its name and can contain one or multiple values. Each document may have different set of fields; there is no schema or imposed structure. [1]

3. Explanation of Elasticsearch working through examples
   (i) Creation of inverted index using Elasticsearch
   (ii) Working of Elasticsearch with stopword list concept

(i) Creation of inverted index using Elasticsearch

If the user wants to find articles that talk about aircraft, you may search using the word “aircraft”. If you

don't have a special indexing technique, then all the records will be scanned to find a match, which is inefficient technique. "inverted index" offers the solution for this.

The following example elaborates the searching using Elasticsearch. Consider these documents to be indexed with the contents in it.

Document1: "Elasticsearch is fast"

Document2: "Elasticsearch is efficient"

Document3: "Elasticsearch offers high speed"

Now to create the inverted index for the above documents, the contents of each document is split into separate words. After then, the lists of unique words, the document ids in which they are contained and the word frequency list are generated. So the inverted index generated for the above documents would be below:

Terms	Document	Position	Term Frequency
Elasticsearch	Document1,Document2 , Document 3	1,1,1	3
is	Document1,Document2	2,2	2
fast	Document1	3	1
efficient	Document2	3	1
offers	Document3	3	1
High	Document3	4	1
speed	Document3	5	1

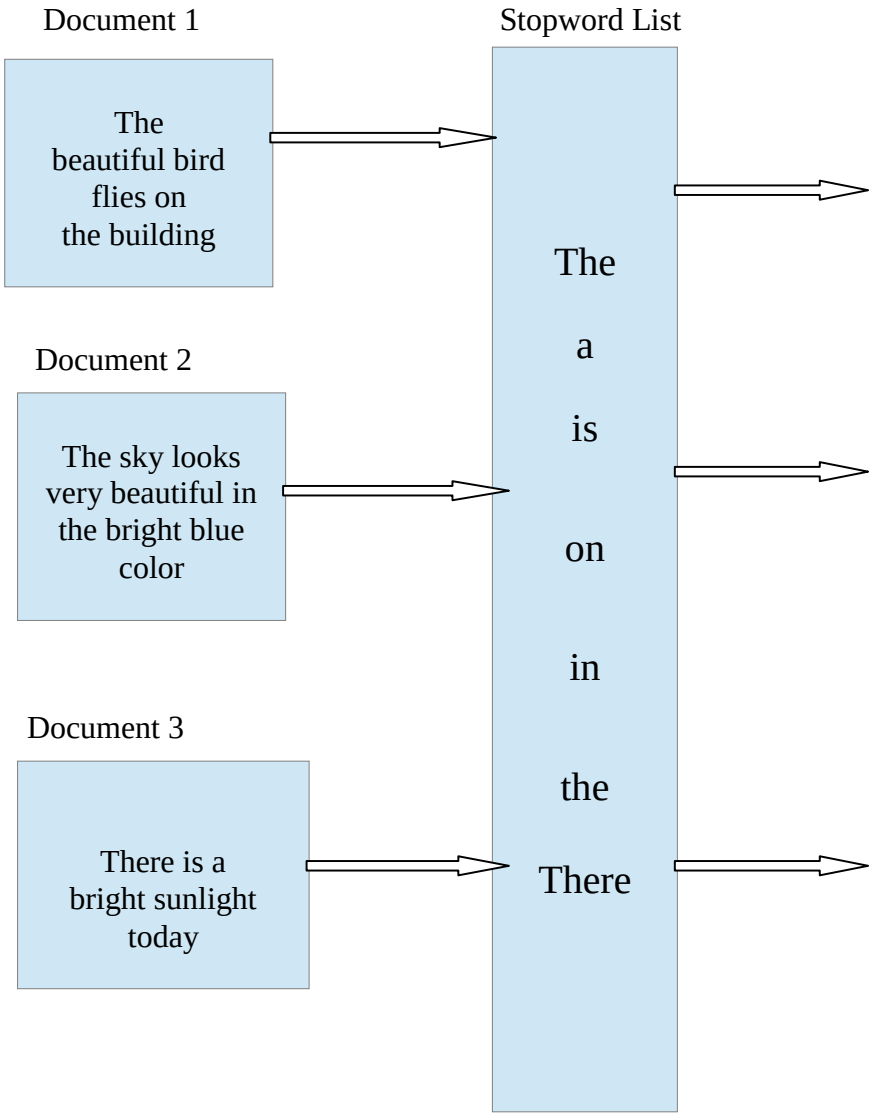
Figure 3: Creation of Inverted Index using Elasticsearch

If the query is fired to search the word "offers", then Elasticsearch will look into its inverted index table. It will find that the word occurs in document 3 and will show that document. So due to the inverted index creation, the search becomes faster.

(ii) Working of Elasticsearch with Stopword list concept

The following diagram elaborates the working of Elasticsearch. It creates an inverted index internally.

DOI: 10.18535/ijecs/v6i6.44



ID	Term	Document
1	beautiful	Document1, Document2
2	Bird	Document1
3	Blue	Document2
4	Bright	Document2, Document3
5	building	Document1
6	Color	Document2
7	Flies	Document1
8	Looks	Document2
9	Sky	Document2
10	sunlight	Document3
11	Today	Document3

Figure 4: Working of Elasticsearch with Stopword List concept

Some words Seldom add value to a search. As per the above figure, the Elasticsearch uses the concept of Stopword list to fasten the search process. Elasticsearch has its own list of predefined stopwords. *S*topwords can usually be filtered out before indexing. The inverted index is then created over the terms of document to make search faster.

3. Future Scope:

The stopword list can be improved to fasten the search. The process of tokenization and normalization which is called analysis, can be improved to fasten the search process. The practical scoring function can also be improved for better relevancy of documents.

4. Conclusion:

Search is a core part of Elasticsearch. If the analysis process is defined on the index, the retrieval process would be more efficient. Elasticsearch is a good technology of big data for searching operation. It is near real time. Elasticsearch uses inverted index for search. Elasticsearch can search full text fields and most relevant result is returned. The search becomes quicker.

References:

[1]Survey Paper on Elastic Search Pragya Gupta, Sreeja Nair International Journal of Science and Research (IJSR)
[2]R. Vidhya and G. Vadivu Indian Journal of Science and Technology, Vol 9(37), DOI: 10.17485/ijst/2016/v9i37/102108, September 2016
[3] Urvi Thacker ; Manjusha Pandey ; Siddharth S. Rautaray Electrical, Electronics, and Optimization Techniques (ICEEOT), International Cconference
[4]http://www.elasticsearchtutorial.com/basic-elasticsearch-concepts.html
[5]https://www.elastic.co/guide/en/elasticsearch/guide/current/inverted-index.html
[6]https://www.google.co.in/search?q=elasticsearch+architecture+diagram&source=lnms&tbm=isch&sa=X&ved=0ahUKEwi-osDhn LTAhXML48KHe16DRgQ_AUIBigB&biw=821&bih=572#tbm=isch&q=elasticsearch+diagram&imgrc=n0PCYMi6gSQVvM
[7]https://www.tutorialspoint.com/elasticsearch/elasticsearch_basic_concepts.html
[8]Use of Elastic Search for Intelligent Algorithms to Ease the Healthcare Industry C. Bhadane, H. A. Mody, D. U. Shah, P. R. Sheth
[9]Full-Text Search on Data with Access Control Ahmad Zaky, Rinaldi Munir, S.T., M.T. School of Electrical Engineering and Informatics Institut Teknologi Bandung
[10]Elasticsearch: How to Add Full-Text Search to Your Database:https://medium.com/@MentorMate/elasticsearch-how-to-add-full-text-search-to-your database-ee2f3ea4d3f3
[11]Use Cases for Elasticsearch: Full Text Search:http://blog.florian-hopf.de/2014/07/use-cases-for-elasticsearch-full-text.html
[12]https://www.elastic.co/guide/en/elasticsearch/guide/current/stopwords.html#stopwords
[13]https://qbox.io/blog/optimizing-elasticsearch-how-many-shards-per-index
[14] https://www.projectguru.in/publications/difference-traditional-data-big-data/