ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/311563960

Comparison of Solr and Elasticsearch Among Popular Full Text Search Engines and Their Security Analysis

Poster · October 2016

DOI: 10.13140/RG.2.2.24563.32803

CITATIONS
4

READS
2,964

2 authors:

uğut kılıç
Erzurum Teknik Üniversitesi
2 PUBLICATIONS **4** CITATIONS

SEE PROFILE

Isil Karabey Aksakalli
Hacettepe University
17 PUBLICATIONS **14** CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Bilateral Filtering and DCT based detection of copy move forgery in images View project

All content following this page was uploaded by Isil Karabey Aksakalli on 11 December 2016.

The user has requested enhancement of the downloaded file.

Comparison of Solr and Elasticsearch Among Popular Full Text Search Engines and Their Security Analysis

Uğur Kılıç[1], Işıl Karabey[2]
[1]Department of Computer Engineering, Erzurum Technical University, Erzurum, Turkey
{ugur.kilic}@erzurum.edu.tr
[2]Department of Computer Engineering, Hacettepe University, Ankara, Turkey
{isilkarabey}@cs.hacettepe.edu.tr

***Abstract*—Nowadays, a large-scale data is increasingly generated from medium and large scale companies offer service in social media, video sharing sites, communication, health, security and other fields in today's informatics world. An important part of this data has been unstructured, irregular and is not very meaningful alone at the beginning. Process to search and store this big amount of data through standard programming methods has become difficult increasingly over the time and times of process have been longer. Several search engines are produced in order to process huge amount of data safely and search in a short time. In this study, Solr and Elasticsearch that are of popular full text search engines have been compared in terms of productivity, ease of use, speed and safety and advantages and disadvantages of either search engines have been included.**

Keywords*—Database Search Engine, Full Text Engine, Lucene, Solr, Elasticsearch, Big Data, Distributed Systems.

I. INTRODUCTION

A huge scale of data is produced, at any moment, in medium and large scale companies which offer service in social media, video sharing sites, communication, health, security and other fields in today's informatics world. An important part of this data we mentioned has been unstructured, irregular and is not very meaningful alone at the beginning. Process to search and store this big amount of data through standard programming methods has become difficult increasingly over the time and times of process have been longer. Not only known companies of the world but also many local and middle scale companies have had to process this huge amount of data formed. Today, total amount of data produced increases by 40 percent every year and total amount of data becomes almost double every year [1]. If size of our data is not substantially big, process to search can be handled with filtered SQL inquiries. When the size of our data begins to increase, it is filtered more under the name of performance works and it turns into more complex SQL inquiries increasingly. And while everybody tries to make a better and faster one in the globalizing competition environment later on, it is exposed to response time delay in today's speed technology. For that reason, certain needs are in question to record, analyze and process this huge amount of data in performance. Even though there are certain programs to process this big data, especially Lucene, Solr and Elasticsearch are popular tools developed in order to deal with problems mentioned in the world of big data.

II. APACHE LUCENE

Lucene is an open source substructure of search engine which was used developed by Apache and used by Google. It constitutes substructure of Elasticsearch and Solr among search engines which keep their popularity same in today's technology. Lucene has been developed essentially with Java. Apache Lucene is a library that has been developed to fulfill search processes of full text. The data to be indexed is sent and it is indexed by Lucene over the file system. It enables indexing over the area as much as requested. When search process will be conducted, it carries out search over the file system it indexed. It presents results with more performances than search processes of full text in RDBMS. It is an ideal library to index over huge amount of data and carry out a search. Working style of Lucene is illustrated in Figure 1.

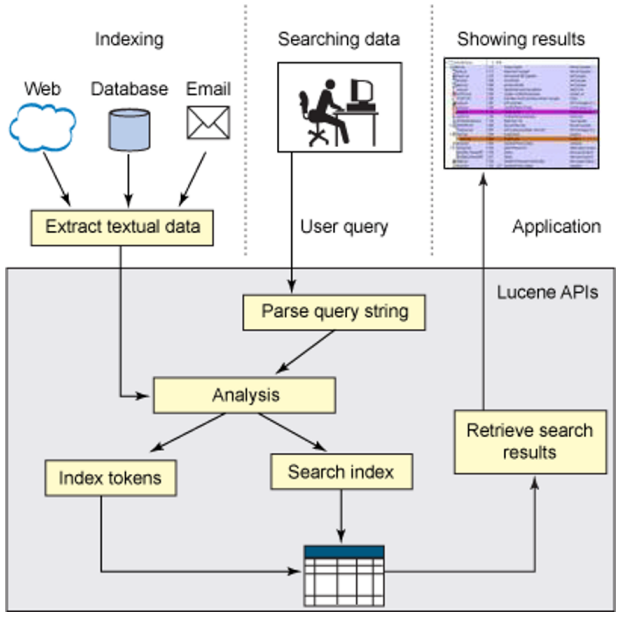

Figure 1: Working process of Lucene [2]

978-1-5090-1679-2/16/$31.00 ©2016 IEEE

Apache Lucene carries out content indexing by taking only way of documents. In this case, it is not required that manager reads and gives the content. It is enough to give way of documents for indexing. A sample search developed using Lucene is illustrated in Figure 2.

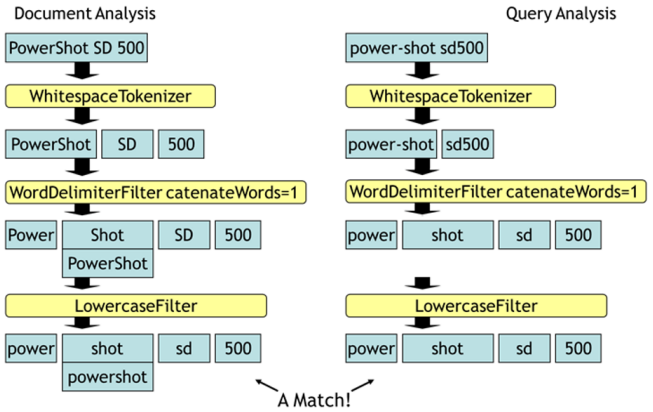

Figure 2: A sample search in Lucene [3]

III. SOLR

Full text search that is a part of Apache Lucene project is an open source and quite flexible search engine which has characteristics such like multi-directional search, mobile clustering, being integrated with database, indexing documents like Microsoft Word or PDF. Solr supports also NoSQL characteristics along with its 4th version. With Solr 5, it has stopped being an application that has Java package and operated with "war" and it has turned into a different application. Today, Solr is used in content searches, data analysis and inquiries for projects such like Instagram, SourceForge, eBay. Solr operates such like REST type API server using Lucene libraries. Ist operates with protocols supported by several programming languages such like HTTP/XML or JSON, it can operate independently with Java and it supports the libraries allowing high-level customizations. Apache Lucene and Apache Solr were written separately by improvement team of Apache Software Foundation, as for 2010 year, these two projects were associated. While Lucene essentially offers basic functions such like text analysis, indexing and searching, Solr uses base of Lucene and also offers advanced filtering, highlighting, multi-dimensional searching, caching, Rest Api, distributed architecture support [4].

A. Rest Api

Solr provides more advanced and flexible layers over Lucene indexing substructure. Index of data sources can be changed over Http protocol by using Rest Api. Also, applications making presentation over Http protocol can carry out search over Solr. Usage of Solr Rest Api is illustrated in Figure 3.

B. Content Insertion

Content insertion to Solr is carried out using Http based UpdateHandler as seen in Figure 4.

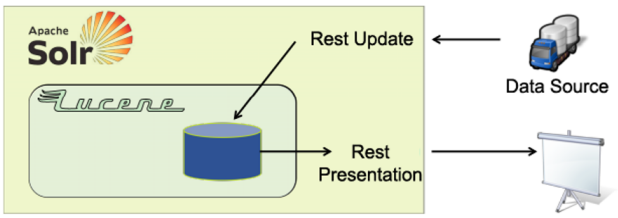

Figure 3: The usage of Solr Rest Api [5]

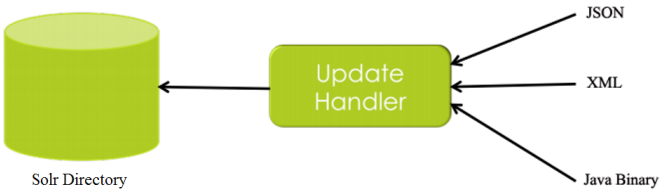

Figure 4: Content adding in Solr [5]

C. Caching

Solr finds the documents in which the word is mentioned, for every word into every query index. This process starts to take time as the index grows. Memory cache is one of the most critical effects for performance in search engines. Memory cache keeps the word lists in the memory. When a query comes with the same word, it gives response from the memory.

D. The Usege of Solr

Solr operates as default in 8983 port. When http://localhost:8983/solr/ address is visited on browser, a management panel interface is seen as in Figure 5. Current collections are listed in lower-left corner of the page. For testing; All documents can be viewed with http://localhost:8983/solr/collection1/query?q=* [6].

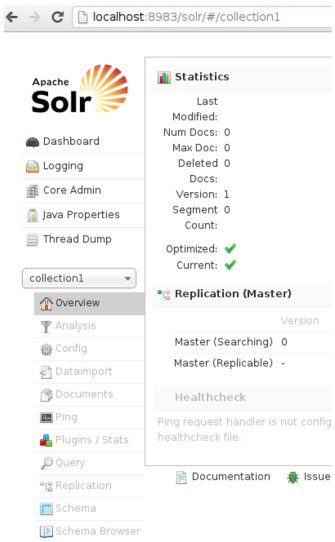

Figure 5: Solr management panel

E. Enabling a basic authentication in Solr

The following steps are performed to enable a basic authentication:

Step 1: JSON file namely security.json is saved as shown in below:

```
{
"authentication":{
"class":"solr.BasicAuthPlugin",
"credentials":{"solr":
"IV0EHq1OnNrj6gvRCwvFwTrZ1+z1oBbnQdiVC3otuq0=
Ndd7LKvVBAaZIF0QAVi1ekCfAJXr1GGfLtRUXhgrF8c="}
},
"authorization":
"class":"solr.RuleBasedAuthorizationPlugin",
"user-role":{"solr":"admin"},
"permissions":[{"name":"security-edit",
"role":"admin"}]
}
```

In this code, first of all authentication and authorization are enabled and a user called "solr" and a password named 'SolrRocks' is created. Then a role 'admin' is assigned to user. The permissions to security edits are restricted by role: 'admin' [7]

Step 2: security.json file is uploaded to ZooKeeper

```
server/scripts/cloud-scripts/zkcli.sh -zkhost localhost:9983
-cmd putfile /security.json security.json
```

After this operation all solr nodes will watch the /security.json and this change affect the nodes' behaviour. The operation is verified with the following commands:

```
curl http://localhost:8983/solr/admin/authentication
curl http://localhost:8983/solr/admin/authorization
```

F. BasicAuthPlugin in Solr

The BasicAuthPlugin uses HTTP's Basic authentication mechanism and a user is authenticated using this mechanism. Authentication is done against the user name and salted sha256 hash of the password stored in ZooKeeper. Editing credentials An API is exist to add, edit or remove users, but this commands as shown in below is specific to Basic authentication implementation mentioned above , so the implementation class must be solr. BasicAuthPlugin, if not, the same set of commands are not valid [7].

Example 1: Adding a user and editing a password

```
curl –user solr:SolrRocks
http://localhost:8983/solr/admin/authentication
-H 'Content-type:application/json'-d '{
"set-user": {"tom" : "TomIsCool" ,
"harry":"HarrysSecret"}}'
```

Example 2: Deleting a user

```
curl –user solr:SolrRocks
http://localhost:8983/solr/admin/authentication
-H 'Content-type:application/json'-d '{
"delete-user": ["tom","harry"]}'
```

IV. ELASTICSEARCH

Elasticsearch is a search engine and data analysis tool that has been developed from Apache Lucene substructure and that is light, easily installed, open source coded and scalable. This search engine that offers service over Restful API is very fast and practical thanks to its certain 3rd part visual tools and safety options. Elasticsearch is less complex, less detailed compared to Solr search engine produced from Apache Lucene but it is as successful as Solr. It is strong and flexible. One of the most important advantages is that it is real-time and distributed. Today Elasticsearch is used in content searching, data analysis and queries in the projects such like Mozilla, Foursquare, GitHub [8].

A. Full Text Searching

As the data stored at databases grows, we encounter speed/performance problems in query operations carried out over this data. For a remedy, words that are in text fields are indexed and catalogued. By this means, it is ensured that databases give responses that are faster and with more performance even it is practiced with a large scale data. Elasticsearch has many full text search capacities such like multi-language option, a strong query language and autocomplete.

B. Index

Elasticsearch is a document oriented search engine. Each record in Elasticsearch is a structured JSON document. In other words, each data sent to Elasticsearch for indexing is a JSON document. All fields of the document are indexed as default and can be used in single query. Compared to database management systems, Elasticsearch indexes can be considered as databases. Just as a database is a regular group of information, Elasticsearch indexes are structured group of JSON documents.

C. Type

Data types can be considered as tables again compared to database management systems. Indexes may include one and more types.

D. Mapping

Mapping is the period to define how a document should be transferred to search engine. While types are created, mapping information is used. Elasticsearch creates the mapping automatically from the data sent (for example, string, integer, double, Boolean) (explicit mapping). A default mapping can be invalidated by defining a mapping.

E. Restful Api

Elasticsearch is Restful Api oriented. Each action can be carried out with Restful Api using JSON over HTTP.

F. Elastic

Elasticsearch does not request definitions such like index, type, field type before the indexing process as it is schema free. When a record is inserted, Elasticsearch tries to determine type and index of the data and makes it searchable. If required, index, type, field and field type definitions can be changed before or after record insertion. Here, it is important to comprehend the flexibility offered by Elasticsearch. Of course it is an important advantage to be able to keep documents of different type, name and amount into the same index. For example, field and field types must be defined previously in Solr that is a different popular full text search engine. When it is intended to insert a new field, it is required to transfer all available records to Solr again. Elasticsearch, as being without these limitations, may be likened to table and column independency provided by NoSQL architectures[9].

G. Cluster

Elasticsearch has been constituted as horizontal scaled. When more capacity is required, it is enough to increase number of node. In this case, cluster will reorganize itself again in order to benefit from extra hardware. In the same network, cluster name and standard Elasticsearch installation at the same x number will find each other automatically and connect. Client support is available for many platforms such like Java, Php, Python, Perl, Ruby, .NET.

H. Basic Steps for Text Processing

Independently of language of the text, basic steps in text processing:

- Determination of the language and field, if possible, such like Turkish text, sports, literature, military, social media field. This reference to be used is required for the processes in following steps which will be performed by using a corpus (collection).
- Separation of the text into words (tokenization),
- Cleaning the surplus markings and unnecessary words (stopwords),
- Standardization of remaining words and finding the stem (normalization, stemming),
- Determination of the likes of words (synonyms),
- Correction of the faulty words (types and misspellings),
- Keeping properly (indexing)
- No matter what language these processes are in, it is valid for all text processing and these steps are carried out voluntarily or involuntarily when text processing is performed by using an available library. When viewed from this aspect, it is seen that Elasticsearch has a pretty explicit and comprehensible logic of processing and that it provides the user with conveniences to process texts and allows uses to perform this complex process with default values.

First of all, language and a domain, if possible, should be determined. The reason is that corpus to use, while text is processed, is selected accordingly. Corpus is a catalogue in which any word is kept systematically we can encounter in the works written in a language. By looking this catalogue, it can be determined if a word is in that language, is misspelled, and whether this word could be at the beginning or at the end of the sentence. Two different methods are used to determine the language in which document was written, and Elasticsearch provides the users with these methods. First of them, a standard language is determined and configuration is made accordingly, and if there will be any language apart from this, a parameter is reported. As it has become an obligation to determine an automatic text language for those who are interested in especially multi-lingual texts, chromium-compact-language-detector supports different languages which is an open source project offered by Elasticsearch in this kind of situations. Determination of the domain provides certain advantages in terms of restricting Corpus while processing the text, it is not an obligation. It provides recovery especially in query performance [10].

I. Transferring data to Elasticsearch

To put the text into Elasticsearch means indexing in terminology of Elasticsearch. The reason is that one index is corresponded to each word through processes such like separating the words, eliminating the surplus in the period from the second step to seventh step above. This index created is called inverted index. Transferring data to Elasticsearch can be explained basically. Words are written into the first column in Table I, one column is inserted for each document and details if the word is present in that document is determined with a bit across each word.

Words	Documents		
	Doc1	Doc2	Doc3
Word1	1	1	0
Word2	0	0	1
Word3	1	1	1
Word4	0	1	1
Word5	0	1	0

Table I: Elasticsearch indexing method – Inverted index

J. Inverted Index

This kind of processes are intended to give response quickly during the query and return the document list suitable for the query to the user. In queries, words mentioned in the query are brought with the relevant documents by means of inverted index using Boolean Model that is an information access method. This model allows to use logical operators such like "and", "or" and "not" and finds easily and quickly which documents of the index table are proper to the query. Query: (Word1 and Word3) or Word5 Response: doc1, doc2 The reason why Doc3 doesn't return in the above query is that word1 is not found in doc3. Even though word5 is not found in doc1 or as it is connected with doc1, it is found in the response. In this phase, it becomes important to the results would be offered to the user in which line, especially in the situations where the number of results is much. Therefore, Elasticsearch

produces values between the range f (0,1) in score parameter using substructure of Lucene and making very complicated calculations. To know how these values are calculated, which we will qualify them as relevance values, is the response to how intended quality of query would be increased.

K. Relevance Value

Algorithms, used in document databases in order to determine and even array which one of available documents should be returned to the user through a query done, is maybe the most important one of basic reasons why similar effective results could not be obtained from the relevance databases used for this work. The following three values calculated for each word become prominent in this regard.

- Term frequency (tf) If one of the words in the query passes six times in the document, then the document in which it passes six times is considered as more relevant than the document in which word passes once. Where T is term or word, d is document or domain;

$$tf(tind) = sqrt(frequeny) \quad (1)$$

- Inverse document frequency (idf) Inverse document frequency (idf) is a value which indicates passing frequency for words in the documents into collection. It is based on the acceptance that a word passes into more documents, the more its distinguishing characteristic is weak. For that reason, the less idf value is, the more weight of the word is increased. The bigger it is, the more it is reduced. Full value of idf is calculated with formula 2.

$$df(t) = 1 + log(numDocs/(docFreq + 1)) \quad (2)$$

- Domain length rule (norm) The more length of documents or domain is, the less importance this word will have in terms of being present in that document. In this regard, a word mentioned in the title is more important than a word mentioned in subject.

$$norm(d) = 1/sqrt(numTerms) \quad (3)$$

Calculation of these values is made at the indexing phase when the document is registered to Elasticsearch for the first time. Elasticsearch uses the following method in order that a score is produced using these three values calculated for each word and also this process is performed for queries containing more terms just as in many queries. Relevance problem is turned into the problem where vectors would find angle between each other, in vector spaces, using vectors created with these existing values. In this way, scores are obtained by making vector calculations in multi-dimensional space between the vector obtained from query and vector of each document. These calculations become input to Practical Scoring function of Lucene. [11]

L. Query Performance Increasing

It is need to know what is efficient to calculate score parameter and how these would be used in order to increase query results. Depending on these details, the following are some of the first things coming to mind to increase query performance in Elasticsearch.

- It is important to introduce the synonymous words. Term frequency can be increased in this way.
- It is important to normalize the words. It is generally recommended to not insert the words into indexing which they are considered as useless in the query such like pronouns expected to pass in most documents, for example “this, that and certain interrogative particles.
- To emit the field length, unnecessary words in the indexing such like "and", "or" that pass frequently in the documents increase the performance.

Elasticsearch is pretty elastic in terms of both configuration and use compared to the counterparts, it is a very attractive option for dealing with searching processes which might result in I/O bottlenecks in the systems operated with big data, and for data analysis [12].

M. Creating a simple secure layer over Elasticsearch

As well as Elasticsearch is installed easily, it is a practical and advanced Text search engine. It doesn’t comprise any safety layer and standard loading package. 2 types of approach exist in providing safety in Elasticsearch. First of them Is that containers such like docker or jetty provide safety in which Elasticsearch is installed. As for the second approach, Elasticsearch meet the deficit with its safety plugins. In this study, we have established a Elasticsearch secure layer with the second approach.

```
String url = "http://127.0.0.1:9200/_search;
URL object = new URL(url);
HttpURLConnection con = (HttpURLConnection) object.openConnection();
String authStr = "ISIL:KARABEY";
byte[] authEncBytes = Base64.encodeBase64(authStr.getBytes());
String authStringEnc = new String(authEncBytes);
con.setRequestProperty("Authorization","Basic "+authStringEnc);
con.setDoOutput(true);
con.setDoInput(true);
con.setRequestProperty("Content-Type", "application/json");
con.setRequestProperty("Accept", "application/json");
con.setRequestMethod("POST");
OutputStreamWriter wr = new OutputStreamWriter(con.getOutputStream());
wr.write(parent.toString());
wr.flush();
```

When it is tried to reach Elasticsearch with this safety layer, user name, password and/or IP control is performed. There are many safety plug-ins for Elasticsearch on internet. First and most popular of them is Asquera/ Elasticsearch-http-basic. This plug-in simply inserts user name and password/IP based authorization and logging characteristics into Elasticsearch. In the three following steps, it is explained hor the relevant plug-in will be installed and how it can be inserted into our JAVA program.

Step 1: Downloading Jar

Jar file is downloading from address called “https://github.com/Asquera/elasticsearch-http-basic/releases

(in this study, Elasticsearch 1.5.1 version has been used). The downloaded Jar file is loaded to the file called "elasticsearch/plugins/http-basic". It is required to create a file called "http-basic" into Plugins file. If Elasticsearch has been used using dpkg, it is required to insert Jar to both"/usr/share/$NAME/plugins/http-basic" file and"/usr/share/$NAME/bin/plugins/http-basic" file.

Step 2: Organize the settings

Elasticsearch.yml file is opened that is in config file of Elasticsearch and the following code blocks are inserted into the bottom line of this file.

```
http.basic.enabled: true
http.basic.log: true
http.basic.user: "ISIL"
http.basic.password: "KARABEY"
http.basic.whitelist: ["localhost","127.0.0.1"]
```

Variables can be organized or removed at user's pleasure. For example, if certain Log or IP based lines are removed, then system will continue to perform identity validation through user name and password. The first line is mandatory. When this change is performed for the first time and/or during each period when user name, etc. is changed later, Elasticsearch must be restarted.

Step 3: Java Program is organized

```
JSONObject cred = new JSONObject();
JSONObject auth = new JSONObject();
JSONObject parent = new JSONObject();
cred.put("query", "UGUR");
auth.put("query_string", cred);
parent.put("query", auth);
```

As in the sample code above, HttpURLConnection Class is used in order to send one JSON to elastic. With the following code part, elastic search JSON is sent without any identity validation.

```
String url = "http://127.0.0.1:9200/_search;
URL object = new URL(url);
HttpURLConnection con = (HttpURLConnection) object.openConnection();
con.setDoOutput(true);
con.setDoInput(true);
con.setRequestProperty("Content-Type", "application/json");
con.setRequestProperty("Accept", "application/json");
con.setRequestMethod("POST");
OutputStreamWriter wr = new OutputStreamWriter(con.getOutputStream());
wr.write(parent.toString());
wr.flush();
```

It is got in contact by performing elasticsearch identity validation by means of inserting the code lines mentioned below with red into the code part above in order to send elasticsearch JSIN by making identity validation with configuration carried out in Step 1.

V. COMPARISON OF ELASTICSEARCH AND SOLR

As Elasticsearch is schema-less; it doesn't request the domains such like index, type, field, field type in advance. While inserting a record, it can be inserted as the index is specified by x, type is by y, field and field values by user. X index, y type, field and field type are formed depending on the types specified by user. These definitions can be changed before or after inserting the record, as well. As for Solr, field and field types are required to be specified in advance. While trying to insert a new field, it is required to transfer all the record to Solr again. Sharding and Replication numbers are specified in advance and cannot be changed in Solr. These changes can be performed at the beginning of scanning, not in advance in Elasticsearch. Let's think we scan IM log, change default definitions for this index or type. The changes made recently will be valid for the records came recently. Previous IM record is not included in these definitions. When we finally mention cluster structure in comparing Solr and Elasticsearch, when two computers of which standard Elasticsearch installation is performed are at the same network and same cluster name, they see each other automatically. As for Clustername, it can be changed from the line into config/elasticsearch.yml. Elasticsearch doesn't need to do anything else [13] [14].

REFERENCES

[1] D. İ. Tolga, Y. Doğan,C. Şener "Büyük veri anlamlandırmada panoramik yaklaşım", UYMS, 2014.

[2] http://www.ibm.com/developerworks/library/os-apache-lucenesearch/, Access date: 09.07.2016

[3] M. McCandless, E. Hatcher, O. Gospodnetic, "Lucene in Action: Covers Apache Lucene 3.0", Manning Publications Co, 2010

[4] T. Grainger, and T. Potter, Solr in action. Manning Publications Co, 2014, https://www.manning.com/books/solr-in-action, Access date: 25.07.2016

[5] E. Gönenç,"Açık Kaynak Arama Teknolojileri:Lucene, Solr ve Nutch", Mantis Yazılım Ltd, 2012

[6] D. Smiley, and E. Pugh, "Solr 1.4 Enterprise Search Server", Packt Publishing Ltd, 2009

[7] C. Targett,"Apache Solr Reference Guide:Authentication and Authorization Plugins",Apache Software Foundation, 2015

[8] Elasticsearch Wikipedia, the free encyclopedia, http://en.wikipedia.org/wiki/Elasticsearch, Access date: 18.07.2016

[9] R. Kuć,M. Rogoziński, "Mastering ElasticSearch", Packt Publishing Ltd, 2013

[10] M. S. Divya, S. K. Goyal, "ElasticSearch: An advanced and quick search technique to handle voluminous data", COMPUSOFT : International Journal of Advanced Computer Technology, 2013

[11] O. Kononenko, O. Baysal, R. Holmes, M.W. Godfrey, "Mining modern repositories with elasticsearch",Proceedings of the 11th Working Conference on Mining Software Repositories, ACM, 2014

[12] G. Clinton, Z. Tong, "Elasticsearch: The Definitive Guide", O'Reilly Media, Inc, 2015

[13] http://solr-vs-elasticsearch.com/, Access date: 02.08.2016

[14] J. Manyika, M. Chui, B. Brown, J. Bughin,R. Dobbs, C. Roxburgh, A. H. Byers, "Big data: The next frontier for innovation, competition, and productivity", McKinsey Global Institute, 2011

[15] A. A. Müngen, S. kayaokay, G. Yılmaz, "Özgür Teknolojiler ile Büyük Miktarda Dosya için Gelişmiş Bir Dosya Sistemi Yapısı Önerimi", XVII. Akademik Bilişim Konferansı, Eskişehir, 2015

[16] IDC Big Data and Business Analytics Forum, 2013. http://idc-cema.com/eng/events/54106-idc-big-data-and-business-analytics-forum-2013, Access date: 11.07.2016

[17] S. Ömer, E. Kılıç, "Coğrafik Yer Bilgilerinin Elde Edilmesi ve Sorgu Genişlemesi Yöntemi ile Sorgulanması", XVII. Türkiye'de İnternet Konferansı, Eskişehir / Türkiye,2012

View publication stats