International Journal of Science and Research (IJSR)
ISSN: 2319-7064
SJIF (2020): 7.803

A Comprehensive Study of Elasticsearch

Nikita Kathare[1], O. Vinati Reddy[2], Dr. Vishalakshi Prabhu[3]

[1, 2]Student, Dept. of Computer Science and Engineering, R V College of Engineering, Bengaluru, India

[3]Assistant Professor, Dept. of Computer Science and Engineering, R V College of Engineering, Bengaluru, India

Abstract: ***With the ever-increasing demand for data storage, querying and retrieving data from abundant data sources is a tedious and time-consuming task. Hence, we require a system for querying data that is highly available, has high capacity and can scale out easily without the need to add more hardware onto a single device. In the paper, we discuss one such heavy full-text search and analytics engine called Elasticsearch. Elasticsearch is designed to work with various types of data such as structured, unstructured, geospatial, graphical and numerical data. It was built on top of Lucene and has been improvised with better features. The power of Elasticsearch is amplified with the help of a number of technologies that provide a visualization platform, data processing pipeline, monitoring, machine learning, data shipping etc. They, together with Elasticsearch, are called the Elastic Stack (ELK Stack). Comparison of Elasticsearch with other recent search engine technologies such as Solr, Sphinx and Azure search is provided, which would help readers better understand which technology to choose. Elasticsearch is being used in a number of organizations today as a powerful search engine and has been preferred over databases like MongoDB for querying over stored data, both being JSON document oriented, distributed datastores. But Elasticsearch provides a better searching capability like full-text search unlike MongoDB which is only preferred for CRUD operations. Elasticsearch is also relatively very fast compared to its counterparts and comes with real-time search capabilities thereby having negligible latency, hence making it viable to analyze billions of documents within a few seconds. Besides that, it also has a high throughput, being able to search through and analyze a number of documents concurrently within a limited response time. Elasticsearch also deals with failure of any node of a cluster and loss of shards on it by replicating primary shards into a number of replica shards and distributing them across multiple nodes. This distributed nature of Elasticsearch makes it highly available and robust.***

Keywords: cluster, Elasticsearch, Elastic stack, node, search engine, shard

1. Introduction

Elasticsearch[1] is a heavy full-text search and analytics engine. It is an open source and distributed search engine which is capable of handling various types of data such as alphabetic, numerical, structured as well as unstructured data. It is a distributed document store system built on top of Lucene[2]. Because of its distributed nature, it is said to be a highly available search engine and can also be scaled easily. It bestows us with JSON based REST API which helps us cite Lucene features. It facilitates us to perform some very complex data aggregations, some of which can't be supported by Lucene. Elasticsearch is a resource hungry search engine and requires a large heap space, Lucene[3] on the other hand requires a small heap of about 1GB, however the problem can be solved by either freezing the shards or distributing the load over multiple nodes. The relationship between elasticsearch and Lucene is like that of a vehicle and its engine, hence we can say the former is powered by the later. Elasticsearch is more than just a search engine, it can be used for a variety of other applications like analytics, document store, auto suggesting. Lucene is an aberrant tool in itself and does not overwhelm us with the choice of APIs as elasticsearch does.

There are databases like MySQL[4] that store data and help us query over it. Unlike MySQL, Elasticsearch is a JSON document store that uses a method of indexing, where it creates inverted indices for the input text, which makes Elasticsearch very fast and nearly in-real time search engine that produces results of a search within a few milliseconds. Also, Elasticsearch is a part of ELK stack or Elastic Stack[5], which helps us visualize data using Kibana and helps us in shipping data using Logstack.

2. Key Concepts

One of the main upgrades in Elasticsearch over Lucene is that it is distributed in nature and can be scaled over clusters of nodes. This feature of elasticsearch is because of its architecture whose understanding is crucial in knowing how elasticsearch operates.

1) **Document:** Data within Elasticsearch is stored in the form of JSON objects which are considered to be the smallest unit of data storage. Documents correspond to rows of a table in a relational database[6]. Each of the documents comprise of fields which correspond to columns of a table. An index consists of one or more documents and documents in turn have one or more fields. Data stored in the documents is queried by the values of the fields in it. As soon as a document is put into an Elasticsearch index, an inverted index is created for it and it becomes fully searchable in real time.
2) **Type:** Documents that are stored in Elasticsearch have a specific type[7]. The type of the documents is defined by the _type field of that document. Elasticsearch allows us to store documents of different mapping and different type within the same index.
3) **Index:** Index in Elasticsearch is similar to a database in SQL. An index may consist of one or more documents in JSON format, each of which may be of different type and may have different mapping but are in some way related to each other[8]. Elasticsearch stores index in one or more primary shards, for which corresponding replica shards are created.
4) **Mapping:** Mapping in Elasticsearch is a way of defining the structure of the documents i.e. the fields in the documents, the datatype of values stored in each of the fields and the metadata associated with the type of

document. It is analogous to the table schema of a relational database. In Elasticsearch, there are two basic approaches to mapping, dynamic and explicit. Using explicit mapping the users can define fields and their data types on their own. But to make Elasticsearch easier to use, dynamic mapping was introduced which creates a field mapping automatically when a new field is encountered whose mapping was not specified explicitly by the user, for example if a field with string value was created, Elasticsearch would map that field to having a text datatype. Hence explicit and dynamic mapping can be combined, hence making mapping flexible in Elasticsearch.

5) **Node:** A node is an instance of Elasticsearch that is responsible for storing data and indexing it. A collection of nodes makes up a cluster. All nodes in a cluster have information about every other node, and they forward the requests from client to the appropriate node. Nodes can take up a number of different roles, which can be specified by the user or they are set by default. The responsibilities or the roles taken up by nodes are as: master, data, client, tribe, ingestion and machine learning nodes. The master nodes are responsible for overseeing the management of the cluster and configuring them, by creating and removing nodes. Data nodes store data and carry out operations on that data. The client nodes act as mediators that balance the request load by forwarding the cluster-related request to the master node and the data-related requests to the data nodes. The tribe nodes perform read and write operations on all the nodes in the cluster and it connects one or more clusters making them seem like one big cluster. Ingestion nodes are used for preprocessing the documents before indexing them and the machine learning nodes help in carrying out machine learning tasks.
6) **Cluster:** A cluster in Elasticsearch is a group of one or more nodes that work together. Elasticsearch is distributed in nature, which is a property that is induced by having the capability of adding nodes to it and grouping them into clusters, thereby reducing the load on a single node and dispersing it amongst multiple nodes.
7) **Shard:** An index in Elasticsearch[9] is divided into a number of shards which are then distributed across a number of nodes, hence we can say that an index is a logical integration of one or more shards. The documents in an index are distributed across multiple shards and these shards are in turn distributed across multiple nodes. When the load on a particular cluster grows, Elasticsearch migrates some shards from that cluster to other clusters, thereby balancing the data load. There are two kinds of shards: primary and replica. Whenever an index is being created, the user can specify the number of shards it is supposed to have, i.e. the number of primary and the number of replica shards for each of the primary shards. The data of the index is divided amongst a number of primary shards, hence the primary shards have the original copy of the data, while the replica shards for each of the primary shards hold the copy of data of that primary shard, thereby increasing the redundancy of data and preventing loss of data due to failure of a node. Every index in Elasticsearch is composed of at least one primary shard as it consists of the original copy of data. Fig-1 shows an index in elasticsearch which is divided into three shards. The shards are then dispersed across three nodes with one replica shard each. Even if one of the nodes in the Elasticsearch cluster goes down, there would remain at least one copy of each of the shards, thereby making it a highly available technology.

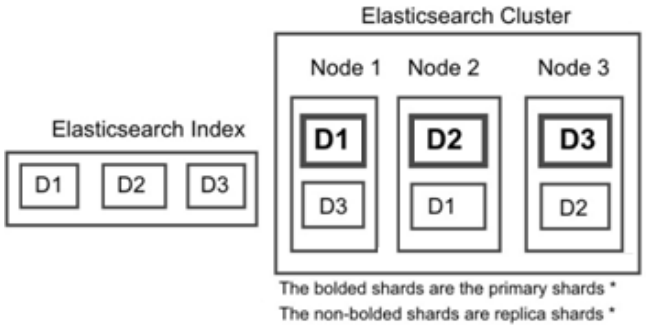

Figure 1: Primary and Replica Shards in Elasticsearch

3. Text Search in Elasticsearch

Analysis or text analysis is a process that is applicable to the text fields or values[10]. In elastic search text has to be processed before being stored, this processing happens in the analysis phase. Text values are analysed when indexing documents by an analyzer and the result is stored in data structures that would make the process of searching more efficient.

An analyser consists of three building blocks : character filters, tokenizers and token filters. A character filter receives the original text and transforms it by adding, removing and changing characters. An analyser may have one or more character filters that are applied in a particular order as specified by the user . But an analyser can contain only one tokenizer that tokenizes a string by splitting it into tokens. Some characters such as punctuations and white spaces may be stripped as a result of tokenization for example splitting a sentence into words. The tokenizer also records the character offsets for each token[11]. Token filters receive the output of the tokenizers as input and they add, remove or modify tokens. Similar to Character filters an analyser may contain one or more Token filters and are applied in the order in which they are specified for example the lower case filter that converts all the characters in each of the tokens to lowercase. Elasticsearch comes with a number of built-in analyzers, character filters, tokenizers and token filters. A number of different combinations of character filters, tokenizers and token filters can be used by the user to build a custom analyzer. The analyzer used by Elasticsearch by default is the standard analyzer which does not consist of a character filter but uses a Standard Tokenizer which tokenises by removing white spaces and punctuation and a lowercase Token filter.

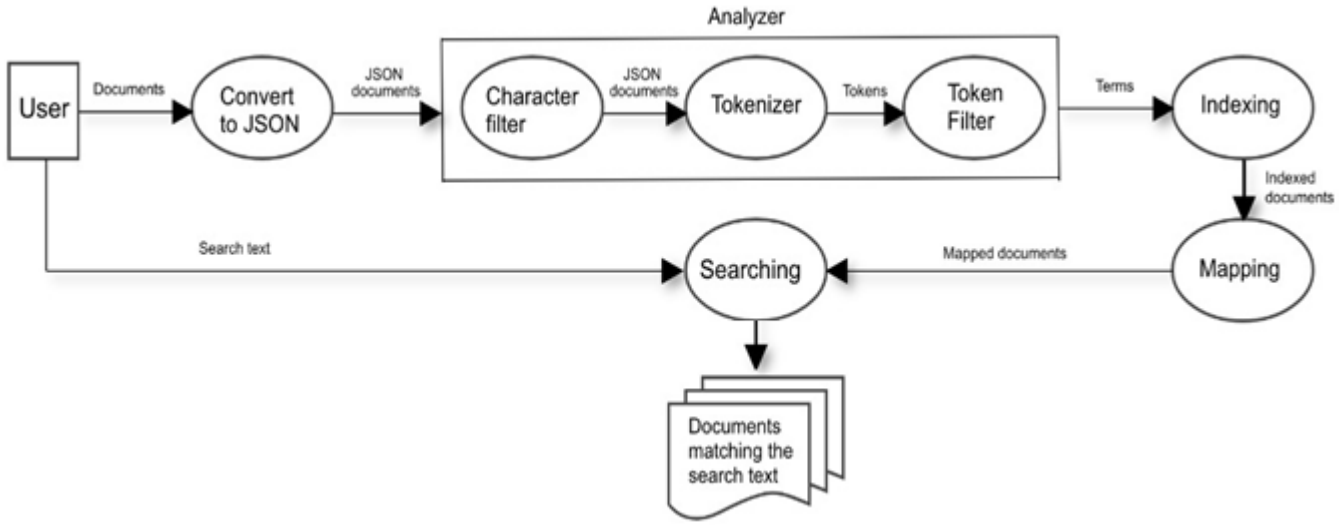

Figure 2: Text Searching in Elasticsearch

The tokens from the analyzer are stored in data structures, a different data structure being used for different fields depending on the field's data types. Using several data structures for storing the field values instead of one ensures efficient data access. These data structures for text fields make up inverted indices which help in fastening full-text searches. Inverted indices are mapping between terms i.e. the tokens from the analyzer and documents containing them. Terms in inverted indices are sorted alphabetically. Inverted indices also contain information about relevance scoring, which while performing full text search helps return documents based on how well they match to the search text. Inverted index is created for each text field in the documents and the fields of data types other than text use a different data structure like BKD trees for numerical and date data types.

After indexing the documents are mapped. Mapping is used to define the structure of documents in Elasticsearch and configure how they're indexed. Mapping is done by specifying the properties of the fields of the documents and their data types, i.e. equivalent to the schema of a table in relational database. Mapping here can either be done explicitly by the user or implicitly by Elasticsearch.

4. Features

Elasticsearch exhibits a number of features as follows -

1) Highly Scalable - Elasticsearch can scale horizontally upto few petabytes of structured as well as unstructured data. We can increase the capacity of storage by adding more nodes to the cluster. Though there is no upper limit on the size, the preferred limit per shard is 50 GB.
2) Highly Secure - All the data stored in Elasticsearch can be password-protected to prevent any unauthorized users from accessing the data. Elasticsearch also provides various other security mechanisms such as role-based access control, access control based on attributes, audit logging, IP filtering and communication encryption.
3) Highly available - Elasticsearch is based on the concept of using clusters, where clusters are an assemblage of one or more nodes or servers which together hold all of the data and provides amalgamated indexing and search functionality across all nodes. Clusters in Elasticsearch feature primary and replica shards to impart failover, if in case a node goes down. When a node containing primary shard shuts down and goes offline due to some problem, a replica shard is promoted as the primary shard thereby making the whole cluster highly available.
4) Full text Search Engine - Traditional SQL database management systems are not designed for full-text searches against vast quantities of data. Whereas, Elasticsearch offers one of the most powerful full-text search capabilities and can perform and combine various types of searches, from structured, unstructured, geo, to metric data.
5) Analytics - Other than being used to build a complex search engine using its text querying capabilities, Elasticsearch can also be used to query structured data such as numbers and aggregate data and is hence used as an analytics platform. The data can be queried and analyzed pictorially with the help of line charts, pie charts.
6) Index Management - Elasticsearch provides a suite of features to monitor and manage indices. Index State Management provides an automated system for defining custom policies and for optimizing, monitoring and managing indices. It eliminates the need to rely on external systems to periodically execute the tasks. Index State Management plugin from Kibana provides users facility to monitor the indices and apply custom policies such as criteria based on index age, size and number of documents

5. The Elastic Stack

The Elastic stack consists of technologies developed and maintained by Elastic NV, the company behind Elasticsearch. Elasticsearch is the heart of the Elastic stack, i.e. most of the technologies that are part of the Elastic stack interact with Elasticsearch and have a strong synergy between them , hence they are frequently used together. The products that are a part of Elastic stack are:

1) Kibana - It is an analytics and visualization platform that easily lets us visualize data from Elasticsearch and analyze it, which helps us understand it better[12]. It is comparable to a dashboard or an interface where visualizations of data can be created, for example maps

(coordinate map , region map) and charts (pie, line, area and bar chart)[13].

2) Logstash - It is a free, open source and lightweight server side data processing pipeline that consumes data from various sources and sends them to Elasticsearch[14]. The data that Logstash receives can be handled as events like log file entries , e-commerce orders, customer details, chat messages etc. These events are then processed by Logstash and shipped off to one or more destinations like Elasticsearch, Kafka queue , a HTTP endpoint etc. A Logstash pipeline consists of three stages : i) Inputs ii) Filters iii) Outputs . Each stage makes use of a plugin. While the input plugins are how Logstash receives the events, the output plugins are all about how Logstash processes them. An output plugin is where the processed events are sent, which are formally called stashes. Multiple pipelines can be run under the same Logstash instance, and it is horizontally scalable.
3) X-Pack - It is an extension to Elasticsearch and Kibana that adds extra features to them like security, monitoring, machine learning, alerting and reporting. While providing security it facilitates the users with authentication by integrating with authentication providers and helps control permissions with fine-grained authorization. It helps monitor the performance of Elastic stack like CPU and memory usage, disk space etc. by providing an insight into how it is running. Alerting is specific to the monitoring of Elastic stack that gives an alert to the user if something erroneous happens, for example if the web server's CPU usage exceeds a certain limit or if the application errors reach a threshold. Reporting helps export Kibana visualizations and data to another file format like PDF or CSV. X-Pack is also what enables Kibana to use machine learning to perform abnormality detection, forecasting future values on data which is a functionality provided by X-Pack whereas the interface is provided by Kibana. One of the most significant features is the Graph, that helps analyze relationships in data and uses the relevance feature of Elasticsearch to determine which parts of the data are related and also provides a plugin for Kibana to visualize data as an interactive graph. Graph exposes an API that helps integrate this ability into applications.
4) Beats - Beats is a collection of data shippers. They are lightweight agents that can be installed on servers which send data to Logstash or Elasticsearch. There are a number of data shippers like filebeat which collects various log files like access logs and error logs and sends these log entries to Logstash or Elasticsearch , metricbeat that collects system and service metrics like memory and CPU usage, packbeat that collects network data (HTTP requests or database transactions), auditbeat that collects audit data from Linux, Winlogbeat that collects windows event logs etc.

Comparison of Elasticsearch and Other Open Source Search Engines

1) Comparison between Elasticsearch and Solr -
Both Elasticsearch and Solr[15] are open source search engines that are built on top of Lucene, but they vary in terms of scalability, performance, optimized query execution, cluster management and shard placement[16]. Shard placement in Solr is static in nature and usually requires manual work for migrating shards whereas in Elasticsearch, shard placement is dynamic where migration of shards is automated based on cluster state. SolrCloud supports splitting of existing shards but not shrinking of shards like Elasticsearch. Cluster coordination in Elasticsearch uses built-in Zen discovery modules whereas SolrCloud requires Apache Zookeeper, an additional service. In case of a shard or node failure, Elasticsearch does shard rebalancing itself and rarely requires manual intervention[17]. In SolrCloud, rebalancing is complex and hard to manage. Routing is supported by Solr but not by Elasticsearch.

2) Comparison between Elasticsearch and Sphinx -
Both Elasticsearch and Sphinx are well known open source search engines but they differ in some features such as memory and scalability. Elasticsearch consumes more memory hence it is scaled over multiple nodes whereas Sphinx consumes less memory as compared to other search engines. Sphinx works more tightly with structured data associated with relational databases, like MySQL whereas Elasticsearch can handle various types of data from structured, unstructured to graphical type of data. Sphinx can't index document types such as pdf, ppt, doc directly. To handle text documents in various formats, the textual contents are imported into a database, or into an XML format that Sphinx can understand and later, processing is performed. Sphinx is written in C++ whereas Elasticsearch is written in Java. Elasticsearch engine allows executing aggregation queries in search indices. Elasticsearch engine, along with optimized querying also speeds up the generation time of layered navigation block and lists of products filtered by some attributes. However, Sphinx search engine does not allow to perform aggregation queries.

3) Comparison between Elasticsearch and Azure Search
Azure search[18] is a cloud based service that provides searching as a service for mobile and web application development. Azure search is powered by Artificial Intelligence (AI) for easy identification, analysis, and exploration of data. It helps in reducing the vast complexity of data ingestion as well as index creation, using its unique storage solutions and offers index functionality. Azure search supports a number of languages as compared to Elasticsearch which supports a large number of data types. Elasticsearch has in-memory capabilities using Memcached and Redis Integration, whereas Azure search doesn't support in-memory capabilities. Elasticsearch supports eventual consistency whereas Azure search supports immediate consistency.

6. Conclusion

Searching is one of the key features of elastic search. It can be used to search documents based on diverse constraints and gives near real time search facility. It uses inverted indices for searching which makes the process very fast. Elasticsearch provides users with the facility of setting scoring schemes for text searching so the documents are returned in order of their relevance scores. It's architecture

is distributed in nature hence elastic search has a failure recovery mechanism that ensures that data is not lost even if some node in the cluster fails, which is what makes it highly available.

References

[1] R. Vidhya, G. Vadivu, "Research Document Search Using Elasticsearch", Indian Journal of Science and Technology, Vol 9(37), DOI: 10.17485/ijst/2016/v9i37/102108, September 2016
[2] Mitra, M. J. (2016), " The Rise of Elastic Stack" (November), https://doi.org/10.13140/RG.2.2.17596.03203
[3] Cornelia Gyorodi, Robert Gyorodi, George Pecherle, and Andrada Olah, " A comparative study: Mongodb vs. Mysql", 13th International Conference on Engineering of Modern Electric Systems (EMES), Oradea, Romania,11–12 June 2015; pp. 1–6
[4] Clinton Gormley & Zachary Tong, Elasticsearch, "The Definitive Guide : A Distributed real-time search and analytics engine", O'Reilly, January 2015
[5] Sematext Blog "Elastic Search:Distributed, Lucene-based Search Engine. Available from: https://sematext.com/ blog/2010/05/03/elastic-search-distributed-lucene/
[6] Pankaj Sareen, P.K.,"NoSQL Database and its Comparison with SQL Database",Int. J. Comput. Sci.Commun. Netw.2015,5, 293–298
[7] Oleksii Kononenko, Olga Baysal, Reid Holmes, Michael W. Godfrey, "Mining modern repositories with elasticsearch", ICSE '14: 36th International Conference on Software Engineering, Association for Computing MachineryNew YorkNYUnited States
[8] Li, X.-M., & Wang, Y., "Design and Implementation of an Indexing Method Based on Fields for Elasticsearch",2015 Fifth International Conference on Instrumentation and Measurement, Computer, Communication and Control (IMCCC). doi:10.1109/imccc.2015.137
[9] Kalyani, D., & Mehta, D. (2017). Paper on searching and indexing using elasticsearch. Int. J. Eng. Comput. Sci, 6(6), 21824-21829.
[10] Voit A., Stankus A., Magomedov Sh., Ivanova I., "Big data processing for full-text search and visualization with elasticsearch", International Journal of Advanced Computer Science and Applications",2017T8 No12. C.76-83. DOI: 10.14569/IJACSA.2017.081211
[11] Subhani shaik, Nallamothu Naga Malleswara Rao, "Enhancement of Searching and analyzing the document using Elastic Search",International Research Journal of Engineering and Technology (IRJET),Volume: 04 Issue: 11 | Nov -2017
[12] Neel Shah, Darryl Willick, Vijay Mago, "A framework for social media data analytics using Elasticsearch and Kibana", Wireless Networks. doi:10.1007/s11276-018-01896-2 , 2018
[13] Shah, N., Willick, D., Mago, V., "A framework for social media data analytics usingElasticsearch and Kibana" - Wireless Networks (2018, in press)
[14] Marcin Bajer, "Building an IoT Data Hub with Elasticsearch, Logstash and Kibana", 5th International Conference on Future Internet of Things and Cloud Workshops, 2017
[15] Vikash Kumar and P.N. Barwal, "Implementation of Highly Optimized Search Engine Using Solr", International Journal of Innovative Research in Science, Engineering and Technology, Vol. 5, Issue 3, March 2016
[16] Nikola Luburić, Dragan Ivanović, "Comparing Apache Solr and Elasticsearch search servers", 6th International Conference on Information Society and Technology ICIST 2016
[17] M. A. AKCA, T. Aydoğan, and M. İlkuçar, "An Analysis on the Comparison of the Performance and Configuration Features of Big Data Tools Solr and Elasticsearch", IJISAE, pp. 8-12, Dec. 2016.
[18] Pratiksha P. Nikam and Ranjeetsingh S. Suryawanshi, "Microsoft Windows Azure: Developing Applications for Highly Available Storage of Cloud Service" , International Journal of Science and Research (IJSR) ISSN (Online): 2319-7064 Index Copernicus Value (2013): 6.14 | Impact Factor (2014): 5.611

Author Profile

Nikita Kathare, B.E., RV College of Engineering - nikita.kathare@gmail.com

O. Vinati Reddy, B.E, RV College of Engineering - reddyvinati@gmail.com.

Dr. Vishalakshi Prabhu, PhD, Assistant Professor at RV College of Engineering - vishalaprabhu@rvce.edu.in