Wireless Networks (2022) 28:1179–1187
https://doi.org/10.1007/s11276-018-01896-2

A framework for social media data analytics using Elasticsearch and Kibana

Neel Shah[1] · Darryl Willick[1] · Vijay Mago[1]

Published online: 11 December 2018

Abstract
Real-time online data processing is quickly becoming an essential tool in the analysis of social media for political trends, advertising, public health awareness programs and policy making. Traditionally, processes associated with offline analysis are productive and efficient only when the data collection is a one-time process. Currently, cutting edge research requires real-time data analysis that comes with a set of challenges, particularly the efficiency of continuous data fetching within the context of present NoSQL and relational databases. In this paper, we demonstrate a solution to effectively adsress the challenges of real-time analysis using a configurable *Elasticsearch* search engine. We are using a distributed database architecture, pre-build indexing and standardizing the Elasticsearch framework for large scale text mining. The results from the query engine are visulized in almost real-time.

Keywords Social media · Big data · Real-time analysis · Elasticsearch · Visualization

1 Introduction

The exponential growth of online data poses a significant challenge in the process of fetching a representative data set that can be translated into tangible results [1, 2]. Pre-processing in real-time adds another layer of complexity, especially when the data is textual and unstructured [3] or crowd sourced [4]. Solutions to processing big data sets in the fields of cloud computing and storage are growing at rapid speed, but when we consider big data on a scale of petabytes [5], cloud based analytics are limited by network inefficiencies for transporting the data; and recurring costs for the computational resources required to perform analysis in real-time [6]. Access and privacy also pose a challenge in cloud based storage as server administrators maintain the rights to view both the data and its flow. Security solutions such as encrypted searching are not feasible to implement specific to real-time analysis because of computational limitations [7]. Currently, the top three tools used for analyzing large databases are Elasticsearch, Hadoop and Spark [8]. Elasticsearch is a distributed search and analytical engine which allows for real-time data transformations, search queries, document stream processing and indexing at a relatively high speed. Additionally, Elasticsearch can index numbers, geographical coordinates, dates and almost any datatype while supporting multiple languages (i.e., Python, Java, Ruby). The speed of the Elasticsearch engine is founded on its ability to perform aggregation, searching and processing the index of the data [9]. Hadoop is a distributed batch computing platform, using the MapReduce algorithm, that includes data extraction and transformation capabilities. While the platform is based on NoSQL technology that makes uploading unstructured data easy, its query processing HBASE does not have advanced analytical search capabilities like Elasticsearch. Elasticsearch is a text search and analytics tool with a visualization plugin for real-time analysis with an open source license. Finally, Elasticsearch hosts plugins for Hadoop and Spark to reduce the distance between the two different technologies and allows for a hybrid system to be implemented [10].

✉ Vijay Mago
vmago@lakeheadu.ca

Neel Shah
nshah5@lakeheadu.ca

Darryl Willick
dwillic1@lakeheadu.ca

[1] Deparment of Computer Science, Lakehead University, Thunder Bay, ON, Canada

Tools that support the management of large data sets and real-time data fetching include *relational* (MySQL, Oracle Database, SQLite), *Graph* (Neo4j, Oracle Spatial) and *NoSQL* (MongoDB, IBM Domino, Apache CouchDB). Limiting factors related to all types of databases include lack of support for full-text searches in real-time. While NoSQL is functional for full text searching it lacks reliability when compared to relational database models [3]. Traditional databases require that the data is first uploaded and then the administrator must actively decide which data should be indexed which adds one more layer of processing making it infeasible for real-time analysis. Elasticsearch provides a solution to these limiting factors [3] by providing a highly efficient data fetching and real-time analysis system that:

- Performs pre-indexing before storing the data to avoid the need to fetch and query specific data in real-time;
- Requires limited resources and computing power in relation to traditional solutions; and
- Provides a system that is distributed and easy to scale.

The capacity for Elasticsearch to contribute to high efficiency, real-time data analysis is enhanced through a standardized configuration process, *shard size* management and standardizing the data before upload into Elasticsearch and demonstrated through a discussion of both the working architecture as well as a real-time visualization of social media data collected during December 2017 and May 2018, a repository of over 1 billion twitter data points.

1.1 Key contributions

- Optimizing and standardizing twitter data for Elasticsearch
- Creating a configuration file and choosing the optimal *shard* size
- Demonstrating the real-time visualization of a very large scale social media data set

2 Architecture for real-time analysis and storage

2.1 Elasticsearch

Elasticsearch was started in the year 2004 as an open source project called *compass*, which was based on Apache Lucene [11]. Elasticsearch is a distributed and scalable full-text search engine written in Java that is stable and platform independent. These features combined with requirement specific flexibility and easy expansion options are helpful for real-time big data analysis [12]. We will discuss some of the general functions of Elasticsearch to provide context for the Elasticsearch configurization and data standardization and *shard* management procedure resulting from this research.

2.2 Abstract view

Figure 1 illustrates the framework for real-time analysis of very large scale data based on Elasticsearch and Kibana [13]. In the first step, the Twitter API is used for scraping twitter data (approximately 1400 tweets per minute) that is stored in a MongoDB database, which is installed on a Network Attached Storage (NAS) with a capacity of 16TB. The twitter data is transfered to preprocssing units which handle the data and transfer it to High Performance Computing (HPC) infrastructure in *almost* real-time. As traditional databases, including MongoDB, are not efficient enough to handle real-time query, we transfer the processing and analsis of data to Elasticsearch, which is implemented via HPC lab resources. Before uploading the data, we standardize the twitter object for Elasticsearch and use multithreading to upload the data for better real-time performance and to shorten the gap between receiving and processing data. When a user needs any data, a query will be sent to Elasticsearch using the Kibana front-end. Elasticsearch processes that query and sends the query result object (JSON format) to Kibana, where Kibana shows the query object to the user.

Within the general functioning of the search engine, Elasticsearch uses a running instance called a node which can take on one or more roles including a master or a data node (see Sect. 2.1, Fig. 2). Dataset clusters within Elasticsearch require at least one master and one data node, however it is possible that a cluster can consist of a single node since a node may take on multiple roles. The only data storage format compatible with Elasticsearch is JSON and therefore requires data mapping for producing functional analysis and visualizations due to the unstructured format of the twitter data. We observed that reliance on the JSON format makes the system more flexible than MySQL and other RDBMS, but less than MongoDB. While a traditional database such as RDBMS use tables to store the data, MongoDB uses BSON (like JSON) format, and Elasticsearch uses an inverted index via the Apache Lucene architecture to store the data [11]. A typical index in Elasticsearch is a collection of documents with different properties that have been organized through user defined mapping that outlines document types and fields for different data sources; similar to a table in an SQL database. The index is then split into *shards* housed in multiple nodes where a shard is part of an index distributed on different nodes. Within the Elasticsearch framework, the inverted index allows a more categorical storage of big data sets

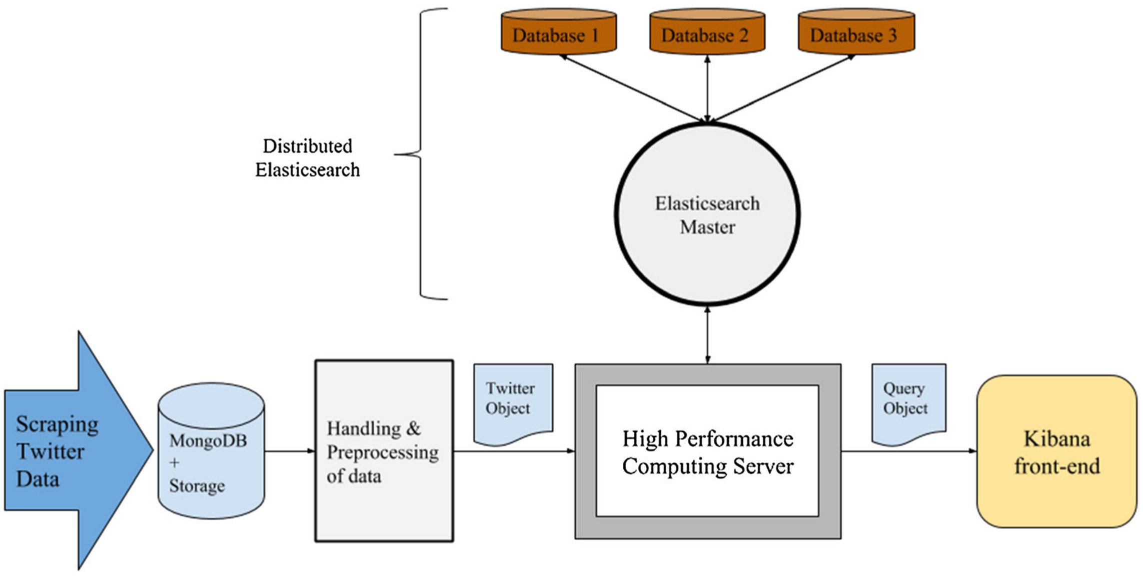


Fig. 1 Framework for real-time analysis using Elasticsearch

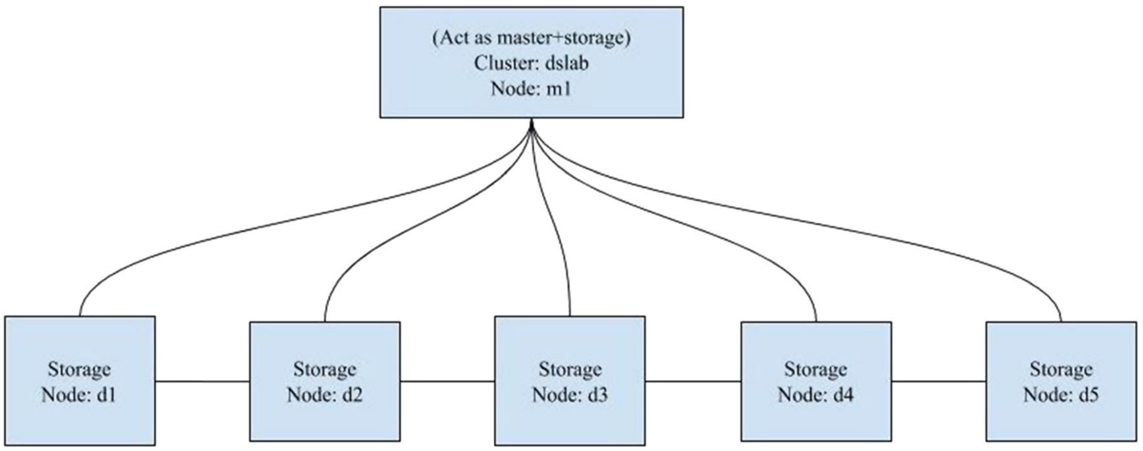


Fig. 2 Elasticsearch cluster architecture hosted on the HPC at Lakehead University

within nodes and *shards* so that real-time search queries are more efficient. Elasticsearch uses RESTful API to communicate with users, see Table 1 for a basic architecture comparison. Additionally, there are different libraries such as Elasticsearch in Python [14] and Java [15] for better integration.

Table 1 Comparison between Elasticsearch and RDBMS basic architecture

Elasticsearch	RDBMS
Index	Database
Mapping	Table
Document	Tuple

2.2.1 Backbone

While Elasticsearch is a powerful tool, a model is required to optimize functionality for the purpose of real-time big data analysis specific to social media. The purpose of this research is to provide (1) a specific configuration file to optimize the organization of the data set, (2) an optimized *shard* size for maximum efficiency in storage and processing, and (3) a standardized structure for data fields present within Twitter to eliminate over-processing of irrelevant information When the data is stored in Elasticsearch, it stores the data in an index first, and then the index data is stored as an inverted-index using an automatic tokenizer. When we search in Elasticsearch, we get a 'snapshot' of the data, which means that Elasticsearch does not require the hosting of actual content but instead links to documents stored within a node to provide a result through the inverted index. These results are not real data but a representation of the query's linkages to all associated documents stored in each node. As a component of this project, the following configuration file was developed and can be replicated in Elasticsearch on any HPC by editing the config files as per number of nodes and capacity of server. Table 2 describes the basic configuration file for Elasticsearch.

Here, the name of a cluster is *dslab* and a cluster name is necessary, even if only a single node is present. As the Elasticsearch is a scattered database, where one or many nodes work as heads and others as data, this parameter is used to interconnect all the nodes in the cluster. We can create numerous clusters with the same hardware using different instances of Elasticsearch and different configuration files.

Table 3 is an example of a configuration file features for any Elasticsearch node. In every node for the distributed Elasticsearch we have to configure the same file in each and every instance. When the data is stored we use the index to store a specific type of data similar to a dataset in MySQL. The performance of Elasticsearch is based on the mapping of the index and how we size the *shards* of the data set. The formula to decide the size of the *shards* is given in Eq. 1.

$$Number\ of\ shards = (Size\ of\ index\ in\ \ \mathrm{GB})/50 \qquad (1)$$

The reason behind the consideration of using 50 GB as a *shard* size is due to the architecture in Elasticsearch. The architecture supports 32 GB index size and 32 GB cache memory so ideally the *shard's* memory should be less than 64 GB and through experimentation we observed that the best results are achieved at *shard* size of 50 GB.

2.3 Kibana: visualization

In addition to Elasticsearch being efficient for real-time analysis, extended plugins such as kibana [13] and Logstash [16] make it convenient for functional representations of big data in real-time. It is part of the *elastic stack* and is freely available under open source license. Kibana has multiple standard visualizations available by default and simplifies the process of developing visualizations for end users with a drag and drop feature. As Kibana is backed by the Elasticsearch architecture, it functions quickly and is efficient enough for real-time analysis. Finally it provides the opportunity for graphical interaction in the process of building and handling queries with an accessible visualization of the cluster health and properties within the database.

3 Social media data analysis

3.1 Configuration of the Elasticsearch

Live social media streaming data is stored in elastic clusters. Each elastic cluster contains 6 nodes, with each node having 2 threads and 12 GB of memory. Within these 6 nodes one node works as a master and the remaining 5 work as data nodes. Architecture of the elastic cluster is shown in Fig. 2.

3.2 Social media dataset

We used Elasticsearch to analyze 250+ million out of 1 billion tweets scraped between December 2017 and May 2018 using the Twitter API. Since the Twitter API response is in JSON format and contains unstructured and inconsistent data the sequential collection of all data fields within the tweet JSON object is not guaranteed. Standardization of the data and conversion into a structured format is therefore necessary for Elasticsearch mapping so that each field of data is present when loaded into the index. To optimize the Elasticsearch we changed the storage format of the tweet so that all the data is required to

Table 2 Master and data node configuration file

Master node config file	Data node config file
cluster.name: dsla	cluster.name: dslab
node.name: m1	node.name: d1
node.master: true	node.master: false
node.data: true	node.data: true
path.data: /data/nshah5/dataset	path.data: /data/nshah5/dataset
path.logs: /data/nshah5/log	path.logs: /data/nshah5/log
network.host: x.x.x.x	network.host: x.x.x.x
network.bind_host: 0	network.bind_host: 0
network.publish_host: x.x.x.x	network.publish_host: x.x.x.x
discovery.zen.ping.unicast.hosts: ["x.x.x.x"]	discovery.zen.ping.unicast.hosts: ["x.x.x.x"]
bootstrap.system_call_filter: false	bootstrap.system_call_filter: false

Table 3 Elasticsearch node configuration file features

Config file properties	Explanation
cluster.name	It is the name of cluster where present node will join.
node.name	It gives the name of your current node
node.master	The role of master-eligible is decided based on true or false function (Boolean function). The master node manages the overall state of the cluster including node monitoring, index creation and deletion, and *shard* to node assignments.
node.data	The role of data is decided based on true or false function (Boolean function). It stores the physical data *shards*, performs reads, writes, searches and aggregations. Any node can be master and data, both or individual.
path.data	The location of the actual data in present node is represented.
path.logs	Location where the logs of the present nodes are stored. Logs are important to diagnose problems and monitor working status.
network.host	It's an address of the present node which is unique for the individual node in the cluster.
network.publish_host	It's a public address where other nodes communicate with the present node.

be at *depth level* one in JSON format. Table 4 depicts the basic example of restructured data in Elasticsearch.

As we mentioned previously, the data is stored as an inverted index that is optimized for text searches and therefore very efficient. For example, if we search for the keyword "pizza" within the context of all tweets (250+ millions) in Elasticsearch, the time taken is 4060 ms (4.06 s) to find a total of 192,118 tweets where the "pizza" keyword is present in tweet text. Table 5 shows the example of the keyword "pizza" text search query response from Elasticsearch. Figure 3a shows a pie chart of tweets mapping the geographical distribution by nation of "pizza" tweets where the United States alone is responsible for 47% of total tweets and other countries excluding the top five are 30%, which is 77% of total tweets. Additionally, the visualization shows the time taken to perform the query is 13 ms (0.013 s). Figure 3b shows five most used languages in the tweet text related to "pizza" where the English language is used in more than 77% tweets while Spanish is used 12%, Portuguese at third spot with 6%, French at 3% and Japanese at 2% tweets. In this instance Elasticsearch took 17 ms for query processing.

Table 4 Difference between normal and updated structure

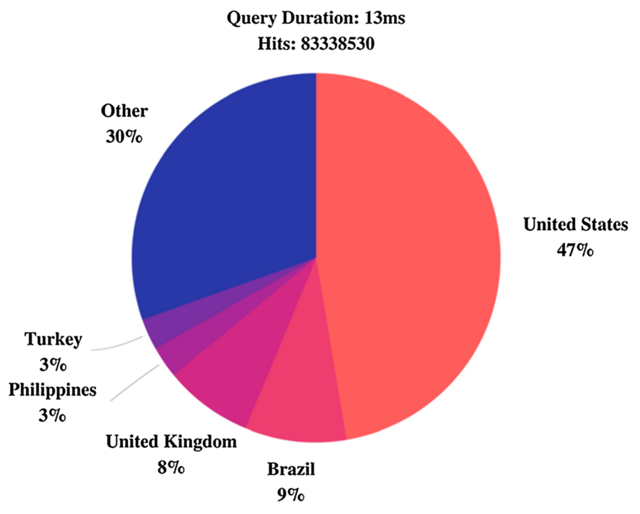
Original tweet structure	Updated structure
{	{
"??Tweet":{	"Id":
"User"??:{	"Name":
"Id"??:	...
"Name"??:	}
}	
},	
...	
}	

Table 5 Search query result of "pizza" keyword

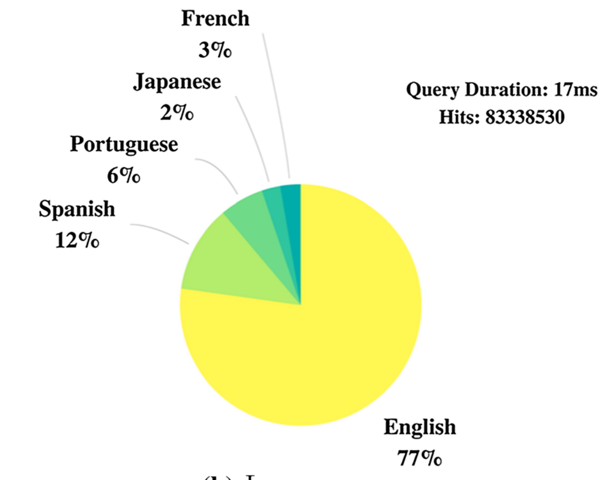
Result of keyword "pizza" from all tweets from database
{
"took": 4060,
"timed out": false,
"shards": {
"total": 106,
"successful": 106,
"skipped": 0,
"failed": 0
}
"hits": {
"total": 192118,
"max_score": 15.110959,
"hits": [???]
}

Figure 3c shows the devices used to tweet with 38% of tweets coming from the iPhone twitter app, the Android twitter app was used for 29%, twitter web clients were used for only 11% and Twitter lite and Tweetdeck combined were used for around 7%. Other sources were indicated for the remaining 15% tweets. This query took 11 ms to execute, which is quite reasonable given the structure and amount of data.

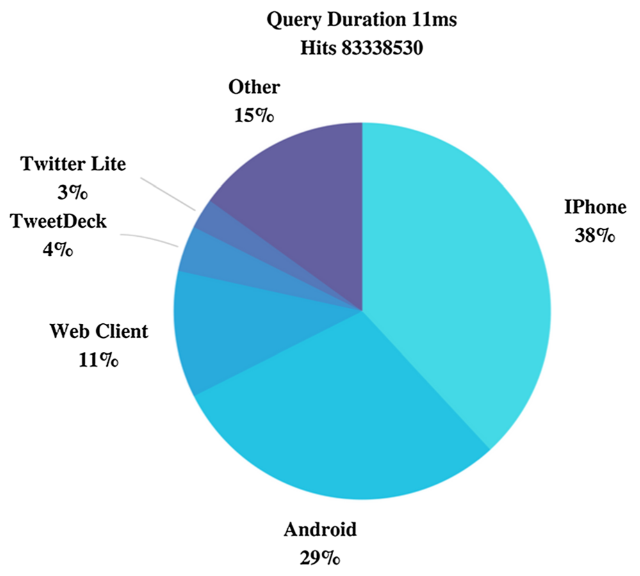
The above results demonstrate the efficiency of this data analysis system in that all three tasks (fetching the data, performing descriptive analysis and creating graphs), were accomplished in less than 15 s from a database size of 250+ million tweets. Clearly, this framework has proven



(a) Location



(b) Language



(c) Source

Fig. 3 Real-time analysis of Twitter data for the term "pizza"

suitable for the analysis of large text data in real-time without losing accuracy. It also shows that the restructuring and standardization procedures used on the data assisted in optimizing the accuracy of the results and efficiency of the processes in a context with limited resources.

3.3 Visualization dashboard

At present, the monitoring framework described in this paper is used to display data coming from Twitter stream. For example, in Fig. 4 we show a snapshot of the *Kibana* dashboard. The top-most plot is a pie chart of tweet source, which displays the results from which device they use to tweet, such as iPhone, web browser etc. The second top-most plot is pie chart of the languages used to tweet. In the middle, first histogram shows the the time and amount of twitter data flow. And, the second shows the word cloud and the bottom left shows the top ten users who are actively twitting. Similar dynamic dashboard creation is possible in minutes without knowledge of any programming knowledge and back-end system understanding.

3.4 Limitation

As Elasticsearch is designed to be used for real-time analysis, there are databases which provide functions that perform better in offline mass data analysis such as NoSQL databases (e.g., MongoDB) that support MapReduce [3]. Elasticsearch does not support MapReduce as it instead relies on the inverted index [17]. Additionally, Elasticsearch can be slow when new data is added to the index and it currently lacks support for more popular data formats (e.g., XML, CSV) and only supports JSON format which can be challenging for users unfamiliar with JSON [18].

4 Related work

Marcos [6] suggests that cloud computing is elastic in nature as the user can adjust it as per his/her data needs from processing power to storage. While it does seem ideal in theory, cloud computing comes with several challenges including both network inefficiency in data transport as well as issues related to data privacy and access control. Additionally, Hashem refers to 'data stabbing', which are problems associated with storing and analyzing the heterogenous and complex structure of big datasets [19]. As a solution, other authors such as Oleksii [3] support and highlight the benefits of Elasticsearch as a tool for real-time analysis in modern data mining repositories. In this research we attempted to address and resolve problems associated with data preprocessing and efficiency while also discussing the elastic cluster framework in more depth.

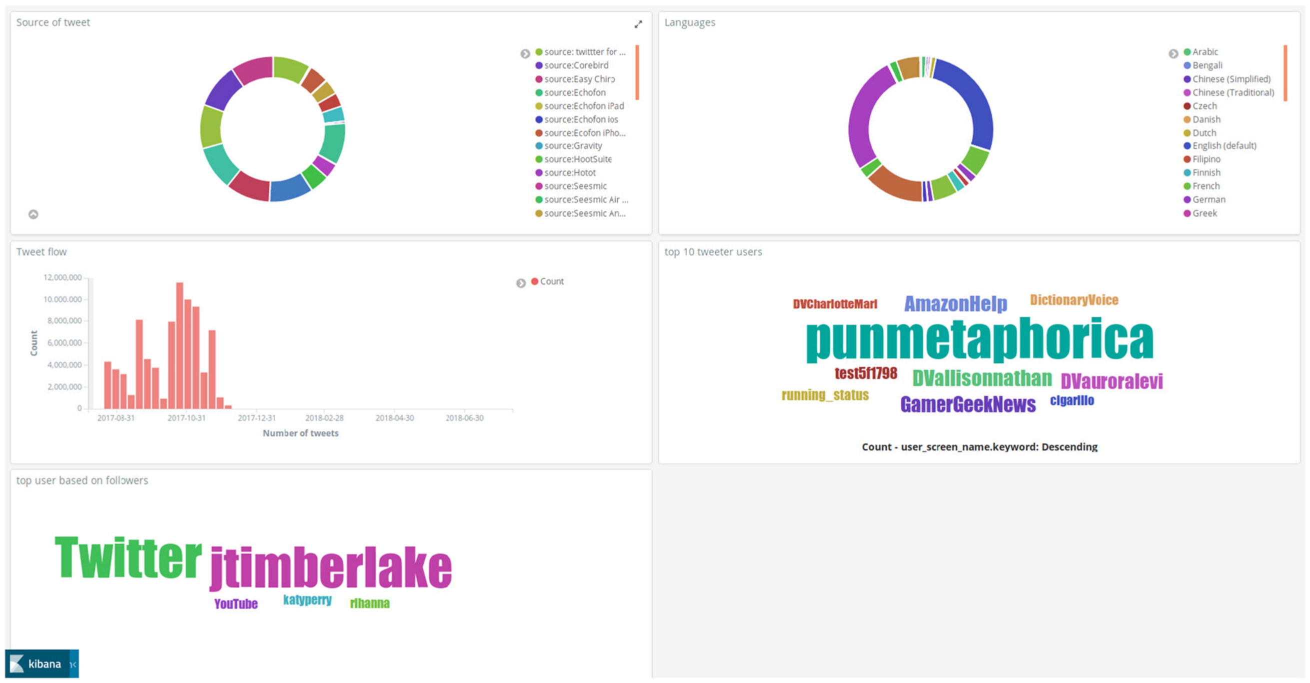


Fig. 4 Partial view of the Kibana dashboard for the twitter analysis

Currently there are very few research studies on frameworks for big data analysis in real-time although several discuss the application of practices in manufacturing [20] and gene coding [21]. Some researchers have used Elasticsearch cluster via a logstash plugin and MySQL databases for heterogenous accounting information system [22]. The data is monitored using MySQL server before inserting it into Elasticsearch. The researchers observed that there might be an issue of duplication of data and storage space, but the architecture ensures flexibility and modularity for the monitoring the system. They choose Elasticsearch as text search engine in real-time which allows them to search historical data. Mayo Clinic healthcare system developed a big data hybrid system using Hadoop and Elasticsearch technology. In healthcare, real-time result is essential for effective decision making. Before that, they used traditional RDBMS database to store and process data. But, it lacks integration between different platforms and inability to querying/ingest of healthcare data in a real-time or near real-time. In Mayo Clinic system Hadoop is used as a distributed file system and on top of it Elasticsearch works as a real-time text search engine. When there is a need for raw data Hadoop is used, and for real-time analysis Elasticsearch is used. Their experimentation showed very promising results, like searching 25.2 million HL7 records took just 0.21 s [23].

Designsafe web portal by Natural Hazards Engineering Research(NHER) analyze and share experimental data in real-time with researchers across the world. The user of their system sends the large amount of data which is stored in distributed NFS. During the preprocessing of the data, which includes analysis of string and basic cleaning, they index the data and make it compatible for Elasticsearch. This model allows users in a different location to query the same experimental data which is computed in different part of the world in real-time. All these present environments needs to be correctly configured as per the data and the requirements [24].

5 Conclusion

Elasticsearch provides a functional system to store, pre-index, search and query very large scale data in real-time. In particular, the capability of expanding the cluster size without stopping service as per user's requirement makes it suitable for this application. This research provides insights on how to standardize and configure the processes of Elasticsearch which result in increased analysis efficiency. To demonstrate the functionality and interactivity for users, the Kibana plugin was used as an interface. In conclusion, a proper configuration of Elasticsearch and Kibana makes real-time analysis of large scale data efficient and can help policy makers see the results instantaneously and in an accessible format that allows for decision making.

Acknowledgements This research is funded by the NSERC Discovery Grant; computing resources are provided by the High Performance Computing (HPC) Lab and Department of Computer Science at Lakehead University, Canada. Authors are grateful to Gaurav Sharma for initially setting up the data collection stream, Salimur Choudhury for providing insight on the data analysis and Andrew Heppner for reviewing and editing drafts.

References

1. Cervellini, P., Menezes, A. G., & Mago, V. K. (2016). Finding trendsetters on yelp dataset. In *2016 IEEE symposium series on computational intelligence (SSCI)* (pp. 1–7). IEEE.
2. Belyi, E., Giabbanelli, P. J., Patel, I., Balabhadrapathruni, N. H., Abdallah, A. B., Hameed, W., et al. (2016). Combining association rule mining and network analysis for pharmacosurveillance. *The Journal of Supercomputing*, *72*(5), 2014–2034.
3. Kononenko, O., Baysal, O., Holmes, R., & Godfrey, M. W. (2014). Mining modern repositories with Elasticsearch. In *Proceedings of the 11th working conference on mining software repositories* (pp. 328–331). ACM.
4. Liu, Q., Kumar, S., & Mago, V. (2017). Safernet: Safe transportation routing in the era of internet of vehicles and mobile crowd sensing. In *2017 14th IEEE annual consumer communications and networking conference (CCNC)* (pp. 299–304). IEEE.
5. Kim, M. G., & Koh, J. H. (2016). Recent research trends for geospatial information explored by twitter data. *Spatial Information Research*, *24*(2), 65–73.
6. Assunção, M. D., Calheiros, R. N., Bianchi, S., Netto, M. A., & Buyya, R. (2015). Big data computing and clouds: Trends and future directions. *Journal of Parallel and Distributed Computing*, *79*, 3–15.
7. Bsch, C., Hartel, P., Jonker, W., & Peter, A. (2014). A survey of provably secure searchable encryption. *ACM Computing Surveys*, *47*(2), 18:1–18:51. https://doi.org/10.1145/2636328.
8. Kumar, P., Kumar, P., Zaidi, N., & Rathore, V. S. (2018). Analysis and comparative exploration of elastic search, Mongodb and Hadoop big data processing. In *Soft computing: Theories and applications*, (pp. 605–615). New York: Springer.
9. Cea, D., Nin, J., Tous, R., Torres, J., & Ayguadé, E (2014). Towards the cloudification of the social networks analytics. In *Modeling decisions for artificial intelligence* (pp. 192–203). New York: Springer.
10. Bai, J. (2013). Feasibility analysis of big log data real time search based on hbase and elasticsearch. In *2013 ninth international conference on natural computation (ICNC)* (pp. 1166–1170). IEEE.
11. Elasticsearch-elastic.co. Retrieved April 30, 2018, from https://www.elastic.co/guide/en/elasticsearch/reference/6.2/index.html.
12. Gormley, C., & Tong, Z. (2015). *Elasticsearch: The definitive guide: A distributed real-time search and analytics engine*. Sebastopol: O'Reilly Media, Inc.
13. Your Window into the Elastic Stack. Retrieved 30, 2018, from https://www.elastic.co/products/kibana.
14. Python Elasticsearch Client. Retrieved April 30, 2018, from https://elasticsearch-py.readthedocs.io/en/master/.
15. Java Elasticsearch library-Elastic. Retrieved April 30, 2018, from https://www.elastic.co/guide/en/Elasticsearch/client/java-api/6.2/index.html.
16. Getting Started with Logstash. Retrieved April 30, 2018, from https://www.elastic.co/guide/en/logstash/current/getting-started-with-logstash.html.
17. Yang, F., Tschetter, E., Léauté, X., Ray, N., Merlino, G., & Ganguli, D. (2014). Druid: A real-time analytical data store. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data* (pp. 157–168). ACM.
18. Burkitt, K. J., Dowling, E. G., & Branon, T. R. (2014). System and method for real-time processing, storage, indexing, and delivery of segmented video. *US Patent 8,769,576*.
19. Hashem, I. A. T., Yaqoob, I., Anuar, N. B., Mokhtar, S., Gani, A., & Khan, S. U. (2015). The rise of big data on cloud computing: Review and open research issues. *Information Systems*, *47*, 98–115.
20. Yang, H., Park, M., Cho, M., Song, M., & Kim, S. (2014). A system architecture for manufacturing process analysis based on big data and process mining techniques. In *2014 IEEE international conference on big data* (pp. 1024–1029). IEEE.
21. Stelzer, G., Plaschkes, I., Oz-Levi, D., Alkelai, A., Olender, T., Zimmerman, S., et al. (2016). Varelect: The phenotype-based variation prioritizer of the genecards suite. *BMC Genomics*, *17*(2), 444.
22. Bagnasco, S., Berzano, D., Guarise, A., Lusso, S., Masera, M., & Vallero, S. (2015). Monitoring of IAAS and scientific applications on the cloud using the elasticsearch ecosystem. In *Journal of physics: Conference series* (Vol. 608, p. 012016). Bristol: IOP Publishing.
23. Chen, D., Chen, Y., Brownlow, B. N., Kanjamala, P. P., Arredondo, C. A. G., Radspinner, B. L., et al. (2017). 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*, *13*(2), 595–606.
24. Coronel, J. B., & Mock, S. (2017). Designsafe: Using elasticsearch to share and search data on a science web portal. In *Proceedings of the practice and experience in advanced research computing 2017 on sustainability, success and impact* (p. 25). ACM.

Neel Shah is a graduate student at Lakehead University, Canada Currently, he is working on analyzing social media data to gain insight of Canadian healthy behaviours. He is an active open source coder and maintains two open-source Python libraries. His core areas of interest are deep learning and data science.

Darryl Willick received the B.Sc. (1988) and M.Sc. (1990) degrees in Computational Science from the University of Saskatchewan, Canada. Throughout his career he has worked in the areas of High Performance Computing, Visualization, System administration, and Cyber Security. Currently he is a Technology Security Specialist/HPCC Analyst at Lakehead University, Canada.

Vijay Mago is also an Associate Professor in the Department of Computer Science at Lakehead University in Ontario, Canada where he teaches and conducts research in areas including big data analytics, machine learning, natural language processing, artificial intelligence, medical decision making and Bayesian intelligence. He received his Ph.D. in Computer Science from Panjab University, India in 2010. In 2011 he joined the Modelling of Complex Social Systems program at the IRMACS Centre of Simon Fraser University. He has served on the program committees of many international conferences and workshops. Recently in 2017, he joined Technical Investment Strategy Advisory Committee Meeting for Compute Ontario. He has published extensively (more than 50 peer reviewed articles) on new methodologies based on soft computing and artificial intelligent techniques to tackle complex systemic problems such as homelessness, obesity, and crime. He currently serves as an associate editor for IEEE Access and BMC Medical Informatics and Decision Making and as co-editor for the Journal of Intelligent Systems.