Computers, Materials & Continua CMC, vol.61, no.3, pp.1271-1283, 2019

SNES: Social-Network-Oriented Public Opinion Monitoring Platform Based on ElasticSearch

Chuiju You[1], Dongjie Zhu[2,*], Yundong Sun[2], Anshan Ye[3], Gangshan Wu[4], Ning Cao[1], Jinming Qiu[1] and Helen Min Zhou[5]

Abstract: With the rapid development of social network, public opinion monitoring based on social networks is becoming more and more important. Many platforms have achieved some success in public opinion monitoring. However, these platforms cannot perform well in scalability, fault tolerance, and real-time performance. In this paper, we propose a novel social-network-oriented public opinion monitoring platform based on ElasticSearch (SNES). Firstly, SNES integrates the module of distributed crawler cluster, which provides real-time social media data access. Secondly, SNES integrates ElasticSearch which can store and retrieve massive unstructured data in near real time. Finally, we design subscription module based on Apache Kafka to connect the modules of the platform together in the form of message push and consumption, improving message throughput and the ability of dynamic horizontal scaling. A great number of empirical experiments prove that the platform can adapt well to the social network with highly real-time data and has good performance in public opinion monitoring.

Keywords: Social network, public opinion monitoring, elasticsearch, scrapy-redis.

1 Introduction

With the rapid development of Internet technology, the number of netizens has increased rapidly. According to statistics, the number of Weibo users in China reached 337 million in the first half of 2018. The number of instant messaging users represented by WeChat reached 756 million, and the usage rate has reached 94.3% [Li, Wei and Xi (2018)]. Social networks have entered an era that emphasizes user engagement and experience, users can create contents, share contents and interact with each other on the web anytime, anywhere. So, there are massive amounts of user data on social networks. Containing great value and reflecting the public opinion orientation of the whole society, these massive social media data attracts more and more researchers to conduct research and

[1] College of Information Engineering, Sanming University, Sanming, 365004, China.

[2] School of Computer Science and Technology, Harbin Institute of Technology, Weihai, 264209, China.

[3] College of Mathematics and Computer Science, Xinyu University, Xinyu, 338004, China.

[4] School of Information Engineering, Jiangsu Polytechnic College of Agriculture and Forestry, Jurong, 212400, China.

[5] School of Engineering, Manukau Institute of Technology, Auckland, 2241, New Zealand.

[*] Corresponding Author: Dongjie Zhu. Email: zhudongjie@hit.edu.cn.

exploration on online public opinion and hot topics. Hindelang [Hindelang (1974)] found that the link between criminal justice and the public opinion is close. Burstein [Burstein (2003)] found that the impact of public opinion remains strong even the activities of political organizations and elites are considered. Firstly, data on social network is characterized by immediacy and rapid explosiveness, crawling data from social platforms in real-time is necessary for discovering public opinions and extracting hot topics timely and accurately. Secondly, UGC (User generated content) is massive and unstructured, storing and retrieving unstructured data quickly is critical to a social network public opinion monitoring platform. Lastly, a platform is not static, but can be expanded and adjusted continually with ever-expanding of data scale and performance requirements, therefore, the scalability of the platform and the low coupling between modules are indispensable. Researchers have made some progress in the design of online public opinion monitoring platform. The framework of the public opinion monitoring system of Weibo based on Spark proposed by Shen [Shen (2018)] achieved good results in predicting the detonation of the Weibo content source, however, the system does not adapt well to the real-time nature of social media data because of lacking data acquisition module and cannot retrieve target information fastly because of the absence of search engine. Tang et al. [Tang (2013); Jie and Jungang (2009)] added a web crawler module in their platform, which can crawl the latest data on social net-works in real time. However, the modules in their platforms performing functions directly by calling each other are highly coupled and do not have good scalability.

In this paper, we propose a novel platform called the "SNES", which is based on Elasticsearch (an open source distributed full-text retrieval system), and we design subscription module based on Apache Kafka (a high throughput distributed publish and subscribe message system) and Spark Streaming (a real-time streaming computing framework). A great number of empirical experiments prove that the platform can adapt well to social media data and obtains good performance in public opinion monitoring. The SNES has the following advantages:

(1) Crawling ever-changing social media data in real time.

(2) Storing and retrieving massive data quickly and efficiently.

(3) Excellent performance, scalability and fault tolerance.

2 Related work

Firstly, mining public opinion and extracting hot topics on social networks require massive and multi-source social media data, and the explosion and immediacy of social media data determine that data cannot be acquired and processed in the traditional way. Scrapy is an application framework for crawling web sites which can be used for a wide range of useful applications, like data mining, information processing or historical archival. Wang et al. [Wang and Guo (2012)] crawled the data of Taobao share-platform using Scrapy crawl architecture and analyzed the format of web pages. Scrapy-redis is a Scrapy-based distributed crawler component. It uses Redis to store and schedule requests, and store the items generated by the crawl for subsequent processing. Fan [Fan (2018)] designed and implemented a distributed crawling system based on scrapy-redis. Xie et al. [Xie and Xia (2014)] designed a topic-focused web crawler, which can crawl and gather

the subject-related web pages as soon as possible. After crawling the data, we need to process the data, such as filtering out illegal data and converting data. Spark Streaming can process real-time streaming data from multiple data sources including Kafka, Flume, Twitter, ZeroMQ, Kinesis and TCP sockets with high-throughput, fault-tolerant by using advanced algorithms such as map, reduce, join etc. And finally, the processing results can be stored in the file system, database, etc. Tan et al. [Tan and Zhou (2018)] designed a real-time traffic processing platform based on Spark Streaming and verified that the system can be applied to large-scale high concurrent data streams in real time. Zhang et al. [Zhang, Li, Liu et al. (2018)] proposed a more effective strategy to schedule tasks and added it to the source code of Spark. Yan et al. [Yan and Wang (2018)] designed a real-time movie recommendation system based on Spark streaming computing, which can better meet the real-time needs of users. Therefore, the SNES architecture proposed in this paper incorporates a distributed crawler cluster module, which enables the platform to crawl ever-changing social media data in real time.

Secondly, mining social network public opinion and extracting hot topics require efficient storage and retrieval of massive UGC which are unstructured data. Elasticsearch is an open source, distributed, full-text search engine that supports RESTful interfaces. It can store, search and analyze large amounts of data in a very short time and it usually be used as a core engine with a complex search scheme. We can set up the same cluster name to form a distributed cluster in the same network segment and balance the load by adding more nodes to the cluster for horizontal expansion. ES supports full-text search, each field can be indexed, and the data of each field can be searched. ES adopts a non-centralized cluster design, after the central node in the cluster fails, ES will automatically select a new node as the central node and automatically migrate the data fragments to ensure the security and access of user data. Its many advantages make it a good performance in solving big data related problems and a preferred tool in enterprise big data solutions. Zhou et al. [Zhou and Han (2015)] designed and implemented elastic search cluster based on ES for an e-commerce system, which significantly reduced CPU usage, memory usage, maintenance costs of database server and greatly improved search efficiency and system stability. Li et al. [Li, Li, Zhang et al. (2018)] designed a high-performance chemical structure & data search engine called DCAIKU, built on CouchDB and ElasticSearch engines, which can handle both keyword search and structural search for millions of records with both high accuracy and low latency. Therefore, the SNES architecture proposed in this paper incorporates the open source high-distribution distributed full-text search engine, which can store and retrieve massive unstructured data in near real time.

Finally, a platform should have good scalability, fault tolerance and decoupling between modules. Apache Kafka is a distributed flow processing platform, which can publish, subscribe and persist data with high timeliness and fault tolerance, so it can be used as message middleware. Kafka has the following advantages over other mainstream messaging middleware including ActiveMQ and RabbitMQ:

(1) Simple and intuitive operation.

(2) Excellent message throughput.

(3) Supporting load balancing and dynamic horizontal scaling with a fully distributed

architecture.

These advantages of Kafka are necessary to build a highly available and efficient messaging subscription service. Lu [Lu (2018)] designed and implemented a Kafka-based messaging subscription service called MPS, extending the way of communication between micro-services and providing asynchronous decoupling between services by solving many problems, such as highly coupled HTTP and RPC calls between services within the platform, easily blocked request, unbuffered request. Yan et al. [Yan and Yu (2018)] used Kafka as a message platform to complete the transmission of massive data and concurrent real-time processing. Therefore, we design a subscription module based on Apache Kafka which connects the modules of the platform together in the form of pushing and consuming message to increase message throughput and the ability of dynamic horizontal scaling.

3 The design of SNES

3.1 The overall design

In terms of multi-source data acquisition, SNES improves the speed of data crawling and the schedulability of crawling tasks by combining distributed crawling frameworks with subscription module based on Kafka message subscription service; in terms of massive data storage and retrieval, we design and add ES-centric full-text storage and search engines to optimize the storage and retrieval efficiency of the entire system platform; in the way of combining different modules, the Kafka-based message subscription service is designed, tasks and instructions are transmitted between modules in the form of messages, which greatly reduces the coupling between modules and improves the scalability of the platform.

The overall topology of the architecture is shown in Fig. 1,

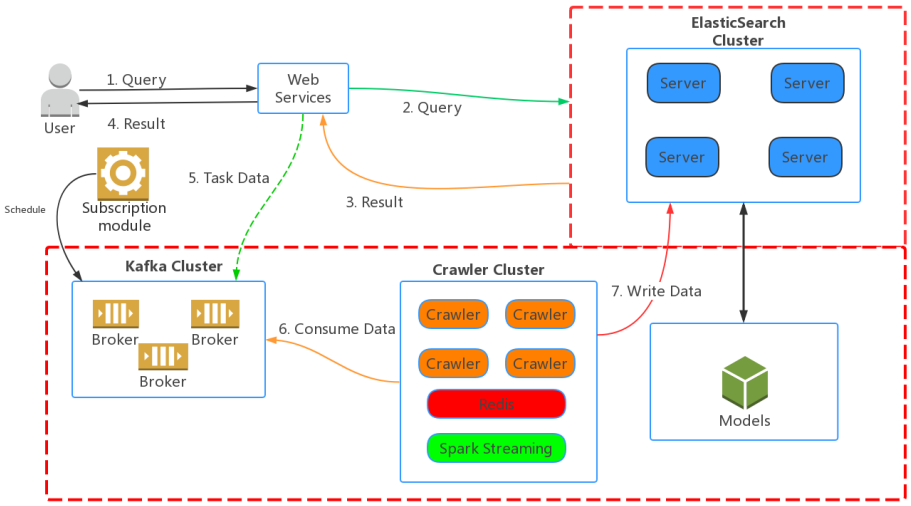

Figure 1: The overall topology of SNES

Considering the requirements of data size, performance and function for public opinion

and hot topics mining on social network, we add the data storage and retrieval module, the message publishing subscription service module, the webservice logic processing module and the multi-source distributed information and stream processing module into SNES architecture. The work steps can be summarized as follows:

Step 1: Users search for specified content through web services;

Step 2: If there are hit information in ES, go to Step 7, else go to Step 3;

Step 3: The service issues a message crawling command through the message subscription

service module;

Step 4: The data crawling module receives the relevant command and crawl the specified data;

Step 5: The stream processing module processes the crawled data;

Step 6: ES stores the processed data;

Step 7: Return the hit information to web services.

The network topology of the entire platform is shown in Fig. 2. There are four clusters, including multi-source data crawling cluster, storage and search engine, Kafka-based messaging service and Spark-based data processing cluster. Each module is connected through an intranet switch, which reduces the coupling and increases the rate of communication. The above parts are in the internal LAN environment, and in order to improve security, the data communication with the external network must pass through the firewall. In order to avoid the limit of IP access frequency from the target website, we use proxy IP services to climb domestic social networking sites such as Sina Weibo and Wechat, and we adopt virtual cloud hosting services to climb foreign social networking sites such as Twitter and FaceBook. Fig. 3 depicts the internal structure of the system. It consists of four parts: the message publishing subscription service, the social network information crawling service, data storage and retrieval clusters and public opinion information processing models.

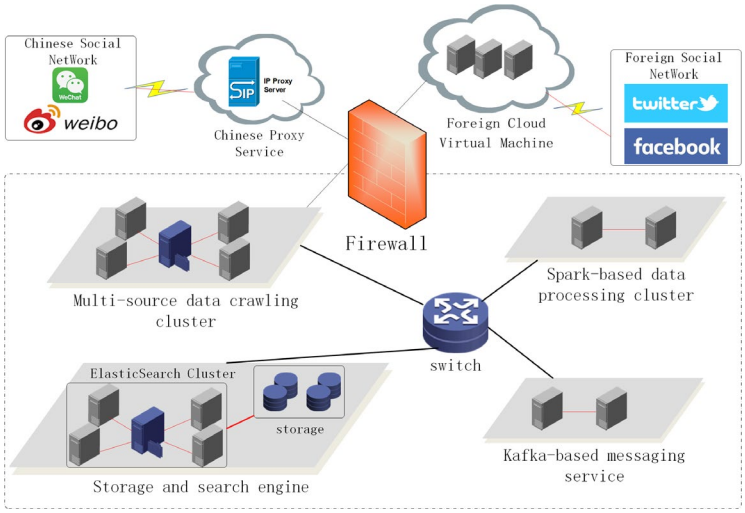

Figure 2: The network topology of SNES

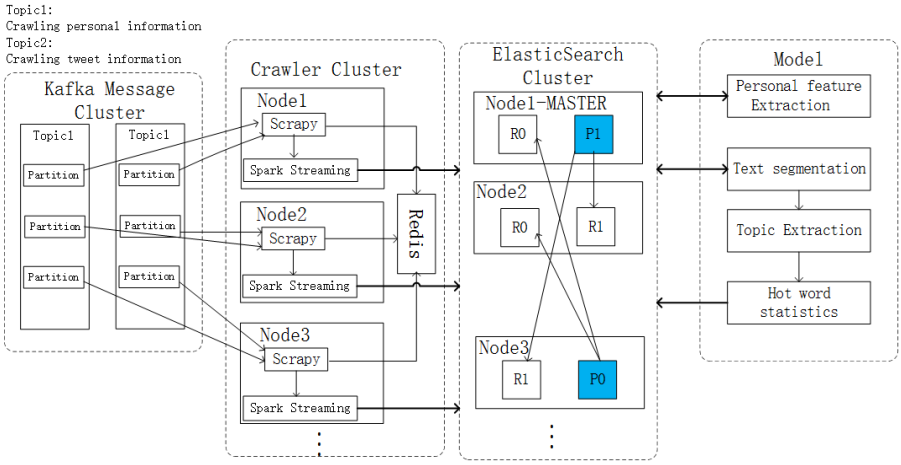

Figure 3: Internal structure diagram of SNES

3.2 Data crawling module

The crawler cluster consists of several nodes, each of which acts as a consumer of message subscription module (Chapter 3.3) and each node contains the crawler framework Scrapy-redis, Spark Streaming framework and Redis database. The frame structure and internal data flow of the data crawling module is shown in Fig. 4. To improve data crawl speed and avoid interruption due to network problems or the restrictions of target websites, we design the top scheduler to communicate with the message subscription module and the brief work flows are as follows:

Step 1: The top scheduler requests a new data crawling task from message subscription module;

Step 2: If the task is not null, go to Step 1, else go to Step 3;

Step 3: The top scheduler issues a data crawling command to spider;

Step 4: The spider gets the first URL from redis database and send to engine;

Step 5: If the URL is already in the crawled queue, go to Step 1, else go to Step 6;

Step 6: The engine requests random cookie from cookie pool and random the cookie and proxy Ip;

Step 7: The spark streaming module processes the obtained data and stores it to massive data storage and extraction module (Chapter 3.4);

Step 8: Put the URL into crawled queue and go to Step 1.

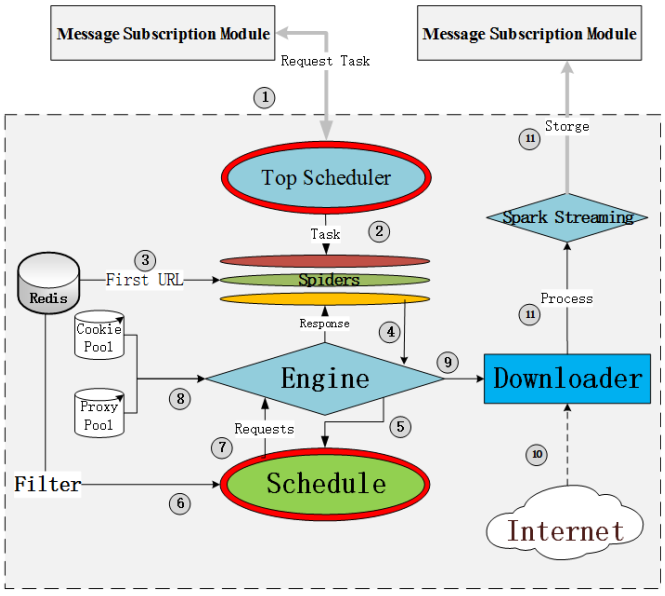

Figure 4: Frame structure and internal data flow of data crawling module

3.3 Message subscription module

To improve the scalability, fault tolerance and decoupling between modules in our platform, we design the message subscription module based on Kafka. The message subscription cluster consists of two working nodes. Each message topic in the figure is divided into three partitions, each consumer node consumes one partition of each topic. After empirical experiment, it is concluded that to get better performance, the number of topic's partition is preferably set to a multiple of the consumer and it is best to ensure that the number of message partitions that each consumer monitors is equal. Users can subscribe to specific events or topics as they want and the data crawling module in Chapter 3.1 would consume the message and crawl the corresponding data.

For each topic, the message subscription maintains a partitioned message that looks like the left of Fig. 5. Each partition is an ordered, immutable sequence of records that is continually appended to a structured commit message. Each record in the partitions is assigned a sequential ID number called the offset that uniquely identifies each record within the partition. In fact, the only metadata retained on a per-consumer basis is the offset or position of that consumer in the message. This offset is controlled by the consumer: normally a consumer will advance its offset linearly as it reads records, but, in fact, since the position is controlled by the consumer it can consume records in any order it likes. As shown in the right of Fig. 5, a consumer can reset to an older offset to reprocess data from the past or skip ahead to the most recent record and start consuming from now.

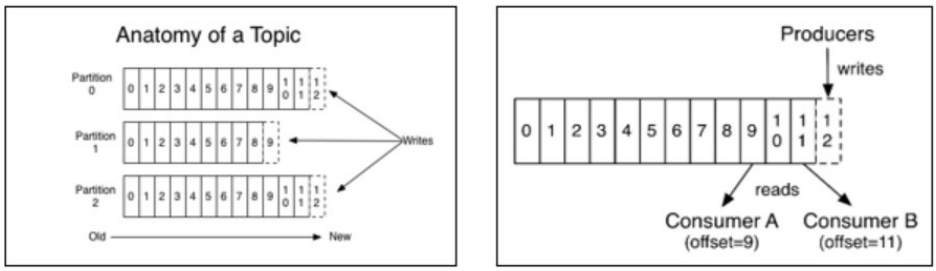

Figure 5: The theory of producing and consuming topics

3.4 Massive data storage and extraction module

Mining social network public opinion and extracting hot topics require efficient storage and retrieval of massive UGC which are unstructured data. We design a massive data storage and extraction module based on Elastic Search. The data structure of the massive data storage and extraction module is shown in Fig. 6. Each Node has three shards, where P is the primary shards and R is the redundant shards. The primary shard P1, P2, and the redundant shard R0 are stored in node 1, the redundant shard R0, R1 and R2 are stored in node 2, the primary shard P0 and the redundant shard R1, R2 are stored in node 3 There are a total of 3 primary shards and 6 redundant shards. At the same time, we also noticed that Node 1 also has a master ID, which means it is a master node. It is more special than other nodes. It has the authority to control the entire cluster, such as resource allocation, node modification.

Figure 6: The theory of producing and consuming topics

4 Experiment

We evaluate the effects of public opinion monitoring and performance of the SNES by applying it to Sina Weibo real-time data.

4.1 Experiment environment

To ensure the data acquisition, processing and retrieval capability of the plat- form, and consider the scalability and load balance of the system, we constructed the experimental platform environment as shown in Tab. 1. The network bandwidth test of the platform is shown in Fig. 7.

Table 1: The hardware facility and application of experiment platform

Server	CPU	Memory	Disk	application
Server-1	3.60 GHz 2 core 4 threads	8 GB	500 GB	ES-Master
Server-2	3.60 GHz 2 core 4 threads	8 GB	500 GB	ES-Data
Server-3	3.60 GHz 2 core 4 threads	8 GB	500 GB	ES-Data
Server-4	3.60 GHz 2 core 4 threads	8 GB	500 GB	ES-Data
Server-5	3.60 GHz 2 core 4 threads	8 GB	250 GB	Kafka-Broker
Server-6	3.60 GHz 2 core 4 threads	8 GB	500 GB	Kafka-Broker
Server-7	3.60 GHz 2 core 4 threads	8 GB	250 GB	Kafka-Broker
Server-8	3.60 GHz 2 core 4 threads	8 GB	500 GB	Spider&Spark
Server-9	3.60 GHz 2 core 4 threads	8 GB	250 GB	Spider&Spark
Server-10	3.60 GHz 2 core 4 threads	8 GB	500 GB	Spider&Spark

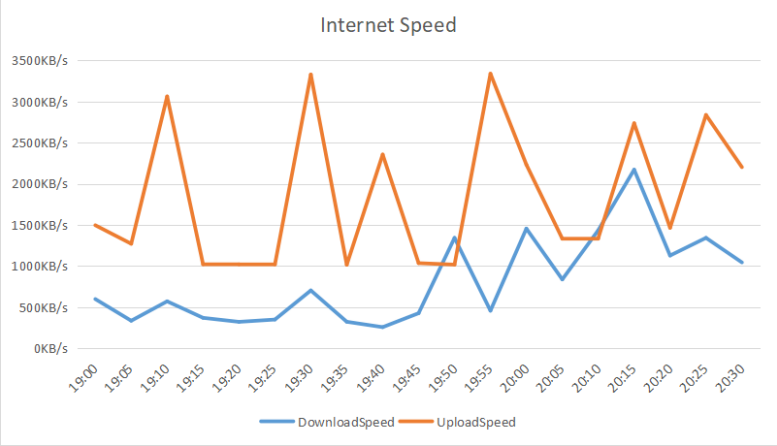

Figure 7: Network bandwidth of the experimental environment

4.2 Experiment data

The experimental data in this paper is collected from Sina Weibo data in real time, including personal information, relationship network, tweets and interactive information (like or dislike, comment), etc. Detailed data information for incident moni-toring and data crawling test is shown in Tab. 2; the data format for searching and writing latency test is shown in Tab. 3.

Table 2: Statistics of the experimental data for incident monitoring

Data Source	Persons	Relations	Tweets	Interacts
Sina Weibo	130,028	71,032	1,501,202	4,130,187

Table 3: Data format for searching and writing

Field	Type	Length
userId	Integer	32bit
userName	Char (20)	160bit
content	Char (150)	1200bit
date	Long	64bit
platform	Short	16bit
thumb	Integer	32bit
comment	Integer	32bit
forward	Integer	32bit

4.3 Experiment result

4.3.1 Performance evaluation

In this paper, the search efficiency, the speed of crawling data and writing efficiency are used as evaluation indicators for this platform. The speed of crawling data. We use the Eq. (1) to measure the speed of crawling data.

$$sp=\frac{\mathrm{c}}{n\times \mathrm{h}\times tr\times \mathrm{t}\times bw} \tag{1}$$

where n is the number of crawling cluster nodes; h is the average CPU frequency, the unit is GHz; t is the total crawling time, the unit is second(s); c is the total number of pages crawled during t seconds; bw represents the average of network bandwidth during the time, the unit is MB/S. Therefore, sp represents the average value of the page crawled by the single-threaded single node during the CPU clock cycle under the unit bandwidth condition. The value of sp can reduce the influences of the number of nodes, CPU threads, CPU frequency, and network bandwidth to some extent. In the experiment conducted in this paper, the crawling time is 48 hours, the total number of pages crawled is 864078, h=3.6, the bw=2 and the result sp=0.07, which is calculated by Eq. (1).

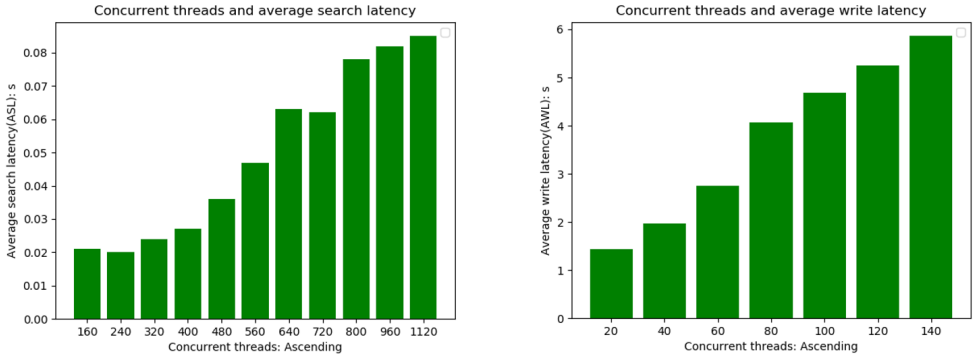

Figure 8: Search and write latency of SNES

We evaluate the system's search latency and index latency with concurrent read and write scripts, with a test time of 50 minutes. We measure the delay by the Eq. (2)

$$ASL = (\sum_{i=1}^{n} t_i) / n \tag{2}$$

where n is the number of experiments, t_i is the delay of the i^{th} experiment, the unit is second; ASL is average search latency. The left of Fig. 8 shows the result of our concurrent searching test, we conduct 4000 experiments with each specific number of concurrent requests and get the search latency. The right of Fig. 8 shows the result of our concurrent writing test, we conduct 2000 experiments with each specific number of concurrent requests and get the writing latency. All data is randomly generated and the format is shown in Tab. 3.

4.3.2 Incident monitoring

This experiment is aimed at "The Event of HuaZhu Information Leakage on 2018-08-28 in China", the search keyword is "HuaZhu Leak" and the monitoring time is from August 28, 2018 to September 10, 2018. The left of Fig. 9 shows the trend of the transmission volume monitored by the system about the "The Event of HuaZhu Information Leakage". It can be seen that the line is rising from August 28 to September 2, representing that the event is heating up. After September 04, it is gradually declining until it is at a lower level. This is consistent with the development of the entire event. The right of Fig. 9 shows the hot word cloud of the event obtained by extracting and filtering the content topic from the crawled microblog. It can be seen that words such as "HuaZhu", "Information", "Leakage", "Data" and "Cybersecurity" have the highest proportion in the word cloud, and these words can properly describe and summarize the content of this event and the discussions and opinions of Weibo users and related media on this incident.

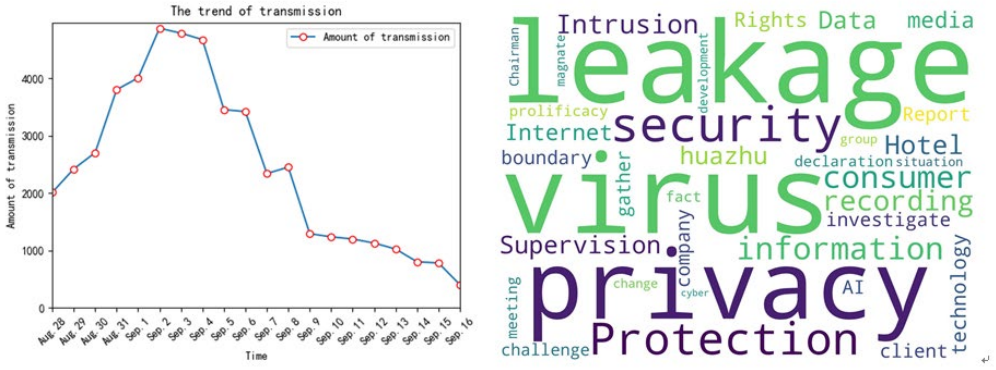

Figure 9: Daily spread and hot words cloud of "Huazhu Info Leakage"

5 Conclusion and discussion

SNES we proposed in this paper creatively integrates high-throughput subscription module based on Kafka, distributed crawling framework, real-time streaming computing framework and massive data storage and extraction module based on ElasticSearch. After the above experiment, the SNES architecture has the following advantages:

(1) Crawling ever-changing social media data in real time.

(2) Storing and retrieving massive data quickly and efficiently.

(3) Good scalability and fault tolerance.

Currently, our platform has defects in data security storage and encryption protection, our future work will focus on improving the security of the platform.

Acknowledgement: This work is supported by State Grid Science and Technology Project under Grant Nos. 520613180002, 62061318C002, the Fundamental Research Funds for the Central Universities (Grant Nos. HIT.NSRIF.201714),Weihai Science and Technology Development Program (2016DXGJMS15) and Key Research and Development Program in Shandong Provincial (2017GGX90103), Fujian Young and Middle-aged Teacher Education Research Project, Grant No. JAT160466, Jiangsu Polytechnic College of Agriculture and Forestry Key R&D Projects (2018kj11), Study and Development of Smart Agriculture Control System Based on Spark Big Data Decision (2017N0029).

References

Burstein, P. (2003): The impact of public opinion on public policy: a review and an agenda. *Political Research Quarterly*, vol. 56, no. 1, pp. 29-40.

Fan, Y. (2018): Design and implementation of distributed crawler system based on scrapy. *IOP Conference Series: Earth and Environmental Science*, vol. 108.

Hindelang, M. J. (1974): Public opinion regarding crime, criminal justice, and related topics. *Journal of Research in Crime and Delinquency*, vol. 11, no. 2, pp. 101-116.

Jie, D.; Jungang, X. (2009): IPOMS: an internet public opinion monitoring system. *Applications of Digital Information and Web Technologies*, pp. 433-437.

Li, J.; Wei, Q.; Xi, Y. (2018): Analysis and research on the development of university libraries based on wechat and facebook social network platform. *Journal of Sichuan Library Science*, no. 6, pp. 69-74.

Li, R.; Li, B.; Zhang, G.; Jiang, J.; Luo, Y. (2018): A high-performance and flexible chemical structure&data search engine built on couchdb&elasticsearch. *Chinese Journal of Chemical Physics*, vol. 31, no. 3, pp. 341-349.

Lu, S. (2018): *Design and Implementation of a Message Publishing Service Based on Kafka (Ph.D. Thesis)*. Nanjing University.

Shen, L. (2018): *Design and Implementation of Weibo Public Opinion Monitoring System Based on Spark (Ph.D. Thesis)*. University of Electronic Science and Technology.

Tan, L.; Zhou, J. (2018): Real-time traffic data processing platform based on spark streaming. *Computer System Application*, vol. 27, no. 10, pp. 133-139.

Tang, Y. (2013): *Design and Implementation of Internet Public Opinion Monitoring System (Ph.D. Thesis)*. Beijing University of Posts and Telecommunications.

Wang, J.; Guo, Y. (2012): Scrapy-based crawling and user-behavior characteristics analysis on taobao. *Cyber-Enabled Distributed Computing and Knowledge Discovery*, pp. 44-52.

Xie, D. X.; Xia, W. F. (2014): Design and implementation of the topic-focused crawler based on scrapy. *Advanced Materials Research*, vol. 850, pp. 487-490.

Yan, H.; Yu, X. (2018): Software system design based on Kafka message platform. *Electronic Technology and Software Engineering*, no. 18, pp. 38.

Yan, L.; Wang, X. (2018): Real-time movie recommendation research based on spark streaming computing. *Software Guide*, pp. 1-5.

Zhang, X.; Li, Z.; Liu, G.; Xu, J.; Xie, T. et al. (2018): A spark scheduling strategy for heterogeneous cluster. *Computers, Materials & Continua*, vol. 55, no. 3, pp. 405-441.

Zhou, Y.; Han, X. (2015): Application examples of elasticsearch in e-commerce system. *Information Technology and Standardization*, vol. 5, pp. 30.