EasyChair Preprint

№ 10472

A Highly Accurate Data Synchronization and Full-Text Search Algorithm for Canal and Elasticsearch

Peiyang Wei, Xiaoyu Shi and Gang Zhang

June 30, 2023

A Highly Accurate Data Synchronization and Full-text Search Algorithm for Canal and Elasticsearch

Peiyang Wei
School of Computer Science and Technology
Chongqing University of Posts and Telecommunications
School of Software Engineering,
Chengdu University of Information Technology Chongqing
Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences
Chongqing School, University of Chinese Academy of Sciences
Chongqing, China
E-mail: weipy@cuit.edu.cn

Xiaoyu Shi
Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences
Chongqing School, University of Chinese Academy of Sciences
Chongqing, China
E-mail: xiaoyushi@cigit.ac.cn

Gang Zhang
Center for Fusion Science,
SouthwesternInstitute of Physics,
CNNC, Chengdu, Sichuan
E-mail: zgang@swip.ac.cn

***Abstract*—Currently, there are numerous thorny issues in structured data and semi-structured full-text search scheme with large-scale text nature, like long data synchronization delay, inconvenient personalized business processing and low efficiency. To address these issues, this paper proposes an efficient algorithm based on Canal data synchronization framework and Elasticsearch full-text search engine. Firstly, we rewrite the Canal adapter component to obtain the flexible configuration of business data processing, thereby enhancing the secondary data processing ability of the framework and achieving the purpose of improving the efficiency of data synchronization. Secondly, by recording the synchronization time of nearby data in Canal framework, the weight of time series data is gradually decreased by combining with the exponential weighted average function to highlight the influence of recent data and present the novelty of data, which can achieve effective control the synchronization interval and duration by dynamically and flexibly setting the synchronization trigger period. Lastly, the Elasticsearch word tokenizer is modified, and then the configuration of custom expansion words and stop words dictionary are proposed to filter the query data effectively, thereby enhancing the query hit rate and accuracy. Extensive experiments on the data of traditional Chinese medicine demonstrate that the designed algorithm obtains high data synchronization efficiency, full text search speed and accuracy. Hence, the proposed algorithm is a milestone in smart healthcare.**

***Keywords—Elasticsearch, Canal, Real-Time Synchronization, Full Text Search*.**

I. INTRODUCTION

With the rapid development of cloud computing and Internet technology, a large amount of structured, semi-structured and unstructured data are produced in our daily life. Correspondingly, Synchronization and retrieval of big data on edge devices, storage devices, mobile devices and various software have always been the focus of research in the field. For these different directions, extensive scholars have made useful exploration and practice.

This research is supported in part by National Key Research and Development Program of China under Grant 2022YFB3305100 and Youth Innovation Program of CUIT under Grant KYQN202222(Corresponding author: P. Wei).

Zhang *et al.* [1] propose a series of efficient methods to perform algorithm/accelerator co-design and co-search for optimized edge AI search, which demonstrates the proposed methods on popular edge AI applications (object detection and image classification), thereby achieving a significant improvement than previous designed schemes. Farbeh *et al.* [2] make a method called Alternating Cache Allocation to Conduct Higher Endurance (A-CACHE), which improves the lifetime of frequently-written D-cache by exploiting rarely-written I-cache. Han *et al.* [3] present a fault tolerant cache system of automotive vision processors. The cache system has the small redundant memory, which decreases the transient error rates with the proposed mechanism, thereby increasing the error recovery rate. The above studies mainly focus on data synchronization, cache and fault tolerance of hardware devices, which provide a solid foundation for data search and high-quality applications.

Amato *et al.* [5] add the transform CNN features into textual representations and index them with the well-known full-text retrieval engine Elasticsearch. Bajer *et al.* [6] describe the process of using Elastisearch, Logstash and Kibana (ELK) tool to process IoT data, which expands the application scope of ELK tool. Shah *et al.* [7] present a solution to effectively address the challenges of real-time analysis using a configurable Elasticsearch search engine. Dhulavvagol *et al.* [8] enhance the performance of distributed processing systems by applying effective shard partitioning, efficient shard selection techniques and perform the comparative study analysis of shard selection techniques on Elastisearch considering precision, MAP, cost measures. Voit *et al.* [9] present a system based on Elasticsearch engine, MapReduce model for the solution of full-text search and data visualization. Devins *et al.* [10] enhance the seamless replicability and interoperability between Elasticsearch and the Pyserini IR toolkit by competitive bag-of-words first-stage retrieval baselines for the MS MARCO document ranking task, both of them are based on the open-source Lucene search library. Bhatnagar *et al.* [11] describe the working of open source tools like ELK, which have been clubbed together to complete insight and visualization of data. By implementing these tools, they are performing sentiment analysis of data taken from social networking blogging service like twitter. Han *et al.* [12] use the Flask framework in python to write a

system for rapid retrieval and visualization of media data by using MongoDB and Elasticsearch. Meanwhile, Kibana can be used to visually display and present the data. Zmaranda *et al.* [13] perform a comparative evaluation of two popular open-source DBMSs: MySQL Document Store and Elasticsearch as non-relational DBMSs, and this comparison is based on a detailed analysis of CRUD operations for different amounts of data showing how the databases could be modeled and used in an application. Kim *et al.* [14] built a topic classification index search technology using Elasticsearch and LDA model. Ahmed *et al.* [15] design a set of log monitoring system based on ELK framework, the system can process any source, any format of data, and provide real-time analysis, search and monitoring. Zhou *et al.* [16] propose a cache optimization model based on HBase and Redis for image storage. Liu *et al.* [17] propose a high-performance memory key-value pair database Redis++. Zamfir *et al.* [18] and Kim *et al.* [21] use the Elasticsearch framework to achieve optimizing its performance and querying efficiency, analyzing log data and tracking system failures, thereby improving its robustness. Ghosh *et al.* [19] propose a caching mechanism to reduce response time for the performance loss in serverless architectures. Li *et al.* [20] employ Spark and Elasticsearch to build a log mining framework based on cloud architecture to solve the problem of limited performance of large amount of data query analysis.

According to the above literatures, there are few works on the configurability of data synchronization, the secondary transformation of business and the optimization of index in the research on various improvements of data synchronization and full-text search. Therefore, this paper proposes an optimized and improved search algorithm with high real-time, high concurrency based on Canal for data synchronization and Elasticsearch for inverted index construction. Finally, the algorithm is deployed in the Multi-dimensional Prescription Training Platform (MPTP), and then the effectiveness of the algorithm is proved by the test of high concurrency, multi-user and large amount of data.

For the rest of our paper, Section II discusses the related work. Framework and model are introduced in Section III. Section IV presents the application and test. Lastly, Section V concludes our paper.

II. Related Work

A. Elasticsearch

Elasticsearch is an open source full-text search and analysis engine that can store, retrieve, analyze large amounts of text data in near real time. Moreover, Elasticsearch is implemented by using Lucene as its core, which can easily scale to hundreds of servers and handle petabytes of data. Most importantly, it also has the following characteristics:

a) Convenient configuration and use. Users can create an index, and then they perform the full-text search, analysis, sorting and presentation of documents. Structured and partially semi-structured data types are supported for the distributed search and aggregate queries;

b) Shard storage and automatic backup. The index is the basic unit of data storage in the framework when the amount of data in the index is large, then the data can be split horizontally to form shards. Meanwhile, the data can be automatically backed up to form a copy, which could ensure that the data may not be lost after a server is closed, thereby ensuring the high availability of the service;

c) Real-time monitoring and rapid response. It is convenient to obtain data changes in a specified period of time, which helps users find data anomalies and issues.

Specifically, Elasticsearch also has issues like high requirements for hardware resources, long time for indexing and searching, which is the critical points for this paper.

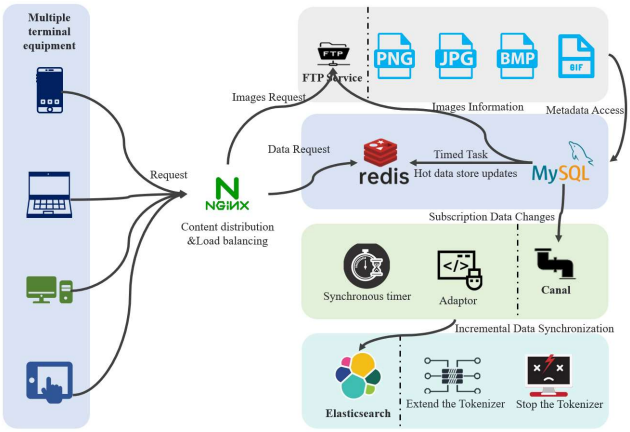

Fig. 1. Diagram of a training platform for formulations.

B. Canal

In general, Canal is an open source database management system with the superior performance and security, which is widely utilized in large enterprises and high performance computing environments. Due to Canal adopting the incremental log parsing technology, it can synchronize changes of different data in near real time and pass them to subsequent business processing. Meanwhile, Canal supports multiple deployment modes like single point, cluster, embedded and cloud, which can also choose the appropriate way to ensure real-time performance and reliability by corresponding requirements. Consequently, they can satisfy the requirements of most real-time data synchronization and data distribution scenarios.

III. Framework and Model

Fig. 1 shows the block diagram of the formula science training platform based on real-time synchronization and full-text search, which mainly consists of request distribution, data storage, data synchronization, full text search.

A. Module Explanation

1) Request Dispatching

The request distribution module is centered on the Nginx network server, which is mainly set up for the system with large capacity and multi-user concurrent online use. Nginx can handle different demand types like images, HTTP requests, full-text retrieval and other responses to improve system loading speed. Meanwhile, in the face of high concurrency, load balancing and dynamic content distribution, services can be enabled to share the request load for improving the availability and performance of the server.

2) Data Storage

The data storage module consists of three types of storage services: MySQL database for relational data persistence, FTP file service for image storage and Redis for caching data. Among them, MySQL is mainly used to store the structured

data information of the platform system and provide basic data services for the management, application layer of the system. FTP file service mainly aims at the specific scene of the application layer of the platform to provide image reading, loading, storage and other services. In addition, Redis can improve system access performance through the regular task hot update mechanism. Furthermore, the frequently accessed data is stored in Redis to improve the hit rate of request access, thereby achieving the purpose of effectively optimizing access speed.

3) Data Synchronism

The data synchronization module is mainly implemented based on the open source Canal framework. By parsing the MySQL binlog of the subscribed storage system's main data, the database change operations are parsed into SQL statements in time, and then these SQL statements are sent to the target Elasticsearch for achieving data synchronization.

The main steps of Canal to achieve data synchronization are as follows:

a) Connecting to the MySQL database of the system's data storage, it obtains the file name and location of the binlog to implement the subscription function of the change log;

b) Parse each change in the MySQL database by using the data in the binlog file. Canal uses a Canal Server with a separate service process that connects to a MySQL database at one end and listens for binlog changes. When the database changes, the Canal Server parses the change operation into SQL statements and sends them to the Canal Client;

c) When the Canal Client receives the SQL statement, which sends it to Elasticsearch, thereby allowing for fast indexing and data synchronization.

In the Canal application of data synchronization, the specific business requirements of the prescription science's training platform, like simplified and traditional Chinese character processing are optimized and improved with a purpose to further enhance the efficiency and accuracy of synchronization. Moreover, The details of the algorithm is presented in Algorithm I.

4) Full-text Search

The full-text search module uses Elasticsearch for responding to all kinds of query requests from users by the system in a timely manner, reaching a near-real-time state and realizing the full retrieval of business data. However, in terms of the adaptability of traditional Chinese medicine (TCM) prescription business, its performance still has room for optimization. To address this hot issue, this paper conducts an in-depth study on the underlying principle of Elasticsearch. By modifying the Elasticsearch index, word segmenter, setting the black and white list mechanism, this paper formulates a custom expansion word and stop word dictionary to achieve further filtering requirements, thereby improving data query performance. Furthermore, the details of algorithm optimization is given in Algorithm II.

B. Executing Process Steps

With the platform block diagram in Fig. 1, this paper shows the execution process based on the data retrieval flow.

step1: Multi-user requests from different types of terminals like PC and mobile terminals are forwarded by the Nginx network server.

step2: Forward to the corresponding object by different request types, Redis is accessed first for business data. If it fails to hit, MySQL database is accessed again. Meanwhile, the access information is stored in Redis by the round robin update mechanism, thereby enhancing the efficiency of subsequent access. For advanced retrieval requests, we send them to the Elasticsearch server.

step3: Each type of request response server obtains relevant resource data and feedback results. Among them, pictures are loaded by the FTP file server. Hot data of business information is obtained from Redis, then ordinary data is responded by MySQL. Moreover, the high-level retrieval requirements are responded by Elasticsearch.

step4: If Canal detects a change in the MySQL binlog, which automatically initiates a synchronization operation to keep the business data consistent with the data in the Elasticsearch index library.

step5: The Elasticsearch server displays the data information filtered by the algorithm in the index library.

C. Design of Canal Synchronization Algorithm

There are some issues in TCM, like different forms of characters, conversion between simplified and traditional Chinese, meaningless modal particles in traditional Chinese, scientific calculation of part of business data. Meanwhile, there are small batch and multi-frequency data update, which harms the synchronization efficiency. To address the above issues, this paper optimizes and improves the open source Canal framework, which makes improvements in monitoring synchronization interval and replacing adapter components. In addition, the specific optimization block diagram of Canal synchronization algorithm is shown in Fig. 2, which sets the current data obtained from MySQL as G_{data}, the timestamp of the start of synchronization is defined as T_s, the synchronization data is given as S_{data}, then the adapter $IS_{abaptor}$ is enabled.

1) Monitor synchronization interval

Canal obtains the data G_{data} by subscribes from the data synchronization event triggered through MySQL, while the timer records the timestamp T_s of the batch data processing starting for this batch.

Through the exponential weighted average function of formula (1), the effect of removing noise and smoothing data can be achieved, then the trend of original data can be easily described. Moreover, combined with the upcoming data synchronization cycle, the time period of Elasticsearch synchronization is dynamically calculated and adjusted to achieve the rebalance between synchronization speed and resource consumption.

$$v_t = \beta v_{t-1} + (1-\beta)\theta_t, \tag{1}$$

where θ_t is the actual response time at iteration t, v_t is the estimate to replace θ_t, which is the exponentially weighted average at iteration t, and β is the weight of v_{t-1}, which is an adjustable hyper-parameter ($0<\beta<1$). Fix to set the synchronization period as v_t.

2) Rewrite the Canal adapter component

By establishing various processing functions to satisfy the requirements in advance, which enable corresponding functions in this Batch with business requirements. For example, processing simplified, traditional Chinese

conversion function SimAndComFun, heterogeneous character function DiffCharFun, removing modal words function DropModalFun.

By passing the parameter $IS_{abaptor}$, which determines whether we add the configuration filter method to perform the secondary optimization of Canal synchronization.

If the value of $IS_{abaptor}$ is false, the filter would not be enabled; Otherwise, the filter is enabled, then the processing functions would be configured by the specific requirements.

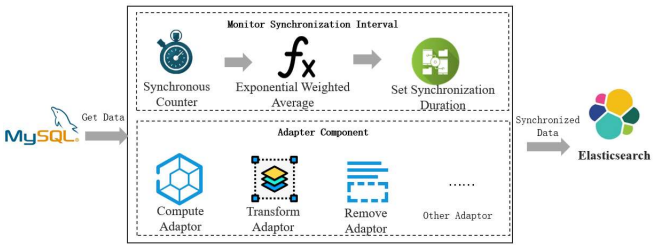

Fig. 2. Diagram of the canal synchronization algorithm optimization.

Algorithm I Canal Synchornization Algorithm

Input: G_{data} , $IS_{abaptor}$
Output: S_{data}

```
Function Sync_data(Gdata , ISabaptor) :
  connector.connect()
  connector.subscribe(".*\\..*")
  Ts=startSycTimer()
while ISabaptor :
   message = getWithoutAck(Gdata)
   batchId = message.id()
   entries = message.entries()
for entry in entries :
    db= entry.schemaName
    tb = entry.tableName
    change = entry.storeValue
    type = change.eventType();
    for ch in change :
     SimAndComFun (ch)
     DiffCharFun(ch)
     DropModalFun(ch)
     ComputeFun(ch)
     id = findId(ch)
     if type == INSERT do
      json = toInsertJson(ch,tb,db,id)
      insertES(json)
    if type == UPDATE do
     json = toUpdateJson(ch,th,db,id)
     updateES(json)
    if type == DELETE do
     deleteES(th,db,id)
   Sdata = ack(batchId)
  Ts=endSycTimer() - Ts
  RTs=IndexWeightFun(Ts)
  SetSynTime(RTs)
```

Output: S_{data}

At the end of the filter function, the cleaned synchronous data S_{data} is returned. Moreover, The critical code for the synchronization algorithm is depicted in Algorithm I, which can help researchers implement the system and make related test experiments.

D. Elasticsearch Dictionary Optimization Algorithm Design

There are structured data and unstructured data, like pictures and texts in TCM prescription data. Meanwhile, there are stop words like "hu" "xi" and "Zhi", which do not need to be involved in retrieval, expansion words composed of proper nouns and high-frequency words in business data. Therefore, in view of the above mentioned business requirements, this paper establishes a custom expansion word segmentation dictionary and stop word dictionary for Elasticsearch to build the index, thereby optimizing and addressing the existing issues. In addition, the specific optimization block diagram is shown in Fig. 3.

Let the received data be R_{msg}, IS_f is the text data, S_{msg} represents structured data, F_{msg} is text data, W_u is word segmentation word element, E_w represents expansion word segmentation, S_w is stop word segmentation, I_{data} is inverted index.

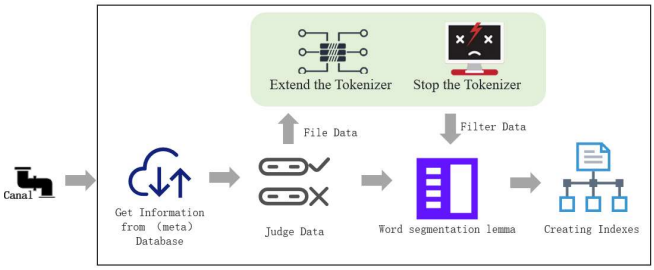

Fig. 3. Diagram of the elasticsearch dictionary optimization algorithm.

Algorithm II ElasticSearch Dict Algorithm

Input: R_{msg},IS_f
Output: I_{data}

```
function mainFunc(Rmsg, ISf):
 while ISf:
  if func == "Sw":
   file = new File("stopwords")
   tmpS = getIkSWord(file, response)
  else if func == "Ew ":
   file = new File("extwords")
   tmpE = getIkEWord(file, response)
  Rmsg = ContactFile(tmpS, tmpE)
 SWmsg =SplitWords(Rmsg)
 Idata =ConReverseIndex(SWmsg)
```

Output: I_{data}

When using Elasticsearch for full-text search, we need to index the received data R_{msg}. In this case, we also adopt the IS_f flag to determine whether the data is text. For non-text data, metadata is used to create it. With text data, F_{msg} needs to read data in turn by file operations, and then call custom word segmenting devices E_w and S_w to perform operations before word segmentation. After the above operation, the word segmentation W_u is adopted to construct the inverted index. Among them, the custom word segmentation E_w and S_w are achieved by hot update technology, thereby improving the efficiency and accuracy of search results. By updating the dictionary of the word segmentation in real time, the custom word segmentation can analyze the text in real time without restarting the application, thereby satisfying the efficient search needs of users. Furthermore, the Elasticsearch algorithm optimization code is shown in Algorithm II.

IV. Application and Test

A. Comparison of Synchronization Delay Between Canal and Logstash

The current mainstream synchronization framework is Logstash. However, there are some issues like poor timeliness, high resource consumption, insufficient data secondary processing and filtering ability. Thus, this paper proposes a solution to optimize the open source synchronization framework Canal combined with SpringBoot technology.

To verify the performance of the synchronization technology used in MPTP, a synchronization delay

experiment based on two Intel(R) Core i7 cpus and 16G RAM computer equipment is designed. Moreover, MySQL and Elasticsearch are installed to provide data support, real-time synchronization monitoring. More importantly, the other machine is configured with Canal and Logstash environments, which is connected to MySQL and Elasticsearch.

Table I shows the comparison of the delay (unit: seconds) of the two synchronization frameworks when the data are 10,000, 100,000 and 1000,000 records respectively.

TABLE I. COMPARISON OF TIME DELAY FOR DATA SYNCHRONIZATION

Test Name	*10000*	*100000*	*1000000*
Canal	2.830	10.510	155.927
Logstash	2.1	10.769	204.318

TABLE II. THE CPU USAGE WHEN THERE IS NO DATA CHANGE

Test Name	Average CPU Usage/%
Canal	0.6
Logstash	2.1

TABLE III. CPU USAGE OF MYSQL DURING DATA SYNCHORNIZATION

Test Name	Average CPU usage for MySQL/%
Canal	4.0
Logstash	21.6

TABLE IV. COMPARISON OF RETRIEVAL CAPABILITIES BETWEEN MYSQL AND ELASTICSEARCH

Test Name	Index Speed/ms
Canal	84
Logstash	4

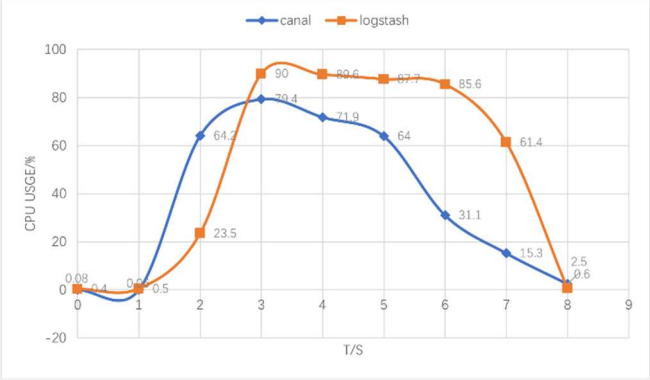

Fig. 4. The comparison of CPU usage.

Firstly, a real-time monitoring file is configured in the synchronization process of a large amount of data to track data changes, which also updates index in Elasticsearch.

Secondly, the synchronization time for Canal is calculated as the normal time interval from MySQL to Elasticsearch, which is optimized using an exponentially weighted average. Thereafter, the synchronization time of Logstash is the ideal interval from MySQL to Elasticsearch, where the current batch of data is updated, and then the new batch of data is synchronized immediately. Moreover, the synchronization delay of Logstash shown in Table I is based on ideal interval completion. Under normal circumstances, Logstash synchronization data time follows the following formula:

$$t = T^{\text{interval}} - sp + I^{data}. \quad (2)$$

In equation (2), (T^{interva}) is the Logstash time interval set by custom, *sp* is the time from the completion of the last synchronization when the data is updated to MySQL. In addition, *data* is the amount of data at this time, (I^{data}) is the ideal synchronization time corresponding to the amount of data.

Thirdly, the analysis in Table I shows that the delay of Canal is lower than that of Logstash when the amount of data is no more than 10000, which indicates that the performance of Canal is not fully demonstrated when dealing with small amount of data. However, the excellent performance advantage of Canal in dealing with large amount of data synchronization gradually manifests after the data size exceeds 100000. Specifically, the results of 1000000 show that Canal has opened a gap with Logstash.

B. CPU consumption comparison between Canal and Logstash

The CPU consumption experiment is carried out on the constraint of 100000 data, and then the proportion of CPU consumption resources in the process of synchronizing MySQL to Elasticsearch by Canal and Logstash is mainly used to illustrate the performance of the framework.

This experiment is based on the comparison of CPU resources consumed by the two frameworks in the whole process of one batch data processing, and then the results are shown in Fig. 4. Due to Canal conducting business adaptation operation and synchronization cycle monitoring, the CPU resources occupy a large proportion in the early stage of processing. After the related operation completing, then synchronization work is done, thus the CPU resources share of Canal decreases significantly compared with Logstash framework.

Actually, the data synchronization task in the system is not always carried out. When there is no data synchronization task, the average CPU share of the two framework processes is shown in Table II. It is clear that Canal consumes less server resources than Logstash.

Typically, the synchronization task needs to interact with MySQL database. Therefore, MySQL's average CPU usage may also reflect the performance of the framework. Table 3 shows the statistical average CPU occupancy of MySQL during the execution of synchronization tasks. Table III shows that Canal not only occupies relatively less resources in the synchronization process, but also it has a relatively small load on the MySQL database.

In summary, when Canal is used as the data synchronization technique, which not only consumes less system resources, but also it has higher processing efficiency. Especially for the large amount of data synchronization, the performance is more obvious, and then there is the advantage of low resource occupancy in the case of not executing tasks. It has great significance to save system resource consumption in the case of small data changes like early morning and holidays. Moreover, these features are also more suitable for current business scenarios

C. Comparison of retrieval capabilities between MySQL and Elasticsearch

MySQL database also has the ability of retrieval and query. Compared with the improved Elasticsearch, whether there is a significant difference in their retrieval speed. Considering the data pressure of MySQL, this paper makes comparative experiments on retrieval performance with 10000 data from MySQL and Elasticsearch.

Fuzzy query of MySQL database and multi-field retrieval of Elasticsearch are used in the experiment. Besides, the same fields and retrieval values are adopted for data search, and the retrieval results are obtained and compared. Based on numerous experiments, the retrieval performance of MySQL and Elasticsearch is obtained by averaging, thus the related results are shown in Table IV. Additionally, the experimental results show that the retrieval performance of optimized Elasticsearch is 21 times than that of MySQL under the condition when the current data volume is 10000. Beyond that, Elasticsearch has extensive advantages in the retrieval speed of millions of data. Therefore, it can be seen that the optimized and improved Elasticsearch can fully satisfy the search requirements of MTPF no matter what data level, which can provide more efficient full-text search capabilities that are more consistent with user semantics.

V. CONCLUSIONS

There are two main contributions in this paper. Firstly, aiming at the situation that some business data need to be processed again in the process of data synchronization, the component adapter is rewritten based on the Canal framework, then it can flexibly configure business processing functions and improve the efficiency of data synchronization. Secondly, the other one is mainly used to solve the impact of high-frequency words and useless words on the performance in the process of full-text search. By constructing a new tokenizer in the Elasticsearch engine, different weights are given to different words in the search process to solve such issues. And finally, by deploying the application in the MPTP platform, the test and application have achieved brilliant results.

In the future, we plan to address some hot issues like data conflicts, network issues and synchronization delays, which may be a critical point in the field of artificial intelligence.

REFERENCES

[1] X. Zhang, Y. Li, J. Pan, D. Chen, “Algorithm/Accelerator co-design and co-search for edge AI,” *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 69, no. 7, pp. 3064-3070, Jul. 2022.

[2] H. Farbeh, A. M. H. Monazzah, E. Aliagha and E. Cheshmikhani, “A-CACHE: Alternating cache allocation to conduct higher endurance in NVM-based caches,” *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 66, no. 7, pp. 1237-1241, Jul. 2019.

[3] J. Han, Y. Kwon, K. Byun and H.-J. Yoo, “A fault-tolerant cache system of automotive vision processor complying with ISO26262,” *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 63, no. 12, pp. 1146-1150, Dec. 2016.

[4] J. Han, Y. Kwon, Y. C. P. Cho and H.-J. Yoo, “A 1GHz fault tolerant processor with dynamic lockstep and self-recovering cache for ADAS SoC complying with ISO26262 in automotive electronics,” *2017 IEEE Asian Solid-State Circuits Conference (A-SSCC)*, Seoul, Korea (South), 2017, pp. 313-316.

[5] Giuseppe Amato, Paolo Bolettieri, Fabio Carrara, Fabrizio Falchi, and Claudio Gennaro. “Large-Scale Image Retrieval with Elasticsearch,” In The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval (SIGIR '18). Association for Computing Machinery, New York, NY, USA, 2018, pp.925–928.

[6] M. Bajer, "Building an IoT Data Hub with Elasticsearch, Logstash and Kibana," 2017 5th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW), Prague, Czech Republic, 2017, pp. 63-68.

[7] Shah, Neel, Darryl Willick, and Vijay Mago. "A framework for social media data analytics using Elasticsearch and Kibana." Wireless networks, 2022,pp.1-9.

[8] Praveen M Dhulavvagol, Vijayakumar H Bhajantri, S G Totad, “Performance Analysis of Distributed Processing System using Shard Selection Techniques on Elasticsearch,” Procedia Computer Science, vol 167, pp. 1626-1635,2020.

[9] Voit, Aleksei, et al. "Big data processing for full-text search and visualization with Elasticsearch." International journal of advanced computer science and applications, vol 8, no.12, pp.76-83, 2017.

[10] Josh Devins, Julie Tibshirani, and Jimmy Lin. 2022. “Aligning the Research and Practice of Building Search Applications: Elasticsearch and Pyserini,” In Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining (WSDM '22). Association for Computing Machinery, New York, NY, USA,2022, pp.1573–1576.

[11] D. Bhatnagar, R. J. SubaLakshmi and C. Vanmathi, "Twitter Sentiment Analysis Using Elasticsearch, LOGSTASH And KIBANA," 2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE), Vellore, India, 2020, pp. 1-5.

[12] L. Han and L. Zhu, "Design and Implementation of Elasticsearch for Media Data," 2020 International Conference on Computer Engineering and Application (ICCEA), Guangzhou, China, 2020, pp. 137-140.

[13] Zmaranda, Doina R., et al. "An analysis of the performance and configuration features of MySQL document store and elasticsearch as an alternative backend in a data replication solution." Applied Sciences 11.24 (2021): 11590.

[14] Kim, Mi, and Dosung Kim. "A Suggestion on the LDA-Based Topic Modeling Technique Based on ElasticSearch for Indexing Academic Research Results." Applied Sciences 12.6 (2022): 3118.

[15] F. Ahmed, U. Jahangir, H. Rahim, K. Ali and D. -e. -S. Agha, "Centralized Log Management Using Elasticsearch, Logstash and Kibana," 2020 International Conference on Information Science and Communication Technology (ICISCT), Karachi, Pakistan, 2020, pp. 1-7.

[16] L. Zhou, B. Lu, S. Zhang, L. Qi, “Data cache optimization model based on hbase and redis,” in *Proceedings of the 3rd International Conference on Data Science and Information Technology*, Xiamen, China, 2020, pp. 31-35.

[17] Q. Liu, H. Yuan, “A high performance memory key-value database based on redis,” *Journal of Computers*, vol. 14, no. 3, pp. 170-183, Feb. 2019.

[18] V.-A. Zamfir, M. Carabas, C. Carabas and N. Tapus, “Systems monitoring and big data analysis using the Elasticsearch system,” *2019 22nd International Conference on Control Systems and Computer Science (CSCS)*, Bucharest, Romania, 2019, pp. 188-193.

[19] B. C. Ghosh, S. K. Addya, N. B. Somy, S. B. Nath, S. Chakraborty and S. K. Ghosh, “Caching techniques to improve latency in serverless architectures,” *2020 International Conference on COMmunication Systems & NETworkS (COMSNETS)*, Bengaluru, India, 2020, pp. 666-669.

[20] Y. Li, Y. Jiang. Y, J. Gu, M. Lu, M. Yu, “A cloud-based framework for large-scale log mining through apache spark and Elasticsearch,” *Applied Sciences*, vol. 9, no. 6, pp. 1114, Mar. 2019.

[21] S.-C. Kim, H.-D. Jang, “A study on system and application performance monitoring system using mass processing engine (Elasticsearch),” *Journal of Digital Convergence*, vol. 17, no. 9, pp. 147-152, 2019.