An Optimization Strategy of Shard on Elasticsearch

Zhanglong Wang[1, 2, a], Yang Pi[1, 2, b]

[1]School of Computer Science and Technology, Chongqing University of Posts and Telecommunications, Chongqing, 400065, China

[2]Chongqing Engineering Research Center of Mobile Internet Data Application, Chongqing, 400065, China

[a]1129279512@qq.com, [b]1056864298@qq.com

Keywords: ElasticSearch, Index, Shard, Load balancing, Mathematical modeling, Linear weighting method

Abstract: With the development of big data application, there are more demands on big data storage and retrieval. Thus, Elasticsearch, a distributed full-text search engine, has appeared, which can well meet these demands. However, Elasticsearch has the following disadvantages: First, shard settings are based on user experience and may degrade retrieval performance due to human factors. Second, the factors considered in the distribution strategy of shards are incomplete. And the last, without effective processing of concurrent access to hot index data, the average performance of each node in the Elasticsearch cluster varies greatly. In this paper, we propose a shard optimization strategy of Elasticsearch, through data and performance analysis, to obtain reasonable shard settings. After that, the shards are placed in nodes with better performance which are evaluated by the linear weighting method. Then, the optimized load balancing strategy will migrate hot shards caused by hot data to make the cluster load balanced. The experimental results show that the proposed shard optimization strategy can achieve better index retrieval performance and better cluster load.

1. Introduction

With the explosive growth of Internet data, the Internet industry has put forward higher requirements on the storage and management technology of big data [1]. Therefore, Elasticsearch, an open source distributed full-text search engine, shows its ability to perform full-text retrieval of big data within milliseconds [2]. This causes Elasticsearch to be widely applied in the industry. However, Elasticsearch still has its performance problem of shard strategy to be optimized in practical applications. For example, users set the quantity of shard according to their experience, and improper setting may lead to index performance degradation. In addition, the existing placement strategy of shard mainly considers the principle of dispersion, does not consider the performance of nodes. Moreover, when Elasticsearch cluster has large concurrent access to hot data, it may cause some nodes to be overloaded, it brings bad impact to retrieval performance. Therefore, in order to further improve Elasticsearch's performance, it is of great significance to research Elasticsearch's shard mechanism.

To solve the above-mentioned problems, this paper proposes a shard optimization of Elasticsearch. The content includes: According to the performance of nodes and the estimation data size of index, the quantitative model of shard is constructed, and the reasonable quantity result of shard was calculated with the model; According to the load performance of nodes, we optimized the placement strategy of shard, so that nodes with better performance can place shard as a priority. And then, an optimized load balancing strategy is proposed based on the load performance of nodes, so that it can load balancing the cluster by migrating the hot shard in high-load nodes.

In order to achieve the above mentioned goals, we studied and analyzed Elasticsearch's existing mechanism of index-shard by reading relevant literature on Elasticsearch. Therefore, the optimization method and theoretical basis for solving the above problems are found out. By using mathematical modeling methods, the quantitative model of shard is established. After referring to other literatures on the evaluation of machine node performance, the linear weighting method was

used to evaluate the nodes performance of cluster. And then, according to the evaluation results of node performance, we select nodes with better performance for placing shard, and the idle nodes which the hot shards will be migrate to. As can be seen from the experimental results, the optimization method proposed in this paper reduces the query delay of Elasticsearch index, and improves index retrieval performance, and makes the average load of the cluster more balanced.

In this paper, we focus on Elasticsearch's index shard mechanism research to improve Elasticsearch performance. The contributions of this paper are summarized as follows:

(1) To solve the problem that there is no reference for users to set the quantity of shards, this paper establishes the quantitative model of shard according to the performance factors of cluster nodes and the amount of index's business data estimated by indices, and calculates the reasonable quantity of shard with this model, to ensure index performance and improve cluster stability.

(2) In order to make full use of nodes, we use the linear weighting method to evaluate the performance of the nodes. According to the performance evaluation results, the nodes with better performance are selected for placing shards.

(3) In order to solve the problem like that when there is large concurrent access to hot data, there may be some nodes with too high load. This paper adopts the method of migrating hot shard on high-load nodes to idle nodes to achieve load balancing of cluster nodes.

The organizational structure of this paper is as follows: Section 2 introduces Elasticsearch's relevant research and introduces the thinking and methods of this paper. In section 3, there is an analysis of Elasticsearch's existing problems. Then there is a detailed introduction of shard optimization strategy of Elasticsearch proposed in this paper. Section 4 introduces the experimental design, shows the experimental results and conducts the experimental analysis. In section 5, the research of this paper is summarized and the future research is introduced.

2. Related work

Elasticsearch has gained wide attention due to its features of distributed storage, inverted index and data shard. The current research mainly focuses on its index storage, index structure, and index comprehensive application.

In terms of index storage: Yicheng Zheng, Feng Deng et al. proposed a platform for virtualization combined with Elasticsearch [3]. Based on this platform, they explore the feasibility and advancement for storing and searching spatio-temporal data. And the time period index is imported as spatio-temporal data records both time and location; Dequan Chen, Yi Chen, Brian N et al. proposed optimization of daily medical data storage cluster topology based on HDFS and Elasticsearch [4]. This method establishes two big data platforms with the same Hadoop environment. Each cluster contains one Elasticsearch cluster and one storm topology instance to achieve real-time or near-real-time storage, analysis and retrieval; S Gupta, R Rani have done storage comparative research on Elasticsearch and CouchDB document-oriented database, in this paper, Elasticsearch and CouchDB's performance on image data set is analyzed, proving that Elasticsearch performance is much better than CouchDB in the retrieval operation process [5].

In terms of index structure: Cun Mu, Jun Zhao, Guang Yang proposed a novel and exciting visual search solution, we can utilize Elasticsearch to efficiently retrieve similar images based on similarities within encoded sting tokens [6]; Xuemeng Li, Yongyi Wang et al. proposed Elasticsearch-based retrieval method design and implementation. This method optimizes the index data structure and optimizes the retrieval strategy. They also adopt the corresponding compression algorithm to ensure compression efficiency and improve retrieval performance [7].

In terms of index comprehensive application: Fadi Mohsen, Hamed Abdelhaq, Halil Bisgin were proposed and evaluated a new security-centric ranking algorithm built on top of the Elasticsearch engine to assist users evade installing intrusive apps [8]; Giuseppe Amato, Paolo Bolettieri, Fabio Carrara proposed to transform CNN features into textual representations and index them with the well-known full-text retrieval engine Elasticsearch [9].

The above research work finded and solved some limitations or performance problems of Elasticsearch. Studying Elasticsearch's index shard mechanism is an effective way to further

optimize it. This paper proposes a shard optimization strategy of Elasticsearch aims at its index shard performance problem. In this strategy, the quantity of shard is reasonably set, the placement of shard is based on node performance, and the load balancing problem caused by hot shard was also solved.

3. Description of Problems

This paper studies and analyzes the working principle of existing Elasticsearch index-shard mechanism, On this basis, optimizes Elasticsearch's shard setup, the placement strategy of shard and the load balancing problem of cluster caused by high concurrent access to its hot indices. In this paper, the method mainly considers to reduce the error caused by human experience. The linear weighting method was used to evaluate the performance of cluster nodes. According to the result, select nodes with better performance to place shards, and the hot shards will be migrated to perform cluster load balancing.

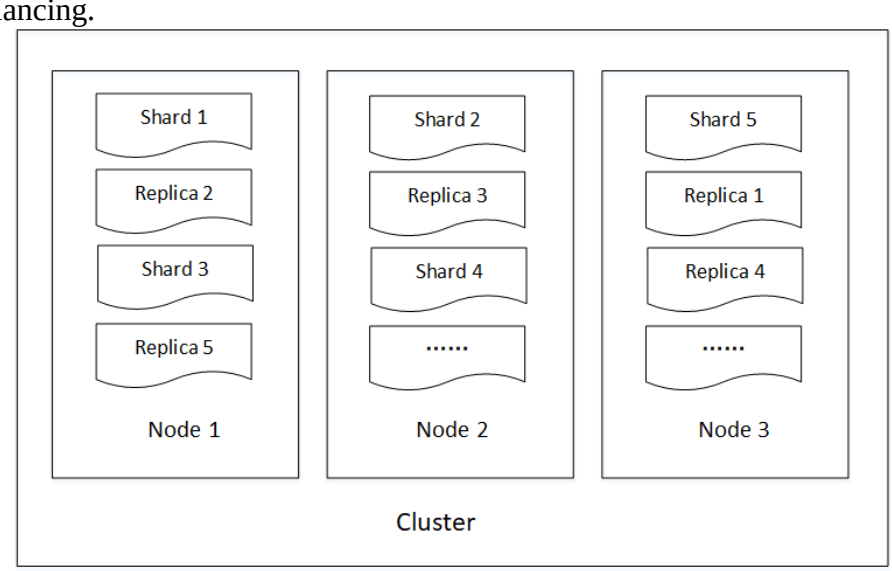

Figure 1. Index shard storage mechanism

The Fig.1 shows the structure of index-shard storage mechanism. There is an Elasticsearch cluster with several nodes, the primary shard of the index and the corresponding replica shard are stored in the cluster node. The shards of index are evenly distributed across the nodes of the cluster, and primary and corresponding replica shards are not stored on the same node. Elasticsearch shard mechanism has the following problems:

(1) The quantity of shard settings: The shard setup of Elasticsearch has no strict basis. Users can set this by default or by experience. Under the condition of fixed quantity of cluster nodes and index data, excessive shards will lead to excessive performance and resource waste. On the contrary, too few shards results in poor index performance and poor clustering performance. Therefore, the quantity of shards Settings are necessary to accommodate the size of index business data volume and node performance.

(2) The placement of shards: The shard of Elasticsearch usually adopts primary shards and copy shards modes, and the storage of shards can ensure as far as possible that the primary shards are stored separately in the cluster. Under this principle, and then to check whether the node size of disk space to limit threshold, index in a single node to create the maximum number of shard and the filter conditions set by users and so on. However, without considering the performance of the nodes in the cluster, some nodes will be busy or idle, and the performance of the cluster cannot be given full play.

(3) Load balancing problem: When there is a large amount of access to hot index data in the cluster and the shards of these indices are concentrated on some nodes, it will cause high load on these nodes. Furthermore, since the quantity of shard can be set by the user, it is easier to aggravate the cluster load imbalance. Therefore, it is necessary to study the dynamic migration strategy of hot shards to balance the load of cluster nodes.

4. Model and strategies

This paper aims at Elasticsearch's reasonable setting and placement of shards as well as its load problem with hot shards, the quantity of shard is calculated according to the performance of cluster nodes and the estimated amount of data of index, such as formula (1), (2), after fragmentation results, according to the distribution of cluster nodes shard performance optimization, finally, dynamic load balancing optimization is carried out for hot shards. Through the above optimization, the goal of making full use of cluster node performance, node load balancing and improving cluster stability is achieved.

4.1 A Quantitative Model of Shard

In order to guarantee the performance of index, it is necessary to determine the appropriate quantity of shard at first. In this paper, based on the performance of cluster nodes and the estimated amount of index data, by means of mathematical modeling [10], the quantitative model of shard is established to obtain a reasonable quantity of shard. The performance factor of nodes is to verify whether the nodes meet the performance requirements of creating shard, furthermore the amount of index data is an important factor affecting the performance of shard. The details are as follows:

The cluster node formation list is named nodeList, the performance of the nodes in nodeList is verified, and the results are saved in the array, which is named nodeArr. If the verification passes, the corresponding nodeArr element is set to 1, otherwise it is set to 0. As the volume of index business data is an important factor affecting shard performance, furthermore, Elasticsearch officially recommends a single slice with a data storage size of 25GB, and certain extensibility in mind. Formula 1 is used to calculate the quantity of shard:

$$shardNum = \left\lfloor \left(e * \frac{D}{25GB} + k * \sum_{i=1}^{n} nodeArr_i \right) * (1 + \theta) \right\rfloor \quad (1)$$

Where D is the estimated amount of index business data, and θ is the extension coefficient, and e and k are the weight coefficients.

If the above result, shardNum, exceeds the quantity of nodes currently available, then the quantity of shard should be set based on the quantity of currently available nodes. That is, formula 2 is used to calculate the quantity of shard:

$$shardNum = \sum_{i=1}^{n} nodeArr_i \quad (2)$$

Based on the result shardNum calculated by the above methods, index is created and configured. That is, number_of_shards: shardNum. Meanwhile we adjust the index parameter, total_shards_per_node: x, so that the shard placement strategy process shards. x is calculated by formula 3:

$$x = \left\lceil \frac{shardNum}{\sum_{i=1}^{n} nodeArr_i} \right\rceil \quad (3)$$

4.2 The Placement Strategy of Shard

In this paper, the performance factors of the nodes are considered, and the performance of each node is evaluated by using the linear weighting method [11]. Then, the nodes with better performance are selected for placing shard, which can not only ensure the dispersibility of shards placement and verification rules, but also consider the performance of the cluster nodes.

(1) The performance evaluation of cluster nodes:

The performance evaluation of nodes that meet the requirements in the quantity model of shard is carried out by using the linear weighting method, that is, formula 4:

$$Q_i = a * LA_i + s * SN_i + b * DsR_i \quad (4)$$

Qi represents the performance evaluation result of I node, and the smaller the value is, the better the performance of I node. LAi, SNi and DsRi represent the average load, the quantity of shards and the disk utilization rate of I node respectively. And a, s and b are weight coefficients (For the actual cluster environment, the weight coefficient can be obtained by statistical analysis or expert

consultation).

(2) The placement of shard:

To place the shards with shardNum number in algorithm 1. In the process of shard placement, the nodes with better performance are selected first for placing shard. Meanwhile, check the limits of the parameter total_shard_per_node during this process, and also consider the number of candidate nodes versus shardNum.

The Strategy finally determines whether there are any remaining shards that have not been created. If there is, create these shards in the cluster, do not place them but mark their state as unassigned. The unassigned tag is a mark on unassigned shards in the cluster of Elasticsearch.

4.3 An Optimization Load Balancing Strategy by Dynamically Migrating Hot Shards

In order to solve the problem of load imbalance of cluster nodes caused by hot shards, in this paper, the quantity of hot shard in cluster nodes is detected periodically. When the quantity of hot shards in nodes reaches the threshold, these nodes load will be relatively high. We use the method of hot data migration to reduce the pressure of high-load nodes. Here we migrate the hot shard among these nodes. We use the linear weighting method to evaluate the load of cluster nodes, and select idle nodes as the migration targets of hot shards, so as to share the load on busy nodes and achieve the goal of cluster load balancing.

(1) Hot shards statistics: We periodically monitor the retrieval frequency of indices by using the kibana tool [12]. If the threshold α is reached, then, the index is set to the hot index. The shards to which the hot indices belong are all hot shards. Then, determine whether the number of hot shards in each node reaches γ. If so, add this type of node to the list to be adjusted.

(2) The load evaluation of nodes: By monitoring and obtaining the I/O utilization rate, network bandwidth, CPU utilization rate, RAM utilization rate and other parameters of the cluster nodes, the load evaluation result Qi of each node was calculated by using the linear weighting method through formula (5):

$$E_i = o * IO_i + d * MBPS_i + c * CPU_i + r * RAM_i \quad (5)$$

The Ei value represents the load evaluation value of node i, the greater the value shows that the higher the load of the node i. The IOi, MBPSi, CPUi, RAMi, respectively represent the I/O utilization, network bandwidth usage, CPU usage, memory usage. The o, c, d, r, respectively represent the weight coefficient of the four parameters (according to the actual cluster environment, can use statistical analysis or experts for its weight coefficient), and o + d + c + r = 1.

(3) Dynamically Migrating Hot Shards:

First, determine whether all the values of E reach the threshold β (this indicates that all the cluster nodes are busy). If so, cancel the load adjustment and wait for the detection of the next cycle. Otherwise, traverse the list of pending load adjustments, successively move the hot shard in node i into node j with smaller Ej values. After shard migration is completed and the performance of node i and j is stable, the values of node Ei and Ej are calculated again, and the number of hot shards in node i and j is updated. Similarly, the list of pending load adjustment is updated. Repeat this process until the load list is empty or the load adjustment repeats reach the upper limit.

Before shard load adjustment, specify the shard migration repetitions m = ⌊ n / 2 ⌋, where n represents the number of cluster nodes. We set the number of repetitions m to avoid getting stuck in an infinite adjustment.

5. Experiments

The Elasticsearch cluster in experiments has 10 nodes and up to 270GB of data needs to be stored. The data set is: the taxi driving location record of New York in 2013 (15.3GB). Table 1 shows the experimental environment.

Table.1. The description of the experiment environment

NO of CPUs	16
CPU frequency	2.40GHz
Memory	40GB
Hard Disk Size	400GB
NO of ES servers	10
Network Bandwidth	100M
Operating System	Ubuntu 16

(1) The test of shard quantity between the shard quantitative model and native model:

In this paper, the factors of node performance and the estimated amount of index data are taken into account, and a quantitative model of shard is established. And the estimated amount of index data is an important factor affecting the calculation results of this model. Therefore, this paper firstly compares Elasticsearch's native default shard number results with the quantitative model of shard in the case of different index data volumes. According to the experimental results in Fig.2, we can get an intuitive comparison of the results. With the increase of index data volume, the shard number obtained by the shard quantitative model also increases linearly, and it is not just a multiple of 25GB, but also takes into account the index extensibility and the utilization rate of cluster nodes. By default, any data size is fixed by the number of shard, which will cause performance waste for small data volume and performance degradation for large data volume.

(2) The test of query delay between the shard quantitative model and native model:

In this paper, after using the quantitative model of shard to confirm the number of shard, since the query delay of the index is an important criterion to measure its performance. Therefore, this experiment was designed to compare the query delay of index generated by the native and the shard quantitative model under the circumstance of different index data volumes, so as to verify whether the shard quantitative model in this paper has performance improvement. It can be seen from the experimental results of Fig.3 that under the same index data volume, the query delay of index generated by the optimized shard model is generally 20~60ms lower than that generated by the native model. It shows that the query performance of the indices generated by the quantitative model of shard is improved.

(3) The test of the cluster’s variance of average load between the optimized shard placement strategy and native default:

In this paper, several performance factors are used to calculate the performance evaluation values of cluster nodes by linear weighting method. In this set of performance evaluation results, the optimal node is selected to place shard in turn. In this experiment, a series of indices were generated in the cluster by using the native default method and the optimized placement strategy of shard respectively. By comparing the variance of the average load of the cluster with the same number of indices, to verify that the average load of the cluster is more balanced under the optimized shard placement strategy. It can be concluded from Fig.4 that under the optimized shard placement strategy, the variance results of the average load of the cluster nodes are generally smaller than those obtained by the native default placement method, which indicates that the optimized shards placement strategy makes the load of the cluster nodes more balanced. As can be seen from the above figure, under the native default placement method, there is a sudden drop under 8 indices. This is because when the indices in the cluster reaches a certain number, the gap of the quantity of shard in each node will become smaller, that is, all nodes are fully utilized, so the average load variance of the cluster nodes decreases. This tipping point occurs depending on the number of nodes in the cluster and the performance.

(4) The test of the cluster’s variance of average load between the optimized load balancing strategy and native default:

This paper uses the monitoring tool kibana to monitor the index retrieval frequency in the cluster, and determines the hot shards according to the shard belongs to hot index has the same load, and the number of hot shard in all nodes of the cluster is counted. This experiment by constructing different number of hot index, compared to native default load balancing strategy and dynamic migration hot

shard of the load balancing strategy, the indices in the same cluster load average variance, to verify the proposed dynamic migration hot shard of load balancing strategy can make the cluster load more balanced. As can be seen from the Fig.5, With Elasticsearch's native default strategy, the variance of the average performance of cluster nodes is fast increasing as the number of hot indices increases. However, when the number of hot indices increases to a certain number, as in this experiment, when the number of indexes is 8, the variance result growth trend of the average load of cluster nodes slows down obviously. This is because as the hot indices spread across the cluster, the average load on each node becomes more and more similar. By contrast, the load balancing strategy of dynamically migrating hot shards proposed in this paper makes the variance of the cluster nodes average load more stable and smaller. This shows that the load balancing strategy proposed in this paper can achieve better load balancing and better effect.

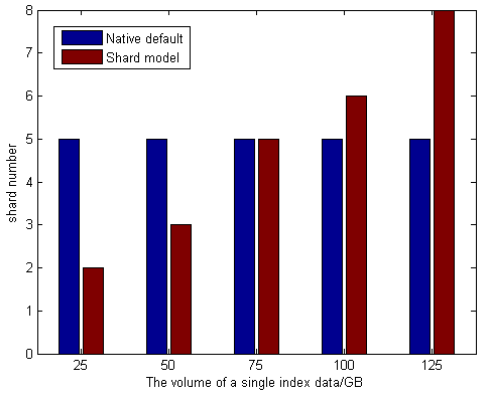

Figure 2. The shard number result of model

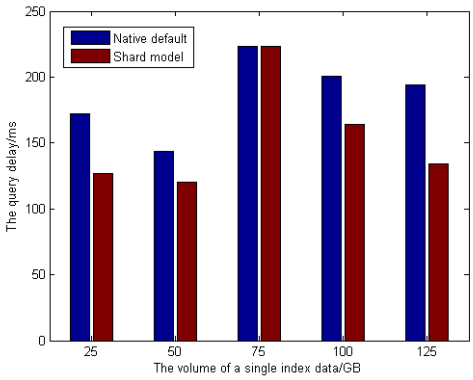

Figure 3. The query delay

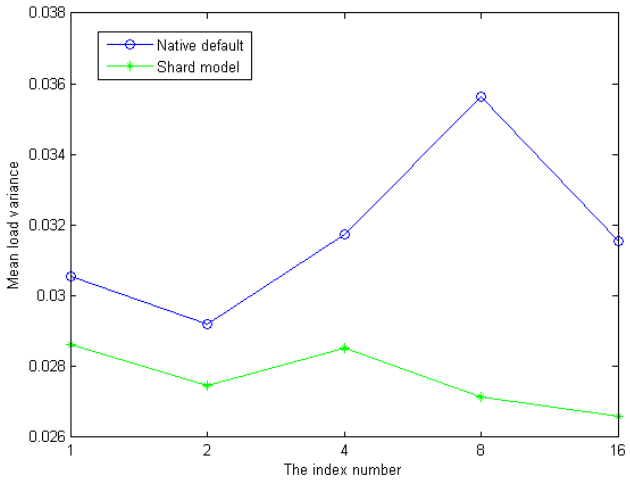

Figure 4. The placement strategy of shard

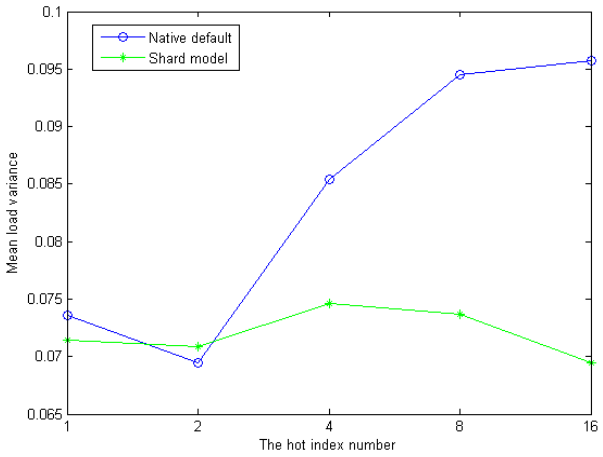

Figure 5. The optimized load balancing strategy

6. Conclusion

This paper aims at improving Elasticsearch's performance of index shard. First, Elasticsearch sets up a computational model to obtain the number of shard by comprehensively considering the performance factors of cluster nodes and the estimated amount of index data. Then, the performance evaluation results of cluster nodes are calculated by linear weighting method which considered several performance factors, and the nodes with better performance are selected for placing shards. Finally, we periodically detect the quantity of hot shards in the cluster nodes by using monitoring tools. For nodes whose quantity of hot shards reaches the threshold, we migrate the hot shards of those nodes to the idle nodes which we selected by performance evaluation results. From the experimental results, the optimization strategy proposed in this paper reduces Elasticsearch's query delay of index, improves index retrieval performance, and makes the average load of the cluster more balanced and improves its stability.

In the future, we will improve the performance evaluation algorithm of cluster nodes to make the results more accurate and reliable. We will also find a better optimized shard strategy to avoid migrating shards because of the performance overhead associated with doing so.

References

[1] Qureshi S R, Gupta A. Towards efficient Big Data and data analytics: A review [C] // IT in Business, Industry and Government (CSIBIG). New Jersey, USA: IEEE, 2014: 1-2.

[2] Singh P K, Suryawanshi A, Gupta S, et al. Elasticsearch and Carrot 2 -Based Log Analytics and Management [J]. 2016: 1-2.

[3] Zheng Y , Deng F , Zhu Q , et al. Cloud storage and search for mass spatio-temporal data through Proxmox VE and Elasticsearch cluster [C] // IEEE International Conference on Cloud Computing & Intelligence Systems. IEEE, 2015: 1-2.

[4] Chen D, Chen Y, Brownlow B N, et al. Real-Time or Near Real-Time Persisting Daily Healthcare Data into HDFS and ElasticSearch Index inside a Big Data Platform [J]. IEEE Transactions on Industrial Informatics, 2016, PP (99): 1-3.

[5] Gupta S, Rani R. A comparative study of elasticsearch and CouchDB document oriented databases [C] // International Conference on Inventive Computation Technologies. IEEE, 2017:1-4.

[6] Mu C, Zhao J, Yang G, et al. Towards Practical Visual Search Engine within Elasticsearch [J]. 2018: 1-2.

[7] Li X M, Wang Y Y. Design and Implementation of an Indexing Method Based on Fields for Elasticsearch [C] // International Conference on Instrumentation & Measurement. IEEE, 2015:

626-630.

[8] Fadi Mohsen, Hamed Abdelhaq, Halil Bisgin, Andrew Jolly, Michael Szczepanski: Countering Intrusiveness Using New Security-Centric Ranking Algorithm Built on Top of Elasticsearch. TrustCom/BigDataSE 2018: 1048-1057.

[9] Giuseppe Amato, Paolo Bolettieri, Fabio Carrara, Fabrizio Falchi, Claudio Gennaro: Large-Scale Image Retrieval with Elasticsearch. SIGIR 2018: 925-928.

[10] Tinsley H E A, Brown S D. Handbook of applied multivariate statistics and mathematical modeling [M]. 2000: 24-38.

[11] Carstoiu D, Lepadatu E, Gaspar M, et al. Hbase - non SQL Database, Performances Evaluation.[J]. International Journal of Advancements in Computing Technology, 2010, 2 (5): 42-52.

[12] Bajer M. Building an IoT Data Hub with Elasticsearch, Logstash and Kibana [C] // IEEE, International Conference on Future Internet of Things and Cloud Workshops. IEEE, 2017: 2-6.